CS4001NI Programming

30% Individual Coursework - 2

2023-24 Sprin

Student Name: Sikum Hangma Madi

London Met ID: 22085627

College ID: NP01CP4S230077

Group: C20

Assignment Due Date: Friday, August 4, 2023

Assignment Submission Date: Friday, August 11, 2023

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

Table of Contents

Introduction

Welcome to my Java coursework. The module Programming has assigned me this coursework as second part to introduce us to the basics of GUI with the intention of using Java's awt event approach to provide solutions to the difficulties encountered in the real world. This coursework's objective is to instruct us in the basic concepts and apabilities of Java through practical coursework. I learned a lot from working on this coursework, including a strong foundation in Java programming, as well as Object-Oriented Principles like inheritance, polymorphism, encapsulation, and abstraction, along with how to control structures, data types, behaviours, and many other things. In addition, we discussed about working with collections, handling variations, and file input and output.

Java is powerful, popular, and distinguished for its accessibility, independence from one platform, and simplicity. It has offered programmers an array of applications, making it a great choice for creating various apps, including desktop software, web applications, and even mobile applications. It was developed by Sun Microsystems, which is now known as Oracle Corporation, and launched in 1995. (Hortsmann, 2007) Since then, it has grown significantly in popularity due to the fact that it can be used for creating a variety of applications. Independent of platform is one of Java's most significant advantages. (Selawsky, 2023)Java is portable thanks to its "write once, run anywhere" characteristic, which also allows developers to create an array of apps.

The purpose of this assignment is to develop and implement a student management system using object-oriented programming ideas. The system is built around a core class called 'Student,' which serves as the foundation for other student categories. This includes the subclasses 'Regular' and 'Dropout,' each with its own set of characteristics and behaviours. Furthermore, the 'StudentGUI' class generates a user-friendly interface with data input frames, allowing for the efficient handling of both regular and dropout student information. This project highlights the use of

object-oriented ideas in the development of a complete and interactive student management solution.

Blue J

A software program named BlueJ provides a single environment for Java programming by combining multiple development skills. It was developed specifically for using Java to educate and research Object-Oriented Programmin(OOP). It provides a waelcoming and engaging environment. (Herbit, 2008) Beginners will find it easier to understand the basic idea of program concepts because to its simplified interface and graphical representation of objects and classes. (https://www.bluej.org/)

The Object-centric approach of BlueJ is one of its main features. Users can interact directly with things, making it easier to understand how they behave and interact with other things in a program. In addition, BlueJ allows object investigation while the program is running, which helps to understand more clearly.

Microsoft Word

Microsoft created the well-known word processing program called Microsoft Word. A full set of features and tools are available in Word for creating, editing, formatting, and sharing documents. It has a user-friendly interface and an array of choices for modifying the appearance and layout of the text, graphics, and other document elements. (https://www.microsoft.com/en-us/microsoft-365/word)

Since it provides a variety of capabilities and features and is frequently utilized in professional, educational, and personal circumstances, Microsoft Word is a capable word processing program.

Draw.io

Draw.io is an user-friendly online diagramming tool that allows users to generate a variety of diagrams, charts, and flowcharts. Its user-friendly interface enables simple drag-and-drop capability, making it appropriate for both technical and non-technical users. Draw.io has a large library of forms, icons, and connectors for creating mind maps, organisational charts, and network diagrams. The collaborative features of the application enable real-time teamwork and diagram sharing, making it an invaluable resource for visual communication and documentation. It helped me a lot for drawing class diagram.

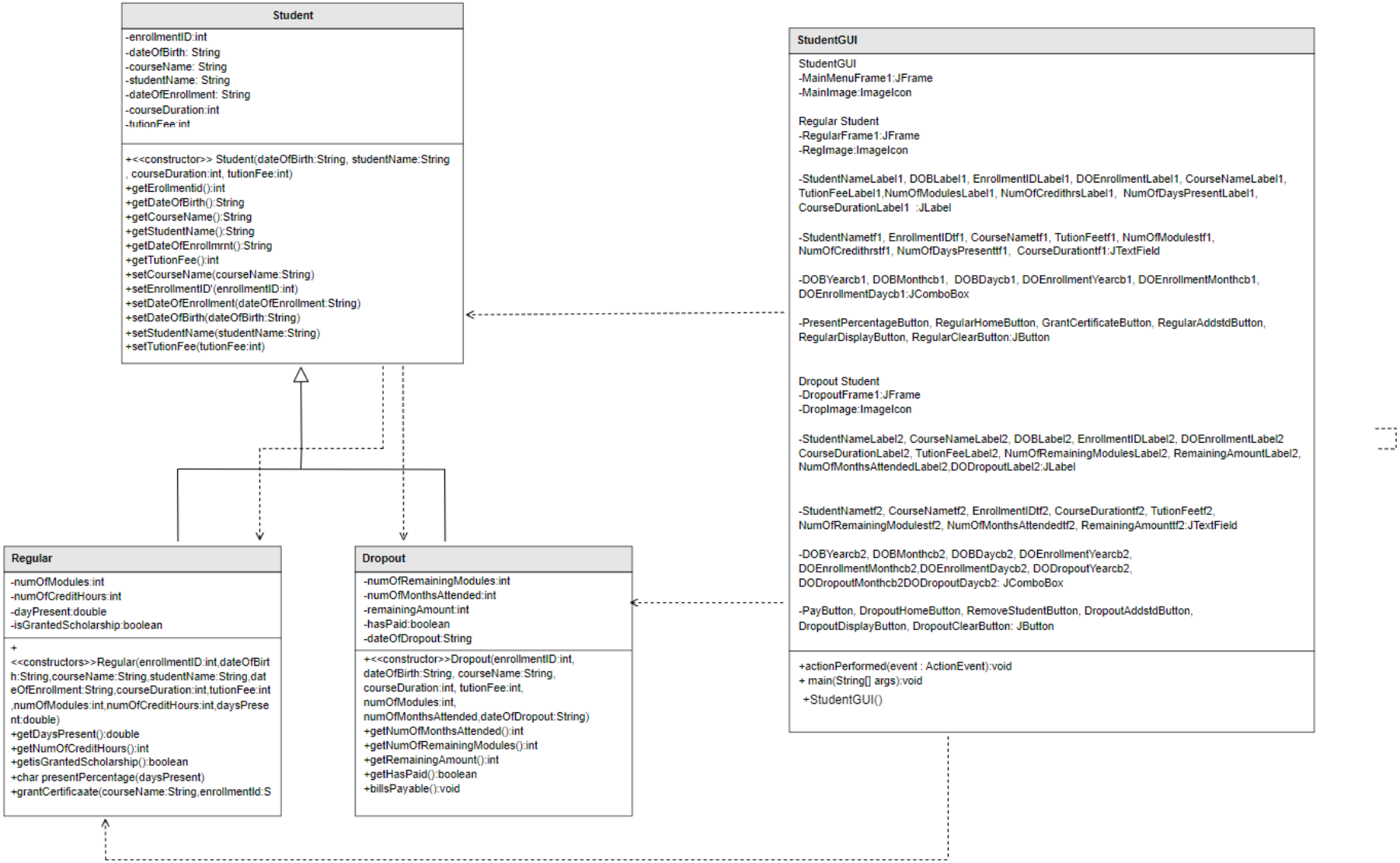
Moqups

Moqups is an advanced online wireframe that allows designers and teams to generate interactive prototypes and moqups. Moqups' user-friendly interface enables users to quickly create user interfaces, web pages, and app screens by mixing elements such as buttons, text boxes, and photos. The interactive capabilities of the tool enable clickable prototypes, which improves the capacity to replicate user interactions and workflows. Moqups encourages effective collaboration through real-time editing and comments, allowing team members to communicate seamlessly during the design and brainstorming process. It helped me in building my own GUI form with precise set bounds values which saved me a lot of time and it was very easy to use.

Class Diagram

An object-oriented system's class structure and relationships are shown in a class diagram, a particular type of Unified Modelling Language. Furthermore, it offers a virtual representation of the classes, together with their associated characteristics, methods, and components. Classes are represented by rectangles in a class diagram. For the purpose of indicating the class's access level, the class name is expressed as (+ for public and - for private). In light of this, class diagrams are an important tool for planning and describing the structures of object-oriented systems.

Figure 1 Class Diagram



f

Pseudocode

A method of describing an algorithm using both natural language and programming languages is referred to as pseudocode. It shouldn't be considered as a language for general-purpose programming, rather it is meant to be a description of a problem's solution.

CREATE a parent class StudentGUI that implements Action Listener

DO

DECLARE MainMenuFrame1, RegularFrame1, DropoutFrame1 as JFrame

DECLARE MainImage, RegImage, DropImage as ImageIcon

DECLARE Arraylist named Student students

DECLARE instance variable StudentNameLabel1, DOBLabel1, EnrollmentIDLabel1,

DOEnrollmentLabel1, CourseNameLabel1, TutionFeeLabel1, NumOfModulesLabel1, NumOfCredithrsLabel1, NumOfDaysPresentLabel1, CourseDurationLabel1 as JLabel

DECLARE instance variable StudentNametf1, EnrollmentIDtf1, CourseNametf1, TutionFeetf1, NumOfModulestf1,NumOfCredithrstf1, NumOfDaysPresenttf1, CourseDurationtf1 as JTextField

DECLARE instance variable DOBYearcb1, DOBMonthcb1, DOBDaycb1, DOEnrollmentYearcb1, DOEnrollmentMonthcb1, DOEnrollmentDaycb1, as JComboBox

DECLARE instance variable PresentPercentageButton, RegularHomeButton, GrantCertificateButton, RegularAddstdButton, RegularDisplayButton, RegularClearButton, as JButton

DECLARE instance variable StudentNameLabel2, CourseNameLabel2,DOBLabel2,EnrollmentIDLabel2, DOEnrollmentLabel2, CourseDurationLabel2, TutionFeeLabel2, NumOfRemainingModulesLabel2, NumOfMonthsAttendedLabel2, RemainingAmountLabel2, DODropoutLabel2as JLabel

DECLARE instance variable StudentNametf2, CourseNametf2, EnrollmentIDtf2, CourseDurationtf2, TutionFeetf2, NumOfRemainingModulestf2, NumOfMonthsAttendedtf2, RemainingAmounttf2as JTextField

DECLARE instance variable DOBYearcb2, DOBMonthcb2, DOBDaycb2, DOEnrollmentYearcb2, DOEnrollmentMonthcb2, DOEnrollmentDaycb2, DODropoutYearcb2, DODropoutMonthcb2, DODropoutDaycb2as JComboBox

DECLARE instance variable PayButton, DropoutHomeButton, RemoveStudentButton, DropoutAddstdButton, DropoutDisplayButton, DropoutClearButton as JButton

CREATE StudentGUI for Frame Regular and Dropout

CALL method actionPerformed

DO

IF eventSource is RegularButton

DO

Set visibility of RegularButton to true

Set visibility of MainMenuFrame1 to false

ELSE IF event.getSource() is equal to DropoutButton

DO

Set visibility of Dropout to true

Set visibility of MainMenuFrame1 to false

ELSE IF event.getSource() is equal to RegularHomeButton

DO

Set visibility of MainMenuFrame1 to true

Set visibility of RegularFrame1 to false

ELSE IF eventSource is DropoutHomeButton

DO

Set visibility of MainMenuFrame1 to true

Set visibility of DropoutFrame1 to false

ELSE IF eventSouce is GrantCertificateButton

DO

IF (EnrollmentIDtf1, CourseNametf1 is empty)

SHOW message ("Error, Empty textFields .Please input properly!")

ELSE

DO

TRY

DO

Retrieve values from the textfields in String datatype.

Boolean check equals to true

FOR each Student in students array list:

IF regstudent is an instance of Regular

Convert regstudent to a Regular named regularStudent

IF regularStudents's enrollmentID is equal to given enrollmentID:

CALL regularStudent's grantCertificate method with parameters courseName, enrollmentID and dateOfEnrollment

SHOW message("student with enrolment id [enrollmentID] has been Granted Certificate!")

```
                    END IF
            END IF
        END FOR
        CATCH exception
            DO
                    SHOW message (“Error, Please input again!”)
            END DO
        END DO
    END DO


    ELSE IF  eventSource is RegularAddstdButton
        IF (EnrollmentIDtf1,  CourseNametf1, StudentNametf1, TutionFeetf1,
        NumOfModulestf1, NumOfCredithrstf1, NumOfDaysPresenttf1,
        CourseDurationtf1  is empty)
            SHOW message (“Error, Please fill in all details..”)
        END IF
        ELSE
        DO
            TRY
            DO
                    Retrieve values from the textfields in String datatype.
                    IF students is Empty
                    DO
                        STORE addStudent as new Regular
                        ADD students in addStudent
```

```
        SHOW message arraylist has been added
    END DO
    ELSE
    DO
        Integer EnrollmentIDnew equals enrollmentID
        Boolean isDuplicate equals false
        FOR each Student existingStudent in students:
            IF existingStudent is instance of Regular
            THEN Regular regstd equals Regular
            existingStudent
            IsDuplicate equals to true
            END DO
        END FOR
    END DO
    IF isDuplicate equals to true
        SHOW message (“The enrolmentID already
        exists”)
    END IF
    ELSE
        CALL addStudent Regular method with
        parameters  dateOfEnrollment, enrollmentID,
        courseName, studentName, dateOfBirth,
        courseDuration,  tutionFee, numOfModules,
        numOfCreditHours and daysPresent
        THEN student addStudent
```

SHOW message(“Student has been successfully added”)

END ELSE

CATCH exception

DO

PRINT exception message

SHOW message (“Error, Invalid input. Please check the entered values.”)

END DO

END DO

END DO

ELSE IF eventSource is PresentPercentageButton

IF (courseName and EnrollmentID is empty)

SHOW message (“Error, Please Add the details..”)

END IF

ELSE

TRY

DO

Retrieve values from the textfields in String datatype.

CALL student Regular method with parameters dateOfEnrollment, enrollmentID, courseName, studentName, dateOfBirth, courseDuration, tutionFee, numOfModules, numOfCreditHours and daysPresent

DECLARE char datatype as grade

IF daysPresent Greater then courseDuration from class Student

SHOW message (“Error, numbers of days present cannot be greater than course duration..”)

