

Islington College



Programming CS4001NI

Coursework 1

Submitted By:

Abiral Joshi-Shrestha

ID: 17030696

Group: L1N3

Date: 22nd April, 2018

Submitted To:

Mr. Dhruva Sen

Lecturer, IT

1. Introduction

Java is a programming language, its target is to write a program once and then run that program on multiple operating systems. The JAVA program used in this project was created using Blue J.

The program is a class added to the project we had developed for the first part of the coursework. The aim to make a graphical user interface (GUI) for a system that stores details of IT Courses in our IT Training Courses. It contains a main method and can be tested using the command prompt.

This program can be used by any educational organization or institutes to manage the information required by them to maintain the proper functioning of their daily activities.

While doing this project, I learnt many things about the JAVA programming language. The study and work I did while doing this coursework helped me better understand JAVA, cleared my confusions and misunderstandings. I also learnt how to make programs compact and yet, effective. The knowledge I gained while preparing this project will be very helpful in my future career as a programmer.

2. Class Diagram



- ProfAdd : JButton
- Complete : JButton
- ProfEnrol : JButton
- CertAdd : JButton
- CertEnrol : JButton
- Clear : JButton
- Display : JButton

All methods

- + FormCreate() : void
- + actionPerformed(ActionEvent e) : void
- + main(String args[]) : void

3. Pseudocode

3.1. Pseudocode for formCreate()

```
List=new ArrayList();

    JFrame frame = new JFrame("Form");

    JLabel lblProfessionalcourse = new JLabel("Professional Course");

lblProfessionalcourse.setBounds(20,10,410,25);

frame.add(lblProfessionalcourse);

    END DO

DO

    JLabel lblEmpty = new JLabel("_____");

lblEmpty.setBounds(20,10,910,25);

frame.add(lblEmpty);

    END DO

DO

    JLabel lblCertificationcourse = new JLabel("Certification Course");

lblCertificationcourse.setBounds(20,360,410,25);

frame.add(lblCertificationcourse);

    END DO

DO

    JLabel lblY = new JLabel("_____");

lblY.setBounds(20,360,910,25);    frame.add(lblY);

    END DO

DO
```

```

        JLabel lblcourseName = new JLabel("Course Name: ");
lblcourseName.setBounds(30,50,110,25);      frame.add(lblcourseName);

        END DO DO      txtcourseName = new
JTextField();
txtcourseName.setBounds(150,45,610,35);
frame.add(txtcourseName);

        END DO

DO

        JLabel lblInstructor = new JLabel("Instructor Name: ");
lblInstructor.setBounds(30,80,110,25);
frame.add(lblInstructor);

        END DO DO      txtInstructor = new
JTextField();
txtInstructor.setBounds(150,80,610,35);
frame.add(txtInstructor);

        END DO

DO

        JLabel lbltotalHour = new JLabel("Total  Hours: ");
        lbltotalHour.setBounds(30,120,110,25); frame.add(lbltotalHour);

        END DO DO      txttotalHour = new
JTextField();
txttotalHour.setBounds(150,115,240,35);
frame.add(txttotalHour);

        END DO

```

DO

```
        JLabel lblDailyHour = new JLabel("Daily Hour: ");  
lblDailyHour.setBounds(390,120,110,25);  
frame.add(lblDailyHour);
```

```
        END DO DO          txtdailyHour = new  
JTextField();  
txtdailyHour.setBounds(485,115,275,35);  
frame.add(txtdailyHour);
```

END DO

DO

```
        JLabel lblFees = new JLabel("Course Fee: ");  
lblFees.setBounds(30,155,110,25);  
frame.add(lblFees);
```

END DO DO

```
        txtFees = new JTextField();  
        txtFees.setBounds(150,150,240,35); frame.add(txtFees);
```

END DO

DO

```
        JLabel lblCourseNo = new JLabel("Course No.: ");  
lblCourseNo.setBounds(30,225,110,25);  
frame.add(lblCourseNo);
```

```
        END DO DO          txtcourseNo = new  
JTextField();
```

```
txtcourseNo.setBounds(150,220,610,35);
frame.add(txtcourseNo);
    END DO
DO
    JLabel lblStudent = new JLabel("Student Name: ");
    lblStudent.setBounds(30,260,110,25);
    frame.add(lblStudent);
    END DO DO        txtStudent = new
    JTextField();
    txtStudent.setBounds(150,255,210,35);
    frame.add(txtStudent);
    END DO
DO
    JLabel lblDownPayment = new JLabel("Down Payment: ");
    lblDownPayment.setBounds(390,260,120,25);
    frame.add(lblDownPayment);
    END DO DO        txtDownPayment = new
    JTextField();
    txtDownPayment.setBounds(485,255,275,35);
    frame.add(txtDownPayment);
    END DO
DO
    JLabel lblEnroll = new JLabel("Enroll Date: ");
    lblEnroll.setBounds(30,295,110,25);
    frame.add(lblEnroll);
```



```

        END DO DO          txtEnroll = new
JTextField();
txtEnroll.setBounds(150,290,210,35);
frame.add(txtEnroll);

        END DO

DO

        JLabel lblRoom = new JLabel("Room No.: ");
lblRoom.setBounds(390,295,110,25);
frame.add(lblRoom);

        END DO

DO

        txtRoom = new JTextField();
txtRoom.setBounds(485,290,275,35);      frame.add(txtRoom);

        END DO

DO

        JLabel lblcertcourseName = new JLabel("Course Name: ");
lblcertcourseName.setBounds(30,395,110,25);
frame.add(lblcertcourseName);

        END DO DO          txtcertcourseName = new
JTextField();
txtcertcourseName.setBounds(150,390,610,35);
frame.add(txtcertcourseName);

        END DO

DO

```

```
JLabel lblcertInstructor = new JLabel("Instructor Name: ");
lblcertInstructor.setBounds(30,430,110,25);
frame.add(lblcertInstructor);

    END DO DO          txtcertInstructor = new
JTextField();
txtcertInstructor.setBounds(150,425,610,35);
frame.add(txtcertInstructor);

    END DO
DO

        JLabel lblcerttotalHour = new JLabel("Total Hour: ");
lblcerttotalHour.setBounds(30,465,110,25);
frame.add(lblcerttotalHour);

        END DO DO          txtcerttotalHour = new
JTextField();
txtcerttotalHour.setBounds(150,460,240,35);
frame.add(txtcerttotalHour);

        END DO
DO

        JLabel lblcertdailyHour = new JLabel("Daily Hour: ");
lblcertdailyHour.setBounds(390,465,110,25);
frame.add(lblcertdailyHour);

        END DO DO          txtcertdailyHour = new
JTextField();
```

```
txtcertdailyHour.setBounds(485,460,275,35);
frame.add(txtcertdailyHour);

    END DO

DO

    JLabel lblcertFees = new JLabel("Course Fee: ");
    lblcertFees.setBounds(30,500,110,25); frame.add(lblcertFees);

    END DO DO        txtcertFees = new
JTextField();
txtcertFees.setBounds(150,495,240,35);
frame.add(txtcertFees);

    END DO

DO

    JLabel lblAward = new JLabel("Awarded By: ");
    lblAward.setBounds(390,500,110,25);
    frame.add(lblAward);

    END DO DO        txtAward = new
JTextField();
txtAward.setBounds(485,495,275,35);
frame.add(txtAward);

    END DO

DO

    JLabel lblValid = new JLabel("Valid Till: ");
    lblValid.setBounds(30,535,110,25);
    frame.add(lblValid);
```

END DO DO

txtValid = new JTextField();

txtValid.setBounds(150,530,240,35); frame.add(txtValid);

END DO

DO

JLabel lblcertStudent = new JLabel("Student Name: ");

lblcertStudent.setBounds(30,605,110,25);

frame.add(lblcertStudent);

END DO DO txtcertStudent = new

JTextField();

txtcertStudent.setBounds(150,600,240,35);

frame.add(txtcertStudent);

END DO

DO

JLabel lblstartDate = new JLabel("Start Date: ");

lblstartDate.setBounds(30,640,110,25);

frame.add(lblstartDate);

END DO DO txtstartDate = new

JTextField();

txtstartDate.setBounds(150,635,240,35);

frame.add(txtstartDate);

END DO

DO

JLabel lblexamDate = new JLabel("Exam Date: ");

lblexamDate.setBounds(390,640,110,25);

```
        frame.add(lblexamDate);

    END DO DO        txtexamDate = new
    JTextField();
    txtexamDate.setBounds(485,635,275,35);
    frame.add(txtexamDate);

    END DO

DO

    JLabel lblexamCenter = new JLabel("Exam Center: ");
    lblexamCenter.setBounds(30,675,110,25);
    frame.add(lblexamCenter);

    END DO DO        txtexamCenter = new
    JTextField();
    txtexamCenter.setBounds(150,670,610,35);
    frame.add(txtexamCenter);

    END DO

DO

    JLabel lblcertCourseNo = new JLabel("Course No: ");
    lblcertCourseNo.setBounds(30,580,110,25);
    frame.add(lblcertCourseNo);

    END DO

DO

    txtcertCourseNo = new JTextField();
    txtcertCourseNo.setBounds(150,570,610,35);    frame.add(txtcertCourseNo);

    END DO
```

