Islington College

islington college

Programming

CS4001NI

Coursework 2

Submitted By:

Rajat Shrestha

17030954

Group: L1C2

Date: 22nd April 2018

Submitted To:

Mr. Dhurba Sen

Module Leader

Programming

Contents:

Table of Figures:

Tables:

1. Introduction:

The main objective of this project is to create a code which adds GUI aspects to the previous coursework classes for easier entry of data. The entered data should have exception handling features and the courses should be stored in an array list constructed. This program must also be modular so that a same piece of code can be reused. As java is an object oriented programming language the codes written in it can also be run anywhere. Inheritance is a feature of java which allows one class to incorporate another class into its declaration which is done by using “extends” keyword (Kolling, 2009).

Objective and Approach:

To complete the given task planning will greatly help the progression of the project so to complete the project the following tasks were done:

1. To gathering information about the project and keywords used in it.
2. To constructing some simple models to aid the development of the project (like class diagram and pseudocode).
3. To code the program.
4. To test the program to find errors and fix them.
5. Documenting the process

Scope:

This project will be useful for future reference on developing any java program based on the object-oriented format for any scholars, programmers or associations as it utilizes the object-oriented programming method in java. The object-oriented method focuses more on the objects than actions to be executed (Rouse, 2014).

2. Class Diagram:

UML Class diagram is a simple way of representing the hierarchy of java classes in pictorial form. Class diagram is a simply the interpretation of the model of code in a pictorial and easier form (Knoernschild, 2002). The class diagram will simplify the idea of the code. The class diagram for TrainingInstitute class is as follows:

Table 1: Class Diagram

TrainingInstitute
- frame: JFrame - sepVl: JSeparator - lblDescription: JLabel - lblInstructor: JLabel - lblCourseDuration: JLabel - lblFee: JLabel - lblDailyHour: JLabel - lblDownPayment: JLabel - lblStudentName: JLabel - lblEnrollDate: JLabel - lblRoomNo: JLabel - lblDescription2: JLabel - lblInstructor2: JLabel - lblCourseDuration2: JLabel - lblFee2: JLabel - lblStudentName2: JLabel - lblStartDate: JLabel - lblExamDate: JLabel - lblCourseNo: JLabel - lblCourseNo2: JLabel - lblExamCenter: JLabel - lblAwardedBy: JLabel - lblValidDuration: JLabel - lblCertificationCourse: JLabel - lblProfessionalCourse: JLabel - lblBorder: JLabel - lblBorder2: JLabel - txtDescription: JTextField - txtInstructor: JTextField - txtCourseDuration: JTextField - txtFee: JTextField - txtDailyHour: JTextField

- txtDownPayment: JTextField
- txtStudentName: JTextField
- txtEnrollDate: JTextField
- txtRoomNo: JTextField
- txtStartDate: JTextField
- txtExamDate: JTextField
- txtDescription2: JTextField
- txtInstructor2: JTextField
- txtCourseDuration2: JTextField
- txtFee2: JTextField
- txtStudentName2: JTextField
- txtExamCenter: JTextField
- txtAwardedBy: JTextField
- txtValidDuration: JTextField
- txtCourseNo: JTextField
- txtCourseNo2: JTextField
- btnAdd: JButton
- btnAdd2: JButton
- btnComplete: JButton
- btnEnrollStudent: JButton
- btnEnrollStudent2: JButton
- btnDisplayAll: JButton
- btnClear: JButton

+ actionPerformed(ActionEvent ae): void
+ guiComponents (): void
+ addProfessional(): void
+ addCertification(): void
+ checkCompletion(): void
+ clearAll(): void
+ dispalyMethod(): void
+ enrollProfessional(): void
+ enrollCertification(): void

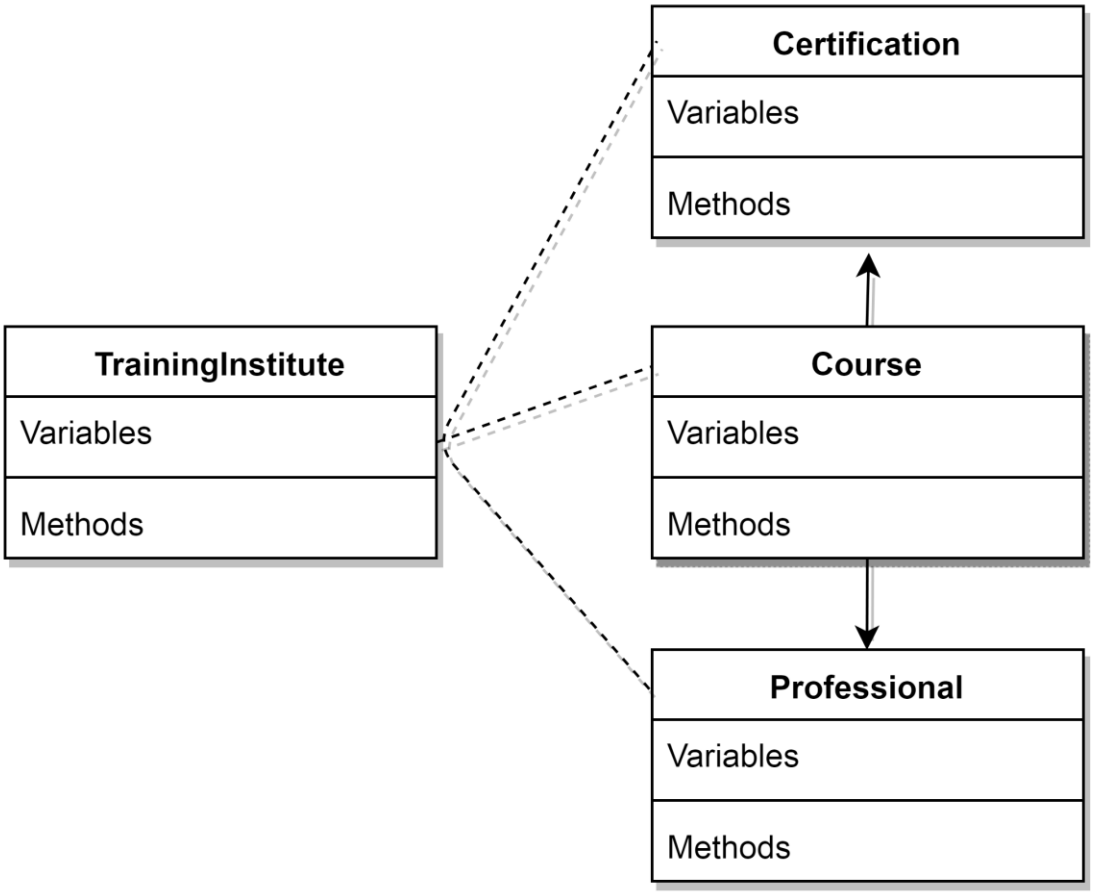

Figure 1: Overall Class diagram

3. Pseudocode:

Pseudocode represents an idea in a form of code but is not a real programming code. It is a simple way of describing an algorithm in a false code manner so that it becomes easier for the developer to convert it into a real code (Farrell, 2009). The pseudocodes for each method in the class is as follows:

TrainingInstitute class

```
CALL guiComponents():
    DO
        CREATE frame, lbl*, txt*, btn*, sepV;
        ADD frame;
        ADD LABEL lbl* TO frame;
        ADD TEXTFIELD txt* TO frame;
        ADD BUTTON btn* TO frame;
        ADD SEPARATOR sepV TO frame;

        SET BOUNDS
            DO
                lblProfessionalCourse(10, 5, 200 ,20);
                lblBorder(0, 29, 500 ,14);
                lblDescription(10, 80, 100 ,14);
                lblInstructor(10, 105, 100 ,14);
                lblCourseDuration(10, 130, 100 ,14);
                lblFee(10, 155, 100 ,14);
                lblDailyHour(10, 180, 100 ,14);
                lblCourseNo(10, 305, 100 ,14);
                lblStudentName(10, 330, 100 ,14);
```

```
lblEnrollDate(10, 355, 100 ,14);
lblRoomNo(10, 380, 100 ,14);
lblDownPayment(10, 405, 100 ,14);
lblCertificationCourse(550, 5, 200 ,20);
lblBorder2(550, 29, 498 ,14);
lblDescription2(550, 80, 100 ,14);
lblInstructor2(550, 105, 100 ,14);
lblCourseDuration2(550, 130, 100 ,14);
lblFee2(550, 155, 100 ,14);
lblAwardedBy(550, 180, 100 ,14);
lblValidDuration(550, 205, 100 ,14);
lblCourseNo2(550, 305, 100 ,14);
lblStudentName2(550, 330, 100 ,14);
lblStartDate(550, 355, 100 ,14);
lblExamDate(550, 380, 100 ,14);
lblExamCenter(550, 405, 600 ,18);
txtDescription(110, 80, 330 ,22);
txtInstructor(110, 105, 330 ,22);
txtCourseDuration(110, 130, 130 ,22);
txtFee(110, 155, 130 ,22);
txtDailyHour(110, 180, 130 ,22);
txtCourseNo(110, 305, 100 ,22);
txtStudentName(110, 330, 330 ,22);
txtEnrollDate(110, 355, 330 ,22);
txtRoomNo(110, 380, 130 ,22);
txtDownPayment(110, 405, 130 ,22);
txtDescription2(650, 80, 330 ,22);
txtInstructor2(650, 105, 330 ,22);
txtCourseDuration2(650, 130, 130 ,22);
txtFee2(650, 155, 130 ,22);
txtAwardedBy(650, 180, 330 ,22);
```

```
                    txtValidDuration(650, 205, 330 ,22);
                    txtCourseNo2(650, 305, 100 ,22);
                    txtStudentName2(650, 330, 330 ,22);
                    txtStartDate(650, 355, 230 ,22);
                    txtExamDate(650, 380, 230 ,22);
                    txtExamCenter(650, 405, 230 ,22);
                    btnAdd(150, 220, 210 ,25);
                    btnComplete(250, 300, 200 ,25);
                    btnEnrollStudent(250, 435, 200 ,27);
                    btnAdd2(690, 230, 210 ,25);
                    btnEnrollStudent2(790, 435, 200 ,27);
                    btnDisplayAll(520, 500, 200 ,27);
                    btnClear(290, 500, 180 ,27);
                    sepVl(495, 0, 10 ,700);
                END DO
            SET BUTTON ACTION
                DO
                    btnAdd DO (THIS);
                    btnComplete DO (THIS);
                    btnEnrollStudent DO (THIS);
                    btnAdd2 DO (THIS);
                    btnEnrollStudent2 DO (THIS);
                    btnDisplayAll DO (THIS);
                    btnClear DO (THIS);
                END DO
        END DO
```

```
CALL actionPerformed(ActionEvent e):
    DO
        IF (ACTION = btnAdd):
            DO
                METHOD addProfessional ():
            END DO
        END IF

        ELSE IF (ACTION = btnComplete):
            DO
                METHOD checkCompletion ():
            END DO
        END ELSE IF

        ELSE IF (ACTION = btnEnrollStudent):
            DO
                METHOD EnrollProfessional():
            END DO
        END ELSE IF

        ELSE IF (ACTION = btnAdd2):
            DO
                METHOD addCertification ():
            END DO
        END ELSE IF

        ELSE IF (ACTION = btnEnrollStudent2):
            DO
                METHOD EnrollCertification ():
            END DO
        END ELSE IF
```

```
        ELSE IF (ACTION = btnDisplayAll):
            DO
                METHOD dispaly ():
            END DO
        END ELSE IF

        ELSE IF (ACTION = btnClear):
            DO
                METHOD clearAll ():
            END DO
        END ELSE IF
    END DO


METHOD dispaly ():
    DO
        FOR (i: list):
            IF (list.GET(courseNo) IS INSTANCEOF Professional):
                DO
                    CALL Professional.display METHOD
                    list.GET(courseNo).display;
                END DO
            END IF
            IF (list.GET(courseNo) IS INSTANCEOF Certification):
                DO
                    CALL Certification.display METHOD
                    list.GET(courseNo).display;
                END DO
            END IF
    END DO
```

```
METHOD addProfessional ():
    DO
        TRY:
            DO
                GET INPUT FROM txtfields;
                STORE INPUTS TO variables;
                IF (variables ARE VALID):
                    DO
                        CREATE Professional OBJECT p;
                        CALL Professional CONSTRUCTOR p(variables);
                        list.ADD(p);
                        SHOW SUCESS dialog;
                    END DO
                ELSE :
                    DO
                        SHOW ERROR dialog;
                    END DO
            END DO
        CATCH (EXCEPTION e):
            DO
                SHOW ERROR dialog;
            END DO
    END DO
```

```
METHOD addCertification ():
    DO
        TRY:
            DO
                GET INPUT FROM txtfields;
                STORE INPUTS TO variables;
                IF (variables ARE VALID):
                    DO
                        CREATE Certification OBJECT c;
                        CALL Certification CONSTRUCTOR c(variables);
                        list.ADD(c);
                        SHOW SUCESS dialog;
                    END DO
                ELSE :
                    DO
                        SHOW ERROR dialog;
                    END DO
            END DO
        CATCH (EXCEPTION e):
            DO
                SHOW ERROR dialog;
            END DO
    END DO
```

```
METHOD checkCompletion ():
    DO
        TRY:
            DO
                GET courseNo FROM txtfields;
                IF (courseNo IS VALID);
                    IF (list.GET(courseNo) IS INSTANCEOF Professional);
                        DO
                            CALL Professional.setCompleted METHOD
                             list.GET(courseNo).setCompleted;
                            SHOW SUCESS dialog;
                        END DO
                    END IF
                END IF
                ELSE :
                    DO
                        SHOW ERROR dialog;
                    END DO
            END DO
        CATCH (EXCEPTION e):
            DO
                SHOW ERROR dialog;
            END DO
    END DO
```

```
METHOD EnrollProfessional():
    DO
        TRY:
            DO
                GET INPUT FROM txtfields;
                STORE INPUTS TO variables;
                IF (variables ARE VALID):
                    IF (list.GET(courseNo) IS INSTANCEOF Professional):
                        IF (variables !EMPTY):
                            DO
                                CALL Professional.setEnroll
                                METHOD list.GET(courseNo).setEnroll;
                                SHOW SUCESS dialog;
                            END DO
                        END IF
                    END IF
                END IF
                ELSE :
                    DO
                        SHOW ERROR dialog;
                    END DO
            END DO
        CATCH (EXCEPTION e):
            DO
                SHOW ERROR dialog;
            END DO
    END DO
```

```
METHOD EnrollCertification ():
    DO
        TRY:
            DO
                GET INPUT FROM txtfields;
                STORE INPUTS TO variables;
                IF (variables ARE VALID):
                    IF (list.GET(courseNo) IS INSTANCEOF Certification):
                        IF (variables !EMPTY):
                            DO
                                CALL Certification.setEnroll METHOD
                                 list.GET(courseNo).setEnroll;
                                SHOW SUCESS dialog;
                            END DO
                        END IF
                    END IF
                END IF
                ELSE :
                    DO
                        SHOW ERROR dialog;
                    END DO
            END DO
        CATCH (EXCEPTION e):
            DO
                SHOW ERROR dialog;
            END DO
    END DO
```

```
METHOD clearAll ():
    DO
        SET txt* = " ";
        SHOW SUCESS dialog;
    END DO
```

4. Method Description

A Java method contains statements that are put together to perform an operation in a java code (Jenkov, 2015). It helps to make the code modular and reuse the code. Making the code modular also makes it easier to find hidden bugs and solve them so this kind of modular programming is highly recommended. There are 9 methods in total for this class. The method's description for each method are as follows:

guiComponents

This method sets up all the necessary GUI elements such as frame, label, buttons and text fields. Its whole purpose is to provide a decent GUI framework which handles the aesthetics and the functionality of several GUI elements.

addProfessional

This method takes valid input from the text fields for the professional course by taking course name, instructor name, course duration, fee and daily hour from it. This method then checks the validity of the input data and gives out error messages if any error occurs so that valid data is passed to the constructer called while making a professional object which is then added to an array list of courses.

addCertification

Similarly, this method also takes valid input from the text fields for the certification course by taking course name, instructor name, course duration, fee, certification awarded by and valid till from it. This method then checks the validity of the input data and gives out error messages if any error occurs so that valid data is passed to the constructer called while making a certification object which is then added to an array list of courses.

clearAll

The purpose of this method is to clear all the text fields on the GUI.

checkCompletion

The purpose of this method is to take a valid value from the course number text field, check whether the corresponding object linked to course number entered is an instance of professional class or not. If the number indexed is of professional class, it will call the “setCompletion” method of the professional class. If the input number is invalid it will display an error.

dispalyMethod

This method iterates through all the elements stored in the array list and calls the display methods from professional and certification code to display all the entered details.

enrollProfessional

This method enrolls student to the professional class by calling the enroll method on the professional class if the course is available and if all the inputs are valid. Otherwise it will show an error prompt. It takes course number, student name, enroll date, room no and down payment from the corresponding text fields.

enrollCertification

This method enrolls the student to certification class by calling the enroll method on the certification class by taking valid inputs. Course number, student name, exam date, start date and exam center must be entered to corresponding text fields. If the input is invalid or the course number is of professional course it displays an error in a prompt message.

actionPerformed

This method overrides the method in the “ActionListener” class. The main purpose of this method is to assign corresponding methods to the buttons, so that when the button is pressed the corresponding method is invoked.

