CS4001NI Programming

30% Individual Coursework - 2

2023-24 Spring

Student Name: Shamel Rai

London Met ID: 22085617

College ID: NP01CP4S230135

Group: C21

Assignment Due Date: Friday, August 11, 2023

Assignment Submission Date: Thursday, August 10, 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 1
BlueJ 2
Microsoft Word 2
Draw.io 3
Class Diagram 3
Pseudocode 5
StudentGUI 5
Method 23
StudentGUI 23
Testing 29
Test One 29
Test Two 30
Test Three 36
Error detection and error correction 38
Syntax Error 38
Runtime Error 38
Logical Error 40
Conclusion 41
References 43
Appendix 45

Table of Figure

Figure 1:Class Diagram 4
Figure 2: Compiling StudentGUI 29
Figure 3: Running StudentGUI 30
Figure 4: Add Regular 33
Figure 5: Add Dropout 33
Figure 6: Calculate Percentage 34
Figure 7:Grant Certificate 34
Figure 8: Pay Bills 35
Figure 9:Remove Student 35
Figure 10:Unsuitable value 37
Figure 11: Unsuitable Value 37
Figure 12: Syntax Error 38
Figure 13: Syntax Error Correction 38
Figure 14:Runtime Error 39
Figure 15: Runtime Error 39
Figure 16: Logical Error 40
Figure 17: Logical Error 40

Table of Table

Table 1:Method Description 28
Table 2:Test One 29
Table 3:Test Two 32
Table 4:Test Three..... 36

Introduction

The development of the GUI will transform the student management system from a functional but uninteresting interface to an engaging platform that supports a pleasant user experience. Its aesthetically appealing design will inspire workers and administrators to interact with the system more frequently, resulting in improved data management and decision-making.

The capacity of the GUI to deliver real-time insights and analytics is one of its primary benefits. Administrators now have access to thorough information and visualizations of student performance, attendance patterns, and scholarship allocations. This enables educational institutions to discover areas for improvement and execute focused interventions to help their students succeed academically.

Furthermore, the GUI will make it possible to communicate between various departments of the institution on a seamless basis. Faculty members are able to easily report progress with their students, while the Finance Departments can more effectively monitor fees and payments. A coherent and connected curriculum is ensured by the collaborative nature of the GUI.

The Guidance User Interface will prioritize accessibility, taking into account the variety of users who interact with this system. It shall ensure that the system is accessible for all students, personnel and administrators irrespective of their physical or mental disability in conformity with industry standards on web accessibility. The commitment of the institution to provide equitable opportunities for all is further strengthened by such inclusion.

Apart from its functionality, this GUI will also be tested extensively to make sure it is usable and user friendly. In improving the interface and integrating user preferences, it will be beneficial for interested parties to share their views so as to create a more intuitive and personalized experience.

As we work on this project, it is evident that the success of GUI depends upon a concerted effort by developers, education professionals and end users. We seek to

develop a global user interface that will meet the needs and objectives of an educational institution, by integrating technological knowhow with useful insight from teachers and administrators.

Lastly, a major step forward in boosting overall efficiency, accessibility and usability of the system has been achieved with the development of an interface for managing student accounts. We seek to provide education institutions with a robust instrument for supporting and nurturing students' learning journeys through the blending of functionality, aesthetics, real-time insight as well as inclusiveness.

BlueJ

BlueJ is a superb ide that sticks out in the minds of both aspiring programmers and professional developers. Its simple interface and mild learning curve make it an excellent alternative for individuals just beginning to learn how to write. BlueJ is a beautiful blend of simplicity and utility, encouraging exploration and nurturing curiosity; it's like a patient tutor guiding pupils into the intriguing world of Java programming. The visual tools, simple connection with JDK, and ability to interact directly with objects make it simpler to build a more natural programming experience that instills enthusiasm and delight in every successful code line. As we look at the depths of BlueJ, it becomes obvious that this was built with passion for education and nurtured by a generation of programmers eager to transform their ideas into reality.

Microsoft Word

A popular word-processing software, Word can be used to create documents. For example, brochures, letters, in addition to college students' homework assignments may be created with this software. It is simple to create professional-looking files with Microsoft Word thanks to its diverse tools and features. There are many purposes for which Word may be used, from basic textual content formatting to advanced capabilities like mail merge. There are many motives because it is a move-to choose for plenty users, together with its smooth-to-use interface and extensive variety of templates and styles.

Draw.io

Draw.io is a fantastic web diagramming tool, allowing users to create numerous diagrams and flowcharts with ease. Even for people who do not have expertise in designing, the simple interface makes this a useful tool. With a wide range of shapes, icons and themes to choose from, creating professional looking images is easy and pleasant. Whether you are working on a technical project, thinking of ideas or plotting procedures, Draw.io is here to solve your problem. It's perfect for teamwork, providing a seamless collaboration that allows several users to collaborate in real time. The best thing about this is that it's fully Web based, so you don't need to install any programs. In many organizations and individuals, Draw.io has clearly become the most important tool that they use to develop diagrams making it a satisfying and efficient process.

Class Diagram

Classes in Unified Modelling Language (UML) can be seen as a diagram illustrating their relationships and dependencies. An item in a application (or the unit of code that represents it) is known as a category, because classes outline methods and variables. The use of sophistication diagrams is vital for all kinds of item-oriented programming (OOP). OOP modelling paradigms have developed over numerous years, bringing new refinements to the idea.

An arrangement of training in a class diagram includes agencies having a not unusual characteristic. Classes are represented as containers on a class diagram like a flowchart, and every container has three rectangles internal it. A magnificence is described by its name, attributes, techniques, and operations. The top rectangle consists of the class call; the centre rectangle consists of the magnificence attributes; and the lower rectangle carries the magnificence techniques. Boxes are linked with the aid of strains that can be arrowed at one or both ends. Those lines imply the relationships between the instructions, additionally referred to as institutions.

