



 slington college  
(इरिलइटन कलेज)

**Module Code & Module Title**

**CC4001NI Programming**

**COURSEWORK-2**

**Assessment Weightage & Type**

**30% Individual Coursework**

**Semester and year**

**Spring 2021**

**Student Name: Kishor Shrestha**

**Group: C6 (Computing)**

**London Met ID: 20048913**

**College ID: NP01CP4S210161**

**Assignment Due Date: 2021/08/17**

**Assignment Submission Date: 2021/08/17**

*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

1.0 Introduction .....	7
1.1 Introduction of JAVA .....	7
1.2 Abstract class and abstract method .....	8
1.3 Graphical User Interface (GUI): .....	9
1.4 Java Swing: .....	10
1.5 Event Handling .....	11
1.6 Array List .....	11
1.7 Aims and Goals: .....	13
1.8 Objectives .....	13
2.0 Class Diagram.....	14
2.1 INGCollege .....	14
3.0 Pseudocode: .....	16
4.0 Description of Method .....	24
4.1 public INGCollage() .....	24
4.2 Public void Academic Add() .....	25
4.3 Public void Academic Display().....	26
4.4 Public void Academic Registered() .....	26
4.5 Public void Academic Clear() .....	27
4.6 Public void Non Academic Add() .....	27
4.7 Public void Non Academic Display() .....	28
4.8 Public void Non Academic Regsitered() .....	28
4.9 Public void Non Academic Clear() .....	29
4.10 Public void Non Academic Removed() .....	29
4.11 public static void main{String[] args) .....	30

5.0 Testing .....	31
5.1 Test 1.0.....	31
5.2 Test 2.0 .....	33
5.2.1 Test 2.1.....	33
5.2.2 Test 2.2.....	34
5.2.3 Test 2.3.....	35
5.2.4 Test 2.4.....	36
5.2.5 Test 2.5.....	38
5.3 Test 3.0.....	38
5.3.1 Addition of duplicate courseID.....	38
5.3.2 Registration of previous registered value .....	39
5.3.3 Removing the nonacademic course that is already registered .....	39
6.0 ERROR DECTION AND CORRECTION.....	40
6.1 Syntax error .....	40
6.2 Semantic error .....	41
6.3 Logical error.....	44
7.0 Conclusion .....	46
8.0 References.....	47
9.0 Appendix .....	48

## Table of Figure

Figure 1: Java .....	8
Figure 2: GUI (Graphical user interface) .....	9
Figure 3: Java swing .....	10
Figure 4: Array list .....	12
Figure 5: Class diagram of classes .....	14
Figure 6: Class diagram .....	15
Figure 7: Public ING College() .....	25
Figure 8: Public void Academic Add() .....	25
Figure 9: Public void Academic Display() .....	26
Figure 10: Public void Academic Registered() .....	26
Figure 11: Public void Academic Clear() .....	27
Figure 12: Public void Non Academic Add() .....	27
Figure 13: Public void Non Academic Display() .....	28
Figure 14: Public void Non Academic Regsitered() .....	28
Figure 15: Public void Non Academic Clear() .....	29
Figure 16: Public void Non Academic Removed() .....	30
Figure 17: public static void main{String[] args) .....	30
Figure 18: Compiling data from cmd .....	32
Figure 19: Result of testing from cmd.....	32
Figure 20: GUI Result of Addition of Data of Academic course.....	34
Figure 21: GUI Result of Addition of Data of Non Academic course .....	35
Figure 22: GUI Result of Registration of Data in Academic course.....	36
Figure 23: GUI Result of Registration of Data in Non Academic course .....	37
Figure 24: Removing of data from Non Academic course .....	38
Figure 25 : Test to check the message when Same ID is Added .....	38
Figure 26: Test to check the message when Same value are Added .....	39

Figure 27: Removing the nonacademic course that is already registered .....	39
Figure 28: Detection of syntax error .....	40
Figure 29: Correction of Syntax error .....	41
Figure 30: Detection of semantic error .....	42
Figure 31: Correction of semantic error .....	43
Figure 32: Detection of logical error .....	44
Figure 33: Correction of logical error .....	45

## Table of Tables`

Table 1: Testing whether program compile and run using command prompt.....	31
Table 2: Addition of data in Academic course .....	33
Table 3: Addition of data in Non Academic course.....	34
<i>Table 4: Registration of data in Academic course .....</i>	<i>35</i>
<i>Table 5: Registration of data in Non Academic course.....</i>	<i>37</i>

## 1.0 Introduction

The coursework provided by our college focuses on the topic of graphical user interface (GUI) from the programming module. According to the assignment, we must develop a program that includes inheritance and the use of interfaces by establishing a new class called INGCollege. In this assignment, the graphical user interface (GUI) is designed during the first part of the class. The coursework consists of four classes in total which are (Course, Academic Course, NonAcademic Course).

In this coursework, all programs or codes are written in the INGCollege class by inheriting methods from other classes and using interfaces. All classes have methods and attributes in order to complete coursework, we must first understand instance of, inheritance, and interface.

## 1.1 Introduction of JAVA

Java is an object-oriented and yet sophisticated language of programming and similar to C++ in many areas. Java was founded in 1991 by Sun Microsystems, Inc. It was designed at Sun Microsystems, Inc. by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan. It was created to offer an independent language for platform programming. You can also read Java Programming with several java examples Java includes some of the strong qualities such as code reusability, emphasis on data instead of procedure, and data is also concealed, so that external functions cannot be accessible in which an object communicates with each other via functions, and new data may be added. from this website. It's a whole java education for beginners to experienced java programming (Jain).



Figure 1: Java

## 1.2 Abstract class and abstract method

In JAVA abstract is a keyword which is categorized into abstract class as well as abstract method, whereas abstract class is a limited class which could even be utilized to generate objects and should be inherited from these other classes in order to be accessed. So, an abstract method is a procedure that is accessed through an abstract class or an abstract modifier, and it lacks a body but can be supplied by subclasses. Abstraction is the technique of concealing actual facts while displaying merely capability to the user. Multiple inheritance is not supported in abstract classes.



### 1.3 Graphical User Interface (GUI):

GUI is a graphical interface that visually illustrates the user's communication to facilitate interaction with the machine. Graphical user interface meaning graphical user interface. It's the normal user interface with graphics such as buttons and icons and the interaction with these icons can be achieved through communication, rather than by the traditional communication based on the text or on a command.

The use of a pointer to interface with various graphical icons that are visually attractive. Abstraction is a fundamental idea employed in a GUI system. The clicking of the symbol can be used by users to activate a number of actions. An program or functionality is usually started. Then the user must supply the input or tasks to produce the required machine activity. Actually, the GUI translates user-language, which includes simple one-line commands, one-click and a double click on it. Machine recognizes the language of the machine and so the machine responds to the task initiated by the user translating into language and using GUI (pedamkar).

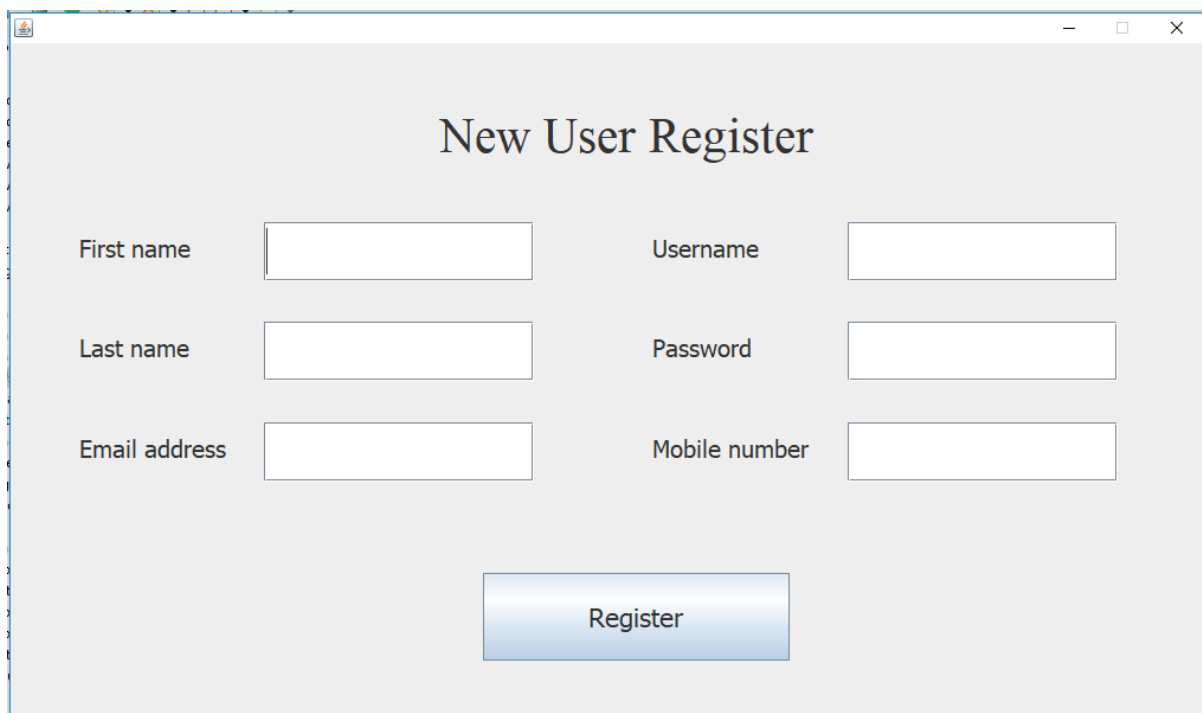


Figure 2: GUI (Graphical user interface)

## 1.4 Java Swing:

Swing Framework contains a set of classes which provides GUI components that are stronger and more flexible than AWT. Swing gives the current Java GUI look and feel. The Swing library is an official Sun Micro-Systems Java GUI tool package. It is used for the creation of graphical Java user interface.

Without any changes to the application code, Swing can customize the look and feel of any of the component. It also features a plugging look and feel functionality that permits the imitation of native components while still benefiting from the freedom of the platform. This specific feature facilitates the writing of Swing applications and separates them from other native programs (kant).



The image shows a Java Swing window titled "New User Register". The window has a light gray background and a standard Windows-style title bar with minimize, maximize, and close buttons. The title "New User Register" is centered at the top in a large, dark blue serif font. Below the title, there are six input fields arranged in two columns. The left column contains "First name", "Last name", and "Email address". The right column contains "Username", "Password", and "Mobile number". Each label is followed by a white rectangular text box with a thin blue border. At the bottom center of the window is a blue rectangular button with the word "Register" in white text.

