Islington College

Programming

CS4001NI

Coursework 1

Submitted By:

Rajat Shrestha

17030954

Group: L1C2

Date: 19th Jan 2018

Submitted To:

Mr. Bhim Bahadur Sunar

Lecturer, IT

Programming

Proposal:

This proposal is written to address the first individual coursework given out on Programming Module. The project involves detailed documentation and development of a java program to demonstrate how java can be used to handle problems using object oriented programming method. The coursework was handed out at 8th week and was to be submitted in 12th week.

Purpose:

The purpose of this project is to develop a code in object oriented manner by using classes which are linked to each other in a hierarchy. The program should be aided by class diagram, pseudocode and research for a better result.

Mission:

The Task is to simply to create 3 classes: course, professional and certification in which the latter two are extended from the super class course. These classes must have all the constructor, accessor and setter methods coded according to the question.

Table of Contents:

Table of figures:

Table of tables:

1. Introduction:

The main objective of this project is to develop codes on three classes: course, professional and certification. The classes professional and certification are inherits attributes from course class so they have the common attributes: Course name, Instructor name, total hours and student name which is initialized to “ “. Simply put Inheritance is ability of a subclass to take general properties of super class (Poo, et al., 2007). The other classes have their own attributes. All the classes consist of accessor methods for each variable and setter method for some specified variables. The display method on the super class is called on both the sub classes with the same signature to display details on each class. There are also specific methods in professional and certification class as instructed in the paper. To develop the program a class diagram is prepared to simplify the process.

This project documents the development of the program which involves working with java so for understand this project one must first understand these essential keywords included in this project. The keywords are Class, Method, Variables, Super, Datatypes, and Various other java keywords.

While working on the project these are tools were used:

1. **Notepad**
   Text editor software for editing codes and texts

2. **Microsoft Word 2016**
   Word processing tool to create the report

3. **BlueJ**
   IDE for java to prepare the code

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. Gathering information about the project and keywords used in it.
2. Constructing some simple models to aid the development of the project (like class diagram and pseudocode).
3. Coding the program.
4. Testing 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). UML stands for Unified Modeling Language, it is like a graphical language and was created to model various types of codes to simplify them (Jacobson, et al., 2000). The use of UML class diagram will make the process of writing the code very easier so using it is highly recommended. The simplified version of the overall class diagram is as following:

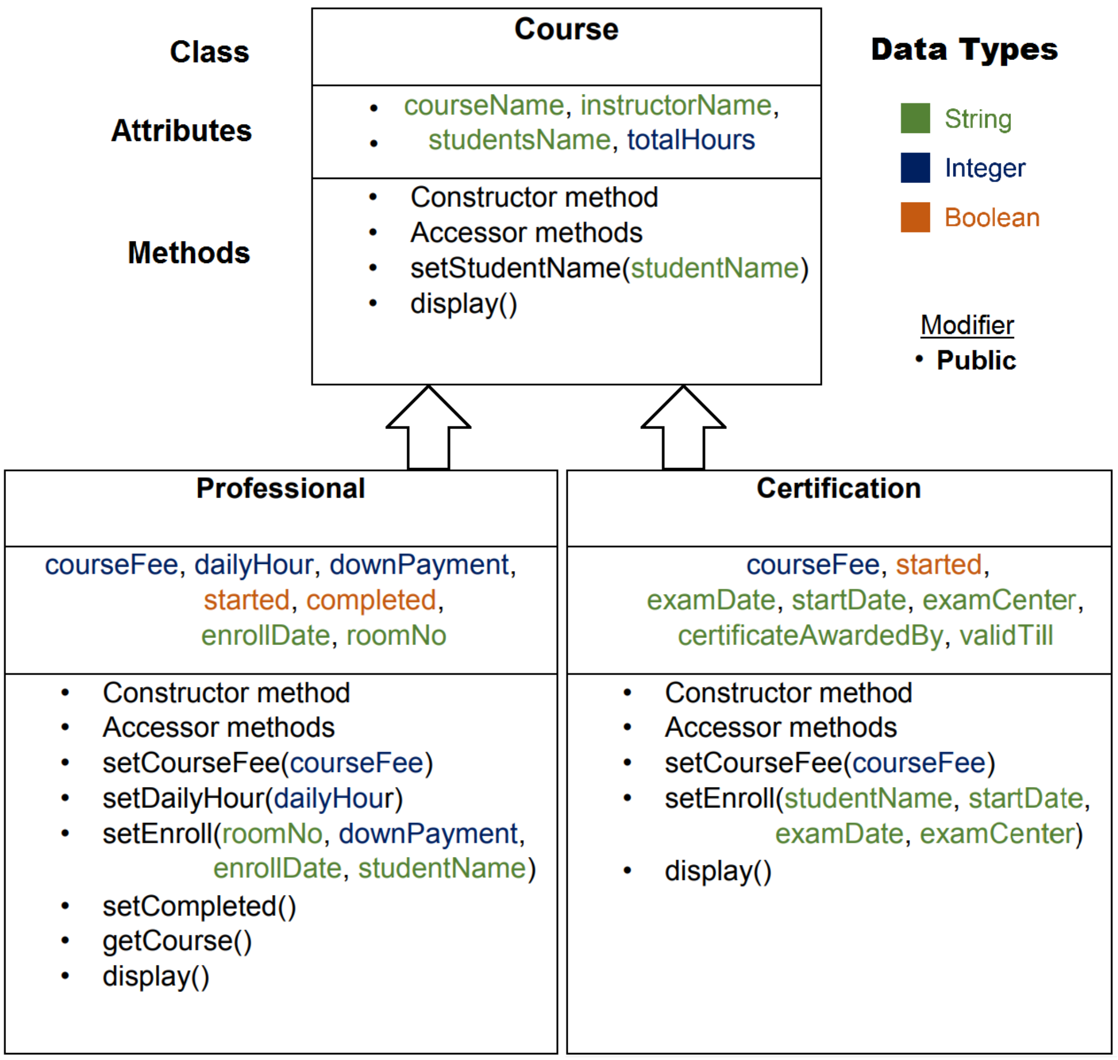

Class
Attributes
Methods
Course
courseName, instructorName, studentsName, totalHours
Constructor method
Accessor methods
setStudentName(studentName)
display()
Data Types
String
Integer
Boolean
Modifier
Public
Professional
courseFee, dailyHour, downPayment, started, completed, enrollDate, roomNo
Constructor method
Accessor methods
setCourseFee(courseFee)
setDailyHour(dailyHour)
setEnroll(roomNo, downPayment, enrollDate, studentName)
setCompleted()
getCourse()
display()
Certification
courseFee, started, examDate, startDate, examCenter, certificateAwardedBy, validTill
Constructor method
Accessor methods
setCourseFee(courseFee)
setEnroll(studentName, startDate, examDate, examCenter)
display()

Figure 1: Overall Class Diagram

The more detailed class diagram is as follows which includes all the information in detail

Table 1: Class diagram for Course which is super class

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

Table 2: Class diagram for Professional class

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 : Integer) : void + setDailyHour(dailyHour : Integer) : void + setEnroll(roomNo : String, downPayment : Integer, enrollDate : String, studentName: String) : void + setCompleted() : void + getCourse() : void + display() : void

Table 3: Class diagram for Certification class

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

By using the class diagram, we can easily model the code and how it should be written. The primary objective of a class diagram is to model the code so it can be used to find out how a code is written and also how a code can be written if it is not.

