LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

Spring 2021

Student Name: Aadarsha Muni Shakya

Group: N1

London Met ID: 20049438

College ID: NP01NT4S210023

Assignment Due Date: 20th August 2021

Assignment Submission Date: 20th August 2021

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

Table of Contents

1. Introduction 1
2. Class Diagram 2
3. Pseudocode 4
4. Description 11
   - INGCollege() 11
   - main(String []args) 12
5. Test 12
   - 5.1 Test 1- Compile Using Command prompt 12
   - 5.2 Test 2 13
     - 5.2.1 Test 2.1 Add Course for Academic course 13
     - 5.2.2 Test 2.2 Add Course for NonAcademic course 15
     - 5.2.3 Test 2.3 Register academic course 17
     - 5.2.4 Test 2.4 Register non-academic course 19
     - 5.2.5 Test 2.5 Remove non-academic course 21
   - 5.3 Test 3 23
     - 5.3.1 Test 3.1 Trying to add duplicate CourseID 23
     - 5.3.2 Test 3.2 Trying to register already registered course 24
     - 5.3.3 Test 3.3 Trying to remove the non-academic course which is already removed. 26
6. Error 27
   - 6.1 Syntax error 27
   - 6.2 Logical error 29
   - 6.3 Run time error 30
   - 6.4 Semantic Error 32
7. Conclusion 33
8. Appendix1 34
9. Appendix2 66
10. References 79

List of Figures

Figure 1: Class diagram of classes in BlueJ 2
Figure 2: Class Diagram 3
Figure 3:Codes used in Command Prompt 12
Figure 4: Opening GUI using Command Prompt 13
Figure 5: Assigning value in AcademicCourse 14
Figure 6: Dialog box when added inAcademicCourse 14
Figure 7: Display when only Add button is clicked in AcademicCourse 15
Figure 8:Assigning value in NonAcademicCourse 16
Figure 9:Dialog box when added in NonAcademicCourse 16
Figure 10:Display when only Add button is clicked in NonAcademicCourse 17
Figure 11: Assigning value in AcademicCourse 18
Figure 12: Dialog box when registered in AcademicCourse 18
Figure 13: Display when Add and Register button is clicked in AcademicCourse 19
Figure 14: Assigning value in NonAcademicCourse 20
Figure 15: Dialog box when registered in NonAcademicCourse 20
Figure 16: Display when Add and Register button is clicked in NonAcademicCourse .. 21
Figure 17: Assigned value in NonAcademicCourse 22
Figure 18: Dialog box when Removed in NonAcademicCourse 22
Figure 19: Display when Add and Remove button is clicked in NonAcademicCourse .. 23
Figure 20: Assigning duplicat CourseID in AcademicCourse 24
Figure 21:Assigning duplicat CourseID in NonAcademicCourse 24
Figure 22: Registring already registered data in AcademicCourse 25
Figure 23: Registring already registered data in NonAcademicCourse 25
Figure 24: Removing non-acadamic course which is alerady removed 26
Figure 25: Syntax error 27
Figure 26: Solved Syntax error 28
Figure 27: Logical Error 29
Figure 28: Solved Logical Error 29
Figure 29: Run time error 30
Figure 30: Run time Error (BlueJ Terminal Error) 30
Figure 31: Solved Run time Error 31
Figure 32: Semantic Error 32
Figure 33: Solved Semantic Error 32

List of Tables

Table 1: Test 1 12
Table 2: Test 2.1 13
Table 3: Test 2.2 15
Table 4:Test 2.3 17
Table 5:Test 2.4 19
Table 6:Test 2.5 21
Table 7: Test 3.1 23
Table 8: Test 3.2 24
Table 9: Test 3.3 26

1. Introduction

Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centres, game consoles, scientific supercomputers, cell phones, etc (Guru99, 2021). In this module a program called BlueJ was used. This program allows you to develop java programs quickly and easily. This program is simple, designed for teaching, interactive, portable, mature and innovative. (BlueJ, n.d.).

In this coursework tools like BlueJ, Draw.io and Word were used. BlueJ was mainly used to develop a working graphical user interface (GUI). And the remaining tools like Draw.io, MS - Word were used for the report of this coursework.

For this module, everyone was given a task of creating a working GUI.The GUI is created in a class called INGCollege winch is linked with the classes of previous coursework. Namely, Course class AcademicCourse class and NonAcademicCourse class. Course class is linked with INGCollege to create array lists. Similarly, AcademicCourse and NonAcademicCourse is linked with INGCollege to create an object of AcademicCourse and NonAcademicCourse to access the methods of those classes.

The GUI of INGCollege is allows the users to assign values to the GUI. If the assigned is valid tasks like add, register and remove can be performed. Else, error dialog box is displayed. Additionally, the users can clear the text fields and display the contents which are successfully added and registered.

2. Class Diagram

Class diagrams are the main building block in object-oriented modelling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them (Nishadha, 2020). The main purpose of class diagram is to create a rough sketch of the program which will help in visually representing the program.

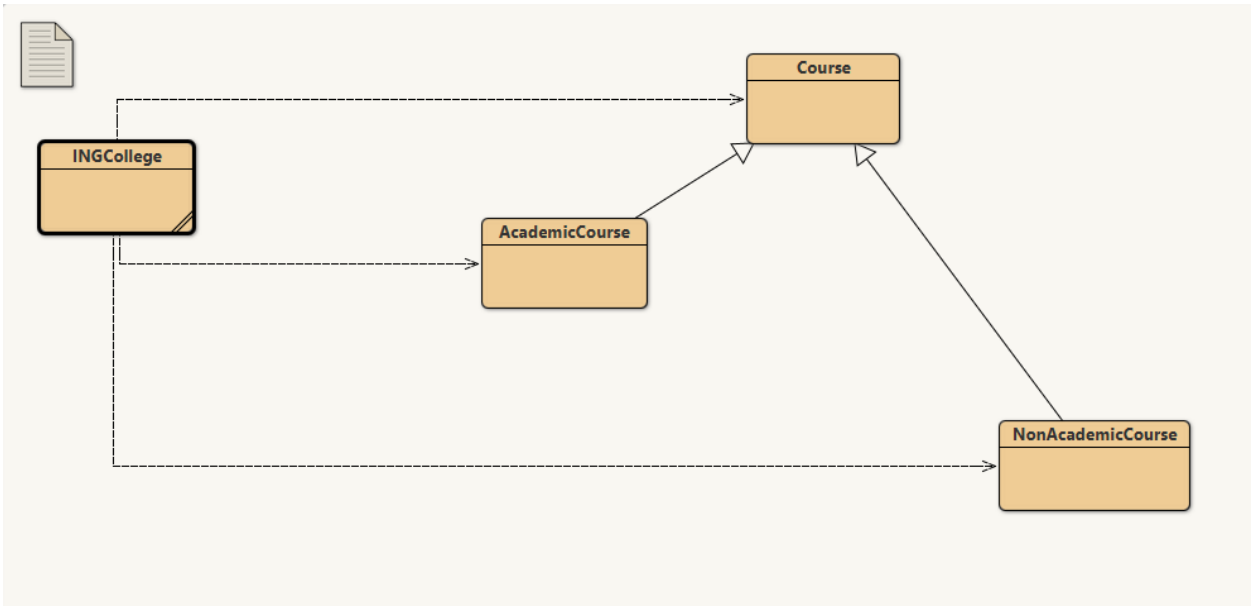

Figure 1: Class diagram of classes in BlueJ

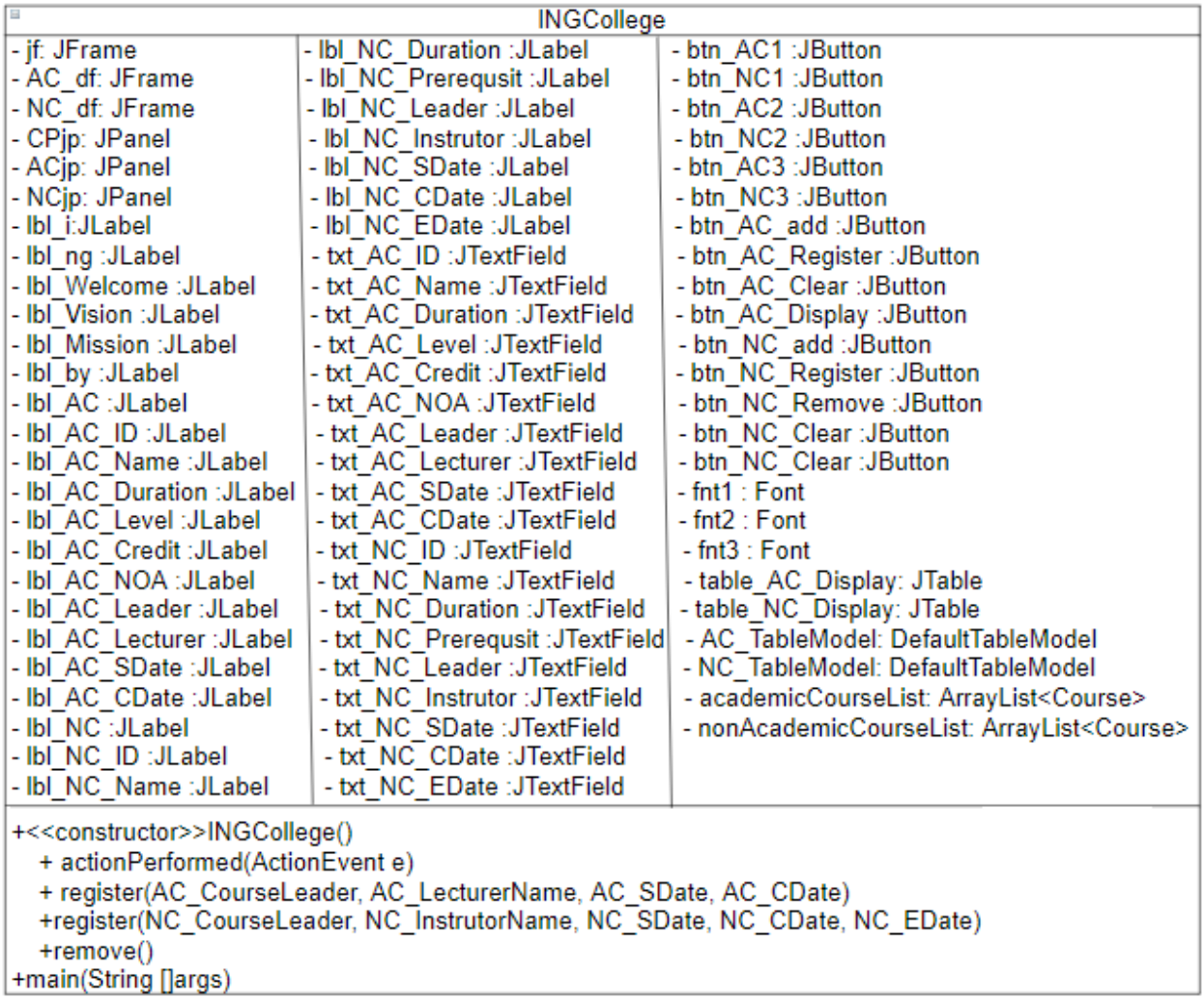

Figure 2: Class Diagram

3. Pseudocode

Pseudo code is a term which is often used in programming and algorithm-based fields. It is a methodology that allows the programmer to represent the implementation of an algorithm. Simply, we can say that it's the cooked-up representation of an algorithm. Often at times, algorithms are represented with the help of pseudo codes as they can be interpreted by programmers no matter what their programming background or knowledge is. Pseudo code, as the name suggests, is a false code or a representation of code which can be understood by even a layman with some school level programming knowledge (Theprogrammedwords, 2021).

```
IMPORT packages

CREATE INGCollege class

    DEFINE Frame

        DECLARE PRIVATE jf, AC_df, NC_df

    DEFINE Panel

        DECLARE PRIVATE CPjp, ACjp, NCjp

    DEFINE Label

        DECLARE PRIVATE lbl_i, lbl_ng, lbl_Welcome, lbl_Vision,
        lbl_Mission,lbl_by, lbl_AC, lbl_AC_ID,
        lbl_AC_Name,lbl_AC_Duration, lbl_AC_Level,lbl_AC_Credit,
        lbl_AC_NOA, lbl_AC_Leader, lbl_AC_Lecturer, lbl_AC_SDate,
        lbl_AC_CDate, lbl_NC, lbl_NC_ID, lbl_NC_Name,
        lbl_NC_Duration, lbl_NC_Prerequsit, lbl_NC_Leader,
        lbl_NC_Instrutor,lbl_NC_SDate, lbl_NC_CDate, lbl_NC_EDate

    DEFINE Text Field

        DECLARE PRIVATE txt_AC_ID, txt_AC_Name, txt_AC_Duration,
        txt_AC_Level, txt_AC_Credit, txt_AC_NOA, txt_AC_Leader,
```

txt_AC_Lecturer, txt_AC_SDate, txt_AC_CDate, txt_NC_ID, txt_NC_Name, txt_NC_Duration, txt_NC_Prerequsit, txt_NC_Leader, txt_NC_Instrutor,txt_NC_SDate, txt_NC_CDate, txt_NC_EDate

DEFINE Button

DECLARE PRIVATE btn_AC1, btn_NC1, btn_AC2, btn_NC2, btn_AC3, btn_NC3, btn_AC_add, btn_AC_Register, btn_AC_Clear, btn_AC_Display, btn_NC_add, btn_NC_Register, btn_NC_Remove, btn_NC_Clear, btn_NC_Display

DEFINE Font

DECLARE PRIVATE fnt1, fnt2, fnt3

DEFINE Table

DECLARE PRIVATE table_AC_Display, table_NC_Display

DEFINE DefaultTableModel

DECLARE PRIVATE AC_TableModel, NC_TableModel

DEFINE ArrayList of Course type

DECLARE PRIVATE nonAcademicCourseList, academicCourseList

CREATE constructor method for INGCollege class

SET Bounds for jf

SET Layout null in jf

SET Visible true jf

SET Bounds for CPjp

SET Layout null in CPjp

SET Visible true CPjp

SET Bounds for lbl_i, lbl_ng, lbl_Welcome, lbl_Vision, lbl_Mission,lbl_by

ADD lbl_i, lbl_ng, lbl_Welcome, lbl_Vision, lbl_Mission,lbl_by in CPjp

SET Bounds for btn_AC3, btn_NC3

ADD btn_AC3, btn_NC3 in CPjp

SET Bounds for ACjp

SET Layout null in ACjp

SET Visible false ACjp

SET Bounds for lbl_AC, lbl_AC_ID, lbl_AC_Name,lbl_AC_Duration, lbl_AC_Level,lbl_AC_Credit, lbl_AC_NOA, lbl_AC_Leader, lbl_AC_Lecturer, lbl_AC_SDate, lbl_AC_CDate

