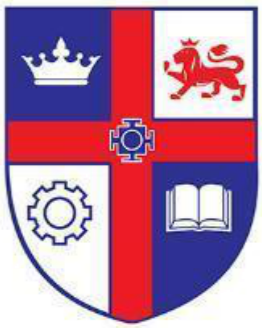
LONDON METROPOLITAN UNIVERSITY



islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI Programming

COURSEWORK-2

Assessment Weightage & Type

30% Individual Coursework

Semester and Year

Spring 2021

Student Name: ASAL PANDEY

Group: C7

London Met ID: 20049387

College ID: NP01CP4S210181

Assignment Due Date: 20th August, 2021

Assignment Submission Date: 20th August, 2021

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

Contents

Table of figures

1) Introduction

This coursework was given in the 20th week of college and had to be submitted before 24th. This coursework helped greatly to revise all the knowledge and skills which were gained from studying this module up until now. Many new things were learned while doing this coursework. This coursework was follow up for coursework-1 which was given to us in 9th week and it mainly focuses on GUI interfaces and teaches students many things about GUI mechanism and it's functioning in java. In this course work,a GUI is made. And the buttons are made functional afterwards for Adding, Removing and Registering academic courses and nonAcademicCourses from the Academic course class and non AcademicCourse class.

2) Class diagram

Table 1

Course
-duration: int -courseLeader: String -courseName: String -courseID

+getcourseID(): String +getcourseName(): String +getcourseLeader(): String +getduration(): int +setcourseLeader(String nameofthenewcoureLeader):void +display(): void

Table 1

AcademicCourse
-Lecturername: String -courseName: String -Level: Strings -Credit: int -NumberOfAssesment: int -StartingDate: String -CompletionDate: String
+get Lecturername (): String + courseName (): String +getLevel(): String +getCredit(): int +getNumberofAssesment(): int +getStartingDate(): string +getCompletionDate(): string +getisRegistered(): Boolean +setLecturername (String nameofthenewcoureLeader):void +setNumberOfAssesment(int newNumberOfAssesment):void + register(String Lecturername,String StartingDate,String

CompletionDate,String courseLeader):void

Table 2

nonAcademiccourse
-Lecturername: String -courseName: String -prerequisite: Strings -duration: int -ExamDate: String -StartingDate: String -CompletionDate: String -isRegistered:boolean -isRemoved:boolean
+get Lecturername (): String +getStartingDate(): string +getCompletionDate(): string +getPrerequsite(): string +getisRegistered(): Boolean +getisRemoved(): Boolean +setLecturername (String nameofthenewcoureLeader):void

+setNumberOfAssesment(int newNumberOfAssesment):void + register(String Lecturername, String StartingDate,String CompletionDate, String courseLeader, String ExamDate):void

Table 4

INGcollege
- courselist: Arraylist - frame: jframe - panel : JPanel - left_panel: JPanel - right_panel: JPanel - left_a1 : JPanel - left_a2: JPanel - left_a3: JPanel - right_p1: JPanel - right_p2: JPanel - right_p3: JPanel
- title: jlabel - title1: jlabel - labelAcademic: jLabel - alblID: jLabel -alblName: jLabel -alblDuration: jLabel

-alblLevel: jLabel
-alblCredit: jLabel
-alblAssessment; : jLabel
-alblLeader: jLabel
-alblLecturer: jLabel
-alblStart: jLabel
-alblCompletion: jLabel
-labelNonAcademic: jLabel
-nlblID: jLabel
-nlblName: jLabel
-nlblDuration: jLabel
-nlblPrerequisites: jLabel
-nlblLeader: jLabel
- nlblInstructor: jLabel
- nlblStart: jLabel
-nlblCompletion: jLabel
-nlblexam: jLabel
-atextID: JTextFeild
-atextName: JTextFeild
-atextDuration: JTextFeild
-atextLecturer: JTextFeild
-atextLevel: JTextFeild
-atextCredit: JTextFeild
-atextStart: JTextFeild
-atextCompletion: JTextFeild
-atextAssessment: JTextFeild
-ntextID: JTextFeild
-ntextName: JTextFeild
-ntextDuration: JTextFeild
-ntextLeader: JTextFeild
-ntextInstructor: JTextFeild

-ntextPrerequisite: JTextFeild -ntextStart: JTextFeild -ntextComplete: JTextFeild -ntextExam: JTextFeild - bttnAddAca: JButton -bttnAddNonAca: JButton -bttnRegisterAca: JButton -bttnRegisterNonAca: JButton -bttnRemoveNonAca: JButton -bttnDisplayAca: JButton -bttnDisplayNonAca: JButton -bttnClearAca: JButton -bttnClearNonAca: JButton
<constructor>+INGcollege ActionPerformed(Action Event e): void

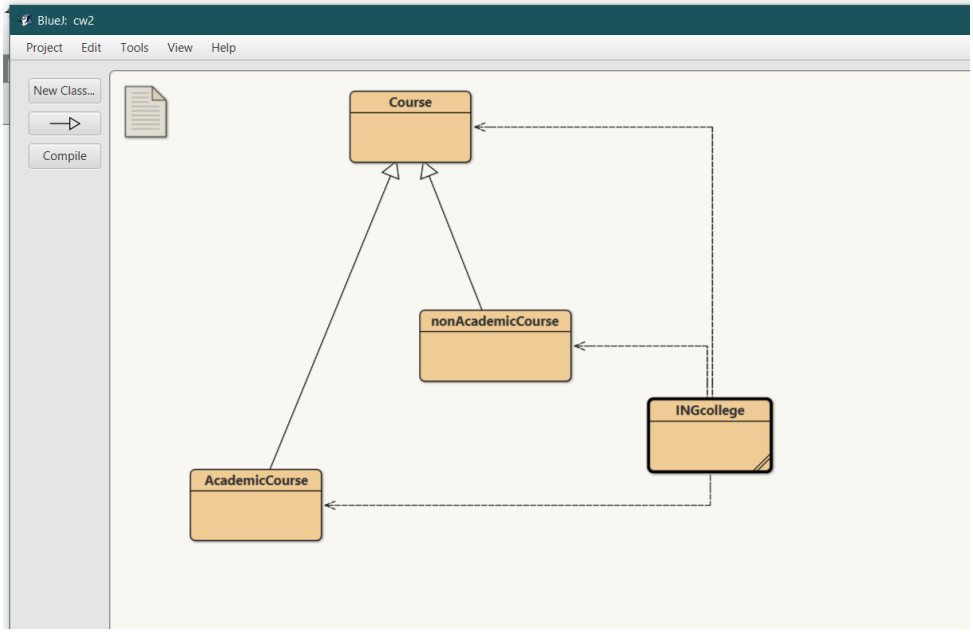


Figure 1: classdiagram layout

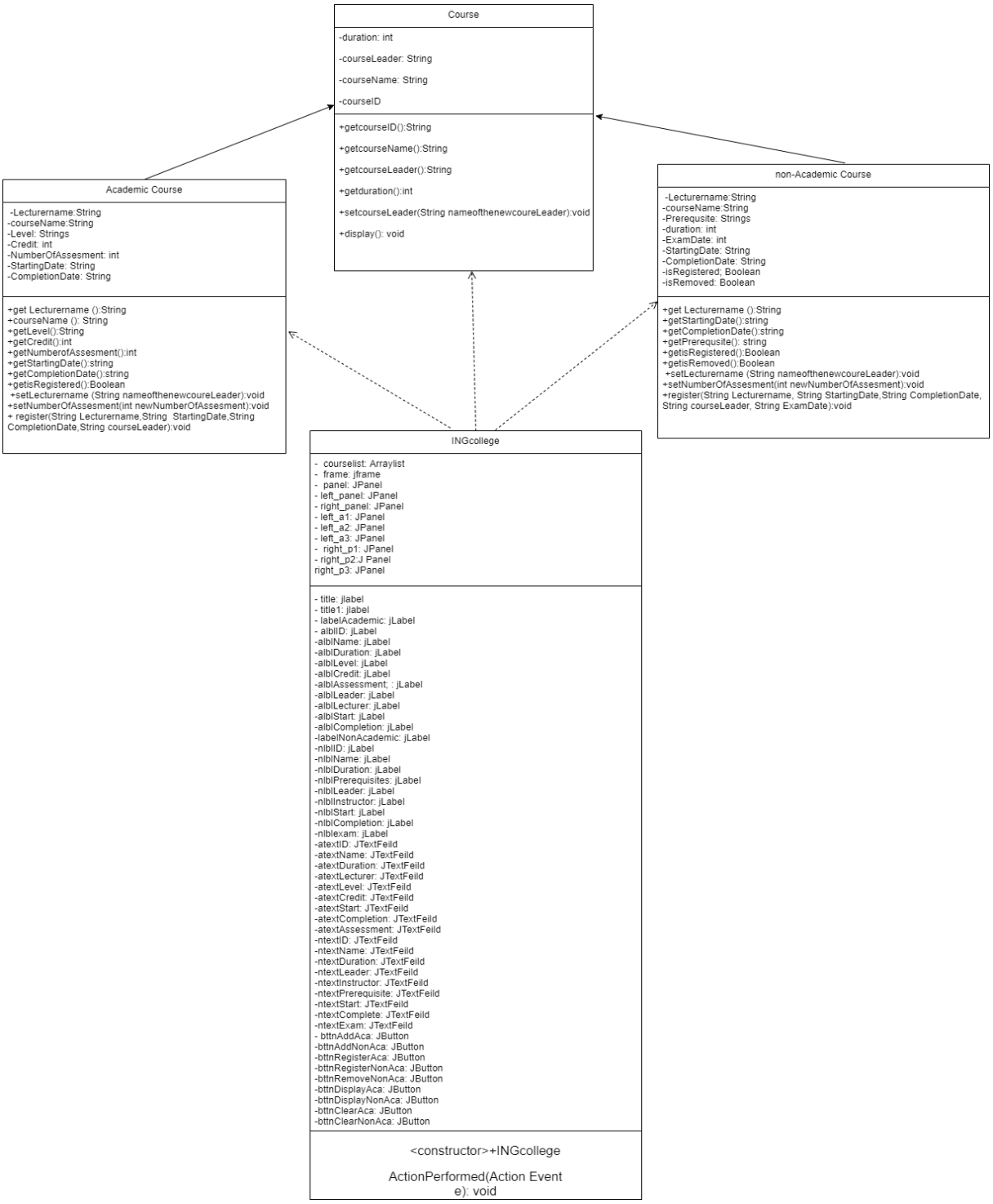


Figure 2: Class Diagram

3) Pseudocode

Pseudocode for Course

```
FUNCTION getcourseID ()
RETURN courseID;
END FUNCTION


FUNCTION getcourseName ()
RETURN courseName;
END FUNCTION


FUNCTION getcourseLeader()
RETURN courseLeader;
END FUNCTION


FUNCTION getduration ()
RETURN duration;
END FUNCTION


FUNCTION
setcourseLeader()
this.courseLeader=courseLeader;
END FUNCTION

FUNCTION display()
 DISPLAY "the ID of the course is ",get courseID
```

```
DISPLAY "Name of the course is ",get courseName
DISPLAY "Duration of the course is ",get duration

If
this.courseLeader==(” ”)
DISPLAY "empty shell"

Else
DISPLAY "the leader of the course is ",get courseLeader
```