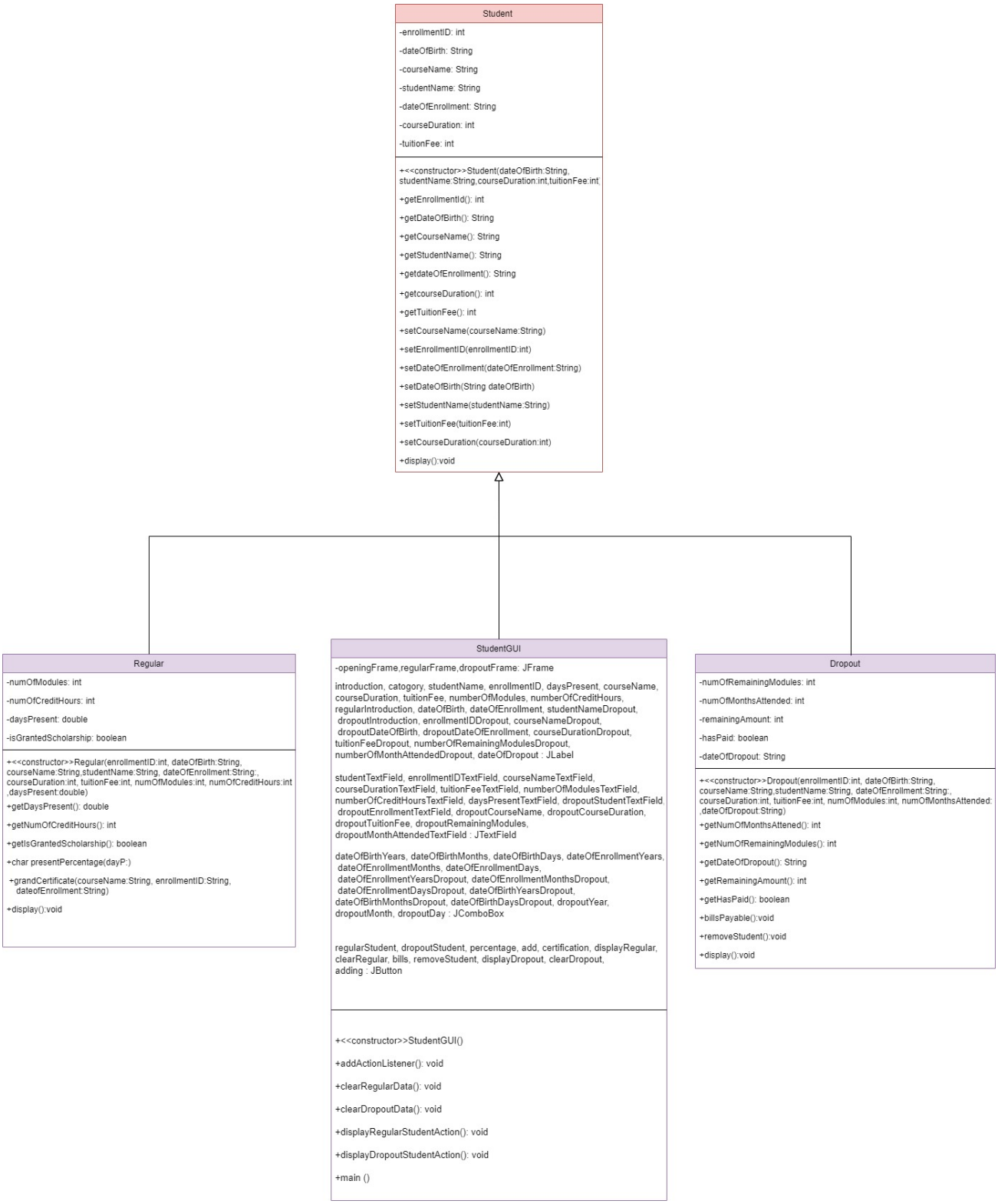

Figure 1:Class Diagram

Pseudocode

The pseudocode is a formal description expressed in natural language rather than a programming language, of what a computer program or algorithm must do. The development of a program may sometimes include pseudocode as a detailed step. (TechTarget,1999).

StudentGUI

CLASS StudentGUI:

DO

DECLARE ArrayList allStudents

DECLARE JFrame openningFrame, regularFrame, dropoutFrame

DECLARE JLabel introduction, catogory, studentName, enrollmentID, daysPresent, courseName, courseDuration, tuitionFee, numberOfModules, numberOfCreditHours, regularIntroduction, dateOfBirth, dateOfEnrollment, studentNameDropout, dropoutIntroduction, enrollmentIDDropout, courseNameDropout, dropoutDateOfBirth, dropoutDateOfEnrollment, courseDurationDropout, tuitionFeeDropout, numberOfRemainingModulesDropout, numberOfMonthAttendedDropout, dateOfDropout

DECLARE JTextField studentTextField, enrollmentIDTextField, courseNameTextField, courseDurationTextField, tuitionFeeTextField, numberOfModulesTextField, numberOfCreditHoursTextField, daysPresentTextField, dropoutStudentTextField, dropoutEnrollmentTextField, dropoutCourseName, dropoutCourseDuration, dropoutTuitionFee, dropoutRemainingModules, dropoutMonthAttendedTextField

DECLARE JComboBox dateOfBirthYears, dateOfBirthMonths, dateOfBirthDays, dateOfEnrollmentYears, dateOfEnrollmentMonths, dateOfEnrollmentDays, dateOfEnrollmentYearsDropout, dateOfEnrollmentMonthsDropout, dateOfEnrollmentDaysDropout, dateOfBirthYearsDropout,

dateOfBirthMonthsDropout, dateOfBirthDaysDropout, dropoutYear, dropoutMonth, dropoutDay

DECLARE JButton percentage, add, certification, displayRegular, clearRegular, bills, removeStudent, displayDropout, clearDropout, adding, regularStudent, dropoutStudent

CONSTRUCTOR StudentGUI():

DO

INITIALIZE allStudents as ArrayList

INITIALIZE openningFrame as new JFrame("Student Management System")

INITIALIZE introduction as new JLabel("Welcome to the Student Management System")

INITIALIZE catogory as new JLabel("Please Select A Catogry")

INITIALIZE regularStudent as new JButton("Regular Student")

INITIALIZE dropoutStudent as new JButton("Dropout Student")

SET introduction bounds, font, and foreground color

SET catogory bounds, font, and foreground color

SET regularStudent bounds, background color, font, and foreground color

SET dropoutStudent bounds, background color, font, and foreground color

ADD introduction, catogory, regularStudent, dropoutStudent to

openningFrame

SET background color of openningFrame

SET layout to null for openningFrame

SET resizable to false for openningFrame

SET default close operation to JFrame.EXIT_ON_CLOSE for openningFrame

SET size of openningFrame

SET visibility of openningFrame to true

SET location relative to null for openningFrame

INITIALIZE regularFrame as a new JFrame(“Regular Student”)

INITIALIZE regularIntroduction, studentName, enrollmentID, courseDuration, tuitionFee, numberOfModules, courseName, numberOfCreditHours, daysPresent, dateOfBirth, dateOfEnrollment as new JLabel