Table 2: Method Summary

Method Summary	
void	**actionPerformed(java.awt.event.ActionEvent e)** Override method for ”ActionListener class” is called whenever buttons are clicked and events are invoked according to the buttons that are clicked.
static void	**main(java.lang.String[ ] args)** The main method to start program.
void	**addProfessional()** Takes input of data from the user about the course name, instructor, course duration, fee, and daily hour. It creates an object and appends it to a list
void	**addCertification()** Takes input of data from the user about the course name, instructor, course duration, fee, awarded by, and valid duration. It creates an object and appends it to a list
void	**checkCompletion()** takes course no as input and calls the setComplete method from professional class to set the course as completed.
void	**EnrollProfessional()** Takes course no, student name, enroll date, room no, and down payment and enrolls the student to the respective course.
void	**EnrollCertification()** Takes course no, student name, start date, exam date, and exam center and enrolls the student to the respective course.
void	**display()** Displays all the entered details.
void	**clearAll()** clears all the text fields

Table 3:Constructor Summary

Constructor Summary
TrainingInstitute () Constructor of the class. It is empty as the object doesn't require any variable to initialize

Important Details on class TrainingInstitute:

- Classified as: java.lang.Object
- public class so it can be accessed by other classes.
- Adds GUI interface
- Implements action listener:
    - java.awt.event.ActionListener, java.util.EventListener
- Imports pakages from:
    - java.util
        - ArrayList
    - java.awt
        - Font
        - ActionEvent (event)
        - ActionListener (event)
    - javax.swing
        - JTextField
        - JLabel
        - JFrame
        - JSeparator
        - JOptionPane

5. Tests:

Testing ensures the functionality, correctness and stability of the code. To make sure that the code works as intended, several tests must be conducted.

5.1. Test 1:

Table 4: Test 1

Objective	Checking whether the program can be compiled and run using command prompt.
Action	Opening command prompt and giving the location of java file. Then compiling and running TrainingInstitute
Expected Result	File gets compiled and GUI opens.
Actual Result	File compiled and GUI is opened after giving the path where the file is located.
Conclusion	Test Successful.

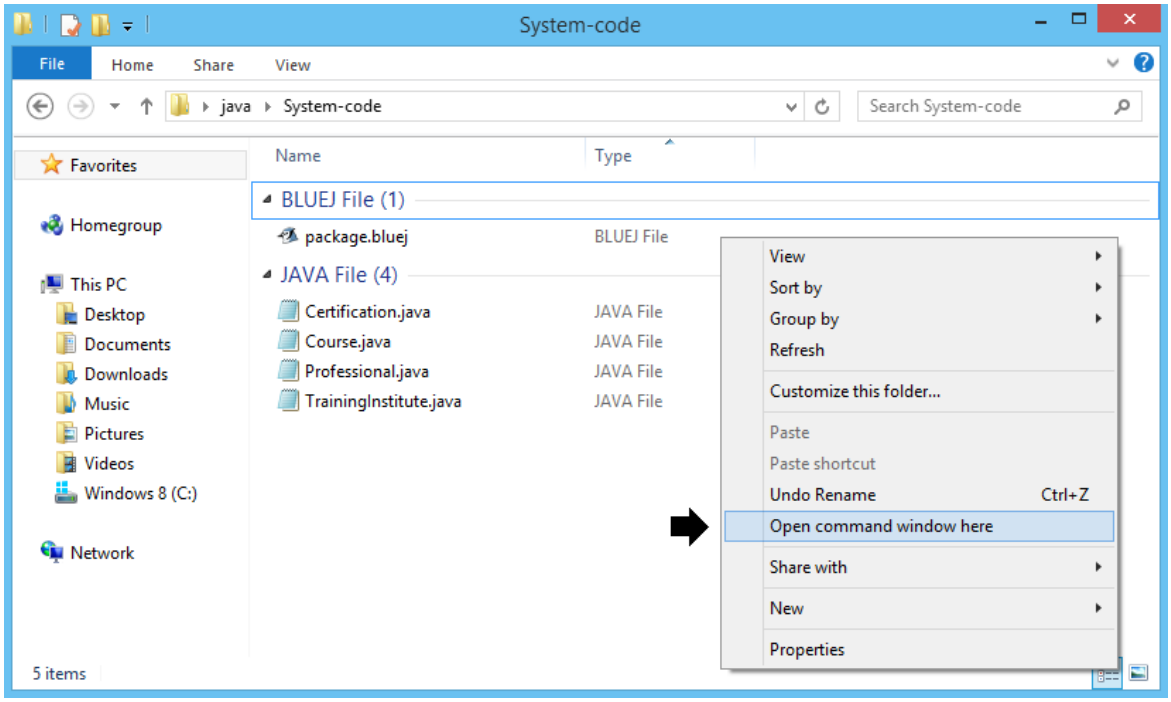

Figure 2: Opening the command prompt in the code directory

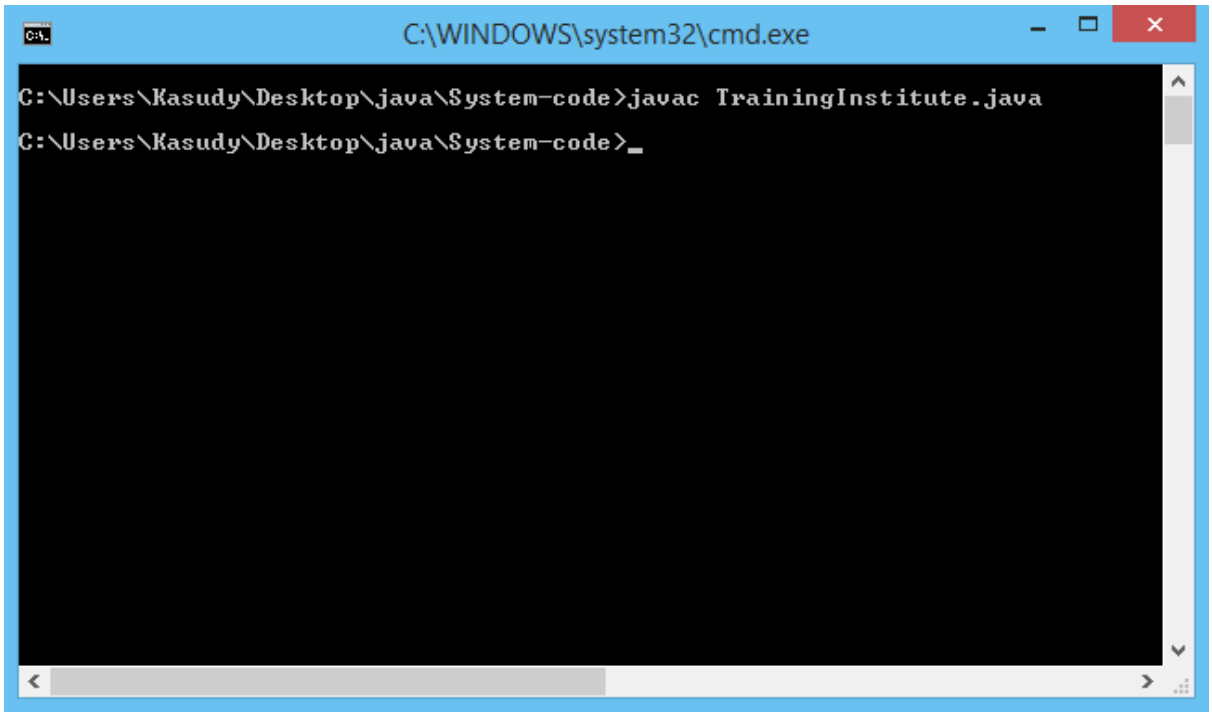

Figure 3: Compiling TrainingInstitute.java file

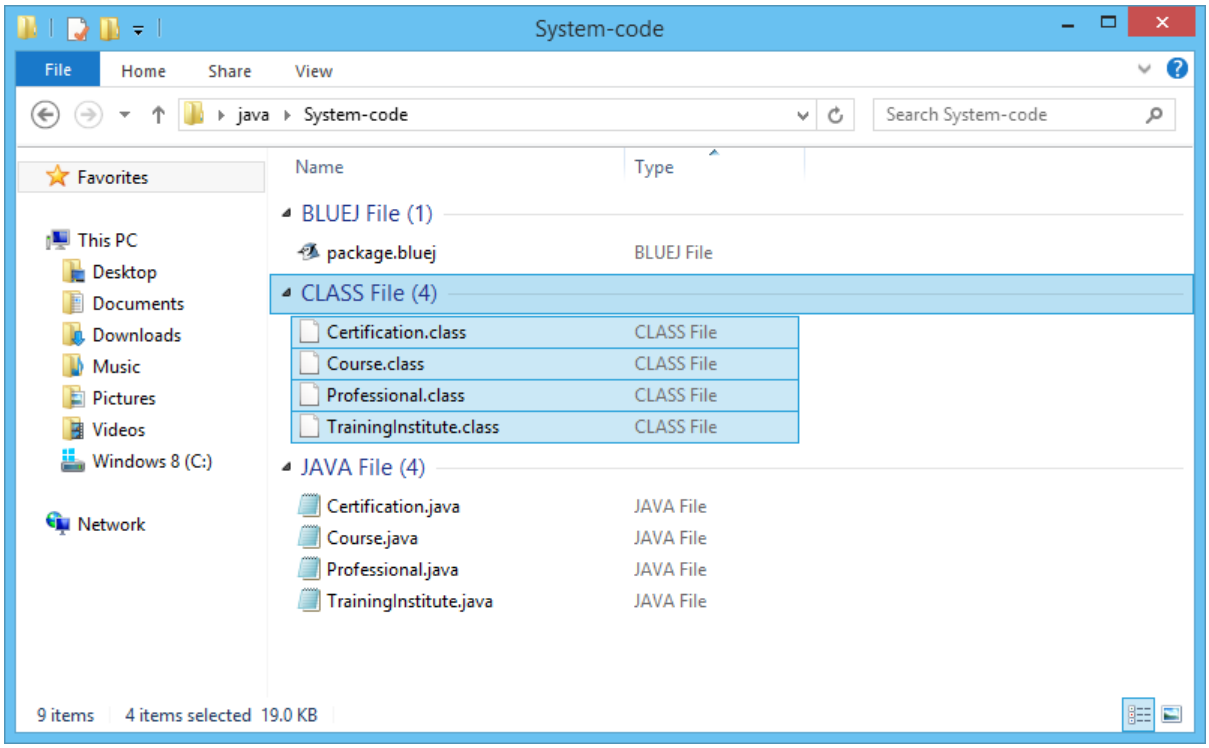

Figure 4: class files created after compiling .java files

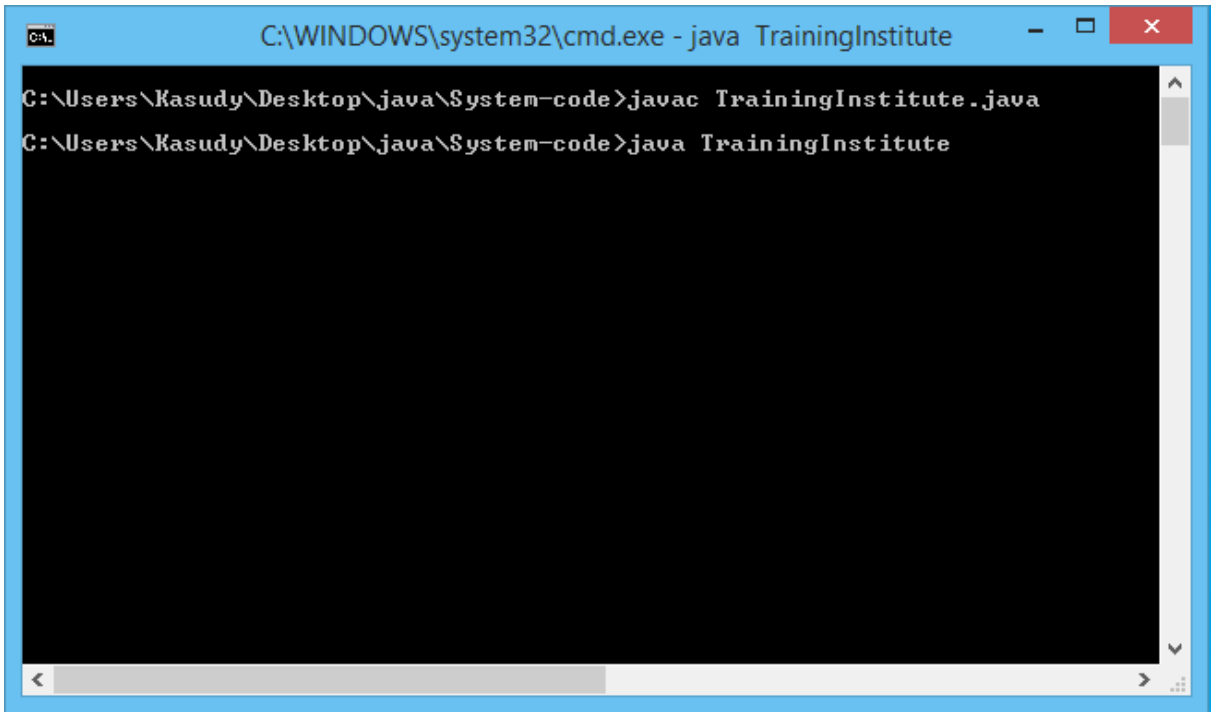

Figure 5: running the code

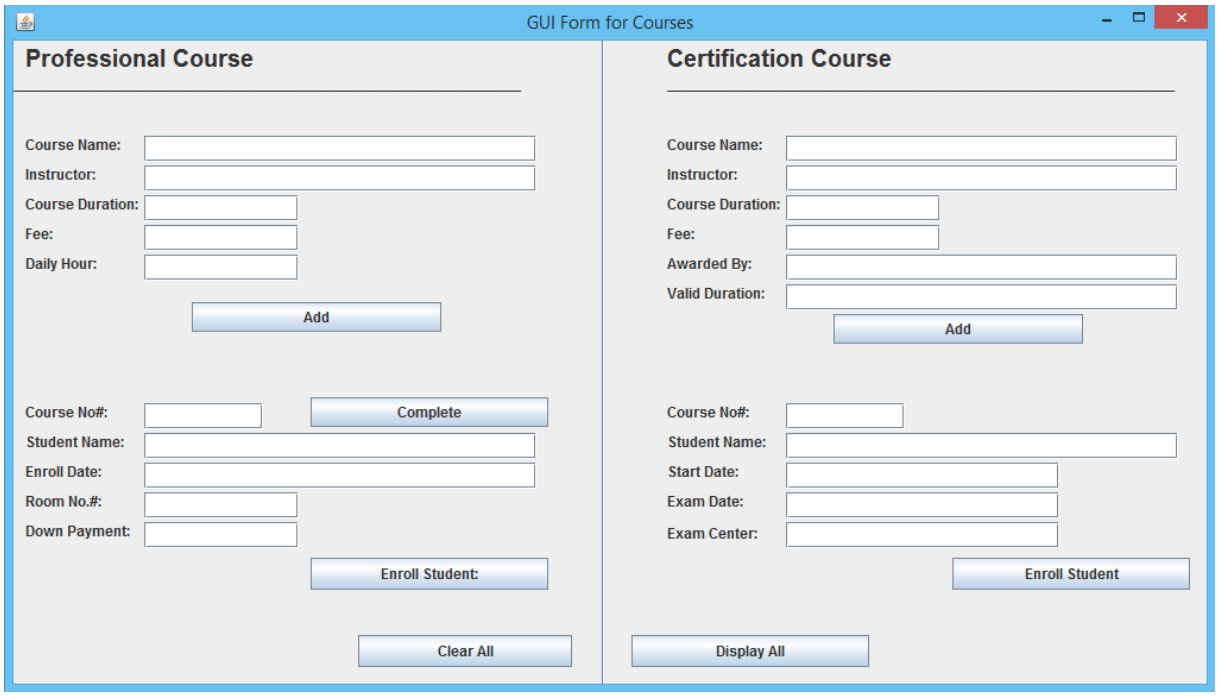

Figure 6: GUI opens after running the code

5.2. Test 2:

Table 5: Test 2

Objective	Adding Professional course and certification course. Enrolling each student to both course. Completing the professional course.
Action	Inputting data to add and enroll. Complete button to set the professional course as completed.
Expected Result	The courses must be added, students must be enrolled and the professional course must be set completed.
Actual Result	The courses were added and the student enrolled. The completion button set the course to completed.
Conclusion	Test Successful