ADD lbl_AC, lbl_AC_ID, lbl_AC_Name,lbl_AC_Duration, lbl_AC_Level,lbl_AC_Credit, lbl_AC_NOA, lbl_AC_Leader, lbl_AC_Lecturer, lbl_AC_SDate, lbl_AC_CDatein ACjp in ACjp

SET Bounds for txt_AC_ID, txt_AC_Name, txt_AC_Duration, txt_AC_Level, txt_AC_Credit, txt_AC_NOA, txt_AC_Leader, txt_AC_Lecturer, txt_AC_SDate, txt_AC_CDate

ADD txt_AC_ID, txt_AC_Name, txt_AC_Duration, txt_AC_Level, txt_AC_Credit, txt_AC_NOA, txt_AC_Leader, txt_AC_Lecturer, txt_AC_SDate, txt_AC_CDate in ACjp

SET Bounds for btn_AC1, btn_NC1, btn_AC_add, btn_AC_Register, btn_AC_Clear, btn_AC_Display

ADD btn_AC1, btn_NC1, btn_AC_add, btn_AC_Register, btn_AC_Clear, btn_AC_Display in ACjp

SET Bounds for NCjp

SET Layout null in NCjp

SET Visible false NCjp

SET Bounds for lbl_NC, lbl_NC_ID, lbl_NC_Name, lbl_NC_Duration, lbl_NC_Prerequsit, lbl_NC_Leader, lbl_NC_Instrutor,lbl_NC_SDate, lbl_NC_CDate, lbl_NC_EDate

ADD lbl_NC, lbl_NC_ID, lbl_NC_Name, lbl_NC_Duration, lbl_NC_Prerequsit, lbl_NC_Leader, lbl_NC_Instrutor,lbl_NC_SDate, lbl_NC_CDate, lbl_NC_EDate in ACjp

SET Bounds for txt_NC_ID, txt_NC_Name, txt_NC_Duration, txt_NC_Prerequsit, txt_NC_Leader, txt_NC_Instrutor, txt_NC_SDate, txt_NC_CDate, txt_NC_EDate

ADD txt_NC_ID, txt_NC_Name, txt_NC_Duration, txt_NC_Prerequsit, txt_NC_Leader, txt_NC_Instrutor, txt_NC_SDate, txt_NC_CDate, txt_NC_EDate in ACjp

SET Bounds for btn_AC3, btn_NC3, btn_NC_add, btn_NC_Register, btn_NC_Remove, btn_NC_Clear, btn_NC_Display

ADD btn_AC3, btn_NC3, btn_NC_add, btn_NC_Register, btn_NC_Remove, btn_NC_Clear, btn_NC_Display in ACjp

CLICK btn_AC1 Action Performed

SET Visible true ACjp

SET Visible false NCjp

SET Visible false CPjp

CLICK btn_NC1 Action Performed

SET Visible false ACjp

SET Visible true NCjp

SET Visible false CPjp

CLICK btn_AC2 Action Performed

SET Visible true ACjp

SET Visible false NCjp

SET Visible false CPjp

CLICK btn_NC2 Action Performed

SET Visible false ACjp

SET Visible true NCjp

SET Visible false CPjp

CLICK btn_AC3 Action Performed

SET Visible true ACjp

SET Visible false NCjp

SET Visible false CPjp

CLICK btn_NC1 Action Performed

SET Visible false ACjp

SET Visible true NCjp

SET Visible false CPjp

ADD CPjp in jf

ADD ACjp in jf

ADD NCjp in jf

CLICK btn_AC_add Action Performed

IF txt_AC_ID, txt_AC_Name, txt_AC_Duration, txt_AC_Level, txt_AC_Credit, txt_AC_NOA is Empty

THEN Reassign message

ELSE ADD Entered value in academicCourseList

CLICK btn_AC_Register Action Performed

IF txt_AC_Leader, txt_AC_Lecturer, txt_AC_SDate, txt_AC_CDate is Empty

THEN Reassign message

ELSE

IF arraylist CourseID equal txt_NC

THEN Call Register and assign in AcademicCourse method Register

ELSE Invalid ID message

CLICK btn_AC_Clear Action Performed

Assign empty string to all text field

CLICK btn_AC_Display Action Performed

Display all available data added or registered

CLICK btn_NC_add Action Performed

IF txt_NC_ID, txt_NC_Name, txt_NC_Duration, txt_NC_Prerequsit is Empty

THEN Reassign message

ELSE ADD Entered value in nonAcademicCourseList

CLICK btn_NC_Register Action Performed

IF txt_NC_Leader, txt_NC_Instrutor, txt_NC_SDate, txt_NC_CDate, txt_NC_EDate is Empty

THEN Reassign message

ELSE

IF arraylist CourseID equal txt_NC

THEN Call Register and assign in NonAcademicCourse method Register

ELSE Invalid ID message

CLICK btn_NC_Remove Action Performed

IF txt_NC_Leader, txt_NC_Instrutor, txt_NC_SDate, txt_NC_CDate, txt_NC_EDate is Empty

THEN Already removed message

ELSE

IF arraylist CourseID equal txt_NC

THEN Call Remove method of NonAcademicCourse class

ELSE Invalid ID message

CLICK btn_NC_Clear Action Performed

Assign empty string to all text field

CLICK btn_NC_Display Action Performed

Display all available data added or registered

END constructor method

CREATE main method

PASS new INGCollege constructor method

END main method

END INGCollege class

4. Description

- INGCollege()

This is a constructor which is used to create a graphical user interface (GUI) and is linked with Course, AcademicCoures and NonAcademicCoures class to access its methods.

Inside the INGCollege method several other methods like actionPerformed (ActionEvent e), register (AC_CourseLeader, AC_LecturerName, AC_SDate, AC_CDate), register (NC_CourseLeader, NC_InstrutorName, NC_SDate, NC_CDate, NC_EDate) and removed ().

The method actionPerformed(ActionEvent e) is used whenever a button is pressed. This method is used for switching between tabs, add, register, remove data and clear text field and display entered data.

For register (AC_CourseLeader, AC_LecturerName, AC_SDate, AC_CDate) method, it is used to call register method of AcademicCourse class. In this method AC_CourseLeader, AC_LecturerName, AC_SDate, AC_CDate are passed as parameters. This method allows the users to register data in AcademicCourse class.

Similarly, register (NC_CourseLeader, NC_InstrutorName, NC_SDate, NC_CDate, NC_EDate) method is used to call register method of NonAcademicCourse class. In this method NC_CourseLeader, NC_InstrutorName, NC_SDate, NC_CDate, NC_EDate are passed as parameters. This method allows the users to register data in NonAcademicCourse class.

Also, remove () method is called to remove the values of NC_CourseLeader, NC_InstrutorName, NC_SDate, NC_CDate, NC_EDate of NonAcademicCourse class. Only if the data is already registered.

- main(String []args)
  In this method a new constructor class which is ING College is called.

5. Test

5.1 Test 1- Compile Using Command prompt

Table 1: Test 1

Test No.	1
Objective:	Compile Using Command prompt
Action:	>>Find INGCollege.java file in cammand prompt >>javac INGCollege.java >>java INGCollege.java
Expected Result:	GUI must be displayed.
Actual Resulr:	GUI is displayed.
Conclusion:	The test is successful

```
C:\Users\Adarsha>cd Desktop\All\islington\S2\Programming\Coursework\Coursework2

C:\Users\Adarsha\Desktop\All\islington\S2\Programming\Coursework\CourseWork2>javac INGCollege.java

C:\Users\Adarsha\Desktop\All\islington\S2\Programming\Coursework\CourseWork2>java INGCollege.java
```

Figure 3:Codes used in Command Prompt

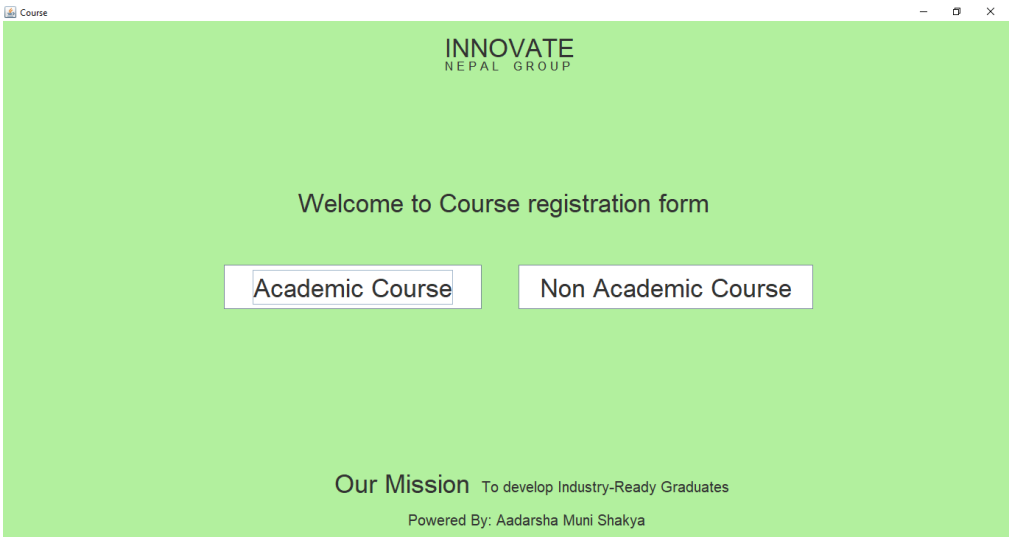

Figure 4: Opening GUI using Command Prompt

5.2 Test 2

5.2.1 Test 2.1 Add Course for Academic course

Table 2: Test 2.1

Test No.	2.1
Objective:	Add course for Academic course
	>>Assign values in Courses ID, Course Name, Duration, Level, Credit, Number of Assessments Course ID: 111aaa Course Name: ComputerScience Duration: 2 Level: 11 Credit: 2 Number of Assessments: 4 >>Click on Add button >>Click on Display button
Expected Result:	Should display "The records are added" dialog box
Actual Resulr:	"The records are added" dialog box displayed
Conclusion:	The test is successful

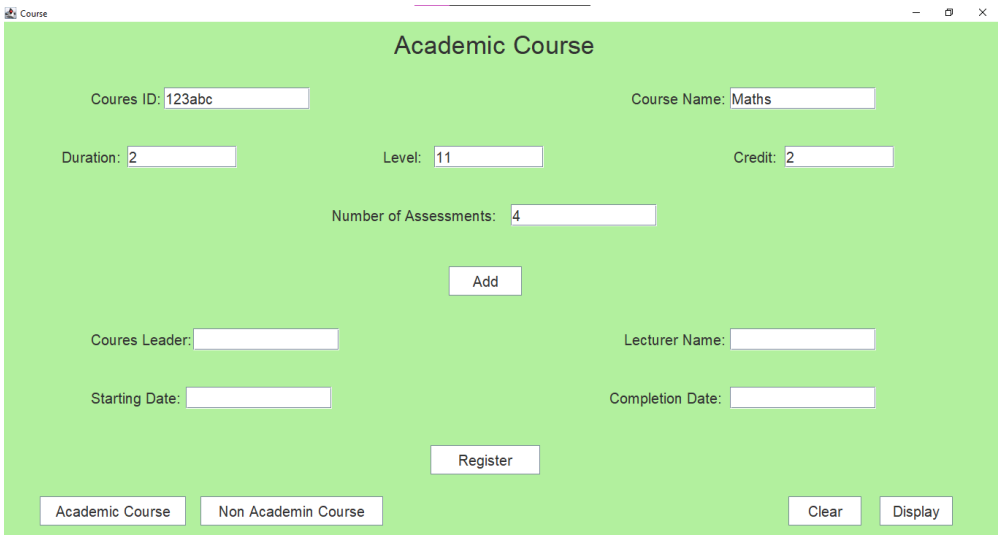

Figure 5: Assigning value in AcademicCourse

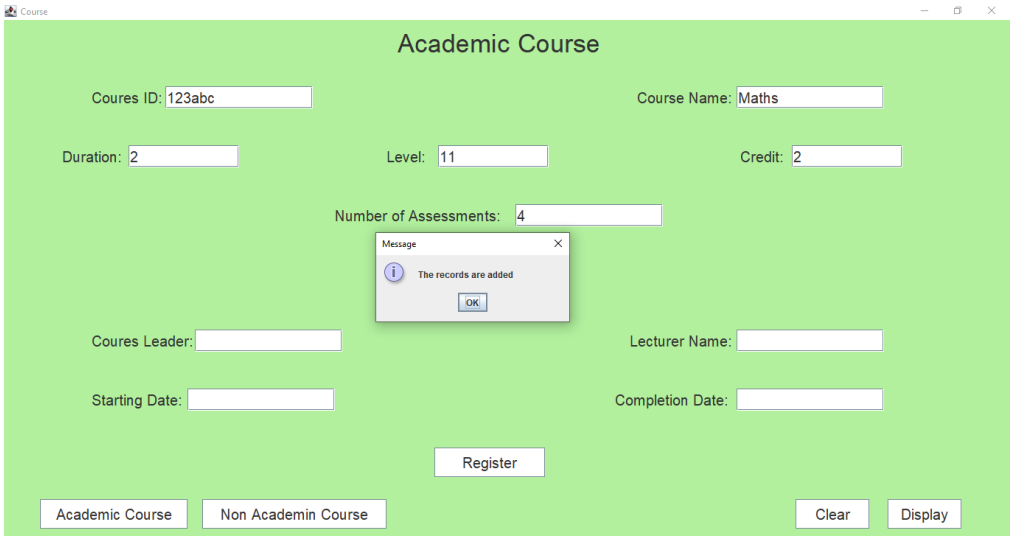

Figure 6: Dialog box when added inAcademicCourse

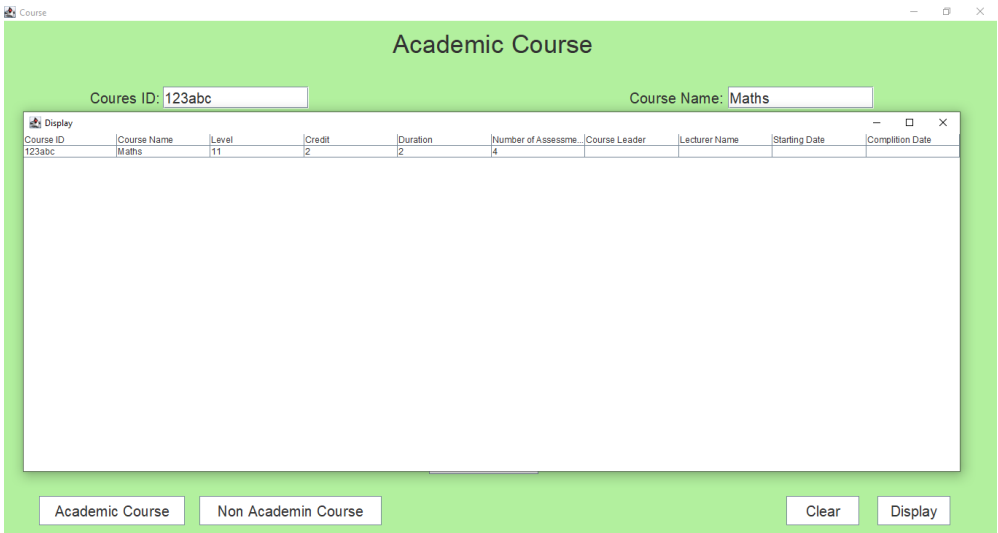

Figure 7: Display when only Add button is clicked in AcademicCourse

5.2.2 Test 2.2 Add Course for NonAcademic course

Table 3: Test 2.2

Test No.	2.2
Objective:	Add course for Non-academic course
	>>Assign values in Courses ID, Course Name, Duration, Prerequsit Course ID: 111aaa Course Name: ComputerScience Duration: 2 Prerequsit: C or above >>Click on Add button >>Click on Display button
Expected Result:	Should display "The records are added" dialog box
Actual Resulr:	"The records are added" dialog box displayed
Conclusion:	The test is successful