Figure 3: Java swing

## 1.5 Event Handling

Events are signals that report changes in a browser or operating system environment within the browser window. Programmers can develop event manager code that works when an event fires, so web pages can react to changes. Events for the JavaScript objects which issue them are detailed in and/or below the pages. For example, in the Window and Document section to locate events fired from the browser window or the present document. The event reference can be used to figure out which objects fire events JavaScript objects for specific APIs, such animation, media etc.

When an action is initiated by setting it to a relevant on event property of the target item, or by registering the handler as an element listener using the `addEventListener()` method, an event handler code can be executed. An object matching the event interface will be sent to the handler in any scenario (or a derived interface). The key distinction is that you can add (or delete) numerous event handlers using the event listener methods (whitman).

## 1.6 Array List

`ArrayList` belongs to the collection framework and is included in the package `Java.util`. It gives us dynamic Java arrays. Although it may be slower than normal arrays, it might be advantageous in applications that require a lot of manipulation. This class is available in the packet `java.util`.

1. Adding elements We are able to utilize the `add()` method to add an element to an `ArrayList`. This procedure is overloaded to conduct several operations based on various parameters. They are as follows:
  - `Add(object)` This technique is being used to add a layer at the end of the arraylist.

- Index add(int index, Object): This technique is used to add a layer in the array list to a particular (jain).

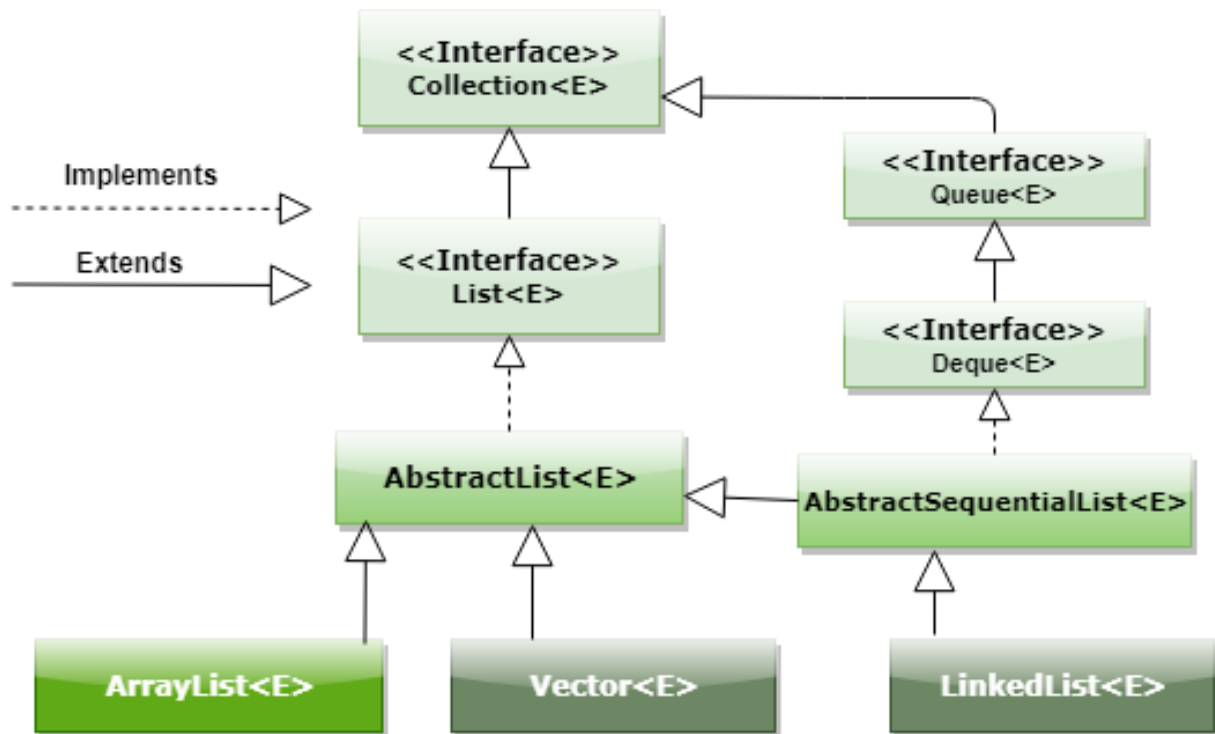


Figure 4: Array list

## 1.7 Aims and Goals:

The aims and goals of utilizing the Java programming language to create apps for the graphical user interface in this course (GUI). To create a graphical user interface, we'll need to use a lot of Java Swing components. To discover the proper location or distance between these components GUI, the x, y-axis, length, and breadth of each component must be defined. One can improve the efficiency and accessibility of digital equipment by allowing users to engage directly with devices and complete specific tasks by manipulating features like as icons and scroll bars. As a result, the course's primary goal is to learn how to program in Java and how to use its Swings components to create a graphical user interface.

## 1.8 Objectives

1. To make use of the human-computer interaction as seamless as possible.
2. To allow cut-and-paste or 'drag-and-drop' information sharing between applications
3. To gain an understanding of the systematic method of creating test cases in order to evaluate the system's functioning and design components.
4. To learn how to transfer files using drag-and-drop and to use familiar icons, such as a trash bin for deleted files, to create an environment where computer activities are straightforward and simple to grasp without any prior experience or knowledge of computing machinery or languages.

## 2.0 Class Diagram

A class diagram is divided into three components. As a result, a separate table diagram is created for each element to make it easier to interpret. The first section provides the class name, the second part contains class attributes, and the third section carries methods or class tasks that are connected to two classes. This coursework is about the first coursework that is mostly concerned developing the GUI for the previous project. In terms of class diagrams, the earlier project included a three-class diagram (Course, Academic Course, Non Academic course). As a result, the ING College class diagram is drawn as follows: -

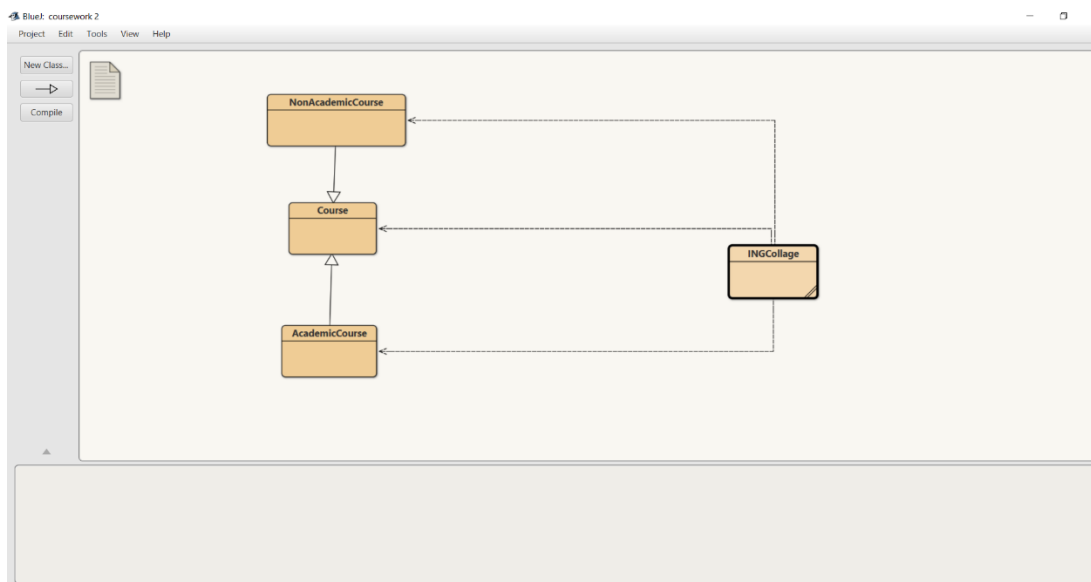


Figure 5: Class diagram of classes

## 2.1 INGCollege

ING College is the major class in which all GUI activity is done with an action listener, which performs the action after it is called. In the ING College class diagram, the table contains the class, attributes, and methods, whereas the attributes or variables include string and integer data types, as seen in the following figure:-



Figure 6: Class diagram

### 3.0 Pseudocode:

Pseudocode is an in-depth yet accessible explanation, expressed in a language formally natural instead of just a language of programming. The pseudocode is an allegorical depiction. Occasionally pseudo codes are used to develop a program as a detailed step. It enables designers or managers to explain the layout in great depth and offers programmers with a detail pattern in a particular programming language for the next stage of writing code (Heller).

**INITIALIZE** frame as a panel

**INITIALIZE** lbp1 CourseID, lbp1CourseName, lbp1 Duration, lbp1  
NumberOfAssessment, lbp1 LecturerName, lbp1 CourseLeader, lbp1v Level,  
lbp1 Credit, lbp1 Course, lbp1 StartDate, lbp1 CompletionDate, lbp2 CourseID,  
lbp2 CourseName ,lbp2 Duration, lbp2 Prerequisite, lbp2 Course, lbp2  
CourseLeader, lbp2 InstructorName, lbp2 Startdate, lbp2 Completiondate, lbp2  
Examdate as a JLabel.

**INITIALIZE** txtP1CourseID, txtP1CourseName, txtP1Duration,  
txtP1NumberOfAssessment, txtP1LecturerName, txtP1CourseLeader, txtP1Level,  
txtP1Credit, txtP1Startdate, txtP1CompletionDate, txtP2CourseID,  
txtP2CourseIDCheck, txtP2CourseName, txtP2Duration, txtP2Prerequisite,  
txtP2Course, txtP2CourseLeader, txtP2InstructorName, txtP2StartDate,  
txtP2Completiondate, txtP2ExamDate as a JTextField.

**INITIALIZE** btn1AddAcademic, btn1DisplayAcademic, btn1RegisterAcademic,  
btn1ClearAcademic, btnAddNonAcademic, btnDisplayNonAcademic,  
btnRegisterNonAcademic, btnClearNonAcademic, btnRemovedNonAcademic as  
a JButton.