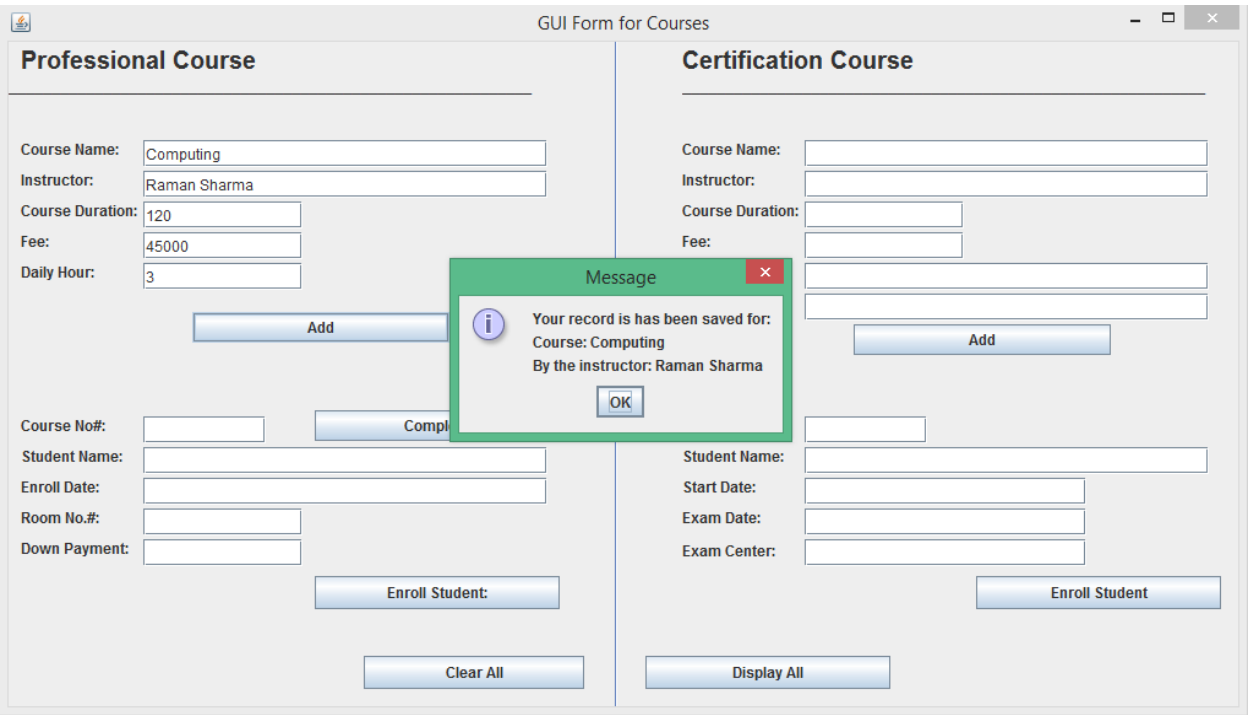

Figure 7: Adding a course to professional

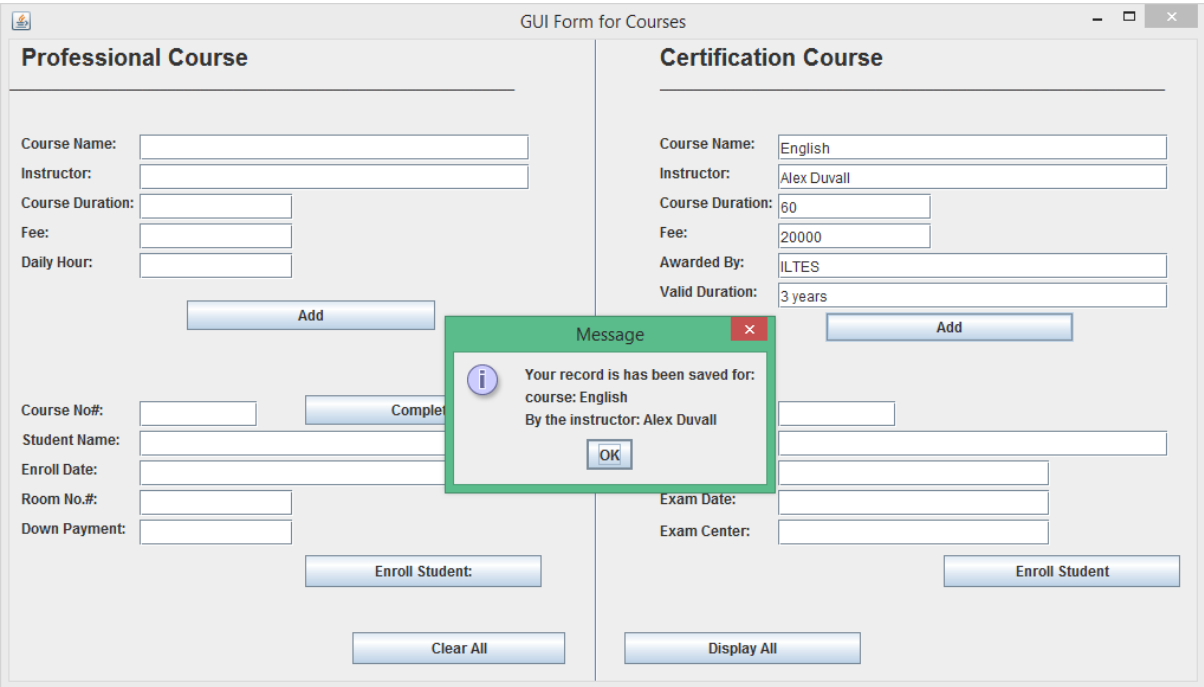

Figure 8: Adding a course to Certification

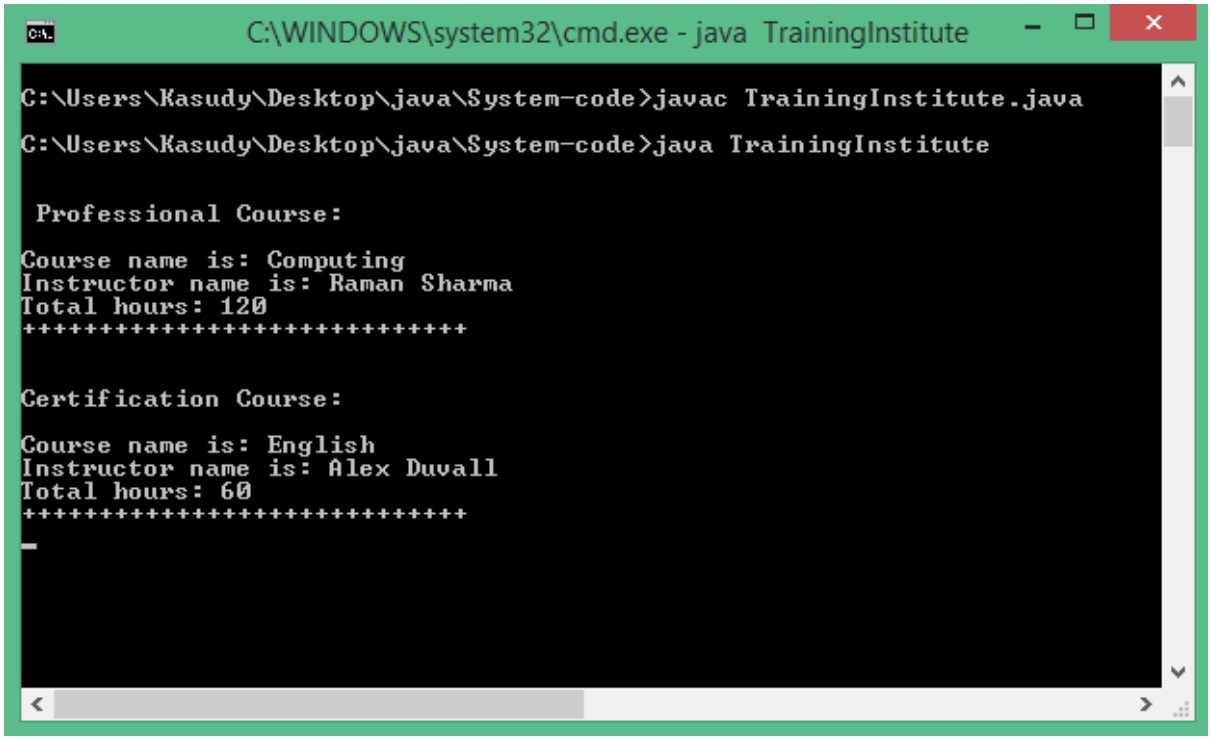

Figure 9: The previously added details displayed by pressing the display button

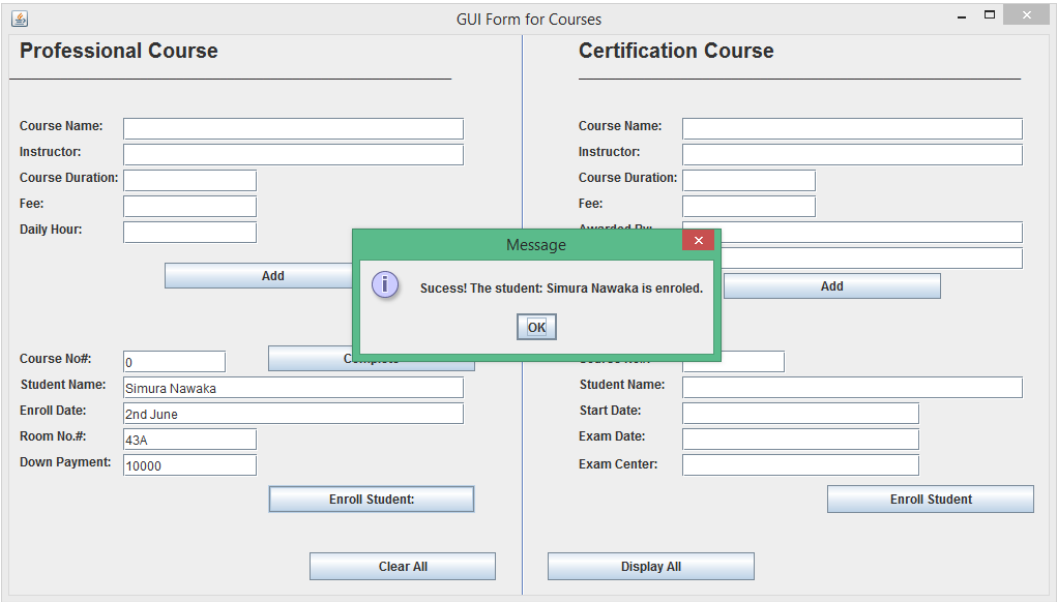

Figure 10: Enrolling student to Professional course

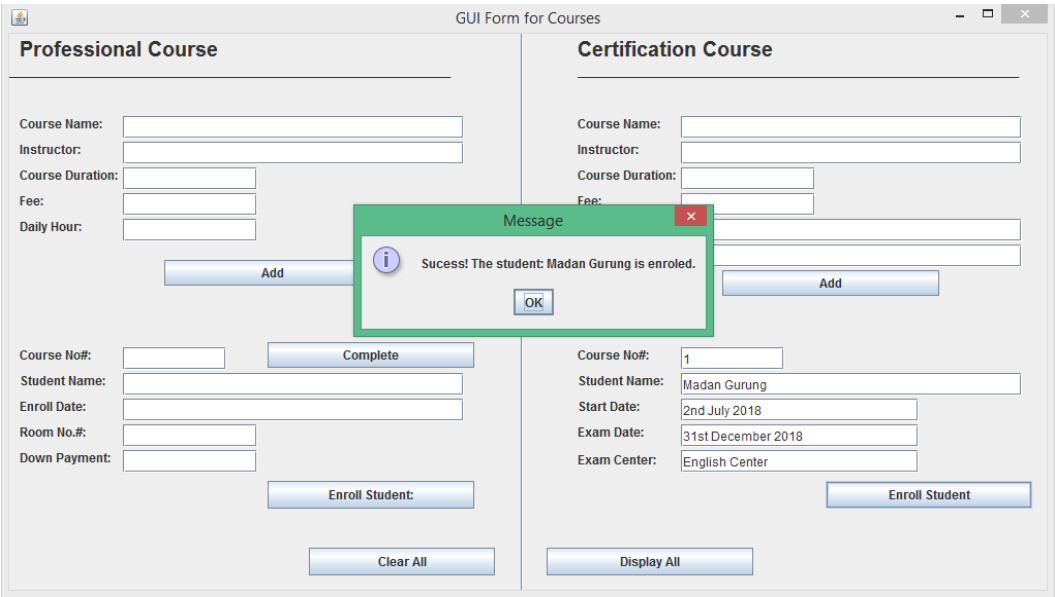

Figure 11: Enrolling student to Certification course

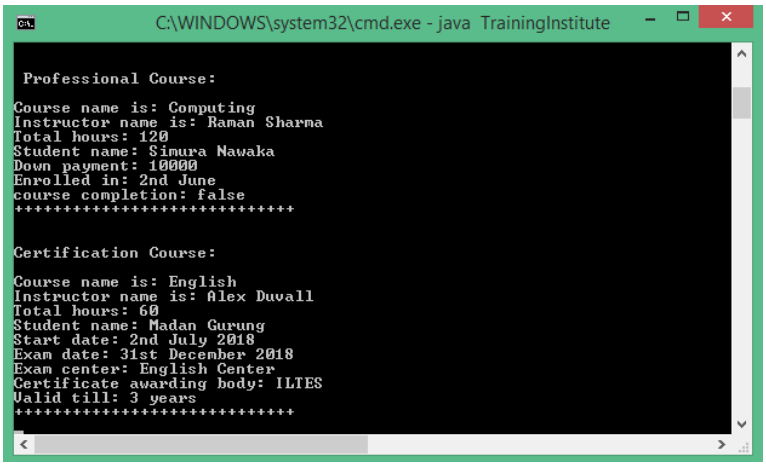

Figure 12: Details displayed after enrolling the students

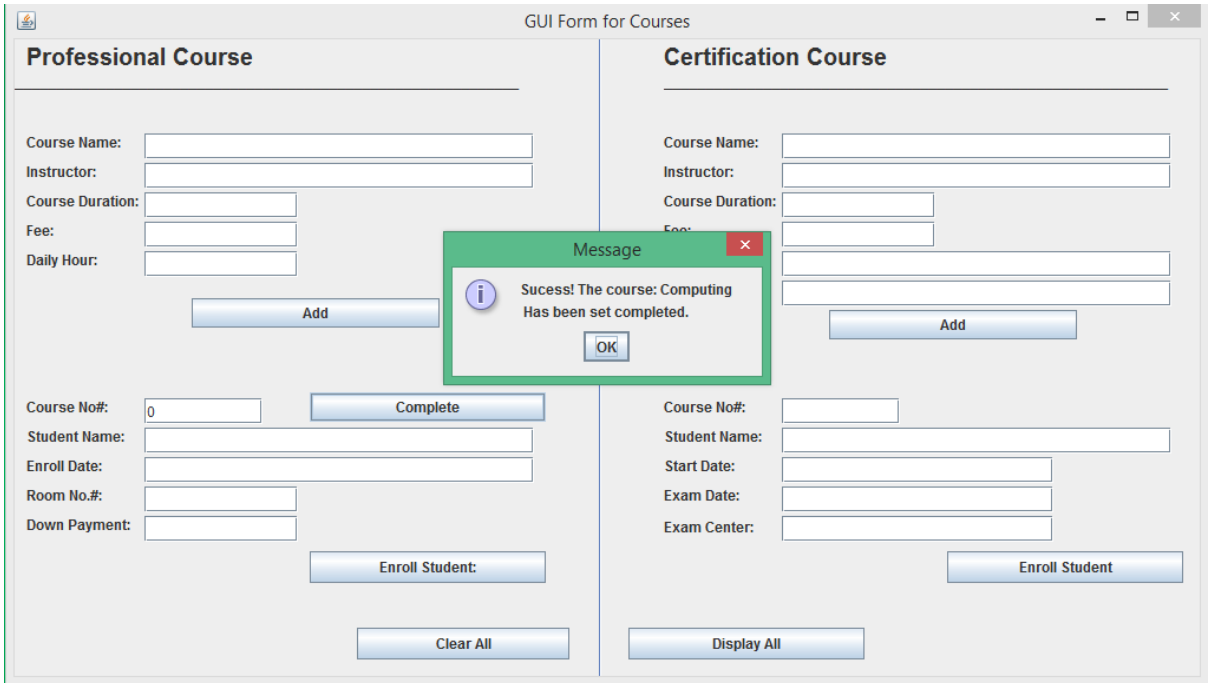

Figure 13: setting professional course as completed.

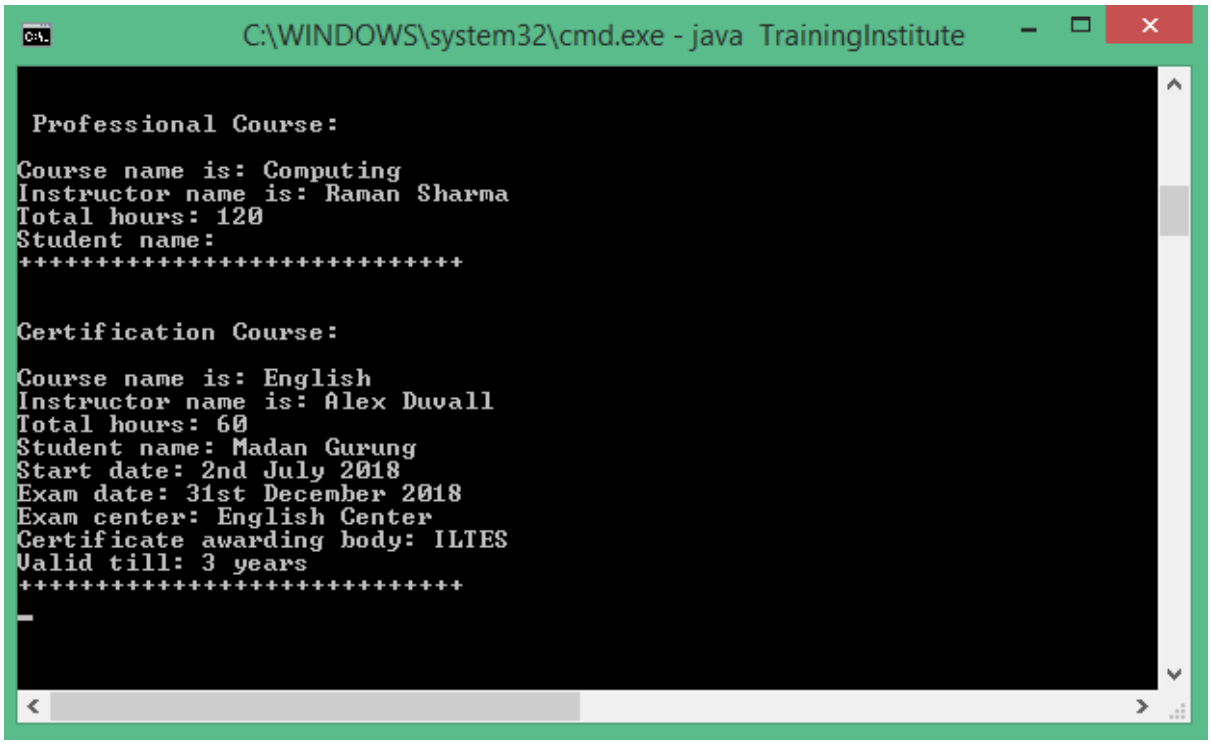

Figure 14: Enrolled student details gets erased from the program

5.3. Test 3:

Table 6: Test 3

Objective	Checking how the program handles invalid inputs
Action	Several invalid inputs are entered to test the program
Expected Result	The program should detect invalid inputs and handle exceptions
Actual Result	The invalid inputs are detected and appropriate messages were shown
Conclusion	Test Successful.