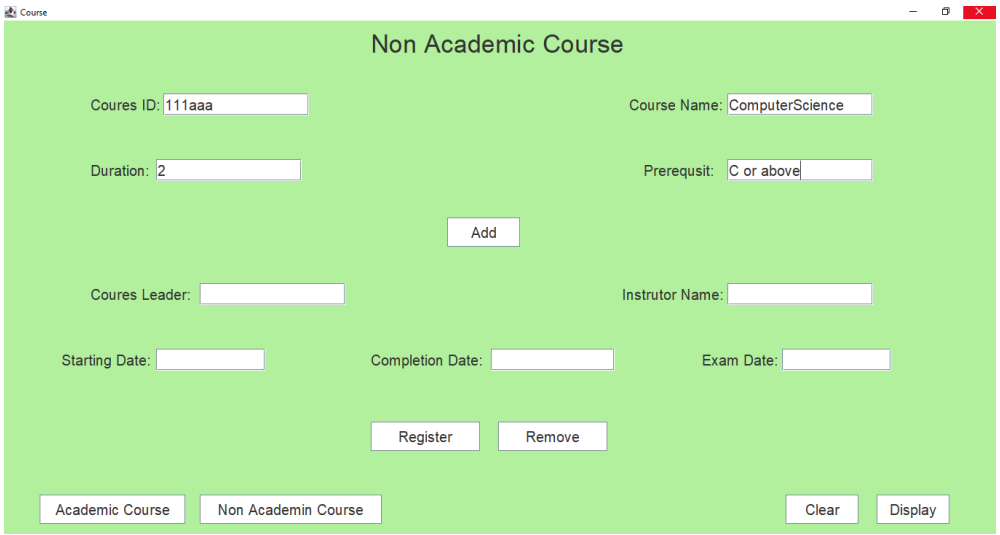

Figure 8:Assigning value in NonAcademicCourse

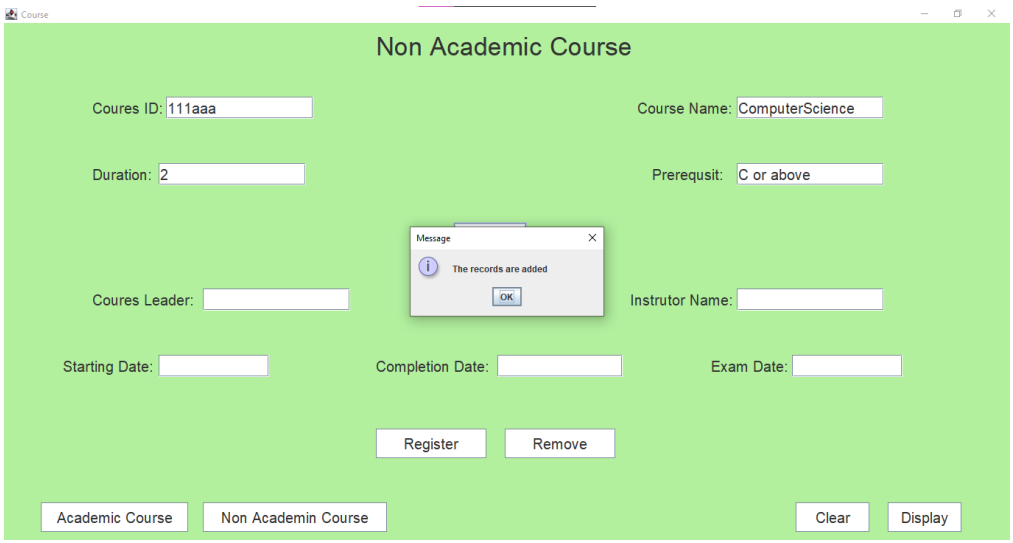

Figure 9:Dialog box when added in NonAcademicCourse

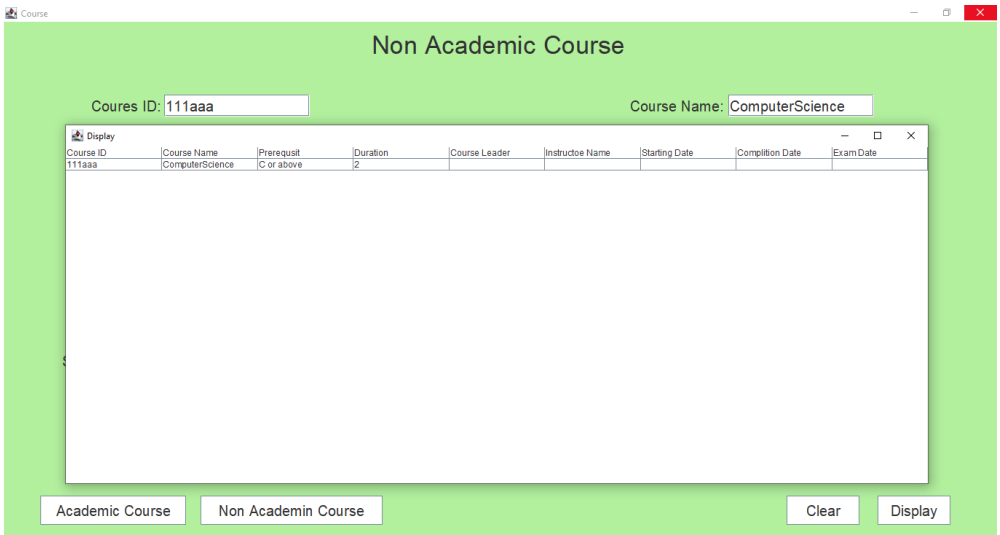

Figure 10:Display when only Add button is clicked in NonAcademicCourse

5.2.3 Test 2.3 Register academic course

Table 4:Test 2.3

Test No.	2.3
Objective:	Register academic course
	>>Assign values in Course Leader, Lecturer Name, Starting Date, Complition Date Course Leader: JJ Lecturer Name: Tobi Starting Data: April Completion Date: June >>Click on Register button >>Click on Display button
Expected Result:	Should display "The Academic coures is registered" dialog box and all records in a new frame
Actual Resulr:	"The Academic coures is registered" dialog box and display frame are displayed
Conclusion:	The test is successful

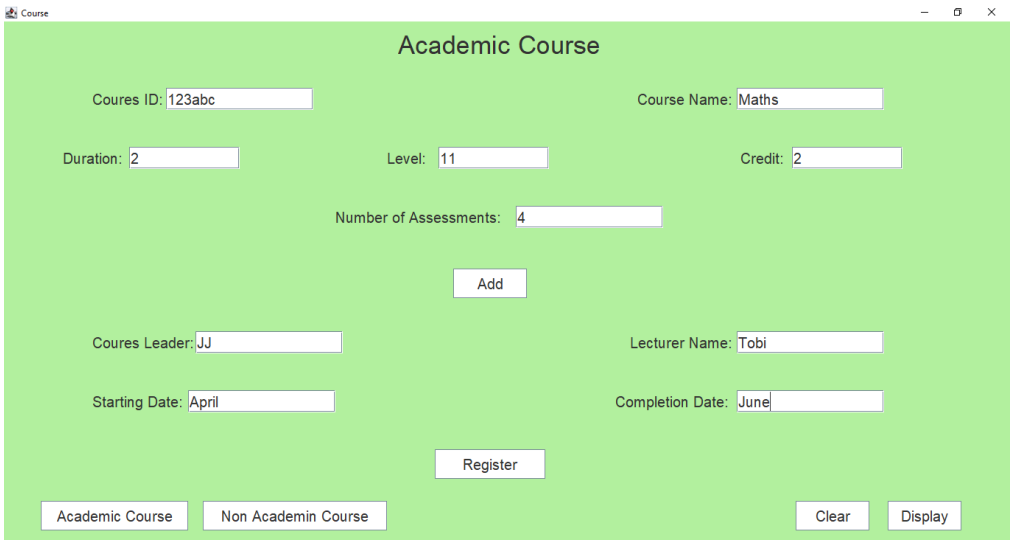

Figure 11: Assigning value in AcademicCourse

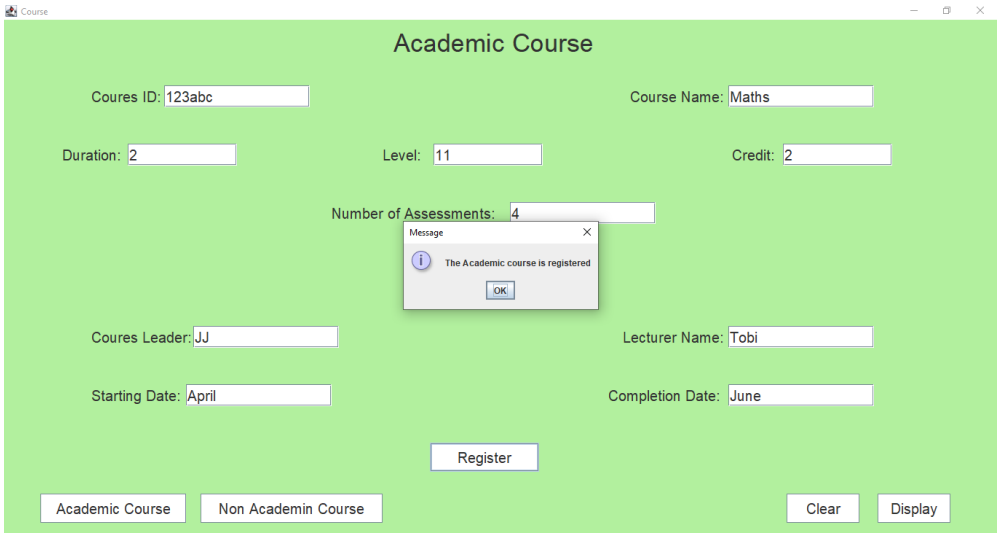

Figure 12: Dialog box when registered in AcademicCourse

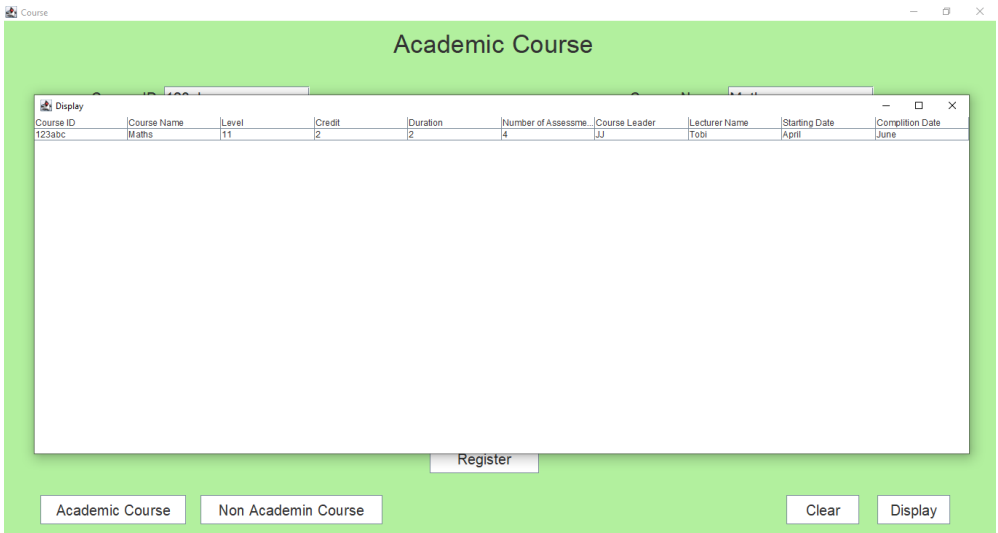

Figure 13: Display when Add and Register button is clicked in AcademicCourse

5.2.4 Test 2.4 Register non-academic course

Table 5:Test 2.4

Test No.	2.4
Objective:	Register non-academic course
	>>Assign values in Course Leader, Instructor Name, Starting Date, Complition Date, Exam Date Course Leader: Vikram Lecturer Name: Ethan Starting Data: June Completion Date: August Exam Date: September >>Click on Register button >>Click on Display button
Expected Result:	Should display "The Non Academic Coures is registered" dialog box and all records in a new frame
Actual Resulr:	"The Non Academic Coures is registered" dialog box and display frame are displayed
Conclusion:	The test is successful