RETURN

ELSE percentage equals daysPresent divides courseDuration multiplied by hundred

DO

IF Percentage Greater Than or Equals to ***EIGHTY***

ASSIGN isGrantedScholardhip as True

RETURN A

ELSE IF Percentage Greater Than or Equals to ***SIXTY***

RETURN B

ELSE IF Percentage Greater Than or Equals to ***FOURTY***

RETURN C

ELSE IF Percentage Greater Than or Equals to ***TWENTY***

RETURN D

ELSE

RETURN E

END IF

SHOW message(“Present Percentage ” [grade] “Result”)

END DO

```
                    CATCH exception
                    DO
                        SHOW message ("Error, Invalid inputfor days
                        Present. Please check the entered values.")
                    END DO
            END DO
    END DO

    ELSE IF eventSource is  RegularDisplayButton
        DO
            IF students is Empty
                SHOW message ("Error: No Information to Display")
            END IF
            ELSE
            FOR each student belongs to arraylist students
                IF student is the instanceof Regular
                THEN Regular RegObj equals to student
                PRINT ("The information of Regular students are given:")
                CALL display method on RegObj
                END IF
                ELSE
                    SHOW message("Error: No information on Regular
                    Student to display")
            END FOR
        END DO
```

ELSE IF eventSource is RegularClearButton

DO

StudentNametf1.setText() as **null**

EnrollmentIDtf1.setText() as **null**

CourseNametf1.setText() as **null**

TutionFeetf1.setText() as **null**

NumOfModulestf1.setText() as **null**

NumOfCredithrstf1.setText() as **null**

NumOfDaysPresenttf1.setText() as **null**

CourseDurationtf1.setText() as **null**

END DO

ELSE IF eventSource is DropoutAddstdButton

DO

IF (StudentNametf2, CourseNametf2, EnrollmentIDtf2, CourseDurationtf2, TutionFeetf2, NumOfRemainingModulestf2, NumOfMonthsAttendedtf2, RemainingAmounttf2 is Empty)

THEN Show message (“Error, Please fill in all the details.”)

END IF

ELSE

TRY

DO

Retrieve values from the textfields in String datatype.

```
        IF students is empty
        CALL addDStudent Dropout method with parameters
        dateOfBirth, tutionFee, courseDuration, studentName,
        numOfRemainingModules, numOfMonthsAttended and
        dateOfDropout
        ADD students addDStudent
        SHOW message arraylist has been added
    END DO
    ELSE
    DO
        NewEnrollmentID equals enrollmentID
        Boolean isDuplicate equals to false
        FOR each Student DropoutStudent belongs to students
        IF DropoutStudent is instanceof Dropout
        THEN Dropout dropstd equals to DropoutStudent
            IF dropstd.getEnrollmentID() equals to newEnrollmentID
                THEN isDuplicate equals to true
            END IF
        END IF
    END FOR
    IF isDuplicate equals to true
        SHOW message("this enrollmentid already exists")
    END IF
    ELSE
        DO
```

CALL addDStudent Dropout method with parameters dateOfBirth, tutionFee, courseDuration, studentName, numOfRemainingModules, numOfMonthsAttended and dateOfDropout

ADD students addDStudent

SHOW message arraylist has been added

END DO

END ELSE

END DO

CATCH exception

DO

PRINT exception message

SHOW Message(“Invalid input. Please check the entered values.")

END DO

END ELSE

END DO

ELSE IF eventSource is PayButton

DO

TRY

DO

Retrieve values from the textfields in String datatype

ITERATE through the students list:

FOR each Student student in students

IF students enrollmentID equals to enrollmentID

IF student is the instanceof Dropout

DOWNCAST student to a Dropout

THEN Dropout dropout equals (Dropout) student

SHOW message(“bills paid successfully”)

END IF

ELSE

SHOW message(“student is not a dropout”)

END ELSE

BREAK

END IF

END FOR

END DO

CATCH exception

DO

SHOW message(“Invalid enrollment ID format. Please enter a valid number.”)

END DO

END DO

ELSE IF eventSource is RemoveStudentButton

DO

TRY

DO

Retrieve values from the textfields in String datatype.

```
ITERATE through the students list:
FOR each Student student in students
    IF students enrollmentID equals to enrollmentID
        DOWNCAST student to a Dropout
        THEN Dropout dropout equals (Dropout)   student
            IF dropout.HasPaid() is true
                CALL Student to Dropout
                Remove Dropout from students
                SHOW message("Dropout student removed     successfully")
                END IF
                ELSE
                    SHOW message("Student is not a dropout")
                END ELSE
                BREAK
        END DO
        CATCH exception
        DO
            SHOW message("Invalid enrollment ID format. Please enter a valid number.")
        END DO
    END DO
END DO
```

```
ELSE IF eventSource is DropoutDisplayButton
    DO
    IF students is empty
        SHOW message ("Error: No information to display")
    END IF
    ELSE
        FOR each student belongs to students
            IF student is the instanceof Dropout
                THEN Dropout DropObj equals to (Dropout) student
                    PRINT ("The information of Dropout students are
                    given: " )
                CALL display method on DropObj
            END IF
            ELSE
                SHOW message("Error: No information on Dropout
                Student to display")
            END ELSE
        END FOR
    END DO

    ELSE IF eventSource is  DropoutClearButton
    DO
        EnrollmentIDtf2.setText() as null
        StudentNametf2.setText() as null
        CourseNametf2.setText() as null
```

CourseDurationtf2.setText() as **null**

TutionFeetf2.setText() as **null**

RemainingAmounttf2.setText() as **null**

NumOfRemainingModulestf2.setText() as **null**

NumOfMonthsAttendedtf2.setText() as **null**

END DO

END DO

DO

ASSIGN StudentGUI as new StudentGUI

END DO

END DO

Method Description

A class constructor was created that had Action Listener to it. In the button functionality:

Add button for regular student

The button RegularAddstdButton required StudentNametf1, EnrollmentIDtf1, CourseNametf1, TutionFeetf1, NumOfModulestf1, NumOfCredithrstf1, NumOfDaysPresenttf1 and CourseDurationtf1 from StudentGUI. If the required parameter fields were empty, a dialog box was made to pop through the JOptionPane.showMessageDialog. Else, parameters were taken from the GUI input. String parameters input from the users were converted into int. Checking if the Array List of StudentGUI students is empty. Button RegularAddstdButton took in constructors, the objects were added to Regular and then added the Array list.
If successful, then again a message was set to be popped. If it wasn't successful, loop was used to check if Regular was already present, if yes then the loop would break and not add. It'd again pop text message. Or else the button would perform it's task, add and then again pop a message on it's succession.

Grant Certificate for Regular Student

If button GrantCertificateButton was clicked and if the TextFields EnrollmentIDtf1 and CourseNametf1 were empty, a text message was designed to pop. grantCertificate method was called. Parameters were taken from GUI input by the user. Parameters were contained and converted from string to int. dateOfEnrollment was stored in a single variable. A text message was designed to pop under the successful graduating from
Regular.

PresentPercentage Button for Regular student

If button PresentPercentageButton was clicked and if the TextFields EnrollmentIDtf1 and CourseNametf1 were empty, a text message was designed to pop. presentPercentage method was called. The if else loop from the presentPercentage method returns the value in char which represents the grade of the student. Parameters were taken from GUI input by the user. Parameters were contained and converted from string to int.dateOfEnrollment and dateOfBirth was stored in a single variable. A text message was designed to pop presenting the grade of the student.

Display button for regular student

On clicking RegularDisplayButton button, checking if the ArrayList is Empty, a message would pop up. Looping through the ArrayList is done, it was checked whether students belonged to Regular or not. If yes the system would display the given user information on the terminal Or else an error message would pop

Clear button for regular student

On clicking of RegularClearButton button, every TextFields from the GUI i.e StudentNametf1, EnrollmentIDtf1, CourseNametf1, TutionFeetf1, NumOfModulestf1, NumOfCredithrstf1, NumOfDaysPresenttf1 and CourseDurationtf1 were set null to clear out the information input by the user.

Add button for Dropout Student

The button DropoutAddstdButton required StudentNametf2, CourseNametf2, EnrollmentIDtf2, CourseDurationtf2, TutionFeetf2, NumOfRemainingModulestf2, NumOfMonthsAttendedtf2 and RemainingAmounttf2 from StudentGUI. If the required parameter fields were empty, a dialog box was made to pop through the

JOptionPane.showMessageDialog.Else, parameters were taken from the GUI input. String parameters input from the users were converted into int. Checking if the Array List of StudentGUI students is empty. Button DropoutAddstdButton took in constructors, the objects were added to Dropout and then added the the Arraylist. If successful, then again a message was set to be popped. If it wasn't successful, loop was used to check if Dropout was already present, if yes then the loop would break and not add. It'd again pop text message.or else the button would perform it's task, add and then again pop a message on it's succession.

Pay button for Dropout Student

The button PayButton would take EnrollmentIDtf2 and NumOfMonthsAttendedtf2 from StudentGUI. Both of these parameters were integers so they were converted into String data type. The list was iterated to find a match while initiating a for loop Checking if the student is a Dropout Downcasting the student to a Dropout.Calling billsPayable method and updating the display.Displays a message where Bills were cleared or Display a message if the student was not a dropout.

Remove Student Button for dropout student

The RemoveStudentButton button required only EnrollmentIDt2 from the StudentGUI. Since it was integer data type it was converted into String. Iterating throughout the list to find a match through for loop .Checking if the student is from dropout.Downcasting the student to a Dropout. Checking if the bills are cleared calling hasPaid parameter.Calling removeStudent method from the dropout class and updating in the display.If yes then message is popped up saying student is removed or display a message saying the student is not a dropout.

Display Button for dropout student

On clicking DropoutDisplayButton button, checking if the ArrayList is Empty, a message would pop up. Looping through the Array List is done, it was checked whether students belonged to Dropout or not.If yes the system would display the given user information on the terminal Or else an error message would pop

Clear Button for Dropout Student

The RegularClearButton button takes every TextFields from the GUI i.e StudentNametf2, CourseNametf2, EnrollmentIDtf2, CourseDurationtf2, TutionFeetf2, NumOfRemainingModulestf2,NumOfMonthsAttendedtf2 and RemainingAmounttf2which were set null to clear out the information input by the user.

Testing

Test-1

Test No.:	1
Objective:	Compiling and Running using Command Prompt
Action:	C:\Users\sikum\22085627_SikumHangmaMadi> C:\Users\sikum\22085627_SikumHangmaMadi> javac StudentGUI.java C:\Users\sikum\22085627_SikumHangmaMadi> java StudentGUI.java
Expected Result:	The program would be compiled and run using cmd prompt.
Actual Result:	The program was compiled and ran using command prompt
Conclusion:	The test was successful

Table 1 Test no.1

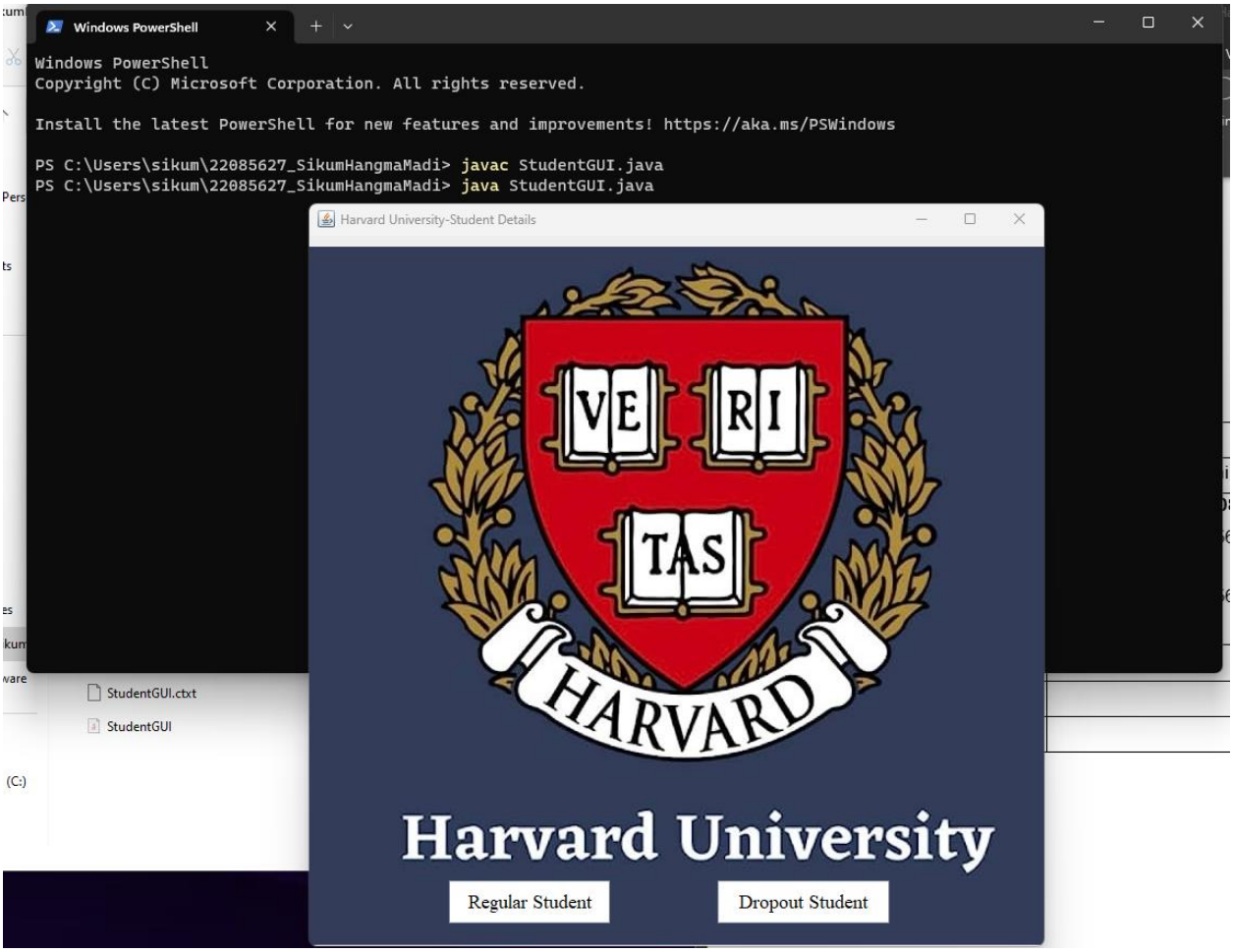


Figure 2 Testno.1

Test-2

Test No.:	2
Objective:	Adding Regular and Dropout Student, calculate the Present Percentage and Grant Certificate of Regular Class. Calculate Pay the bills and Remove Student from Dropout Class.
Action:	Add Regular Student Enrollment ID: 4 Date of Birth: 2002-05-5 Course Name: Java Student Name: sikum Course Duration: 14 months Tution Fee: $23000

	Date of Enrollment: 2002-05-8 Number of Modules: 3 Number Of Credit Hours: 15 Days Present: 12 Add Dropout Student Enrollment ID: 3 Date of Birth: 2002-06-7 Course Name: Java Student Name: sikum Course Duration: 3 months Tution Fee: $23000 Date of Enrollment: 2004-08-4 Number of remainig modules: 7 Number of months attended: 3 Date of drop out: 2002-06-04 Remaining amount: 0 Calculate the Present Percentage of Regular Student Present percentage = (double) daysPresent / CourseDuration * 100 Grant Certificate of Regular Student Pay the bills of Dropout Student remainingAmount = (CourseDuration - numOfMonthsAttended) * TutionFee Remove the student
Expected Result:	Students was updated where user input values were displayed.
Actual Result:	Student was updated and the values were displayed in the terminal.