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). So, to develop the final code, pseudocodes can help a lot in the process. The pseudocodes for the program are as follows:

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
```

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. As there are there classes their methods are described below:

4.1. 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.

4.2. 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.

4.3. 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.

5. Tests:

To ensure that the if program works we need to ensure it by testing it. For doing so several tests were done to find out problems and solve them. Testing ensures the functionality of the code so it is an important part of any development project. There are various different types of testing like black box, grey box and white box testing (Khan & Khan, 2012). So, for to know if the program works following tests were done.

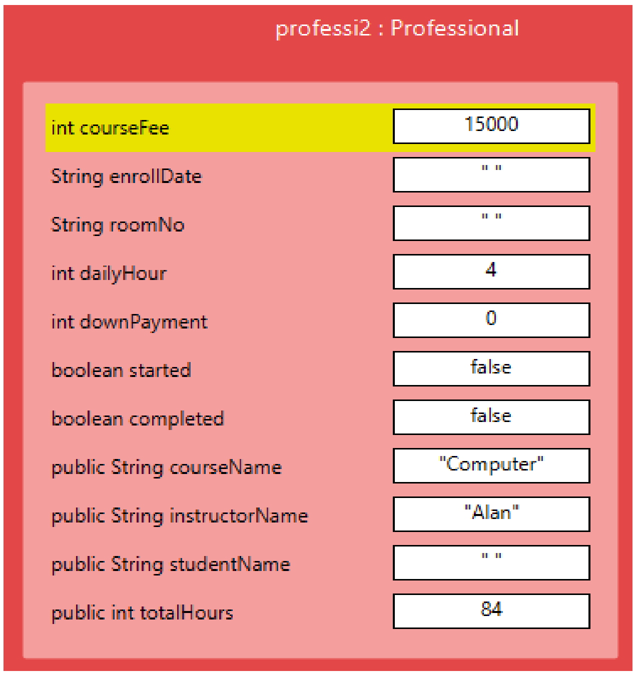
5.1. Test 1:

Table 4: Test 1

Objective	To Inspect an object of Professional Class, enroll student for that particular course, and re-inspect the object.

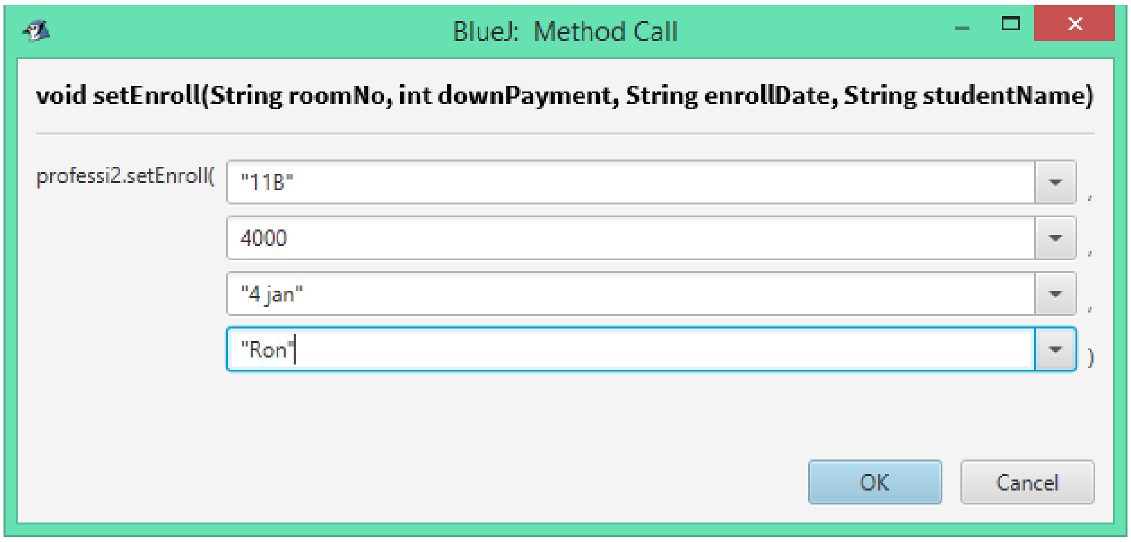
Action	Constructor is called;
	CourseName = “Computer”
	instructorName = “Alan”
	totalHours = 84
	dailyHour = 4
	courseFee = 15000
	The object is inspected
	setEnroll is called
	roomNo = “11B”
	downPayment = 4000
	enrollDate = “4 jan”
	studentName = “Ron”
	The object is reinspected

Expected Result	The input value to be assigned to the corresponding variables of the object.
Actual Result	The input value was assigned to the object as shown by the inspection.
Conclusion	Test successful


professi2 : Professional

int courseFee	15000
String enrollDate	" "
String roomNo	" "
int dailyHour	4
int downPayment	0
boolean started	false
boolean completed	false
public String courseName	"Computer"
public String instructorName	"Alan"
public String studentName	" "
public int totalHours	84

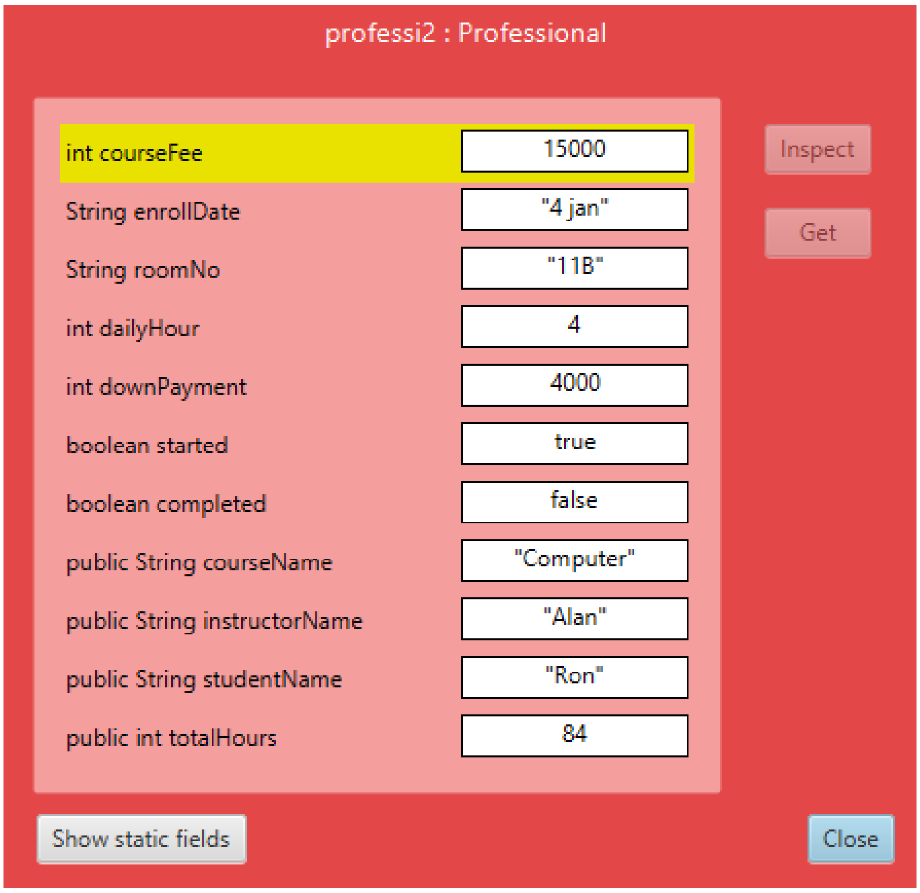
Figure 2: Inspecting object


BlueJ: Method Call

void setEnroll(String roomNo, int downPayment, String enrollDate, String studentName)