Pseudocode for AcademicCourse

```
FUNCTION getLecturername ()
RETURN Lecturername;
END FUNCTION

FUNCTION
setLecturername ()
this. Lecturername=newLecturername;
END FUNCTION

FUNCTION getNumberofAssesment()
RETURN NumberofAssesment;
END FUNCTION

FUNCTION
setNumberofAssesment ()
this.NumberofAssesment=newNumberofAssesment;
END FUNCTION


FUNCTION getLevel ()
RETURN Level;
END FUNCTION

FUNCTION getCredit ()
RETURN Credit;
END FUNCTION

FUNCTION getStartingDate ()
```

RETURN StartingDate;
END FUNCTION

FUNCTION getComplitionDate ()
RETURN ComplitionDate;
END FUNCTION

FUNCTION getisRegistered()
RETURN isRegistered;
END FUNCTION

Function register(String Lecturername,String StartingDate,String CompletionDate,String courseLeader)

If,
isRegistered==true

DISPLAY “Lecturername+has already started at+StartingDate+and ends on+CompletionDate”

Else,
    super.setcourseLeader(courseLeader);
    this.Lecturername=Lecturername;
    this.StartingDate=StartingDate;
    this.CompletionDate=CompletionDate;
    isRegistered=true;

FUNCTION display()

```
super.display()
IF
(isRegistered=true)
DISPLAY "the name of the lecturer is=”,get Lecturername
DISPLAY "the level of the course is=",get Level
DISPLAY "total credit of the course is=" ,get Credit
DISPLAY "total no of assessment in this course is=",get NumberofAssesment
DISPLAY "starting date=",get StartingDate
DISPLAY “completion date”,get CompletionDate
End FUNCTION
```

PSEUDOCODE FOR nonAcademicCourse

```
FUNCTION getLecturername ()
RETURN Lecturername;
END FUNCTION


FUNCTION getduration ()
RETURN duration;
END FUNCTION



FUNCTION getStartingDate ()
RETURN StartingDate;
END FUNCTION

FUNCTION getComplitionDate ()
RETURN ComplitionDate;
END FUNCTION

FUNCTION getExamDate()
RETURN ExamDate;
END FUNCTION

FUNCTION getprerequisite ()
RETURN prerequisite;
END FUNCTION


FUNCTION getisRegistered()
```

```
RETURN isRegistered;
END FUNCTION

FUNCTION getisRemoved ()
RETURN isRemoved;
END FUNCTION

FUNCTION
setLecturername(String newLecturername)

if(isRegistered==false)
this.Lecturername=newLecturername;
DISPLAY “lecturer name is changed”
Else,
DISPLAY”lecturerer is already registered”
END FUNCTION

Function register(String Lecturername, String  StartingDate, String CompletionDate,
String courseLeader, Sring ExamDate)

If(isRegistered==false)
setLecturername(Lecturername);
isRegistered=true;
else,
Display”this course is already registered”

FUNCTION remove()
If(IsRemoved==true)
DISPLAY”the course is already been removed”
Else
super.setcourseLeader("");
```

this.Lecturername="";
this.StartingDate="";
this.CompletionDate="";
this.ExamDate="";
this.isRegistered=false;
this.isRemoved=true;
Display “this.Lecturername+ has already started at +this.StartingDate+ and ends on+this.CompletionDate”

FUNCTION display()
super.display()
IF (isRegistered=true)
DISPLAY "lecturer name is",get Lecturername
DISPLAY "starting date is",get StartingDate
DISPLAY "complition date is",get ComplitionDate
DISPLAY "exam date is",get ExamDate
END FUNCTION

Pseudocode for INGcollege

For action perform method()
Declare variable for loop
Duration, Assessment, credit =0
FUNCTION
if (e.getSource().equals(bttnAddAca))

```
try
    Duration=INT
    Assessment=INT
    Credit=INT
Catch
End catch
End try
End if
If
courselist.isEmpty()
ac = newAcademicCourse(duration,acourseID,acourseName,alevel,credit,assessment);
            courselist.add(ac)
            Panemessage= “Academic Course addition sucessful!"

else

            for(Course c:courselist)

               if(c instanceof AcademicCourse)

                  if(c.getCourseID().equals(acourseID))

panemessage “Course ID already exist.Please give unique course ID"
                     return;
                  else
                     isUnique = true;

            if (isUnique == true)
               ac = new
AcademicCourse(duration,acourseID,acourseName,alevel,credit,assessment);
               courselist.add(ac);
```

```
                isUnique = false;


     Panemessage= "Course has been successfully added");

       if (e.getSource().equals(bttnAddNonAca))

         try

   duration = (int)Double.parseDouble(nDuration);
         catch(NumberFormatException ex)
Panemessage= “please enter the data in required field


         If (courselist isEmpty())
            nac = new
nonAcademicCourse(duration,ncourseID,ncourseName,nprerequisites);
            courselist.add(nac);
            Panemessage= “Non-Academic Couse has been successfully added");

         else
            for
                (Course c:courselist)
               if (c instance of nonAcademicCourse)
                  if (c.getCourseID().equals(ncourseID))
Panemessage= "Course ID already exist.Please give another course ID!"
                     return;
                  else
                     isUnique=true;

         end for
```

```
end if
end if

        if(isUnique == true)
            nac = new
nonAcademicCourse(duration,ncourseID,ncourseName,nprerequisites);
            courselist.add(nac);
            isUnique = false;
            end if
end if
Panemessage= “Course has been added sucessfully"

End if




    if(e.getSource() equals(bttnRegisterAca))

        if(courselist isEmpty())

            Panemessage=”Course list is empty"

          else

                for(Course c:courselist){
                  if(c.getCourseID() equals(acourseID))
                    if(c instanceof AcademicCourse){
                      ac = (AcademicCourse)c

                        if(ac.getisRegistered()==false){
```

```
                    ac.register(acourseName,alecturer,astart,acompletion);
                  Panemessage= “Academic Course is registered"

                    return;
                else

Panemessage= “Academic Course is already registered!"

                    End if
                    else
Panemessage=” CourseID has been used for NonAcademicCourse"
                     }
                  End if
              else
Panemessage= "Course ID is doesnt exist in the list");

      End if
End for
End if
End if

    If(e.getSource() equals(bttnRegisterNonAca))

       if (courselist.isEmpty())
            Panemessage= "Course list is empty"

          else

              for(Course c:courselist){
                 if(c.getCourseID() equals(ncourseID))
```

```
                    if(c instance of nonAcademicCourse)
                        nac = (nonAcademicCourse)c
                        if
                    nac.getisRegistered()==false

nac.register(ncourseName,ninstructor,nstart,ncompletion,nexam);

                            Panemessage=”Non-Academic Course is registered
successfully!"
                            return;
                        else
                    Panemessage=  "Non-Academic Course is already registered!"

                End if
                    else
                Panemessage= “CourseID has been used for AcademicCourse"
                    End if
                    else
                        Panemessage= “CourseID is doesn't exist in the list"
End if
End for
End if
End if


      if(e.getSource().equals(bttnRemoveNonAca))
            System.out.println("Inside remove");
            If(
courselist isEmpty())

Panemessage="Course list is empty"
```

```
            else
                for(Course c:courselist)

                    if(c.getCourseID() equals(ncourseID)){

                        if(c instance of nonAcademicCourse){

                            nac = (nonAcademicCourse)c

                                if (nac.getisRemoved()==true)
Panemessage= "Non Academic Course is do not exist"



                                    else
                                        nac.remove();
Panemessage= “Non-Academic Course is removal sucessful"
                                    return;

                    end if
                        else
Panemessage= "CourseID has been used for AcademicCourse"

                     End if
                    else
                      Panemessage= "CourseID is not present in the list"

End if
End for
End if
End if
```

```
        if  (e.getSource() equals(bttnDisplayAca))

            if (courselist.isEmpty())
                Panemessage= "Blank List,Please fill the form");

                 System.out.println("CourseID is Blank");
               else
                    for(Course aac: courselist){
                       System.out.println("Displaying inside list--> Academic Course");

                       //for academic course

                        if(aac instanceof AcademicCourse)
                             System.out.println("----------------------------------------");
                             ac = (AcademicCourse)aac; //downcasting to call display method
                             ac.display();
end if
end for
end if
end if

        if(
e.getSource() equals(bttnDisplayNonAca))

            if(
courselist.isEmpty()==true)
Panemessage="Empty List, Please fill the form"
                System.out.println("Courselist is empty")
```

```
        else
            for(Course nc: courselist)
                System.out.println("Displaying inside list--> Non Academic Course")

                if(nc instance of nonAcademicCourse)
                    System.out.println("----------------------------------------");
                    nac = (nonAcademicCourse)nc
                    nac.display();
    end if
end for
end if
end if
    if (e.getSource() equals(bttnClearAca)

        atextID.setText("");
        atextName.setText("");
        ntextDuration.setText("");
        atextLevel.setText("");
        atextCredit.setText("");
        atextAssessment.setText("");
        atextLeader.setText("");
        atextLecturer.setText("");
        atextStart.setText("");
        atextCompletion.setText("");

    end if

    If (e.getSource().equals(bttnClearNonAca)

        ntextID.setText("");
        ntextName.setText("");
```

```
            ntextDuration.setText("");
            ntextPrerequisites.setText("");
            ntextLeader.setText("");
            ntextInstructor.setText("");
            ntextStart.setText("");
            ntextCompletion.setText("");
            ntextExam.setText("");

end if
static method()

create static method main(String args[]):void
create object of class named main
call constructor using object

end
```

4) Methods

bttnAddAca.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called and object of AcademicCourse class will be created and then added to the list, verifying its unique courseID

bttnAddNonAca.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called and object of nonAcademicCourse class will be created and then added to the list, verifying its unique courseID

bttnRegisterAca.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called. This method calls register method in order to register the course.

bttnRegisterNonAca.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called. This method calls register method in order to register the course.

abttndisplay.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called. This method calls display method in order to display the registered course.

nbttndisplay.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called. This method calls display method in order to display the registered course.

bttnClearAca.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called which sets all the value of given textfeilds to “ ” which is empty

bttnClearNonAca.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called which sets all the value of given textfeilds to “ ” which is empty

bttnRemoveNonAca.addActionListener(this): if the button is pressed, actionPerformed(ActionEvent: e) will be called. This method calls remove method for removing the course.

setLayout(): this method calls layout manager in order to set layout

setBounds(): this is a inbuild method of java.awt packaged which sets the location and boundries for the frame and user interface component of GUI

5) Testing

Test 1)

Objective	To check if the program compiles and run through command prompt
Action	Compile the .java extension file through javac filename.java command and Run the program through java filename command.
Expected result	The program will be compiled and run successfully
Actual result	The program got compiled and ran successfully
conclusion	Test was successful.