INITIALIZE studentTextField, enrollmentIDTextField, tuitionFeeTextField, numberOfModulesTextField, numberOfCreditHoursTextField, daysPresentTextField as JTextField

INITIALIZE dateOfBirthYears, dateOfBirthMonths, dateOfBirthDays, dateOfEnrollmentYears, dateOfEnrollmentMonths, dateOfEnrollmentDays as JComboBox<String>

INITIALIZE add, percentage, certification, displayRegular, clearRegular as JButton

ADD JLabel, JTextField, JComboBoxes and JButtons to the regularFrame

SET setbounds, fonts, and properties for JLables, JTextFields, and JComboBox inside regularFrame

SET setbounds, fonts, and properties for JButtons inside regularFrame

SET background color for the regularFrame

SET layout to null for the regularFrame

SET desire size for the regularFrame

SET resizable to false for the regularFrame

SET defaultCloseOperation to JFrame.EXIT_ON_CLOSE for regularFrame

INITIALIZE dropoutFrame as a new JFrame (“Dropout Student”)

INITIALIZE dropoutIntroduction, studentNameDropout, enrollmentIDDropoout, courseNameDropout, courseDurationDropout, tuitionFeeDropout, numberOfRemainingModulesDropout, numberOfMonthAttendedDropout, dropoutDateOfBirth, dropoutDateOfEnrollment, dateOfDropout as JLabel

INITIALIZE dropoutStudentTextField, dropoutEnrollmentTextField, dropoutCourseName, dropoutTuitionFee, dropoutRemainingModules, dropoutMonthAttendedTextField as JTextField

INITIALIZE dateOfEnrollmentYearsDropout, dateOfEnrollmentMonthsDropout, dateOfEnrollmentDaysDropout, dateOfBirthYearsDropout, dateOfBirthMonthsDropout, dateOfBirthDaysDropout, dropoutYears, dropoutMonths, dropoutDays as JComboBox<String>

INITIALIZE bills, removeStudent, displayDropout, clearDropout, adding as

JButton

ADD JLabel, JTextField, JComboBox, and JButton to the dropoutFrame

SET setbounds, fonts, and properties for JLables, JTextFields, and JComboBox inside dropoutFrame

SET setbounds, fonts, and properties for JButtons inside dropoutFrame

SET background color for the dropoutFrame

SET layout to null for the dropoutFrame

SET desire size for the dropoutFrame

SET resizable to false for the dropoutFrame

SET defaultCloseOperation to JFrame.EXIT_ON_CLOSE for dropoutFrame

CALL the addActionListner() method

END DO

CREATE a method addActionListeners() return type void

DO

CALL regularStudent and **ADD** actionListener with parameter ActionListener() method.

DO

CREATE a method actionPerformed **WITH** return type void and parameter ActionEvent

DO

SET openningFrameVisibiliy to **TRUE**

SET regularFrameVisibilty to **TRUE**

END DO

END DO

CALL dropoutStudent and **ADD** actionListener with parameter

ActionListener() method.

DO

CREATE a method actionPerformed **WITH** return type void and

Parameter ActionEvent

DO

SET openningFrameVisibility to **TRUE**

SET dropoutFrameVisibility to **TRUE**

END DO

END DO

CALL displayDropout and **ADD** actionListener with parameter

ActionListener() method

DO

CREATE a method actionPerformed **WITH** return type void and

parameterActionEvent

DO

CALL displayDropoutAction() method

END DO

END DO

CALL removeStudent and **ADD** actionListener with parameter

ActionListene()r method

DO

CREATE a method actionPerformed with return type void and parameter ActionEvent

DO

Get the text values from the corresponding text fields

SET dropoutStudentName **TO** dropoutStudentTextField.GetText()

SET dropoutEnrollmentID **TO** dropoutEnrollmentTextField.GetText()

SET dropoutCourseDurationText **TO** dropoutCourseDuration.GetText()

SET dropoutTuitionFeeText **TO** dropoutTuitionFee.GetText()

SET dropoutRemainingModulesText **TO** dropoutRemainingModules.GetText()

SET dropoutMonthAttendedText **TO** dropoutMonthAttendedTextField.GetText()

SET dropoutDateOfBirthYear **TO** (String) dateOfBirthYearsDropout.GetSelectedItem()

SET dropoutDateOfBirthMonth **TO** (String) dateOfBirthMonthsDropout.GetSelectedItem()

SET dropoutDateOfBirthDay **TO** (String) dateOfBirthDaysDropout.GetSelectedItem()

SET dateOfBirthCombined **TO** dropoutDateOfBirthYear, dropoutDateOfBirthMonth ,dropoutDateOfBirthDate

SET dropoutDateOfEnrollmentYear **TO** (String)

dateOfEnrollmentYearsDropout.GetSelectedItem()

SET dropoutDateOfEnrollmentMonth **TO** (String)

dateOfEnrollmentMonthsDropout.GetSelectedItem()

SET dropoutDateOfEnrollmentDays **TO** (String)

dateOfEnrollmentDaysDropout.GetSelectedItem()

SET dateOfEnrollmentCombined **TO** dropoutDateOfEnrollmentYear dropoutDateOfEnrollmentMonth,dropoutDateOfEnrollmentDay

SET dropoutYears **TO** (String) dropoutYear.GetSelectedItem()

SET dropoutMonths **TO** (String) dropoutMonth.GetSelectedItem()

SET dropoutDays **TO** (String) dropoutDay.GetSelectedItem()

SET dateOfDropoutCombined **TO** dropoutYears, dropoutMonths, dropoutDays

TRY

SET dropoutEnrollmentIDValue **TO**

Integer.parseInt(dropoutEnrollmentID)

SET dropoutCourseDurationValue **TO**

Integer.parseInt(dropoutCourseDuration)

SET dropoutTuitionFeeValue **TO**

Integer.parseInt(dropoutTuitionFee)