**INITIALIZE** ING College as a frame

**SET** layout of panel null

**SET** size (1250, 580)

**SET** Resizable (true)

**SET** frame as a Default Close operation (JF. Frame. EXIT\_ON\_Close)

**Pseudo code of Academic course**

**INITIALIZE** Academic as an panel1

**SET** Bounds to (10, 70, 610, 600)

**SET** layout to (null)

**FUNCTION** get CourseID()

Return text value of CourseID()

**END**

**FUNCTION** get CourseID()

Return text value of CourseID

**END**

**FUNCTION** get Duration()

**SET** Duration as as the value from txtDuration

**SET** Duration as = -1

**TRY**

Convert string Duration as the numeric form and store in numeric duration

**CATCH** exception show message" Incorrect value for number of Duration.\nPlease add numeric value", "Invalid Input", 0

**RETURN****FUNCTION** get Number of Assessment()**SET** Number of Assessment as the value of txtNumber of Assessment**Set** Number of Assessment = -1**Try**

Convert Number of Assessment String into numeric form and store in  
Number of

Assessment

**CATCH** Exception

Show message" Please enter the numeric values for  
numberof assessment","Invalid Data",2)

**RETURN****END****FUNCTION** get Level()

Return text value of Level

**END****FUNCTION** get Credit()

Return text value of Credit

**END****FUNCTION** get Lecturer Name()

Return text value of Lecturer Name

**END**

**FUNCTION** get Course Leader()

Return text value of Course leader

**END**

**FUNCTION** get Start Date()

Return text value of StartDate

**END**

**FUNCTION** get CompletionDate()

Return text value of CompletionDate

**END**

**FUNCTION** get Academic StartDate()

Return text value of Academic StartDate

**END**

**SET** lbp1 Course ID text as CourseID

**SET** lbp1 CourseID bounds(15, 30, 75, 25 )

**ADD** lbp1 CourseID on panel

**SET** lbp1 Course Name text as CourseID

**SET** lbp1 Course Name bounds(30, 20, 100, 25 )

**ADD** lbp1 Course Name on pannel

**SET** lbp1 Course Name text as CourseName

**SET** lbp1 Course Name bounds (280, 30, 100, 25);

**ADD** lbp1 Course Name on panel

**SET** lbp1 Duration text as Duration

**SET** lbp1 Duration bounds (15, 80, 75, 25);

**ADD** lbp1 Duration on panel

**SET** lbp1 Number of Assessment text as Number of Assessment

**SET** lbp1 Number of Assessment bounds (280, 80, 150, 25)

**ADD** lbp1 Number of Assessment on panel

**SET** lbp1 Level text as Level

**SET** lbp1 Level bounds (15, 130, 150, 25)

**ADD** lbp1 Level on panel

**SET** lbp1 Credit text as Credit

**SET** lbp1 Credit bounds(15, 110, 150, 25)

**ADD** lbp1 Credit on panel

**SET** lbp1 lecturerName text as lecturerName

**SET** lbp1 lecturerName bounds (15, 280, 150, 25)

**ADD** lbp1 lecturerName on panel

**SET** lbp1 Course leader as Course leader

**SET** lbp1 Course leader bounds (330, 230, 150, 25)

**ADD** lbp1 Course leader on panel

**SET** lbp1 Start Date text as Start Date

**SET** lbp1 as Start Date bounds (330, 270, 150, 25)

**ADD** lbp1 as Start Date on panel

Pseudocode of Non Academic Course

**SET** lbp1 CourseID text as CourseID

**SET** lbp1 CourseID bounds(15, 20, 75, 25)

**ADD** lbp1 as CourseID on panel

**SET** lbp1 CourseName text as CourseName

**SET** lbp1 CourseName bounds(300, 30, 100, 25)

**ADD** lbp1 as CourseName on panel

**SET** lbp1 Duration text as Duration

**SET** lbp1 Duration bounds(15, 80, 75, 25)

**ADD** lbp1 Durationon panel

**SET** lbp1 prerequisite text as prerequisite

**SET** lbp1 prerequisite (300, 80, 75, 25)

**ADD** lbp1 prerequisite panel

**SET** lbp1 Course leader as Course leader

**SET** lbp1 Course leader bounds (290, 200, 150, 25)

**ADD** lbp1 Course leader on panel

**SET** lbp1 InstructorName as InstructorName

**SET** lbp1 InstructorName bounds (15, 240, 100, 25)

**ADD** lbp1 InstructorName on panel

**SET** lbp1 Start Date text as Start Date

**SET** lbp1 as Start Date bounds (290, 240, 150, 25)

**ADD** lbp1 as Start Date on panel

**SET** lbp1 Exam Date text as Start Date

**SET** lbp1 as Exam Date bounds (390, 240, 150, 25)

**ADD** lbp1 as Exam Date on panel

**SET** lbp1 Completion Date text as Completion Date

**SET** lbp1 as Completion Date bounds (15, 280, 100, 25)

**ADD** lbp1 as Completion Date on panel

## 4.0 Description of Method

A method is a collection of code created to do a specified task that may or may not return a value. The following are the four basic components of a method description: -

- The type of method returns
- The name of method
- A list of parameters
- The body of the method

It consists of an access control modifier that shows the visibility or accessibility of the methods to other subclasses. System enables us to reuse the code without retyping the code that lets us save time (Anon.).

### 4.1 public INGCollage()

This method makes extensive usage of Java Swing components including JFrame, JLabel, JTextField, and JButton. Initially, as a window, an ING Collage for GUI frame is built (Graphical User Interface). Various elements should be incorporated in our GUI, as per the question. JLabel is often used in this for component labeling. The data is obtained using JTextField. Clients enter values, then JButton is utilized to create buttons. And for proper location, the appropriate spacing between every component, as well as the boundaries of the x-axis, y-axis, length, and height, are defined. There are 9 buttons for adding, clearing, displaying, clearing, and removing. The crew performs several activities that are carried out through the use of decentralized event management.



```

import javax.swing.*;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;
import javax.swing.BorderFactory;
import javax.swing.border.Border;
import java.awt.FlowLayout;

public class INGCollage {
    private ArrayList<Course> list = new ArrayList<>();
    private JFrame frame;
    private JPanel panel1, panel2;
    private JLabel lbP1CourseID;
    private JLabel lbP1CourseName;
    private JLabel lbP1Duration;
    private JLabel lbP1NumberofAssessment;
    private JLabel lbP1LecturareName;
    private JLabel lbP1CourseLeader;
    private JLabel lbP1Level;
    private JLabel lbP1Credit;
    private JLabel lbP1Course;
    private JLabel lbP1StartDate;
    private JLabel lbP1CompletionDate;

```

Figure 7: Public INGC College()

## 4.2 Public void Academic Add()

This button executes the Academic Course Add() function, that inserts the CourseID, CourseName, Duration, Level, Credit, and NumberofAssessments to the arraylist.

```

btn1AddAcademic.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        addAcademicCourse();
    }
});

```

Figure 8: Public void Academic Add()

### 4.3 Public void Academic Display()

This button is often used to run the function Academic Course Display() to show each of the information that are saved in the array list whenever the CourseID, CourseName, Duration, Level, Credit, NumberofAssessment, Completion date, LecturerName, CourseLeader, Start Date are added using the Register method.

```
btn1DisplayAcademic.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent ae) {  
        displayAcademicCourse();  
    }  
});
```

Figure 9: Public void Academic Display()

### 4.4 Public void Academic Registered()

This button is needed to execute the method Academic Course Registered(), that has entered all academic course information to the array collection.

```
JButton Register = new JButton("Register ");  
panel1.add(Register);  
Register.setBounds(280, 320, 95, 30);  
Register.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent ae) {  
        registerAcademicCourse();  
    }  
});
```

Figure 10: Public void Academic Registered()

## 4.5 Public void Academic Clear()

This button is often used to execute the method Academic Course Clear(), that deletes all data entered by the user into Academic Course.

```
btn1ClearAcademic = new JButton("Clear");  
panel1.add(btn1ClearAcademic);  
btn1ClearAcademic.setBounds(480, 340, 95, 30);  
  
btn1ClearAcademic.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent ae) {  
        clearAcademicCourse();  
    }  
});
```

Figure 11: Public void Academic Clear()

## 4.6 Public void Non Academic Add()

This button executes the Non-Academic Course Add() function, that inserts the CourseID, CourseName, requirement, and Duration towards the arraylist.

```
btnClearNonAcademic.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent ae) {  
        clearNonAcademicCourse();  
    }  
});
```

Figure 12: Public void Non Academic Add()

## 4.7 Public void Non Academic Display()

This button is have been using to approach the method Academic Course Display() to represent all of the information that are collected in the array list whenever the CourseID, CourseName, prerequisite, Duration, Credit,, Completion date, LecturerName, CourseLeader, CourseID,Start Date, InstructorName,Exam Date, Completion Date, is added.

```
public void displayNonAcademicCourse() {
    for (Course dev : list) {
        if (dev instanceof NonAcademicCourse) {
            NonAcademicCourse AB = (NonAcademicCourse) dev;
            AB.display();
        }
    }
}
```

Figure 13: Public void Non Academic Display()

## 4.8 Public void Non Academic Regsitered()

This button is often used to execute the method Non-Academic Course Registered(), that has updated every Non-academic course data to the array list.

```
if (ac.getRegistered() == true) {
    JOptionPane.showMessageDialog(frame, "The course is already register.", "register",1);
    return;
} else {
    ac.register(courseLeader,lectureName,startingDate,completionDate);
    String message = "CourseID : " + courseID + "\nCourseLeader : " + courseLeader
        + "\nLeactureName : " + lectureName + "\nStartingDate : " + startingDate
        + "\nCompletionDate" + completionDate;
    JOptionPane.showMessageDialog(frame, message, "Data Added", 1);
    return;
}
```

Figure 14: Public void Non Academic Regsitered()

## 4.9 Public void Non Academic Clear()

This button is often used to execute the method Non-Academic Course Clear(), that clears any data entered from the client into Non-Academic Course.

```
public void clearNonAcademicCourse(){
    txtP2CourseID.setText("");
    txtP2CourseName.setText("");
    txtP2InstructorName.setText("");
    txtP2CourseID.setText("");
    txtP2Duration.setText("");
    txtP2Prerequisite.setText("");
    txtP2CourseLeader.setText("");
    txtP2StartDate.setText("");
    txtP2Completiondate.setText("");
    txtP2ExamDate.setText("");
}
```

Figure 15: Public void Non Academic Clear()

## 4.10 Public void Non Academic Removed()

This button is often used to execute the method Non-Academic course(), that deletes all of information recorded in the array list whenever a Non-Academic Course is entered and registered

```

public void RemovedNonAcademicCourse() {
    String id = this.getNonAcademicCourseID();
    if(id.equals("")){
        JOptionPane.showMessageDialog(frame,"Please add ID for which you want course to be removed.", "Remove",1);
        return;
    }else{
        if(list.isEmpty()){
            JOptionPane.showMessageDialog(frame," Please Add and register the course inorder to removed.", "Warning",1);
            return;
        }else{
            for(Course cou:list){
                if(cou.getCourseID().equals(id)){
                    if(cou instanceof NonAcademicCourse){
                        NonAcademicCourse nac = (NonAcademicCourse) cou;
                        if(nac.getRemoved() == false){
                            nac.Remove();
                            JOptionPane.showMessageDialog(frame," remove sucessfull", "remove",1);
                        }else{

```

Figure 16: Public void Non Academic Removed()

#### 4.11 public static void main{String[] args)

This is the major function of a INGCollage class which delivers an INGCClass object. The complete code is executed inside this method.

```

    }

    public static void main(String[] args) {
        new INGCollage().frame.setVisible(true);
    }
}

```

Figure 17: public static void main{String[] args)

## 5.0 Testing

After the execution of the program using BlueJ, we must test the program to ensure the success of the program which is built. The purpose of program testing is to ensure that now the software is suitable for usage.