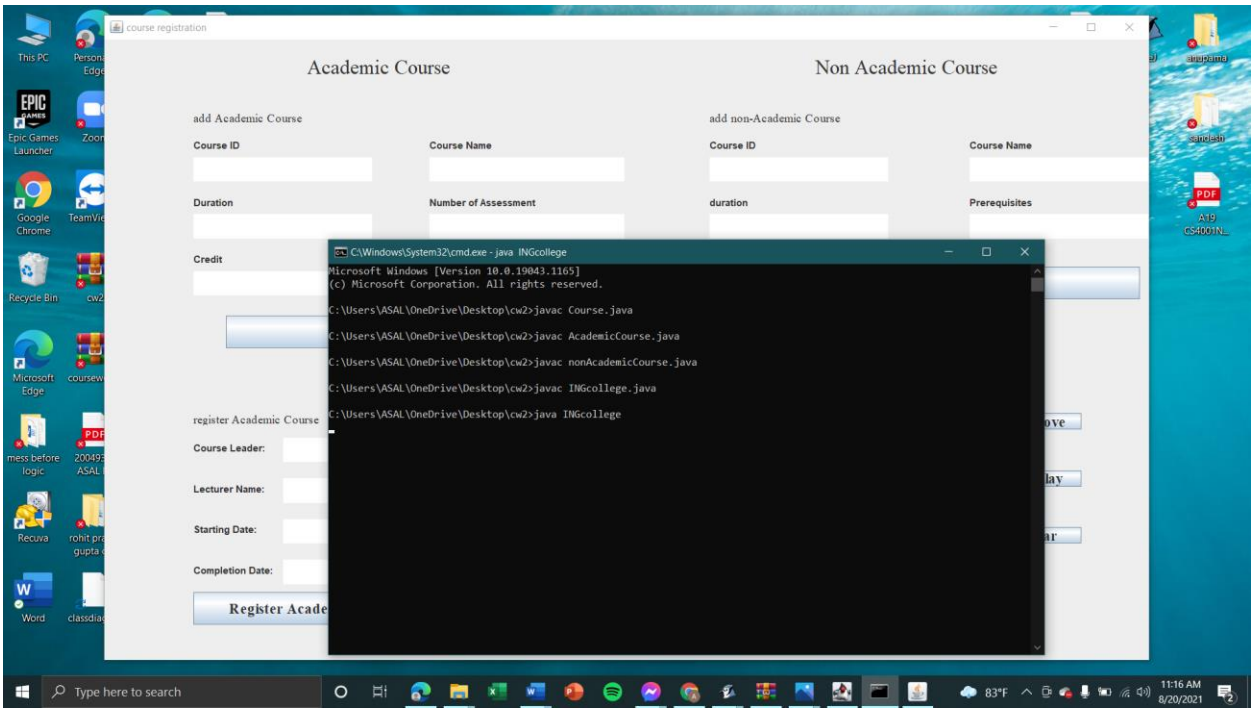


Figure 3 : checking the program runs through commandprompt or not

Test 2.1)

Objective	To add the course in acdemic course
Action	All the required textfeilds are filled and add button is pressed
Expected result	The course will be added to academic course
Actual result	The course was added to academic course successfully
conclusion	Test was sucessful

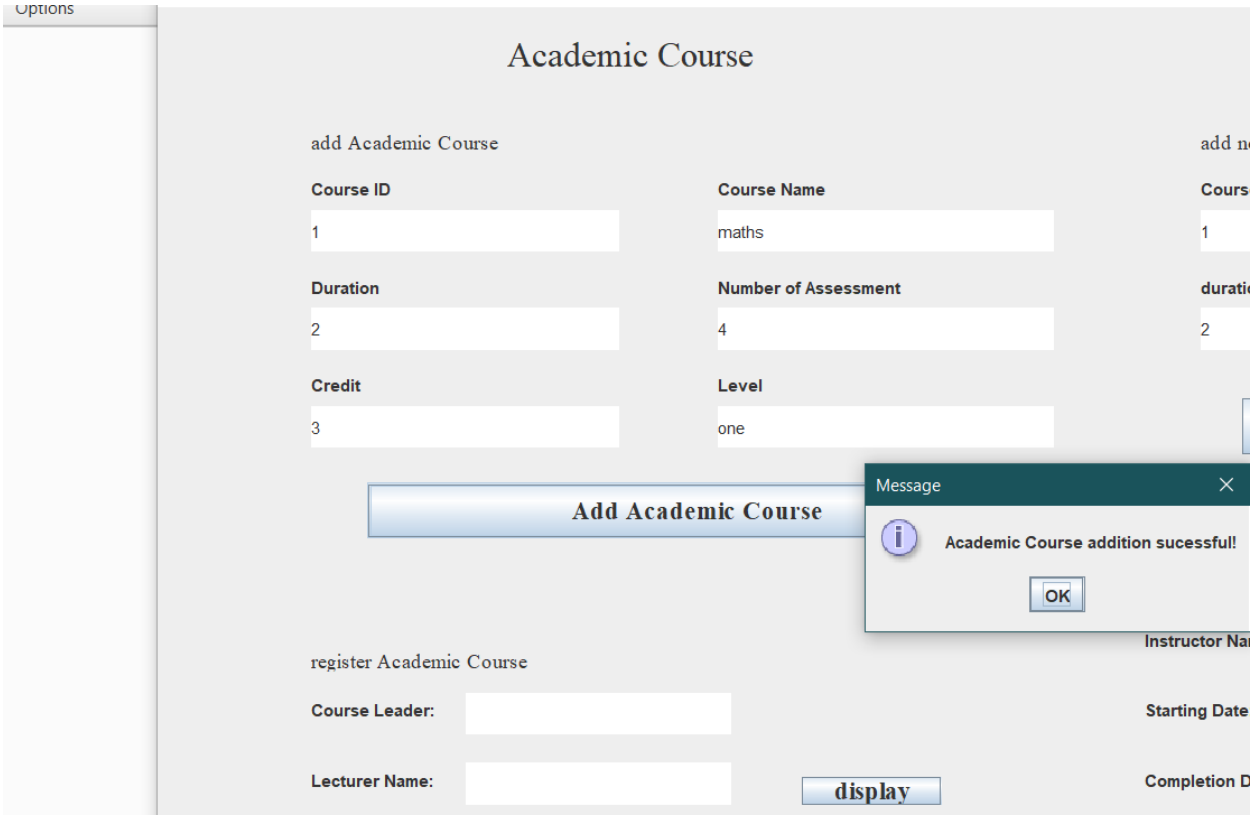


Figure 4:adding academic course

Test 2.2)

Objective	To add the course in non-acdemic course
Action	All the required textfeilds are filled and add button is pressed
Expected result	The course will be added to non-academic course
Actual result	The course was added to non-academic course successfully
conclusion	Test was sucessful

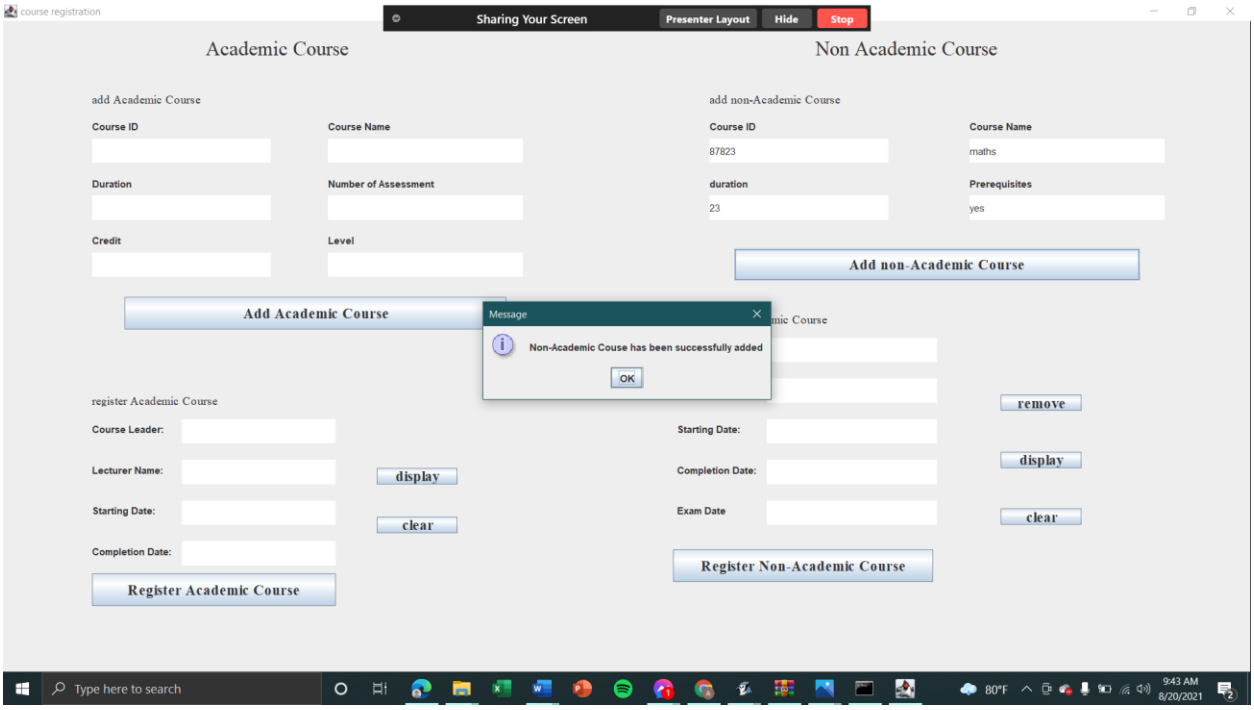


Figure 5: adding non academic course

Test 2.2)

Objective	Register academic course
Action	All the required textfeilds are filled and register button is pressed
Expected result	Academic course will be registered and dialoug box will appear stating “Academic Course is registered”
Actual result	Academic course was registered and dialogue box saying “Academic Course is registered” was shown
conclusion	Test was sucessful

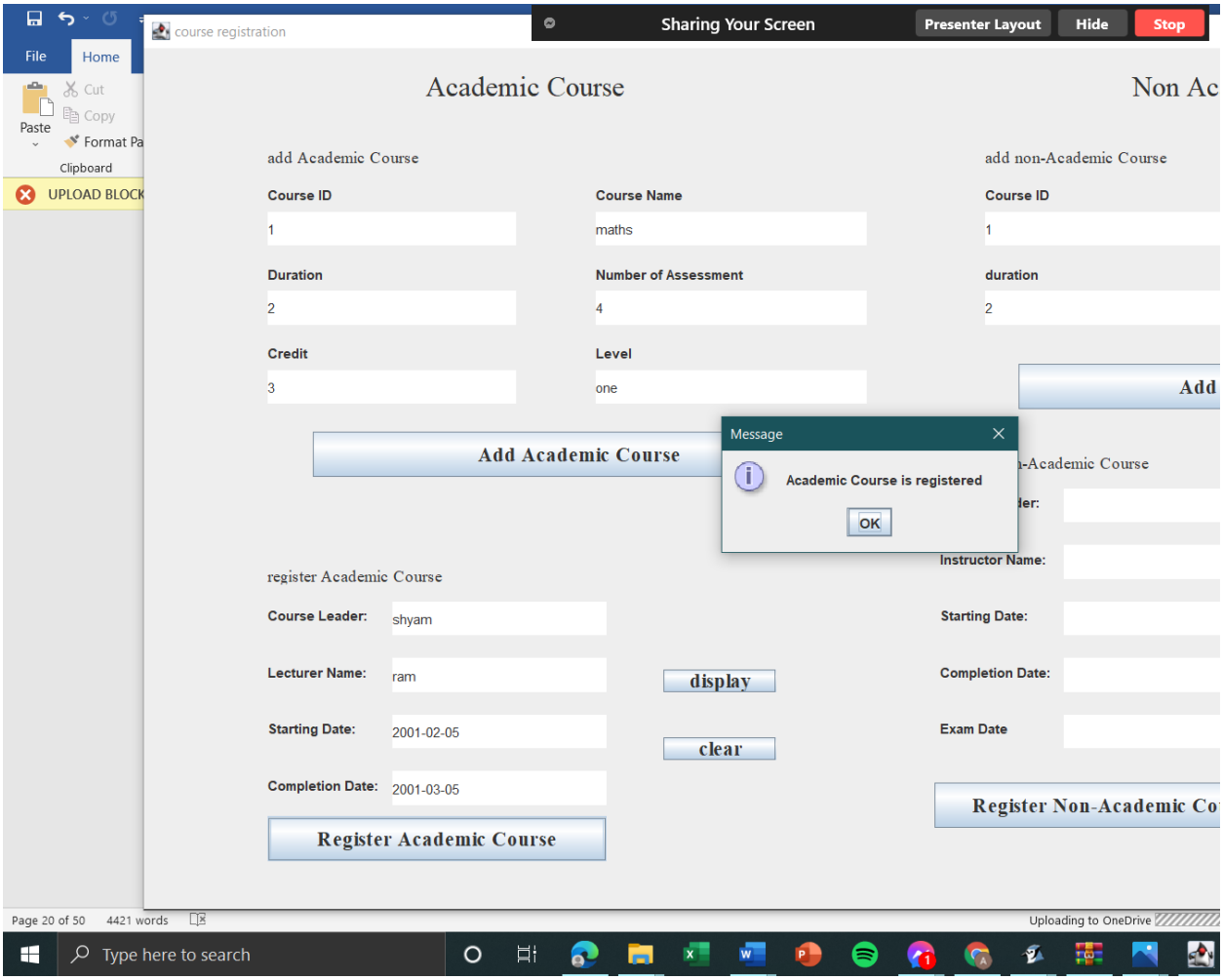


Figure 6: registering academic course

Test 2.4)