DO

```
    ProfAdd = new JButton("Add");  
    ProfAdd.setBounds(650,185,110,35);  
    frame.add(ProfAdd);  
    frame.setSize(400,400);  
    ProfAdd.addActionListener(this);
```

END DO

DO

```
    Complete = new JButton("Complete");  
    Complete.setBounds(440,325,160,35);  
    frame.add(Complete);  
    frame.setSize(400,400);  
    Complete.addActionListener(this);
```

END DO

DO

```
    ProfEnrol = new JButton("Enroll Student");  
    ProfEnrol.setBounds(600,325,160,35);  
    frame.add(ProfEnrol);  
    frame.setSize(400,400);  
    ProfEnrol.addActionListener(this);
```

END DO

DO

```
        CertAdd = new JButton("Add");
CertAdd.setBounds(650,530,110,35);
frame.add(CertAdd);
frame.setSize(400,400);

        CertAdd.addActionListener(this);
    END DO
DO

        CertEnrol = new JButton("Enroll");
CertEnrol.setBounds(600,705,160,35);
frame.add(CertEnrol);
frame.setSize(400,400);

        CertEnrol.addActionListener(this);
    END DO
DO

        Clear = new JButton("Clear");
Clear.setBounds(600,740,160,35);
frame.add(Clear);
frame.setSize(400,400);

        Clear.addActionListener(this);
    END DO
DO

        Display = new JButton("Display All");
Display.setBounds(440,705,160,35);
frame.add(Display);      frame.setSize(400,400);
```

```
        Display.addActionListener(this);  
    END DO DO  
frame.setSize(900,900);  
frame.setLayout(null);  
frame.setVisible(true);  
    END DO
```

3.2. Pseudocode for actionPerformed(ActionEvent e)

```
DO  
if (e.getSource()==ProfAdd)  
    {  
try  
    {
```



```

        String courseName= txtcourseName.getText();
String instructorName= txtInstructor.getText();          int
courseFee= Integer.parseInt(txtFees.getText());          int
totalHours= Integer.parseInt(txttotalHour.getText());    int
dailyHours= Integer.parseInt(txtdailyHour.getText());

        END DO

DO

        Professional prof = new Professional(courseName, instructorName,
courseFee, totalHours, dailyHours);

        END DO

DO

        List.add(prof);

        END DO

DO

txtcourseName.setText("");
txtInstructor.setText("");
txtAward.setText("");
txtValid.setText("");
txtFees.setText("");
txttotalHour.setText("");

        END DO

        DO

                JOptionPane.showMessageDialog(null,"Added Success!");

        }

```

```

        catch(Exception exc)
        {
            JOptionPane.showMessageDialog(null,"Error, Try Again!");
        }
    }
END DO

DO      if
(e.getSource()==CertAdd)
    {
try
    {
        String courseName= txtcertcourseName.getText();
String instructorName= txtcertInstructor.getText();      int
totalHour= Integer.parseInt(txtcerttotalHour.getText());
int courseFee= Integer.parseInt(txtcertFees.getText());

        String certificateAwardedBy= txtAward.getText();
        String validTill=txtValid.getText();


        Certification cert = new Certification(courseName, instructorName, totalHour,
courseFee, certificateAwardedBy, validTill);


        List.add(cert);


        txtcertcourseName.setText("");
txtcertInstructor.setText("");

```

```
txtcerttotalHour.setText("");
```

```
txtcertFees.setText("");
```

```
txtAward.setText("");
```

```
txtValid.setText("");
```

```
txtcertdailyHour.setText("");
```

```
    JOptionPane.showMessageDialog(null,"Added Success!");
```

```
}
```

```
catch(Exception exc)
```

```
{
```

```
    JOptionPane.showMessageDialog(null,"Error!!");
```

```
}
```

```
}
```

```
END DO
```

```
DO
```

```
if(e.getSource()==ProfEnrol)
```

```
{
```

```
try
```

```
{
```

```

        int CourseNo = Integer.parseInt(txtcourseNo.getText());
        Professional prof = (Professional) List.get(CourseNo);          int
        downPayment = Integer.parseInt(txtdownPayment.getText());
        if(CourseNo >= 0|| CourseNo < List.size())
        {
            if(List.get(CourseNo) instanceof Professional)
            {

                prof.enrolStudent(txtStudent.getText(),txtEnroll.getText(),downPayment,txtR
oom.getText());

                JOptionPane.showMessageDialog(null,"Student successfully enrolled!");
            }
        }
        else
        {
            JOptionPane.showMessageDialog(null," Course number not available
");
        }
    }
    else
    {
        JOptionPane.showMessageDialog(null," Course number not available ");
    }
}

catch(NumberFormatException exec)

```

```

    {
        JOptionPane.showMessageDialog(null," Enter a number ");
    }
    catch(Exception exec)
    {
        JOptionPane.showMessageDialog(null," Error ");
    }
}
END DO

```

```

DO
if(e.getSource()==CertEnrol)
    {
try
{
int
courseNo
1 =
Integer.p
arseInt(tx
tcertCour
seNo.get
Text());

```

```

        int downPayment = Integer.parseInt(txtdownPayment.getText());
if(courseNo1 >= 0|| courseNo1 < List.size())
    {
        if(List.get(courseNo1) instanceof Certification)
        {
            Certification cert = (Certification) List.get(courseNo1);

cert.enrolStudent(txtStudent.getText(),txtstartDate.getText(),txtEnroll.getText
(),txtRoom.getText());

            JOptionPane.showMessageDialog(null,"Student successfully enrolled!");
        }
else
    {
        JOptionPane.showMessageDialog(null," Course number not available
");
    }
}
else
    {
        JOptionPane.showMessageDialog(null," Course number not available ");
    }

}

catch(NumberFormatException exec)
{

```

```

        JOptionPane.showMessageDialog(null," Enter a number ");
    }
    catch(Exception exec)
    {
        JOptionPane.showMessageDialog(null," Error ");
    }
}
END DO
DO
if (e.getSource()==Display)
{
    for(Course loop : List)
    {
        if (loop instanceof Professional)
        {
            Professional prof = (Professional) loop;
            prof.display();
        }
    }
    for(Course loop : List)
    {
        if (loop instanceof Certification)
        {

```

```

        Certification cert = (Certification) loop;
cert.display();
    }
}
}
END DO DO
    if(e.getSource() == Complete)
    {
try
    {
        int courseNo = Integer.parseInt(txtcourseNo.getText());
Professional prof = (Professional)List.get(courseNo);
if(List.get(courseNo) instanceof Professional)
    {
        prof.courseCompletion();
        JOptionPane.showMessageDialog(null,"The course has been completed");

        txtStudent.setText(" ");
txtcourseNo.setText(" ");          txtEnroll.setText("
");          txtRoom.setText(" ");
txtdownPayment.setText(" ");

    }
else

```



```

        {
            JOptionPane.showMessageDialog(null, "Course number is not available");
        }
    }
    catch(NumberFormatException exec)
    {
        JOptionPane.showMessageDialog(null, "Enter a number");
    }
    catch(Exception exec)
    {
        JOptionPane.showMessageDialog(null, "Error");
    }
}

END DO DO

if( e.getSource() == Clear)
{
    txtcourseName.setText("");
    txtInstructor.setText("");
    txttotalHour.setText("");
    txtdailyHour.setText("");
    txtFees.setText("");
    //txtcourseStart.setText("");
    txtcourseNo.setText("");
}

```

```
txtStudent.setText("");
txtdownPayment.setText("");
    txtEnroll.setText("");
txtRoom.setText("");
txtcertcourseName.setText("");
txtcertInstructor.setText("");
txtcerttotalHour.setText("");
txtcertdailyHour.setText("");
txtcertFees.setText("");
txtAward.setText("");
txtValid.setText("");
txtcertStudent.setText("");
txtexamDate.setText("");
txtstartDate.setText("");
txtexamCenter.setText("");
txtcertCourseNo.setText("");
END DO
```

3.3 Pseudocode for main (String args[])

```
DO

    new TrainingInstitute().formCreate();

END DO
```

4. Method Description

4.1. FormCreate()

It is the main method where various text field and buttons of the GUI are declared.

4.2. actionPerformed(ActionEvent e)

It is the method where the actions performed by the buttons are declared.

4.3. main(String args[])s

It is the method that calls the main method and displays the output result.