In this program, I tested the INGCollage class and methods described below:

### 5.1 Test 1.0

Objective	Compiling and running program using command prompt
Action	Open command prompt. The folder containing the program is opened. Enter <code>javac INGCollage.java</code> Enter <code>java ING Collage</code>
Expected Output	Program should be compiled and should run using command prompt.
Actual result	The program has successfully compiled and run using command prompt
Conclusion	The test has been done successfully

Table 1: Testing whether program compile and run using command prompt

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

F:\JAVA GUI PROJECT\INGCollage>javac INGCollage.java

F:\JAVA GUI PROJECT\INGCollage>java INGCollage
```

Figure 18: Compiling data from cmd

The screenshot shows the INGCollage application window with the title bar 'INGCollage'. The main content area is titled 'Registration form' and is divided into two panels: 'Academic Course' and 'NonAcademic Course'.

**Academic Course Panel:**

- Course ID:
- CourseName:
- Duration:
- NumberofAssessment:
- Level:
- Credit:
- Completion date:
- CourseLeader:
- LecturerName:
- Start Date:
- Buttons: Add, Register, Display, Clear

**NonAcademic Course Panel:**

- CourseID:
- CourseName:
- Duration:
- prerequisite:
- CourseID:
- CourseLeader:
- InstructorName:
- Start Date:
- Completion Date:
- Exam Date:
- Buttons: Add, Register, Display, Clear, Remove

Figure 19: Result of testing from cmd



## 5.2 Test 2.0

### 5.2.1 Test 2.1

Adding data in Academic Course:

Objective	Adding data in Academic Course:
Action	Adding data in Academic Course: CourseID = 1 CourseName = Programming Duration = 5 Number Of Assessment = 14 Credit = 10 Level = B
Expected Output	Execution of data in program is successful.
Actual result	Addition of data is successful.
Conclusion	Testing is successful.

Table 2: Addition of data in Academic course

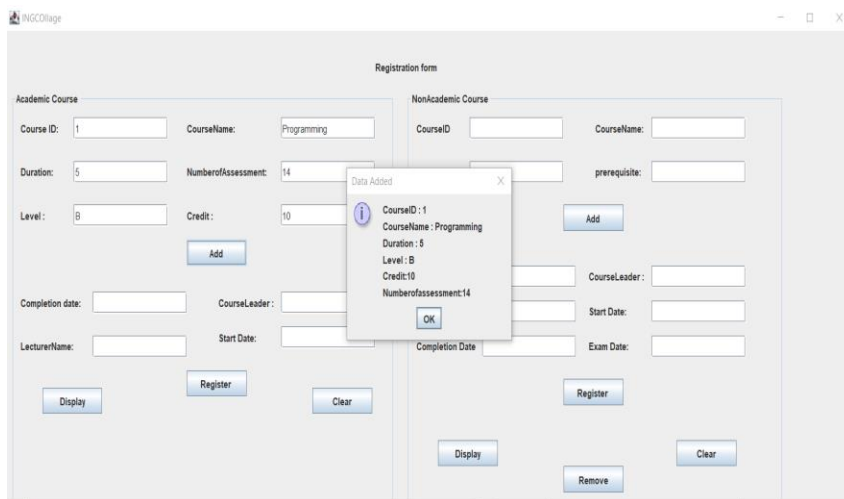


Figure 20: GUI Result of Addition of Data of Academic course

## 5.2.2 Test 2.2

Adding data in Non Academic Course:

Objective	Adding data in Non Academic Course
Action	Adding data in Non Academic Course Course ID = ABC Course Name = Java Prerequisite = 18 Duration = 14
Expected Output	Execution of data in program is successful.
Actual result	Addition of data is successful.
Conclusion	Testing is successful.

Table 3: Addition of data in Non Academic course

The screenshot shows a 'Registration form' window with two main sections: 'Academic Course' and 'NonAcademic Course'. The 'NonAcademic Course' section is active, showing fields for CourseID (ABC), CourseName (Java), Duration (14), prerequisite (18), CourseLeader (Zenith), Start Date (20), Completion Date (30), and Exam Date (22). A 'Data Added' dialog box is overlaid on the form, displaying the following information: CourseID: 4, CourseName: Java, Duration: 14, and Prerequisite: 18. The dialog box has an 'OK' button.

Figure 21: GUI Result of Addition of Data of Non Academic course

### 5.2.3 Test 2.3

Registration of data in Academic course

Objective	Registration of data in Academic course
Action	Registration of data in Academic course Lecturer Name = Avishek Course Leader = Avishek Completion Date = 21 Start date =20
Expected Result	Execution of data in program is successful.
Actual Result	Addition of data is successful.
Conclusion	Testing is successful

Table 4: Registration of data in Academic course

The screenshot shows a web application window titled 'Registration form'. It is divided into two main sections: 'Academic Course' and 'NonAcademic Course'. The 'Academic Course' section is active and contains the following fields: Course ID (1), CourseName (Programming), Duration (5), NumberofAssessment (14), Level (B), Credit (10), Completion date (21), CourseLeader (Avishek), and LecturerName (Avishek). A 'Data Added' dialog box is overlaid on the Academic Course section, displaying the following information: CourseID: 1, CourseLeader: Avishek, LeactureName: Avishek, StartingDate: 20, and CompletionDate: 21. The 'NonAcademic Course' section is inactive and contains fields for CourseID, CourseName, Duration, prerequisite, CourseLeader, Start Date, Completion Date, and Exam Date. Buttons for 'Add', 'Register', 'Display', 'Clear', and 'Remove' are visible at the bottom of each section.

Figure 22: GUI Result of Registration of Data in Academic course

## 5.2.4 Test 2.4

Registration of data in Non Academic course

Objective	Registration of data in Non Academic course
Action	Registration of data in Non Academic course CourseLeader = Zenith Course ID =ABC StartDate = 20 Instructor Name = Hari Exam date = 22

	Completion Date = 30
Expected Result	Execution of data in program is successful.
Actual Result	Addition of data is successful.
Conclusion	Testing is successful

Table 5: Registration of data in Non Academic course

The screenshot shows the INGCollage application window with the 'Registration form' for 'NonAcademic Course'. The form contains several input fields and buttons. A 'Data Added' dialog box is open in the center, displaying the following information:

- CourseID : 4
- CourseLeader : Zenith
- InstructorName : Hari
- StartingDate : 20
- CompletionDate: 30
- ExamDate : 22

The dialog box has an 'OK' button. The background form includes fields for 'CourseID', 'CourseName', 'Duration', 'NumberofAssessment', 'Level', 'Credit', 'Completion date', 'CourseLeader', 'LecturerName', 'Start Date', 'Completion Date', and 'Exam Date'. There are also buttons for 'Add', 'Register', 'Display', 'Clear', and 'Remove'.

Figure 23: GUI Result of Registration of Data in Non Academic course

## 5.2.5 Test 2.5

Removing of data from Non Academic course

The screenshot shows a 'Registration form' window with two main sections: 'Academic Course' and 'NonAcademic Course'. The 'Academic Course' section includes fields for Course ID (1), CourseName (Programming), Duration (5), NumberofAssessment (14), Level (B), Credit (10), Completion date (21), CourseLeader (Avishek), and LecturerName (Avishek). The 'NonAcademic Course' section includes fields for CourseID (ABC), CourseName (Java), Duration (14), prerequisite (18), CourseLeader (Zenith), InstructorName (Hari), Start Date (20), Completion Date (30), and Exam Date (22). A 'remove' dialog box is open in the center, displaying the message 'it does not exist.' with an 'OK' button. The 'Add' button is visible in the 'NonAcademic Course' section.

Figure 24: Removing of data from Non Academic course

## 5.3 Test 3.0

### 5.3.1 Addition of duplicate courseID

Adding same data in the form display given result below

The screenshot shows the same 'Registration form' window. In the 'NonAcademic Course' section, the 'Add' button is visible. A 'register' dialog box is open in the center, displaying the message 'The course is already register.' with an 'OK' button. The 'Add' button is visible in the 'NonAcademic Course' section.

Figure 25 : Test to check the message when Same ID is Added

### 5.3.2 Registration of previous registered value

Registering previous data in the form display given result below

The screenshot shows a web application window titled 'INGCOllage' with a 'Registration form'. The form is divided into two main sections: 'Academic Course' and 'NonAcademic Course'. The 'Academic Course' section has fields for Course ID (1), CourseName (Programming), Duration (5), NumberofAssessment (14), Level (B), Credit (10), Completion date (16), CourseLeader (Ayush), LecturerName (Badal), and Start Date (7). The 'NonAcademic Course' section has fields for CourseID (ABC), CourseName (Java), Duration (14), prerequisite (18), CourseLeader (Zenth), InstructorName (Hari), Start Date (20), Completion Date (30), and Exam Date (22). Both sections have 'Add', 'Register', 'Display', and 'Clear' buttons. A modal dialog box with the title 'register' and an information icon is displayed in the center, showing the message 'The course is already register.' with an 'OK' button.

Figure 26: Test to check the message when Same value are Added

### 5.3.3 Removing the nonacademic course that is already registered

Removing previous data in the form display given result below

The screenshot shows the same 'Registration form' as in Figure 26. The 'NonAcademic Course' section now has empty fields for CourseName, Duration, prerequisite, CourseLeader, InstructorName, Start Date, Completion Date, and Exam Date. The 'Academic Course' section remains the same. A modal dialog box with the title 'Remove' and an information icon is displayed in the center, showing the message 'Please add ID for which you want course to be removed.' with an 'OK' button.

Figure 27: Removing the nonacademic course that is already registered

## 6.0 ERROR DETECTION AND CORRECTION

There are 3 types of error which are listed below:

- Syntax error
- Semantic error
- Logical error

### 6.1 Syntax error

The syntax of a programming language is its words and grammar. Programming languages, like English language, have standards regarding writing, punctuation, and grammar. In programming, a syntax error occurs whenever there is a grammatical error. Syntax errors keep programming from compiling unless their syntax is fixed. Syntax error caused by the absence of a semicolon and an additional bracket. Syntax errors include the absence of the above type of bracket at the end of a line, the absence of a class, and so on.

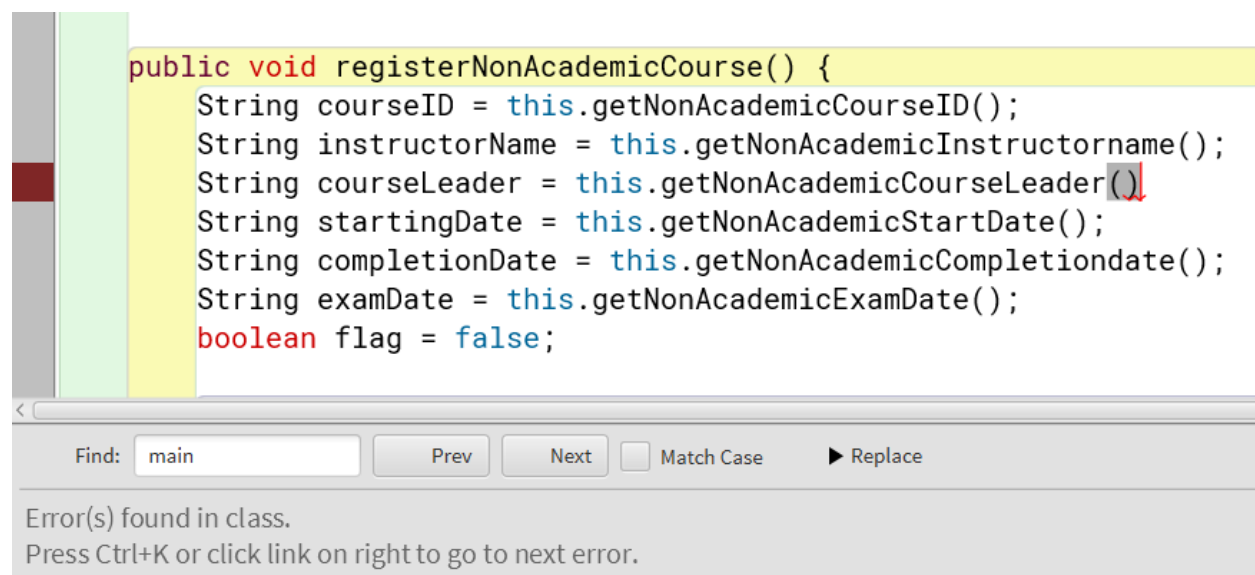
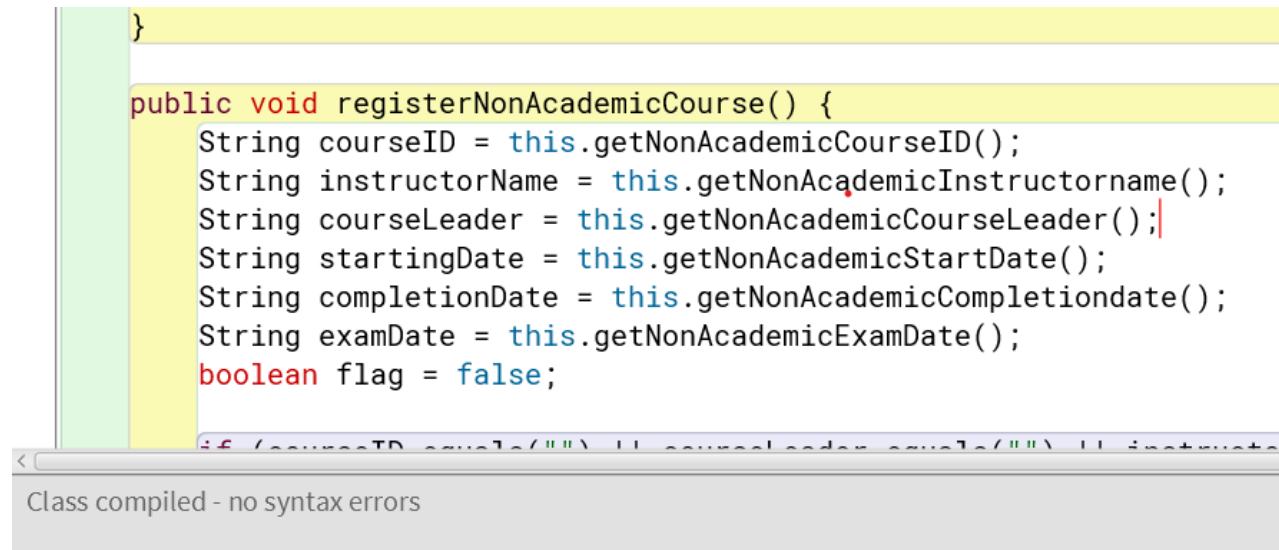


Figure 28: Detection of syntax error



In the above figure, I discovered a syntax mistake: there seems to be a semicolon missing at the ending of the course leader, stopping the program from compiling and displaying error found in class.