Objective	Register non-academic course
Action	All the required textfeilds are filled and register button is pressed
Expected result	Non-Academic course will be registered and dialouge box stating it will be shown
Actual result	Non-Academic course will be registered and dialouge box stating it was be shown
conclusion	Test was sucessful

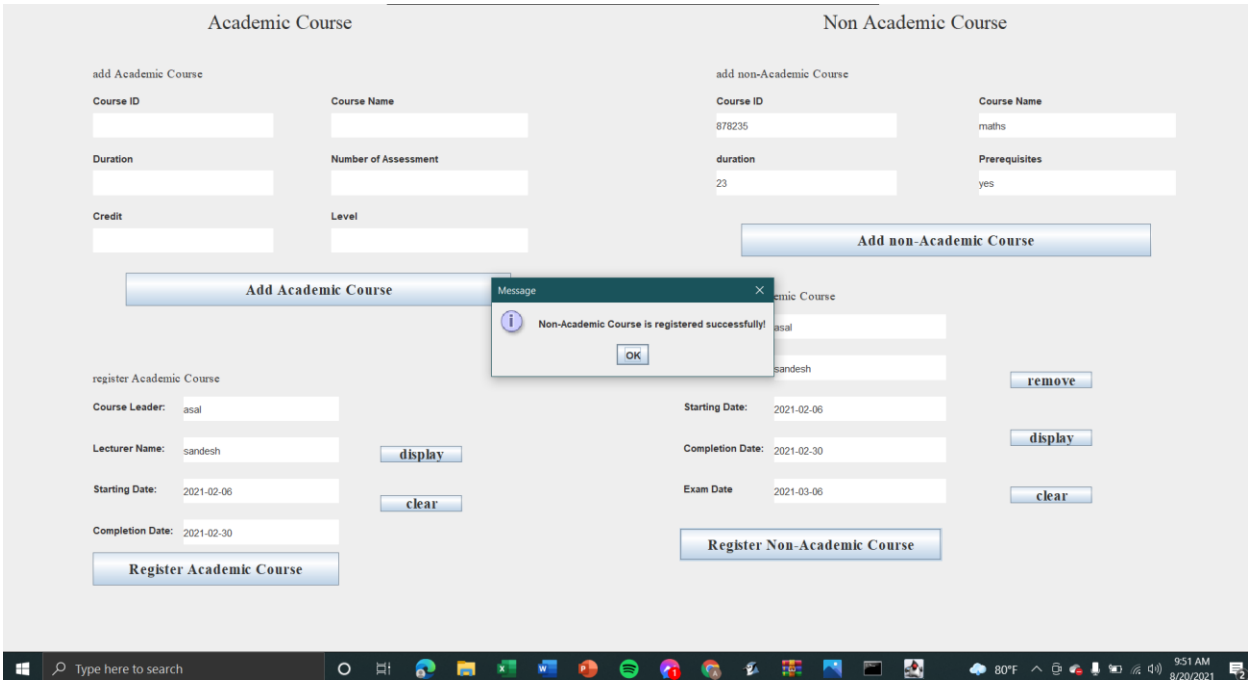


Figure 7: registering non academic course

Test 2.5)

Objective	Remove non-academc course
Action	All the required textfeilds are filled and remove button is pressed
Expected result	Non-Academic course will be removed
Actual result	Non-Academic course was removed
conclusion	Test was sucessful

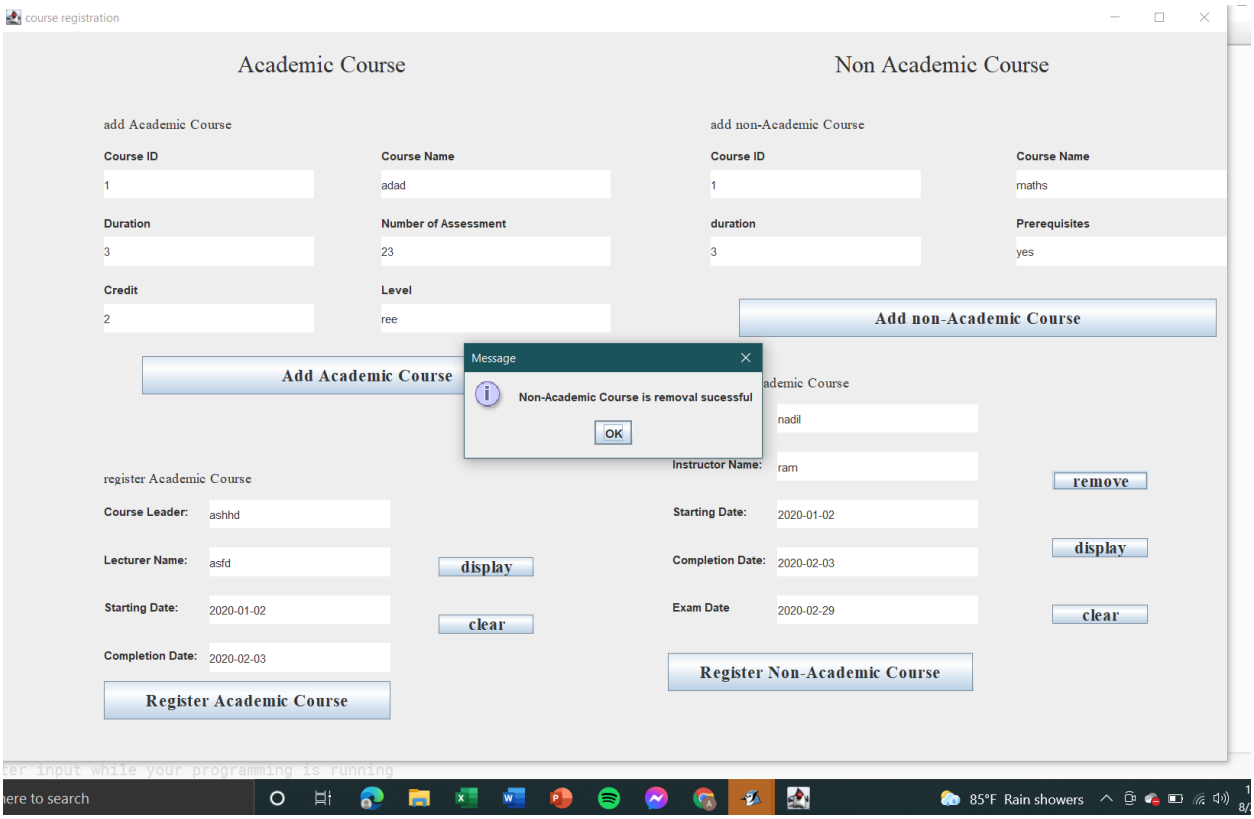


Figure 8: removing non academic course

Test 3.1)

Objective	Trying to Adding duplicate courseID
Action	Adding course with same data and courseID
Expected result	Messagebox saying “Course ID already exist.Please give another course ID!”is displayed
Actual result	Messagebox saying “Course ID already exist.Please give another course ID! ”was displayed
conclusion	Test was sucessful

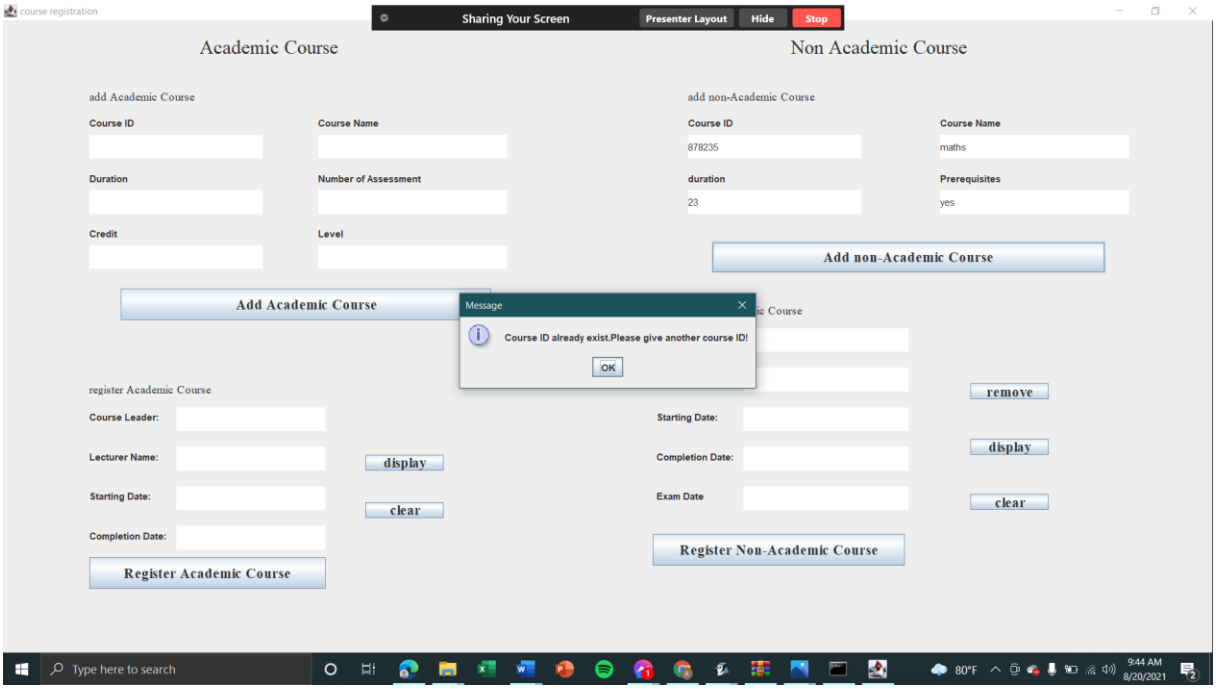


Figure 9:trying to add duplicate courseID

Test 3.2)

Objective	Trying to register already registered course
Action	Registering same course twice
Expected result	Messagebox saying “Academic Course is already registered! ”is displayed
Actual result	Messagebox saying “Academic Course is already registered!”was displayed
conclusion	Test was sucessful