```
professi2.setEnroll( "11B" ,
                     4000 ,
                     "4 jan" ,
                     "Ron" )
```

Figure 3: enrolling a student


professi2 : Professional

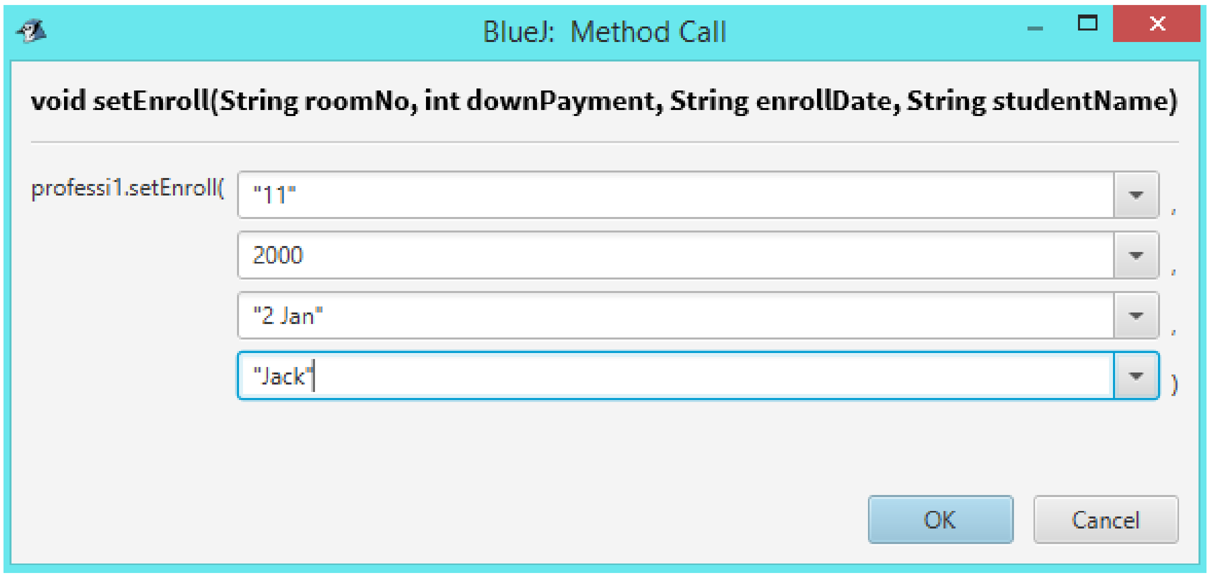
int courseFee	15000
String enrollDate	"4 jan"
String roomNo	"11B"
int dailyHour	4
int downPayment	4000
boolean started	true
boolean completed	false
public String courseName	"Computer"
public String instructorName	"Alan"
public String studentName	"Ron"
public int totalHours	84

Figure 4: Reinspection of object

5.2. Test 2:

Table 5: Test 2

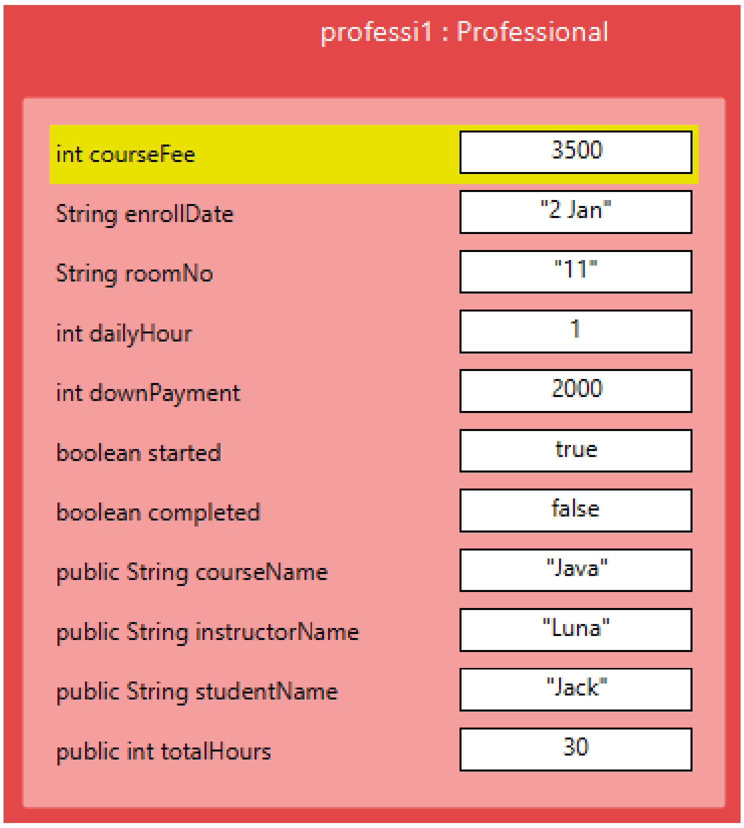
Objective	To Inspect Professional, change the status of course to complete, and re-inspect the Professional Class
Action	A student is enrolled The object is inspected called setCompleted The object is reinspected
Expected Result	The values to be reset for room
Actual Result	The values of enrollDate, studentName, downPayment and roomNo was reset.
Conclusion	Test successful


BlueJ: Method Call

void setEnroll(String roomNo, int downPayment, String enrollDate, String studentName)