```

    }

    public void registerNonAcademicCourse() {
        String courseID = this.getNonAcademicCourseID();
        String instructorName = this.getNonAcademicInstructorname();
        String courseLeader = this.getNonAcademicCourseLeader();
        String startingDate = this.getNonAcademicStartDate();
        String completionDate = this.getNonAcademicCompletiondate();
        String examDate = this.getNonAcademicExamDate();
        boolean flag = false;
    }

```

Class compiled - no syntax errors

Figure 29: Correction of Syntax error

In the above figure, I fixed the syntax error by inserting a semicolon at the ending of the course leader, and the program has now been successfully compiled, and it no longer displays syntax errors and instead displays class created – no syntax errors.

## 6.2 Semantic error

Semantic error is a type of error that occurs most frequently during the semantic analysis process. These mistakes are discovered during the compilation phase. When the incorrect variables is used, the improper operator is used, or the operations are executed in the wrong order, a semantic error occurs. While running the software, it generates an error.

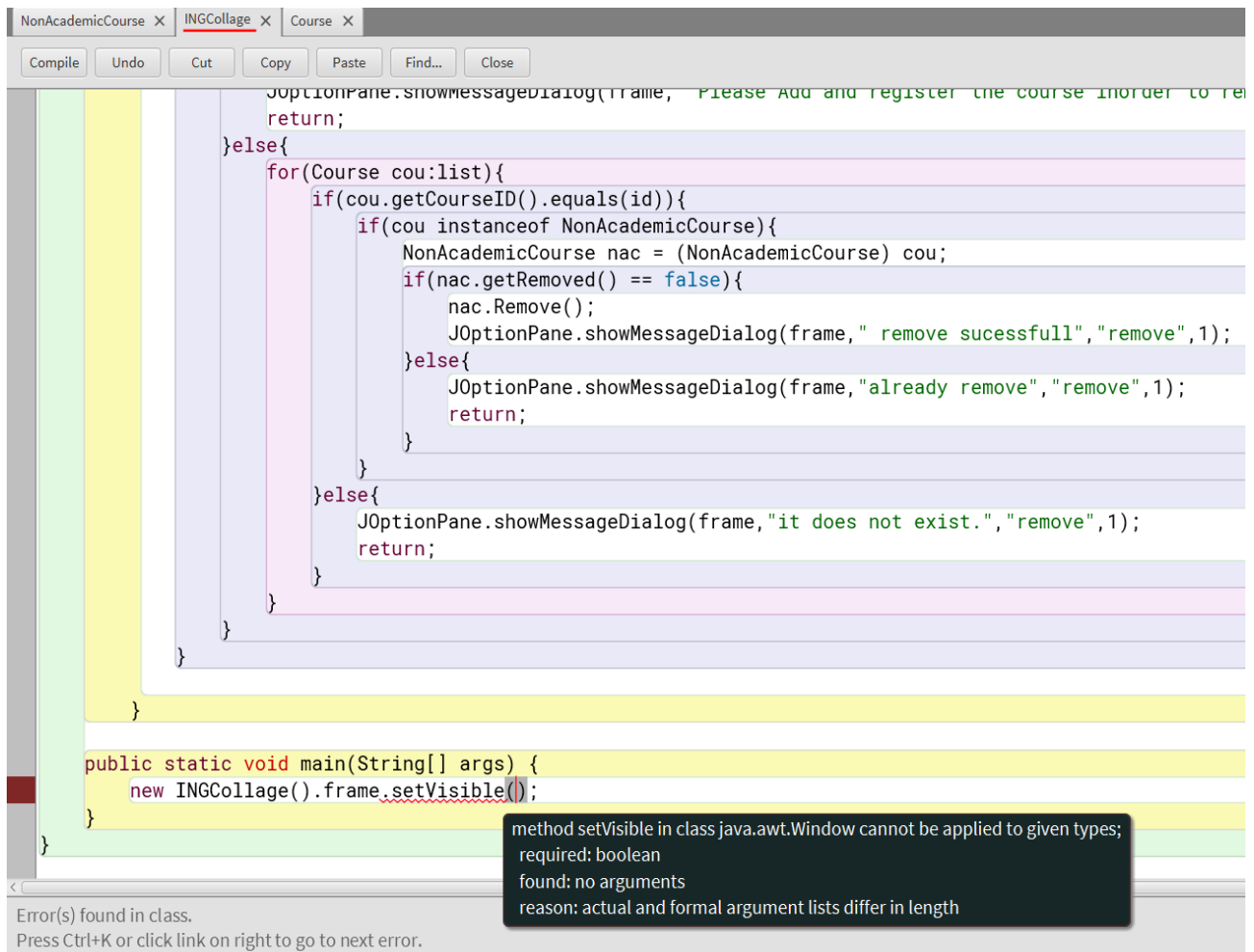


Figure 30: Detection of semantic error

In the above figure I have detect semantic error I have not initialized boolean expression Due to which the program is not compiling and is executing error found in class.

```

NonAcademicCourse x  INGCollage x  Course x
Compile  Undo  Cut  Copy  Paste  Find...  Close

JOptionPane.showMessageDialog(frame, "Please Add and register the course inorder to re
return;
}else{
for(Course cou:list){
if(cou.getCourseID().equals(id)){
if(cou instanceof NonAcademicCourse){
NonAcademicCourse nac = (NonAcademicCourse) cou;
if(nac.getRemoved() == false){
nac.Remove();
JOptionPane.showMessageDialog(frame," remove sucessfull","remove",1);
}else{
JOptionPane.showMessageDialog(frame,"already remove","remove",1);
return;
}
}
}else{
JOptionPane.showMessageDialog(frame,"it does not exist.","remove",1);
return;
}
}
}
}

public static void main(String[] args) {
new INGCollage().frame.setVisible(true);
}
}

Class compiled - no syntax errors

```

Figure 31: Correction of semantic error

In the above figure I have corrected semantic error I have corrected initializing boolean expression while printing in above program and now the program is successfully compiled and it is not showing semantic error and is successfully showing class no syntax error.

## 6.3 Logical error

A logic error occur whenever there is a fault in the logic or structure of a problem. The vast majority of the time, logical errors do not result in a program's failure. Logic errors, and from the other hand, might lead a process to produce unexpected results.