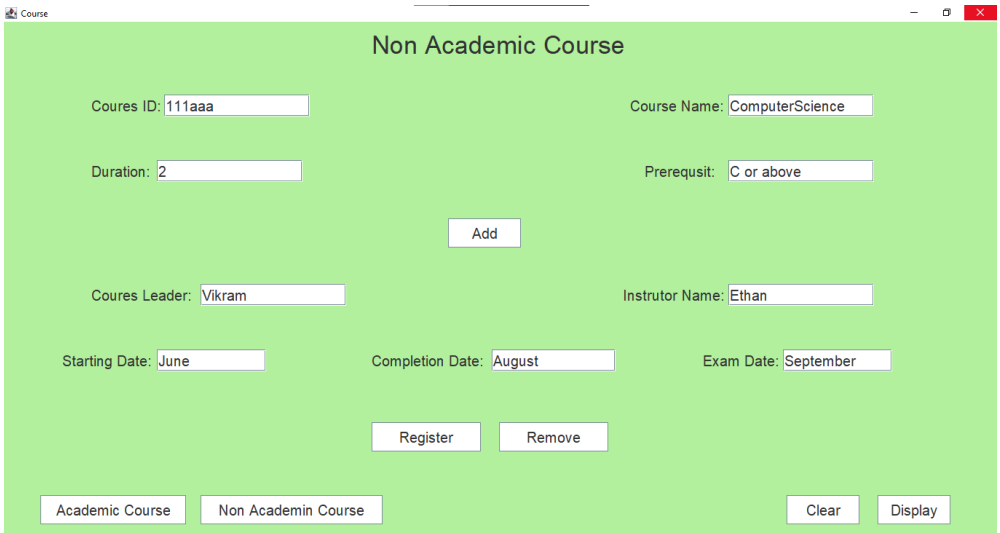

Figure 14: Assigning value in NonAcademicCourse

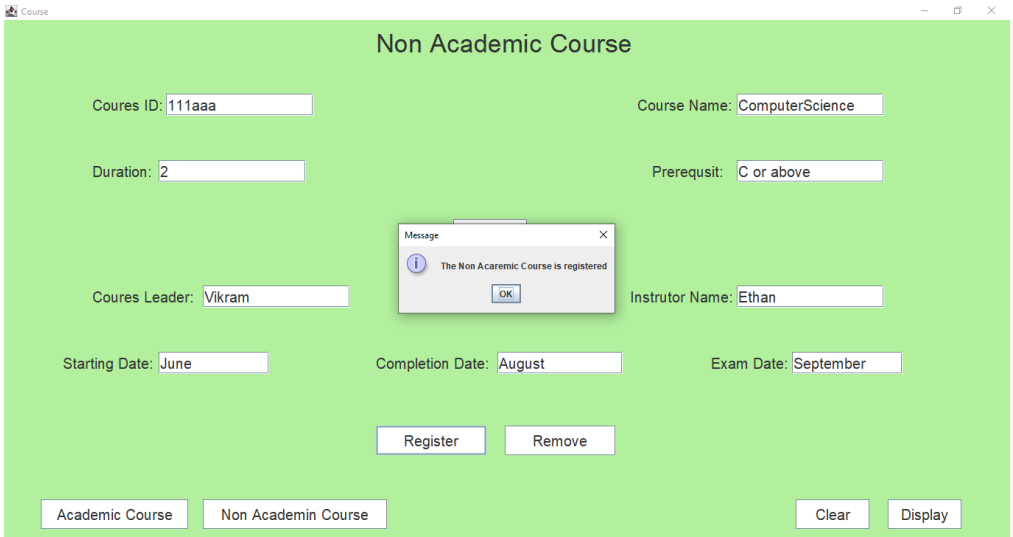

Figure 15: Dialog box when registered in NonAcademicCourse

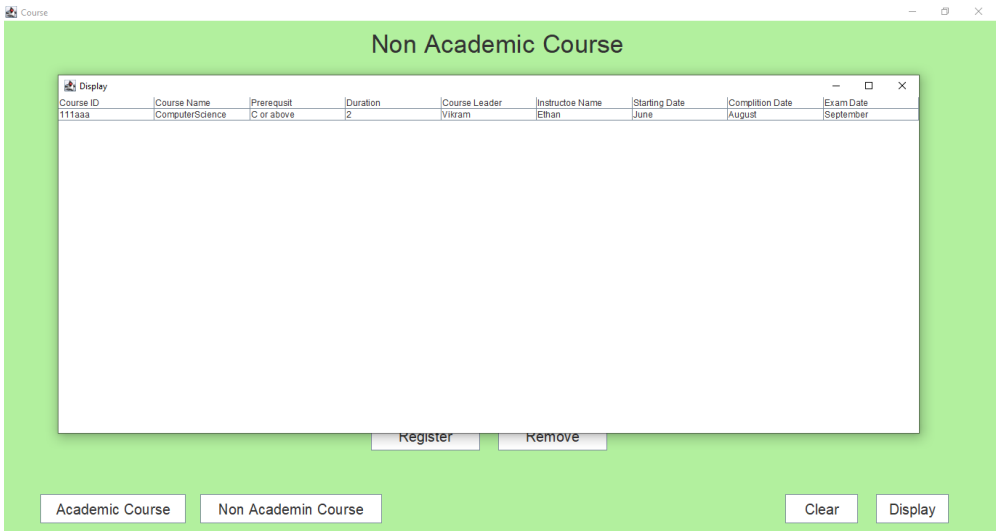

Figure 16: Display when Add and Register button is clicked in NonAcademicCourse

5.2.5 Test 2.5 Remove non-academic course

Table 6:Test 2.5

Test No.	2.5
Objective:	Remove non-academic course
	>>Click on Remove button after registering >>Click on Display button
Expected Result:	Should display "The Non Academic coures is removed" dialog box and all records in a new frame
Actual Resulr:	"The Non Academic coures is removed" dialog box and display with no registered datain display frame are displayed
Conclusion:	The test is successful

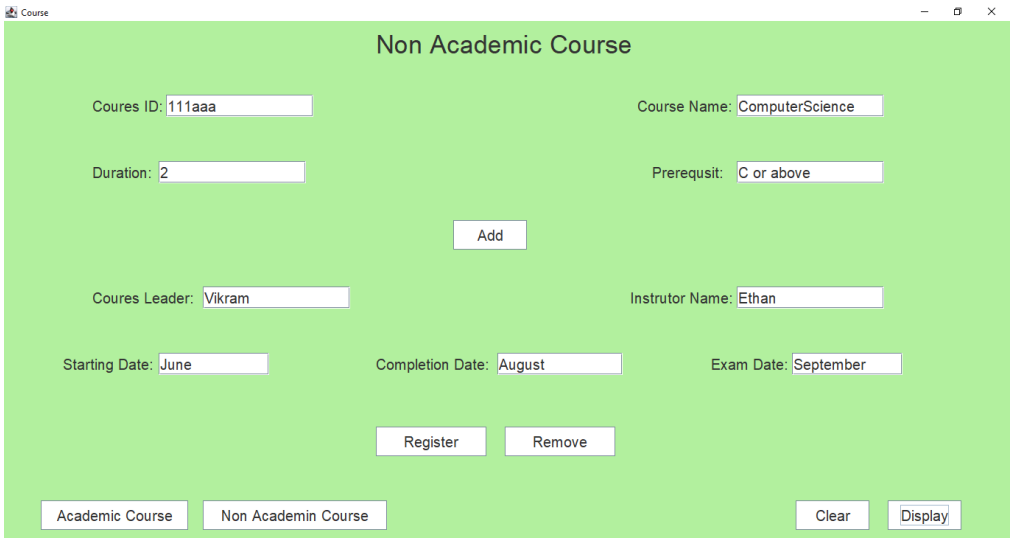

Figure 17: Assigned value in NonAcademicCourse

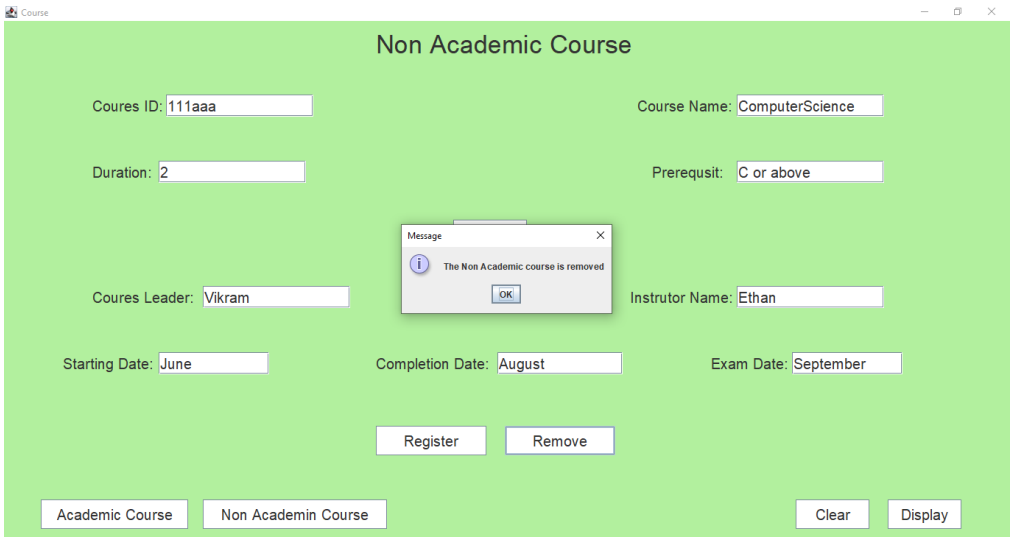

Figure 18: Dialog box when Removed in NonAcademicCourse

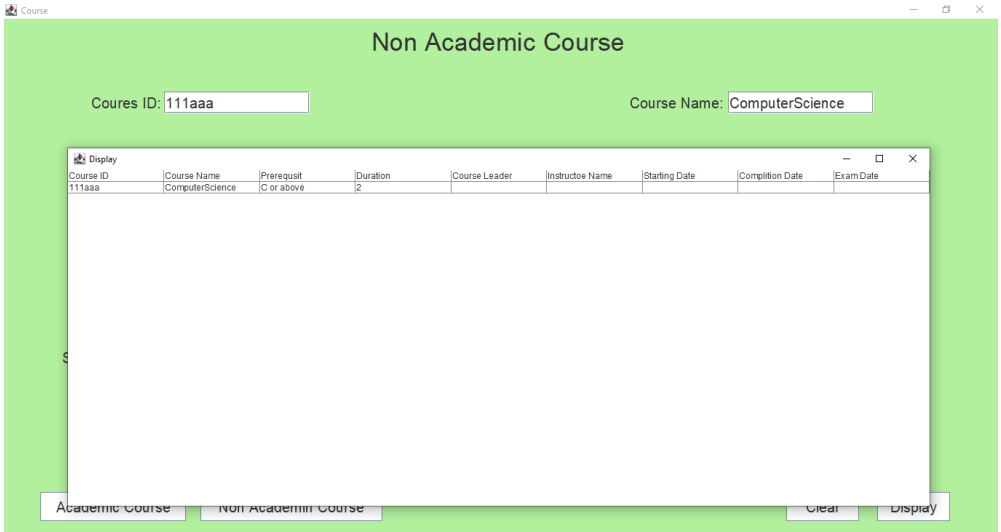

Figure 19: Display when Add and Remove button is clicked in NonAcademicCourse

5.3 Test 3

5.3.1 Test 3.1 Trying to add duplicate CourseID

Table 7: Test 3.1

Test No.	3.1
Objective:	Trying to add duplicate courseID
	>>Assign the same value in text fields as Test 2 >>Click on add button
Expected Result	Should display "Given ID is already added!!! Please try with a different ID" dialog box
Actual Resulr:	"Given ID is already added!!! Please try with a different ID" dialog box is displayed
Conclusion:	The test is successful

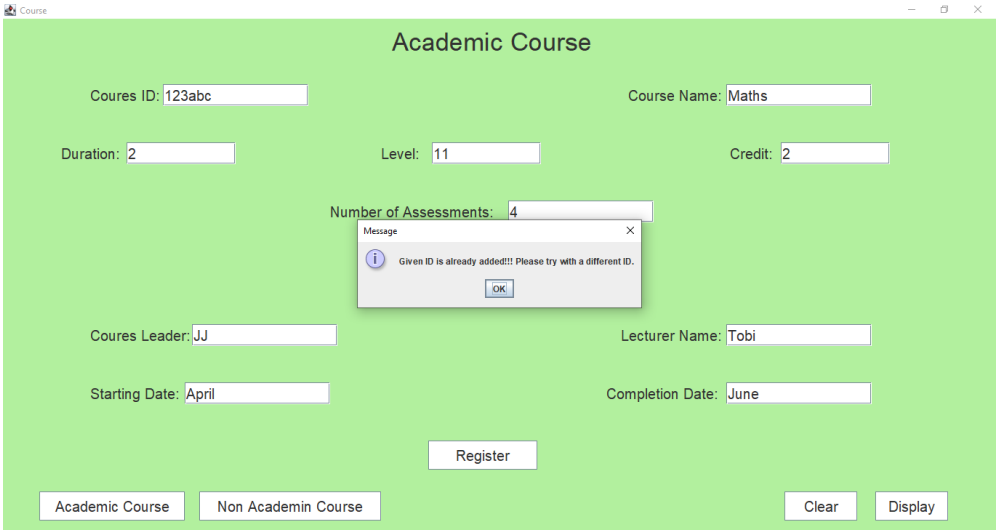

Figure 20: Assigning duplicat CourseID in AcademicCourse

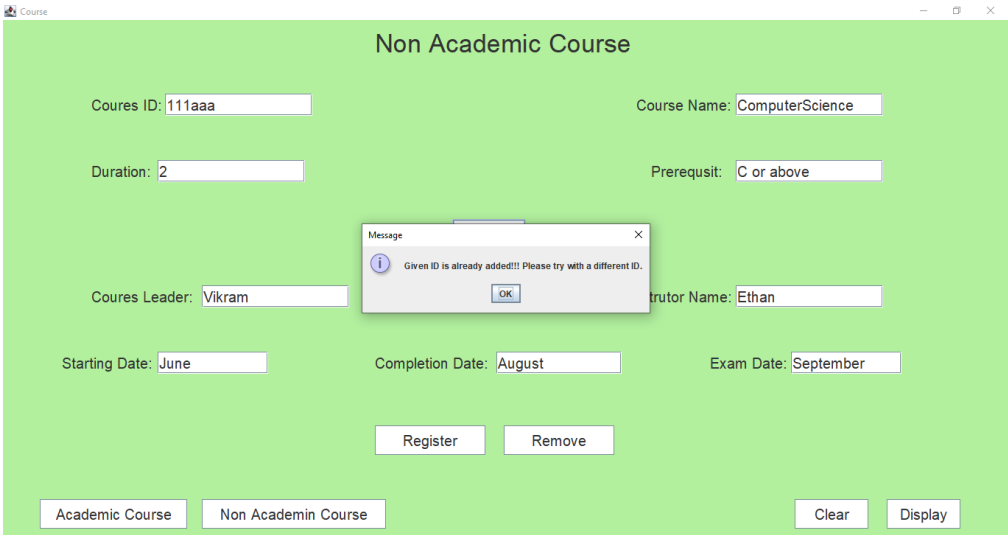

Figure 21:Assigning duplicat CourseID in NonAcademicCourse

5.3.2 Test 3.2 Trying to register already registered course

Table 8: Test 3.2

Test No.	3.2
Objective:	Trying to register already registered course.
	Assign the same value in text fields as Test 2 >>Click on Register button
Expected Result	Should display "The Academic course is already registered" or "The Non Academic course is already registered" dialog box
Actual Resulr:	"The Academic course is already registered" or "The Non Academic course is already registered" dialog box is displayed
Conclusion:	The test is successful