SET dropoutRemainingModules **TO**

```
Integer.parseInt(dropoutRemainingModulesText)
SET dropoutMontAttended TO
Integer.parseInt(dropoutMonthAttendedText)
FOR EACH student IN allStudent
    IF student IS INSTANCE OF Regular and
    student.getEnrollmentid() EQUALS
    dropoutEnrollmentIDValue
    DO
        IF sharedDropoutStudent is NULL
            sharedDropoutStudent  EQUALS
            CREATE Dropout with parameter(
            dateOfBirthCombined,
            dropoutStudentName,
            dropoutTutionFeeValue,
             dropoutRemainingModules,
            dropoutMonthAttended,
            dateOfDropoutCombined)
        END IF
        IF sharedDropoutStudent.getHasPaid()
        EQUALS true
            DISPLAY "All bills cleared! Student has
             Been Removed" AS
            INFORMATION_MESSAGE
```

```
                    REMOVE students FROM allStudents
                    ADD sharedDropoutStudent TO
                    allStudent
                ELSE
                    DISPLAY “Bills not cleared! Please clear
                    all bills”
                END IF
            END IF
        END FOR EACH
    CATCH NumberFormatException AS a
        DISPLAY “Invalid input! Please enter a valid integer.”
        AS INFORMATION_MESSAGE
    END TRY
    END DO
END DO
CALL displayRegular and ADD actionListener with parameter
ActionLisnter() method
DO
    CREATE a method actionPerformend with parameter
    ActionEvent
    DO
        CALL displayRegularStudentAction() method
    END DO
```

END DO

CALL displayRegular and **ADD** actionListener with parameter ActionListener() method

DO

CREATE a method actionPerformed() with return type and parameter ActionEvent

DO

CALL displayRegularStudentAction()

END DO

END DO

CALL add and **ADD** actionListener with parameter ActionListener() method

DO

SET studentName **TO** studentTextField.getText()

SET courseName TO courseNameTextField.getText()

SET enrollmentStr TO enrollmentIDTextField.getText()

SET courseDuration TO courseDurationTextField.getText()

SET tuitionFeeStr TO tuitionFeeTextField.getText()

SET daysPresentStr TO daysPresentTextField.getText()

SET creditHoursStr TO numberOfCreditHoursTextField.getText()

SET dateOfBirthYear TO (String) dateOfBirthYear.getSelectedItem()

SET dateOfBirthMonth TO (String) dateOfBirthMonths.getSelectedItem()

```
SET dateOfBirthDay TO (String) dateOfBirthDays.getSelectedItem()
SET dateOfEnrollmentYear TO (String)
SET dateOfBirthCombined EQUALS dateofBirthYear,
dateOfBirthMonth, dateOfBirthDay
dateOfEnrollmentYears.getSelectedItem()
SET dateOfEnrollmentMonth TO (String)
dropOfEnrollmentMonths.getSelectedItem()
SET dateOfEnrollmentDay TO (String)
dateOfEnrollmentDays.getSelectedItem()
SET dateOfBirthCombined EQAULS dateOfEnrollmentYear,
dateOfEnrollmentMonth,dateOfEnrollmentDays
TRY
    DO
    CONVERT String value TO Integer.parseInt()
    ASSIGN Boolean value found to False
    FOR EACH student in allStudent
        IF student IS INSTANCE OF regular and
        Student.getEnrollmentId() EQUALS
        enrollmentIDValue
            ASSIGN found to TRUE
            BREAK
        END IF
    END FOR
```

```
                IF found
                        DISPLAY "Student with that Enrollment ID already
                        Exist" AS INFORMATION_MESSAGE
                ELSE
                        CREATE newDropoutStudent AS Dropout with
                         Parameter
                        ADD newDropoutStudent TO allStudent
                        DISPLAY "Student has been added" AS
                        INFORMATION_MESSAGE
                END IF
        CATCH NumberFormatException AS a
                DISPLAY "Invalid input! Please enter a valid integer" AS
                INFORMATION_MESSAGE
        END TRY
        END DO
END DO
CALL clearRegular and ADD actionListener with parameter
ActionListener () method
DO
        CREATE method actionPerformed with parameter ActionEvent
        DO
                CALL  clearRegularData()
        END DO
```

```
        END DO
        CALL clearDropout and ADD actionListener with parameter
        ActionListener () method
        DO
            CREATE method actionPerformed with parameter ActionEvent
            DO
                CALL  clearDropoutData()
            END DO
        END DO
    END DO
    CREATE method clearRegularData with return type void
    DO
        SET studentTextField TO empty
        SET enrollmentIDTextfField TO empty
        SET courseDurationTextField TO empty
        SET tuitionFeeTextField TO empty
        SET numberOfModulesTextField TO empty
        SET courseNameTextField TO empty
        SET numberOfCreditHoursTextField TO empty
        SET daysPresentTextField TO empty
        SET dateOfBirthYear SELECTED Index to zero
        SET dateOfBirthMonths SELECTED Index to zero
        SET dateOfBirthDays SELECTED Index to zero
```