```
professi1.setEnroll( "11" ,
                     2000 ,
                     "2 Jan" ,
                     "Jack" )
```

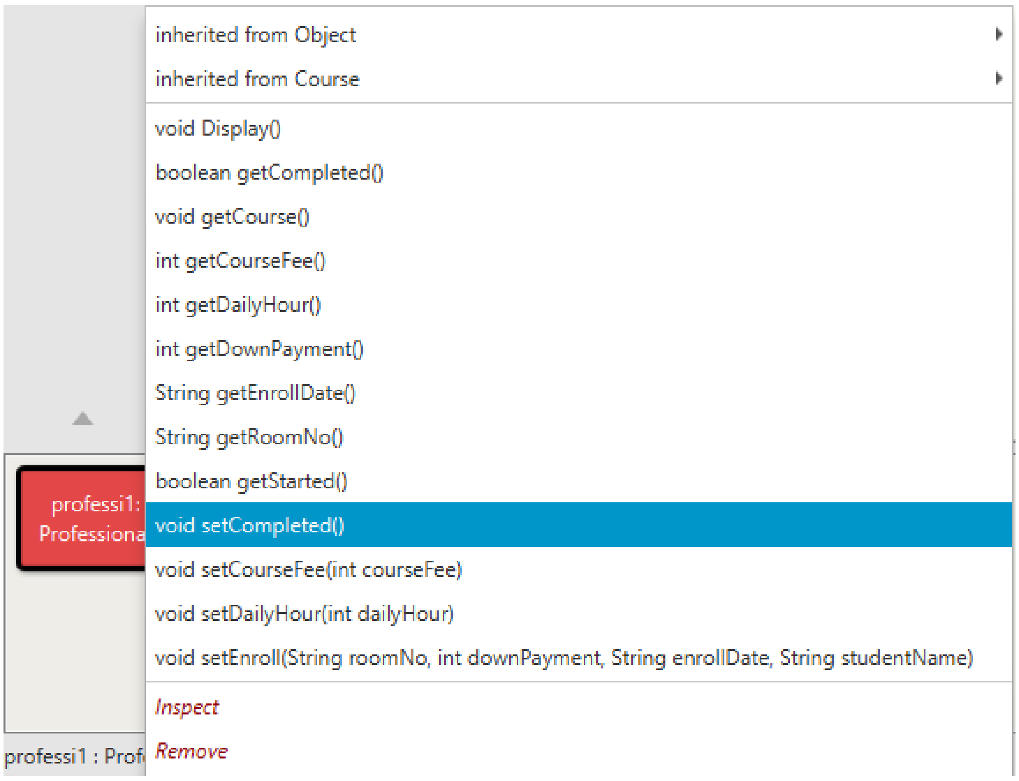
Figure 5: enrolling a student



professi1 : Professional

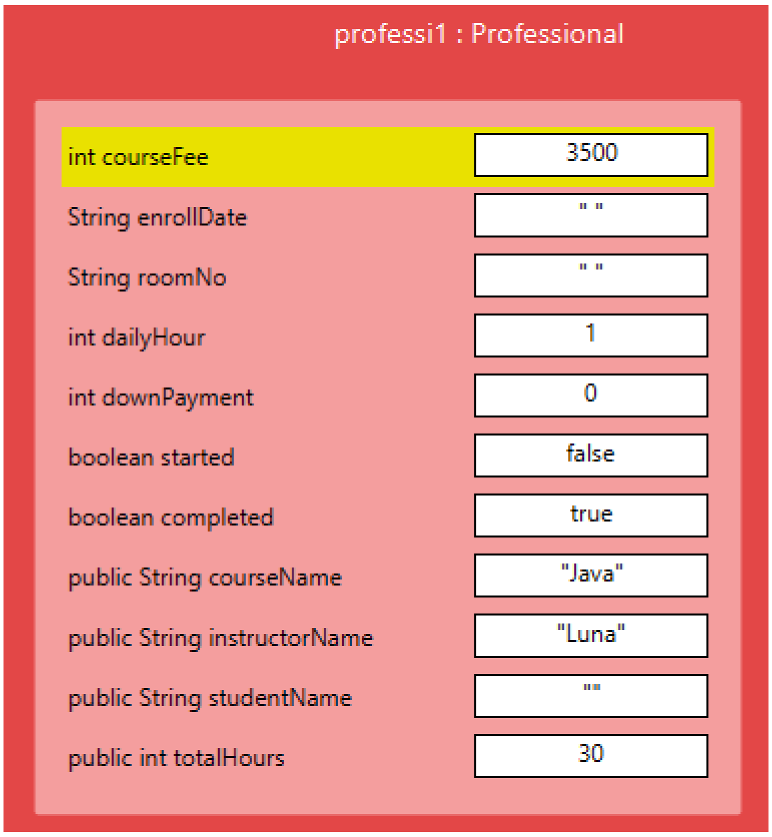
int courseFee	3500
String enrollDate	"2 Jan"
String roomNo	"11"
int dailyHour	1
int downPayment	2000
boolean started	true
boolean completed	false
public String courseName	"Java"
public String instructorName	"Luna"
public String studentName	"Jack"
public int totalHours	30

Figure 6: Inspecting the object