```

JOptionPane.showMessageDialog(frame, "The course is
return;
} else {
    flag = true;
}
}
}
if(flag==true){
    list.add(new NonAcademicCourse(courseID, courseName, duration, pre
String message = "CourseID : " + courseID + "\nCourseName : " +
JOptionPane.showMessageDialog(frame, message, "Data Added", 1);
}
}

public void registerNonAcademicCourse() {
    String courseID = this.getNonAcademicCourseID();
    String instructorName = this.getNonAcademicInstructorname();
    String courseLeader = this.getNonAcademicCourseLeader();
    String startingDate = this.getNonAcademicStartDate();
    String completionDate = this.getNonAcademicCompletiondate();
    String examDate = this.getNonAcademicExamDate();
    boolean flag = true;

    if (courseID.equals("") || courseLeader.equals("") || instructorName
    || completionDate.equals("") || examDate.equals("")) {
        JOptionPane.showMessageDialog(frame, "The fields are empty.\nPl
        2);
    }
}

```

Class compiled - no syntax errors

Figure 32: Detection of logical error

The programming section in the above figure is not compiling due to the present of logical error. Although the code is compiling it is not giving expected output and by checking code I have detected logical error. I have included logic value true inside registered nonacademic method.

```

NonAcademicCourse X  INGCollage X  Course X
Compile  Undo  Cut  Copy  Paste  Find...  Close

        return;
    } else {
        flag = true;
    }
}

}

if(flag==true){
    list.add(new NonAcademicCourse(courseID,courseName,duration,
    String message = "CourseID : " + courseID + "\nCourseName :
    JOptionPane.showMessageDialog(frame, message, "Data Added",

}

}

public void registerNonAcademicCourse() {
    String courseID = this.getNonAcademicCourseID();
    String instructorName = this.getNonAcademicInstructorname();
    String courseLeader = this.getNonAcademicCourseLeader();
    String startingDate = this.getNonAcademicStartDate();
    String completionDate = this.getNonAcademicCompletiondate();
    String examDate = this.getNonAcademicExamDate();
    boolean flag = false;

    if (courseID.equals("") || courseLeader.equals("") || instructor
    || completionDate.equals("")||examDate.equals("")) {
        JOptionPane.showMessageDialog(frame, "The fields are empty.\
        2);
        return;
    }
}

Class compiled - no syntax errors

```

Figure 33: Correction of logical error

The programming section in the above figure is now compiling due to the correction of logical error. The code is compiling and is giving expected output. I have corrected logical error by including logic value false inside registered nonacademic method.

## 7.0 Conclusion

The above project is for designing a graphical user interface (GUI) for the first coursework for INGCollage, which is collected from the Academic and Non academic classes and kept in an array list called course in the INGNepal class. There were different user interface elements present, such as a text field, a button, a scroll pane, a content pane, a label, and so on. Among the several GUI elements, both label and textfield area have been utilized to provide basic details of academic and non academic course. Inside this GUI, each buttons are often used to enter data simple text field information, display all of the personnel information entered in the text fields, and dismiss the data.

The project has various methods that result in errors and flaws in the application. Furthermore, several research was conducted, errors were debugged, and the program was intended to function in a systematic manner. I am actually able to complete this coursework within a week of a lot of hard working. The above coursework is a bit ahead in building code and graphical user interface (GUI) in the future for the development of someone's graphical user interface program. While working on this program, I learned regarding Java GUI and its numerous built-in features. Furthermore, a general knowledge of class diagrams, pseudocode, procedure and researching technique were gained. The above courses will undoubtedly be quite valuable in developing a profession as a skilled programmer. And I was getting help from both my teacher and my seniors. I worked really hard as well as gave it my all to finish this course work. Therefore, I believe that this was a pleasure for me to complete this course work when I learned something new, plus programming is more than just writing code; it is also about analytical thinking and knowledge of code functions.

## 8.0 References

Anon., 2021. *Java T point*. [Online]

Available at: <https://www.javatpoint.com/gui-full-form>

[Accessed 2011].

Anon., 2021. *Java T Point*. [Online]

Available at: <https://www.javatpoint.com/java-arraylist>

Heller, M., 2021. *techtarget*. [Online]

Available at: <https://whatis.techtarget.com/definition/pseudocode>

[Accessed 13 08 2021].

jain, S., 2021. *geeksforgeek*. [Online]

Available at: <https://www.geeksforgeeks.org/arraylist-in-java/>

[Accessed 12 08 2021].

Jain, S., 2021. *wideskills*. [Online]

Available at: <https://wideskills.com/java-tutorial/introduction-to-java-programming>

[Accessed 10 08 2021].

kant, k., 2021. *techopedia*. [Online]

Available at: <https://www.techopedia.com/definition/26102/java-swing>

[Accessed 11 08 2021].

pedamkar, p., 2021. *www.educba.com*. [Online]

Available at: <https://www.educba.com/what-is-gui/>

[Accessed 11 08 2021].

whitman, w., 2021. *developer.mozilla.org*. [Online]

Available at: [https://developer.mozilla.org/en-US/docs/Web/Events/Event\\_handlers](https://developer.mozilla.org/en-US/docs/Web/Events/Event_handlers)

[Accessed 11 08 2021].

## 9.0 Appendix

```
import javax.swing.*;

import java.awt.Color;

import javax.swing.JButton;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.List;

import javax.swing.BorderFactory;

import javax.swing.border.Border;

import java.awt.FlowLayout;

// making access modifier private

public class INGCollage {

    private ArrayList<Course> list = new ArrayList<>();

    private JFrame frame;

    private JPanel panel1, panel2;

    private JLabel lbP1CourseID;

    private JLabel lbp1CourseName;

    private JLabel lbp1Duration;

    private JLabel lbp1NumberofAssessment;

    private JLabel lbp1LecturareName;

    private JLabel lbp1CourseLeader;
```



```
private JLabel lbp1Level;  
private JLabel lbp1Credit;  
private JLabel lbp1Course;  
private JLabel lbp1StartDate;  
private JLabel lbp1CompletionDate;  
  
private JTextField txtP1CourseID;  
private JTextField txtp1CourseName;  
private JTextField txtp1Duration;  
private JTextField txtp1NumberofAssessment;  
private JTextField txtp1LecturerName;  
private JTextField txtp1CourseLeader;  
private JTextField txtp1Level;  
private JTextField txtp1Credit;  
private JTextField txtp1Startdate;  
private JTextField txtp1CompletionDate;  
  
private JLabel lbP2CourseID;  
private JLabel lbp2CourseName;  
private JLabel lbp2Duration;  
private JLabel lbp2Prerequisite;  
private JLabel lbp2Course;  
private JLabel lbp2CourseLeader;
```

```
private JLabel lbp2InstructorName;

private JLabel lbp2Startdate;

private JLabel lbp2Completiondate;

private JLabel lbp2Examdate;


private JTextField txtP2CourseID;

private JTextField txtP2CourseIDCheck;

private JTextField txtP2CourseName;

private JTextField txtP2Duration;

private JTextField txtP2Prerequisite;

private JTextField txtP2Course;

private JTextField txtP2CourseLeader;

private JTextField txtP2InstructorName;

private JTextField txtP2StartDate;

private JTextField txtP2Completiondate;

private JTextField txtP2ExamDate;

private JButton btn1AddAcademic, btn1DisplayAcademic, btn1RegisterAcademic,
btn1ClearAcademic;

private JButton btnAddNonAcademic, btnDisplayNonAcademic,
btnRegisterNonAcademic, btnClearNonAcademic, btnRemoveNonAcademic;

// creating frame a and b in jpannnel

public INGCollage() {

    initialFrame();
```

```
        myFrameA();

        myFrameB();
    }

    // making access modifier public for attributes
    public String getCourseld(){
        return txtP1CourseID.getText();
    }

    public String getCourseName(){
        return txtp1CourseName.getText();
    }

    public int getDuration(){
        String duration = txtp1Duration.getText();
        int Duration = -1;
        try {
            Duration = Integer.parseInt(duration);
        } catch (NumberFormatException nfe) {
            JOptionPane.showMessageDialog(frame, "Incorrect value for number of
Duration.\nPlease add numeric value", "Invalid Input", 0);
        }
        return Duration;
    }
}
```

```
public int getNumberOfAssessment(){
    String numberOfAssessment=txtp1NumberofAssessment.getText();
    int numberOfAssessmentNum=-1;
    try{
        numberOfAssessmentNum=Integer.parseInt(numberOfAssessment);
    }catch(NumberFormatException nfe){
        JOptionPane.showMessageDialog(frame,"Please enter the numeric values for
        numberof assessment","Invalid Data",2);
    }
    return numberOfAssessmentNum;
}

//using gettter and settter method
public String getLevel(){
    return txtp1Level.getText();
}

public String getCredit(){
    return txtp1Credit.getText();
}

public String getLecturerName() {
```

```
        return txtp1LecturerName.getText();  
    }  
  
    public String getCourseLeader(){  
        return txtp1CourseLeader.getText();  
    }  
  
    public String getStartDate(){  
        return txtp1Startdate.getText();  
    }  
  
    public String getCompletionDate(){  
        return txtp1CompletionDate.getText();  
    }  
  
    public String getAcademicStartDate() {  
        return txtp1Startdate.getText();  
    }  
  
    public String getNonAcademicCourseName() {  
        return txtp2CourseName.getText();  
    }  
}
```

```
public String getNonAcademicCourseID() {  
    return txtP2CourseID.getText();  
}
```

```
public String getNonAcademicCourseIDCheck() {  
    return txtP2CourseIDCheck.getText();  
}
```

```
public int getNonAcademicDuration() {  
    String Duration=txtp2Duration.getText();  
    int duration=-1;  
    try{  
        duration=Integer.parseInt(Duration);  
    }catch(NumberFormatException nfe){  
        JOptionPane.showMessageDialog(frame,"please enter the numeric value for  
duration","Invalid Data",2);  
    }  
    return duration;  
}
```

```
public String getNonAcademicStartDate() {  
    return txtp2StartDate.getText();  
}
```

```
public String getNonAcademicInstructorname() {  
    return txtp2InstructorName.getText();  
}
```

```
public String getNonAcademicCourseLeader() {  
    return txtp2CourseLeader.getText();  
}
```

```
public String getNonAcademicCompletiondate() {  
    return txtp2Completiondate.getText();  
}
```

```
public String getNonAcademicExamDate() {  
    return txtp2ExamDate.getText();  
}
```