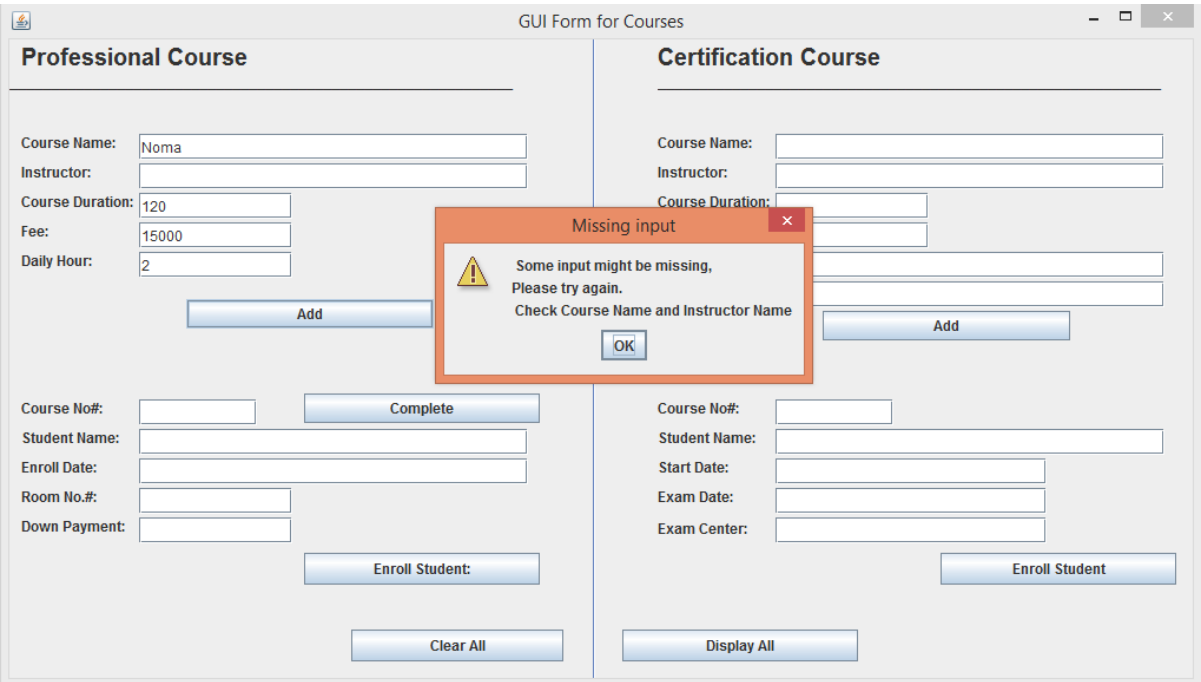

Figure 15: Triggering false entry detection to detect empty string input

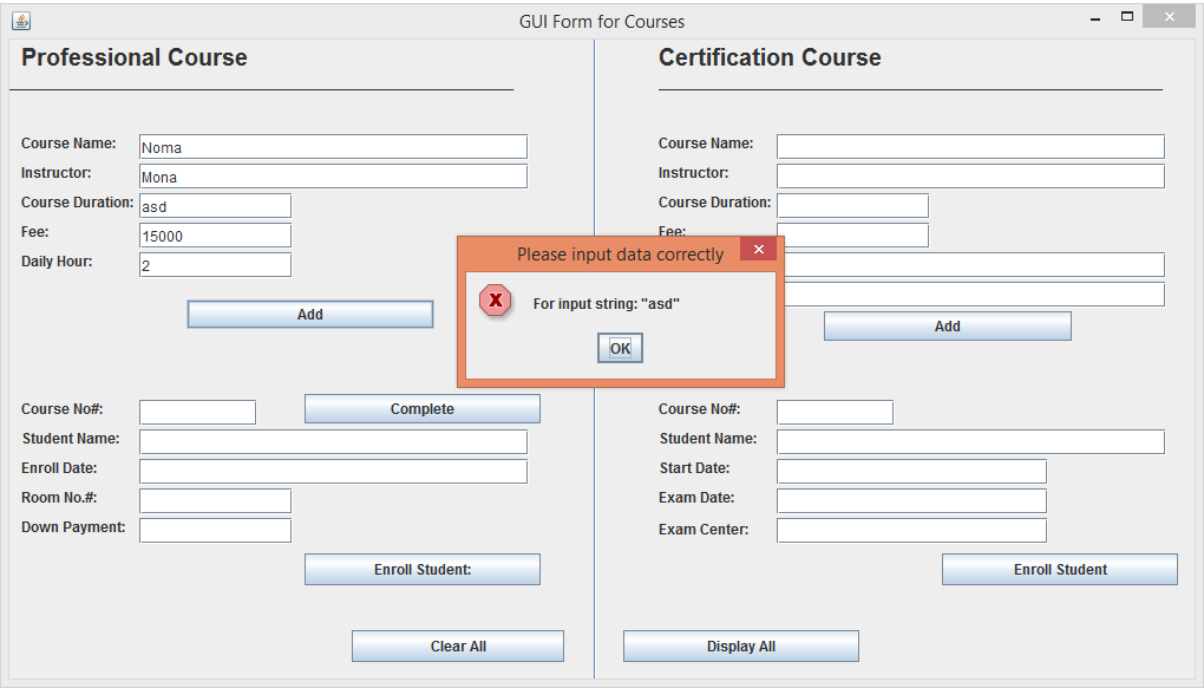

Figure 16: Triggering false entry detection to detect invalid format data

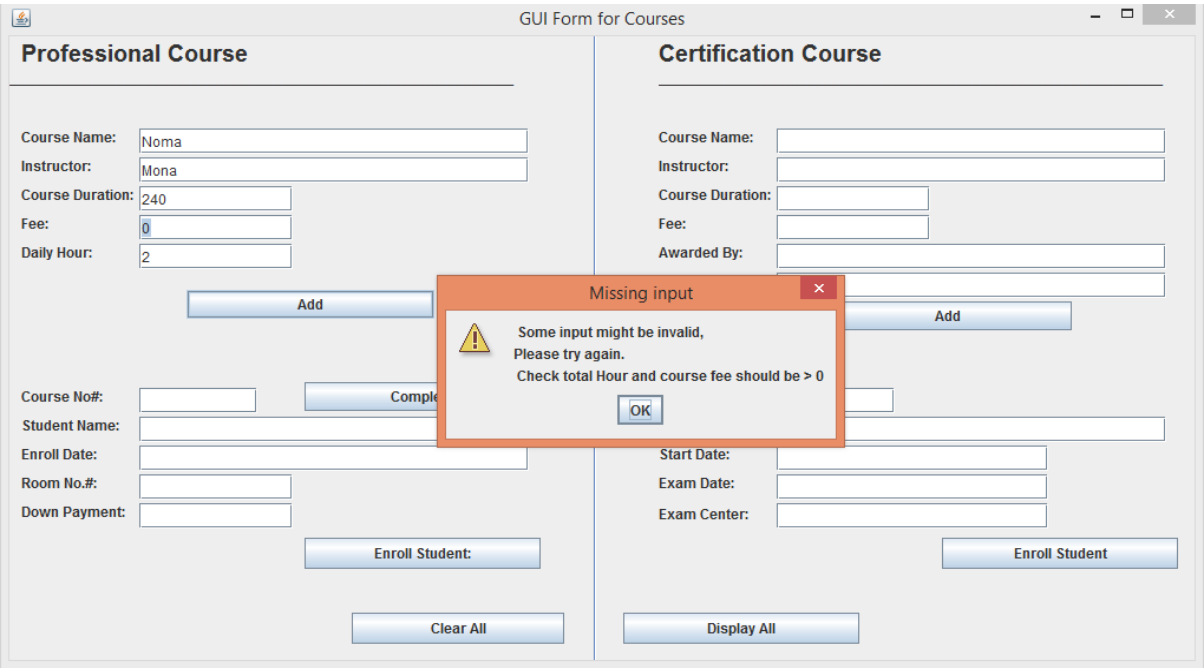

Figure 17: Triggering false entry detection to detect zero value in fields which take integer value

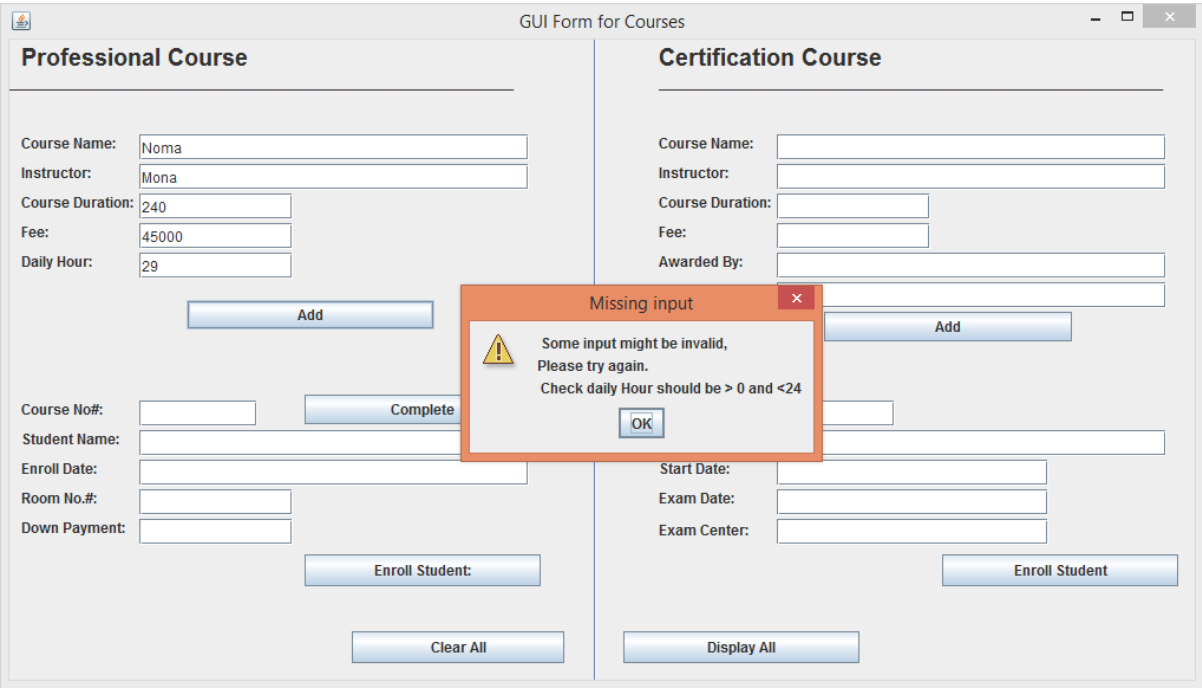

Figure 18: Triggering false entry detection to take valid daily hour

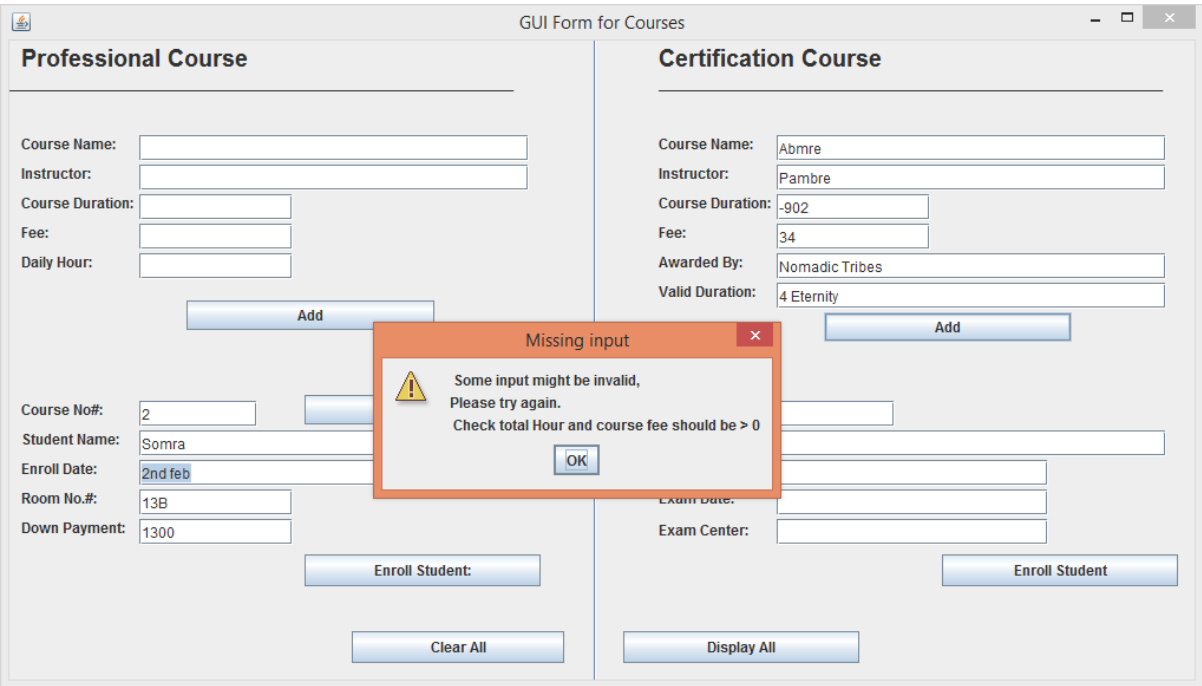

Figure 19: Triggering false entry detection to detect zero or negative value

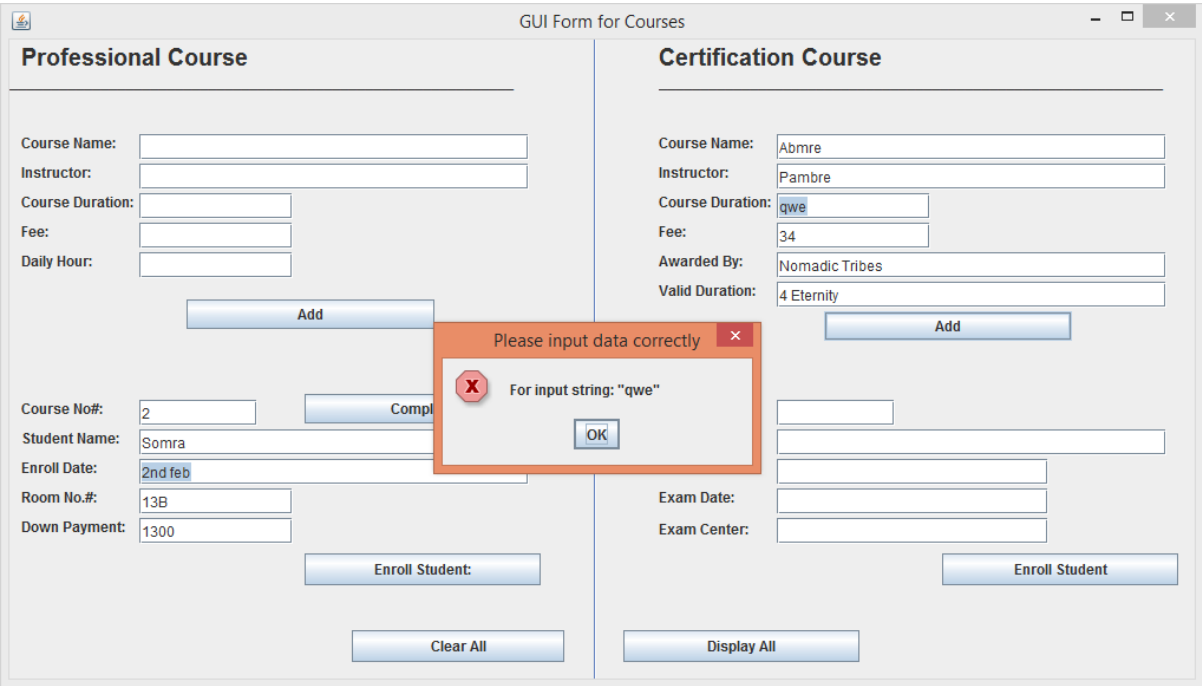

Figure 20: Triggering false entry detection to detect string in integer text field