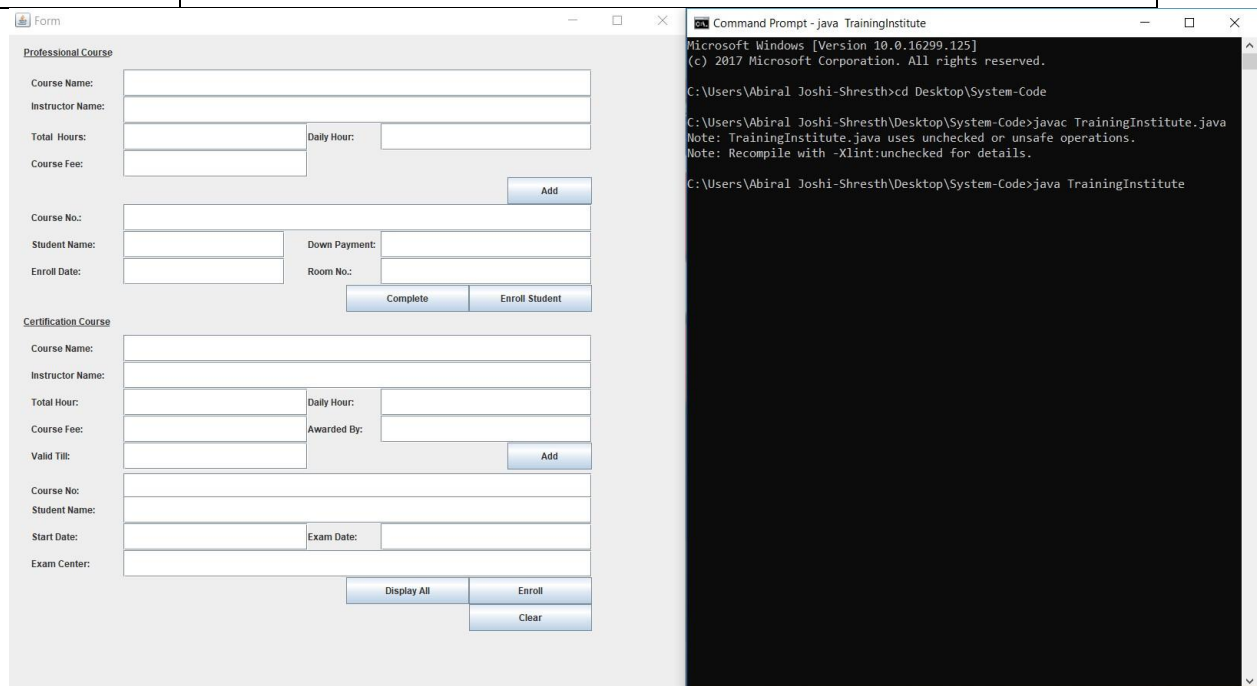
5. Testing

5.1 Test 1

Objective	To compile the program and run using the command prompt.
Action	"javac TrainingInstitute.java" was entered for compilation. "java TrainingInstitute" was entered to run the program.
Expected Result	GUI will be displayed.
Actual Result	GUI was displayed.

Conclusion

Test successful.



5.2 Test 2

Objective	To show evidence of adding courses to professional course list.
Action	Courses were added to Professional course list, generators were added to Certification course list, student was enrolled to Professional course, the complete button was checked and student was enrolled to Certification course.
Expected Result	Courses will be added to Professional course list. Generators will be added to Certification course list. Student will be enrolled to Professional course. Courses will be completed. Student will be enrolled to Certification course.

Actual Result	<p>Courses was added to Professional course list.</p> <p>Generators was added to Certification course list.</p> <p>Student was enrolled to Professional course.</p> <p>Courses was completed.</p> <p>Student was enrolled to Certification course.</p>
Conclusion	Test successful.

Professional Course

Course Name: Science

Instructor Name: Albert Einstein

Total Hours: 120

Daily Hour: 2

Course Fee: 10000

Add

Professional Course

Course Name:

Instructor Name:

Total Hours:

Daily Hour: 2

Course Fee:

Add

Course No.:

Student Name:

Enroll Date:

ROOM NO.:

Message

Added Success!

OK

Certification Course

Course Name: Science

Instructor Name: Newton

Total Hour: 120

Daily Hour: 2

Course Fee: 10000

Awarded By: LondonMet

Valid Till: 2/2/2021

Add

Certification Course

Course Name:

Instructor Name:

Total Hour:

Course Fee:

Valid Till:

Message



Added Success!

OK

Add

Professional Course


Course Name:			
Instructor Name:			
Total Hours:		Daily Hour:	2
Course Fee:			

Add

Course No.:	0		
Student Name:	Harry	Down Payment:	2000
Enroll Date:	2/2/2018	Room No.:	1

Enroll Student

Message X

 Student successfully enrolled!


OK

Certification Course

Course Name:			
Instructor Name:			

Course No.:	0
Student Name:	1
Enroll Date:	1

Message X

 The course has been completed

OK

Enroll Student

Certification Course


Course Name:			
Instructor Name:			
Total Hour:		Daily Hour:	
Course Fee:			
Valid Till:			

Add

Course No:	1		
Student Name:	1		
Start Date:	1	Exam Date:	1
Exam Center:	1		

Display All **Enroll**
Clear

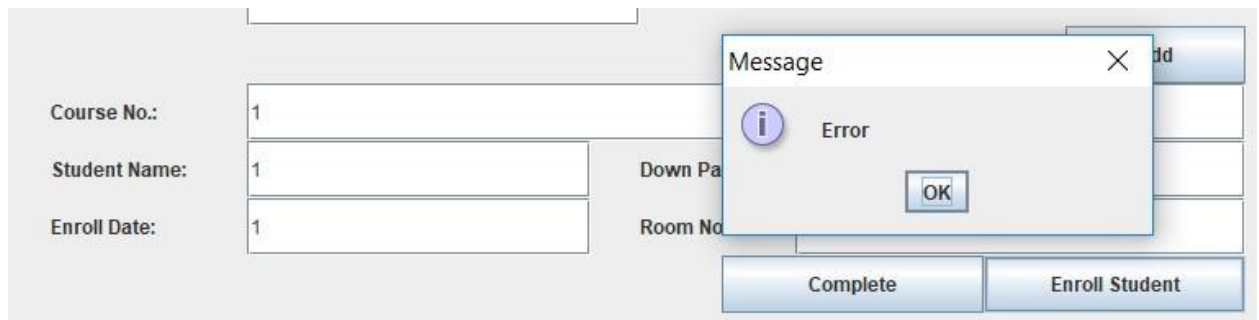
Message X

 Student successfully enrolled!

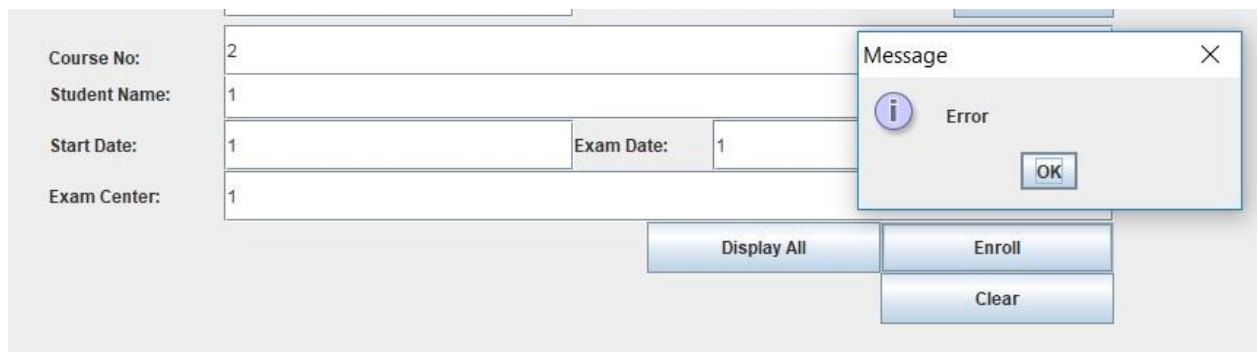
OK

5.3. Test 3

Objective	To show that appropriate dialog boxes appear when unsuitable course numbers are added.
Action	A number other than zero was entered into the enroll button of Professional course. A number other than one was entered into the enroll button of Certification course.
Expected Result	Dialog box with message “Error” will appear.
Actual Result	Dialog box with message “Error” appeared.
Conclusion	Test successful.