Conclusion:	The test was successful.

Table 2 test no.2

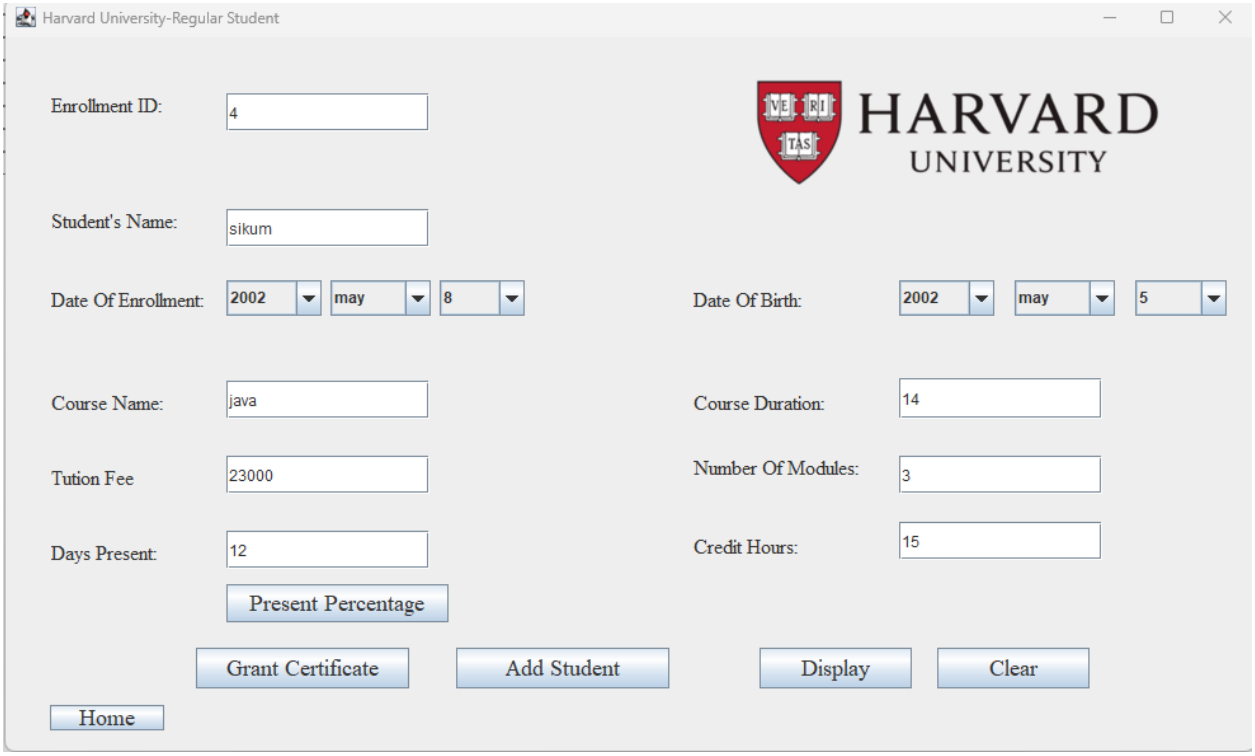


Figure 3 Test no.2.1

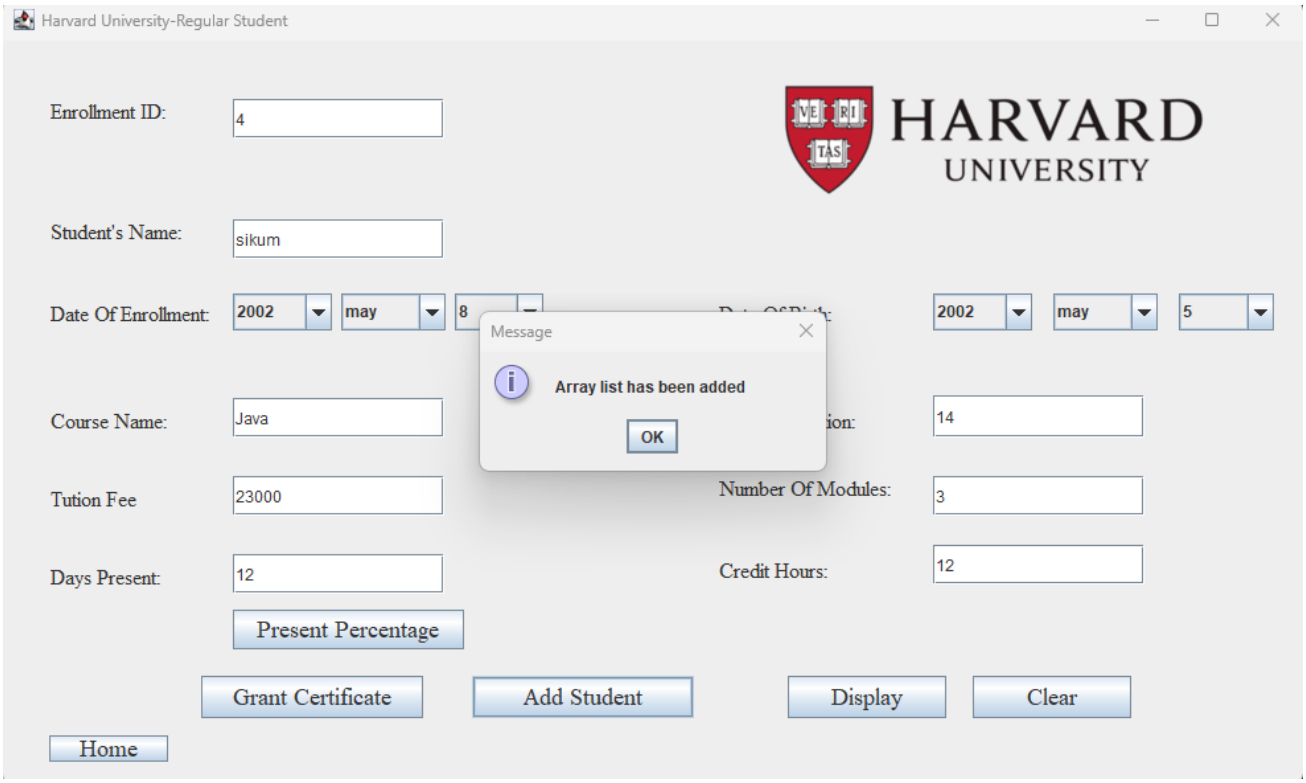


Figure 4 Test 2.2 Add Student button

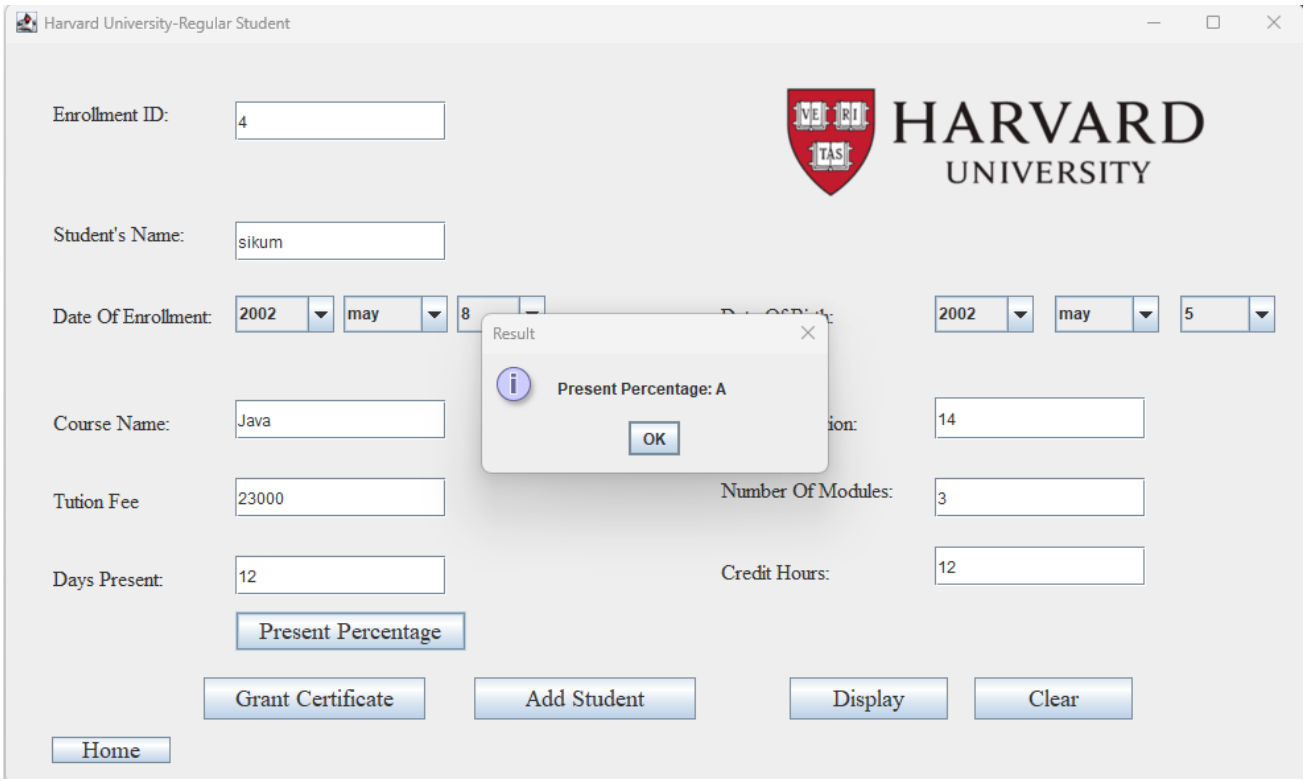


Figure 5 Test 2.3 Present Percentage button

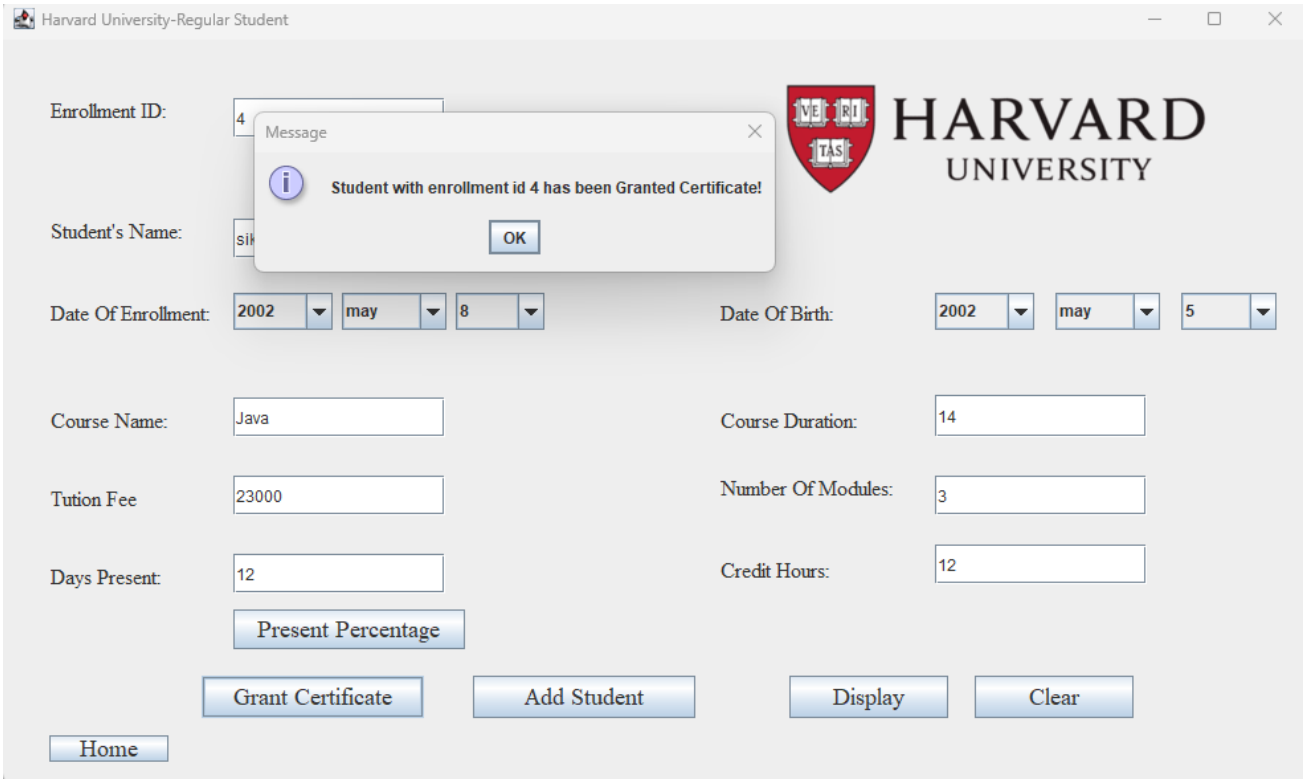


Figure 6 Test 2.4 Grant Certificate button

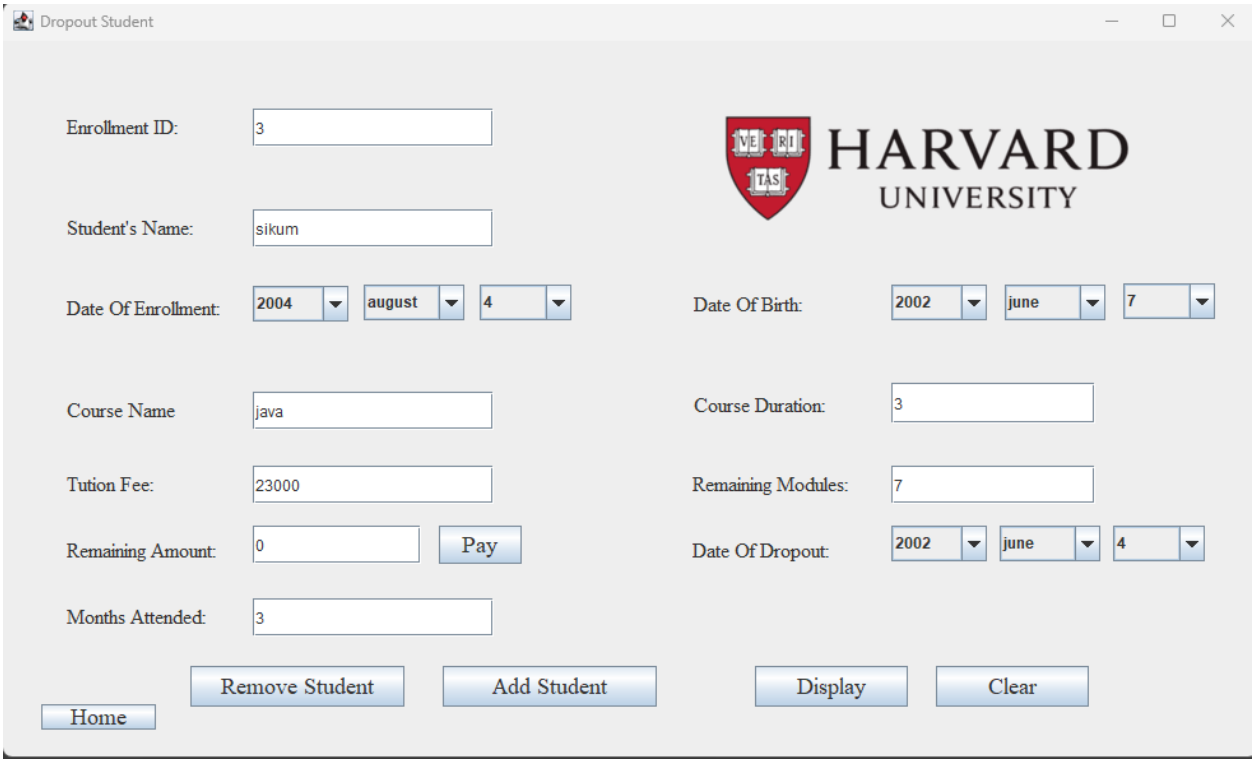


Figure 7 Test No.2.5

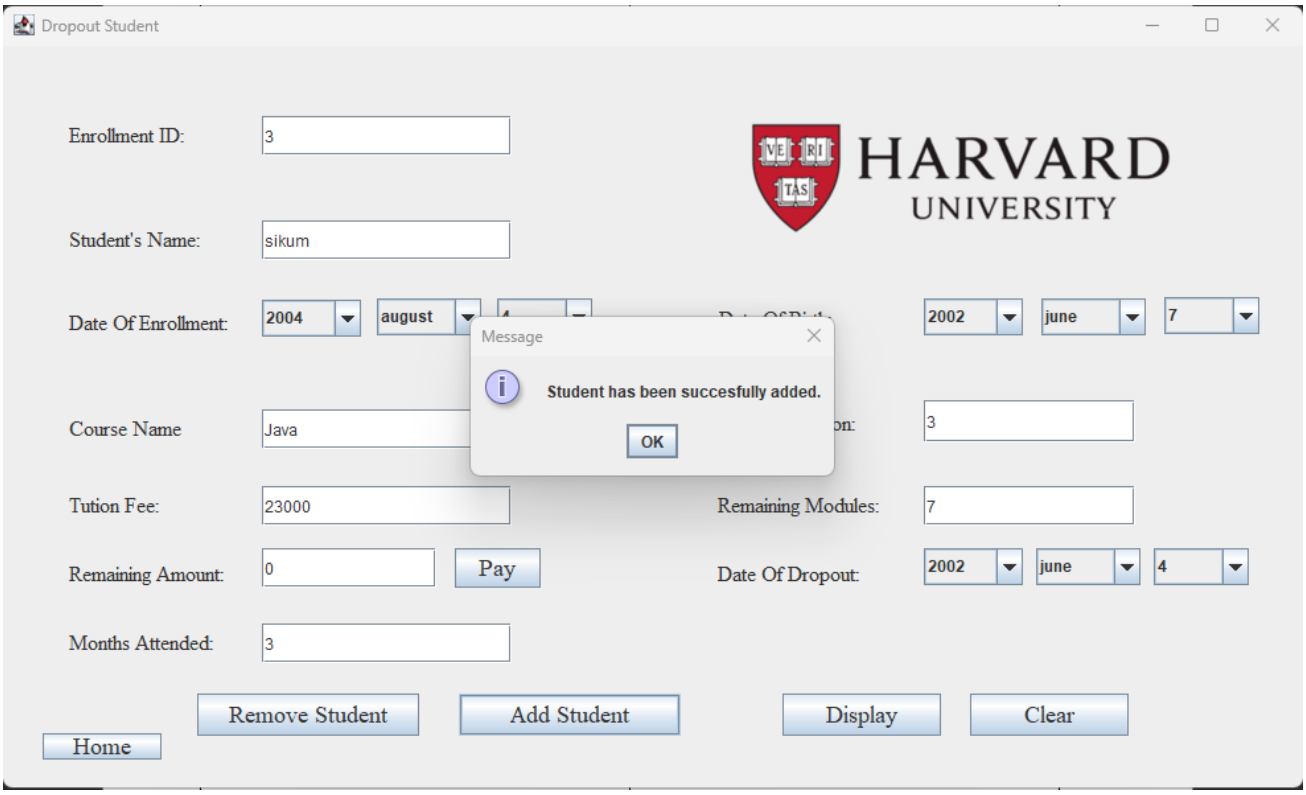


Figure 8 Test 2.5 Add Dropout Student Button