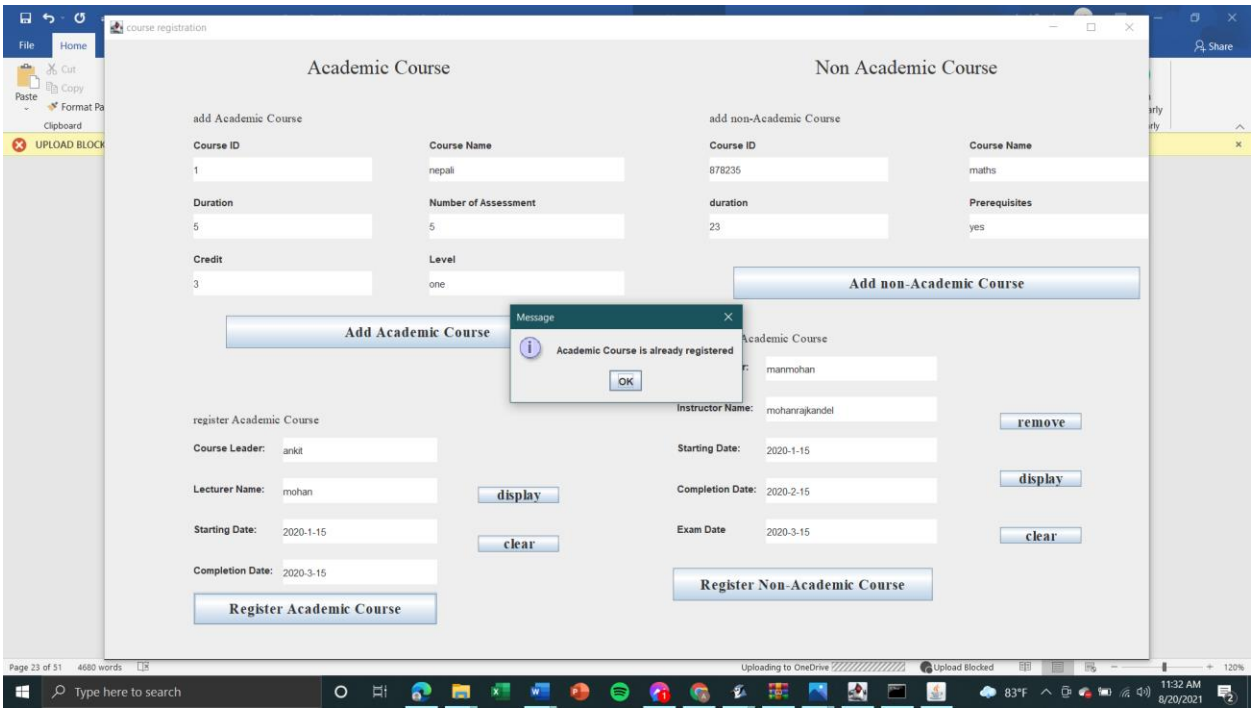


Figure 10:trying to register already registered course

Test 3.3)

Objective	Trying to remove non academic course which is already removed
Action	
Expected result	Messagebox saying " Non Academic Course do no exist"is displayed
Actual result	Messagebox saying "Non Academic Course do no exist "was displayed
conclusion	Test was sucessful

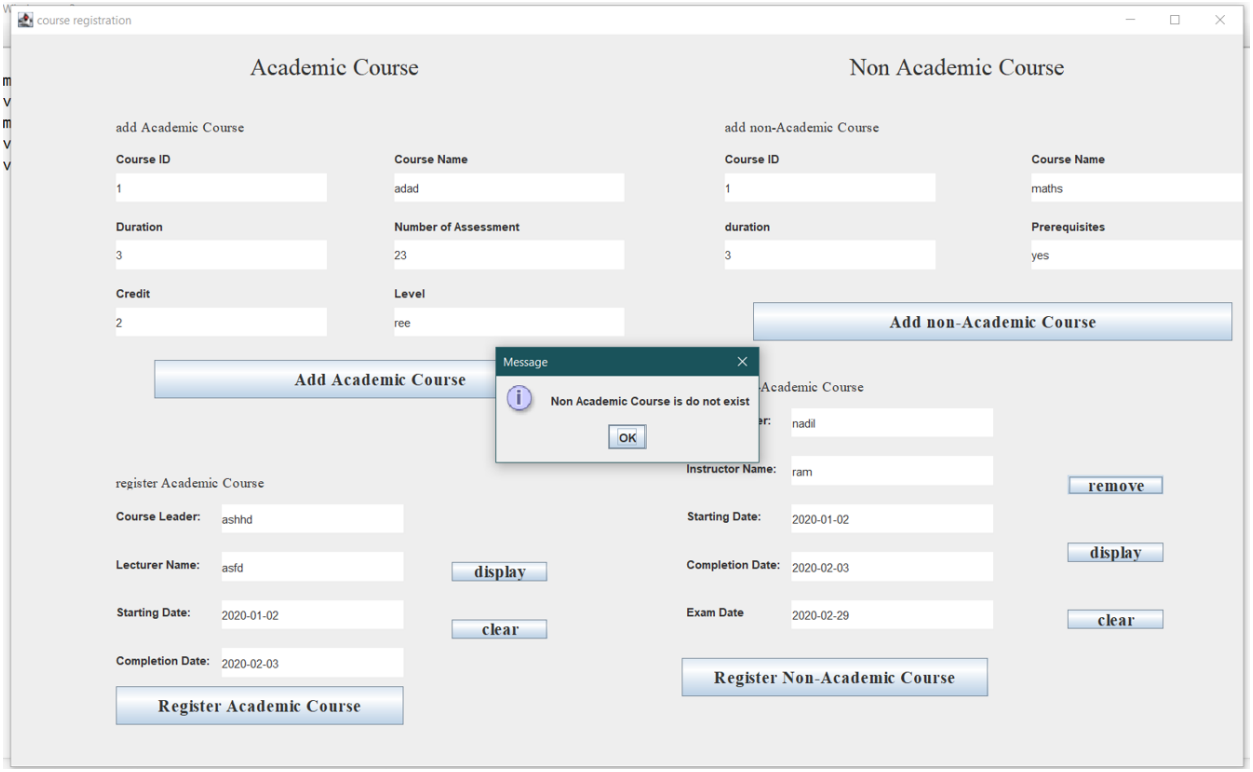


Figure 11: trying to remove non academic course which was already removed

6) Errors

Logical error

An error occurred while running the program. The program was not displayed on the computer. Later I found that frame was not set to visible.

```
    //to make frame visible
    frame.add(panel); //adding panel in the frame
    frame.setSize(1800,1000); //set size of the frame
    frame.setLocationRelativeTo(null); //set location of the frame to center
    //closing frame on clicking cross
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

Figure 12: logical error, frame.set visible(true) missing

After that I added the syntax frame.set visible(true), problem was solved and frame appered when running the program.

```
        //to make frame visible
        frame.add(panel); //adding panel in the frame
        frame.setSize(1800,1000); //set size of the frame
        frame.setLocationRelativeTo(null); //set location of the frame to center
        frame.setVisible(true); //make frame visible
        //closing frame on clicking cross
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```

Figure 13:logical error solved by adding frame.setVisibility(true)

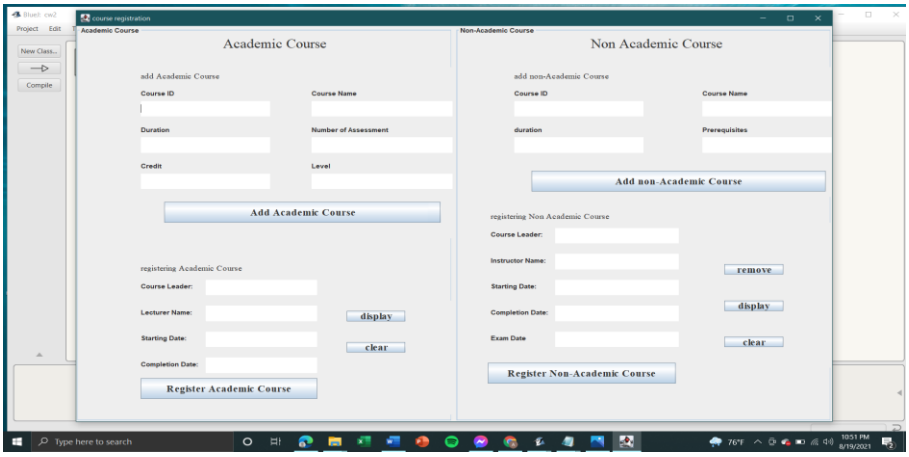


Figure 14:running the program after solving logical error

Syntax error

Some error in syntax of the program occurred after I completed the program. blueJ indicated the problem underlining the part in syntax where there was a problem and then the syntax was corrected. Then program ran smoothly. “;” sign was missing in the syntax

```
        //adding font
        title1.setFont(f2);

        //adding label in the left_a2 panel
        left_a2.add(title1);
        left_a2.add(alblLeader);
        left_a2.add(alblLecturer)
        left_a2.add(alblStart); ';' expected
        left_a2.add(alblCompleti
        //JTextField
        atxtLeader = new JTextField();
        atxtLecturer = new JTextField();
        atxtStart = new JTextField();
        atxtCompletion = new JTextField();

        //setting border;
        atxtLeader.setBorder(b1);
        atxtLecturer.setBorder(b1);
```

Figure 15: syntax error

The error was solved by adding “;” symbol in the syntax

```
title1.setFont(f2);

//adding label in the left_a2 panel
left_a2.add(title1);
left_a2.add(alblLeader);
left_a2.add(alblLecturer);
left_a2.add(alblStart);
left_a2.add(alblCompletion);

//JTextField
atxtLeader = new JTextField();
atxtLecturer = new JTextField();
atxtStart = new JTextField();
atxtCompletion = new JTextField();


//setting border;
atxtLeader.setBorder(b1);
atxtLecturer.setBorder(b1);
```

Figure 16:syntax error solved

Semantic error

There was an semantic error saying “incompatible types:java.lang.String cannot be converted into int” . error was found after checking the child class where placement of duration was wrong.

```
/*checking if arraylist is empty or not using If isEmpty()method
 * inside if loop which return boolean and new object is created
 * else if course is already exist or not
*/

if(courselist.isEmpty()){

    ac = new AcademicCourse(acourseID,acourseName,duration,alevel,credit,assessment);
    courselist.add(ac);
    pane = new JOptionPane();
    pane.showMessageDialog(new JFrame(),"Academic Couse has been successfully added");
}else{

    for(Course c:courselist){
        /*to check the course already exist or not
```

incompatible types: java.lang.String cannot be converted to int

saved
Errors: 4

Figure 17: semantic error

The error was solved by changing the placement of parameter duration.

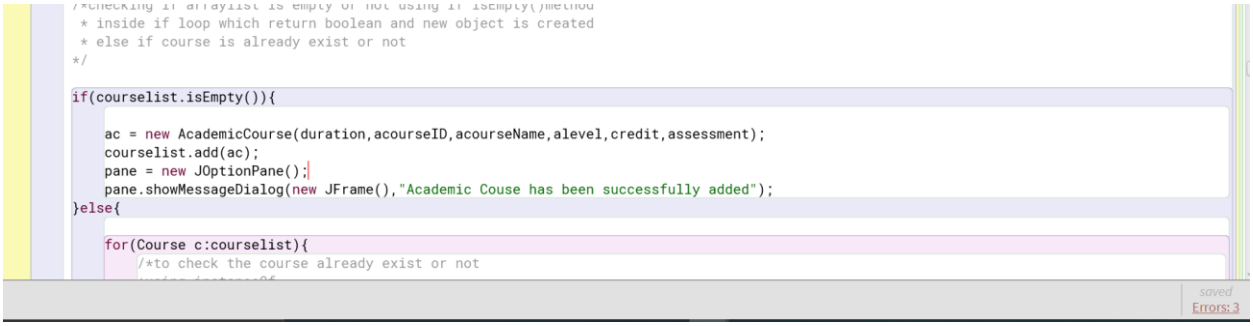


Figure 18: semantic error solved

7) Conclusion

In this coursework , we learned about many new things related to GUI. This coursework focused on GUI Interfaces mainly.

when I saw the coursework 2 questions, I was really excited because this was my first GUI program that was used in real world making.