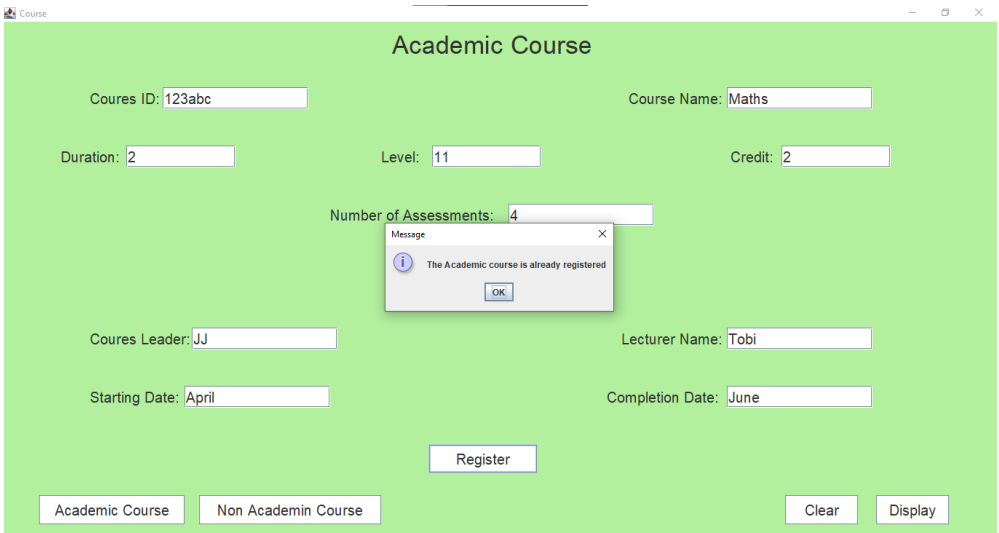

Figure 22: Registring already registered data in AcademicCourse

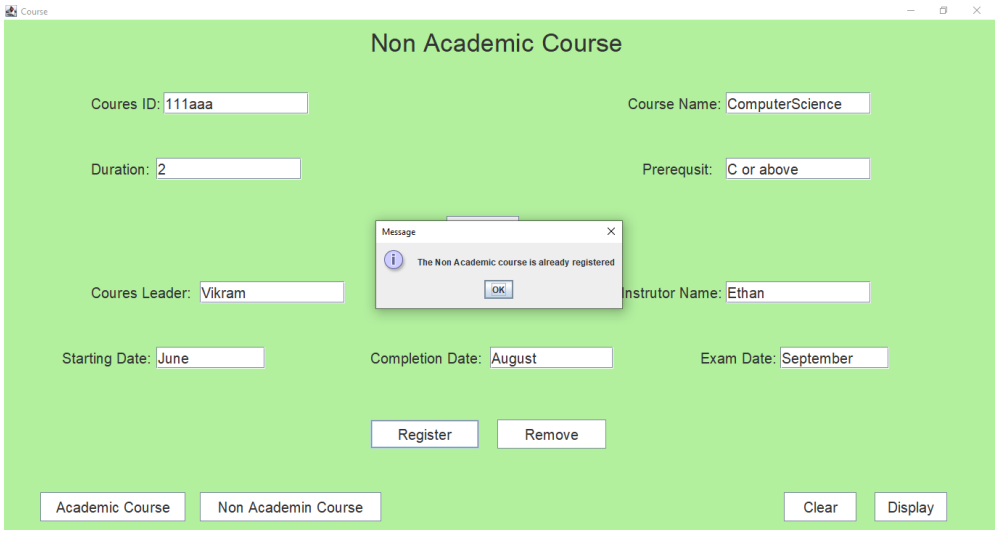

Figure 23: Registring already registered data in NonAcademicCourse

5.3.3 Test 3.3 Trying to remove the non-academic course which is already removed.

Table 9: Test 3.3

Test No.	3.3
Objective:	Trying to remove the non-academic course which is already removed.
	>>Click on Remove button after removind the non-academic course
Expected Result	Should display "The Non Academic course is already removed" dialog box
Actual Resulr:	"The Non Academic course is already removed" dialog box is displayed
Conclusion:	The test is successful

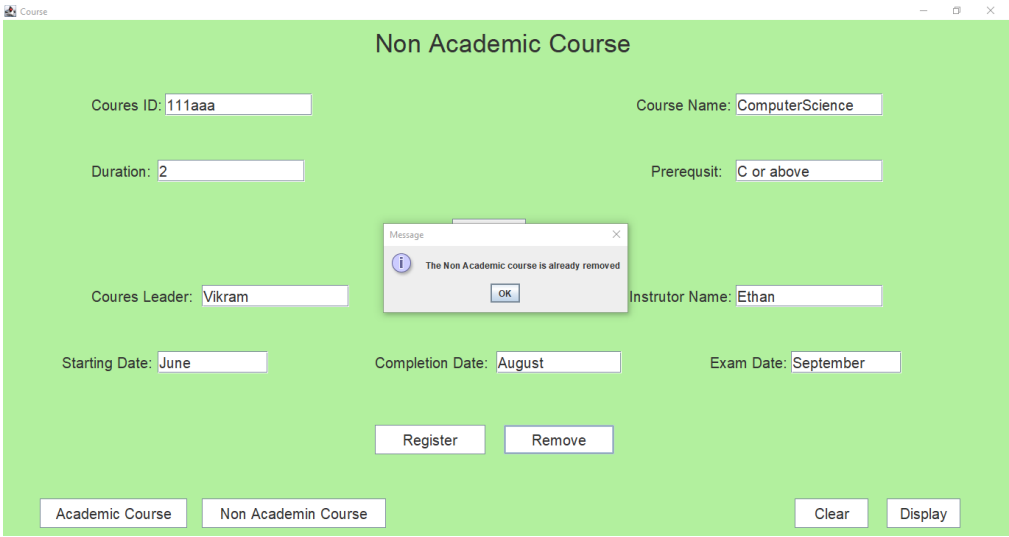

Figure 24: Removing non-acadamic course which is alerady removed

6. Error

6.1 Syntax error

A syntax error is an error in the source code of a program. Since computer programs must follow strict syntax to compile correctly, any aspects of the code that do not conform to the syntax of the programming language will produce a syntax error. (Christensson, 2012)

Here big brackets are used instead of curly brackets

```
///Clear button
btn_AC_Clear.addActionListener(new ActionListener[]
    {
        public void actionPerformed(ActionEvent e)
        {
            txt_AC_ID.setText("");
            txt_AC_Name.setText("");
            txt_AC_Duration.setText("");
            txt_AC_Level.setText("");
            txt_AC_Credit.setText("");
            txt_AC_NOA.setText("");
            txt_AC_Leader.setText("");
            txt_AC_Lecturer.setText("");
            txt_AC_SDate.setText("");
            txt_AC_CDate.setText("");
            JOptionPane.showMessageDialog(jf,"All text fields are cleared");
        }
    }
);
```

Figure 25: Syntax error

To solve these parentheses are used instead of big brackets

```
btn_AC_Clear.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            txt_AC_ID.setText("");
            txt_AC_Name.setText("");
            txt_AC_Duration.setText("");
            txt_AC_Level.setText("");
            txt_AC_Credit.setText("");
            txt_AC_NOA.setText("");
            txt_AC_Leader.setText("");
            txt_AC_Lecturer.setText("");
            txt_AC_SDate.setText("");
            txt_AC_CDate.setText("");
            JOptionPane.showMessageDialog(jf,"All text fields are cleared");
        }
    }
);
```

Figure 26: Solved Syntax error

6.2 Logical error

A logic error or logical error is a mistake in a program's source code that results in incorrect or unexpected behavior. (Christensson, TechTerms, 2012)

Here When the variables are empty “The records are added” dialog box is displayed and when the records are successfully added “The text field is empty” dialog box is displayed.

```
if (AC_CourseID.isEmpty() || AC_Name.isEmpty() || AC_Level.isEmpty() || AC_Credit.isEmpty())
{
    JOptionPane.showMessageDialog(jf,"The records are added");
}
else
{
    for(Course c :academicCourseList)
    {
        if(AC_CourseID.equals(c.getCourseID()))
        {
            JOptionPane.showMessageDialog(jf,"Given ID is already added!!! Please try with a different ID.");
            return;
        }
    }
    Course AC_obj_add = new AcademicCourse(AC_CourseID,AC_Name,AC_Duration,AC_Level,AC_Credit,AC_NOA);
    academicCourseList.add(AC_obj_add);

    JOptionPane.showMessageDialog(jf,"The text field is empty");
}
```

Figure 27: Logical Error

To solve this the message displayed by the dialogbox is interchanged.

```
if (AC_CourseID.isEmpty() || AC_Name.isEmpty() || AC_Level.isEmpty() || AC_Credit.isEmpty())
{
    JOptionPane.showMessageDialog(jf,"The text field is empty");
}
else
{
    for(Course c :academicCourseList)
    {
        if(AC_CourseID.equals(c.getCourseID()))
        {
            JOptionPane.showMessageDialog(jf,"Given ID is already added!!! Please try with a different ID.");
            return;
        }
    }
    Course AC_obj_add = new AcademicCourse(AC_CourseID,AC_Name,AC_Duration,AC_Level,AC_Credit,AC_NOA);
    academicCourseList.add(AC_obj_add);
    JOptionPane.showMessageDialog(jf,"The records are added");
}
```

Figure 28: Solved Logical Error

6.3 Run time error

It is a type of error that may simply produce the wrong output or may cause a program to crash while running. (Christensson, TechTerms, 2012)

Here the assigned string value of duration is converted into integer. When the user enters a string in the text field the program crashes.

```
public void actionPerformed(ActionEvent e)
    {
        String NC_CourseID = txt_NC_ID.getText();
        String NC_Name = txt_NC_Name.getText();
        String NC_Prerequsit = txt_NC_Prerequsit.getText();




            String NC_Duration_d = txt_NC_Duration.getText();
            int NC_Duration = Integer.parseInt(NC_Duration_d);

            if(NC_Duration_d.isEmpty())
            {
                JOptionPane.showMessageDialog(jf,"The text field is empty");
            }
            else
            {
                JOptionPane.showMessageDialog(jf,"Invalid data type!!!");
            }



        if (NC_CourseID.isEmpty() || NC_Name.isEmpty() || NC_Prerequsit.isEmpty())
        {
            JOptionPane.showMessageDialog(jf,"The text field is empty");
        }
        else
        {
```

Figure 29: Run time error

```
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "fg"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.base/java.lang.Integer.parseInt(Integer.java:652)
        at java.base/java.lang.Integer.parseInt(Integer.java:770)
        at INGCollege$11.actionPerformed(INGCollege.java:699)
        at java.desktop/javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1967)
        at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2308)
        at java.desktop/javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:405)
        at java.desktop/javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
        at java.desktop/javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
        at java.desktop/java.awt.Component.processMouseEvent(Component.java:6632)
        at java.desktop/javax.swing.JComponent.processMouseEvent(JComponent.java:3342)
        at java.desktop/java.awt.Component.processEvent(Component.java:6397)
        at java.desktop/java.awt.Container.processEvent(Container.java:2263)
        at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:5008)
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2321)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4840)
        at java.desktop/java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4918)
        at java.desktop/java.awt.LightweightDispatcher.processMouseEvent(Container.java:4547)
```

Figure 30: Run time Error (BlueJ Terminal Error)

To solve this try catch are used.

```
public void actionPerformed(ActionEvent e)
    {
        String AC_CourseID = txt_AC_ID.getText();
        String AC_Name = txt_AC_Name.getText();
        String AC_Level = txt_AC_Level.getText();
        String AC_Credit = txt_AC_Credit.getText();
        /// try catch for integers i.e. duration and number of assessements
        try
        {
            String AC_Duration_d = txt_AC_Duration.getText();
            int AC_Duration = Integer.parseInt(AC_Duration_d);
            String AC_NOA_d = txt_AC_NOA.getText();
            int AC_NOA = Integer.parseInt(AC_NOA_d);
        }
        catch(Exception ex)
        {
            String AC_Duration_d = txt_AC_Duration.getText();
            String AC_NOA_d = txt_AC_NOA.getText();
            if(AC_Duration_d.isEmpty() || AC_NOA_d.isEmpty())
            {
                JOptionPane.showMessageDialog(jf,"The text field is empty");
            }
            else
            {
                JOptionPane.showMessageDialog(jf,"Invalid data type!!!");
            }
        }

        String AC_NOA_d = txt_AC_NOA.getText();
        int AC_NOA = Integer.parseInt(AC_NOA_d);
```

Figure 31: Solved Run time Error

6.4 Semantic Error

A semantic error occurs when a statement is syntactically valid, but does not do what the programmer intended. (Alex, 2019)

Here jf JFrame is not initilized and will compile but thd frame will not be displayed.

```
private JFrame jf,AC_df,NC_df;
private JPanel CPjp,ACjp,NCjp;
private JLabel  lbl_i, lbl_ng, lbl_Welcome, lbl_Vision, lbl_Mission,lbl_by,
                lbl_AC, lbl_AC_ID, lbl_AC_Name,lbl_AC_Duration, lbl_AC_Level,lbl_AC_Credit, lbl_AC_NOA,
                lbl_AC_Leader, lbl_AC_Lecturer, lbl_AC_SDate, lbl_AC_CDate,
                lbl_NC, lbl_NC_ID, lbl_NC_Name, lbl_NC_Duration, lbl_NC_Prerequsit,
                lbl_NC_Leader, lbl_NC_Instrutor,lbl_NC_SDate, lbl_NC_CDate, lbl_NC_EDate;
private JTextField txt_AC_ID, txt_AC_Name, txt_AC_Duration, txt_AC_Level, txt_AC_Credit, txt_AC_NOA, txt_AC_Leader, txt_AC_Lecturer,
                   txt_AC_SDate, txt_AC_CDate, txt_NC_ID, txt_NC_Name, txt_NC_Duration, txt_NC_Prerequsit, txt_NC_Leader, txt_NC_Instrutor,
                   txt_NC_SDate, txt_NC_CDate, txt_NC_EDate;
private JButton btn_AC1, btn_NC1, btn_AC2, btn_NC2, btn_AC3, btn_NC3,
                btn_AC_add, btn_AC_Register, btn_AC_Clear, btn_AC_Display,
                btn_NC_add, btn_NC_Register, btn_NC_Remove, btn_NC_Clear, btn_NC_Display;
private Font fnt1, fnt2, fnt3;
private JTable table_AC_Display, table_NC_Display;
private ArrayList<Course> nonAcademicCourseList, academicCourseList;
private DefaultTableModel AC_TableModel, NC_TableModel;
public INGCollege()
{

    //Frame for Academic Course and Non Academic Course
    jf.setBounds(10,10,1350,800);
    jf.setLayout(null);
    jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);
```