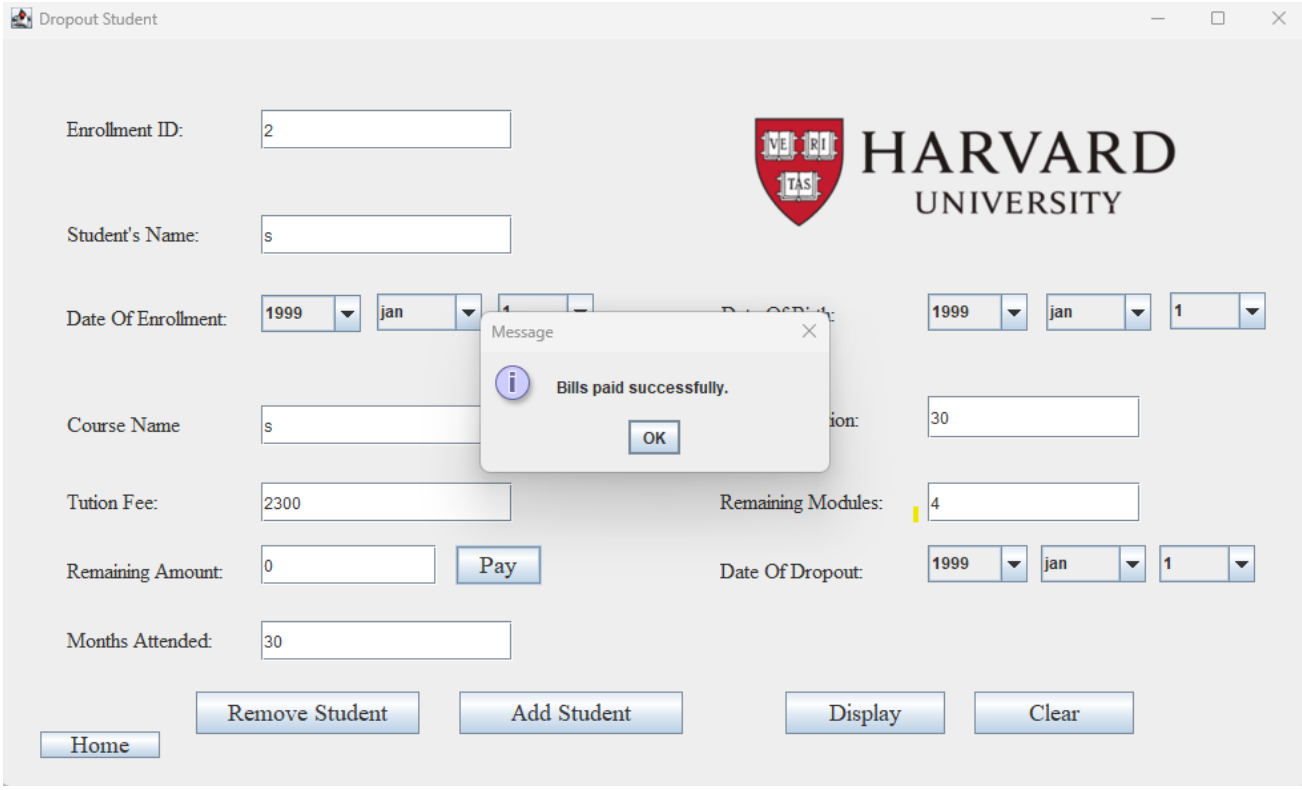


Figure 9 test 2.6 Pay Button of Dropout

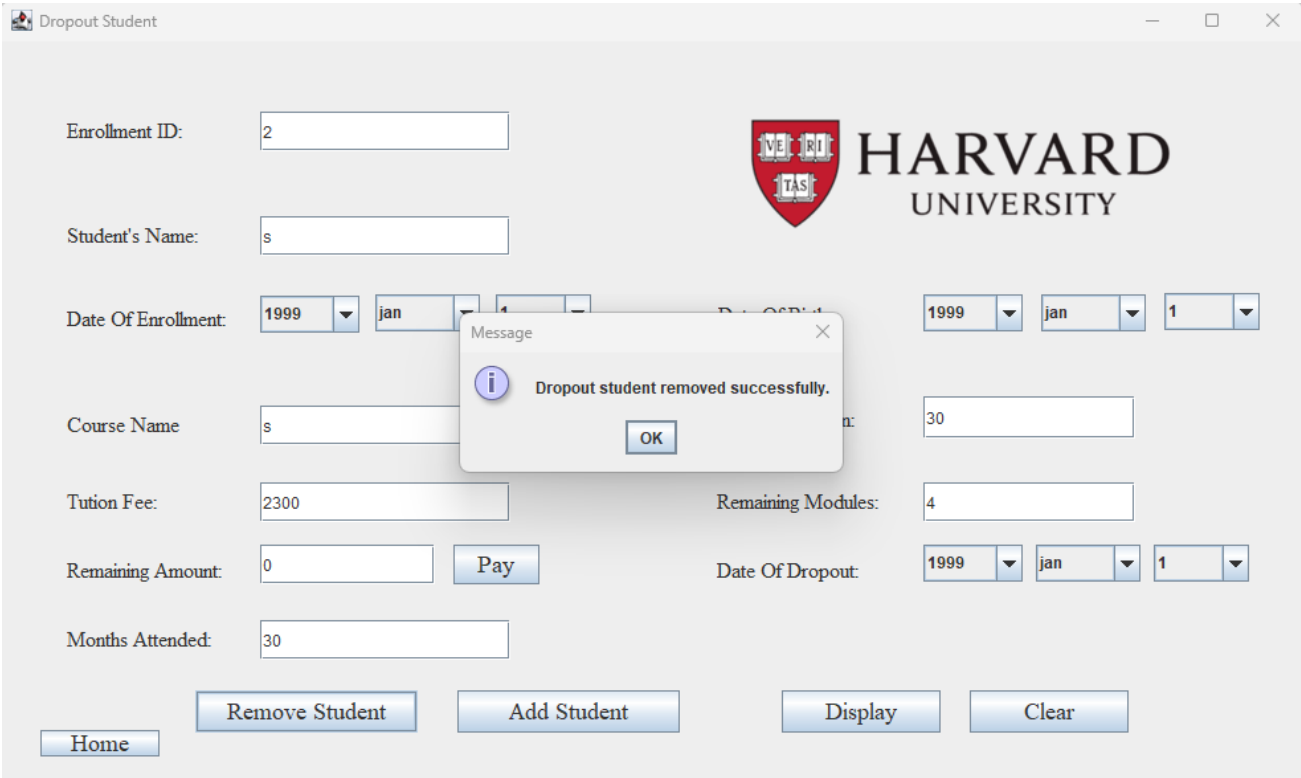


Figure 10 test 2.6 Remove std Button of dropout

Test-3.1

Test No.:	3.1
Objective:	Test that appropriate dialog boxes appear when unsuitable values are entered for the Enrollment ID.
Action:	Adding same Enrollment ID when updating an Array list
Expected Result:	Appearance of an alert dialog box with appropriate message
Actual Result:	Dialogue box appeared with an alert message.
Conclusion:	The test was successful