As compared to the coursework1, coursework2 was more tough and difficult for me to complete on time. I was worried about finishing it in time and I struggled a-bit. In this coursework we were asked to create a GUI for academic course and non academic courses. We were assigned to add, register and remove the courses. We had to play with JPanles, JFrame, JTexFields and so on to create a design for the GUI. These topics were totally new and exciting. The hardest part for me doing the coursework was to debug the code. There were tons of bugs in my code at starting and Due to this covid pandemic situation, consultation with my teachers for support were not easy. A lot of time was consumed in order to make my code bug free. But, overall, I actually had fun doing this coursework. I got to learn a lot of new stuff about GUI interfaces, buttons in GUI, and there working mechanism during this coursework.

A lot of difficulties and problems were faced during this coursework. This topic was totally new for me, and during this pandemic while staying home and attending classes online, it was hard for me to understand and learn as I would do in normal days. Consulting lecturers and module leaders were also difficult.
but with the constant support of module leader, surfing through the internet and researching, I was able to complete the coursework on time and make it bug free. It was difficult for students to learn in online classes compared with physical as the whole world is struggling with COVID-19.

Internet helped a-lot, I emailed my module leader consulting support and I was guided in the right path. With proper effort, hardwork and support, I finished this coursework in time .

8) Appendix

INGcollege

```
import java.util.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import javax.swing.border.*;


public class INGcollege implements ActionListener
{
   //declaring instance variable
```

```
//for window
//JFrame
private JFrame frame;

//JPanel
private JPanel panel;
private JPanel left_panel;
private JPanel left_a1;
private JPanel left_a2;
private JPanel left_a3;
private JPanel right_panel;
private JPanel right_p1;
private JPanel right_p2;
private JPanel right_p3;

//---For UI Component----
//JLabel
 private JLabel title;
 private JLabel title1;
 //for academic
 private JLabel labelAcademic;
 private JLabel alblID,alblName,alblDuration,alblLevel,alblCredit,alblAssessment;
 private JLabel alblLeader,alblLecturer,alblStart,alblCompletion;
 //nonacademic
 private JLabel labelNonAcademic;
 private JLabel nlblID,nlblName,nlblDuration,nlblPrerequisites;
 private JLabel nlblLeader,nlblInstructor,nlblStart,nlblCompletion,nlblexam;

//TextFields
//for academic course
```

```
 private JTextField atextID;
 private JTextField atextName;
 private JTextField atextLeader;
 private JTextField atextDuration;
 private JTextField atextLecturer;
 private JTextField atextLevel;
 private JTextField atextCredit;
 private JTextField atextStart;
 private JTextField atextCompletion;
 private JTextField atextAssessment;

 //for frame of non academic
 private JTextField ntextID;
 private JTextField ntextName;
 private JTextField ntextDuration;
 private JTextField ntextLeader;
 private JTextField ntextInstructor;
 private JTextField ntextStart;
 private JTextField ntextCompletion;
 private JTextField ntextExam;
 private JTextField ntextPrerequisites;

//JButton
 private JButton bttnAddAca;
 private JButton bttnAddNonAca;
 private JButton bttnRegisterAca;
 private JButton bttnRegisterNonAca;
 private JButton bttnRemoveNonAca;
 private JButton bttnDisplayAca;
 private JButton bttnDisplayNonAca;
 private JButton bttnClearAca;
```

```
 private JButton bttnClearNonAca;
//JOptionPane
 private JOptionPane pane;

//Declaring ArrayList this use Course parent class as datatype
public ArrayList<Course> courselist;
AcademicCourse ac;
nonAcademicCourse nac;

 //constructor
public void INGCollege(){
    //---for window------
    //JFrame
    frame = new JFrame("course registration");

    //intializing ArrayList
    courselist = new ArrayList<Course>();

    //Font
    Font f1 = new Font("Times New Roman",Font.PLAIN,25);
    Font f2 = new Font("Times New Roman",Font.PLAIN,15);
    Font f3 = new Font("Times New Roman",Font.BOLD,18);



    //Border
    Border b1 = BorderFactory.createEmptyBorder();


    //Panel
    panel = new JPanel();
```

```
panel.setBounds(0,0,1400,700);
//setting gridlayout
panel.setLayout(new GridLayout(1,2,5,5));
//left_panel
left_panel = new JPanel();

left_panel.setLayout(null); //setting layout manager
//right_panel
right_panel = new JPanel();

right_panel.setLayout(null); //setting layout manager
//adding in panel
panel.add(left_panel);
panel.add(right_panel);


//------Different component for left panel------
left_a1 = new JPanel();
left_a2 = new JPanel();
left_a3 = new JPanel();



//setting bounds for panels in left_panel
left_a1.setBounds(100,80,570,330);
left_a2.setBounds(100,450,320,300);
left_a3.setBounds(450,530,220,220);

//setting layout manager
left_a1.setLayout(null);
left_a2.setLayout(null);
```

```
left_a3.setLayout(null);

//adding in the panel in left_panel
left_panel.add(left_a1);
left_panel.add(left_a2);
left_panel.add(left_a3);


//-----UI component for left_panel----
labelAcademic = new JLabel("Academic Course");
labelAcademic.setBounds(250,20,250,30); //setting bounds
labelAcademic.setFont(f1);  //setting font

left_panel.add(labelAcademic); //adding in the left_panel
//-----End of UI Component for right_panel---------



//-----UI Component for left_a1-------
//JLabel
title = new JLabel("add Academic Course");
alblID = new JLabel("Course ID"); ;
alblName = new JLabel("Course Name");
alblDuration = new JLabel("Duration");;
alblAssessment = new JLabel("Number of Assessment");
alblLevel = new JLabel("Level");
alblCredit = new JLabel("Credit");

//setting bounds for JLabel
title.setBounds(10,10,300,15);
alblID.setBounds(10,40,150,20);
```

```
alblName.setBounds(300,40,150,20);
alblDuration.setBounds(10,110,150,20);
alblAssessment.setBounds(300,110,150,20);
alblCredit.setBounds(10,180,150,20);
alblLevel.setBounds(300,180,150,20);



//adding font
title.setFont(f2);

//adding label in panel left_a1
left_a1.add(title);
left_a1.add(alblID);
left_a1.add(alblName);
left_a1.add(alblDuration);
left_a1.add(alblAssessment);
left_a1.add(alblCredit);
left_a1.add(alblLevel);

//JTextField
atextID = new JTextField();
atextName = new JTextField();
atextDuration = new JTextField();
atextAssessment = new JTextField();
atextCredit = new JTextField();
atextLevel = new JTextField();

//setting border;
atextID.setBorder(b1);
atextName.setBorder(b1);
```

```
atextDuration.setBorder(b1);
atextAssessment.setBorder(b1);
atextCredit.setBorder(b1);
atextLevel.setBorder(b1);

//setting bounds
atextID.setBounds(10,65,220,30);
atextName.setBounds(300,65,240,30);
atextDuration.setBounds(10,135,220,30);
atextAssessment.setBounds(300,135,240,30);
atextCredit.setBounds(10,205,220,30);
atextLevel.setBounds(300,205,240,30);

//adding JTextField
left_a1.add(atextID);
left_a1.add(atextName);
left_a1.add(atextDuration);
left_a1.add(atextAssessment);
left_a1.add(atextCredit);
left_a1.add(atextLevel);

//JButton
bttnAddAca = new JButton("Add Academic Course");
//setting bounds
bttnAddAca.setBounds(50,260,470,40);


//setting font
bttnAddAca.setFont(f3);
//adding button to left_a1
left_a1.add(bttnAddAca);
```

```
//----UI Component for left_a1 ends here



//------UI Component for left_a2 start------
//JLabel
title1 = new JLabel("register Academic Course");
alblLeader= new JLabel("Course Leader:");
alblLecturer = new JLabel("Lecturer Name:");
alblStart = new JLabel("Starting Date:");
alblCompletion = new JLabel("Completion Date:");

//setting bounds
title1.setBounds(10,10,300,15);
alblLeader.setBounds(10,40,150,25);
alblLecturer.setBounds(10,90,150,25);
alblStart.setBounds(10,140,150,25);
alblCompletion.setBounds(10,190,150,25);



//adding font
title1.setFont(f2);

//adding label in the left_a2 panel
left_a2.add(title1);
left_a2.add(alblLeader);
left_a2.add(alblLecturer);
left_a2.add(alblStart);
left_a2.add(alblCompletion);

//JTextField
```

```
atextLeader = new JTextField();
atextLecturer = new JTextField();
atextStart = new JTextField();
atextCompletion = new JTextField();


//setting border;
atextLeader.setBorder(b1);
atextLecturer.setBorder(b1);
atextStart.setBorder(b1);
atextCompletion.setBorder(b1);

//setting bounds
atextLeader.setBounds(120,40,190,30);
atextLecturer.setBounds(120,90,190,30);
atextStart.setBounds(120,140,190,30);
atextCompletion.setBounds(120,190,190,30);

//adding JTextField in left_a2
left_a2.add(atextLeader);
left_a2.add(atextLecturer);
left_a2.add(atextStart);
left_a2.add(atextCompletion);

//JButton
bttnRegisterAca = new JButton("Register Academic Course");
//setting bounds
bttnRegisterAca.setBounds(10,230,300,40);
```

```
//setting font
bttnRegisterAca.setFont(f3);
//adding button to left_a2
left_a2.add(bttnRegisterAca);
//-------End of UI component for left_a2 panel------


//-------UI Component for left_a3 panel----
//JButton
bttnDisplayAca = new JButton("display");
bttnClearAca = new JButton("clear");

//setting bounds
bttnDisplayAca.setBounds(10,20,100,20);
bttnClearAca.setBounds(10,80,100,20);


//setting font
bttnDisplayAca.setFont(f3);
bttnClearAca.setFont(f3);
//adding JButton in left_a3
left_a3.add(bttnDisplayAca);
left_a3.add(bttnClearAca);
//----------end for left_a3 panel-------


//-------------Different UI component for right_p1 panel-----
right_p1 = new JPanel();
right_p2 = new JPanel();
right_p3 = new JPanel();
```

```
//setting bounds for panels in right_panel
right_p1.setBounds(90,80,600,250);
right_p2.setBounds(50,350,360,400);
right_p3.setBounds(450,380,250,280);

//setting layout manager
right_p1.setLayout(null);
right_p2.setLayout(null);
right_p3.setLayout(null);

//adding in the panel in right_panel
right_panel.add(right_p1);
right_panel.add(right_p2);
right_panel.add(right_p3);


//-----UI component for right_panel----
labelNonAcademic = new JLabel("Non Academic Course");
labelNonAcademic.setBounds(230,20,300,30); //setting bounds
labelNonAcademic.setFont(f1);  //setting font
right_panel.add(labelNonAcademic); //adding in the right_panel
//-----End of UI Component for right_panel---------


//-----UI Component for right_p1-------

//JLabel
title = new JLabel("add non-Academic Course");
```

```
nlblID = new JLabel("Course ID"); ;
nlblName = new JLabel("Course Name");
nlblDuration = new JLabel("duration");;
nlblPrerequisites = new JLabel("Prerequisites");

//setting bounds for JLabel
title.setBounds(10,10,300,15);
nlblID.setBounds(10,40,150,20);
nlblName.setBounds(330,40,150,20);
nlblDuration.setBounds(10,110,150,20);
nlblPrerequisites.setBounds(330,110,150,20);



//adding font
title.setFont(f2);

//adding in panel right_p1
right_p1.add(title);
right_p1.add(nlblID);
right_p1.add(nlblName);
right_p1.add(nlblDuration);
right_p1.add(nlblPrerequisites);

//JTextField
ntextID = new JTextField();
ntextName = new JTextField();
ntextDuration = new JTextField();
ntextPrerequisites = new JTextField();

//setting border;
ntextID.setBorder(b1);
```

```
ntextName.setBorder(b1);
ntextDuration.setBorder(b1);
ntextPrerequisites.setBorder(b1);

//setting bounds
ntextID.setBounds(10,65,220,30);
ntextName.setBounds(330,65,240,30);
ntextDuration.setBounds(10,135,220,30);
ntextPrerequisites.setBounds(330,135,240,30);

//adding JTextField
right_p1.add(ntextID);
right_p1.add(ntextName);
right_p1.add(ntextDuration);
right_p1.add(ntextPrerequisites);

//JButton
bttnAddNonAca = new JButton("Add non-Academic Course");
//setting bounds
bttnAddNonAca.setBounds(40,200,500,40);


//setting font
bttnAddNonAca.setFont(f3);
//adding button to right_p1
right_p1.add( bttnAddNonAca);


//---UI component for right_p2;
//JLabel
title1 = new JLabel("register Non-Academic Course");
```

```
nlblLeader= new JLabel("Course Leader:");
nlblInstructor = new JLabel("Instructor Name:");
nlblStart = new JLabel("Starting Date:");
nlblCompletion = new JLabel("Completion Date:");
nlblexam = new JLabel("Exam Date");

//setting bounds
title1.setBounds(10,10,300,15);
nlblLeader.setBounds(10,40,150,25);
nlblInstructor.setBounds(10,90,150,25);
nlblStart.setBounds(10,140,150,25);
nlblCompletion.setBounds(10,190,150,25);
nlblexam.setBounds(10,240,150,25);



//adding font
title1.setFont(f2);

//adding labels in the right_p2 panel
right_p2.add(title1);
right_p2.add(nlblLeader);
right_p2.add(nlblInstructor);
right_p2.add(nlblStart);
right_p2.add(nlblCompletion);
right_p2.add(nlblexam);

//JTextFields
ntextLeader = new JTextField();
ntextInstructor = new JTextField();
ntextStart = new JTextField();
ntextCompletion = new JTextField();
```

```
ntextExam = new JTextField();

//setting border;
ntextLeader.setBorder(b1);
ntextInstructor.setBorder(b1);
ntextStart.setBorder(b1);
ntextCompletion.setBorder(b1);
ntextExam.setBorder(b1);

//setting bounds
ntextLeader.setBounds(120,40,210,30);
ntextInstructor .setBounds(120,90,210,30);
ntextStart.setBounds(120,140,210,30);
ntextCompletion.setBounds(120,190,210,30);
ntextExam.setBounds(120,240,210,30);

//adding JTextFields in right_p2
right_p2.add(ntextLeader);
right_p2.add(ntextInstructor );
right_p2.add(ntextStart);
right_p2.add(ntextCompletion);
right_p2.add(ntextExam);

//JButton
bttnRegisterNonAca = new JButton("Register Non-Academic Course");
//setting bounds
bttnRegisterNonAca.setBounds(5,300,320,40);
//setting font
bttnRegisterNonAca.setFont(f3);
//adding button to right_p2
right_p2.add(bttnRegisterNonAca);
```

```
//-------End of UI component for right_p2 panel------

//-------UI Component for right_p3 panel----
//JButton
bttnDisplayNonAca = new JButton("display");
bttnClearNonAca = new JButton("clear");
bttnRemoveNonAca = new JButton("remove");
//setting bounds

bttnRemoveNonAca.setBounds(8,80,100,20);
bttnDisplayNonAca.setBounds(8,150,100,20);
bttnClearNonAca.setBounds(8,220,100,20);

//setting font
bttnDisplayNonAca.setFont(f3);
bttnClearNonAca.setFont(f3);
bttnRemoveNonAca.setFont(f3);

//adding JButton in right_p3
right_p3.add(bttnDisplayNonAca);
right_p3.add(bttnClearNonAca);
right_p3.add(bttnRemoveNonAca);




//---creating listener-----------
bttnAddAca.addActionListener(this);
bttnAddNonAca.addActionListener(this);
bttnRegisterAca.addActionListener(this);
bttnRegisterNonAca.addActionListener(this);
```

```
    bttnDisplayAca.addActionListener(this);
    bttnDisplayNonAca.addActionListener(this);
    bttnClearNonAca.addActionListener(this);