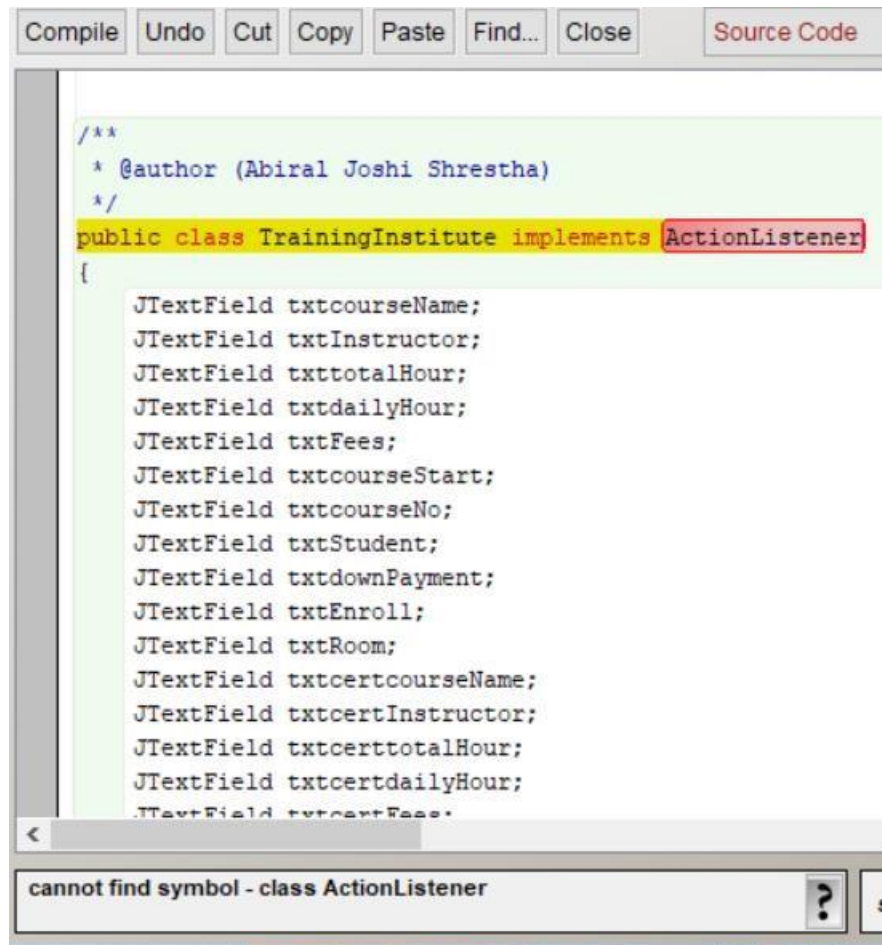


“Error” dialog box appeared when value for course number for professional course was entered 1.



“Error” dialog box appeared when value for course number for certification course was entered 2.

Error 1.



Error occurred in syntax as we accidentally forgot to import the various objects required for the project.

```
Compile Undo Cut Copy Paste Find... Close Source Code
import javax.swing.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;

/**
 * @author (Abiral Joshi Shrestha)
 */
public class TrainingInstitute implements ActionListener
{
    JTextField txtcourseName;
    JTextField txtInstructor;
    JTextField txttotalHour;
    JTextField txtdailyHour;
    JTextField txtFees;
    JTextField txtcourseStart;
    JTextField txtcourseNo;
    JTextField txtStudent;
    JTextField txtdownPayment;
}
```


Class compiled - no syntax errors saved

Using the above lines, the error was solved.

Error 2.

Course Name:			
Instructor Name:			
Total Hour:			
Course Fee:			
Valid Till:			
Course No:	0		
Student Name:	11		
Start Date:	11	Exam Date:	11
Exam Center:	11		
		Display All	Enroll
		Clear	

Message

 Course number not available

OK

Add

This error occurred as the course number value was already taken by the Professional course. The error was fixed as soon as the value 1 was entered.

The image shows a web application interface. A modal dialog box titled "Message" is centered on the screen, displaying an information icon (a lowercase 'i' in a circle) and the text "Student successfully enrolled!". Below the text is an "OK" button. In the background, a form is visible. It includes an "Add" button on the right. Below that, there is a section with the label "Exam Date:" followed by a text input field containing the number "11". At the bottom of the form, there are three buttons: "Display All", "Enroll", and "Clear".

Error 3.

Another error occurred due to which the complete button was not functioning.

Course No.:	<input type="text"/>		
Student Name:	<input type="text"/>	Down Payment:	<input type="text"/>
Enroll Date:	<input type="text"/>	Room No.:	<input type="text"/>
		<input type="button" value="Complete"/>	<input type="button" value="Enroll Student"/>

Certification Course

It was later found that the error had occurred in the syntax and it was solved by adding ActionListener to the declaration of the Complete button.

```
Complete = new JButton("Complete");  
Complete.setBounds(440,325,160,35);  
frame.add(Complete);  
frame.setSize(400,400);  
Complete.addActionListener(this);
```

```
if(e.getSource() == Complete)
{
    try
    {
        int courseNo = Integer.parseInt(txtcourseNo.getText());
        Professional prof = (Professional)List.get(courseNo);
        if(List.get(courseNo) instanceof Professional)
        {
            prof.courseCompletion();
            JOptionPane.showMessageDialog(null, "The course has been completed");

            txtStudent.setText(" ");
            txtcourseNo.setText(" ");
            txtEnroll.setText(" ");
            txtRoom.setText(" ");
            txtdownPayment.setText(" ");
        }
        else
        {
            JOptionPane.showMessageDialog(null, "Course number is not available");
        }
    }
    catch (NumberFormatException exec)
    {
        JOptionPane.showMessageDialog(null, "Enter a number");
    }
    catch (Exception exec)
    {
        JOptionPane.showMessageDialog(null, "Error");
    }
}
```

6. Conclusion

All the tasks in the coursework was finally completed through much trial and errors. The tasks assigned in the coursework were not easy at all. It required lots of labor and research. For the successful completion of all the tasks, each task was carried out in steps and pseudocodes were used. And lastly, the written program was tested to ensure that it had no bugs and errors and delivered accurate result. Finally, after completion of all the assigned tasks, submission was done.

This project didn't only complete all the tasks assigned, but also, helped in developing various skills and taught many things which can be really useful in future career as a programmer. While being involved in this project, sound knowledge of JAVA was obtained. Valuable experience has been gained working on this project. Although this project was intended for successful completion and submission of all the tasks assigned in the coursework, it doesn't mean it has limited purpose. It can be really useful to all the people who have curiosity about the JAVA programming.

7. Appendix 1

Code of Course

```
public class Course
{
    String courseName;
    String instructorName;    String studentName;    int totalHours;
    public Course(String courseName, String instructorName, int totalHours)
    {
        studentName="";
        this.courseName=courseName;
        this.instructorName=instructorName;
        this.totalHours=totalHours;
    }
    public String getcourseName()
    {
        return courseName;
    }
    public String getinstructorName()
    {
        return instructorName;
    }
    public String getstudentName()
    {
        return studentName;
    }
    public int gettotalHours()
    {
        return totalHours;
    }
    public void setstudentName(String studentName)
    {
```



```

        this.studentName=studentName;
    }
    public void display()
    {
        if (studentName.length()==0)
        {
            System.out.println(" The course name is"+" "+courseName);
            System.out.println(" The instructor's name is"+" "+instructorName);
            System.out.println(" Total hours taken by the course is"+" "+totalHours);
        }
        else
        {
            System.out.println(" The course name is"+" "+courseName);
            System.out.println(" The student's name is"+" "+studentName);
            System.out.println(" The insturctor's name is"+" "+instructorName);
            System.out.println(" Total hours taken by the course is"+" "+totalHours);
        }
    }
}

```

Code of Professional

```

public class Professional extends Course

```

```

{
    private String enrolDate;    private String roomNo;    private int
courseFee;    private int dailyHour;    private int downPayment;
private boolean start;    private boolean complete;    public
Professional( String courseName, String instructorName, int
courseFee, int totalHours, int dailyHour)
    {
        super( courseName, instructorName,
totalHours);        this.courseFee=courseFee;
this.dailyHour=dailyHour;        downPayment=0;
enrolDate="";        roomNo="";        start=false;
complete=false;
    }
    public String getenrolDate()
    {
        return enrolDate;
    }
    public String getroomNo()
    {
        return roomNo;
    }
    public int getcourseFee()
    {
        return courseFee;
    }
    public int getdailyHour()
    {
        return dailyHour;
    }
    public int getdownPayment()
    {

```

```

        return downPayment;
    }
    public boolean getstart()
    {
        return
start;
    }
    public boolean getcomplete()
    {
        return complete;
    }
    public void setcourseFee( int courseFee)
    {
        this.courseFee=courseFee;
    }
    public void setdailyHour( int dailyHour)
    {
        this.dailyHour=dailyHour;
    }
    public void enrolStudent( String studentName, String enrolDate, int
downPayment, String roomNo)
    {
        if
(start==false)    {
            super.setstudentName( studentName);
this.enrolDate=enrolDate;
this.roomNo=roomNo;
this.downPayment=downPayment;
            start=true;
complete=false;
        }
    else
    {

```

```

        super.getinstructorName();
        System.out.println("This class has already been started and taken
by"+" "+instructorName+" "+"in room number"+" "+roomNo+"."");
    }
}
public void courseCompletion()
{
    if (complete=true)
    {
        System.out.println("The course has been completed.");
    }
else
    {

super.setstudentName("");
enrolDate="";        roomNo="";
downPayment=0;
start=false;
complete=true;
    }
}

public void printing( String courseName, String instructorName, int
courseFee)
{
    super.getcourseName();
    System.out.println("The course name is " +courseName);
    System.out.println("The instructor's name is " +instructorName);
    System.out.println("The course fee is " +courseFee);
}
public void display()
{