SET dateOfEnrollmentYears **SELECTED** Index to zero

SET dateOfEnrollmentMonths **SELECTED** Index to zero

SET dateOfEnrollmentDays **SELECTED** Index to zero

END DO

CREATE method clearDropoutData with return type void

DO

SET dropoutstudentTextField **TO** empty

SET dropoutenrollmentIDTextfField **TO** empty

SET dropoutcourseDurationTextField **TO** empty

SET dropouttuitionFeeTextField **TO** empty

SET dropoutcourseNameTextField **TO** empty

SET dropoutRemainingModules **TO** empty

SET dropoutMonthAttendedTextField **TO** empty

SET dropoutdateOfBirthYear **SELECTED** Index to zero

SET dropoutdateOfBirthMonths **SELECTED** Index to zero

SET dropoutdateOfBirthDays **SELECTED** Index to zero

SET dropoutdateOfEnrollmentYears **SELECTED** Index to zero

SET dropoutdateOfEnrollmentMonths **SELECTED** Index to zero

SET dropoutdateOfEnrollmentDays **SELECTED** Index to zero

SET dropoutYear **SELECTED** Index to zero

SET dropoutMonth **SELECTED** Index to zero

SET dropoutDay **SELECTED** Index to zero

END DO

```
CREATE method displayReguarStudentAction() with return type void
DO
    CREATE message AS StringBuilder
    ASSIGN hasChildClassObject TO false
    FOR EACH student IN allStudent
        IF student is INSTANCE OF Regular
            SET hasChildClassObject TO true
            BREAK
        END IF
    END FOR EACH
    IF hasChildClassObject is false
        DISPLAY "No regular Student found!"
        RETURN
    END IF
    FOR EACH student IN allStudent
    DO
        IF student is INSTANCE OF Regular
            SET regularStudent EQUALS student TO Regular
            ADD studentName, courseName, enrollmentID,
            courseDuration, numberOfModules,daysPresent,
            CreditHours,TuitionFee,dateOfBirth,dateOfEnrollment
        END IF
    END FOR EACH
```

```
        ASSIGN regularMessage to convert messge to string
        DISPLAY regularMessage USING regularFrame AS PARENT COMPENT
END


CREATE method displayDropoutStudentAction() with return type void
DO
        CREATE message AS StringBuilder
        ASSIGN hasChildClassObject TO false
        FOR EACH student IN allStudent
                IF student is INSTANCE OF Regular
                        SET hasChildClassObject TO true
                        BREAK
                END IF
        END FOR EACH
        IF hasChildClassObject is false
                DISPLAY "No dropout Student found!"
                RETURN
        END IF
        FOR EACH student IN allStudent
        DO
                IF student is INSTANCE OF Regular
                        SET regularStudent EQUALS student TO Regular
                        ADD studentName, courseName, enrollmentID,
```

```
                    courseDuration, numberOfModules,daysPresent,
                    CreditHours,TuitionFee,dateOfBirth,dateOfEnrollment
                END IF
            END FOR EACH
            ASSIGN regularMessage to convert messge to string
            DISPLAY regularMessage USING dropoutFrame AS PARENT COMPENT
        END
        CREATE main method
        DO
            CREATE newStudentGUI AS StudentGUI()
        END DO
END DO
```

Method

Methods in Java are collections of statements that perform specific tasks and then return to their callers. In some cases, Java methods are not required to return because they are used to perform specific tasks. Because Java's methods make reusing code easy, no retyping is required. (GeeksforGeeks, 2023)

StudentGUI

regularStudent	Opens the Regular Frame for the regular student. As soon as the button is clicked, the opening frame is made invisible and the regular frame is displayed to the user, effectively transferring the interface from the opening frame to the regular frame.
dropoutStudent	Opens the Dropout Frame for the dropout student. By clicking this button, the working area is re-opened and the dropping frame is displayed to the user, giving the user the option of navigating from the current view to one that is tailored for managing dropouts.
percentage	The action is to extract input values from various text fields and dropdown menus, such as enrollment ID, days present, name, and enrollment date. A pop-up message displays the student's corresponding grade once it has processed this information, calculated the attendance percentage, and displayed the corresponding grade. Users are given appropriate error messages if their student

	is not found or if they have entered invalid information.
certificate	A click on this button triggers a process in which input values from input fields and dropdown menus are extracted, including enrollment ID, days present, date of birth, and date of enrollment. Based on that enrollment ID, it determines whether the student (probably a "Regular" student) is eligible for a certificate (specifically, if the student is a Grade A student). Success messages confirm that the certificate has been granted if the criteria are met, otherwise errors indicate the certificate has not been granted. A user will receive an appropriate error message if a matching student cannot be found or if the input is invalid.
Add	As soon as the button is clicked, the input boxes and dropdown menus begin capturing values, such as the name of the student, information about the course, enrollment ID, length of time, fees, the number of modules, attendance, and credit hours. Following the conversion, several tests are performed to verify if the input strings are suitable for integer values. As a result, a new instance of a "Regular" student is created using the provided information and added to a student collection if there is no existing

	student with the given enrollment ID. According to the outcome of these checks and operations, users are presented with success or error messages. A message is displayed if there are any invalid inputs.
displayRegular	Information about regular students is displayed by this button. AllStudents are iterated through and the class "Regular" is checked for. A message indicating the absence of regular students is displayed if none are found. Alternatively, the method generates a detailed message containing details such as the name and course of each regular student, the number of modules, their attendance, the credit hours they have earned, the tuition fee they have paid, their date of birth, and their date of enrollment. Using the regularFrame context, the JOptionPane.showMessageDialog function is used to display this message to the user.
clearRegular	A button press triggered by a "clearRegular" button invokes the clearRegularData() method. Thus, the data entry form is reset and ready for new data entry by clearing the input fields and combo boxes in this method. Adding student details for regular students can be streamlined by clearing previously entered information.

bills	Input data for a dropout student are collected by clicking this button, including the student's name, enrollment ID, course duration, tuition fee, remaining modules, modules attended, and different date selections. In the following portion of the code, the inputs are converted into integers and a search is conducted to find a matching "Regular" student with the given enrollment ID from among a collection of students (allStudents). "Dropout" students are created, their information is updated, and they are marked as having paid, and a success message is displayed if all bills are paid. It displays appropriate error messages if there is no matching student found or if the user enters incorrect information.
removeStudent	Input data related to the dropout student will be collected when this button is pressed, including name, enrollment ID, course details, and various dates. In this step, the code converts these inputs into integer values, then searches through the allStudents collection for a student with the same enrollment ID. After finding a student whose bill has been cleared, a "Dropout" student is created. An error message appears if the bill has not been cleared. Whenever a user inputs invalid

	information, they are shown the appropriate error message.
displayDropout	A request for the displayDropoutAction() method is triggered by pressing the button "displayDropout". Information about dropout students is assembled and presented by this method. The class "Dropout" is checked for instances in the collection of students. If no dropouts are found, the absence of these students is indicated by a message. If the method does not construct a detailed message with details like name, enrollment ID, course name, duration, tuition fee, remaining modules, months attended, and dropout dates, then the method constructs a general message. Pop-up messages within dropout frames are used to display the information to the user.
clearDropout	A "clearDropout" button triggers a method call when it is pressed. With the help of this method, dropout students' input fields and combo boxes can be reset or cleared. A call to this method resets combo boxes associated with date selections, including those containing a student name, enrollment ID, course details, and remaining modules. Users can easily clear previous information when adding details for dropout students by doing this. This

	ensures that the form is reset and ready for new data entry.

Table 1:Method Description

Testing

Test One

Test No.	1
Objective	• To that the program can be complied and run using command prompt
Action	• Use command prompt to compile the StudentGUI using the command “javac Student.java”. • Use command prompt to run the StudentGUI program using command using the command “java StudentGUI.java”.
Expected Result	• The program would compile without any errors.
Actual Result	• The program was compiled without any errors.
Conclusion	Test Successful without any Error