```
inherited from Object
inherited from Course
void Display()
boolean getCompleted()
void getCourse()
int getCourseFee()
int getDailyHour()
int getDownPayment()
String getEnrollDate()
String getRoomNo()
boolean getStarted()
void setCompleted()
void setCourseFee(int courseFee)
void setDailyHour(int dailyHour)
void setEnroll(String roomNo, int downPayment, String enrollDate, String studentName)
Inspect
Remove
```

Figure 7: Setting the object as completed



professi1 : Professional

int courseFee	3500
String enrollDate	" "
String roomNo	" "
int dailyHour	1
int downPayment	0
boolean started	false
boolean completed	true
public String courseName	"Java"
public String instructorName	"Luna"
public String studentName	""
public int totalHours	30

Figure 8: Reinspection of the object

5.3. Test 3:

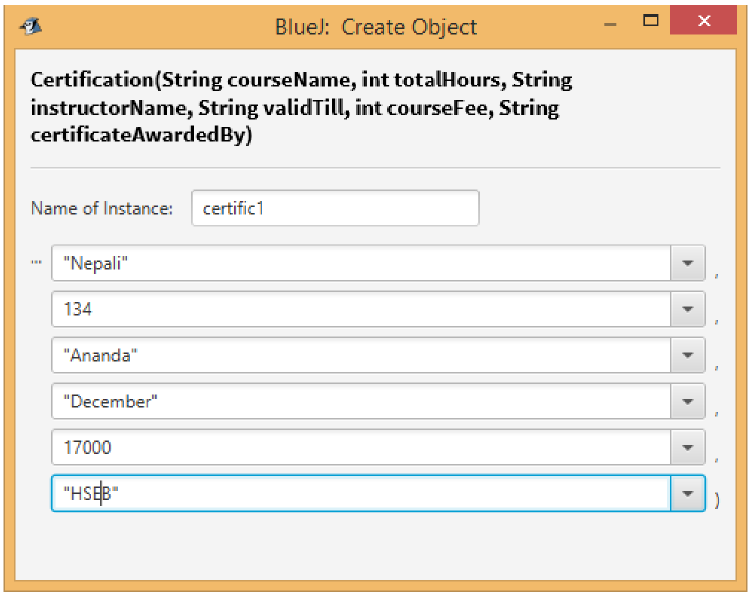
To Inspect Certification, enroll student, and re-inspect.

Table 6: Test 3

Objective	To Inspect an object of Certification Class, enroll student for that particular course, and re-inspect the object.

Action	Constructor is called;
	CourseName = "Nepali"
	instructorName = "Ananda"
	totalHours = 134
	courseFee = 17000
	CertificateAwardedBy = "HSEB"
	The object is inspected
	setEnroll is called
	studentName = "Ronny"
	startDate = "Feb 23"
	examDate = "March 23"
	examCenter = "Oxford"
	The object is reinspected

Expected Result	The input value to be assigned to the corresponding variables of the object.
Actual Result	The input value was assigned to the object as shown by the inspection.
Conclusion	Test successful

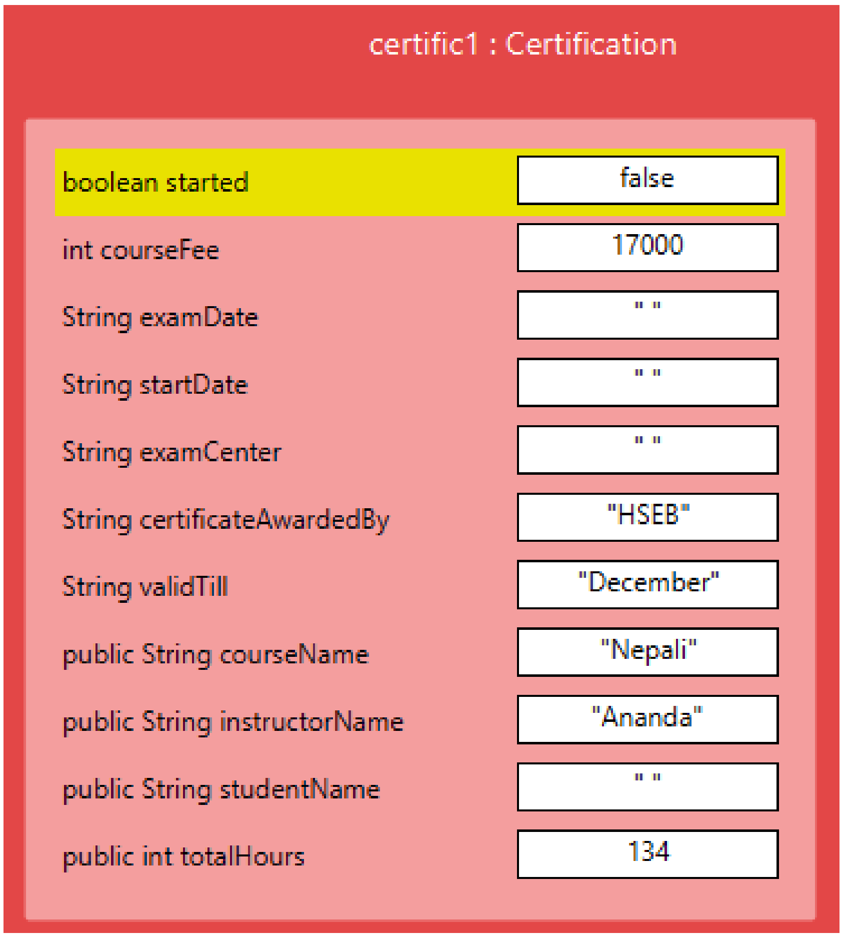


Certification(String courseName, int totalHours, String instructorName, String validTill, int courseFee, String certificateAwardedBy)

Name of Instance: certific1

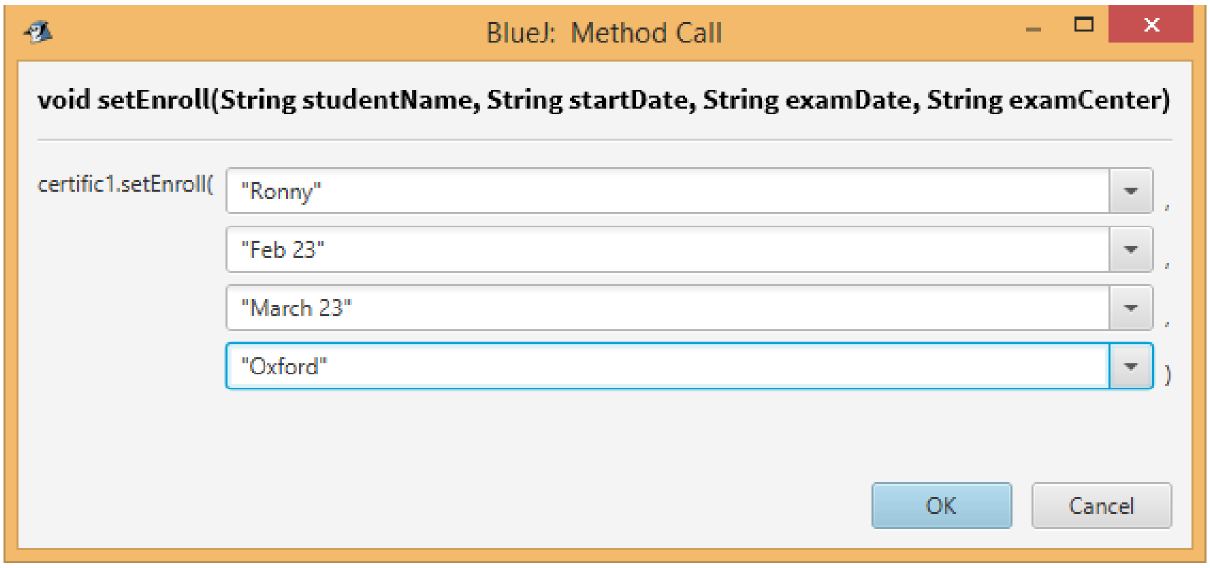
"Nepali" ,
134 ,
"Ananda" ,
"December" ,
17000 ,
"HSEB")

Figure 9: Creating a object


certific1 : Certification

boolean started	false
int courseFee	17000
String examDate	" "
String startDate	" "
String examCenter	" "
String certificateAwardedBy	"HSEB"
String validTill	"December"
public String courseName	"Nepali"
public String instructorName	"Ananda"
public String studentName	" "
public int totalHours	134

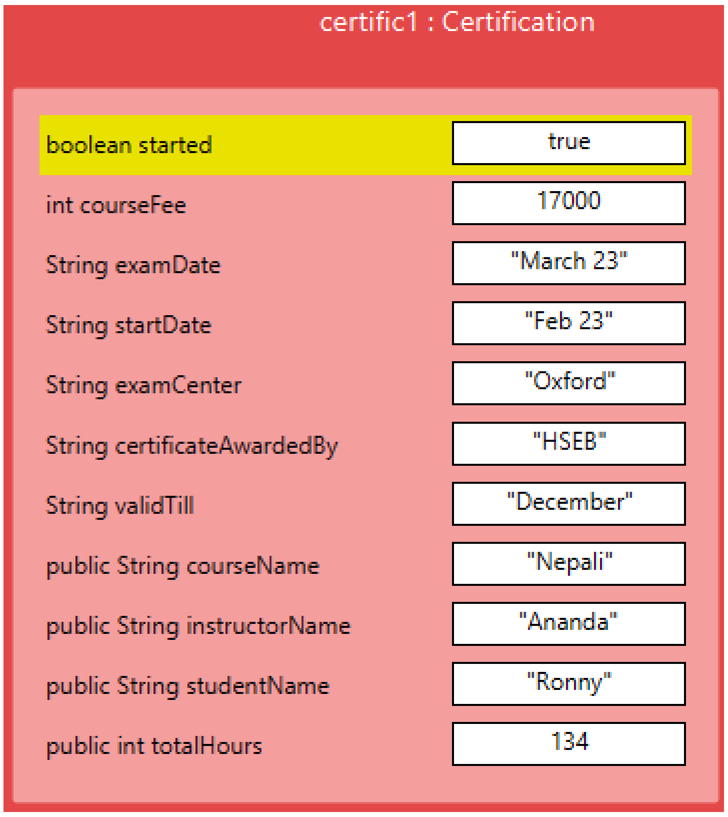
Figure 10: Inspecting the object


BlueJ: Method Call

void setEnroll(String studentName, String startDate, String examDate, String examCenter)