Figure 32: Semantic Error

To solve this jf can be initialized by writing the code “jf = new JFrame(“Course”)”

```
private JFrame jf,AC_df,NC_df;
private JPanel CPjp,ACjp,NCjp;
private JLabel  lbl_i, lbl_ng, lbl_Welcome, lbl_Vision, lbl_Mission,lbl_by,
                lbl_AC, lbl_AC_ID, lbl_AC_Name,lbl_AC_Duration, lbl_AC_Level,lbl_AC_Credit, lbl_AC_NOA,
                lbl_AC_Leader, lbl_AC_Lecturer, lbl_AC_SDate, lbl_AC_CDate,
                lbl_NC, lbl_NC_ID, lbl_NC_Name, lbl_NC_Duration, lbl_NC_Prerequsit,
                lbl_NC_Leader, lbl_NC_Instrutor,lbl_NC_SDate, lbl_NC_CDate, lbl_NC_EDate;
private JTextField txt_AC_ID, txt_AC_Name, txt_AC_Duration, txt_AC_Level, txt_AC_Credit, txt_AC_NOA, txt_AC_Leader, txt_AC_Lecturer,
                   txt_AC_SDate, txt_AC_CDate, txt_NC_ID, txt_NC_Name, txt_NC_Duration, txt_NC_Prerequsit, txt_NC_Leader, txt_NC_Instrutor,
                   txt_NC_SDate, txt_NC_CDate, txt_NC_EDate;
private JButton btn_AC1, btn_NC1, btn_AC2, btn_NC2, btn_AC3, btn_NC3,
                btn_AC_add, btn_AC_Register, btn_AC_Clear, btn_AC_Display,
                btn_NC_add, btn_NC_Register, btn_NC_Remove, btn_NC_Clear, btn_NC_Display;
private Font fnt1, fnt2, fnt3;
private JTable table_AC_Display, table_NC_Display;
private ArrayList<Course> nonAcademicCourseList, academicCourseList;
private DefaultTableModel AC_TableModel, NC_TableModel;
public INGCollege()
{

    //Frame for Academic Course and Non Academic Course
    jf = new JFrame("Course");
    jf.setBounds(10,10,1350,800);
    jf.setLayout(null);
    jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);
```

Figure 33: Solved Semantic Error

7. Conclusion

For conclusion, this project was all about creating a graphical user interface (GUI) for the program designed in coursework 1. In this coursework There is a welcome page; Inside the welcome page there are two buttons which displays Academic or Non-Academic when clicked. Tasks like adding academic and non-academic courses, registering academic and non-academic courses, removing non-academic courses, clearing text fields and displaying multiple data's can be done.

Everything learned in this coursework was new which led to a lot of confusions. For example, in concepts of down casting, upcasting and try-catch. Without down casting an object of parent class type cannot be converted into child class type. Similarly, without upcasting an object of child class type cannot be converted into parent class type. Also, without try-catch the exceptions micht lead to the crashing of program and clearing these doubts was very important.

To clear those confusions the teachers have provided videos regarding those topics which led to solving those difficulties. Down casting was used when an object of Course class was down casted to academic or non-academic class used to access its methods. For upcasting, this concept was used to convert array list which is in Course class type to Academic Course class type and store in a new object of Academic Course class type. Also, try-catch is used for exceptions; If a user assigns string value to an integer text field the program will crash. In these cases, try-catch should be used.

8. Appendix1

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.util.*;

import javax.swing.table.DefaultTableModel;

public class INGCollege

{

    private JFrame jf,AC_df,NC_df;

    private JPanel CPjp,ACjp,NCjp;

    private JLabel  lbl_i, lbl_ng, lbl_Welcome, lbl_Vision, lbl_Mission,lbl_by,lbl_AC,
lbl_AC_ID, lbl_AC_Name,lbl_AC_Duration, lbl_AC_Level,lbl_AC_Credit, lbl_AC_NOA,
lbl_AC_Leader, lbl_AC_Lecturer, lbl_AC_SDate, lbl_AC_CDate, lbl_NC, lbl_NC_ID,
lbl_NC_Name, lbl_NC_Duration, lbl_NC_Prerequsit, lbl_NC_Leader,
lbl_NC_Instrutor,lbl_NC_SDate, lbl_NC_CDate, lbl_NC_EDate;

private JButton btn_AC1, btn_NC1, btn_AC2, btn_NC2, btn_AC3, btn_NC3,
btn_AC_add, btn_AC_Register, btn_AC_Clear, btn_AC_Display, btn_NC_add,
btn_NC_Register, btn_NC_Remove, btn_NC_Clear, btn_NC_Display;

private Font fnt1, fnt2, fnt3;

    private JTable table_AC_Display, table_NC_Display;

    private ArrayList<Course> nonAcademicCourseList, academicCourseList;

    private DefaultTableModel AC_TableModel, NC_TableModel;

public INGCollege()

    {



        //Frame for Academic Course and Non Academic Course

        jf = new JFrame("Course");

        jf.setBounds(10,10,1350,800);

        jf.setLayout(null);

        jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);
```

```
//Font for title
fnt1 = new Font("Areal",Font.PLAIN,35);

//Font for normal content
fnt2 = new Font("Areal",Font.PLAIN,20);

//Font for logo
fnt3 = new Font("Areal",Font.PLAIN,17);

//Cover page panel
CPjp = new JPanel();
CPjp.setBounds(0,0,1440,900);
CPjp.setBackground(new Color(178, 240, 158));
CPjp.setLayout(null);


//logo
lbl_i = new JLabel("INNOVATE");
lbl_i.setBounds(600,10,400,50);
lbl_i.setFont(fnt1);
CPjp.add(lbl_i);

lbl_ng = new JLabel("N E P A L    G R O U P");
lbl_ng.setBounds(600,35,400,50);
lbl_ng.setFont(fnt3);
CPjp.add(lbl_ng);
```

```
//Welcome message
lbl_Welcome = new JLabel("Welcome to Course registration form");
lbl_Welcome.setBounds(400,220,950,50);
lbl_Welcome.setFont(fnt1);
CPjp.add(lbl_Welcome);

//Academic Button
btn_AC3 = new JButton("Academic Course");
btn_AC3.setBounds(300,330,350,60);
btn_AC3.setFont(fnt1);
btn_AC3.setBackground(new Color(255,255,255));
CPjp.add(btn_AC3);
//Non Academic Button
btn_NC3 = new JButton("Non Academic Course");
btn_NC3.setBounds(700,330,400,60);
btn_NC3.setFont(fnt1);
btn_NC3.setBackground(new Color(255,255,255));
CPjp.add(btn_NC3);


//Mission
lbl_Vision = new JLabel("Our Mission");
lbl_Vision.setBounds(450,600,200,50);
lbl_Vision.setFont(fnt1);
CPjp.add(lbl_Vision);

//Message Mission
lbl_Mission = new JLabel("To develop Industry-Ready Graduates");
```

```
lbl_Mission.setBounds(650,605,400,50);
lbl_Mission.setFont(fnt2);
CPjp.add(lbl_Mission);

//Powered-by
lbl_by = new JLabel("Powered By: Aadarsha Muni Shakya");
lbl_by.setBounds(550,650,400,50);
lbl_by.setFont(fnt2);
CPjp.add(lbl_by);

//Academic Course Panel
ACjp = new JPanel();
ACjp.setBounds(0,0,1440,900);
ACjp.setBackground(new Color(178, 240, 158));
ACjp.setLayout(null);



//Row 1
//Title Academic Coures
lbl_AC = new JLabel("Academic Course");
lbl_AC.setBounds(535,5,700,50);
lbl_AC.setFont(fnt1);
ACjp.add(lbl_AC);

//Row 2
///CourseID
lbl_AC_ID = new JLabel("Coures ID:");
lbl_AC_ID.setBounds(120,80,100,50);
```

```
lbl_AC_ID.setFont(fnt2);
ACjp.add(lbl_AC_ID);
///Coures ID text field
txt_AC_ID = new JTextField();
txt_AC_ID.setBounds(220,90,200,30);
txt_AC_ID.setFont(fnt2);
ACjp.add(txt_AC_ID);

//Coures Name
lbl_AC_Name = new JLabel("Course Name:");
lbl_AC_Name.setBounds(860,80,140,50);
lbl_AC_Name.setFont(fnt2);
ACjp.add(lbl_AC_Name);
///Course Name text field
txt_AC_Name = new JTextField();
txt_AC_Name.setBounds(995,90,200,30);
txt_AC_Name.setFont(fnt2);
ACjp.add(txt_AC_Name);

//row 3
///Duration
lbl_AC_Duration = new JLabel("Duration:");
lbl_AC_Duration.setBounds(80,160,90,50);
lbl_AC_Duration.setFont(fnt2);
ACjp.add(lbl_AC_Duration);
///Duration text field
txt_AC_Duration = new JTextField();
txt_AC_Duration.setBounds(170,170,150,30);
```

```
txt_AC_Duration.setFont(fnt2);
ACjp.add(txt_AC_Duration);
///Level
lbl_AC_Level = new JLabel("Level:");
lbl_AC_Level.setBounds(520,160,70,50);
lbl_AC_Level.setFont(fnt2);
ACjp.add(lbl_AC_Level);
///Level text field
txt_AC_Level = new JTextField();
txt_AC_Level.setBounds(590,170,150,30);
txt_AC_Level.setFont(fnt2);
ACjp.add(txt_AC_Level);
///Credit
lbl_AC_Credit = new JLabel("Credit:");
lbl_AC_Credit.setBounds(1000,160,70,50);
lbl_AC_Credit.setFont(fnt2);
ACjp.add(lbl_AC_Credit);
/// Credit Text field
txt_AC_Credit = new JTextField();
txt_AC_Credit.setBounds(1070,170,150,30);
txt_AC_Credit.setFont(fnt2);
ACjp.add(txt_AC_Credit);

//Row 4
//Number of Assessments
lbl_AC_NOA = new JLabel("Number of Assessments:");
lbl_AC_NOA.setBounds(450,240,240,50);
lbl_AC_NOA.setFont(fnt2);
```

```
ACjp.add(lbl_AC_NOA);
///Number of Assessments text field
txt_AC_NOA = new JTextField();
txt_AC_NOA.setBounds(695,250,200,30);
txt_AC_NOA.setFont(fnt2);
ACjp.add(txt_AC_NOA);

//Row 5 Add button
btn_AC_add = new JButton("Add");
btn_AC_add.setBounds(610,335,100,40);
btn_AC_add.setFont(fnt2);
btn_AC_add.setBackground(Color.WHITE);
ACjp.add(btn_AC_add);

//Row 6
//Coures Leader
lbl_AC_Leader = new JLabel("Coures Leader:");
lbl_AC_Leader.setBounds(120,410,150,50);
lbl_AC_Leader.setFont(fnt2);
ACjp.add(lbl_AC_Leader);
///Coures Leader text field
txt_AC_Leader = new JTextField();
txt_AC_Leader.setBounds(260,420,200,30);
txt_AC_Leader.setFont(fnt2);
ACjp.add(txt_AC_Leader);

//Lecturer Name
lbl_AC_Lecturer = new JLabel("Lecturer Name:");
```

```
lbl_AC_Lecturer.setBounds(850,410,150,50);
lbl_AC_Lecturer.setFont(fnt2);
ACjp.add(lbl_AC_Lecturer);
///Course Name text field
txt_AC_Lecturer = new JTextField();
txt_AC_Lecturer.setBounds(995,420,200,30);
txt_AC_Lecturer.setFont(fnt2);
ACjp.add(txt_AC_Lecturer);

//Row 7
//Starting Date
lbl_AC_SDate = new JLabel("Starting Date:");
lbl_AC_SDate.setBounds(120,490,150,50);
lbl_AC_SDate.setFont(fnt2);
ACjp.add(lbl_AC_SDate);
///Starting Date text field
txt_AC_SDate = new JTextField();
txt_AC_SDate.setBounds(250,500,200,30);
txt_AC_SDate.setFont(fnt2);
ACjp.add(txt_AC_SDate);

//Completion Date
lbl_AC_CDate = new JLabel("Completion Date:");
lbl_AC_CDate.setBounds(830,490,170,50);
lbl_AC_CDate.setFont(fnt2);
ACjp.add(lbl_AC_CDate);
///Course Name text field
txt_AC_CDate = new JTextField();
```

```
txt_AC_CDate.setBounds(995,500,200,30);
txt_AC_CDate.setFont(fnt2);
ACjp.add(txt_AC_CDate);

//Row 8
btn_AC_Register = new JButton("Register");
btn_AC_Register.setBounds(585,580,150,40);
btn_AC_Register.setFont(fnt2);
btn_AC_Register.setBackground(Color.WHITE);
ACjp.add(btn_AC_Register);

//Row 9
//Changing AC and NC
btn_AC1 = new JButton("Academic Course");
btn_AC1.setBounds(50,650,200,40);
btn_AC1.setFont(fnt2);
btn_AC1.setBackground(Color.WHITE);
btn_NC1 = new JButton("Non Academin Course");
btn_NC1.setBounds(270,650,250,40);
btn_NC1.setFont(fnt2);
btn_NC1.setBackground(Color.WHITE);
ACjp.add(btn_AC1);
ACjp.add(btn_NC1);

//Clear Button
btn_AC_Clear = new JButton("Clear");
btn_AC_Clear.setBounds(1075,650,100,40);
btn_AC_Clear.setBackground(Color.WHITE);
```

```
btn_AC_Clear.setFont(fnt2);
ACjp.add(btn_AC_Clear);