    bttnClearAca.addActionListener(this);
    bttnRemoveNonAca.addActionListener(this);
    //----End of creating listener---------




    //to make frame visible
    frame.add(panel); //adding panel in the frame
    frame.setSize(1300,800); //set size of the frame
    frame.setLocationRelativeTo(null); //set location of the frame to center
    frame.setVisible(true); //make frame visible
    //closing frame on clicking cross
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }

//method actionPerformed for implement in ActionListener
public void actionPerformed(ActionEvent e){
    //using getText to get text from JTextField

    //from Academic
        String acourseID = atextID.getText();
        String acourseName = atextName.getText();
        String aDuration = ntextDuration.getText();
        String alevel = atextLevel.getText();
        String acredit = atextCredit.getText();
        String anumberofassessment = atextAssessment.getText();
        String aleader = atextLeader.getText();
```

```
    String alecturer = atextLeader.getText();
    String astart = atextStart.getText();
    String acompletion = atextCompletion.getText();

//for Non-Academic
    String ncourseID = ntextID.getText();
    String ncourseName = ntextName.getText();
    String nDuration = ntextDuration.getText();
    String nprerequisites = ntextPrerequisites.getText();
    String nleader = ntextLeader.getText();
    String ninstructor = ntextInstructor.getText();
    String nstart = ntextStart.getText();
    String ncompletion = ntextCompletion.getText();
    String nexam = ntextExam.getText();

    //declaring variables for loop
    int duration,assessment,credit = 0;
    //setting isUnique variable in order to check courseID is unique or not
    boolean isUnique = false;




//functions for button
/*for bttnAddAca
 *create new object of AcademicCourse
 *add to an ArrayList of course class
*/

if (e.getSource().equals(bttnAddAca)){
```

```
        //exceptional handling
         try{

            duration = (int)Double.parseDouble(aDuration);
            assessment = (int)Double.parseDouble(anumberofassessment);
            credit = (int)Double.parseDouble(acredit);
         }catch(NumberFormatException ex){
             pane = new JOptionPane();
             pane.showMessageDialog(new JFrame(),"Please enter data in required
feilds");
             return;
         }

         /*checking if arraylist is empty or not using If isEmpty()method
          * inside if loop which return boolean and new object is created
          * else if course is already exist or not
         */

         if(courselist.isEmpty()){

             ac = new
AcademicCourse(duration,acourseID,acourseName,alevel,credit,assessment);
             courselist.add(ac);
             pane = new JOptionPane();
             pane.showMessageDialog(new JFrame(),"Academic Course addition
sucessful!");
         }else{

             for(Course c:courselist){
                /*to check the course already exist or not
                *using instanceOf
```

```
                *for AcademicCourse child class
                */

                if(c instanceof AcademicCourse){

                    if(c.getCourseID().equals(acourseID)){

                        pane = new JOptionPane();
                        pane.showMessageDialog(new JFrame(),"Course ID already
exist.Please give unique course ID");
                        return;
                    }else{
                        isUnique = true;
                    }
                }
            }

            if(isUnique == true){
                ac = new
AcademicCourse(duration,acourseID,acourseName,alevel,credit,assessment);
                courselist.add(ac);
                isUnique = false;

             }

        }
        pane = new JOptionPane();
        pane.showMessageDialog(new JFrame(),"Course has been successfully
added");
```

```
        }

       /*for bttnAddNonAca
        *create new object of Non-AcademicCourse
        *adding to an ArrayList of course class
       */

        if (e.getSource().equals(bttnAddNonAca)){

           //exceptional handling
           try{

              duration = (int)Double.parseDouble(nDuration);
            }catch(NumberFormatException ex){
                  pane = new JOptionPane();
                  pane.showMessageDialog(new JFrame(),"Please enter data in required
feild!");
                  return;
            }

            /*checking if arraylist is empty or not using If isEmpty()method
             * inside if loop which return boolean and new object is created
             * find else if course is already exist or not
            */
            if(courselist.isEmpty()){
                nac = new
nonAcademicCourse(duration,ncourseID,ncourseName,nprerequisites);
                courselist.add(nac);
                pane = new JOptionPane();
                pane.showMessageDialog(new JFrame(),"Non-Academic Couse has been
successfully added");
```

```
        }else{
            for(Course c:courselist){
                /*to check the course already exist or not
                * using instanceOf
                *for Non-AcademicCourse child class
                */
                if(c instanceof nonAcademicCourse){
                    if(c.getCourseID().equals(ncourseID)){
                        pane = new JOptionPane();
                        pane.showMessageDialog(new JFrame(),"Course ID already
exist.Please give another course ID!");
                        return;
                    }else{
                        isUnique=true;
                    }
                }
            }

            if(isUnique == true){
                    nac = new
nonAcademicCourse(duration,ncourseID,ncourseName,nprerequisites);
                    courselist.add(nac);
                    isUnique = false;

             }
            pane = new JOptionPane();
            pane.showMessageDialog(new JFrame(),"Course has been added
sucessfully");

        }
```

```
        //bttnRegisterAca
       /*compared the courseID
        *if courseID is valid then will be register to list
       */
      if(e.getSource().equals(bttnRegisterAca)){

        if(courselist.isEmpty()){

            pane = new JOptionPane();
            pane.showMessageDialog(new JFrame(),"Course list is empty");

          }else{

                for(Course c:courselist){
                /*to check the course already exist or not
                *downcast is done using instanceOf so it is
                *of AcademicCourse child class
                */
                   if(c.getCourseID().equals(acourseID)){

                      if(c instanceof AcademicCourse){
                        ac = (AcademicCourse)c; //downcasting to call register method

                         if(ac.getisRegistered()==false){
                            ac.register(acourseName,alecturer,astart,acompletion);
                            pane = new JOptionPane();
                            pane.showMessageDialog(new JFrame(),"Academic Course
is registered");
                            return;
                          }else{
```

```
                                pane = new JOptionPane();
                                pane.showMessageDialog(new JFrame(),"Academic Course
is already registered");

                            }

                        }else{
                            pane = new JOptionPane();
                            pane.showMessageDialog(new JFrame(),"CouseID has been
used for NonAcademicCourse");
                         }

                    }else{
                        pane = new JOptionPane();
                        pane.showMessageDialog(new JFrame(),"Couse ID is doesnt exist in
the list");
                    }
                }


            }

        }

          //bttnRegisterNonAca
           /*compared the courseID
            *if courseID is valid then will be register to list
           */
        if(e.getSource().equals(bttnRegisterNonAca)){

           if(courselist.isEmpty()){
```

```
                pane = new JOptionPane();
                pane.showMessageDialog(new JFrame(),"Course list is empty");

            }else{

                for(Course c:courselist){
                    /*to check the course whether it already exist or not
                    *downcast is done using instanceOf so it is
                    *of Non-AcademicCourse child class
                    */
                    if(c.getCourseID().equals(ncourseID)){

                        if(c instanceof nonAcademicCourse){
                                nac = (nonAcademicCourse)c; //downcasting to call register
method
                                if(nac.getisRegistered()==false){

nac.register(ncourseName,ninstructor,nstart,ncompletion,nexam);
                                    pane = new JOptionPane();
                                    pane.showMessageDialog(new JFrame(),"Non-Academic
Course is registered successfully!");
                                    return;
                                }else{
                                    pane = new JOptionPane();
                                    pane.showMessageDialog(new JFrame(),"Non-Academic
Course is already registered!");

                                }

                        }else{
```

```
                    pane = new JOptionPane();
                    pane.showMessageDialog(new JFrame(),"CouseID has been
used for AcademicCourse");
                }

            }else{
                pane = new JOptionPane();
                pane.showMessageDialog(new JFrame(),"CouseID is doesn't exist
in the list");
            }

        }
    }



}



//bttnRemoveNonAca button
/*When the button is pressed first it compared courseId and then
  * call remove method from NonAcademicCourse
  */
if(e.getSource().equals(bttnRemoveNonAca)){
    System.out.println("Inside remove");
    if(courselist.isEmpty()){

        pane = new JOptionPane();
```

```
                pane.showMessageDialog(new JFrame(),"Course list is empty");

            }else{
                for(Course c:courselist){
                    /*to check whether the course already exist or not
                     *downcast and instanceof used to make sure it is
                     *of NonAcademicCourse child class
                     */

                   if(c.getCourseID().equals(ncourseID)){

                       if(c instanceof nonAcademicCourse){

                           nac = (nonAcademicCourse)c; //downcasting to call register
method
                            if(nac.getisRemoved()==true){
                                pane = new JOptionPane();
                                pane.showMessageDialog(new JFrame(),"Non Academic
Course is do not exist");


                           }else{
                                nac.remove();
                               pane = new JOptionPane();
                               pane.showMessageDialog(new JFrame(),"Non-
Academic Course is removal sucessful");
                               return;
                            }

                       }else{
                           pane = new JOptionPane();
```

```
                                pane.showMessageDialog(new JFrame(),"CouseID has been
used for AcademicCourse");
                            }

                        }else{
                            pane = new JOptionPane();
                            pane.showMessageDialog(new JFrame(),"CouseID is not present in
the list");
                        }

                    }

            }
        }


        /*display button
         *Display button to display information regarding respective class
        */

        if (e.getSource().equals(bttnDisplayAca)){

            if (courselist.isEmpty()){
                pane = new JOptionPane();
                pane.showMessageDialog(new JFrame(),"Blank List,Please fill the form");
                System.out.println("CourseID is Blank");
              }else{
                    for(Course aac: courselist){
                        System.out.println("Displaying inside list--> Academic Course");

                        //for academic course
```

```
                if(aac instanceof AcademicCourse){
                    System.out.println("----------------------------------------");
                    ac = (AcademicCourse)aac; //downcasting to call display method
                    ac.display();
                 }

                }
            }

    }


if (e.getSource().equals(bttnDisplayNonAca)){

    if (courselist.isEmpty()==true){
        pane = new JOptionPane();
        pane.showMessageDialog(new JFrame(),"Empty List,Please fill the form");
        System.out.println("Courselist is empty");
    }
    else{
            for(Course nc: courselist){
                System.out.println("Displaying inside list--> Non Academic Course");

                //for nonacademic course
                if(nc instanceof nonAcademicCourse){
                    System.out.println("----------------------------------------");
                    nac = (nonAcademicCourse)nc; //downcasting to call display method
                    nac.display();
                }
            }
```

```
    }
 }

        //for Clear Button
    //when button is pressed all the data from textfields are removed
if(e.getSource().equals(bttnClearAca)){
        //using setText method and giving ("")empty string to textfield
        //from Academic course
       atextID.setText("");
       atextName.setText("");
       ntextDuration.setText("");
       atextLevel.setText("");
       atextCredit.setText("");
       atextAssessment.setText("");
       atextLeader.setText("");
       atextLecturer.setText("");
       atextStart.setText("");
       atextCompletion.setText("");

}

if(e.getSource().equals(bttnClearNonAca)){
       //for Non-Academic course
       ntextID.setText("");
       ntextName.setText("");
       ntextDuration.setText("");
       ntextPrerequisites.setText("");
       ntextLeader.setText("");
       ntextInstructor.setText("");
       ntextStart.setText("");
       ntextCompletion.setText("");
```

```
            ntextExam.setText("");

        }

        //------End of button functionality---------
    }

    public static void main(String args[]){
        INGcollege main=new INGcollege(); //creating class object to call instance method
        main.INGCollege(); //calling instance method
    }

}
```