```
certific1.setEnroll( "Ronny" ,
                     "Feb 23" ,
                     "March 23" ,
                     "Oxford" )
```

Figure 11: enrolling a student



certific1 : Certification

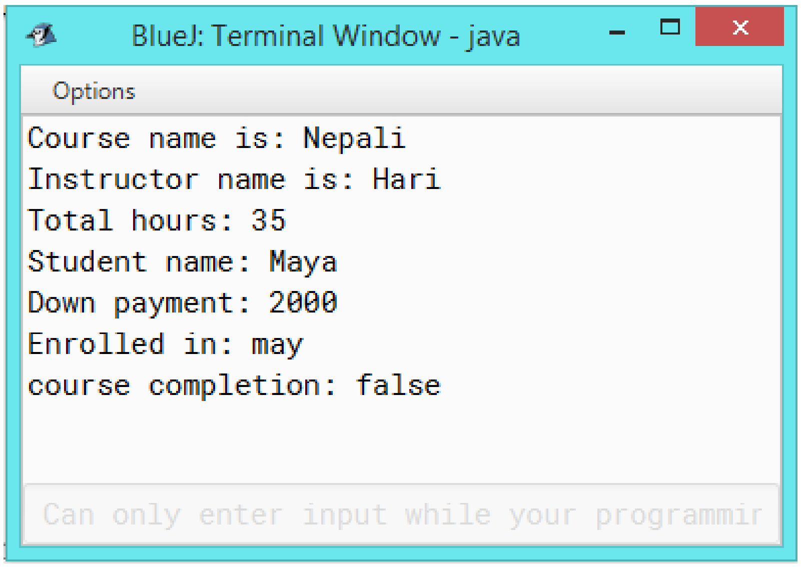
boolean started	true
int courseFee	17000
String examDate	"March 23"
String startDate	"Feb 23"
String examCenter	"Oxford"
String certificateAwardedBy	"HSEB"
String validTill	"December"
public String courseName	"Nepali"
public String instructorName	"Ananda"
public String studentName	"Ronny"
public int totalHours	134

Figure 12: Reinspection of the object

5.4. Test 4:

Table 7: Test 4

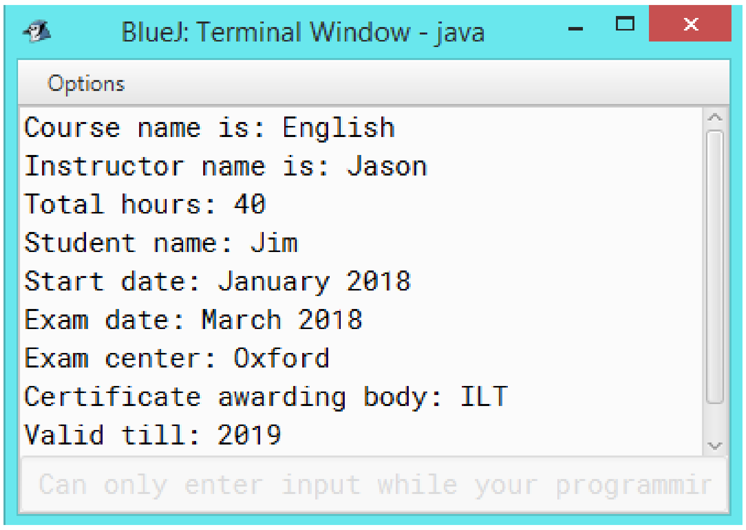
Objective	To display the detail of Professional and Certification Class
Action	Enrol a student in professional class To show output of display method of professional class Enrol a student in certification class To show output of display method of certification class
Expected Result	The display method should display available information
Actual Result	The display method displayed available information
Conclusion	Test successful


BlueJ: Terminal Window - java
Options

```
Course name is: Nepali
Instructor name is: Hari
Total hours: 35
Student name: Maya
Down payment: 2000
Enrolled in: may
course completion: false
```

Can only enter input while your programmir

Figure 14: display of professional class


BlueJ: Terminal Window - java
Options

```
Course name is: English
Instructor name is: Jason
Total hours: 40
Student name: Jim
Start date: January 2018
Exam date: March 2018
Exam center: Oxford
Certificate awarding body: ILT
Valid till: 2019
```

Can only enter input while your programmir

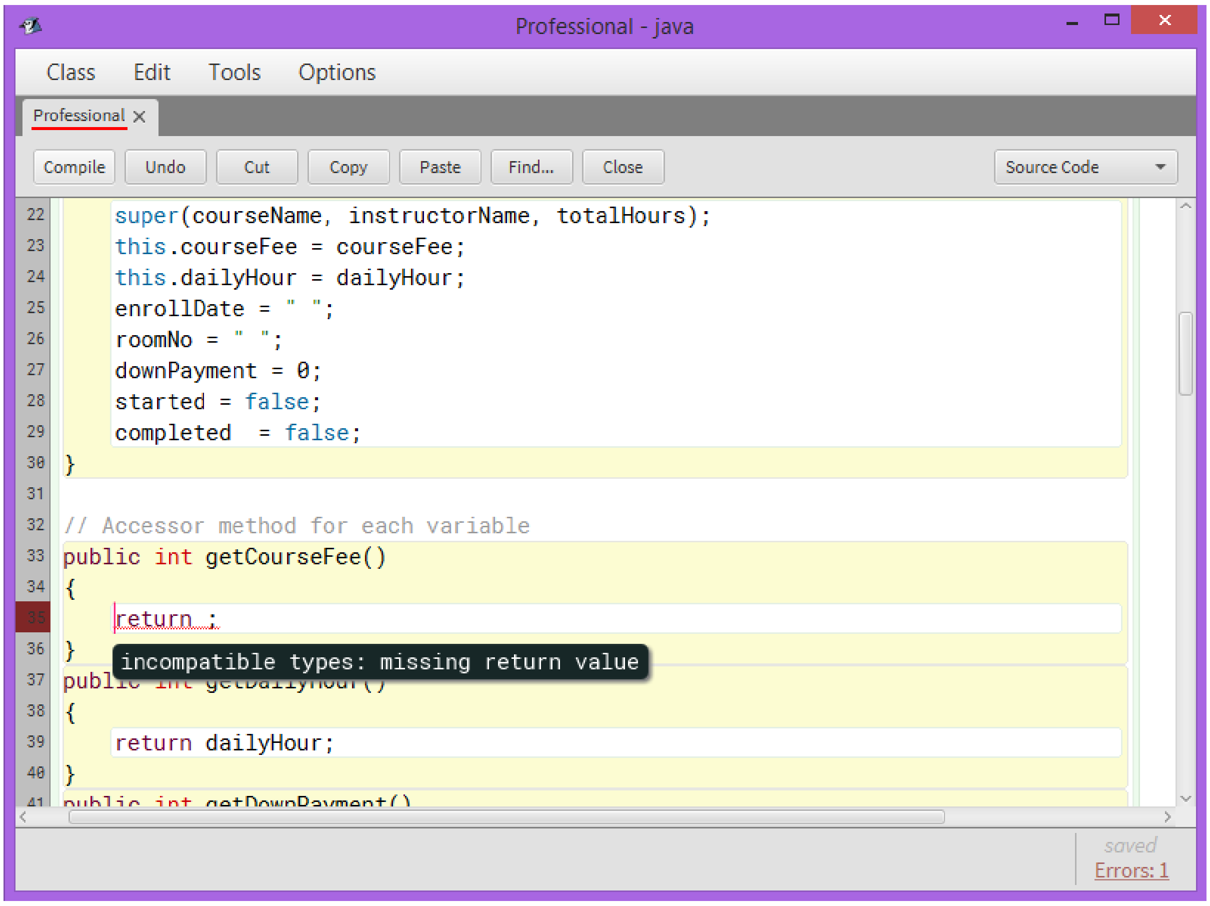
Figure 13: display of certification class

6. Error Detection:

Compiler error messages are created when the Java software code is run through the compiler. 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 following errors were found while compiling the code.

Incompatible types: Missing Return Value

You'll get the "missing return value" message when the return statement includes an incorrect type. It can be solved easily by putting the right return statement.



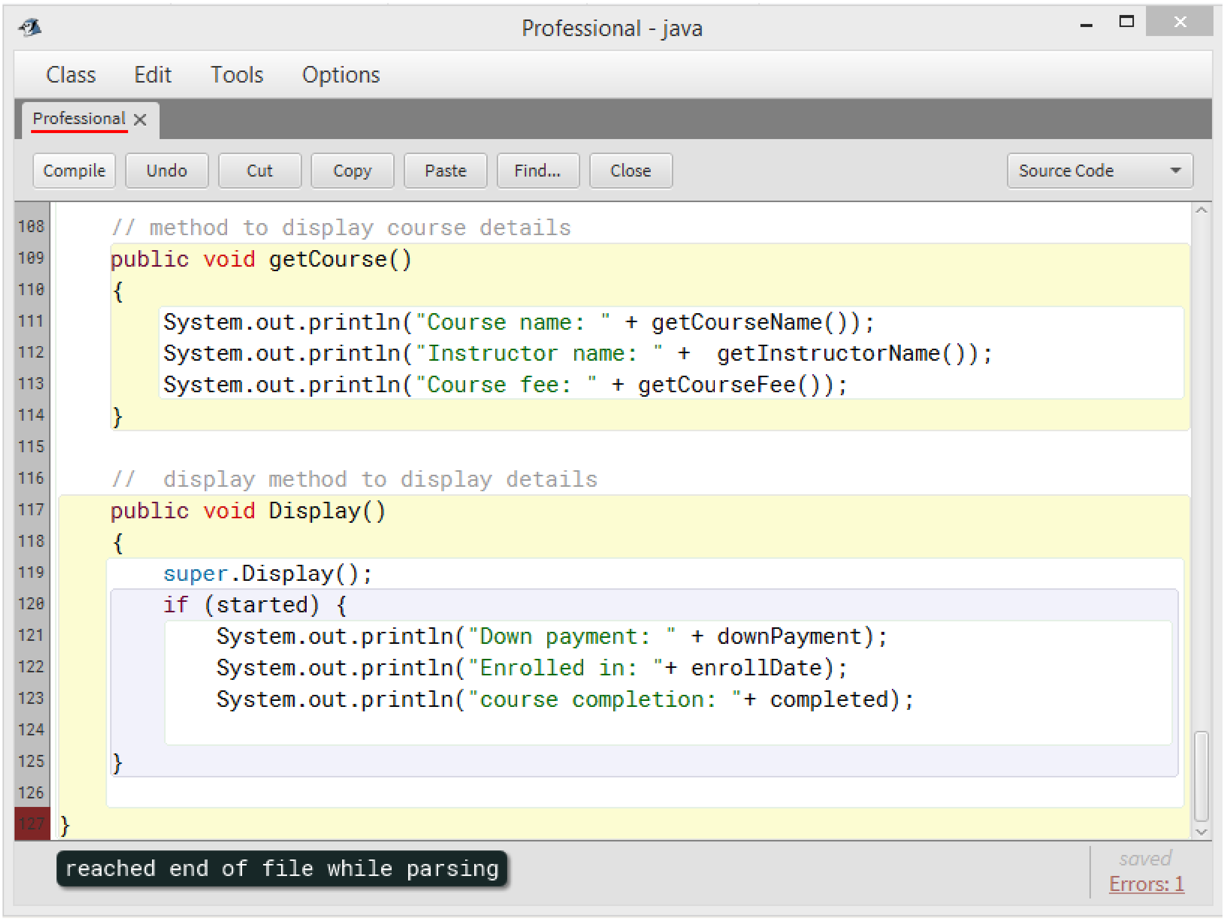
```
    super(courseName, instructorName, totalHours);
    this.courseFee = courseFee;
    this.dailyHour = dailyHour;
    enrollDate = " ";
    roomNo = " ";
    downPayment = 0;
    started = false;
    completed  = false;
}