Table 3 Test no.3.1

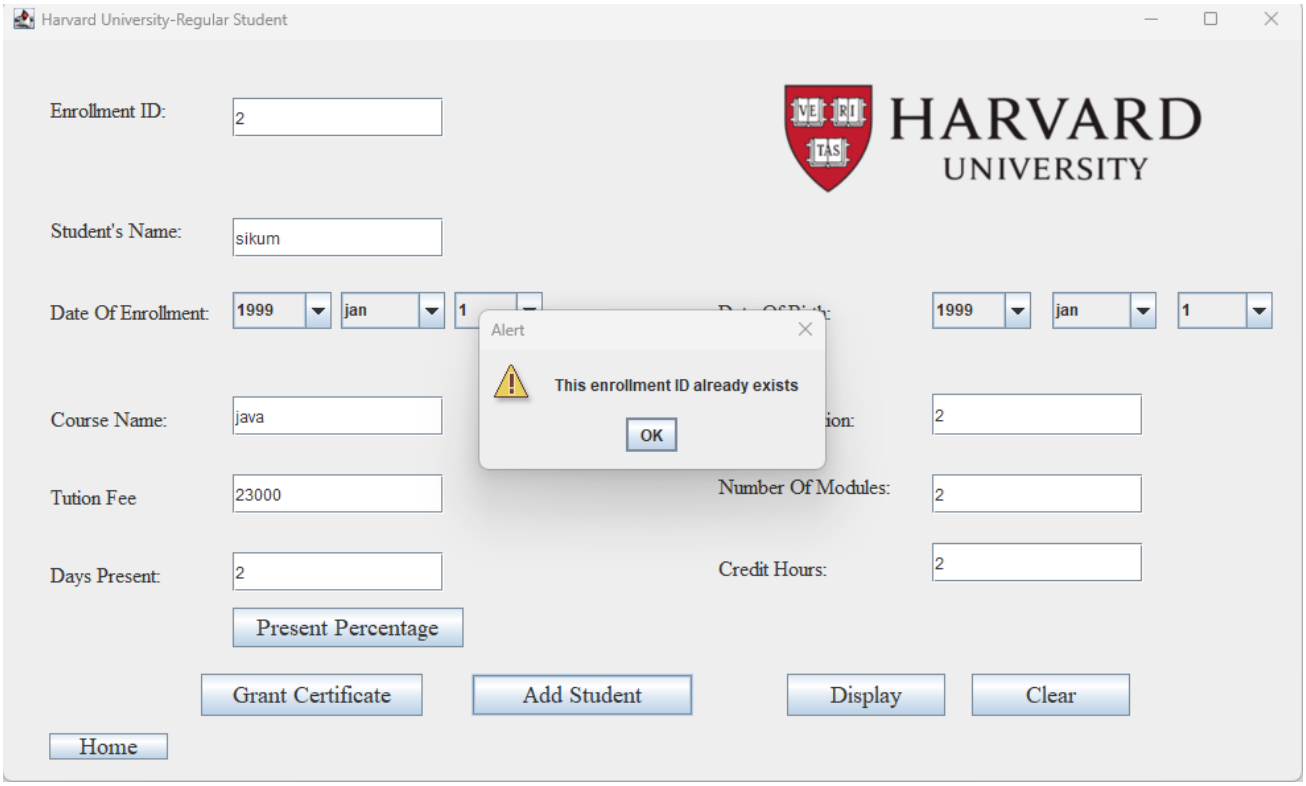


Figure 11 Test 3.1

Test-3.2

Test No.:	3.2
Objective:	Test that appropriate dialog boxes appear when unsuitable values are entered for the Enrollment ID.
Action:	Having the text fields empty so that Number Format Exception will appear.
Expected Result:	Appearance of error dialogue box with number format exception.
Actual Result:	Error dialogue box appeared with the exception message.
Conclusion:	The test was successful.

Table 4 Test No. 3.2

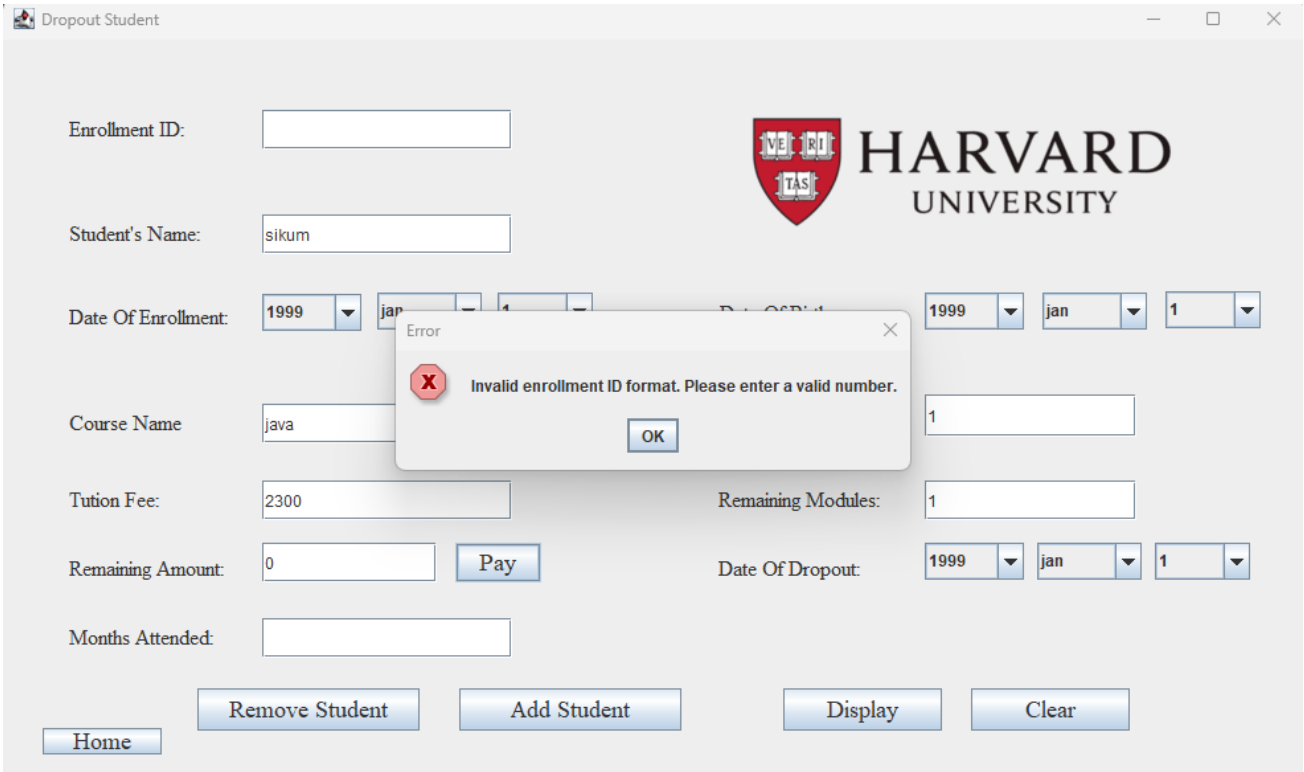


Figure 12 Test no.:3.2

Test-3.3

Test No.:	3.3
Objective:	Test that appropriate dialog boxes appear when unsuitable values are entered for the Enrollment ID.
Action:	Updating the enrollment ID directly for grant certificate without updating the enrollment ID in the new arraylist.
Expected Result:	Appearance of error message with appropriate message.
Actual Result:	Error message appeared with appropriate message.
Conclusion:	The test was successful

Table 5 Test No, 3.3

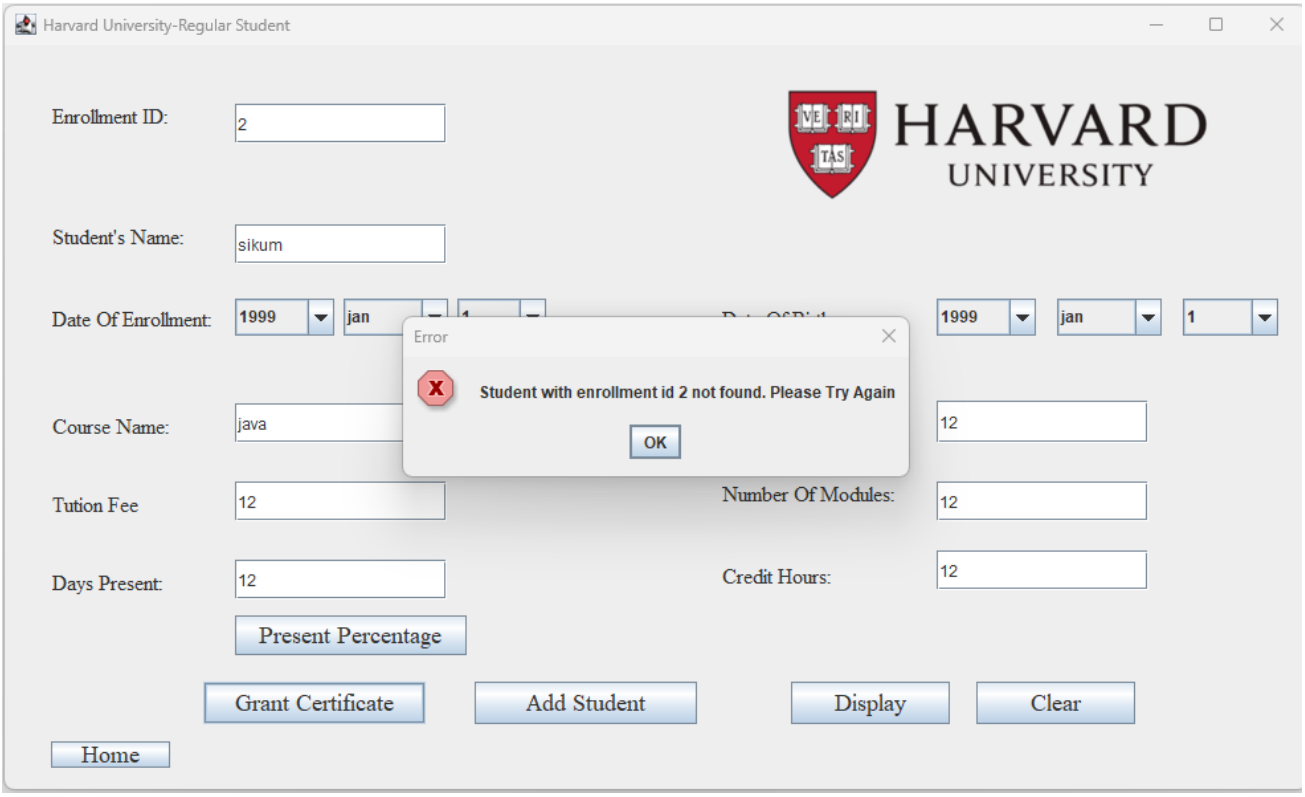


Figure 13 Test no:3.3

Error Detection And Checking

Data transmission from one source to another is significantly aided by error detection and correction code. Error correction codes are generated by using the specific algorithm used for removing and detecting errors .Important or secure data will be lost as a result of the data inaccuracies.

1) Syntax Error

A syntax error is a mistake or error of the rules or structure of a programming language. It usually occurs when the programming language have grammatical error.

```
new Regular(dateOfEnrollment, enrollmentID, courseName, studentName, dateOfBirth, courseDuration,
utionFee, numOfModules, numOfCred
nt);
geDialog(RegularFrame1, "Array list has been added");
```

Undeclared variable: enrollmentID

Figure 14 Syntax Error

Correction

```
arameters is to be same as regular class
new Regular(dateOfEnrollment, enrollmentID, courseName, studentName, dateOfBirth, courseDuration,
utionFee, numOfModules, numOfCreditHours, daysPresent);
nt);
geDialog(RegularFrame1, "Array list has been added");
```

Figure 15 Correction of Syntax Error

2) Semantic Error

A semantic error is a mistake or error in the logic of a program. Unlike syntax error, the semantic error occur when the code is syntactically correct but it does not behave since it has logical error.

```
//if duplication true cannot add student
int EnrollmentIDnew = enrollmentID;
boolean isDuplicate = false;
for (Student existingStudent : students) {
    if (existingStudent instanceof Regular)
```

Figure 16 Semantic Error

Correction

```
//if duplication true cannot add student
int EnrollmentIDnew =  Integer.parseInt(enrollmentID);
boolean isDuplicate = false;
for (Student existingStudent : students) {
    if (existingStudent instanceof Regular) {
```

Figure 17 Correction for Semantic Error

3) Logical Error

A logical error occurs when the program does not produce the expected result due to incorrect algorithm. Unlike syntax error which can be easily identified, logical error are more challenging to identify.

```
String DOEnrollmentMonthcb3 = DOEnrollmentMonthcb2.getSelectedItem().toString();
String DOEnrollmentDayb3 = DOEnrollmentDaycb2.getSelectedItem().toString();
String dateOfEnrollment = DOEnrollmentYearcb3 + "/" +DOEnrollmentMonthcb3 + "/" + DOEnrollmentDaycb3;
```

Figure 18 Logical Error

Correction

```
String DOEnrollmentMonthcb3 = DOEnrollmentMonthcb2.getSelectedItem().toString();
String DOEnrollmentDayb3 = DOEnrollmentDaycb2.getSelectedItem().toString();
String dateOfEnrollment = DOEnrollmentYearcb3 + "/" +DOEnrollmentMonthcb3 + "/" + DOEnrollmentDaycb3;
```

Figure 19 Correction for Logical Error

Conclusion

Throughout this project, I learned a lot about object-oriented programming concepts and GUI development in Java. Working on the Student, Regular, and Dropout classes gave significant hands-on experience in developing a well-structured programme.

What have I learned?

I gained a better knowledge of class hierarchies and inheritance as a result of this assignment. The contrast between the primary class "Student" and its subclasses "Regular" and "Dropout" demonstrated the significance of organising code to enhance reusability and maintainability by the concept of inheritance from OOP. I also improved my skills in creating graphical user interfaces (GUIs) to improve user interaction and overall user experience. It was a good way to gain my experience if i ever have to work on sectors like this in my future.

Difficulties faced

Multiple challenges arose while working on the project, putting my problem-solving skills to the test. One significant challenge was handling the interface between the GUI components and the underlying class structure. Aligning the graphical depiction of student data with the logic of the Regular and Dropout classes necessitated extensive thought and attention to detail. Furthermore, creating error handling and validation techniques was complicated since preserving data integrity across

different class instances required careful planning. And moreover having a good GUI design was definitely a difficulties that i had faced since I had problem choosing the perfect layout for users to input value also maintaining the aesthetics of my GUI.

How I solved those problems?

To overcome these issues, I used a modular design strategy for the interaction between GUI components and class structure, ensuring that each class encapsulated its own behaviour and data. This separation of concerns allowed for clearer coding and easier debugging. I created accurate input checks within the GUI components for data validation, ensuring that the information submitted by users adhered to the anticipated format and limitations. I was able to develop a smoother interplay between the user interface and the underlying business logic through extensive testing .