```

```

        super.display();
    if (start=true)
    {
        System.out.println(" The course isn't complete.");
        System.out.println(" The enrol date is " +enrolDate);
        System.out.println(" The downpaymnet is " +downPayment);
    }
}
}

```

Code of Certification

```

public class Certification extends Course
{
    private int courseFee;    private String examDate;    private
String startDate;    private String examCenter;    private String
certificateAwardedBy;    private String validTill;    private
boolean start;    public Certification( String courseName, String
instructorName, int totalHour, int courseFee, String
certificateAwardedBy, String validTill)
    {

```

```

        super(courseName, instructorName, totalHour);
this.courseFee=courseFee;
this.certificateAwardedBy=certificateAwardedBy;
this.validTill=validTill;    startDate="";
examDate="";    examCenter="";    start=false;
    }
    public int getcourseFee()
    {
        return courseFee;
    }
    public String getexamDate()
    {
        return examDate;
    }
    public String getstartDate()
    {
        return startDate;
    }
    public String getexamCenter()
    {
        return examCenter;
    }
    public String getcertificateAwardedBy()
    {
        return certificateAwardedBy;
    }
    public String getvalidTill()
    {    return
validTill;
    }
    public boolean getstart()

```

```

        {      return
start;
        }
        public void setcourseFee( int courseFee)
        {      if
(start==false)
        {
                this.courseFee=courseFee;
        }
else
        {
                System.out.println("The fee is fixed.");
        }
        }
        public void enrolStudent( String studentName, String startDate,
String examDate, String examCenter)
        {      if
(start==false)
        {
                super.setstudentName( studentName);
                start=true;
        }
else
        {
                System.out.println(" The course has already started.");
        }
                this.startDate=startDate;
this.examDate=examDate;
this.examCenter=examCenter;
        }
        public void display()

```

```

    {
        super.display();
    if (start==true)
        {
            System.out.println(" The course starts from " +startDate);
            System.out.println(" The student's name is "
+studentName);
            System.out.println(" The exam starts from " +examDate);
            System.out.println(" The exam center is " +examCenter);
            System.out.println(" Certificate is awarded by "
+certificateAwardedBy);
            System.out.println(" The certificate is valid til " +validTill);
        }
    }
}

```

Code of TrainingInstitute

```

import javax.swing.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
/**
 * @author (Abiral Joshi Shrestha)
 */
public class TrainingInstitute implements ActionListener
{
    JTextField txtcourseName;
    JTextField txtInstructor;
    JTextField txttotalHour;
    JTextField txtdailyHour;
    JTextField txtFees;
    JTextField txtcourseStart;
    JTextField txtcourseNo;

```



```

    JTextField txtStudent;
    JTextField txtdownPayment;
    JTextField txtEnroll;
    JTextField txtRoom;
    JTextField txtcertcourseName;
    JTextField txtcertInstructor;
    JTextField txtcerttotalHour;
    JTextField txtcertdailyHour;
    JTextField txtcertFees;
    JTextField txtAward;
    JTextField txtValid;
    JTextField txtcertStudent;
    JTextField txtexamDate;
    JTextField txtstartDate;
    JTextField txtexamCenter;
    JTextField txtcertCourseNo;
    JButton ProfAdd;
    JButton Complete;
    JButton ProfEnrol;
    JButton CertAdd;
    JButton CertEnrol;
    JButton Clear;
    JButton Display;

```

```

    ArrayList <Course> List;
public void FormCreate()
{
    List=new ArrayList();
    JFrame frame = new JFrame("Form");
    JLabel lblProfessionalcourse = new
JLabel("Professional Course");