Table 2:Test One

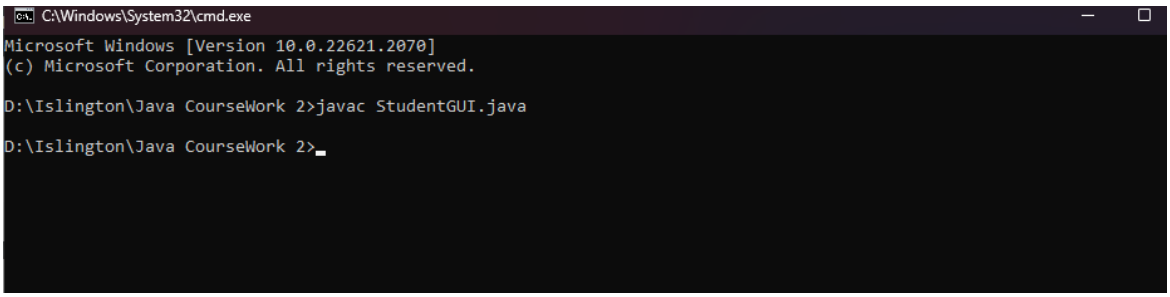

Figure 2: Compiling StudentGUI

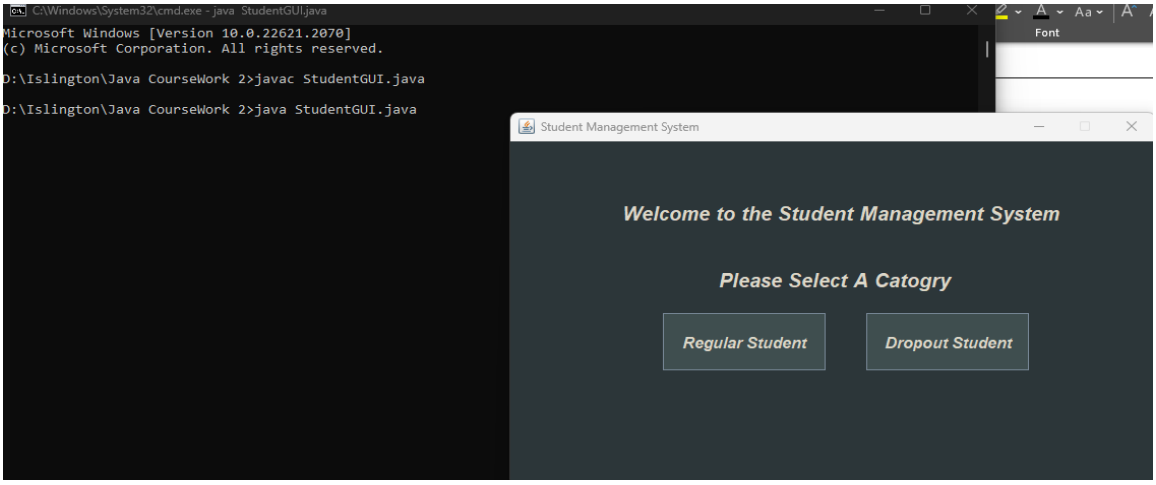

Figure 3: Running StudentGUI

Test Two

Test No	2
Objective	To show the functionality of Every button
Action	• Enter the following inputs and press all the buttons - For Regular Student Name = Shamel Date Of Birth: 2001-7-3 Enrollment ID= 1 Enrollment Date=2004-4-19 Course Duration =30 Tuition Fee = 10000 Number of Modules = 5 Course Name = Computing Course hours = 24 Days present = 16 - For Dropout Student Name = Shamel Date Of Birth: 2001-7-3

	Enrollment ID= 1 Enrollment Date=2004-4-19 Course Name = Computing Course Duration =30 Tuition Fee: 10000 Number of Remaining Modules:8 Credit Hours= 24 Date Of Dropout =2022 -8-11
Expected Result	• Add regular Student would display regular “Student has been added” in a dialog box. • Add dropout Student would display “ Student has been added” in a dialog box. • When the calculate percentage is pressed it would display the grade of the regular student in a dialog box. • When the grant certificate is pressed it would display whether the student would display “Certificate Granted!” or ”Ce rtificate not Granted!” in a dialog box. • When pay bills is pressed it would display “All bills cleared!” in a dialog box. • When the remove student is pressed it would remove the student and display “All bills

	cleared! Student has been removed” in a dialog box.
Action Result	• “Student has been added” was displayed when the add button was pressed in regular student. • “Student has been added” was displayed when add button was pressed in dropout student. • Grade “B” was displayed when the calculate percentage was pressed in regular student. • “Certificate not Granted” was displayed when grant certificate button was pressed. • “All bills cleared!” was displayed when pay bills was pressed. • “All bills cleared! Student has been removed” was displayed when the remove student was pressed.
Conclusion	Test Successful without any Error