```
public String getNonAcademicPrerequisite() {  
    return txtp2Prerequisite.getText();  
}
```

```
public void initialFrame() {  
    // ArrayList<Academic>list = new ArrayList<>();
```

```
frame = new JFrame("INGCOllage");

frame.setLayout(null);

frame.setSize(1250, 580);

frame.setResizable(true);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


panel1 = new JPanel();

frame.add(panel1);

panel1.setBounds(10, 70, 610, 600);

panel1.setLayout(null);

panel1.setBorder(BorderFactory.createTitledBorder("Academic Course"));


panel2 = new JPanel();

frame.add(panel2);

panel2.setBounds(640, 70, 600, 600);

panel2.setLayout(null);


panel2.setBorder(BorderFactory.createTitledBorder("NonAcademic Course"));

}


public void myFrameA() {

    JLabel lbRegistration = new JLabel("Registration form");

    lbRegistration.setBounds(590, 30, 120, 25);
```



```
frame.add(lbRegistration);
```

```
ArrayList<Course> CourseName = new ArrayList<Course>();
```

```
lbP1CourseID = new JLabel("Course ID:");
```

```
panel1.add(lbP1CourseID);
```

```
lbP1CourseID.setBounds(15, 30, 75, 25);
```

```
txtP1CourseID = new JTextField();
```

```
panel1.add(txtP1CourseID);
```

```
txtP1CourseID.setBounds(100, 30, 150, 25);
```

```
Border border = BorderFactory.createLineBorder(Color.RED, 5);
```

```
//.setBorder(BorderFactory.createLineBorder(Color.BLUE,5));
```

```
lbp1Duration = new JLabel("Duration:");
```

```
panel1.add(lbp1Duration);
```

```
lbp1Duration.setBounds(15, 80, 75, 25);
```

```
txtp1Duration = new JTextField();
```

```
panel1.add(txtp1Duration);
```

```
txtp1Duration.setBounds(100, 80, 150, 25);
```

```
lbp1CourseName = new JLabel("CourseName:");
```

```
panel1.add(lbp1CourseName);

lbp1CourseName.setBounds(280, 30, 100, 25);


txtp1CourseName = new JTextField();

panel1.add(txtp1CourseName);

txtp1CourseName.setBounds(430, 30, 150, 25);


lbp1NumberofAssessment = new JLabel("NumberofAssessment:");

panel1.add(lbp1NumberofAssessment);

lbp1NumberofAssessment.setBounds(280, 80, 150, 25);


txtp1NumberofAssessment = new JTextField();

panel1.add(txtp1NumberofAssessment);

txtp1NumberofAssessment.setBounds(430, 80, 150, 25);


lbp1Level = new JLabel("Level :");

panel1.add(lbp1Level);

lbp1Level.setBounds(15, 130, 150, 25);


txtp1Level = new JTextField();

panel1.add(txtp1Level);

txtp1Level.setBounds(100, 130, 150, 25);
```

```
lbp1Credit = new JLabel("Credit :");  
panel1.add(lbp1Credit);  
  
lbp1Credit.setBounds(280, 130, 150, 25);  
  
  
txtp1Credit = new JTextField();  
panel1.add(txtp1Credit);  
  
txtp1Credit.setBounds(430, 130, 150, 25);  
  
  
btn1AddAcademic = new JButton("Add");  
btn1AddAcademic.setBounds(280, 170, 95, 30);  
panel1.add(btn1AddAcademic);  
  
btn1AddAcademic.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent ae) {  
        addAcademicCourse();  
    }  
});  
  
  
// btnAdd.addActionListener(new ActionListener(){  
// public void actionPerformed(ActionEvent ae){  
// String courseID = txtP1CourseID.getText();  
  
  
// String courseName = txtp1CourseName.getText();
```

```
// int duration = txtp1Duration.getText();

// String numberOfAssessment = txtp1NumberOfAssessment.getText();

// String level = txtp1Level.getText();

// String Credit = txtp1Credit.getText();


// Empty text Field, Repeated courseId,duration -> int

// AcademicCourse.add(new AcademicCourse(courseId,courseName,

// duration,level,Credit,numberofassessment ));

// JOptionPane.showMessageDialog(frame,"Data added successfully");

// }

// });

// JButton button = new JButton("Add");

// panel1.add(button);

// button.setBounds(250,130,95,30);

// button.addActionListener(new ActionListener(){

// public void actionPerformed(ActionEvent ae){

// add(Course);

// }

// });

// });


lbp1CompletionDate = new JLabel("Completion date:");

panel1.add(lbp1CompletionDate);

lbp1CompletionDate.setBounds(15, 230, 150, 25);
```

```
txtp1CompletionDate = new JTextField();  
panel1.add(txtp1CompletionDate);  
txtp1CompletionDate.setBounds(130, 230, 150, 25);
```

```
lbp1LecturareName = new JLabel("LecturerName:");  
panel1.add(lbp1LecturareName);  
lbp1LecturareName.setBounds(15, 280, 150, 25);
```

```
txtp1LecturerName = new JTextField();  
panel1.add(txtp1LecturerName);  
txtp1LecturerName.setBounds(130, 280, 150, 25);
```

```
lbp1CourseLeader = new JLabel("CourseLeader :");  
panel1.add(lbp1CourseLeader);  
lbp1CourseLeader.setBounds(330, 230, 150, 25);
```

```
txtp1CourseLeader = new JTextField();  
panel1.add(txtp1CourseLeader);  
txtp1CourseLeader.setBounds(430, 230, 150, 25);
```

```
lbp1StartDate = new JLabel("Start Date:");  
panel1.add(lbp1StartDate);
```

```
lbp1StartDate.setBounds(330, 270, 150, 25);
```

```
ttxp1Startdate = new JTextField();
```

```
panel1.add(ttxp1Startdate);
```

```
ttxp1Startdate.setBounds(430, 270, 150, 25);
```

```
btn1DisplayAcademic = new JButton("Display");
```

```
panel1.add(btn1DisplayAcademic);
```

```
btn1DisplayAcademic.setBounds(50, 340, 95, 30);
```

```
btn1DisplayAcademic.addActionListener(new ActionListener() {
```

```
    public void actionPerformed(ActionEvent ae) {
```

```
        displayAcademicCourse();
```

```
    }
```

```
});
```

```
btn1ClearAcademic = new JButton("Clear");
```

```
panel1.add(btn1ClearAcademic);
```

```
btn1ClearAcademic.setBounds(480, 340, 95, 30);
```

```
btn1ClearAcademic.addActionListener(new ActionListener() {
```

```
    public void actionPerformed(ActionEvent ae) {
```

```
        clearAcademicCourse();
```

```
    }  
    });  
  
    JButton Register = new JButton("Register ");  
    panel1.add(Register);  
  
    Register.setBounds(280, 320, 95, 30);  
  
    Register.addActionListener(new ActionListener() {  
        public void actionPerformed(ActionEvent ae) {  
            registerAcademicCourse();  
        }  
    });  
}
```

```
public void myFrameB() {  
  
    lbP2CourseID = new JLabel("CourseID");  
    panel2.add(lbP2CourseID);  
  
    lbP2CourseID.setBounds(15, 30, 75, 25);  
  
    txtP2CourseID = new JTextField();  
    panel2.add(txtP2CourseID);  
  
    txtP2CourseID.setBounds(100, 30, 150, 25);  
}
```

```
lbp2CourseName = new JLabel("CourseName:");  
panel2.add(lbp2CourseName);  
lbp2CourseName.setBounds(300, 30, 100, 25);
```

```
txtp2CourseName = new JTextField();  
panel2.add(txtp2CourseName);  
txtp2CourseName.setBounds(390, 30, 150, 25);
```

```
lbp2Duration = new JLabel("Duration:");  
panel2.add(lbp2Duration);  
lbp2Duration.setBounds(15, 80, 75, 25);
```

```
txtp2Duration = new JTextField();  
panel2.add(txtp2Duration);  
txtp2Duration.setBounds(100, 80, 150, 25);
```

```
lbp2Prerequisite = new JLabel("prerequisite:");  
panel2.add(lbp2Prerequisite);  
lbp2Prerequisite.setBounds(300, 80, 75, 25);
```

```
txtp2Prerequisite = new JTextField();  
panel2.add(txtp2Prerequisite);
```



```
txtp2Prerequisite.setBounds(390, 80, 150, 25);
```

```
btnAddNonAcademic = new JButton("Add");
```

```
btnAddNonAcademic.setBounds(250, 130, 95, 30);
```

```
panel2.add(btnAddNonAcademic);
```

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

```
    public void actionPerformed(ActionEvent ae) {
```

```
        addNonAcademicCourse();
```

```
    }
```

```
});
```

```
lbP2CourseID = new JLabel("CourseID:");
```

```
panel2.add(lbP2CourseID);
```

```
lbP2CourseID.setBounds(15, 200, 150, 25);
```

```
txtP2CourseID = new JTextField();
```

```
panel2.add(txtP2CourseID);
```

```
txtP2CourseID.setBounds(120, 200, 150, 25);
```

```
lbp2CourseLeader = new JLabel("CourseLeader :");
```

```
panel2.add(lbp2CourseLeader);
```

```
lbp2CourseLeader.setBounds(290, 200, 150, 25);
```

```
txtp2CourseLeader = new JTextField();  
panel2.add(txtp2CourseLeader);  
txtp2CourseLeader.setBounds(390, 200, 150, 25);  
  
lbp2InstructorName = new JLabel("InstructorName");  
panel2.add(lbp2InstructorName);  
lbp2InstructorName.setBounds(15, 240, 100, 25);  
  
txtp2InstructorName = new JTextField();  
panel2.add(txtp2InstructorName);  
txtp2InstructorName.setBounds(120, 240, 150, 25);  
  
lbp2Startdate = new JLabel("Start Date:");  
panel2.add(lbp2Startdate);  
lbp2Startdate.setBounds(290, 240, 150, 25);  
  
txtp2StartDate = new JTextField();  
panel2.add(txtp2StartDate);  
txtp2StartDate.setBounds(390, 240, 150, 25);  
  
lbp2Examdate = new JLabel("Exam Date:");  
panel2.add(lbp2Examdate);
```

```
lbp2Examdate.setBounds(290, 280, 150, 25);
```

```
txtp2ExamDate = new JTextField();
```

```
panel2.add(txtp2ExamDate);
```

```
txtp2ExamDate.setBounds(390, 280, 150, 25);
```

```
lbp2Completiondate = new JLabel("Completion Date");
```

```
panel2.add(lbp2Completiondate);
```

```
lbp2Completiondate.setBounds(15, 280, 100, 25);
```

```
txtp2Completiondate = new JTextField();
```

```
panel2.add(txtp2Completiondate);
```

```
txtp2Completiondate.setBounds(120, 280, 150, 25);
```

```
JButton Register = new JButton("Register ");
```

```
panel2.add(Register);
```

```
Register.setBounds(250, 330, 95, 30);
```

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