In conclusion, this project not only increased my technical skills in Java programming and GUI development, but it also highlighted the need of thorough design and problem-solving in software development. Working on the Student, Regular, and Dropout classes, as well as their associated graphical interfaces, has provided me with essential tools that I will use in future projects and programming endeavours.

Reference

Bibliography

Herbit. (2008). *Java a beginner's guide:Eith edition.* Adventure Works Press.

Hortsmann, C. S. (2007). *Core Java.* Cay S Hortsmann.

(n.d.). *https://www.bluej.org/.*

(n.d.). *https://www.microsoft.com/en-us/microsoft-365/word.*

Selawsky, J. (2023). 10 Deadly Mistakes to Avoid When Learning Java. pp. 50-60.

Appendix

```
import javax.swing.*;

import java.awt.event.*;

import java.awt.*;

import java.util.*;


public class StudentGUI implements ActionListener

{

    //declaring all components

    private JFrame MainMenuFrame1, RegularFrame1, DropoutFrame1;


    private ImageIcon MainImage, RegImage, DropImage;


    //making only one arraylist

    private ArrayList<Student> students = new ArrayList<>();


    private JLabel

    Mainlabel, Reglabel, Droplabel,//for image

        //Regular Student Label

    StudentNameLabel1, DOBLabel1, EnrollmentIDLabel1, DOEnrollmentLabel1,

    CourseNameLabel1, TutionFeeLabel1, NumOfModulesLabel1, NumOfCredithrsLabel1,

    NumOfDaysPresentLabel1, CourseDurationLabel1,
```

```
    //Dropout Student Label
  StudentNameLabel2,CourseNameLabel2, DOBLabel2,EnrollmentIDLabel2,
DOEnrollmentLabel2, CourseDurationLabel2, TutionFeeLabel2,
  NumOfRemainingModulesLabel2,
NumOfMonthsAttendedLabel2,RemainingAmountLabel2,
  DODropoutLabel2;

  private JTextField
    //regular form textfield
  StudentNametf1, EnrollmentIDtf1, CourseNametf1, TutionFeetf1, NumOfModulestf1,
  NumOfCredithrstf1, NumOfDaysPresenttf1, CourseDurationtf1,
    //dropout student form
  StudentNametf2, CourseNametf2, EnrollmentIDtf2, CourseDurationtf2, TutionFeetf2,
NumOfRemainingModulestf2,
  NumOfMonthsAttendedtf2, RemainingAmounttf2;

  private JComboBox<String> //regular student combo box
  DOBYearcb1, DOBMonthcb1, DOBDaycb1,  DOEnrollmentYearcb1,
DOEnrollmentMonthcb1, DOEnrollmentDaycb1,
    //dropout student combo box
  DOBYearcb2, DOBMonthcb2, DOBDaycb2, DOEnrollmentYearcb2,
DOEnrollmentMonthcb2, DOEnrollmentDaycb2,
  DODropoutYearcb2, DODropoutMonthcb2, DODropoutDaycb2;

  private JButton RegularButton, DropoutButton,
    //regular student button
```

```
    PresentPercentageButton, RegularHomeButton, GrantCertificateButton,
RegularAddstdButton, RegularDisplayButton, RegularClearButton,

        //dropout student button

    PayButton, DropoutHomeButton, RemoveStudentButton, DropoutAddstdButton,
DropoutDisplayButton, DropoutClearButton;


    public StudentGUI(){

        String[] Month =
{"jan","feb","march","april","may","june","july","august","sep","oct","nov","dec"};

        String[] Day = {"1", "2", "3", "4", "5", "6", "7", "8"};

        String[] Year = {"1999", "2000", "2001", "2002", "2003", "2004", "2005"};


        MainMenuFrame1 = new JFrame("Harvard University-Student Details");

        MainImage = new ImageIcon("harvard1.jpg");

        Mainlabel = new JLabel(MainImage);


        RegularButton = new JButton("Regular Student");

        RegularButton.setBackground(Color.WHITE);

        RegularButton.setForeground(Color.BLACK);


        DropoutButton = new JButton("Dropout Student");

        DropoutButton.setBackground(Color.WHITE);

        DropoutButton.setForeground(Color.BLACK);


        Font font = new Font("Times New Roman", Font.PLAIN, 18);
```

```
RegularButton.setFont(font);
DropoutButton.setFont(font);




//-----------Regular Student-------------------------------//
RegularFrame1 = new JFrame("Harvard University-Regular Student");
RegImage = new ImageIcon("h3.png");
Reglabel = new JLabel(RegImage);

StudentNameLabel1 = new JLabel("Student's Name:");
StudentNameLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
StudentNametf1 = new JTextField();

DOBLabel1 = new JLabel("Date Of Birth:");
DOBLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
DOBYearcb1 = new JComboBox<String>(Year);
DOBMonthcb1 = new JComboBox<String>(Month);
DOBDaycb1 = new JComboBox<String>(Day);

EnrollmentIDLabel1 = new JLabel("Enrollment ID:");
EnrollmentIDLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
EnrollmentIDtf1 = new JTextField();
```

```
DOEnrollmentLabel1 = new JLabel("Date Of Enrollment:");
DOEnrollmentLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
DOEnrollmentYearcb1 = new JComboBox<String>(Year);
DOEnrollmentMonthcb1 = new JComboBox<String>(Month);
DOEnrollmentDaycb1 = new JComboBox<String>(Day);

CourseNameLabel1 = new JLabel("Course Name:");
CourseNameLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
CourseNametf1 = new JTextField();

CourseDurationLabel1 = new JLabel("Course Duration:");
CourseDurationLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
CourseDurationtf1 = new JTextField();

TutionFeeLabel1 = new JLabel("Tution Fee");
TutionFeeLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
TutionFeetf1 = new JTextField();

NumOfModulesLabel1 = new JLabel("Number Of Modules:");
NumOfModulesLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
NumOfModulestf1 = new JTextField();

NumOfCredithrsLabel1 = new JLabel("Credit Hours:");
NumOfCredithrsLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
```

```
NumOfCredithrstf1 = new JTextField();

NumOfDaysPresentLabel1 = new JLabel("Days Present:");
NumOfDaysPresentLabel1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
NumOfDaysPresenttf1 = new JTextField();
PresentPercentageButton = new JButton("Present Percentage");
PresentPercentageButton.setFont(font);

RegularHomeButton = new JButton("Home");
RegularHomeButton.setFont(font);

GrantCertificateButton = new JButton("Grant Certificate");
GrantCertificateButton.setFont(font);

RegularAddstdButton = new JButton("Add Student");
RegularAddstdButton.setFont(font);

RegularDisplayButton= new JButton("Display");
RegularDisplayButton.setFont(font);

RegularClearButton = new JButton("Clear");
RegularClearButton.setFont(font);

//-----------------dropout student--------------------//
```

```
DropoutFrame1 = new JFrame("Harvard University-Dropout Student");


DropImage = new ImageIcon("h3.png");
Droplabel = new JLabel(DropImage);


DropoutFrame1 = new JFrame("Dropout Student");


StudentNameLabel2 = new JLabel("Student's Name:");
StudentNameLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
StudentNametf2 = new JTextField();


DOBLabel2 = new JLabel("Date Of Birth:");
DOBLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
DOBYearcb2 = new JComboBox<String>(Year);
DOBMonthcb2 = new JComboBox<String>(Month);
DOBDaycb2 = new JComboBox<String>(Day);


EnrollmentIDLabel2 = new JLabel("Enrollment ID:");
EnrollmentIDLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
EnrollmentIDtf2 = new JTextField();


DOEnrollmentLabel2 = new JLabel("Date Of Enrollment:");
DOEnrollmentLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
DOEnrollmentYearcb2 = new JComboBox<String>(Year);
```

```
DOEnrollmentMonthcb2 = new JComboBox<String>(Month);
DOEnrollmentDaycb2 = new JComboBox<String>(Day);

CourseNameLabel2 = new JLabel("Course Name");
CourseNameLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
CourseNametf2 = new JTextField();

CourseDurationLabel2 = new JLabel("Course Duration:");
CourseDurationLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
CourseDurationtf2 = new JTextField();

TutionFeeLabel2 = new JLabel("Tution Fee:");
TutionFeeLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
TutionFeetf2 = new JTextField();

NumOfRemainingModulesLabel2 = new JLabel("Remaining Modules:");
NumOfRemainingModulesLabel2.setFont(new Font("Times New Roman", Font.PLAIN,
16));
NumOfRemainingModulestf2 = new JTextField();

NumOfMonthsAttendedLabel2 = new JLabel("Months Attended:");
NumOfMonthsAttendedLabel2.setFont(new Font("Times New Roman", Font.PLAIN,
16));
NumOfMonthsAttendedtf2 = new JTextField();
```

```
RemainingAmountLabel2 = new JLabel("Remaining Amount:");
RemainingAmountLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
RemainingAmounttf2 = new JTextField();
PayButton = new JButton("Pay");
PayButton.setFont(font);



DODropoutLabel2 = new JLabel("Date Of Dropout:");
DODropoutLabel2.setFont(new Font("Times New Roman", Font.PLAIN, 16));
DODropoutYearcb2 = new JComboBox<String>(Year);
DODropoutMonthcb2 = new JComboBox<String>(Month);
DODropoutDaycb2 = new JComboBox<String>(Day);

RemoveStudentButton = new JButton("Remove Student");
RemoveStudentButton.setFont(font);
DropoutHomeButton = new JButton("Home");
DropoutHomeButton.setFont(font);
DropoutAddstdButton = new JButton("Add Student");
DropoutAddstdButton.setFont(font);
DropoutDisplayButton = new JButton("Display");
DropoutDisplayButton.setFont(font);
DropoutClearButton = new JButton("Clear");
DropoutClearButton.setFont(font);
```

```
//--------------setBounds------------//
Mainlabel.setBounds(0, 10, 700, 650);
RegularButton.setBounds(130, 600, 150, 40);
DropoutButton.setBounds(380,600,160,40);
//---------------set bounds for regular student--------//
Reglabel.setBounds(0, 0, 1500, 150);
//RegularFrame1.setContentPane(Reglabel);

EnrollmentIDLabel1.setBounds(36,44,89,20);
EnrollmentIDtf1.setBounds(174,44,160,30);

StudentNameLabel1.setBounds(36,135,118,20);
StudentNametf1.setBounds(174,135,160,30);

DOBLabel1.setBounds(542,197,97,20);
DOBYearcb1.setBounds(704,191,75,28);
DOBMonthcb1.setBounds(795,191,79,28);
DOBDaycb1.setBounds(890,191,72,28);

DOEnrollmentLabel1.setBounds(36,197,141,20);
DOEnrollmentYearcb1.setBounds(174,191,75,28);
DOEnrollmentMonthcb1.setBounds(256,191,79,28);
DOEnrollmentDaycb1.setBounds(342,191,67,28);
```

```
CourseNameLabel1.setBounds(36,278,103,20);
CourseNametf1.setBounds(174,270,160,30);

CourseDurationLabel1.setBounds(542,278,117,20);
CourseDurationtf1.setBounds(704,268,160,32);

TutionFeeLabel1.setBounds(36,337,80,20);
TutionFeetf1.setBounds(174,329,160,30);

NumOfModulesLabel1.setBounds(542,329,148,20);
NumOfModulestf1.setBounds(704,329,160,30);

NumOfCredithrsLabel1.setBounds(542,391,99,20);
NumOfCredithrstf1.setBounds(704,381,160,30);

NumOfDaysPresentLabel1.setBounds(36,396,100,20);
NumOfDaysPresenttf1.setBounds(174,388,160,30);
PresentPercentageButton.setBounds(174,430,175,30);

RegularHomeButton.setBounds(35,525,90,20);    //Home
GrantCertificateButton.setBounds(150,480,168,32);
RegularAddstdButton.setBounds(355,480,168,32);
RegularDisplayButton.setBounds(594,480,120,32);     //display
RegularClearButton.setBounds(734,480,120,32);
```

```
//---------set bounds for dropout student-------------//
Droplabel.setBounds(0, 0, 1450, 200);

EnrollmentIDLabel2.setBounds(50,58,89,20);
EnrollmentIDtf2.setBounds(196,53,189,30);

StudentNameLabel2.setBounds(50,137,103,20);
StudentNametf2.setBounds(196,132,189,30);

DOBLabel2.setBounds(542,197,97,20);
DOBYearcb2.setBounds(697,191,75,28);
DOBMonthcb2.setBounds(786,191,79,28);
DOBDaycb2.setBounds(879,190,72,28);

DOEnrollmentLabel2.setBounds(50,200,127,20);
DOEnrollmentYearcb2.setBounds(196,192,75,28);
DOEnrollmentMonthcb2.setBounds(283,191,79,28);
DOEnrollmentDaycb2.setBounds(374,191,72,28);

CourseNameLabel2.setBounds(50,280,90,20);
CourseNametf2.setBounds(196,275,189,30);

CourseDurationLabel2.setBounds(542,276,117,20);
```

```
CourseDurationtf2.setBounds(697,268,160,32);

TutionFeeLabel2.setBounds(50,338,70,20);
TutionFeetf2.setBounds(196,333,189,30);

RemainingAmountLabel2.setBounds(50,390,122,20);
RemainingAmounttf2.setBounds(196,380,132,30);
PayButton.setBounds(342,380,65,30);


NumOfRemainingModulesLabel2.setBounds(541,338,128,20);
NumOfRemainingModulestf2.setBounds(697,333,160,30);

NumOfMonthsAttendedLabel2.setBounds(50,442,113,20);
NumOfMonthsAttendedtf2.setBounds(196,437,189,30);

DODropoutLabel2.setBounds(541,390,120,20);
DODropoutYearcb2.setBounds(697,380,75,28);
DODropoutMonthcb2.setBounds(782,380,79,28);
DODropoutDaycb2.setBounds(871,380,72,28);

DropoutHomeButton.setBounds(30,520,90,20);          //Home
RemoveStudentButton.setBounds(147,490,168,32);
DropoutAddstdButton.setBounds(345,490,168,32);
```

```
DropoutDisplayButton.setBounds(590,490,120,32);     //Display
DropoutClearButton.setBounds(732,490,120,32);

//-----------------adding the components--------------//
MainMenuFrame1.add(RegularButton);
MainMenuFrame1.add(DropoutButton);

//-----------------adding the components for regular students--------------//
RegularFrame1.add(Reglabel);

RegularFrame1.add(StudentNameLabel1);
RegularFrame1.add(StudentNametf1);

RegularFrame1.add(DOBLabel1);
RegularFrame1.add(DOBYearcb1);
RegularFrame1.add(DOBMonthcb1);
RegularFrame1.add(DOBDaycb1);

RegularFrame1.add(EnrollmentIDLabel1);
RegularFrame1.add(EnrollmentIDtf1);

RegularFrame1.add(DOEnrollmentLabel1);
RegularFrame1.add(DOEnrollmentYearcb1);
RegularFrame1.add(DOEnrollmentMonthcb1);
```

```
RegularFrame1.add(DOEnrollmentDaycb1);

RegularFrame1.add(CourseNameLabel1);
RegularFrame1.add(CourseNametf1);

RegularFrame1.add(CourseDurationLabel1);
RegularFrame1.add(CourseDurationtf1);

RegularFrame1.add(TutionFeeLabel1);
RegularFrame1.add(TutionFeetf1);

RegularFrame1.add(NumOfModulesLabel1);
RegularFrame1.add(NumOfModulestf1);

RegularFrame1.add(NumOfCredithrsLabel1);
RegularFrame1.add(NumOfCredithrstf1);

RegularFrame1.add(NumOfDaysPresentLabel1);
RegularFrame1.add(NumOfDaysPresenttf1);
RegularFrame1.add(PresentPercentageButton);

RegularFrame1.add(RegularDisplayButton);
RegularFrame1.add(RegularHomeButton);