Table 3:Test Two

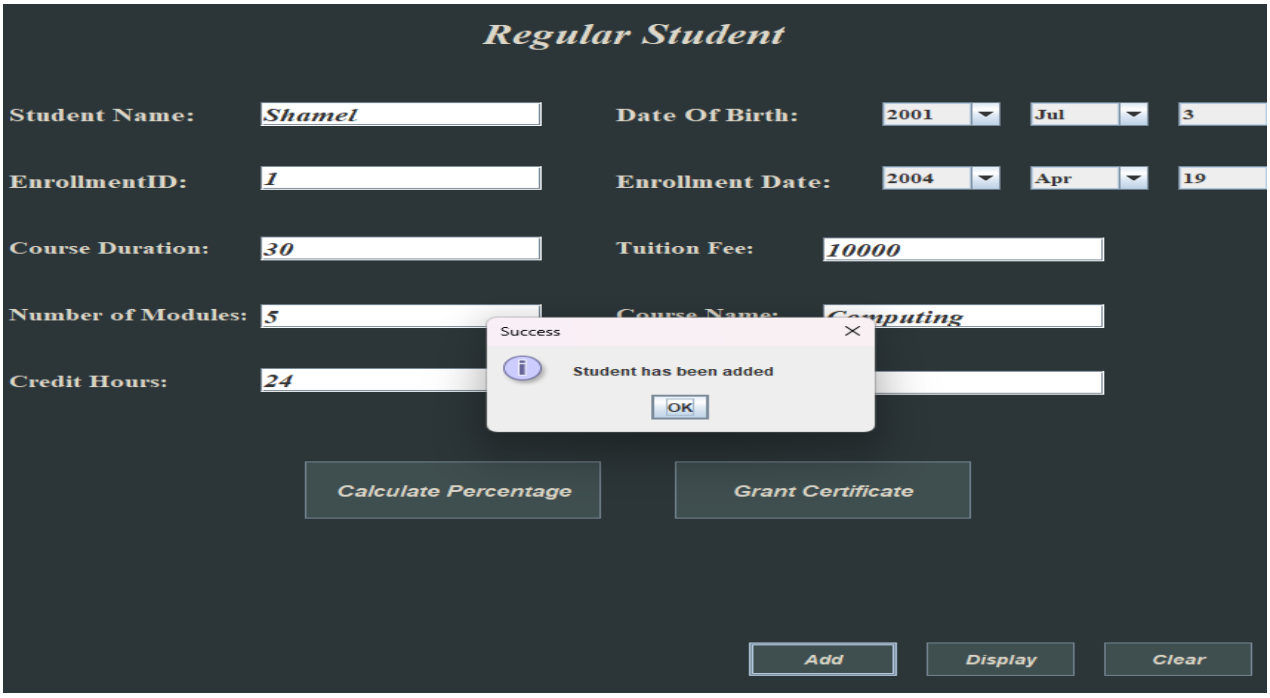

Figure 4: Add Regular

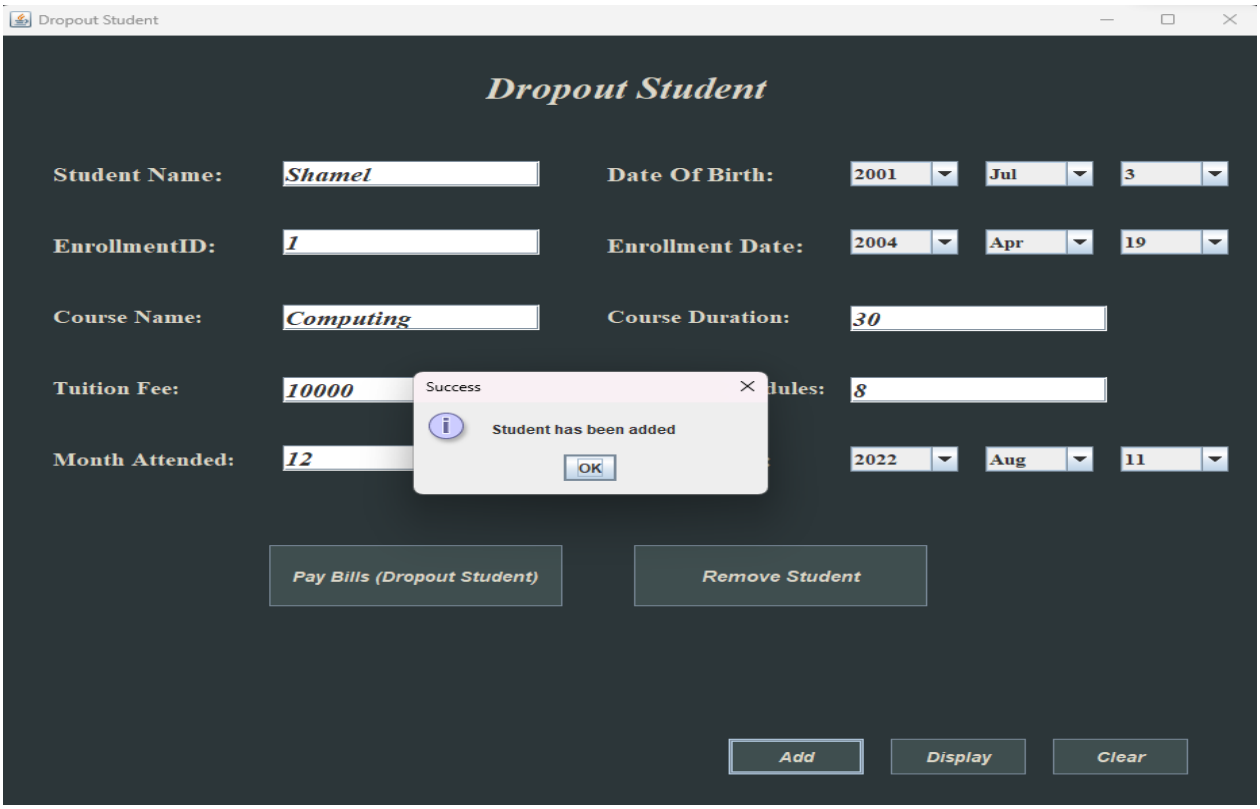

Figure 5: Add Dropout

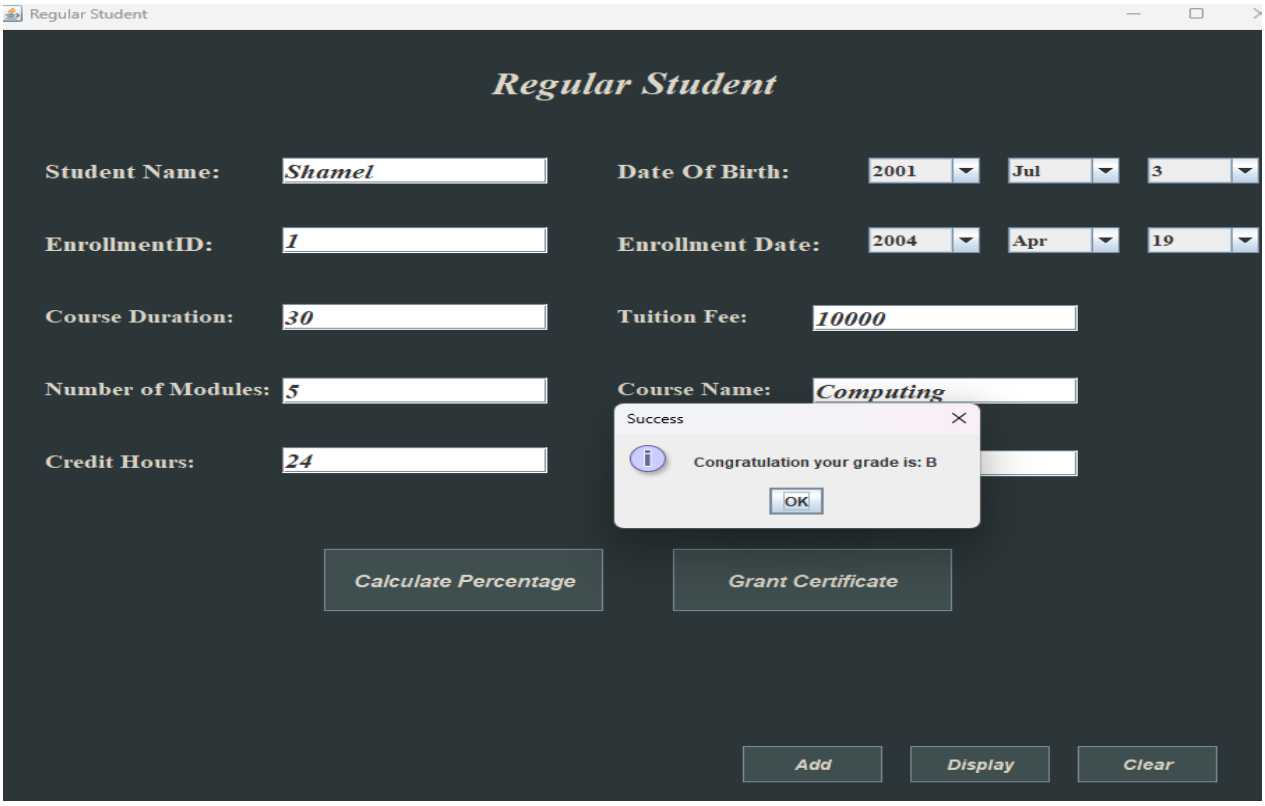