//Display Button
btn_AC_Display = new JButton("Display");
btn_AC_Display.setBounds(1200,650,100,40);
btn_AC_Display.setBackground(Color.WHITE);
btn_AC_Display.setFont(fnt2);
ACjp.add(btn_AC_Display);

//Panel for Non Academic Course
NCjp = new JPanel();
NCjp.setBounds(0,0,1440,900);
NCjp.setBackground(new Color(178, 240, 158));
NCjp.setLayout(null);

//Row 1
//Title Academic Coures
lbl_NC = new JLabel("Non Academic Course");
lbl_NC.setBounds(505,5,400,50);
lbl_NC.setFont(fnt1);
NCjp.add(lbl_NC);

//Row 2
///CourseID
lbl_NC_ID = new JLabel("Coures ID:");
lbl_NC_ID.setBounds(120,90,100,50);
lbl_NC_ID.setFont(fnt2);
```

```
NCjp.add(lbl_NC_ID);
///Coures ID text field
txt_NC_ID = new JTextField();
txt_NC_ID.setBounds(220,100,200,30);
txt_NC_ID.setFont(fnt2);
NCjp.add(txt_NC_ID);

//Coures Name
lbl_NC_Name = new JLabel("Course Name:");
lbl_NC_Name.setBounds(860,90,140,50);
lbl_NC_Name.setFont(fnt2);
NCjp.add(lbl_NC_Name);
///Course Name text field
txt_NC_Name = new JTextField();
txt_NC_Name.setBounds(995,100,200,30);
txt_NC_Name.setFont(fnt2);
NCjp.add(txt_NC_Name);

//row 3
///Duration
lbl_NC_Duration = new JLabel("Duration:");
lbl_NC_Duration.setBounds(120,180,100,50);
lbl_NC_Duration.setFont(fnt2);
NCjp.add(lbl_NC_Duration);
///Duration text field
txt_NC_Duration = new JTextField();
txt_NC_Duration.setBounds(210,190,200,30);
txt_NC_Duration.setFont(fnt2);
```

```
NCjp.add(txt_NC_Duration);
///Prerequsit
lbl_NC_Prerequsit = new JLabel("Prerequsit:");
lbl_NC_Prerequsit.setBounds(880,180,100,50);
lbl_NC_Prerequsit.setFont(fnt2);
NCjp.add(lbl_NC_Prerequsit);
///Prerequsit Text field
txt_NC_Prerequsit = new JTextField();
txt_NC_Prerequsit.setBounds(995,190,200,30);
txt_NC_Prerequsit.setFont(fnt2);
NCjp.add(txt_NC_Prerequsit);

//Row 4
btn_NC_add = new JButton("Add");
btn_NC_add.setBounds(610,270,100,40);
btn_NC_add.setFont(fnt2);
btn_NC_add.setBackground(Color.WHITE);
NCjp.add(btn_NC_add);

//Row 5
//Coures Leader
lbl_NC_Leader = new JLabel("Coures Leader:");
lbl_NC_Leader.setBounds(120,350,140,50);
lbl_NC_Leader.setFont(fnt2);
NCjp.add(lbl_NC_Leader);
///Coures Leader text field
txt_NC_Leader = new JTextField();
txt_NC_Leader.setBounds(270,360,200,30);
```

```
txt_NC_Leader.setFont(fnt2);
NCjp.add(txt_NC_Leader);

//Instrutor Name
lbl_NC_Instrutor = new JLabel("Instrutor Name:");
lbl_NC_Instrutor.setBounds(850,350,140,50);
lbl_NC_Instrutor.setFont(fnt2);
NCjp.add(lbl_NC_Instrutor);
///Course Name text field
txt_NC_Instrutor = new JTextField();
txt_NC_Instrutor.setBounds(995,360,200,30);
txt_NC_Instrutor.setFont(fnt2);
NCjp.add(txt_NC_Instrutor);

//Row6
///Starting Date
lbl_NC_SDate = new JLabel("Starting Date:");
lbl_NC_SDate.setBounds(80,440,150,50);
lbl_NC_SDate.setFont(fnt2);
NCjp.add(lbl_NC_SDate);
///Starting Date text field
txt_NC_SDate = new JTextField();
txt_NC_SDate.setBounds(210,450,150,30);
txt_NC_SDate.setFont(fnt2);
NCjp.add(txt_NC_SDate);
///Completion Date
lbl_NC_CDate = new JLabel("Completion Date:");
lbl_NC_CDate.setBounds(505,440,170,50);
```

```
lbl_NC_CDate.setFont(fnt2);
NCjp.add(lbl_NC_CDate);
///CDate text field
txt_NC_CDate = new JTextField();
txt_NC_CDate.setBounds(670,450,170,30);
txt_NC_CDate.setFont(fnt2);
NCjp.add(txt_NC_CDate);
///EDate
lbl_NC_EDate = new JLabel("Exam Date:");
lbl_NC_EDate.setBounds(960,440,150,50);
lbl_NC_EDate.setFont(fnt2);
NCjp.add(lbl_NC_EDate);
/// EDate Text field
txt_NC_EDate = new JTextField();
txt_NC_EDate.setBounds(1070,450,150,30);
txt_NC_EDate.setFont(fnt2);
NCjp.add(txt_NC_EDate);


//Row 7
//Register Button
btn_NC_Register = new JButton("Register");
btn_NC_Register.setBounds(505,550,150,40);
btn_NC_Register.setFont(fnt2);
btn_NC_Register.setBackground(Color.WHITE);
NCjp.add(btn_NC_Register);
//Remove Button
btn_NC_Remove = new JButton("Remove");
```

```
btn_NC_Remove.setBounds(680,550,150,40);
btn_NC_Remove.setFont(fnt2);
btn_NC_Remove.setBackground(Color.WHITE);
NCjp.add(btn_NC_Remove);

//Row 8
//Changing AC and NC
btn_AC2 = new JButton("Academic Course");
btn_AC2.setBounds(50,650,200,40);
btn_AC2.setFont(fnt2);
btn_AC2.setBackground(Color.WHITE);
btn_NC2 = new JButton("Non Academin Course");
btn_NC2.setBounds(270,650,250,40);
btn_NC2.setFont(fnt2);
btn_NC2.setBackground(Color.WHITE);
NCjp.add(btn_AC2);
NCjp.add(btn_NC2);



//Clear Button
btn_NC_Clear = new JButton("Clear");
btn_NC_Clear.setBounds(1075,650,100,40);
btn_NC_Clear.setFont(fnt2);
btn_NC_Clear.setBackground(Color.WHITE);
NCjp.add(btn_NC_Clear);
//Display Button
btn_NC_Display = new JButton("Display");
```

```
btn_NC_Display.setBounds(1200,650,100,40);
btn_NC_Display.setFont(fnt2);
btn_NC_Display.setBackground(Color.WHITE);
NCjp.add(btn_NC_Display);









//Action Listene
//switching between tabs
btn_AC1.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    ACjp.setVisible(true);
                    NCjp.setVisible(false);
                    CPjp.setVisible(false);
                }
        }
    );
btn_NC1.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    ACjp.setVisible(false);
```

```
                NCjp.setVisible(true);
                CPjp.setVisible(false);
            }
        }
    );
btn_AC2.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                NCjp.setVisible(false);
                ACjp.setVisible(true);
                CPjp.setVisible(false);
            }
        }
    );
btn_NC2.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
               NCjp.setVisible(true);
               ACjp.setVisible(false);
               CPjp.setVisible(false);
            }
        }
    );
btn_AC3.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
```

```
                {
                    ACjp.setVisible(true);
                    NCjp.setVisible(false);
                    CPjp.setVisible(false);
                }
            }
        );
    btn_NC3.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    ACjp.setVisible(false);
                    NCjp.setVisible(true);
                    CPjp.setVisible(false);
                }
            }
        );
    jf.add(CPjp);
    jf.add(ACjp);
    jf.add(NCjp);
///creating array list of course class
    academicCourseList = new ArrayList<Course>();
    nonAcademicCourseList = new ArrayList<Course>();

    ///For academic course
    //add button

    btn_AC_add.addActionListener(new ActionListener()
```

```
{
    public void actionPerformed(ActionEvent e)
      {
        String AC_CourseID = txt_AC_ID.getText();
        String AC_Name = txt_AC_Name.getText();
        String AC_Level = txt_AC_Level.getText();
        String AC_Credit = txt_AC_Credit.getText();
        /// try catch for integers i.e. duration and number of assessements
        try
        {
          String AC_Duration_d = txt_AC_Duration.getText();
          int AC_Duration = Integer.parseInt(AC_Duration_d);
          String AC_NOA_d = txt_AC_NOA.getText();
          int AC_NOA = Integer.parseInt(AC_NOA_d);
        }
        catch(Exception ex)
        {
          String AC_Duration_d = txt_AC_Duration.getText();
          String AC_NOA_d = txt_AC_NOA.getText();
          if(AC_Duration_d.isEmpty() || AC_NOA_d.isEmpty())
          {
            JOptionPane.showMessageDialog(jf,"The text field is empty");
          }
          else
          {
            JOptionPane.showMessageDialog(jf,"Invalid data type!!!");
          }
        }
```

```
                    String AC_NOA_d = txt_AC_NOA.getText();
                    int AC_NOA = Integer.parseInt(AC_NOA_d);
                    String AC_Duration_d = txt_AC_Duration.getText();
                    int AC_Duration = Integer.parseInt(AC_Duration_d);
                    if (AC_CourseID.isEmpty() || AC_Name.isEmpty() ||
AC_Level.isEmpty() || AC_Credit.isEmpty())
                    {
                        JOptionPane.showMessageDialog(jf,"The text field is empty");
                    }
                    else
                    {
                        for(Course c :academicCourseList)
                        {
                            if(AC_CourseID.equals(c.getCourseID()))
                            {
                                JOptionPane.showMessageDialog(jf,"Given ID is already
added!!! Please try with a different ID.");
                                return;
                            }
                        }
                        Course AC_obj_add = new
AcademicCourse(AC_CourseID,AC_Name,AC_Duration,AC_Level,AC_Credit,AC_NOA
);
                        academicCourseList.add(AC_obj_add);
                        JOptionPane.showMessageDialog(jf,"The records are added");
                    }
                }
            }
        );
```

```
///Register button
btn_AC_Register.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
            {
                String AC_CourseLeader = txt_AC_Leader.getText();
                String AC_LecturerName = txt_AC_Lecturer.getText();
                String AC_SDate = txt_AC_SDate.getText();
                String AC_CDate = txt_AC_CDate.getText();

                if(AC_CourseLeader.isEmpty() || AC_LecturerName.isEmpty() ||
AC_SDate.isEmpty() || AC_CDate.isEmpty())
                    {
                        JOptionPane.showMessageDialog(jf,"The text field is empty");
                    }
                    else
                    {
                        for (int i = 0 ; i<academicCourseList.size();i++)
                        {

if(academicCourseList.get(i).getCourseID().equals(txt_AC_ID.getText()))
                            {
                                AcademicCourse ac =
(AcademicCourse)academicCourseList.get(i);
                                if(!ac.getIsRegistered())
                                {
                                    ac.register(AC_CourseLeader, AC_LecturerName,
AC_SDate, AC_CDate);
                                    JOptionPane.showMessageDialog(jf,"The Academic
course is registered");
```

```
                                    }
                                    else if(ac.getIsRegistered())
                                    {
                                        JOptionPane.showMessageDialog(jf,"The Academic
course is already registered");
                                    }


                                    else
                                    {
                                        JOptionPane.showMessageDialog(jf,"The Course ID
dosen't match!!!");
                                    }
                                }
                            }
                        }


                    }
                }
            );
        ///Clear button
        btn_AC_Clear.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    txt_AC_ID.setText("");
                    txt_AC_Name.setText("");
                    txt_AC_Duration.setText("");
                    txt_AC_Level.setText("");
                    txt_AC_Credit.setText("");
```

```
            txt_AC_NOA.setText("");
            txt_AC_Leader.setText("");
            txt_AC_Lecturer.setText("");
            txt_AC_SDate.setText("");
            txt_AC_CDate.setText("");
            JOptionPane.showMessageDialog(jf,"All text fields are cleared");
        }
    }
);
/// Display button
btn_AC_Display.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            AC_df = new JFrame("Display");
            AC_df.setBounds(80,150,1300,500);



            AC_TableModel = new DefaultTableModel();

            table_AC_Display = new JTable(AC_TableModel);

            AC_TableModel.addColumn("Course ID");
            AC_TableModel.addColumn("Course Name");
            AC_TableModel.addColumn("Level");
            AC_TableModel.addColumn("Credit");
            AC_TableModel.addColumn("Duration");
            AC_TableModel.addColumn("Number of Assessments");
```

```
AC_TableModel.addColumn("Course Leader");
AC_TableModel.addColumn("Lecturer Name");
AC_TableModel.addColumn("Starting Date");
AC_TableModel.addColumn("Complition Date");




String[] columnNames = {"Course ID","Course
Name","Level","Credit","Duration","Number of Assessments","Course Leader","Lecturer
Name","Starting Date","Complition Date"};


AC_TableModel.addRow(columnNames);


for (int i = 0; i<academicCourseList.size();i++)
{
   if(academicCourseList.get(i).getCourseID().equals(txt_AC_ID.getText()))
   {
      AcademicCourse AC =
(AcademicCourse)(academicCourseList.get(i));
      String AC_CourseID = AC.getCourseID();
      String AC_Name =  AC.getCourseName();
      String AC_Level = AC.getLevel();
      String AC_Credit = AC.getCredit();
      int AC_Duration_d = AC.getDuration();
      String AC_Duration = Integer.toString(AC_Duration_d);
      int AC_NOA_d = AC.getNumberOfAssessments();
      String AC_NOA = Integer.toString(AC_NOA_d);
      String AC_CourseLeader = AC.getCourseLeader();
      String AC_LecturerName = AC.getLecturerName();
```

```
                    String AC_SDate = AC.getStartingDate();
                    String AC_CDate = AC.getCompletionDate();


                    String[] data =
{AC_CourseID,AC_Name,AC_Level,AC_Credit,AC_Duration,AC_NOA,AC_CourseLead
er,AC_LecturerName,AC_SDate,AC_CDate};
                    AC_TableModel.addRow(data);
                }
            }
            AC_df.add(table_AC_Display);
            AC_df.setVisible(true);
        }
    }
);
///Action listener for NonAcademicCourse
///Add button
btn_NC_add.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
                {
                    String NC_CourseID = txt_NC_ID.getText();
                    String NC_Name = txt_NC_Name.getText();
                    String NC_Prerequsit = txt_NC_Prerequsit.getText();