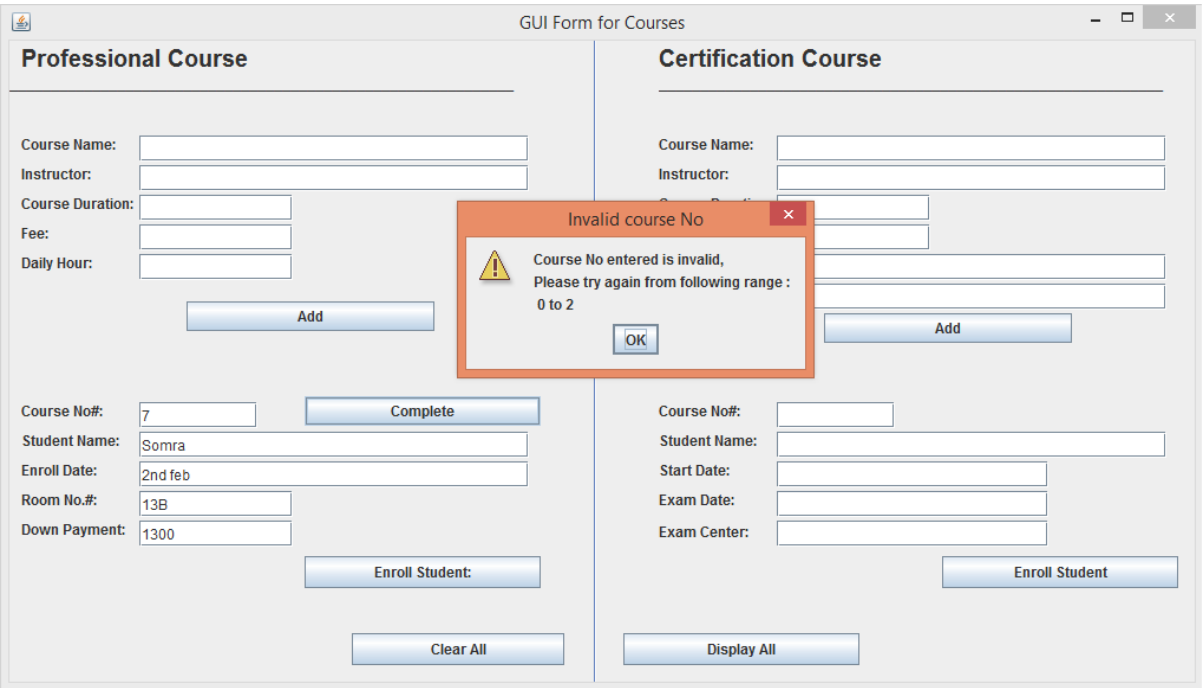

Figure 21: Triggering false entry detection to detect invalid course no

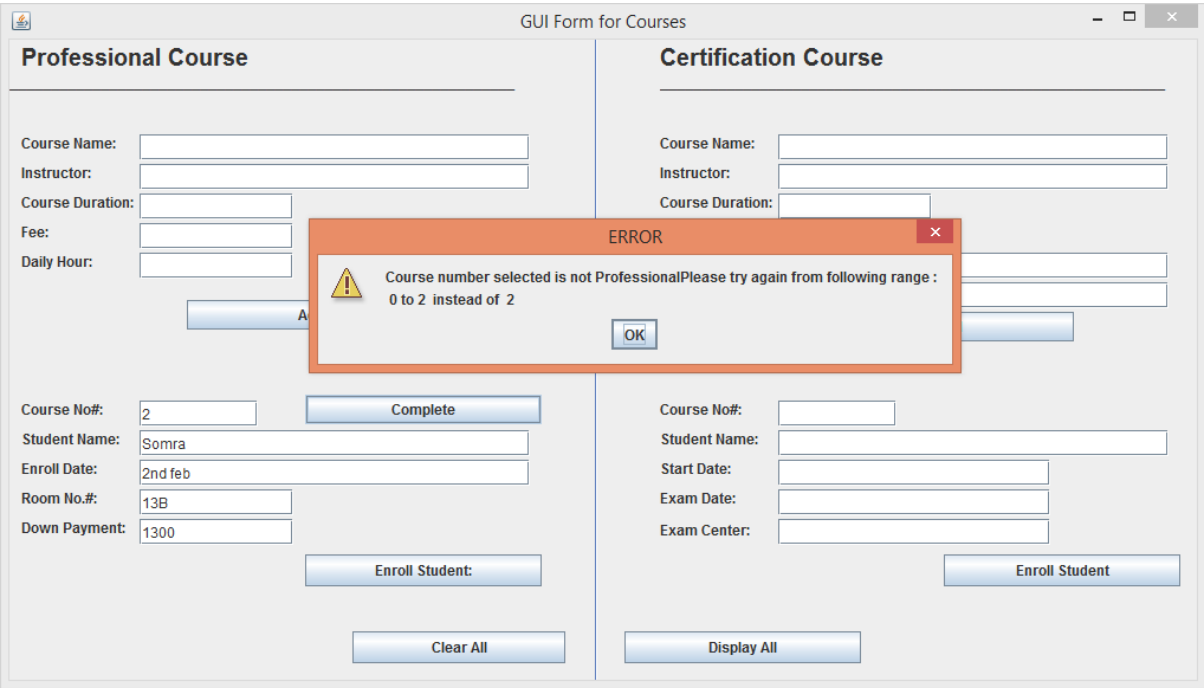

Figure 22: Showing error dialog box after entering certification course no in professional

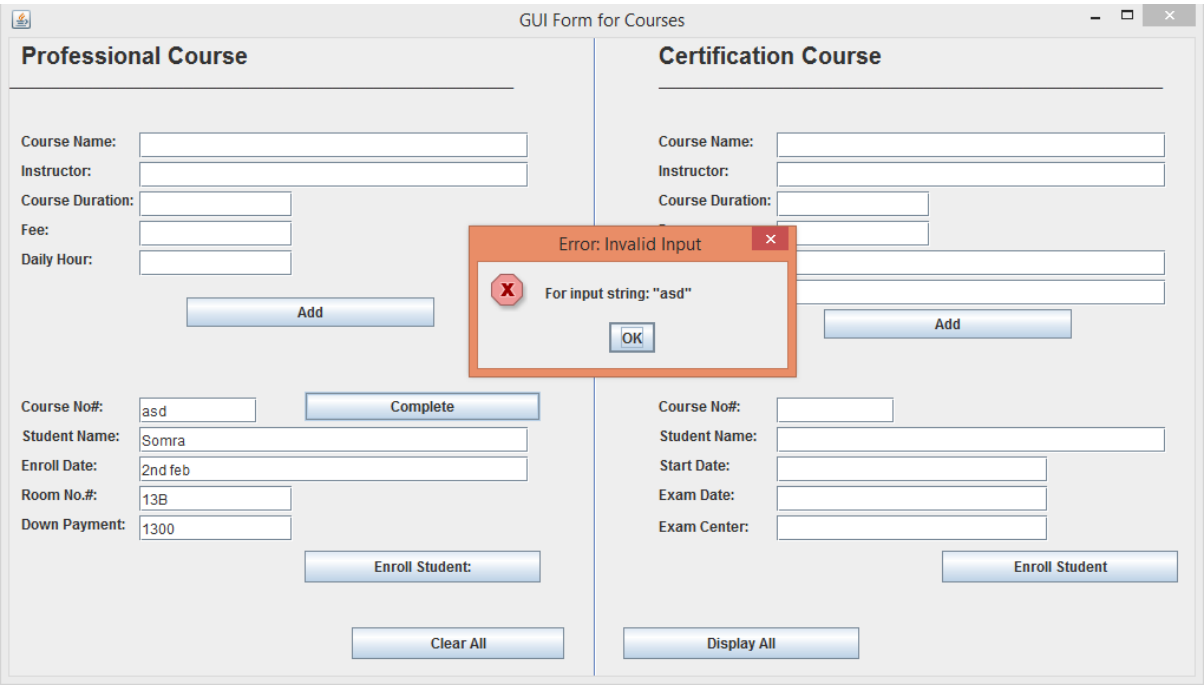

Figure 23: Triggering false entry detection to detect string in integer field

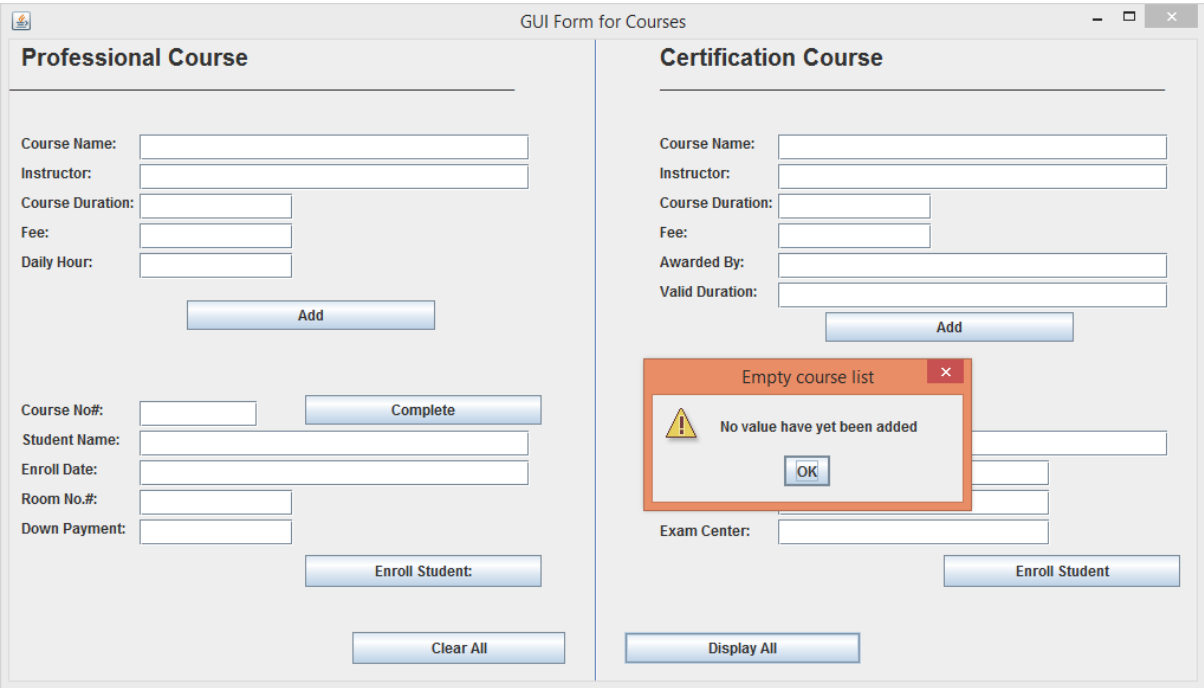

Figure 24: Error dialog box shown after attempting to display details without adding any data

6. Error Detection:

While developing any code in any programming languages we might face various errors due to incomplete or invalid lines of code. Compiler may throw many error messages for one error. So, fixing the first error and recompiling might solve many problems (Stringfellow, 2017). While creating the code various errors were found. The errors found can be classified into three different types and they are:

6.1. Compile-time Errors:

Errors such as Syntax errors, Missing identifiers, computation, no return statements, etcetera. Are caused due to incomplete code or invalid syntax. The examples are:

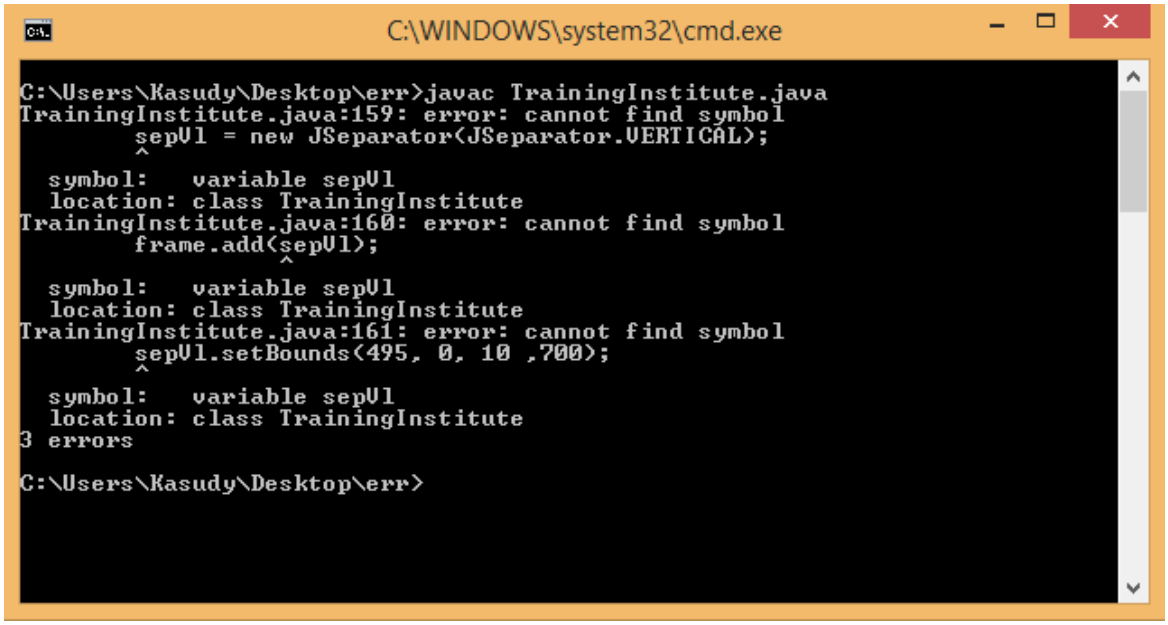

Figure 25: Cannot find symbol error

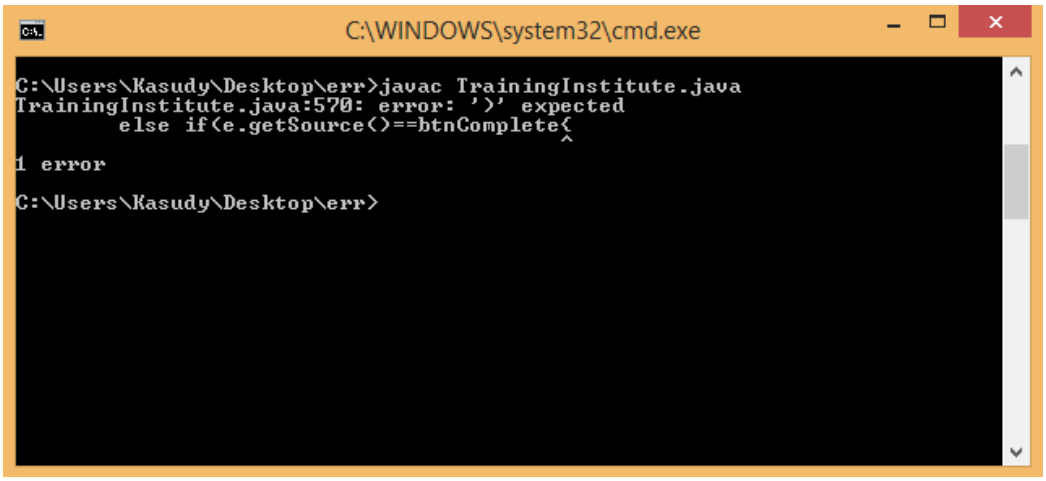

Figure 26: " " Expected error

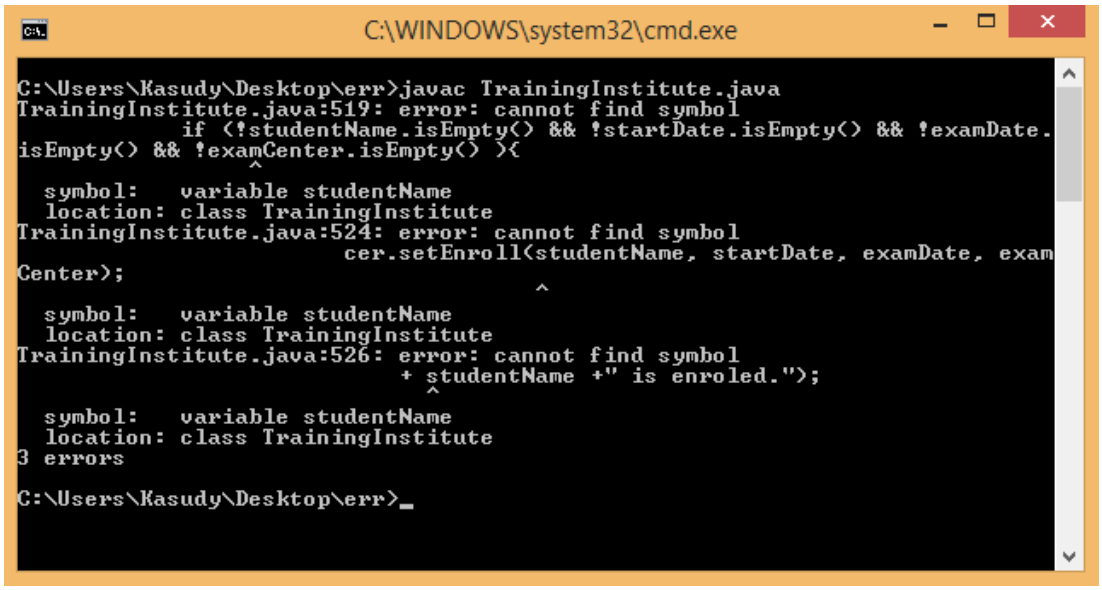

Figure 27: Cannot find symbol error

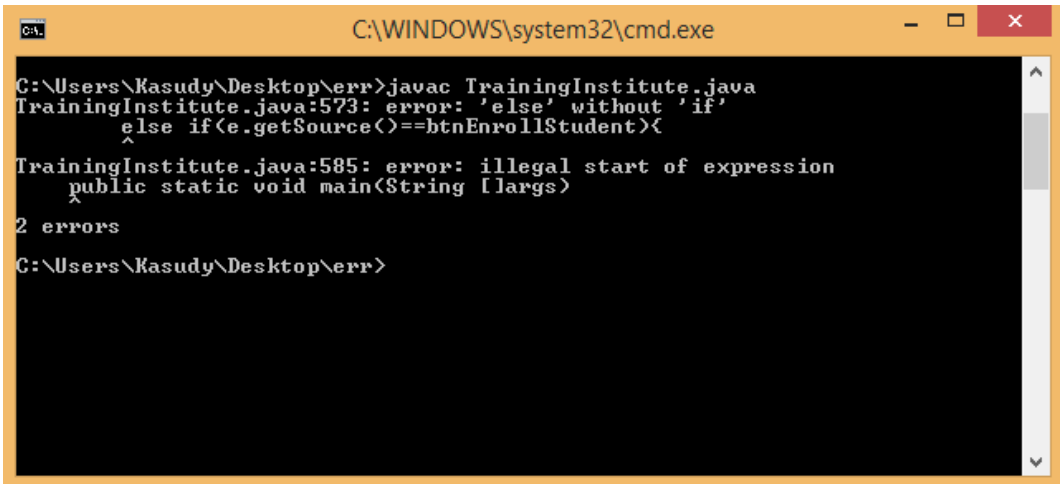

Figure 28: illegal start of expression due to missing braces

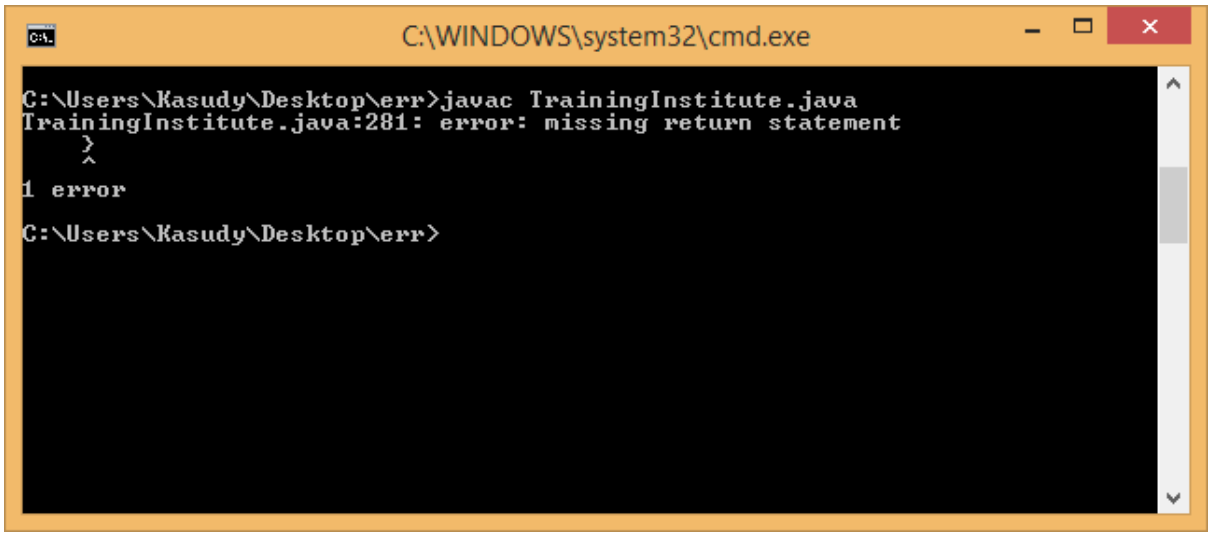

Figure 29: missing return statement error

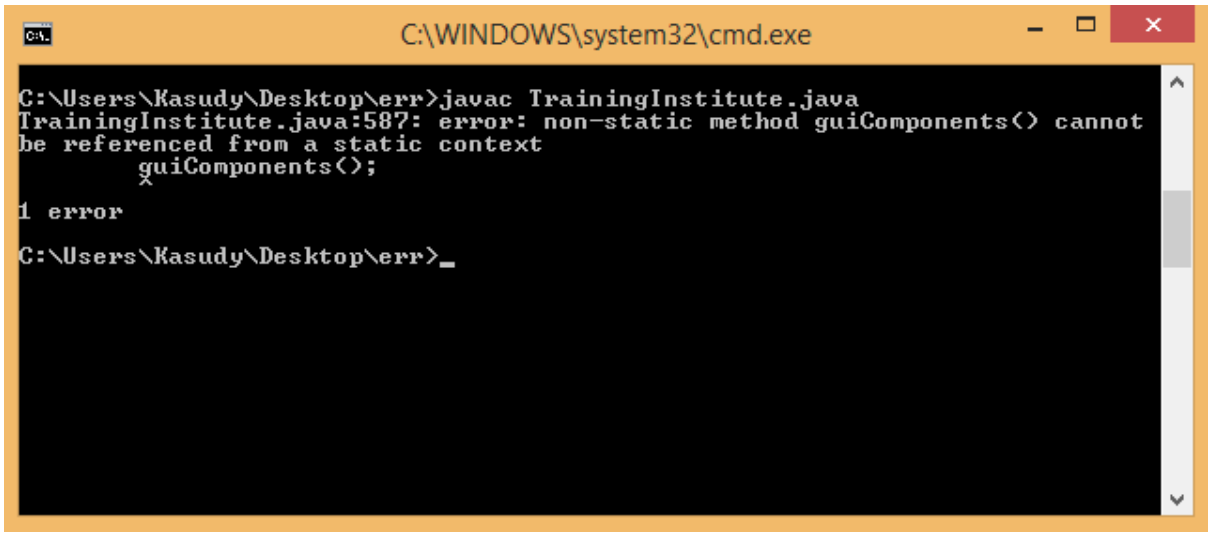

Figure 30: non static method referenced from main method which is static

6.2. Runtime errors:

Runtime error occurs when the java interpreter finds an error throws an exception while the program is running (Ben-Ari, 2007). There are different types of runtime errors Out of range, Dereferencing null (null pointer Exception), Computation (Number Format exception) etcetera. The runtime errors encountered are:

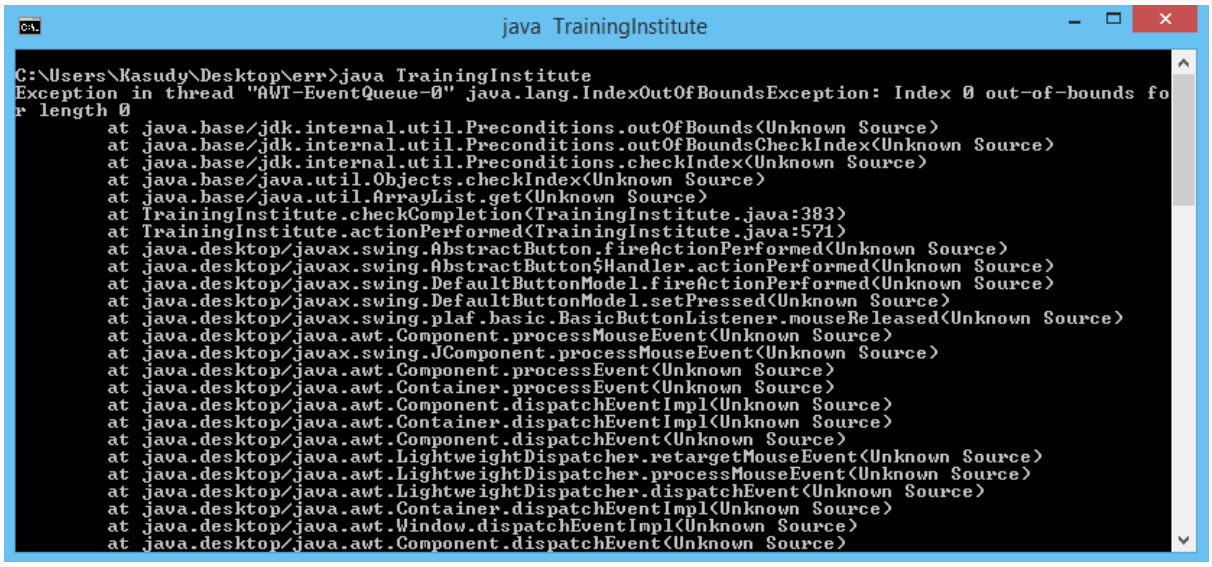

Figure 31: Array index out of bound exception

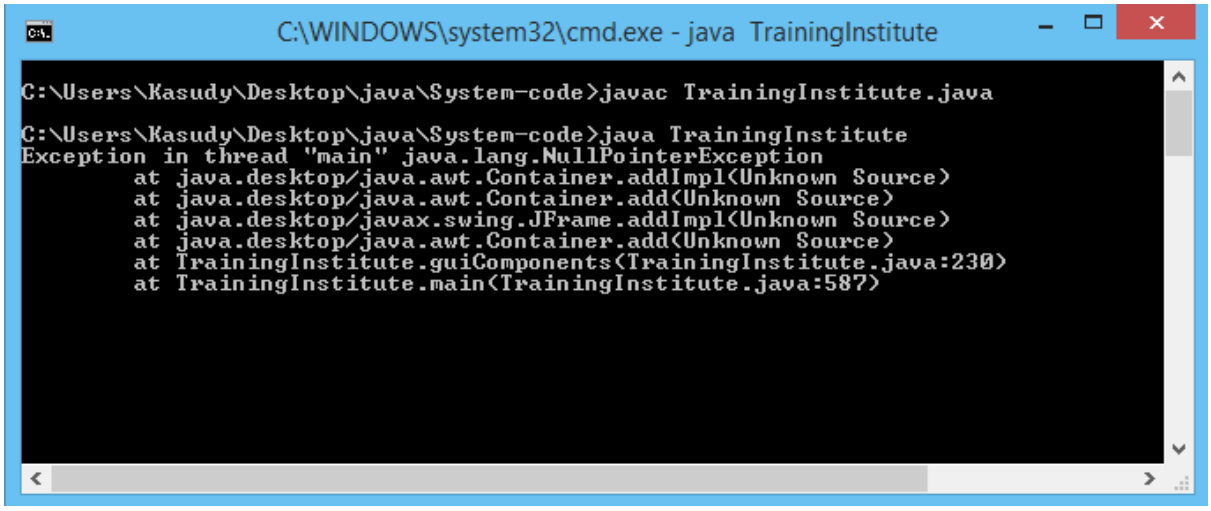

Figure 32: Null pointer exception thrown by an textbox which is not added to the frame but is called

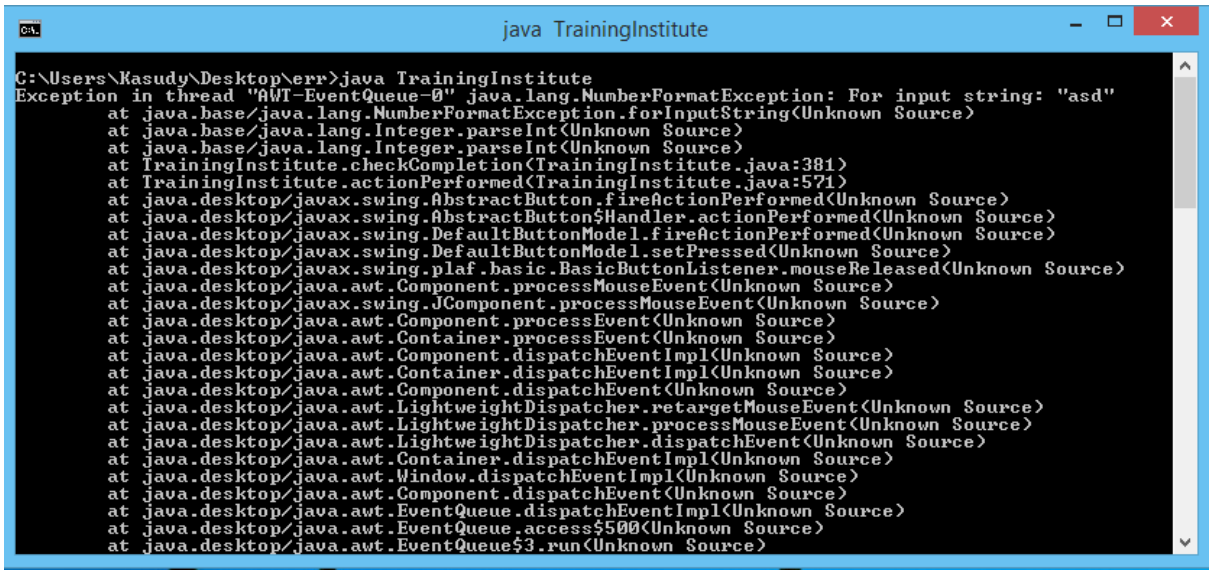

Figure 33: Number format exception caused by entering string in place of integer

6.3. Logical errors:

Logical errors are errors which doesn't show any error messages or any warning sign. It simply doesn't do the requested task and might do something else. An example of the logical error encountered in the process is as follows:

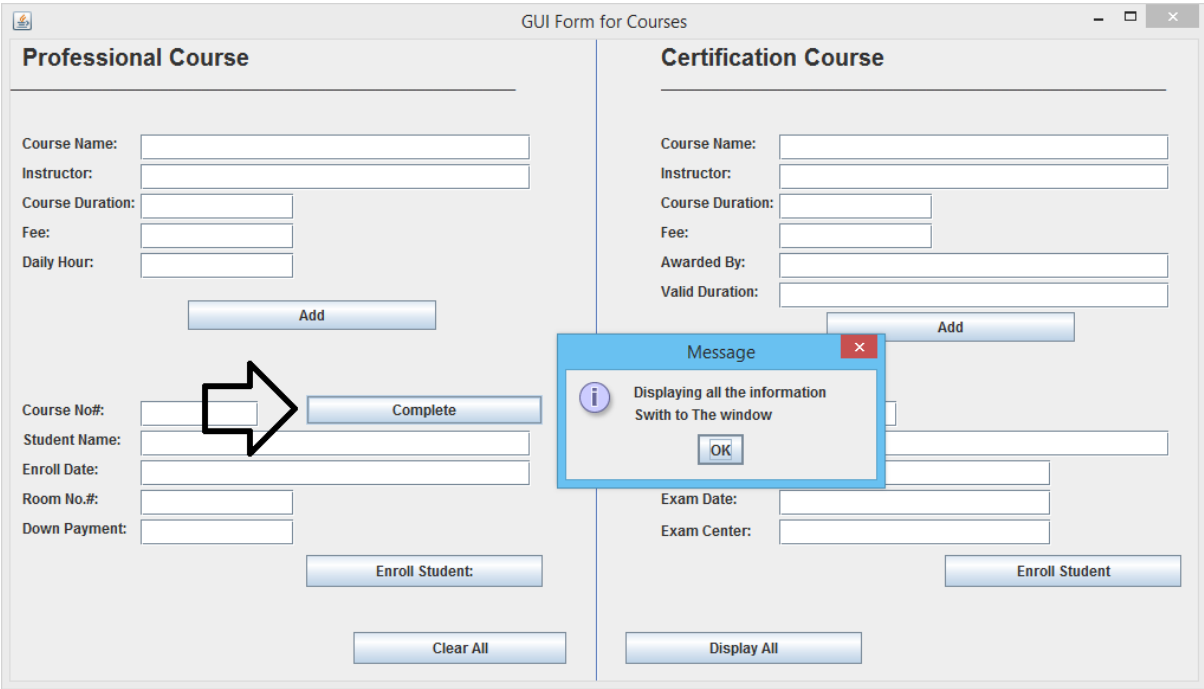

Figure 34: Complete button does not do what it is intended to do due to logical error

6.4. Error Solving:

The above mentioned errors were solved by understanding what was happening in the code. Many times some other errors might be causing other errors so solving the simplest errors from the start should be highly prioritized.

1. Compile time errors.

These errors are fairly simple to solve. As they are caused by simple syntax errors, multiple same code or incomplete coding they are easily pointed out by the compiler while compiling the class.

2. Runtime errors

These errors are little harder to solve as it requires exception handling features to avoid these errors. It is formed while running the program so the error details gets printed on the terminal. The Number format exception can be avoided by entering correct integer value to corresponding integer field. The null pointer exception can be avoided by deleting useless codes which call upon nonexistent entities and methods. The array lists out of bound error can be solved by entering number from a valid range number.

3. Logical errors

These errors cannot be found in compiler or terminal as the program doesn’t have any runtime or syntax errors, it is caused due to incorrect logic that is written in the code. This type of error might take very long time to solve as it depends on how long it takes to find the error. After finding the problem in the code,

7. Conclusion:

The coursework given could not have been completed without researching in the familiar topics and use of proper tools for documenting and coding. A lot of information on the topics were gathered to complete the following coursework which will be quite useful in future for practicing java. For developing the code, class diagram and pseudocode was first prepared so that the coding will be lot easier. As expected the coding was a lot easier due to the model prepared for the code.

The code was created on the basis of object oriented programming. It involves inheritance, exception handling and user interface built for ease of use. The object-oriented method of creating code involves classes and objects rather than logic and commands. This enables the programmer to code the objects in the programs like real world object with attributes and methods. The methods declared can be called any number of times as required which helps the code to be reused.

The code contained some errors caused by invalid syntax, missing variables, lack of exception handling and logical errors but testing the code helped a lot on recognizing the errors and weak points of the code. As a result, the testing made the code more reliable. After the testing was don the errors encountered and the test results were documented in this report.

References

Ben-Ari, M. (., 2007. *Compile and Runtime Errors in Java,* Rehovot : Department of Science Teaching.

Farrell, J., 2009. *Just Enough Programming Logic and Design.* 1 ed. Boston: Cengage Learning.

Jenkov, J., 2015. *Java Methods.* [Online]
Available at: http://tutorials.jenkov.com/java/methods.html
[Accessed 17 Jan 2018].

Knoernschild, K., 2002. *Java Design: Objects, UML, and Process.* 1st ed. Boston: Addison-Wesley Professional.

Kolling, D. B. a. M., 2009. In: S. D. I. Z. G. D. Tracy Dunkelberger, ed. *Objects First with Java.* s.l.:Pearson .

Poo, D., Kiong, . D. & Swarnalatha, . A., 2007. *Object-Oriented Programming and Java.* 2 ed. Berlin: Springer Science & Business Media.

Rouse, M., 2014. *What is object-oriented programming (OOP)? - Definition from WhatIs.com.* [Online]
Available at: http://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP
[Accessed 18 Jan 2018].

Stringfellow, A., 2017. *50 Common Java Errors and How to Avoid Them (Part 1) - DZone Java.* [Online]
Available at: https://dzone.com/articles/50-common-java-errors-and-how-to-avoid-them-part-1
[Accessed 19 1 2018].