RegularFrame1.add(GrantCertificateButton);
```

```
RegularFrame1.add(RegularAddstdButton);
RegularFrame1.add(RegularClearButton);

//-----------------adding the components for dropout students--------------//
DropoutFrame1.add(Droplabel);

DropoutFrame1.add(StudentNameLabel2);
DropoutFrame1.add(StudentNametf2);

DropoutFrame1.add(DOBLabel2);
DropoutFrame1.add(DOBYearcb2);
DropoutFrame1.add(DOBMonthcb2);
DropoutFrame1.add(DOBDaycb2);

DropoutFrame1.add(EnrollmentIDLabel2);
DropoutFrame1.add(EnrollmentIDtf2);

DropoutFrame1.add(DOEnrollmentLabel2);
DropoutFrame1.add(DOEnrollmentYearcb2);
DropoutFrame1.add(DOEnrollmentMonthcb2);
DropoutFrame1.add(DOEnrollmentDaycb2);

DropoutFrame1.add(CourseNameLabel2);
DropoutFrame1.add(CourseNametf2);
```

```
DropoutFrame1.add(CourseDurationLabel2);
DropoutFrame1.add(CourseDurationtf2);

DropoutFrame1.add(TutionFeeLabel2);
DropoutFrame1.add(TutionFeetf2);

DropoutFrame1.add(NumOfRemainingModulesLabel2);
DropoutFrame1.add(NumOfRemainingModulestf2);

DropoutFrame1.add(NumOfMonthsAttendedLabel2);
DropoutFrame1.add(NumOfMonthsAttendedtf2);

DropoutFrame1.add(RemainingAmountLabel2);
DropoutFrame1.add(RemainingAmounttf2);
DropoutFrame1.add(PayButton);

DropoutFrame1.add(DODropoutLabel2);
DropoutFrame1.add(DODropoutYearcb2);
DropoutFrame1.add(DODropoutMonthcb2);
DropoutFrame1.add(DODropoutDaycb2);

DropoutFrame1.add(DropoutDisplayButton);
DropoutFrame1.add(DropoutHomeButton);
```

```
DropoutFrame1.add(RemoveStudentButton);
DropoutFrame1.add(DropoutAddstdButton);
DropoutFrame1.add(DropoutClearButton);

//----------adding or registering buttons to the required listerner interface
RegularButton.addActionListener(this);
DropoutButton.addActionListener(this);


//----------adding or registering buttons to the required listerner interface of regular and dropout class
PresentPercentageButton.addActionListener(this);
PayButton.addActionListener(this);

RegularHomeButton.addActionListener(this);
DropoutHomeButton.addActionListener(this);

RegularDisplayButton.addActionListener(this);
DropoutDisplayButton.addActionListener(this);

GrantCertificateButton.addActionListener(this);
RemoveStudentButton.addActionListener(this);

RegularAddstdButton.addActionListener(this);
DropoutAddstdButton.addActionListener(this);
```

```
        RegularClearButton.addActionListener(this);
        DropoutClearButton.addActionListener(this);



        MainMenuFrame1.setLayout(null);
        MainMenuFrame1.add(Mainlabel);
        MainMenuFrame1.setSize(700, 700);
        MainMenuFrame1.setVisible(true);
        MainMenuFrame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        MainMenuFrame1.setLocationRelativeTo(null);



        RegularFrame1.setLayout(null);
        RegularFrame1.setSize(1000,600);
        RegularFrame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        RegularFrame1.setLocationRelativeTo(null);

        DropoutFrame1.setLayout(null);
        DropoutFrame1.setSize(1000,600);
        DropoutFrame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        DropoutFrame1.setLocationRelativeTo(null);

    }
```

```
    //implement the method
    //action listener

    @Override
    public void actionPerformed(ActionEvent event){
        //logic of button functionality

        if (event.getSource() == RegularButton){
            RegularFrame1.setVisible(true);
            MainMenuFrame1.setVisible(false);

        }else if(event.getSource() == DropoutButton){
            DropoutFrame1.setVisible(true);
            MainMenuFrame1.setVisible(false);

        }else if(event.getSource() == RegularHomeButton){
            MainMenuFrame1.setVisible(true);
            RegularFrame1.setVisible(false);

        }else if(event.getSource() == DropoutHomeButton){
            MainMenuFrame1.setVisible(true);
            DropoutFrame1.setVisible(false);

        }
```

```
        //Add Grant certificate, add student , display and clear
        //regular student functions


        //adding regular student and getting all the parameters to make constructors
        //add object of regular class and creat object using constructor
        //add to the arraylist
        //check if the buttons input and output are valid or not
        //courseName, enrollmentID, dateOfEnrollment
        // Assuming your Regular class has a grantCertificate method



    else if(event.getSource() == GrantCertificateButton) {
        if(EnrollmentIDtf1.getText().isEmpty() || CourseNametf1.getText().isEmpty()){
            JOptionPane.showMessageDialog(RegularFrame1, "Empty textFields.Please input
properly!",
            "Error", JOptionPane.ERROR_MESSAGE);
        }else{
            try{
            // Retrieving input values from GUI fields
            String courseName = CourseNametf1.getText();
            //conveting integer into string and retriving values
            int enrollmentID = Integer.parseInt(EnrollmentIDtf1.getText());
            //using get selected item for combo box
            String DOEnrollmentYearcb2 = DOEnrollmentYearcb1.getSelectedItem().toString();
```

```
        String DOEnrollmentMonthcb2 =
DOEnrollmentMonthcb1.getSelectedItem().toString();

        String DOEnrollmentDaycb2 = DOEnrollmentDaycb1.getSelectedItem().toString();

        String dateOfEnrollment = DOEnrollmentYearcb2 + "/" +DOEnrollmentMonthcb2 +
"/" + DOEnrollmentDaycb2;

        // ypu can reolace "/"(displays 2004/10/24) with "-"(displays 2004-10-24)

        //debugging

        /*System.out.println("Button clicked: Grant Certificate");

      System.out.println("Course Name: " + courseName);

      System.out.println("Enrollment ID: " + enrollmentID);

      System.out.println("Date of Enrollment: " + dateOfEnrollment);*/

      boolean check = true;

      // Iterate through students and call grantCertificate method

        for (Student regstudent : students) {

          if(regstudent instanceof Regular){

          Regular regularStudent = (Regular) regstudent;

            if(regstudent.getEnrollmentID() == enrollmentID){

            regularStudent.grantCertificate(courseName, enrollmentID, dateOfEnrollment);

            JOptionPane.showMessageDialog(null,"Student with enrollment id " +
enrollmentID + " has been Granted Certificate!");

          }else{

          JOptionPane.showMessageDialog(RegularFrame1, "Student with enrollment id "
+ enrollmentID + " not found. Please Try Again" ,

          "Error", JOptionPane.ERROR_MESSAGE);

            }
```

```
            }
        }
    }catch(Exception exp){
    // Display an error message using JOptionPane if there are any number format
exceptiom
    JOptionPane.showMessageDialog(null, "An error occurred: " + exp.getMessage(),
    "Error", JOptionPane.ERROR_MESSAGE);


        }
    }
}
//checking if arraylist is empty
// add student button
/*String dateOfEnrollment, int enrollmentID, String courseName, String studentName,
String dateOfBirth, int courseDuration, int tutionFee,
int numOfModules,  int numOfCreditHours, int daysPresent*/
else if (event.getSource() == RegularAddstdButton) {
    if (StudentNametf1.getText().isEmpty() ||
    EnrollmentIDtf1.getText().isEmpty() ||
    CourseNametf1.getText().isEmpty() ||
    TutionFeetf1.getText().isEmpty() ||
    NumOfModulestf1.getText().isEmpty() ||
    NumOfCredithrstf1.getText().isEmpty() ||
    NumOfDaysPresenttf1.getText().isEmpty() ||
    CourseDurationtf1.getText().isEmpty()) {
```

```
JOptionPane.showMessageDialog(RegularFrame1, "Error: Please fill in all details.");
} else {
    try {
    int enrollmentID = Integer.parseInt(EnrollmentIDtf1.getText());
    int tutionFee = Integer.parseInt(TutionFeetf1.getText());
    int numOfModules = Integer.parseInt(NumOfModulestf1.getText());
    int numOfCreditHours = Integer.parseInt(NumOfCredithrstf1.getText());
    int daysPresent = Integer.parseInt(NumOfDaysPresenttf1.getText());
    int courseDuration = Integer.parseInt(CourseDurationtf1.getText());

    String enrollmentIDString = EnrollmentIDtf1.getText();
    String studentName = StudentNametf1.getText();
    String courseName = CourseNametf1.getText();
    String tutionFeeString = TutionFeetf1.getText();
    String numOfModulesString = NumOfModulestf1.getText();
    String numOfCreditHrsString = NumOfCredithrstf1.getText();
    String numOfDaysPresentString = NumOfDaysPresenttf1.getText();
    String courseDurationString = CourseDurationtf1.getText();

    String DOBYearcb2 = DOBYearcb1.getSelectedItem().toString();
    String DOBMonthcb2 = DOBMonthcb1.getSelectedItem().toString();
    String DOBDaycb2 = DOBDaycb1.getSelectedItem().toString();
    String dateOfBirth = DOBYearcb2 + "/" +DOBMonthcb2 + "/" + DOBDaycb2;
```

```
String DOEnrollmentYearcb2 = DOEnrollmentYearcb1.getSelectedItem().toString();
String DOEnrollmentMonthcb2 =
DOEnrollmentMonthcb1.getSelectedItem().toString();
String DOEnrollmentDaycb2 = DOEnrollmentDaycb1.getSelectedItem().toString();
String dateOfEnrollment = DOEnrollmentYearcb2 + "/" +DOEnrollmentMonthcb2 +
"/" + DOEnrollmentDaycb2;


//checking if the arraylist is emoty
//if empty then check for duplication of enrollment id
//same enrollment id cannot be used since it overlaps
if(students.isEmpty()){
// Creating an instance of the Regular class
//parameters are passed down same as from Regular class
//the order of this parameters is to be same as regular class
Regular addStudent = new Regular(dateOfEnrollment, enrollmentID, courseName,
studentName, dateOfBirth, courseDuration,
                    tutionFee, numOfModules, numOfCreditHours, daysPresent);
students.add(addStudent);
JOptionPane.showMessageDialog(RegularFrame1, "Array list has been added");
}else{
// Checking for duplication
// declaring isDuplicate as a variable to confirm duplication throught boolean data
type
//if duplication true cannot add student
```

```
            int EnrollmentIDnew =  enrollmentID;
            boolean isDuplicate = false;
            for (Student existingStudent : students) {
                if (existingStudent instanceof Regular) {
                Regular regstd = (Regular) existingStudent;
                    if(regstd.getEnrollmentID() == EnrollmentIDnew){
                    isDuplicate = true;
                    }
                }
            }
            if (isDuplicate) {
                JOptionPane.showMessageDialog(RegularFrame1, "This enrollment ID already
exists",
                 "Alert", JOptionPane.WARNING_MESSAGE);
            } else {
            Regular addStudent = new Regular(dateOfEnrollment, enrollmentID, courseName,
studentName, dateOfBirth, courseDuration,
                                tutionFee, numOfModules, numOfCreditHours, daysPresent);
            students.add(addStudent);
            JOptionPane.showMessageDialog(RegularFrame1, "Student has been successfully
added.");
                    }
            }
        } catch (NumberFormatException nfe) {
            System.out.println("NumberFormatException: " + nfe.getMessage());
```

```
            JOptionPane.showMessageDialog(RegularFrame1, "Invalid input. Please check the entered values.");
            }
        }
    }



    //display button
    //displays the input fields by the user
    //calling the display method from regular class
    else if(event.getSource() == RegularDisplayButton){
        if(students.isEmpty()){
            JOptionPane.showMessageDialog(null,"Error: No Information to Display");
            }else{
            //loop through arraylist
            //loop continues uintil its from the same class
                for (Student student: students){
                //checking if the student info belongs to regular class
                //if not error message
                    if(student instanceof Regular){
                        //downcasting student obj into regular
                        //parent to chikd class
                        Regular RegObj = (Regular) student;
                        System.out.println("The information of Regular students are given: \n");
                        RegObj.display();
```

```
                }else{
                    JOptionPane.showMessageDialog(RegularFrame1, "Error: No information
on Regular Student to display");
                }


            }
        }
    }


    //present percentage
    //calling the present percentage method from regular class
    //calculates the present percentage value from the class
    //grade given in char data type within the dialog box
    else if (event.getSource() == PresentPercentageButton) {
        if (CourseNametf1.getText().isEmpty() ||
            EnrollmentIDtf1.getText().isEmpty()) {
            JOptionPane.showMessageDialog(RegularFrame1, "Error: Please Add the
details.");
        } else {
            try {
                int enrollmentID = Integer.parseInt(EnrollmentIDtf1.getText());
                int tutionFee = Integer.parseInt(TutionFeetf1.getText());
                int numOfModules = Integer.parseInt(NumOfModulestf1.getText());
                int numOfCreditHours = Integer.parseInt(NumOfCredithrstf1.getText());
                int daysPresent = Integer.parseInt(NumOfDaysPresenttf1.getText());
```

```
int courseDuration = Integer.parseInt(CourseDurationtf1.getText());


String enrollmentIDString = EnrollmentIDtf1.getText();
String studentName = StudentNametf1.getText();
String courseName = CourseNametf1.getText();
String tutionFeeString = TutionFeetf1.getText();
String numOfModulesString = NumOfModulestf1.getText();
String numOfCreditHrsString = NumOfCredithrstf1.getText();
String numOfDaysPresentString = NumOfDaysPresenttf1.getText();
String courseDurationString = CourseDurationtf1.getText();