Course

```
/**
 * Write a description of class Course here.
 *
 * @author (Asal Pandey)
 * @version (2021-05-22)
 */
public class Course
{
    //variable for Course class
    private int duration;
    private String courseID;
    private String courseLeader;
```

```
    private String courseName;

    //creating constructor to initialize the value of duration,courseID,course Leader and
coursename
    public Course(String courseID, String courseName,int duration)
    {
        this.courseID=courseID;
        this.courseName=courseName;
        this.courseLeader="";
        this.duration=duration;
    }

    //creating get and set method which is used to read and set the values of variables


    public String getCourseID()
   {
    return courseID;
    }
    public String getcourseName(){
     return courseName;
   }
   public String getcourseLeader(){
       return courseLeader;
   }
   public int getduration(){
       return duration;
   }
   public void setcourseLeader(String nameofthenewcoureLeader)
   {
   this.courseLeader=nameofthenewcoureLeader;
```

```
   }

   //creating method for displaying the information

   public void display()
   {
    System.out.println("the ID of the course is " + courseID);
    System.out.println("Name of the course is " + courseName);
    System.out.println("Duration of the course is " + duration);
    if (this.courseLeader==(""))
    {
     System.out.print("empty shell");
    }
    else
    {
        System.out.print("the leader of the course is "+courseLeader);
}
}
}
```

Academic Course

```
/**
 * Write a description of class AcademicCourse here.
 *
 * @author (Asal Pandey)
 * @version (2021-05-22)
 */
public class AcademicCourse extends Course
```

```
{
    //defining variables
  private int NumberOfAssesment;
  private String Lecturername; //empty
  private String Level;
  private int Credit;
  private String StartingDate; //empty
  private String CompletionDate; //empty
  private boolean isRegistered; //false
  //creating constructor for Academic course

    public AcademicCourse(int duration, String courseID, String courseName, String
level, int credit,int NumberOfAssesment){
    //super keyword is used for calling super class from the main class
        super(courseID,courseName,duration);
        this.NumberOfAssesment=NumberOfAssesment;
        this.Level=Level;
        this.Credit=Credit;
        this.Lecturername = "";
        this.StartingDate = "";
        this.CompletionDate = "";
        isRegistered = false;

}

 //creating get and set method which is used to read and set the values of variables
 public String getLecturername(){
        return Lecturername;
  }
  public void setLecturername(String newLecturername){
      this.Lecturername=newLecturername;
```

```
 }
 public int getNumberOfAssesment(){
      return NumberOfAssesment;
 }
 public void setNumberOfAssesment(int newNumberOfAssesment){
    this.NumberOfAssesment=newNumberOfAssesment;
 }
 public String getLevel(){
      return Level;
 }
  public int getCredit(){
      return Credit;
 }
 public String getStartingDate(){
      return StartingDate;
 }
 public String getCompletionDate(){
      return CompletionDate;

 }
 public boolean getisRegistered(){
      return isRegistered;
   }

public void register(String Lecturername,String  StartingDate,String
CompletionDate,String courseLeader){
     if(isRegistered==true)
        System.out.println(Lecturername+" has already started at"+StartingDate+" and
ends on"+CompletionDate);
       else{
        super.setcourseLeader(courseLeader);
```

```
        this.Lecturername=Lecturername;
        this.StartingDate=StartingDate;
        this.CompletionDate=CompletionDate;
        isRegistered=true;

      }
   }
   public void display(){
      super.display();
      if(isRegistered==true){

         System.out.println("the name of the lecturer is="+Lecturername+".");
         System.out.println("the level of the course is="+Level+".");
         System.out.println("total credit of the course is="+Credit+".");
         System.out.println("total no of assessment in this course
is="+NumberOfAssesment+".");
         System.out.println("starting date="+StartingDate+".");
         System.out.println("starting date="+CompletionDate+".");
 }
  }
}
```

Non-Academic course

```
/**
 * Write a description of class nonAcademicCourse here.
 *
 * @author (Asal Pandey)
 * @version (2021-05-22)
 */
```

```
public class nonAcademicCourse extends Course
{
    //defining variables
    private String Lecturername;
    private int duration;
    private String StartingDate;
    private String CompletionDate;
    private String ExamDate;
    private String prerequisite;
    private boolean isRegistered;
    private boolean isRemoved;
    //creating constructor for nonAcademicCourse.

    public nonAcademicCourse(int duration,String courseID,String courseName,String
prerequisite)
    {
        //super keyword is used for calling super class from the main class
        super(courseID,courseName,duration);
        this.prerequisite=prerequisite;
        this.StartingDate="";
        this.CompletionDate="";
        this.ExamDate="";
        isRegistered=false;
        isRemoved=false;

    }
     //creating get and set method which is used to read and set the values of variables
     public String getLecturername(){
       return Lecturername;
  }
  public int getduration()
```

```
{
    return duration;
}
public String getStartingDate(){
     return StartingDate;
}
public String getCompletionDate(){
     return CompletionDate;

}
public String getExamDate(){
     return ExamDate;

}
 public String getprerequisite(){
     return prerequisite;
}
public boolean getisRegistered(){
     return isRegistered;
  }
  public boolean getisRemoved(){
     return isRemoved;
  }

   public void setLecturername(String newLecturername){
   if(isRegistered==false)
   {
       this.Lecturername=newLecturername;
       System.out.println("lecturer name is changed");
   }
   else
```

```
    {
        System.out.println("lecturer is already registered");


    }
  }
  public void register(String Lecturername,String  StartingDate,String
CompletionDate,String courseLeader, String ExamDate){
      if(isRegistered==false)
      {
          setLecturername(Lecturername);
          isRegistered=true;
      }
      else
      {
          System.out.println("this course is already registered");
      }
    }
  public void remove()
  {
      if(isRemoved==true)
       System.out.println("this course is already been removed");
       else
       {
           super.setcourseLeader("");
           this.Lecturername="";
           this.StartingDate="";
           this.CompletionDate="";
           this.ExamDate="";
           isRegistered=false;
           isRemoved=true;
```

```
        }
    }
    public void display(){
          super.display();
          if(isRegistered==true){
           System.out.println("lecturer name is" +Lecturername);
           System.out.println("starting date is" +StartingDate);
           System.out.println("complition date is" +CompletionDate);
           System.out.println("exam date is" +ExamDate);
          }
      }
}
```