```

```
lblProfessionalcourse.setBounds(20,10,410,25);  
frame.add(lblProfessionalcourse);
```

```
        JLabel lblEmpty = new JLabel("_____");  
lblEmpty.setBounds(20,10,910,25);  
frame.add(lblEmpty);
```

```
        JLabel lblCertificationcourse = new JLabel("Certification  
Course");  
lblCertificationcourse.setBounds(20,360,410,25);  
frame.add(lblCertificationcourse);
```

```
        JLabel lblY = new JLabel("_____");  
lblY.setBounds(20,360,910,25);  
frame.add(lblY);
```

```
        JLabel lblcourseName = new JLabel("Course Name: ");  
lblcourseName.setBounds(30,50,110,25);  
frame.add(lblcourseName);
```

```
        txtcourseName = new JTextField();  
txtcourseName.setBounds(150,45,610,35);  
frame.add(txtcourseName);
```

```
        JLabel lblInstructor = new JLabel("Instructor Name: ");  
lblInstructor.setBounds(30,80,110,25);  
frame.add(lblInstructor);
```

```
        txtInstructor = new JTextField();  
txtInstructor.setBounds(150,80,610,35);  
frame.add(txtInstructor);
```

```
JLabel lbltotalHour = new JLabel("Total Hours: ");  
lbltotalHour.setBounds(30,120,110,25);  
frame.add(lbltotalHour);
```

```
txttotalHour = new JTextField();  
txttotalHour.setBounds(150,115,240,35);  
frame.add(txttotalHour);
```

```
JLabel lbldailyHour = new JLabel("Daily Hour: ");  
lbldailyHour.setBounds(390,120,110,25);  
frame.add(lbldailyHour);
```

```
txtdailyHour = new JTextField();  
txtdailyHour.setBounds(485,115,275,35);  
frame.add(txtdailyHour);
```

```
JLabel lblFees = new JLabel("Course Fee: ");  
lblFees.setBounds(30,155,110,25);  
frame.add(lblFees);
```

```
txtFees = new JTextField();  
txtFees.setBounds(150,150,240,35);  
frame.add(txtFees);
```

```
JLabel lblcourseNo = new JLabel("Course No.: ");  
lblcourseNo.setBounds(30,225,110,25);  
frame.add(lblcourseNo);
```

```
txtcourseNo = new JTextField();  
txtcourseNo.setBounds(150,220,610,35);  
frame.add(txtcourseNo);
```

```
JLabel lblStudent = new JLabel("Student Name: ");  
lblStudent.setBounds(30,260,110,25);  
frame.add(lblStudent);
```

```
txtStudent = new JTextField();  
txtStudent.setBounds(150,255,210,35);  
frame.add(txtStudent);
```

```
JLabel lblDownPayment = new JLabel("Down Payment: ");  
lblDownPayment.setBounds(390,260,120,25);  
frame.add(lblDownPayment);
```

```
txtDownPayment = new JTextField();  
txtDownPayment.setBounds(485,255,275,35);  
frame.add(txtDownPayment);
```

```
JLabel lblEnroll = new JLabel("Enroll Date: ");  
lblEnroll.setBounds(30,295,110,25);    frame.add(lblEnroll);
```

```
txtEnroll = new JTextField();  
txtEnroll.setBounds(150,290,210,35);  
frame.add(txtEnroll);
```

```
JLabel lblRoom = new JLabel("Room No.: ");  
lblRoom.setBounds(390,295,110,25);  
frame.add(lblRoom);
```

```
txtRoom = new JTextField();  
txtRoom.setBounds(485,290,275,35);  
frame.add(txtRoom);
```

```
JLabel lblcertcourseName = new JLabel("Course Name: ");  
lblcertcourseName.setBounds(30,395,110,25);  
frame.add(lblcertcourseName);
```

```
txtcertcourseName = new JTextField();  
txtcertcourseName.setBounds(150,390,610,35);  
frame.add(txtcertcourseName);
```

```
JLabel lblcertInstructor = new JLabel("Instructor Name: ");  
lblcertInstructor.setBounds(30,430,110,25);  
frame.add(lblcertInstructor);
```

```
txtcertInstructor = new JTextField();  
txtcertInstructor.setBounds(150,425,610,35);  
frame.add(txtcertInstructor);
```

```
JLabel lblcerttotalHour = new JLabel("Total Hour: ");  
lblcerttotalHour.setBounds(30,465,110,25);  
frame.add(lblcerttotalHour);
```

```
txtcerttotalHour = new JTextField();  
txtcerttotalHour.setBounds(150,460,240,35);  
frame.add(txtcerttotalHour);
```

```
JLabel lblcertdailyHour = new JLabel("Daily Hour: ");  
lblcertdailyHour.setBounds(390,465,110,25);  
frame.add(lblcertdailyHour);
```

```
txtcertdailyHour = new JTextField();  
txtcertdailyHour.setBounds(485,460,275,35);  
frame.add(txtcertdailyHour);
```

```
JLabel lblcertFees = new JLabel("Course Fee: ");  
lblcertFees.setBounds(30,500,110,25);  
frame.add(lblcertFees);
```

```
txtcertFees = new JTextField();  
txtcertFees.setBounds(150,495,240,35);  
frame.add(txtcertFees);
```

```
JLabel lblAward = new JLabel("Awarded By: ");  
lblAward.setBounds(390,500,110,25); frame.add(lblAward);
```

```
txtAward = new JTextField();  
txtAward.setBounds(485,495,275,35);  
frame.add(txtAward);
```

```
JLabel lblValid = new JLabel("Valid Till: ");  
lblValid.setBounds(30,535,110,25);  
frame.add(lblValid);
```

```
txtValid = new JTextField();  
txtValid.setBounds(150,530,240,35);  
frame.add(txtValid);
```

```
JLabel lblcertStudent = new JLabel("Student Name: ");  
lblcertStudent.setBounds(30,605,110,25);  
frame.add(lblcertStudent);
```

```
txtcertStudent = new JTextField();  
txtcertStudent.setBounds(150,600,610,35);  
frame.add(txtcertStudent);
```

```
JLabel lblstartDate = new JLabel("Start Date: ");  
lblstartDate.setBounds(30,640,110,25);  
frame.add(lblstartDate);
```

```
txtstartDate = new JTextField();  
txtstartDate.setBounds(150,635,240,35);  
frame.add(txtstartDate);
```

```
JLabel lblexamDate = new JLabel("Exam Date: ");  
lblexamDate.setBounds(390,640,110,25);  
frame.add(lblexamDate);
```

```
txtexamDate = new JTextField();  
txtexamDate.setBounds(485,635,275,35);  
frame.add(txtexamDate);
```

```
JLabel lblexamCenter = new JLabel("Exam Center: ");  
lblexamCenter.setBounds(30,675,110,25);  
frame.add(lblexamCenter);
```

```
txtexamCenter = new JTextField();  
txtexamCenter.setBounds(150,670,610,35);  
frame.add(txtexamCenter);
```

```
JLabel lblcertCourseNo = new JLabel("Course No: ");  
lblcertCourseNo.setBounds(30,580,110,25);  
frame.add(lblcertCourseNo);
```

```
txtcertCourseNo = new JTextField();  
txtcertCourseNo.setBounds(150,570,610,35);  
frame.add(txtcertCourseNo);
```

```
ProfAdd = new JButton("Add");  
ProfAdd.setBounds(650,185,110,35);  
frame.add(ProfAdd); frame.setSize(400,400);  
ProfAdd.addActionListener(this);
```

```
Complete = new JButton("Complete");  
Complete.setBounds(440,325,160,35); frame.add(Complete);  
frame.setSize(400,400);  
Complete.addActionListener(this);
```

```
ProfEnrol = new JButton("Enroll Student");  
ProfEnrol.setBounds(600,325,160,35);  
frame.add(ProfEnrol);  
frame.setSize(400,400);  
ProfEnrol.addActionListener(this);
```

```
CertAdd = new JButton("Add");  
CertAdd.setBounds(650,530,110,35);  
frame.add(CertAdd);  
frame.setSize(400,400);  
CertAdd.addActionListener(this);
```



```
        CertEnrol = new JButton("Enroll");
        CertEnrol.setBounds(600,705,160,35);
        frame.add(CertEnrol);
        frame.setSize(400,400);
        CertEnrol.addActionListener(this);
```

```
        Clear = new JButton("Clear");
        Clear.setBounds(600,740,160,35);
        frame.add(Clear);
        frame.setSize(400,400);
        Clear.addActionListener(this);
```

```
        Display = new JButton("Display All");
        Display.setBounds(440,705,160,35); frame.add(Display);
        frame.setSize(400,400);
        Display.addActionListener(this);
```

```
        frame.setSize(900,900);
        frame.setLayout(null);
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {
```

```
        if (e.getSource()==ProfAdd)
        {
            try
            {
                String courseName= txtcourseName.getText();
                String instructorName= txtInstructor.getText();          int
                courseFee= Integer.parseInt(txtFees.getText());          int
```

```
totalHours= Integer.parseInt(txttotalHour.getText());           int
dailyHours= Integer.parseInt(txtdailyHour.getText());
```

```
        Professional prof = new Professional(courseName,
instructorName, courseFee, totalHours, dailyHours);
```

```
        List.add(prof);
```

```
        txtcourseName.setText("");
txtInstructor.setText("");
txtAward.setText("");
txtValid.setText(""); txtFees.setText("");
txttotalHour.setText("");
```

```
        JOptionPane.showMessageDialog(null,"Added
Success!");
    }
    catch(Exception exc)
    {
        JOptionPane.showMessageDialog(null,"Error, Try
Again!");
    }
}
```

```
        if (e.getSource()==CertAdd)
        {
try
        {
```

```
String courseName= txtcertcourseName.getText();
String instructorName= txtcertInstructor.getText();      int
totalHour= Integer.parseInt(txtcerttotalHour.getText());
int courseFee= Integer.parseInt(txtcertFees.getText());
String certificateAwardedBy= txtAward.getText();
String validTill=txtValid.getText();
```

```
Certification cert = new Certification(courseName,
instructorName, totalHour, courseFee, certificateAwardedBy,
validTill);
```

```
List.add(cert);
```

```
txtcertcourseName.setText("");
txtcertInstructor.setText("");
txtcerttotalHour.setText("");
txtcertFees.setText("");
txtAward.setText("");
txtValid.setText("");
txtcertdailyHour.setText("");
```

```
JOptionPane.showMessageDialog(null,"Added Success!");
```

```
}
```

```
catch(Exception exc)
```

```
{
```

```
JOptionPane.showMessageDialog(null,"Error!!");
```

```
}
```

```
}
```

```

        if(e.getSource()==ProfEnrol)
        {
try
        {
            int courseNo = Integer.parseInt(txtcourseNo.getText());
Professional prof = (Professional) List.get(courseNo);
int downPayment =
Integer.parseInt(txtdownPayment.getText());
            if(courseNo >= 0|| courseNo < List.size())
            {
                if(List.get(courseNo) instanceof
Professional)
                {

prof.enrolStudent(txtStudent.getText(),txtEnroll.getText(),downPay
ment,txtRoom.getText());

                JOptionPane.showMessageDialog(null,"Student
successfully enrolled!");
            }
        }
else
        {
            JOptionPane.showMessageDialog(null," Course
number not available ");
        }
    }
else
    {

```

```

        JOptionPane.showMessageDialog(null," Course
number not available ");
    }

}

    catch(NumberFormatException exec)
    {
        JOptionPane.showMessageDialog(null," Enter a number
");
    }
    catch(Exception exec)
    {
        JOptionPane.showMessageDialog(null," Error ");
    }
}

```

```

        if(e.getSource()==CertEnrol)
        {
try
        {
            int courseNo1 =
Integer.parseInt(txtcertCourseNo.getText());

            int downPayment =
Integer.parseInt(txtdownPayment.getText());
            if(courseNo1 >= 0|| courseNo1 < List.size())
            {
                if(List.get(courseNo1) instanceof Certification)
                {

```

```

        Certification cert = (Certification)
List.get(courseNo1);

cert.enrolStudent(txtStudent.getText(),txtstartDate.getText(),txtEnro
ll.getText(),txtRoom.getText());

        JOptionPane.showMessageDialog(null,"Student
successfully enrolled!");
    }
else
    {
        JOptionPane.showMessageDialog(null," Course
number not available ");
    }
}
else
{
    JOptionPane.showMessageDialog(null," Course
number not available ");
}

}

catch(NumberFormatException exec)
{
    JOptionPane.showMessageDialog(null," Enter a number
");
}
catch(Exception exec)
{
    JOptionPane.showMessageDialog(null," Error ");
}
}

```

```

        if (e.getSource() == Display)
        {
            for(Course loop : List)
            {
                if (loop instanceof Professional)
                {
                    Professional prof = (Professional) loop;
prof.display();
                }
            }
            for(Course loop : List)
            {
                if (loop instanceof Certification)
                {
                    Certification cert = (Certification) loop;
cert.display();
                }
            }
        }
    }

```

```

        if(e.getSource() == Complete)
        {
try
        {
            int courseNo = Integer.parseInt(txtcourseNo.getText());
Professional prof = (Professional)List.get(courseNo);
if(List.get(courseNo) instanceof Professional)
            {
                prof.courseCompletion();
            }
        }
    }

```

```
        JOptionPane.showMessageDialog(null,"The course  
has been completed");
```

```
        txtStudent.setText(" ");  
txtcourseNo.setText(" ");  
txtEnroll.setText(" ");  
txtRoom.setText(" ");  
txtdownPayment.setText(" ");
```

```
    }
```

```
    else
```

```
        {  
            JOptionPane.showMessageDialog(null, "Course  
number is not available");
```

```
        }
```

```
    }
```

```
    catch(NumberFormatException exec)
```

```
    {
```

```
        JOptionPane.showMessageDialog(null,"Enter a  
number");
```

```
    }
```

```
    catch(Exception exec)
```

```
    {
```

```
        JOptionPane.showMessageDialog(null,"Error");
```

```
    }
```

```
    }
```

```
    if( e.getSource() == Clear)
```

```
    {
```

```
        txtcourseName.setText("");
```

```
txtInstructor.setText("");
```



```

txttotalHour.setText("");
txtdailyHour.setText("");
txtFees.setText("");
//txtcourseStart.setText("");
txtcourseNo.setText("");
txtStudent.setText("");
txtdownPayment.setText("");
txtEnroll.setText("");
txtRoom.setText("");
txtcertcourseName.setText("");
txtcertInstructor.setText("");
txtcerttotalHour.setText("");
txtcertdailyHour.setText("");
txtcertFees.setText("");
txtAward.setText("");      txtValid.setText("");
txtcertStudent.setText("");
txtexamDate.setText("");
txtstartDate.setText("");
txtexamCenter.setText("");
txtcertCourseNo.setText("");
    }
}

public static void main(String args[])
{
    new TrainingInstitute().FormCreate();
}
}

```

8. Appendix 2

Method description of Course `getcourseName()`

This method is used to get the Student Name. **`getinstructorName()`**

This method is used to get the Instructor Name.