                    ///Try catch for integer
                    try
                    {
                        String NC_Duration_d = txt_NC_Duration.getText();
                        int NC_Duration = Integer.parseInt(NC_Duration_d);
```

```
                    }
                    catch(NumberFormatException ex)
                    {
                        String NC_Duration_d = txt_NC_Duration.getText();
                        if(NC_Duration_d.isEmpty())
                        {
                            JOptionPane.showMessageDialog(jf,"The text field is empty");
                        }
                        else
                        {
                            JOptionPane.showMessageDialog(jf,"Invalid data type!!!");
                        }
                    }

                    String NC_Duration_d = txt_NC_Duration.getText();
                    int NC_Duration = Integer.parseInt(NC_Duration_d);
                     if (NC_CourseID.isEmpty() || NC_Name.isEmpty() ||
NC_Prerequsit.isEmpty())
                    {
                        JOptionPane.showMessageDialog(jf,"The text field is empty");
                    }
                    else
                    {
                        for (Course c : nonAcademicCourseList)
                        {
                            if(NC_CourseID.equals(c.getCourseID()))
                            {
                                JOptionPane.showMessageDialog(jf,"Given ID is already
added!!! Please try with a different ID.");
```

```
                return;
            }
        }
        NonAcademicCourse NC_obj_add = new
NonAcademicCourse(NC_CourseID,NC_Name,NC_Duration,NC_Prerequsit);
        nonAcademicCourseList.add(NC_obj_add);
        JOptionPane.showMessageDialog(jf,"The records are added");
    }
  }
}
);
///Register button
btn_NC_Register.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    String NC_CourseLeader = txt_NC_Leader.getText();
    String NC_InstrutorName = txt_NC_Instrutor.getText();
    String NC_SDate = txt_NC_SDate.getText();
    String NC_CDate = txt_NC_CDate.getText();
    String NC_EDate = txt_NC_EDate.getText();

    if(NC_CourseLeader.isEmpty() || NC_InstrutorName.isEmpty() ||
NC_SDate.isEmpty() || NC_CDate.isEmpty() || NC_EDate.isEmpty())
    {
        JOptionPane.showMessageDialog(jf,"The text field is empty");
    }
    else
    {
```

```
                    for(int i = 0;i<nonAcademicCourseList.size();i++)
                    {

if(nonAcademicCourseList.get(i).getCourseID().equals(txt_NC_ID.getText()))
                        {
                            NonAcademicCourse nac = (NonAcademicCourse)
nonAcademicCourseList.get(i);
                            if(!nac.getIsRegistered())
                            {
                                nac.register(NC_CourseLeader, NC_InstrutorName,
NC_SDate, NC_CDate, NC_EDate);


                                JOptionPane.showMessageDialog(jf,"The Non Acaremic
Course is registered");
                            }
                            else if(nac.getIsRegistered())
                            {
                                JOptionPane.showMessageDialog(jf,"The Non Academic
course is already registered");
                            }
                            else
                            {
                                JOptionPane.showMessageDialog(jf,"The Course ID dosent
match!!!");
                            }
                        }


                    }
                }
            }
```

```
            }
        );
    ///Remove button
    btn_NC_Remove.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                    {
                        if (txt_NC_Leader.getText().isEmpty() ||
txt_NC_Instrutor.getText().isEmpty() || txt_NC_SDate.getText().isEmpty() ||
txt_NC_CDate.getText().isEmpty() ||txt_NC_EDate.getText().isEmpty())
                        {
                            JOptionPane.showMessageDialog(jf,"The text field is empty");
                        }
                        else
                        {
                            for(int i = 0; i<nonAcademicCourseList.size(); i++ )
                            {

if(nonAcademicCourseList.get(i).getCourseID().equals(txt_NC_ID.getText()))
                                {
                                    NonAcademicCourse nac = (NonAcademicCourse)
nonAcademicCourseList.get(i);
                                    if(!nac.getIsRemoved())
                                    {
                                        nac.remove();
                                        JOptionPane.showMessageDialog(jf,"The Non Academic
course is removed");
                                    }
                                    else if(nac.getIsRemoved())
                                    {
```

```
                                JOptionPane.showMessageDialog(jf,"The Non Academic
course is already removed");
                            }
                            else
                            {
                                JOptionPane.showMessageDialog(jf,"The Course ID dosent
match!!!");
                            }
                        }
                    }
                }
            }
        }
    );
    ///Clear button
    btn_NC_Clear.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            txt_NC_ID.setText("");
            txt_NC_Name.setText("");
            txt_NC_Duration.setText("");
            txt_NC_Prerequsit.setText("");
            txt_NC_Leader.setText("");
            txt_NC_Instrutor.setText("");
            txt_NC_SDate.setText("");
            txt_NC_CDate.setText("");
            txt_NC_EDate.setText("");
            JOptionPane.showMessageDialog(jf,"All text fields are cleared");
```

```
            }
        }
    );
    /// Display button
    btn_NC_Display.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                //Display frame for Non Academic
                NC_df = new JFrame("Display");
                NC_df.setBounds(80,200,1200,500);



                NC_TableModel = new DefaultTableModel();

                table_NC_Display = new JTable(NC_TableModel);

                NC_TableModel.addColumn("Course ID");
                NC_TableModel.addColumn("Course Name");
                NC_TableModel.addColumn("Prerequsit");
                NC_TableModel.addColumn("Duration");
                NC_TableModel.addColumn("Course Leader");
                NC_TableModel.addColumn("Instructoe Name");
                NC_TableModel.addColumn("Starting Date");
                NC_TableModel.addColumn("Complition Date");
                NC_TableModel.addColumn("Exam Date");

                // Column Names
```

```
            String[] columnNames = {"Course ID","Course
Name","Prerequsit","Duration","Course Leader","Instructoe Name","Starting
Date","Complition Date","Exam Date"};


            NC_TableModel.addRow(columnNames);

            for (int i = 0; i< nonAcademicCourseList.size(); i++)

            {

if(nonAcademicCourseList.get(i).getCourseID().equals(txt_NC_ID.getText()))

              {

                NonAcademicCourse NC =
(NonAcademicCourse)(nonAcademicCourseList.get(i));

                String NC_CourseID = NC.getCourseID();

                String NC_Name = NC.getCourseName();

                String NC_Prerequsit = NC.getPrerequisite();

                int NC_Duration_d = NC.getDuration();

                String NC_Duration = Integer.toString(NC_Duration_d);

                String NC_CourseLeader = NC.getCourseLeader();

                String NC_InstrutorName = NC.getInstrutorName();

                String NC_SDate = NC.getStartDate();

                String NC_CDate = NC.getCompletionDate();

                String NC_EDate = NC.getExamDate();


                String[] data = {NC_CourseID, NC_Name, NC_Prerequsit,
NC_Duration, NC_CourseLeader, NC_InstrutorName, NC_SDate, NC_CDate,
NC_EDate};

                NC_TableModel.addRow(data);

              }

              NC_df.add(table_NC_Display);

              NC_df.setVisible(true);
```

```
                }
            }
        }
    );
    jf.setVisible(true);
    CPjp.setVisible(true);
    ACjp.setVisible(false);
    NCjp.setVisible(false);
  }
public static void main(String []args)
  {
    new INGCollege();
  }
}
```

9. Appendix2

```
public class Course
{
  //ivars decleration
  private String courseID;
  private String courseName;
  private String courseLeader;
  private int duration;

  //Creation of Constructor class
  public Course(String courseID, String courseName, int duration)
  {
    //initalizing ivars to given parameters
    this.courseID = courseID;
```

```
        this.courseName = courseName;
        this.courseLeader = "";
        this.duration = duration;
    }

    //Use of Getter to let the method to be accessed
    public String getCourseID()
    {
        return this.courseID;
    }

public String getCourseName()
    {
        return this.courseName;
    }
    public int getDuration()
    {
        return this.duration;
    }
    public String getCourseLeader()
    {
        return this.courseLeader;
    }

    //Use of setter to set new name
    public void setCourseLeader(String courseLeader)
    {
        //initializing value of parameter to ivar-courseLeader
```

```
        this.courseLeader = courseLeader;
    }

    //Display method









public void display()
    {
        String COutput;//COutput stands for Course Output
        COutput ="The Course ID is "+courseID+" ,Course Name is "+courseName+"
,Duration is "+duration+" months.";
        if(this.courseLeader.isEmpty() == false )
        {
            COutput = COutput +"  and Course Leader is "+ courseLeader;
        }
        System.out.println(COutput);


    }
}
package Programming;


public class AcademicCourse extends Course
{
    //ivars decleration
```

```
    private String lecturerName;
    private String level;
    private String credit;
    private String startingDate;
    private String completionDate;
    private int numberOfAssessments;
    private boolean isRegistered;




//Creation of Constructor class
public AcademicCourse(String courseID,String courseName, int duration, String level,
String credit, int numberOfAssessments)
    {
        //initalizing ivars to given parameters and values
        super(courseID, courseName, duration);
        this.level = level;
        this.numberOfAssessments = numberOfAssessments;
        this.credit = credit;
        this.lecturerName = "";
        this.startingDate = "";
        this.completionDate = "";
        this.isRegistered = false;
    }

    //Use of Getter to let the method to be accessed
    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;
  }

  //Use of setter to set the new lecturer name
  public void setLecturerName(String lectureName)
  {
      this.lecturerName = lectureName;
  }
```

```
 //Use of setter to set the number of assessments
   public void setNumberOfAssessments(int numberOfAssessments )
   {
       this.numberOfAssessments = numberOfAssessments;
   }


   //register method starts here
   public void register(String courseLeader, String lecturerName, String startingDate,
String completionDate)
   {
      if (getIsRegistered())
      {
          System.out.println("The academic course is already registered");


      }


      else
      {
        super.setCourseLeader(courseLeader);//Calling parent method setCourseLeader
and assigning given courseLeader as paremeter
        this.lecturerName = lecturerName;
        this.startingDate = startingDate;
        this.completionDate = completionDate;
        this.isRegistered = true;
      }
   }
```

```
 //display method stsrts here
    public void display()
    {
        super.display();//Calling display method of parent class
        String AOutput;//AOutput stands for AcademicCourse Output
        AOutput = "The Lecture Name is "+lecturerName+", its Level and credit is
"+level+", "+credit+" respectivly. the starting and compleation Date"+"\n"+ " is
"+startingDate+", "+completionDate+" respectivly. Lastly, number of assessments are
"+numberOfAssessments;
        if(getIsRegistered())
        {
          System.out.println(AOutput);
        }
        else
        {
            System.out.println("Academic coures in not registered");
        }

    }
}
```

```
package Programming;


public class NonAcademicCourse extends Course
{
    //decleration of ivars
    private String instructorName;
    private String startingDate;
    private String completionDate;
    private String examDate;
    private String prerequisite;
    private boolean isRegistered;
    private boolean isRemoved;

    //Creation of Constructor class
    public NonAcademicCourse(String courseID,String courseName, int duration, String
prerequisite)
    {
        //initalizing ivars to given parameters and values
        super(courseID, courseName, duration);
        this.prerequisite = prerequisite;
        this.startingDate = "";
        this.completionDate = "";
        this.examDate = "";
        this.isRegistered = false;
        this.isRemoved = false;
    }

    //Use of Getter to let the method to be accessed
    public String getPrerequisite()
```

```
{
    return this.prerequisite;
}
public String getInstrutorName()
{
    return this.instructorName;
}
public String getStartDate()
{
    return this.startingDate;
}
 public String getCompletionDate()
{
    return this.completionDate;
}
 public String getExamDate()
{
    return this.examDate;
}
public boolean getIsRegistered()
{
    return this.isRegistered;
}
public boolean getIsRemoved()
{
    return this.isRemoved;
}
```

```
    //Use of setter to set the new instructor name
    public void setInstructorName(String instructorName)
    {
        if(getIsRegistered())
        {
            System.out.println("your Instructor Name is alredy set as
"+this.instructorName+" and it cannot be changed");


        }
        else
        {
          this.instructorName = instructorName;
        }
    }


    //register method
    public void register(String courseLeader, String instructorName, String startingDate,
String completionDate, String examDate)
    {
        if (getIsRegistered())
        {
            System.out.println("Your Course is already registered");
        }
        else
        {
            super.setCourseLeader(courseLeader);
            this.setInstructorName(instructorName);
            this.startingDate = startingDate;
            this.completionDate = completionDate;
```

```
            this.examDate = examDate;
            this.isRegistered = true;
            this.isRemoved = false;
        }
    }

    //remove method
    public void remove()
    {
        if (getIsRemoved())
        {
            System.out.println("The course is alredy removed");
        }
        else
        {
            super.setCourseLeader("");//Calling parent method setCourseLeader and
assigning empty string as a parameter
            this.instructorName = "";
            this.startingDate = "";
            this.completionDate = "";
            this.examDate = "";
            this.isRegistered = false;
            this.isRemoved = true;
        }
    }

    //display method
    public void display()
    {
```

```
        super.display();//Calling display method of parent class
        String NOutput;//NOutput stands for NonAcademicCourse Output
        NOutput = "The Instructor Name is "+instructorName+", its starting, compeleation
and exam date"+"\n"+"is "+startingDate+", "+completionDate+" and "+examDate+"
respectively.";
        if (getIsRegistered())
        {
            System.out.println(NOutput);
        }
        else
        {
            System.out.println("Non Academic coures in not registered");
        }
        if(getIsRemoved())
        {
            System.out.println("The non academic course is removed");
        }
    }
}
```

10. References

Alex. (2019, February 1). *LearnCpp.com*. Retrieved from 3.1 — Syntax and semantic errors: https://www.learncpp.com/cpp-tutorial/syntax-and-semantic-errors/#:~:text=A%20semantic%20error%20occurs%20when,2

BlueJ. (n.d.). *About BlueJ*. Retrieved from BlueJ: https://www.bluej.org/about.html

Christensson, P. (2012, April 27). *TechTerms*. Retrieved from Syntax Error Defination: https://techterms.com/definition/syntax_error

Christensson, P. (2012, April 27). *TechTerms*. Retrieved from Logic Error Defination: https://techterms.com/definition/logic_error

Guru99. (2021, March 30). *Guru99*. Retrieved from What is Java? Definition, Meaning & Features of Java Platforms: https://www.guru99.com/java-platform.html

Nishadha. (2020, December 1). *creately*. Retrieved from UML Class Diagram Relationships Explained with Examples: https://creately.com/blog/diagrams/class-diagram-relationships/

Theprogrammedwords. (2021, February 2). *Geeksofgeeks*. Retrieved from How to write a Pseudo Code?: https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/