Appendix

```
/**
 * Description
 *
 * @Rajat Shrestha
 * @ID : 17030954
 * @Version 21/4/2018
 */

import java.util.ArrayList;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JLabel;
import javax.swing.JFrame;
import javax.swing.JSeparator;
import javax.swing.JOptionPane;


public class TrainingInstitute implements ActionListener
{
    // defining class variables
    private ArrayList<Course> courseList = new ArrayList<Course>();
    private JFrame frame;
    private JSeparator sepVl;
    private JLabel lblDescription, lblInstructor, lblCourseDuration,
            lblFee, lblDailyHour, lblDownPayment,
            lblStudentName, lblEnrollDate, lblRoomNo,
            lblDescription2, lblInstructor2, lblCourseDuration2,
            lblFee2, lblStudentName2, lblStartDate, lblExamDate,
            lblCourseNo, lblCourseNo2, lblExamCenter,
            lblAwardedBy,lblValidDuration, lblCertificationCourse,
            lblProfessionalCourse, lblBorder, lblBorder2;
    private JTextField txtDescription, txtInstructor,
            txtCourseDuration, txtFee, txtDailyHour, txtDownPayment,
            txtStudentName, txtEnrollDate, txtRoomNo, txtStartDate,
            txtExamDate, txtDescription2, txtInstructor2,
            txtCourseDuration2, txtFee2, txtStudentName2, txtExamCenter,
            txtAwardedBy, txtValidDuration, txtCourseNo, txtCourseNo2;
    private JButton btnAdd, btnAdd2, btnComplete, btnEnrollStudent,
            btnEnrollStudent2, btnDisplayAll, btnClear;
```

```
    // Empty Constructor
    TrainingInstitute()
    {
    }

    // method for designing GUI
    public void guiComponents()
    {
        // Setting Frame for GUI elements
        JFrame frame= new JFrame("GUI Form for Courses");
        frame.setVisible(true);
        frame.setSize(1020,580);
        frame.setLayout(null);

        // GUI for Professional
        lblProfessionalCourse = new JLabel("Professional Course");
        frame.add(lblProfessionalCourse);
        lblProfessionalCourse.setFont(new Font("arial", Font.BOLD, 20));
        lblProfessionalCourse.setBounds(10, 5, 200 ,20);

        lblBorder = new JLabel("____________________________________");

        frame.add(lblBorder);
        lblBorder.setBounds(0, 29, 500 ,14);

        lblDescription = new JLabel("Course Name:");
        frame.add(lblDescription);
        lblDescription.setBounds(10, 80, 100 ,14);

        lblInstructor = new JLabel("Instructor:");
        frame.add(lblInstructor);
        lblInstructor.setBounds(10, 105, 100 ,14);

        lblCourseDuration = new JLabel("Course Duration:");
        frame.add(lblCourseDuration);
        lblCourseDuration.setBounds(10, 130, 100 ,14);

        lblFee = new JLabel("Fee:");
        frame.add(lblFee);
        lblFee.setBounds(10, 155, 100 ,14);

        lblDailyHour = new JLabel("Daily Hour:");
        frame.add(lblDailyHour);
        lblDailyHour.setBounds(10, 180, 100 ,14);
```

```
        btnAdd = new JButton("Add");
        frame.add(btnAdd);
        btnAdd.setBounds(150, 220, 210 ,25);
        btnAdd.addActionListener(this);

        lblCourseNo = new JLabel("Course No#:");
        frame.add(lblCourseNo);
        lblCourseNo.setBounds(10, 305, 100 ,14);

        lblStudentName = new JLabel("Student Name:");
        frame.add(lblStudentName);
        lblStudentName.setBounds(10, 330, 100 ,14);

        lblEnrollDate = new JLabel("Enroll Date:");
        frame.add(lblEnrollDate);
        lblEnrollDate.setBounds(10, 355, 100 ,14);

        lblRoomNo = new JLabel("Room No.#:");
        frame.add(lblRoomNo);
        lblRoomNo.setBounds(10, 380, 100 ,14);

        lblDownPayment = new JLabel("Down Payment:");
        frame.add(lblDownPayment);
        lblDownPayment.setBounds(10, 405, 100 ,14);

        txtDescription = new JTextField();
        frame.add(txtDescription);
        txtDescription.setBounds(110, 80, 330 ,22);

        txtInstructor = new JTextField();
        frame.add(txtInstructor);
        txtInstructor.setBounds(110, 105, 330 ,22);

        txtCourseDuration = new JTextField();
        frame.add(txtCourseDuration);
        txtCourseDuration.setBounds(110, 130, 130 ,22);

        txtFee = new JTextField();
        frame.add(txtFee);
        txtFee.setBounds(110, 155, 130 ,22);

        txtDailyHour = new JTextField();
        frame.add(txtDailyHour);
        txtDailyHour.setBounds(110, 180, 130 ,22);

        txtCourseNo = new JTextField();
        frame.add(txtCourseNo);
```

```
        txtCourseNo.setBounds(110, 305, 100 ,22);

        txtStudentName = new JTextField();
        frame.add(txtStudentName);
        txtStudentName.setBounds(110, 330, 330 ,22);

        txtEnrollDate = new JTextField();
        frame.add(txtEnrollDate);
        txtEnrollDate.setBounds(110, 355, 330 ,22);

        txtRoomNo = new JTextField();
        frame.add(txtRoomNo);
        txtRoomNo.setBounds(110, 380, 130 ,22);

        txtDownPayment = new JTextField();
        frame.add(txtDownPayment);
        txtDownPayment.setBounds(110, 405, 130 ,22);

        btnComplete = new JButton("Complete");
        frame.add(btnComplete);
        btnComplete.setBounds(250, 300, 200 ,25);
        btnComplete.addActionListener(this);

        btnEnrollStudent = new JButton("Enroll Student:");
        frame.add(btnEnrollStudent);
        btnEnrollStudent.setBounds(250, 435, 200 ,27);
        btnEnrollStudent.addActionListener(this);

        sepVl = new JSeparator(JSeparator.VERTICAL);
        frame.add(sepVl);
        sepVl.setBounds(495, 0, 10 ,700);

        // GUI of Certification
        lblCertificationCourse = new JLabel("Certification Course");
        frame.add(lblCertificationCourse);
        lblCertificationCourse.setFont(new Font("arial", Font.BOLD, 20));
        lblCertificationCourse.setBounds(550, 5, 200 ,20);

        lblBorder2 = new JLabel("__________________________________");
        frame.add(lblBorder2);
        lblBorder2.setBounds(550, 29, 498 ,14);

        lblDescription2 = new JLabel("Course Name:");
        frame.add(lblDescription2);
        lblDescription2.setBounds(550, 80, 100 ,14);

        lblInstructor2 = new JLabel("Instructor:");
```

```
        frame.add(lblInstructor2);
        lblInstructor2.setBounds(550, 105, 100 ,14);

        lblCourseDuration2 = new JLabel("Course Duration:");
        frame.add(lblCourseDuration2);
        lblCourseDuration2.setBounds(550, 130, 100 ,14);

        lblFee2 = new JLabel("Fee:");
        frame.add(lblFee2);
        lblFee2.setBounds(550, 155, 100 ,14);

        lblAwardedBy = new JLabel("Awarded By:");
        frame.add(lblAwardedBy);
        lblAwardedBy.setBounds(550, 180, 100 ,14);

        lblValidDuration = new JLabel("Valid Duration:");
        frame.add(lblValidDuration);
        lblValidDuration.setBounds(550, 205, 100 ,14);

        lblCourseNo2 = new JLabel("Course No#:");
        frame.add(lblCourseNo2);
        lblCourseNo2.setBounds(550, 305, 100 ,14);

        lblStudentName2 = new JLabel("Student Name:");
        frame.add(lblStudentName2);
        lblStudentName2.setBounds(550, 330, 100 ,14);

        lblStartDate = new JLabel("Start Date:");
        frame.add(lblStartDate);
        lblStartDate.setBounds(550, 355, 100 ,14);

        lblExamDate = new JLabel("Exam Date:");
        frame.add(lblExamDate);
        lblExamDate.setBounds(550, 380, 100 ,14);

        lblExamCenter = new JLabel("Exam Center:");
        frame.add(lblExamCenter);
        lblExamCenter.setBounds(550, 405, 600 ,18);

        txtDescription2 = new JTextField();
        frame.add(txtDescription2);
        txtDescription2.setBounds(650, 80, 330 ,22);

        txtInstructor2 = new JTextField();
        frame.add(txtInstructor2);
        txtInstructor2.setBounds(650, 105, 330 ,22);
```

```
        txtCourseDuration2 = new JTextField();
        frame.add(txtCourseDuration2);
        txtCourseDuration2.setBounds(650, 130, 130 ,22);

        txtFee2 = new JTextField();
        frame.add(txtFee2);
        txtFee2.setBounds(650, 155, 130 ,22);

        txtAwardedBy = new JTextField();
        frame.add(txtAwardedBy);
        txtAwardedBy.setBounds(650, 180, 330 ,22);

        txtValidDuration = new JTextField();
        frame.add(txtValidDuration);
        txtValidDuration.setBounds(650, 205, 330 ,22);

        txtCourseNo2 = new JTextField();
        frame.add(txtCourseNo2);
        txtCourseNo2.setBounds(650, 305, 100 ,22);

        txtStudentName2 = new JTextField();
        frame.add(txtStudentName2);
        txtStudentName2.setBounds(650, 330, 330 ,22);

        txtStartDate = new JTextField();
        frame.add(txtStartDate);
        txtStartDate.setBounds(650, 355, 230 ,22);

        txtExamDate = new JTextField();
        frame.add(txtExamDate);
        txtExamDate.setBounds(650, 380, 230 ,22);

        txtExamCenter = new JTextField();
        frame.add(txtExamCenter);
        txtExamCenter.setBounds(650, 405, 230 ,22);

        btnAdd2 = new JButton("Add");
        frame.add(btnAdd2);
        btnAdd2.setBounds(690, 230, 210 ,25);
        btnAdd2.addActionListener(this);

        btnEnrollStudent2 = new JButton("Enroll Student");
        frame.add(btnEnrollStudent2);
        btnEnrollStudent2.setBounds(790, 435, 200 ,27);
        btnEnrollStudent2.addActionListener(this);

        btnDisplayAll = new JButton("Display All");
```

```
        frame.add(btnDisplayAll);
        btnDisplayAll.setBounds(520, 500, 200 ,27);
        btnDisplayAll.addActionListener(this);

        btnClear = new JButton("Clear All");
        frame.add(btnClear);
        btnClear.setBounds(290, 500, 180 ,27);
        btnClear.addActionListener(this);
    }

    // method to override the method in ActionListener class
    @Override
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==btnAdd){
            addProfessional();
        }
        else if(e.getSource()==btnAdd2){
            addCertification();
        }
        else if(e.getSource()==btnClear){
            clearAll();
        }
        else if(e.getSource()==btnComplete){
            checkCompletion();
        }
        else if(e.getSource()==btnEnrollStudent){
            enrollProfessional();
        }
        else if(e.getSource()==btnEnrollStudent2){
            enrollCertification();
        }
        else if(e.getSource()==btnDisplayAll){
            dispalyMethod();
        }
    }




    // main method
    public static void main(String []args)
    {
        TrainingInstitute obj = new TrainingInstitute();
        obj.guiComponents();
    }
}
```

```
    // method to create professional object from txtfields
    public void addProfessional()
    {
        try{
            String courseName=txtDescription.getText();
            String instructorName = txtInstructor.getText();
            int totalHour =
                  Integer.parseInt(txtCourseDuration.getText().trim());
            int dailyHour =
                  Integer.parseInt(txtDailyHour.getText().trim());
            int courseFee = Integer.parseInt(txtFee.getText().trim());

            if (!courseName.isEmpty() && !instructorName.isEmpty()){
                if (totalHour > 0 && courseFee > 0){
                    if (dailyHour < 24 && dailyHour > 0){
                        Professional professionalCourse =
                               new Professional(totalHour,
                                    courseName,  instructorName,
                                    courseFee, dailyHour);
                        courseList.add(professionalCourse);

                        JOptionPane.showMessageDialog(null," Your record
                               is has been saved for:\n Course: " +
                              courseName + "\n By the instructor: " +
                              instructorName );
                        txtDescription.setText("");
                        txtCourseDuration.setText("");
                        txtDailyHour.setText("");
                        txtFee.setText("");
                        txtInstructor.setText("");
                    }
                    else {
                        JOptionPane.showMessageDialog(frame, " Some input
                               might be invalid, \n" + "Please try again.
                              \n Check daily Hour should be > 0 and <24",
                              "Missing", JOptionPane.WARNING_MESSAGE);
                    }
                }
                else {
                    JOptionPane.showMessageDialog(frame, " Some input
                         might be invalid, \n" + "Please try again. \n
                        Check total Hour and course fee should be > 0",
                        "Missing input", JOptionPane.WARNING_MESSAGE);
                }
            }
            else {
                JOptionPane.showMessageDialog(frame, " Some input might be
```

```
                 missing, \n" + "Please try again. \n Check
                Course Name and Instructor Name",
               "Missing input", JOptionPane.WARNING_MESSAGE);
            }
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(frame, e.getMessage(),
            "Please input data correctly", JOptionPane.ERROR_MESSAGE);
        }
    }



    // method to set the course as completed
    public void checkCompletion()
    {
        try{
            int courseNo = Integer.parseInt(txtCourseNo.getText().trim());
            if (courseNo >= 0 && courseNo <= courseList.size()) {
                if (courseList.get(courseNo) instanceof Professional) {
                    Professional professional = (Professional)
                                          courseList.get(courseNo);
                    professional.setCompleted();
                    JOptionPane.showMessageDialog(null,"Sucess! The
                      course: " + courseList.get(courseNo).getCourseName()
                        + "\n Has been set completed.");
                }
                else {
                    JOptionPane.showMessageDialog(null, "Course number
                        selected is not Professional" + "Please try again
                        from following range : \n 0 to " +
                        (courseList.size() - 1) + "  instead of  " +
                        (courseNo), "ERROR", JOptionPane.WARNING_MESSAGE);
                }
            }
            else{
                    JOptionPane.showMessageDialog(null, "Course No entered
                        is invalid, \n" + Please try again from following
                        range : \n 0 to " + (courseList.size() - 1) ,
                      "Invalid course No", JOptionPane.WARNING_MESSAGE);
                }
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(frame, e.getMessage(),
            "Error: Invalid Input ", JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
    // method to create certification object from txtfields and append it
to the list

    public void addCertification()
    {
        try {
            String courseName = txtDescription2.getText();
            String instructorName = txtInstructor2.getText();
            String validTill = txtValidDuration.getText();
            String certificateAwardedBy = txtAwardedBy.getText();
            int courseFee = Integer.parseInt(txtFee2.getText().trim());
            int totalHours = Integer.parseInt(txtCourseDuration2.getText(
                                                                ).trim());

            if (!courseName.isEmpty() && !instructorName.isEmpty() &&
                !validTill.isEmpty() && !certificateAwardedBy.isEmpty()){
                if (courseFee > 0 && totalHours > 0){

                    Certification certificationCourse =
                         new Certification(courseName, totalHours,
                                 instructorName, validTill, courseFee,
                                 certificateAwardedBy);
                    courseList.add(certificationCourse);

                    JOptionPane.showMessageDialog(null," Your record
                        has been saved for:\n course: " + courseName + "\n
                        By the instructor: " + instructorName );
                    txtAwardedBy.setText("");
                    txtValidDuration.setText("");
                    txtDescription2.setText("");
                    txtInstructor2.setText("");
                    txtCourseDuration2.setText("");
                    txtFee2.setText("");
                }
                else {
                    JOptionPane.showMessageDialog(frame, " Some input
                        might be invalid, \n" + "Please try again. \n
                        Check total Hour and course fee should be > 0",
                        "Missing input", JOptionPane.WARNING_MESSAGE);
                }
            }
            else {
                JOptionPane.showMessageDialog(frame, " Some input might be
                        missing, \n" + "Please try again.", "Missing
                        input", JOptionPane.WARNING_MESSAGE);
            }
```

```
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(frame, e.getMessage(),
            "Please input data correctly", JOptionPane.ERROR_MESSAGE);
        }
    }









    // method to clear all the
    public void clearAll()
    {
        txtDescription.setText("");
        txtInstructor.setText("");
        txtCourseDuration.setText("");
        txtFee.setText("");
        txtDailyHour.setText("");
        txtDownPayment.setText("");
        txtStudentName.setText("");
        txtEnrollDate.setText("");
        txtRoomNo.setText("");
        txtStartDate.setText("");
        txtExamDate.setText("");
        txtDescription2.setText("");
        txtInstructor2.setText("");
        txtCourseDuration2.setText("");
        txtFee2.setText("");
        txtStudentName2.setText("");
        txtExamCenter.setText("");
        txtAwardedBy.setText("");
        txtValidDuration.setText("");
        txtCourseNo.setText("");
        txtCourseNo2.setText("");
        JOptionPane.showMessageDialog(null, "All fields are cleared");
    }
```

```
    // method to displayy all the entered details
    public void dispalyMethod()
    {
        if (courseList.size() > 0 ){
            JOptionPane.showMessageDialog(null, "Displaying all the
                              information \n" + "Swith to The window");
            for (Course c: courseList){
                if(c instanceof Professional) {
                    System.out.println("\n\n Professional Course:\n");
                    ((Professional)c).display();
                    System.out.println("++++++++++++++++++++++++++++");
                }
                if(c instanceof Certification) {
                    System.out.println("\n\nCertification Course:\n");
                    ((Certification)c).display();
                    System.out.println("++++++++++++++++++++++++++++");
                }
            }
        }
        else {
            JOptionPane.showMessageDialog(null, "No value have yet been
                                        added", "Empty course list",
                                        JOptionPane.WARNING_MESSAGE);
        }
    }
```

```
    // method to enroll a student to professional class
    public void enrollProfessional()
    {
        try{
            int downPayment = Integer.parseInt(txtDownPayment.getText(
                                                        ).trim());
            int courseNo = Integer.parseInt(txtCourseNo.getText().trim());
            String studentName = txtStudentName.getText();
            String enrolDate = txtEnrollDate.getText();
            String roomNo = txtRoomNo.getText();

            if (!studentName.isEmpty() && !enrolDate.isEmpty() &&
                  !roomNo.isEmpty() && downPayment >= 0){

                if(courseNo >= 0 && courseNo < courseList.size()){
                    Course c = courseList.get(courseNo);
                    if(c instanceof Professional){
                        Professional pObj = (Professional)c;
                        pObj.setEnroll(roomNo, downPayment,
                                           enrolDate, studentName);
                        JOptionPane.showMessageDialog(null,"Sucess! The
                              student: "+ studentName +" is enroled.");
                        txtCourseNo.setText("");
                        txtStudentName.setText("");
                        txtEnrollDate.setText("");
                        txtRoomNo.setText("");
                        txtDownPayment.setText("");
                    }
                    else {
                        JOptionPane.showMessageDialog(null, "Course number
                              selected is not Professional" "Please try
                              again from following range : \n 0 to " +
                              (courseList.size() - 1) +
                            " instead of " + (courseNo) , "ERROR",
                            JOptionPane.WARNING_MESSAGE);
                    }
                }
                else{
                    JOptionPane.showMessageDialog(null, "Course No entered
                        is invalid, \n" + "Please try again from following
                        range: \n 0 to " + (courseList.size() - 1) ,
                        "Invalid course No", JOptionPane.WARNING_MESSAGE);
                }
            }
```

```
            else {
                JOptionPane.showMessageDialog(frame, " Some input might be
                        missing, \n" +"Please try again.", "Missing
                        input", JOptionPane.WARNING_MESSAGE);
            }
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(frame, e.getMessage(),
            "Please input data correctly", JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
    // method to enroll a student to certification class
    public void enrollCertification()
    {
          try{
                int courseNo = Integer.parseInt(txtCourseNo2.getText(
                                                        ).trim());
                String studentName = txtStudentName2.getText();
                String startDate = txtStartDate.getText();
                String examDate = txtExamDate.getText();
                String examCenter = txtExamCenter.getText();

                if (!studentName.isEmpty() && !startDate.isEmpty() &&
                         !examDate.isEmpty() && !examCenter.isEmpty() ){
                      if(courseNo >= 0 && courseNo < courseList.size()){
                            Course c = courseList.get(courseNo);
                            if(c instanceof Certification){
                                  Certification cer= (Certification)c;
                                  cer.setEnroll(studentName, startDate,
                                                examDate, examCenter);

                                  JOptionPane.showMessageDialog(null,
                                        "Sucess! The student: "
                                        + studentName +" is enroled.");

                                  txtCourseNo2.setText("");
                                  txtStudentName2.setText("");
                                  txtStartDate.setText("");
                                  txtExamDate.setText("");
                                  txtExamCenter.setText("");
```

```
                        }
                        else {
                              JOptionPane.showMessageDialog(null,
                              "Course number selected is not
                              Certification" + "Please try again
                              from following range : \n 0 to " +
                              (courseList.size() - 1) +" instead of
                              " + (courseNo) , "ERROR",
                              JOptionPane.WARNING_MESSAGE);
                        }
                  }
                  else {
                        JOptionPane.showMessageDialog(null, "Course
                              No entered is invalid, \n" + "Please
                              try again from following range : \n 0
                              to " + (courseList.size() - 1)
                              ,"Invalid course No",
                              JOptionPane.WARNING_MESSAGE);
                  }
            }
            else {
                  JOptionPane.showMessageDialog(frame, " Some input
                         might be missing, \n" + "Please try
                        again.", "Missing input",
                        JOptionPane.WARNING_MESSAGE);
            }
      }
      catch(Exception e) {
            JOptionPane.showMessageDialog(frame, e.getMessage(),
                        "Please input data correctly",
                        JOptionPane.ERROR_MESSAGE);
      }
}
```

```
/**
 * Description
 *
 * @Rajat Shrestha
 * @ID : 17030954
 * @Version 11/1/2018
 */
public class Course
{
    // defining variables
    public String courseName;
    public String instructorName;
    public String studentName;
    public int totalHours;

    // constructor method for Course
    public Course(String courseName, String instructorName,
                  int totalHours)
    {
        this.instructorName = instructorName;
        this.courseName = courseName;
        this.totalHours = totalHours;
        this.studentName = " ";
    }

    //acessor methods for each variable
    public String getCourseName()
    {
        return courseName;
    }
    public String getInstructorName()
    {
        return instructorName;
    }
    public String getStudentName()
    {
```

```
        return studentName;
    }
    public int getTotalHours()
    {
        return totalHours;
    }

    // method to set student's name
    public void setStudentName(String studentName)
    {
        this.studentName = studentName;
    }

    // display method to display details
    public void display()
    {
        System.out.println("Course name is: "+ courseName);
        System.out.println("Instructor name is: "+ instructorName);
        System.out.println("Total hours: "+ totalHours);

        if (!studentName.equals(" ")) {
            System.out.println("Student name: "+ studentName);
        }
    }
}
```

```
/**
 * Description
 *
 * @Rajat Shrestha
 * @ID : 17030954
 * @Version 11/1/2018
 */
public class Professional extends Course
{
    // defining variables
    int courseFee;
    String enrollDate;
    String roomNo;
    int dailyHour;
    int downPayment;
    boolean started;
    boolean completed;

    // Constructor for Professional class
    public  Professional(int  totalHours,  String  courseName,  String
                        instructorName, int courseFee, int dailyHour)
    {
        super(courseName, instructorName, totalHours);
        this.courseFee = courseFee;
        this.dailyHour = dailyHour;
        enrollDate = " ";
        roomNo = " ";
        downPayment = 0;
        started = false;
        completed  = false;
    }

    // Accessor method for each variable
    public int getCourseFee()
    {
        return courseFee;
    }
    public int getDailyHour()
```

```
    {
        return dailyHour;
    }
    public int getDownPayment()
    {
        return downPayment;
    }
    public String getEnrollDate()
    {
        return enrollDate;
    }
    public String getRoomNo()
    {
        return roomNo;
    }
    public boolean getStarted()
    {
        return started;
    }
    public boolean getCompleted()
    {
        return completed;
    }

    // Setting course fee
    public void setCourseFee(int courseFee)
    {
        this.courseFee = courseFee;
    }

    // Setting daily hour
    public void setDailyHour(int dailyHour)
    {
        this.dailyHour = dailyHour;
    }

    // Method for enrolling student
```

```
    public  void  setEnroll(String  roomNo,  int  downPayment,  String
                            enrollDate, String studentName)
    {
        if (started) {
            System.out.println("Sorry, course already started");
            System.out.println("By instructor: " + instructorName);
            System.out.println("In the room: " + roomNo);
        }
        else {
            setStudentName(studentName);
            this.enrollDate = enrollDate;
            this.downPayment = downPayment;
            this.roomNo = roomNo;
            started = true;
            completed  = false;
        }
    }

    // Method to set course completion
    public void setCompleted()
    {
        if (completed){
            System.out.println("The course has already completed");
        }
        else {
            setStudentName("");
            this.enrollDate = " ";
            this.roomNo = " ";
            this.downPayment = 0;
            started = false;
            completed  = true;
        }
    }

    // method to display course details
    public void getCourse()
    {
        System.out.println("Course name: " + getCourseName());
```

```
        System.out.println("Instructor: " +  getInstructorName());
        System.out.println("Course fee: " + getCourseFee());
    }

    //  display method to display details
    public void display()
    {
        super.Display();
        if (started) {
            System.out.println("Down payment: " + downPayment);
            System.out.println("Enrolled in: "+ enrollDate);
            System.out.println("course completion: "+ completed);
        }
    }

}
```

```
/**
 * Description
 *
 * @Rajat Shrestha
 * @ID : 17030954
 * @Version 11/1/2018
 */
public class Certification extends Course
{
    // defining variables
    boolean started;
    int courseFee;
    String examDate;
    String startDate;
    String examCenter;
    String certificateAwardedBy;
    String validTill;

    // constructor method for Certification
    public Certification(String courseName, int totalHours, String
                    instructorName, String validTill, int courseFee,
                     String certificateAwardedBy)
    {
        super(courseName, instructorName, totalHours);
        this.certificateAwardedBy = certificateAwardedBy;
        this.validTill = validTill;
        this.courseFee = courseFee;

        examCenter = " ";
        started = false;
        examDate = " ";
        startDate = " ";
    }

    //accessor methods for each variables
    public boolean getStarted()
    {
        return started;
```

```
    }
    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;
    }

    // method to set course fee
    public void setCourseFee(int courseFee)
    {
        if (started) {
             System.out.println("Sorry, the course has already been
                                started ");
        }
        else {
            this.courseFee = courseFee ;
        }
    }
```

```
    // Method for enrolling student
    public  void  setEnroll(String  studentName,  String  startDate,
                            String examDate, String examCenter)
    {
        if (started) {
            System.out.println("Sorry,   the   course   has   already
started");
            System.out.println("From: " + startDate);
        }
        else {
            setStudentName(studentName);
            this.startDate = startDate;
            this.examDate = examDate;
            this.examCenter = examCenter;
            started = true;
        }
    }

    // display method to display details
     public void display()
    {
        super.Display();
        if (started) {
            System.out.println("Start date: " + startDate);
            System.out.println("Exam date: "+ examDate);
            System.out.println("Exam center: "+ examCenter);
            System.out.println("Certificate   awarding   body:   "+
                               certificateAwardedBy);
            System.out.println("Valid till: "+ validTill);
        }
    }
}
```