`getstudentName()`

This method is used to get student name.

`gettotalHours()`

This method is used to get the total hours. **`setstudentName()`**

This method is used to set student name.

`Display()`

This method is used to display the course name, instructor's name, total hours and student's name.

Method description of Professional

getenrolDate()

This method is used to get the enroll date.

getroomNo()

This method is used to get the room number. **getcourseFee()**

This method is used to get the course fee.

getdailyHour()

This method is used to get the daily hours. **getdownPayment()**

This method is used to get the down payment.

getstart()

This method is used to confirm if the course is started or not. **getComplete()**

This method is used to confirm the end of the course. **setcourseFee()**

This method is used to set the new value to the courseFee. **setdailyHours()**

This method is used to set the daily hours of the classes. **enrolStudent()**

This method is used to enroll new students into the class.

Method description of Certification

getcourseFee()

This method is used to get the course fee.

getexamDate()

This method is used to get the exam date.

getstartDate()

This method is used to get the start of the exam.

getexamCenter()

This method is used to get the center of the exam.

getcertificateAwardedBy()

This method is used to get who the certificate is awarded by.

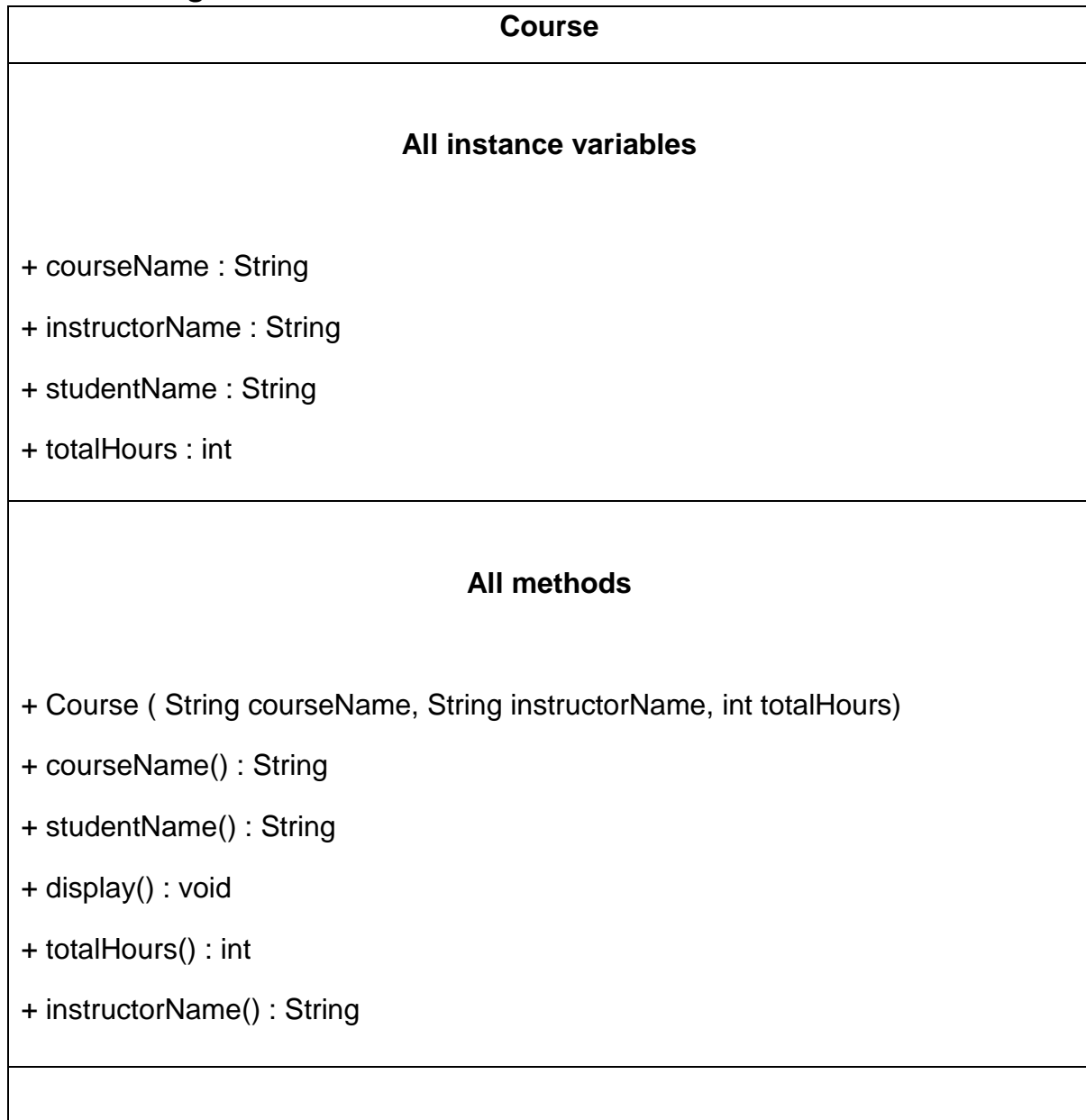
getvalidTill()

This method is used to get the validity period of the certificate.

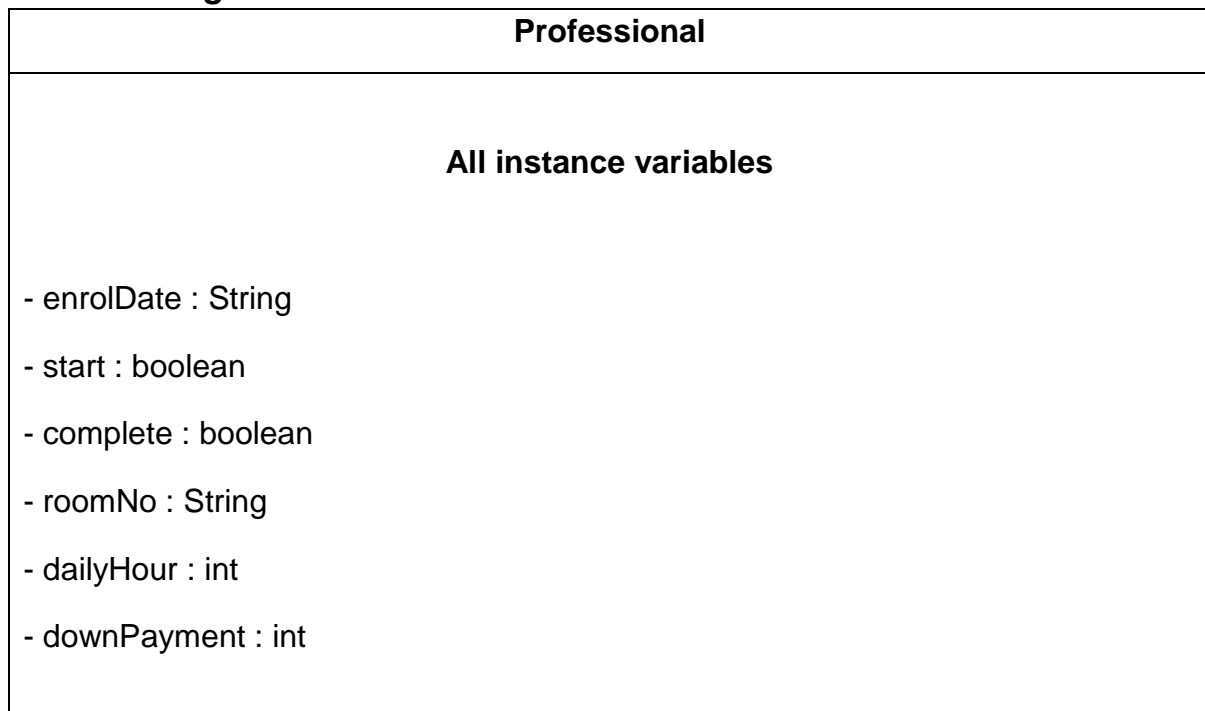
getStart()

This method is used to get the value of start.