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

```
incompatible types: missing return value
```

Figure 15: Error indicating missing return statement

Reached End of File While Parsing

This error message usually occurs in Java when the program is missing the closing curly brace “}”. This error indicates that the class or method is missing a curly brace and the compiler cannot determine which brace closes what. So if this error occurs it is very easy to find as the alignment of code indicates here the brace is missing.



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

    }

}
```

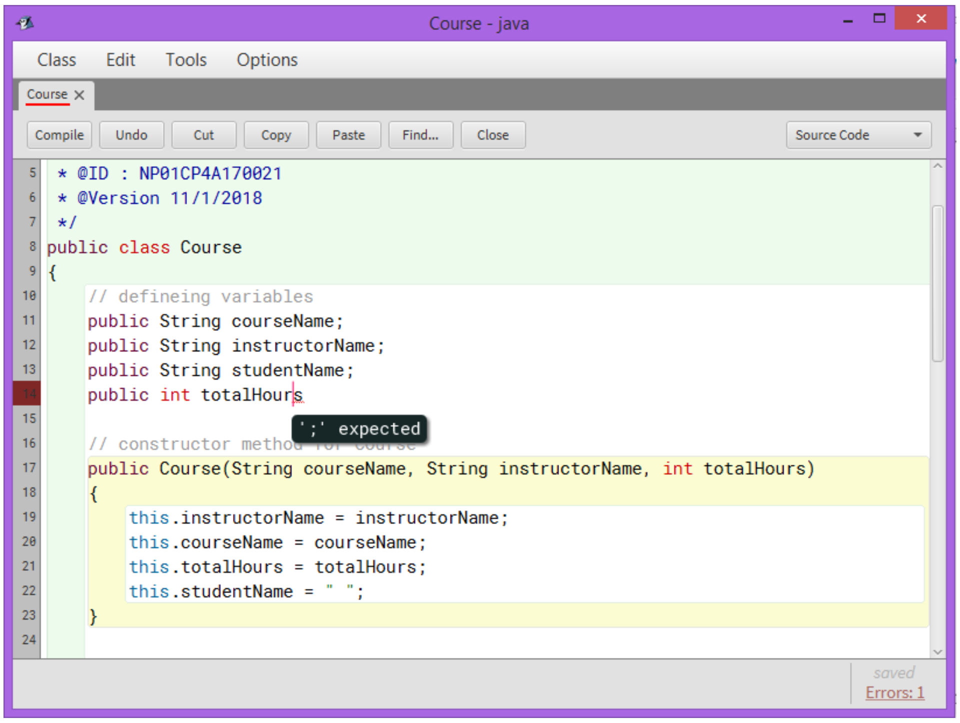
```
reached end of file while parsing
```

saved
Errors: 1

Figure 16: Error due to missing braces

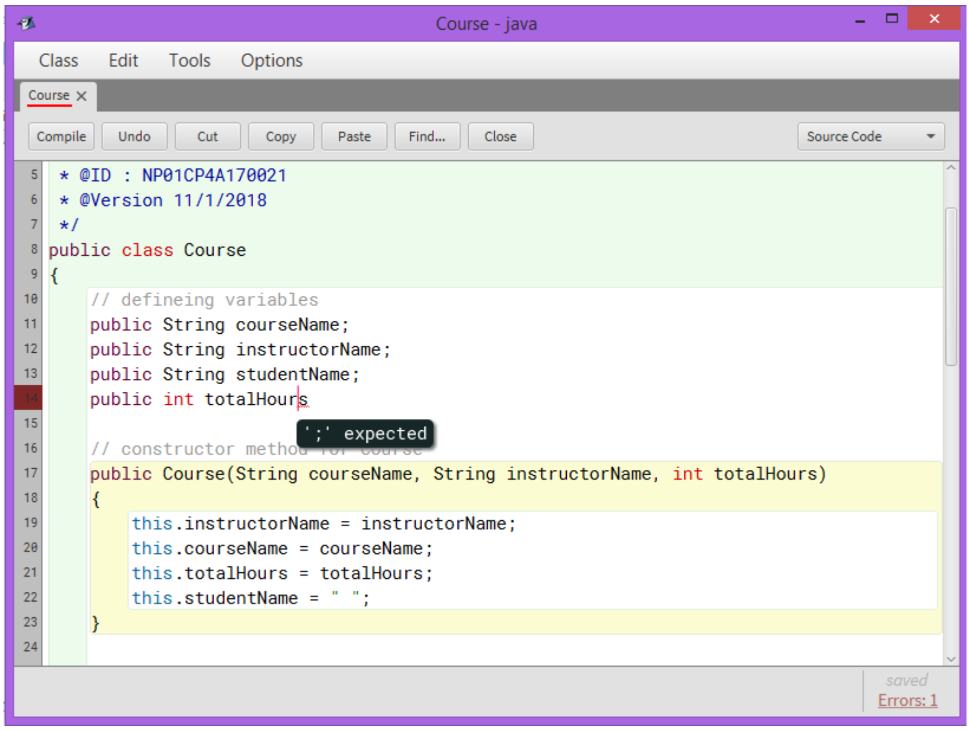
“;” Expected

This error message usually occurs in Java when the program is missing the closing semi-colon “;”. This error also occurred when a bracket was missing after declaring a method.