```
    public void actionPerformed(ActionEvent ae) {
```

```
        registerNonAcademicCourse();
```

```
    }
```

```
});
```

```
btnRemoveNonAcademic = new JButton("Remove");

panel2.add(btnRemoveNonAcademic);

btnRemoveNonAcademic.setBounds(250, 430, 95, 30);

btnRemoveNonAcademic.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent ae) {

        RemovedNonAcademicCourse();

    }

});

btnDisplayNonAcademic = new JButton("Display");

panel2.add(btnDisplayNonAcademic);

btnDisplayNonAcademic.setBounds(50, 400, 95, 30);

btnDisplayNonAcademic.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent ae) {

        displayNonAcademicCourse();

    }

});

btnClearNonAcademic = new JButton("Clear");

panel2.add( btnClearNonAcademic);

btnClearNonAcademic.setBounds(430, 400, 95, 30);
```

```
btnClearNonAcademic.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent ae) {  
        clearNonAcademicCourse();  
    }  
});  
  
}  
  
// using 'this' keyword inside academic course  
public void addAcademicCourse(){  
    String courseID=this.getCourseId();  
    String courseName=this.getCourseName();  
    int duration=this.getDuration();  
    int numberOfAssessments=this.getNumberOfAssessment();  
    String level=this.getLevel();  
    String credit=this.getCredit();  
    boolean flag=false;  
  
    // using if and else condition and printing value  
    if(courseID.equals("")||courseName.equals("")||duration==  
1||numberOfAssessments==1||level.equals("")||credit.equals("")){  
        JOptionPane.showMessageDialog(frame,"Enter the value in every text  
field.", "Empty Data", 2);  
        return;  
    }  
}
```

```
}else{

    if(list.isEmpty()){

        flag=true;

    } else {

        for (Course dep : list) {

            if (dep.getCourseID().equals(courseID)) {

                JOptionPane.showMessageDialog(frame, "The course is already
added.", "add", 1);

                return;

            } else {

                flag = true;

            }

        }

    }

    if(flag==true){

        list.add(new AcademicCourse(courseID,courseName,duration,level,credit,
numberofAssessments));

        String message = "CourseID : " + courseID + "\nCourseName : " +
courseName + "\nDuration : "+ duration + "\nLevel : " + level + "\nCcredit:" + credit +
"\nNumberofassessment:" + numberofAssessments;

        JOptionPane.showMessageDialog(frame, message, "Data Added", 1);

    }

}
```

```
}
```

```
public void registerAcademicCourse() {  
    String courseID = this.getCourseId();  
    String lectureName = this.getLecturerName();  
    String courseLeader = this.getLecturerName();  
    String startingDate = this.getStartDate();  
    String completionDate = this.getCompletionDate();  
    boolean flag = false;  
  
    if (courseID.equals("") || courseLeader.equals("") || lectureName.equals("") ||  
startingDate.equals("")  
        || completionDate.equals("")) {  
        JOptionPane.showMessageDialog(frame, "The fields are empty.\nPlease fill the  
required data", "Empty Field",  
            2);  
        return;  
    } else {  
        if (list.isEmpty()) {  
            JOptionPane.showMessageDialog(frame, "The courseIDdoesnot  
exists.\nPlease add the courseID data",  
                "Incorrect Input", 2);  
            return;  
        }  
    }  
}
```

```
    } else {  
        for (Course cou : list) { // Course obj : arrayList / AcademicCourse &  
NonAcademicCourse XX  
            if (cou instanceof AcademicCourse) {  
                if (cou.getCourseID().equals(courseID)) { // Java Python  
                    AcademicCourse ac = (AcademicCourse) cou;  
                    if (ac.getRegistered() == true) {  
                        JOptionPane.showMessageDialog(frame, "The course is already  
register.", "register",1);  
                        return;  
                    } else {  
ac.register(courseLeader,lectureName,startingDate,completionDate);  
                        String message = "CourseID : " + courseID + "\nCourseLeader : " +  
courseLeader  
                        + "\nLeactureName : " + lectureName + "\nStartingDate : " +  
startingDate  
                        + "\nCompletionDate" + completionDate;  
                        JOptionPane.showMessageDialog(frame, message, "Data Added",  
1);  
                        return;  
                    }  
                }  
            }  
        }  
    }  
}
```



```
        }  
    }  
  
    JOptionPane.showMessageDialog(frame, "The course does not  
exists.\nPlease add the data",  
  
    "No platform", 2);  
  
    }  
  
    }  
  
}
```

```
public void clearAcademicCourse(){  
    txtP1CourseID.setText("");  
    txtp1CourseName.setText("");  
    txtp1Duration.setText("");  
    txtp1Level.setText("");  
    txtp1Credit.setText("");  
    txtp1LecturerName.setText("");  
    txtp1Startdate.setText("");  
    txtp1CompletionDate.setText("");  
    txtp1NumberOfAssessment.setText("");  
    txtp1CourseLeader.setText("");  
  
}
```

```
public void displayAcademicCourse() {  
    for (Course dev : list) {  
        if (dev instanceof AcademicCourse) {  
            AcademicCourse AC = (AcademicCourse) dev;  
            AC.display();  
        }  
    }  
}
```

```
public void addNonAcademicCourse(){  
    String courseID=this.getNonAcademicCourseID();  
    String courseName=this.getNonAcademicCourseName();  
    int duration=this.getNonAcademicDuration();  
    //int numberOfAssessments=this.getNumberOfAssessment();  
    //String level=this.getLevel();  
    String prerequisite =this.getNonAcademicPrerequisite();  
    boolean flag=false;  
    if(courseID.equals("")||courseName.equals("")||duration==0  
1||prerequisite.equals("")){  
        JOptionPane.showMessageDialog(frame,"Enter the value in every text  
field.", "Empty Data",2);  
        return;  
    }else{
```

```
        if(list.isEmpty()){
            flag=true;
        } else {
            for (Course dep : list) {
                if (dep.getCourseID().equals(courseID)) {
                    JOptionPane.showMessageDialog(frame, "The course is already
added.", "add", 1);
                    return;
                } else {
                    flag = true;
                }
            }
        }
    }

    if(flag==true){
        list.add(new
NonAcademicCourse(courseID,courseName,duration,prerequisite));

        String message = "CourseID : " + courseID + "\nCourseName : " + courseName
+ "\nDuration : "+ duration + "\nPrerequisite: " + prerequisite;

        JOptionPane.showMessageDialog(frame, message, "Data Added", 1);

    }
```

```
}
```

```
public void registerNonAcademicCourse() {  
    String courseID = this.getNonAcademicCourseID();  
    String instructorName = this.getNonAcademicInstructorname();  
    String courseLeader = this.getNonAcademicCourseLeader();  
    String startingDate = this.getNonAcademicStartDate();  
    String completionDate = this.getNonAcademicCompletiondate();  
    String examDate = this.getNonAcademicExamDate();  
    boolean flag = false;  
  
    if (courseID.equals("") || courseLeader.equals("") || instructorName.equals("") ||  
startingDate.equals("")  
    || completionDate.equals("")||examDate.equals("")) {  
        JOptionPane.showMessageDialog(frame, "The fields are empty.\nPlease fill the  
required data", "Empty Field",  
            2);  
        return;  
    } else {  
        if (list.isEmpty()) {  
            JOptionPane.showMessageDialog(frame, "The courseIDdoesnot  
exists.\nPlease add the courseID data",  
                "Incorrect Input", 2);  
        }  
    }  
}
```

```

        return;

    } else {

        for (Course cou : list) { // Course obj : arrayList / AcademicCourse &
NonAcademicCourse XX

            if (cou instanceof NonAcademicCourse) {

                if (cou.getCourseID().equals(courseID)) { // Java Python

                    NonAcademicCourse ac = (NonAcademicCourse) cou;

                    if (ac.getRegistered() == true) {

                        JOptionPane.showMessageDialog(frame, "The course is already
register.", "register",1);

                        return;

                    } else {

ac.register(courseLeader,instructorName,startingDate,completionDate,examDate);

                        String message = "CourseID : " + courseID + "\nCourseLeader : " +
courseLeader

                            + "\nInstructorName : " + instructorName + "\nStartingDate : " +
startingDate

                            + "\nCompletionDate" + completionDate+"\nExamDate
:" +examDate;

                        JOptionPane.showMessageDialog(frame, message, "Data Added",
1);

                        return;

                    }

```

```
        }
    }
}

JOptionPane.showMessageDialog(frame, "The CourseID doesnot
exists.\nPlease add the Course data",

    "No CourseID", 2);

}

}

}

// public void clearAcademicCourse(){

// txtP1CourseID.setText("");

// }

public void displayNonAcademicCourse() {

    for (Course dev : list) {

        if (dev instanceof NonAcademicCourse) {

            NonAcademicCourse AB = (NonAcademicCourse) dev;

            AB.display();

        }

    }

}

public void clearNonAcademicCourse(){
```

```
txtP2CourseID.setText("");
txtp2CourseName.setText("");
txtp2InstructorName.setText("");
txtP2CourseID.setText("");
txtp2Duration.setText("");
txtp2Prerequisite.setText("");
txtp2CourseLeader.setText("");
txtp2StartDate.setText("");
txtp2Completiondate.setText("");
txtp2ExamDate.setText("");

}

public void RemovedNonAcademicCourse() {
    String id = this.getNonAcademicCourseID();
    if(id.equals("")){
        JOptionPane.showMessageDialog(frame,"Please add ID for which you want
course to be removed.", "Remove", 1);
        return;
    }else{
        if(list.isEmpty()){
            JOptionPane.showMessageDialog(frame," Please Add and register the
course inorder to removed.", "Warning", 1);
```

```
        return;
    }else{
        for(Course cou:list){
            if(cou.getCourseID().equals(id)){
                if(cou instanceof NonAcademicCourse){
                    NonAcademicCourse nac = (NonAcademicCourse) cou;
                    if(nac.getRemoved() == false){
                        nac.Remove();
                        JOptionPane.showMessageDialog(frame," remove
sucessfull","remove",1);
                    }else{
                        JOptionPane.showMessageDialog(frame,"already
remove","remove",1);
                        return;
                    }
                }
            }
        }else{
            JOptionPane.showMessageDialog(frame,"it does not
exist.","remove",1);
            return;
        }
    }
}
```



```
    }  
  
    }  
    //creating main method  
    public static void main(String[] args) {  
        new INGCollage().frame.setVisible(true);  
    }  
}
```