Class Diagram of Course



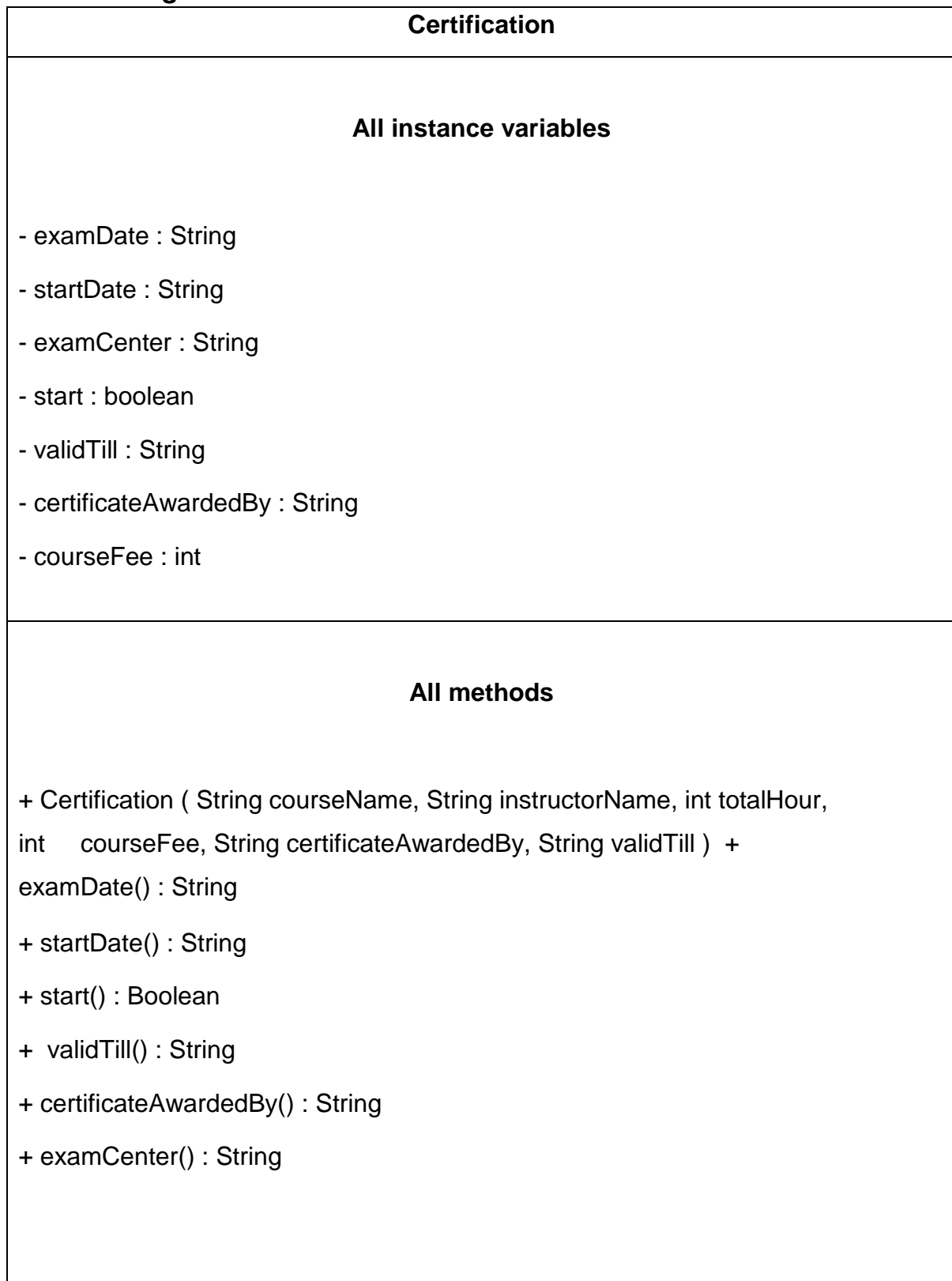
Class diagram of Professional



All methods

+ Professional (String courseName, String instructorName, int courseFee, int totalHours)
+ enrolStudent (String studentName, String enrolDate, int downPayment, String roomNo) : void + enrolDate() : String + roomNo() : String
+ courseFee() : int
+ dailyHour() : int
+downPayment() : int
+ start() : boolean
+ complete() : boolean
+ courseFee(int courseFee) : void
+ dailyHour(int dailyHour) : void

Class diagram of Certification



Pseudocode of Course

Call Course(String courseName, String instructorName, int totalHours)

DO

studentName = “ ”;

this.courseName = courseName;

this.instructorName = instructorName;

this.totalHours = totalHours

END DO

Call String getcourseName()

DO

Return courseName;

END DO

Call String getinstructorName()

DO

Return instructorName;

END DO

Call String studentName()

DO

Return studentName;

END DO

Call String gettotalHours()

DO

Return totalHours;

END DO

Call String setstudentName(Stringstudent)

DO

 this.studentName = studentName;

END DO

Call void display()

DO

 IF (studentName.lenght()==0)

 DO

 DISPLAY("The name of course is " +courseName)

 DISPLAY("The instructor's name is "
+instructorName)

 DISPLAY("The total hours taken is " +totalHours) END

 DO

 ELSE

 DO

 DISPLAY("The name of course is " +courseName)

 DISPLAY("The name of the instructor is "
+instructorName)

 DISPLAY("Total hours taken is " +totalHours)

 DISPLAY("Student's name is " +studentName)

 END DO

END DO

Pseudocode of Professional

Call Professional(String courseName, String instructorName, int
courseFee, int totalHours, int dailyHours)

DO

Super(courseName, instructorName, totalHours);
this.courseFee = courseFee; this.dailyHour =
dailyHour;

downPay
ment = 0; enrolDate
= "";
roomNo =
""; start = false;
complete
= false;

END DO

Call String getenrolDate()

DO

Return enrolDate;

END DO

Call String getroomNo()

DO

Return roomNo;

END DO

Call String getcourseFee()

DO

Return courseFee;

END DO

Call int getdailyHour()

DO

Return dailyHour;

END DO

Call int downPayment()

DO

Return downPayment;

END DO

Call boolean getstart()

DO

Return start;

END DO

Call boolean getcomplete()

DO

Return complete;

END DO

Call void setcourseFee(int courseFee)

DO

this.courseFee = courseFee;

END DO

Call void setdailyHour(int dailyHour)

DO

this.dailyHour = dailyHour;

END DO

Call void enrolStudent(String studentName, String enrolDate, int
downPayment, String roomNo)

DO

If(complete = true)

DO

DISPLAY("Course has been completed.");

END DO

ELSE

DO

Super.studentName(" ");

enrolDate = "";

roomNo = "";

downPayment = 0;

start = false;

complete = true;

END DO

END DO

Call void print (String courseName, String instructorName, int courseFee)

DO

Super.getcourseName();

DISPLAY("Name of course is " +courseName)

DISPLAY("Enrol date is " +enrolDate)

DISPLAY(" Down payment is " +downPayment)

END DO

Call void display()

DO

If(start = true)

DO

DISPLAY("Course is pending.")

DISPLAY("Enrol date is " +enrolDate)

DISPLAY("Down payment is "

+downPayment)

END DO

Pseudocode of Certification

Call Certification (String courseName, String instructorName, int totalHour, int courseFee, String certificateAwardedBy, String validTill)

DO

Super (courseName, instructorName,
totalHours); this.courseFee = courseFee;
this.certificateAwardedBy = certificateAwardedBy;
this.validTill = validTill; startDate = ""; examDate
= ""; examCenter =
""; start =
false;

END DO

Call int getcourseFee()

DO

Return courseFee;

END DO

Call String getexamDate()

DO

Return examDate;

END DO

Call String startDate()

DO

Return startDate;

END DO

Call String getexamCenter()

DO

Return examCenter;

END DO

Call String gecertificateAwardedBy()

DO

Return certificateAwardedBy;

END DO

Call String getvalidTill()

DO

Return validTill;

END DO

Call boolean getstart()

DO

Return start;

END DO

Call void setcourseFee(int courseFee)

DO

If (start==falce)

DO

this.courseFee = courseFee;

ELSE DO

DISPLAY("The fee is fixed.")

END DO

Call void enrolStudent(String studentName, String startDate, String examDate,
String examCenter)

DO

IF(start==false)

DO

Super.setstudentName(studentName);

start=true;

END DO

ELSE

DO

DISPLAY("The course has already started.")

END DO

this.startDate = startDate;

this.examDate = examDate;

this.examCenter = examCenter;

END DO

Call void display()

DO

Super.display()

If(start==true)

DO

System.out.println("The course starts from "
+startDate);

System.out.println("The student's name is "
+studentName)

```
System.out.println("      Exam date is      "  
+examDate);
```

```
System.out.println(" Exam center is " +examCenter);
```

```
System.out.println("Certificate awarded by "  
+certificateAwardedBy);
```

```
System.out.println(" Validity till " +validTill);
```

```
END DO
```

```
END DO
```