```
 * @ID : NP01CP4A170021
 * @Version 11/1/2018
 */
public class Course
{
    // defineing variables
    public String courseName;
    public String instructorName;
    public String studentName;
    public int totalHours
                       ';' expected
    // constructor method for course
    public Course(String courseName, String instructorName, int totalHours)
    {
        this.instructorName = instructorName;
        this.courseName = courseName;
        this.totalHours = totalHours;
        this.studentName = " ";
    }
```

Figure 17: Error due to missing semi colon



```
 * @ID : NP01CP4A170021
 * @Version 11/1/2018
 */
public class Course
{
    // defineing variables
    public String courseName;
    public String instructorName;
    public String studentName;
    public int totalHours
                       ';' expected
    // constructor method for course
    public Course(String courseName, String instructorName, int totalHours)
    {
        this.instructorName = instructorName;
        this.courseName = courseName;
        this.totalHours = totalHours;
        this.studentName = " ";
    }
```

Figure 18: Error due to missing braces

Java.lang.String

This error causes when the default keywords of java is used as a class or method. This causes the keywords to be reassigned and the problem might seem unsolvable. This might cause errors even though the code is perfectly normal and can only be solved by removing the conflicting source saved in the computer.

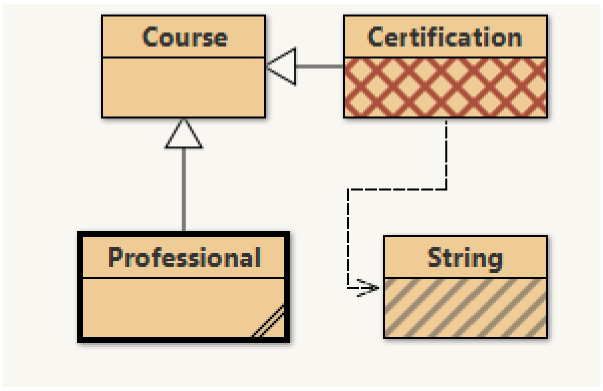
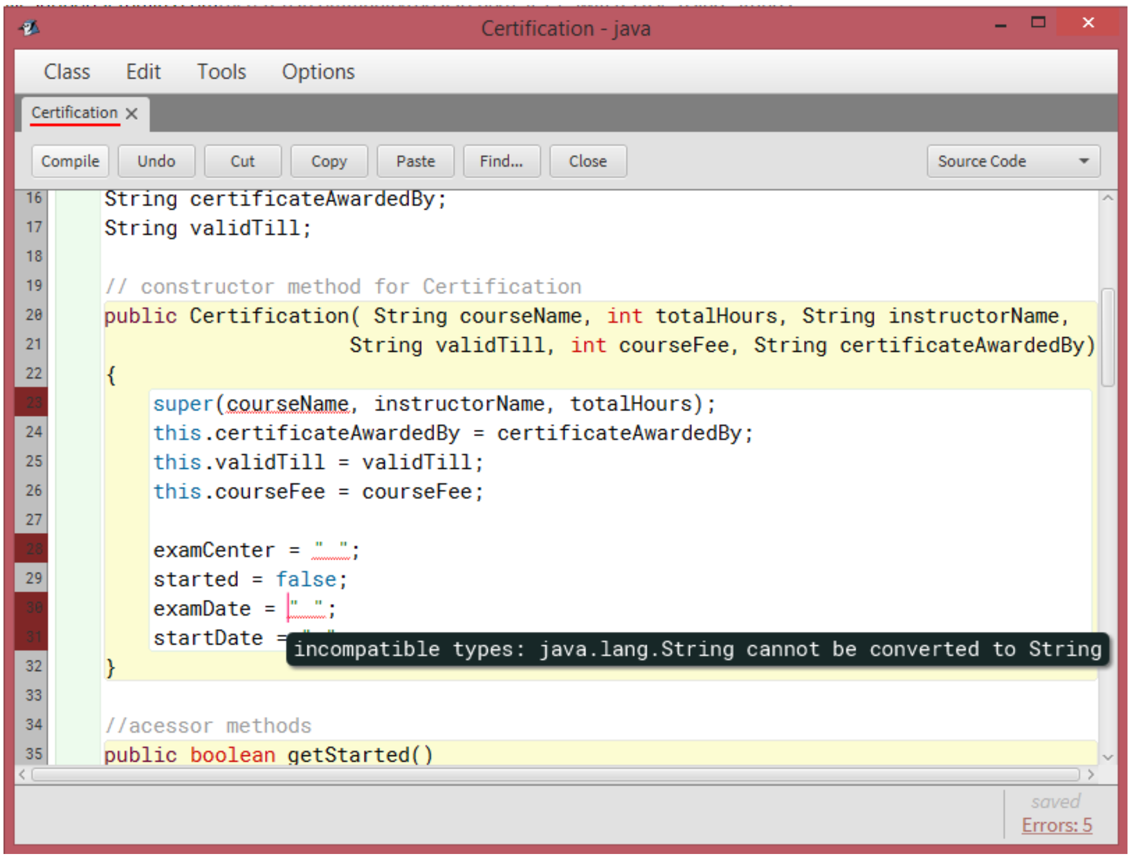


Figure 19: Declaring a class named String



```
    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 =
    }

    //acessor methods
    public boolean getStarted()
```

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

Figure 20: Multiple error caused due to the conflicting keyword

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 the hierarchical class system to make the codes in the super class to be reused in the further code. 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 code contained some errors caused by invalid syntax and missing variables 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 reliant. After knowing and correcting errors, the code was again tested but now as the code is completed the tasks assigned were done and presented in this report to ensure the functions of the program.

References

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

Jacobson, I., Booch, G. & Rumbaugh, J., 2000. *The Unified Modeling Language User Guide.* 1st ed. Boston: Addison Wesley Longman.

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

Khan, F. & Khan, M. . E., 2012. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. *(IJACSA) International Journal of Advanced Computer Science and Applications,* 3(6), p. 15.

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

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

Reed, P. R., 2002. *Developing Applications with Java and UML.* 1st ed. Boston: Addison-Wesley Professional.

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].

Program:

```
/**
 * 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);
        }
    }
}
```