Appendix 2

Course (super class)
+ courseName: String + instructorName: String + studentsName: String + totalHours: Integer
+ getTotalHours(): Integer + getCoursename(): String + getInstructorName(): String + getStudentName(): String + setStudentName(studentName) : void + display(): void

Professional
– courseFee: Integer – dailyHour: Integer – downPayment: Integer – started: Boolean – completed: Boolean – enrollDate: String – roomNo: String
+ getCourseFee(): Integer + getDailyHour() : Integer + getDownPayment(): Integer + getStarted(): Boolean + getCompleted(): Boolean + getEnrollDate(): String + getRoomNo(): String + setCourseFee(courseFee): void + setDailyHour(dailyHour): void + setEnroll(roomNo, downPayment, enrollDate, studentName): void + setCompleted(): void + getCourse() : void + display(): void

Certification
– courseFee: Integer – started: Boolean – examDate: String – startDate: String – examCenter: String – certificateAwardedBy: String – validTill: String
+ getCourseFee(): Integer + getStarted(): Boolean + getExamDate(): String + getStartDate(): String + getExamCenter(): String + getCertificateAwardedBy(): String + getValidTill(): String + setCourseFee(courseFee): void + setEnroll(studentName, startDate, examDate, examCenter): void + display(): void

Pseudocode

Course class

```
CALL getCourseName() : String
        DO
                RETURN courseName
        END DO
CALL getInstructorName() : String
        DO
                RETURN instructorName
        END DO
CALL getTotalHours() : Integer
        DO
                RETURN totalHours
        END DO
CALL getStudentName() : String
        DO
                RETURN studentName
        END DO



CALL setStudentName(String studentName)
        DO
                THIS studentName = studentName
        END DO

CALL display()
        IF (studentName != " ")
```

```
        DO
            DISPLAY(courseName, InstructorName, totalhours)
        END DO
    ELSE
        DO
            DISPLAY(studentName)
        END DO
```

Professional class

```
CALL getCourseFee() : Integer
        DO
                RETURN courseFee
        END DO
CALL getEnrollDate() : String
        DO
                RETURN enrollDate
        END DO
CALL getRoomNo() : String
        DO
                RETURN roomNo
        END DO
CALL getDailyHour() : Integer
        DO
                RETURN dailyHour
        END DO
CALL getDownPayment() : Integer
        DO
                RETURN downPayment
        END DO
CALL getStarted() : Boolean
        DO
                RETURN started
        END DO
CALL getCompleted() : Boolean
        DO
                RETURN completed
        END DO
```

```
CALL setCourseFee(courseFee)
        DO
                THIS.courseFee =  courseFee
        END D
CALL setDailyHour(dailyHour)
        DO
                THIS.dailyHour = dailyHour
        END DO



setEnroll(roomNo, dowanPayment,  enrollDate, studentName)
        IF (started = true)
                DO
                        Display("Course started" + instructorName + roomNo)
                END DO
        ELSE
                DO
                        THIS.studentName = studentName
                        THIS.enrollDate = enrollDate
                        THIS.studentName = StudentName
                        THIS.downPayment = downPayment
                        started = false
                END DO
```

```
CALL setCompleted()
        IF (completed = true)
                DO
                        DISPLAY("course is already completed")
                END DO
        ELSE
                DO
                        setStudentName(" ")
                        THIS.completed = completed
                        THIS.downPayment = 0
                        THIS.enrollDate = " "
                        THIS.roomNo = " "
                        THIS.started = false
                END DO


CALL getCourse()
        DO
                DISPLAY(CourseName, InstructorName, CourseName)
        END DO

CALL display()
        DO
                SUPER.Display()
                IF (started = true)
                        DO
                         DISPLAY("studentName, downPayment, enrollDate, competed")
                        END DO
        END DO
```

Certification class

```
CALL getStarted()   :  Boolean
        DO
                RETURN started
        END DO
CALL getCourseFee()   : Integer
        DO
                RETURN courseFee
        END DO
CALL getExamDate()   : String
        DO
                RETURN examDate
        END DO
CALL getStartDate()  :  String
        DO
                RETURN startDate
        END DO
CALL getExamCenter()   :  String
        DO
                RETURN examCenter
        END DO
CALL getCertificateAwardedBy()   : String
        DO
                RETURN certificateAwardedBy
        END DO
CALL getValidTill()  : String
        DO
                RETURN validTill
        END DO
```

```
CALL setCourseFee(courseFee)
        IF (started = true)
                DO
                        DISPLAY("The courese has already started")
                END DO
        ELSE
                DO
                        THIS.started = true
                END DO

CALL setEnroll(studentName, startDate, examDate, examCenter)
        IF (started = true)
                DO
                        DISPLAY("The courese has already started from" + startDate)
                END DO
        ELSE
                DO
                        THIS.studentName = studentName
                        THIS.startDate = startDate
                        THIS.examDate = examDate
                        THIS.examCenter = examCenter
                        started = true
                END DO

CALL display()
        DO
                SUPER.Display()
                IF (started = true)
                        DISPLAY(studentName, startDate, examDate, examCenter,
                        certificationAwardedBy, validTill")
        END DO
```

Method Description

Course (Super Class) :

There are 7 methods in total for course class which includes:

Course

It is the constructor method for class Course. It takes course name, instructor's name and total hours to complete course as parameter and assigns them. It initializes students name to “ “.

getTotalHours

Accessor method which returns Total Hours variable in integer form.

getCoursename

Accessor method which returns Course name in string form.

getInstructorName

Accessor method which returns Instructor name in string form.

getStudentName

Accessor method which returns student name string form.

setStudentName

Setter method to overwrite the student name

display

The display method displays the course name, total hours and instructor's name. If the student's name is not an empty string, it will display student's name too.

Professional (secondary class):

Professional

Constructor method for Professional class which takes course name, instructor's name, course fee, total hours and course hours per day as parameter. Constructor from super class are also called from the super class in this method with three parameters and remaining parameters are assigned to global variables.

getCourseFee

Accessor method which returns Course fee in integer form.

getDailyHour

Accessor method which returns hours per day in integer form.

getDownPayment

Accessor method which returns down payment in integer form.

getStarted

Accessor method which returns Course status in Boolean form.

getCompleted

Accessor method which returns Course completion status in Boolean form.

getEnrollDate

Accessor method which returns enroll date in string form.

getRoomNo

Accessor method which returns room detail in string form.

setCourseFee

Setter method to set the global course fee variable by taking new course fee as parameter

setDailyHour

Setter method to set the global daily hours for course variable by taking new daily hour data as parameter

setEnroll

Setter method to enroll a student to a course by taking new student's name, enroll date, amount the student paid at the time of enrollment, room number assigned for particular class if the course is not started. Else this method will display a message including the instructor's name and room number indicating the class had already started.

setCompleted

Setter method to set completed as true if it is false. It also set the student's name is called with " " as a parameter to the setStudentName method in super class, the room no and enroll Date are set to " ", down payment is set to 0, the started status is set to false. Else it will display that the course has already been completed.

getCourse

Method to print the course name, instructor's name and course fee by calling through super class

display

The display method from the super class displays the course name, total hours and instructor's name. If the student's name is not an empty string, it will display student's name too. The overridden display method from the professional class will display completed Status, enroll Date, down Payment and student's name if course has already started.

Certification (secondary class):

Certification

Constructor method for Certification class which takes course name, instructor's name, total hours to complete course, course fee, certificate awarding body and valid till as parameter. Constructor from super class are also called from the super class in this method with three parameters and remaining parameters are assigned to their corresponding global variables. The variables for start date, exam date, exam center are set to “ “ and course started to false.

getExamDate

Accessor method which returns the exam date in string form

getStartDate

Accessor method which returns start date in string form

getExamCenter

Accessor method which returns exam center in string form

getValidTill

Accessor method which returns validity period in string form

getCourseFee

Accessor method which returns Course fee in integer form.

getStarted

Accessor method which returns started information in Boolean value

getCertificateAwardedBy

Accessor method which returns certificate awarding body in string form

setCourseFee

The method to set new course fee if the course is not started. Else it will display that the fee cannot be changed.

setEnroll

The method to enroll a new student to a course. If the course hasn’t started s, the student's name, start date, exam date, exam center is taken as parameters and assigned to their corresponding global variables. Else displays that the course has already been started.

display

The display method from the super class displays the course name, total hours and instructor's name. If the student's name is not an empty string, it will display student's name too. The overridden display method from the certification class will display start date, student's name, exam date, exam center name of the certificate awarding body and certification validity duration if course has already started.