String DOBYearcb2 = DOBYearcb1.getSelectedItem().toString();
String DOBMonthcb2 = DOBMonthcb1.getSelectedItem().toString();
String DOBDaycb2 = DOBDaycb1.getSelectedItem().toString();
String dateOfBirth = DOBYearcb2 + "/" +DOBMonthcb2 + "/" + DOBDaycb2;


String DOEnrollmentYearcb2 =
DOEnrollmentYearcb1.getSelectedItem().toString();
String DOEnrollmentMonthcb2 =
DOEnrollmentMonthcb1.getSelectedItem().toString();
String DOEnrollmentDaycb2 =
DOEnrollmentDaycb1.getSelectedItem().toString();
String dateOfEnrollment = DOEnrollmentYearcb2 + "/" +DOEnrollmentMonthcb2
+ "/" + DOEnrollmentDaycb2;
//creating an instance of the studentGui class and calling the present percentage
method
```

```
//calculates the present percentage and returns value in char data type within the
dialog box
Regular student = new Regular(dateOfEnrollment, enrollmentID, courseName,
studentName,dateOfBirth,courseDuration,
        tutionFee ,numOfModules,  numOfCreditHours, daysPresent);
// Calculate present percentage and grade
//if else loop returns grade in char similar to what it did in regular method
char grade;
if (daysPresent > courseDuration) {
    JOptionPane.showMessageDialog(null,
    "Error: Number of days present cannot be greater than course duration.",
    "Error",
    JOptionPane.ERROR_MESSAGE);
    return;
    } else {
    double percentage = (double) daysPresent / courseDuration * 100;
    if (percentage >= 80) {
        grade = 'A';
        } else if (percentage >= 60) {
        grade = 'B';
        } else if (percentage >= 40) {
        grade = 'C';
        } else if (percentage >= 20) {
        grade = 'D';
        } else {
```

```
                grade = 'E';
            }
            // Display result in a message dialog
            JOptionPane.showMessageDialog(RegularFrame1,
            "Present Percentage: " + grade,
            "Result",
            JOptionPane.INFORMATION_MESSAGE);
            }
        } catch (NumberFormatException ex) {
            //if number format exception found error message is displayed
            JOptionPane.showMessageDialog(RegularFrame1,
            "Invalid input for days present. Please enter a valid number.",
            "Error",
            JOptionPane.ERROR_MESSAGE);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(RegularFrame1,
            "An error occurred: " + ex.getMessage(),
            "Error",
            JOptionPane.ERROR_MESSAGE);
        }
    }
}

//clear button clears all the textfields and users can reinput the values in the form
```

```
    // dateOfEnrollment,  enrollmentID,  courseName,  studentName,  dateOfBirth,
courseDuration,  tutionFee,numOfModules,   numOfCreditHours,  daysPresent
    else if (event.getSource() == RegularClearButton){
        StudentNametf1.setText("");
        EnrollmentIDtf1.setText("");
        CourseNametf1.setText("");
        TutionFeetf1.setText("");
        NumOfModulestf1.setText("");
        NumOfCredithrstf1.setText("");
        NumOfDaysPresenttf1.setText("");
        CourseDurationtf1.setText("");
        DOEnrollmentYearcb1.setSelectedItem("1999");//set back to 1999
        DOEnrollmentMonthcb1.setSelectedItem("Jan");
        DOEnrollmentDaycb1.setSelectedItem("1");
        DOBYearcb1.setSelectedItem("1999");
        DOBYearcb1.setSelectedItem("Jan");
        DOBYearcb1.setSelectedItem("1");
    }



    //pay button for the remaining amount
    //calling billspayable method from dropout class
    //calculaates the remaining amount
    //if pay bills 0 then student removed
    //bils payable button
```

```
    else if (event.getSource() == PayButton) {
            try {
                // Get the enrollment ID entered by the user
                int enrollmentID = Integer.parseInt(EnrollmentIDtf2.getText());
                int numOfMonthsAttended =
Integer.parseInt(NumOfMonthsAttendedtf2.getText());
                String enrollmentIDString = EnrollmentIDtf2.getText();
                String numOfMonthsAttendedString = NumOfMonthsAttendedtf2.getText();

                // Iterate through the list of students to find a match
                for (Student student : students) {
                    if (student.getEnrollmentID() == enrollmentID) {
                        // Check if the student is a Dropout
                        if (student instanceof Dropout) {
                            // Downcast the student to a Dropout
                            Dropout dropout = (Dropout) student;

                            // Call the billsPayable method and update the display
                            dropout.billsPayable(dropout.getNumOfMonthsAttended());
                            JOptionPane.showMessageDialog(DropoutFrame1, "Bills paid
successfully for dropout student.");
                        } else {
                            // Display a message if the student is not a dropout
                            JOptionPane.showMessageDialog(DropoutFrame1, "Student is not a
dropout.");
```

```
                }
                break; // Exit the loop after finding the student
            }
        }
    } catch (NumberFormatException ex) {
        // incase for exception handling
        JOptionPane.showMessageDialog(DropoutFrame1,
            "Invalid enrollment ID format. Please enter a valid number.",
            "Error",
            JOptionPane.ERROR_MESSAGE);
    }

} else if (event.getSource() == RemoveStudentButton) {
    try {
    // Get the enrollment ID entered by the user
    int enrollmentID = Integer.parseInt(EnrollmentIDtf2.getText());
    String enrollmentIDString = EnrollmentIDtf2.getText();

    // Iterate through the list of students to find a match
    for (Student student : students) {
        if (student.getEnrollmentID() == enrollmentID) {
            // Check if the student is a Dropout
            if (student instanceof Dropout) {
            // Downcast the student to a Dropout
```

```
            Dropout dropout = (Dropout) student;
            //checking if the bills are all cleared and doinf the same as the dropout class
            if (dropout.getHasPaid() ){
            // Call the removeStudent method and update the display
            dropout.removeStudent();
            students.remove(dropout);
            JOptionPane.showMessageDialog(DropoutFrame1, "Dropout student removed
successfully.");


            } else {
            // Display a message if the student is not a dropout
            JOptionPane.showMessageDialog(DropoutFrame1, "Student is not a dropout.");
                }
            break; // Exit the loop after finding the student
            }
        }
    }
    } catch (NumberFormatException ex) {
        // Handle invalid enrollment ID format
        JOptionPane.showMessageDialog(DropoutFrame1,
                "Invalid enrollment ID format. Please enter a valid number.",
                "Error",
                JOptionPane.ERROR_MESSAGE);
    }
  }
```

```
    //adds new dropout student
    //first checks if arraylist is empty then duplicate enrollment id
    //same as for the regular add student button
    //add button for dropout class
    else if(event.getSource() == DropoutAddstdButton ){
        if(StudentNametf2.getText().isEmpty() ||
        EnrollmentIDtf2.getText().isEmpty() ||
        TutionFeetf2.getText().isEmpty() ||
        NumOfRemainingModulestf2.getText().isEmpty() ||
        NumOfMonthsAttendedtf2.getText().isEmpty() ||
        CourseDurationtf2.getText().isEmpty() ||
        RemainingAmounttf2.getText().isEmpty()) {
        JOptionPane.showMessageDialog(DropoutFrame1, "Error: Please fill in all details.");
        }else {
            try{
            //calling parameters from dropout class
            //String dateOfBirth, int tutionFee, int courseDuration, String studentName,  int
numOfRemainingModules, int numOfMonthsAttended,String dateOfDropout
            /*setter/mutator method
              super.setCourseName("Java");
              super.setEnrollmentID(4);
              super.setDateOfEnrollment("2003-09-21");*/
            int enrollmentID = Integer.parseInt(EnrollmentIDtf2.getText());
            int tutionFee = Integer.parseInt(TutionFeetf2.getText());
```

```
        int numOfRemainingModules =
Integer.parseInt(NumOfRemainingModulestf2.getText());
        int numOfMonthsAttended = Integer.parseInt(NumOfMonthsAttendedtf2.getText());
        int courseDuration = Integer.parseInt(CourseDurationtf2.getText());
        int remainingAmount = Integer.parseInt(RemainingAmounttf2.getText());

        String enrollmentIDString = EnrollmentIDtf2.getText();
        String studentName = StudentNametf2.getText();

        String tutionFeeString = TutionFeetf2.getText();
        String numOfRemainingModulesString = NumOfRemainingModulestf2.getText();
        String numOfMonthsAttendedString = NumOfMonthsAttendedtf2.getText();
        String courseDurationString = CourseDurationtf2.getText();
        String remainingAmountString = RemainingAmounttf2.getText();



        String DOBYearcb3 = DOBYearcb2.getSelectedItem().toString();
        String DOBMonthcb3 = DOBMonthcb2.getSelectedItem().toString();
        String DOBDaycb3 = DOBDaycb2.getSelectedItem().toString();
        String dateOfBirth = DOBYearcb3 + "/" +DOBMonthcb3 + "/" + DOBDaycb3;

        String DOEnrollmentYearcb3 = DOEnrollmentYearcb2.getSelectedItem().toString();
        String DOEnrollmentMonthcb3 =
DOEnrollmentMonthcb2.getSelectedItem().toString();
        String DOEnrollmentDaycb3 = DOEnrollmentDaycb2.getSelectedItem().toString();
```

```
        String dateOfEnrollment = DOEnrollmentYearcb3 + "/" +DOEnrollmentMonthcb3 +
"/" + DOEnrollmentDaycb3;


        String DODropoutYearcb3 = DODropoutYearcb2.getSelectedItem().toString();

        String DODropoutMonthcb3 = DODropoutMonthcb2.getSelectedItem().toString();

        String DODropoutDaycb3 = DODropoutDaycb2.getSelectedItem().toString();

        String dateOfDropout = DODropoutYearcb3 + "/" +DODropoutMonthcb3 + "/" +
DODropoutDaycb3;



        //checking if the arraylist is emoty

        if(students.isEmpty()){

        // Creating an instance of the Regular class

        Dropout addDStudent = new Dropout(dateOfBirth, tutionFee, courseDuration,
studentName,

                numOfRemainingModules, numOfMonthsAttended,dateOfDropout);


        students.add(addDStudent);

        JOptionPane.showMessageDialog(DropoutFrame1, "array list has been updated");

        }else{

        // Checking for duplication

        int newEnrollmentID = enrollmentID;

        boolean isDuplicate = false;

        for (Student DropoutStudent : students) {

            if (DropoutStudent instanceof Dropout) {
```

```
                Dropout dropstd = (Dropout) DropoutStudent;
                if(dropstd.getEnrollmentID() ==newEnrollmentID){
                isDuplicate = true;
                }
            }
        }if (isDuplicate) {
            JOptionPane.showMessageDialog(DropoutFrame1, "This enrollment ID already
exists");
        } else {
        Dropout addDStudent = new Dropout(dateOfBirth, tutionFee, courseDuration,
studentName,
                numOfRemainingModules, numOfMonthsAttended,dateOfDropout);


        students.add(addDStudent);
        JOptionPane.showMessageDialog(DropoutFrame1, "Student has been succesfully
added.");
        }
        }
    }catch (NumberFormatException numfe){
         System.out.println("NumberFormatException: " + numfe.getMessage());
        JOptionPane.showMessageDialog(DropoutFrame1, "Invalid input. Please check the
entered values.");
        }
    }
  }
```

```
    //display button
    //displays all the values input but the users
    else if(event.getSource() == DropoutDisplayButton ){
        if(students.isEmpty()){
        JOptionPane.showMessageDialog(null, "Error: No information to display");
        } else{
        //loop throught the arraylist
            for(Student student: students){
            //checking if the students info belongs to the class dropout
                if(student instanceof Dropout){
                //downcasting student obj into dropoit
                //parent to child class
                Dropout DropObj = (Dropout) student;
                System.out.println("the information for Dropout class is give as: \n");
                DropObj.display();
                }else{
                    JOptionPane.showMessageDialog(DropoutFrame1, "Error: No information on
Regular Student to display");
                }
            }
        }
    }

    //clear button
```

```
    //emptys all the textfield of the dropout form
    else if(event.getSource() == DropoutClearButton){
        EnrollmentIDtf2.setText("");
        StudentNametf2.setText("");
        CourseNametf2.setText("");
        CourseDurationtf2.setText("");
        TutionFeetf2.setText("");
        RemainingAmounttf2.setText("");
        NumOfRemainingModulestf2.setText("");
        NumOfMonthsAttendedtf2.setText("");
        DOEnrollmentYearcb2.setSelectedItem("1999");// Set back to 1999
        DOEnrollmentMonthcb2.setSelectedItem("Jan");// Set back to jan
        DOEnrollmentDaycb2.setSelectedItem("1");// Set back to 1
        DOBYearcb2.setSelectedItem("1999");// Set back to 1999
        DOBMonthcb2.setSelectedItem("Jan");// Set back to jan
        DOBDaycb2.setSelectedItem("1");// Set back to 1
        DODropoutYearcb2.setSelectedItem("1999");// Set back to 1999
        DODropoutMonthcb2.setSelectedItem("Jan");// Set back to jan
        DODropoutDaycb2.setSelectedItem("1");// Set back to 1
    }
}
    public static void main(String[]args){
         new StudentGUI();
    }
```

}