Figure 6: Calculate Percentage

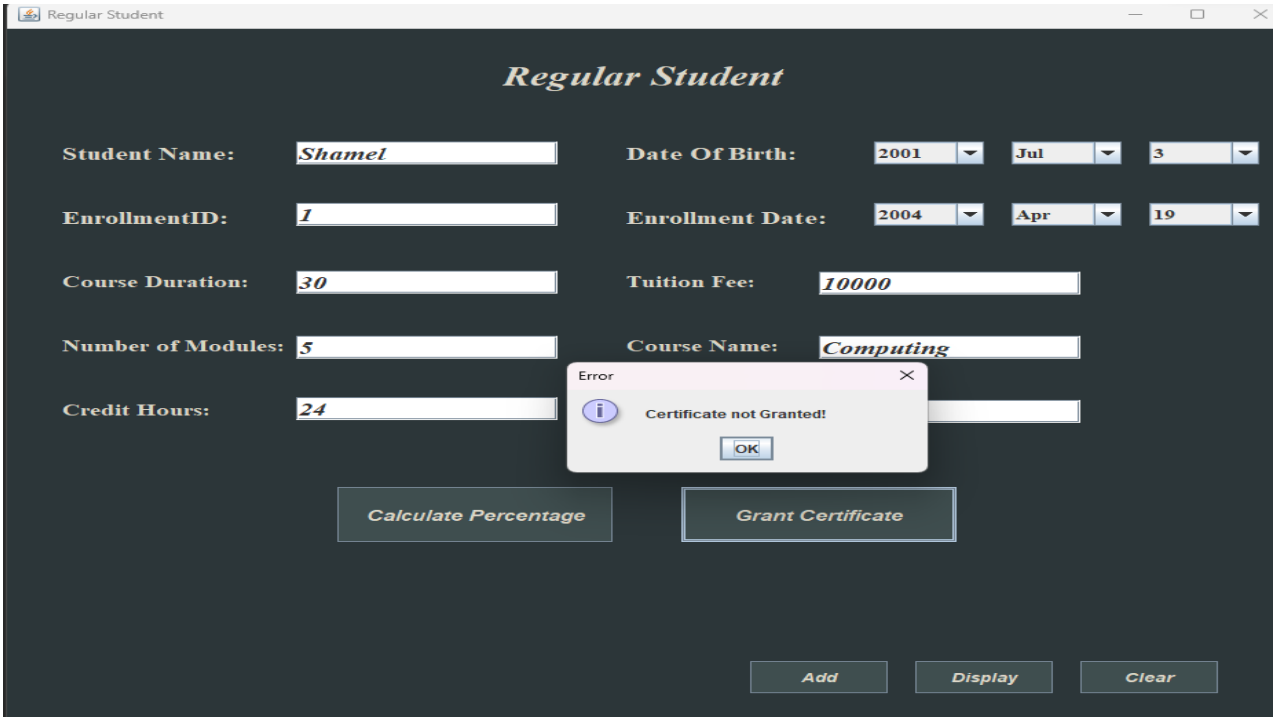

Figure 7:Grant Certificate

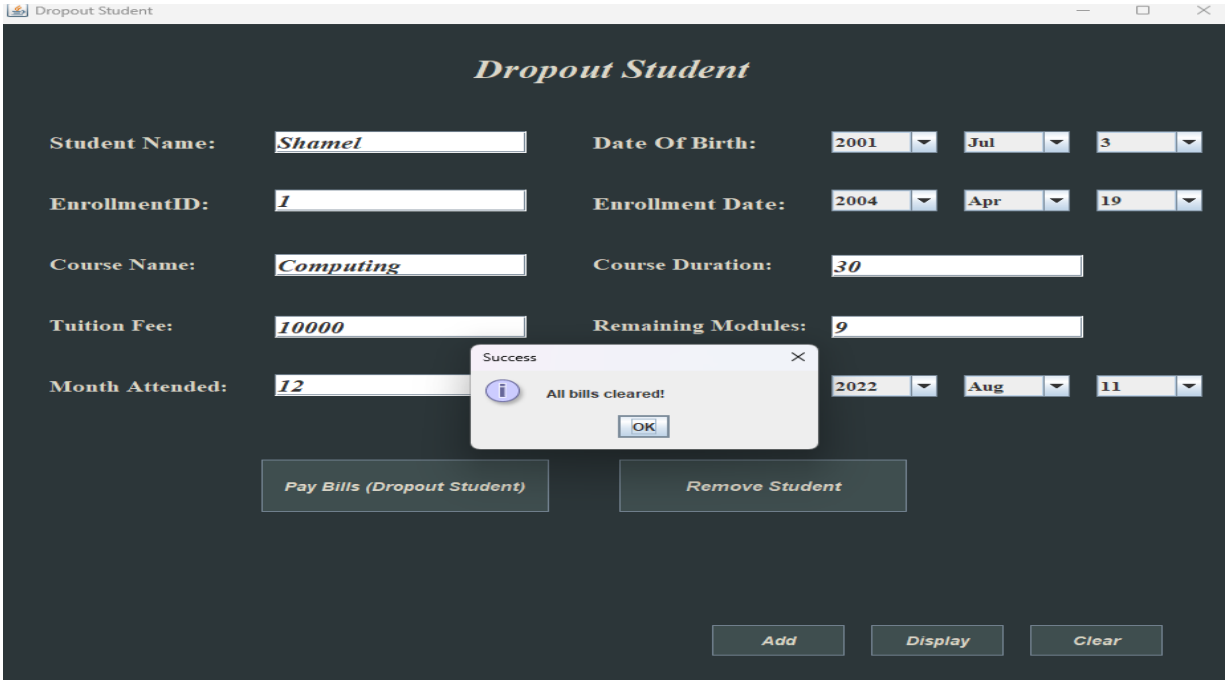

Figure 8: Pay Bills

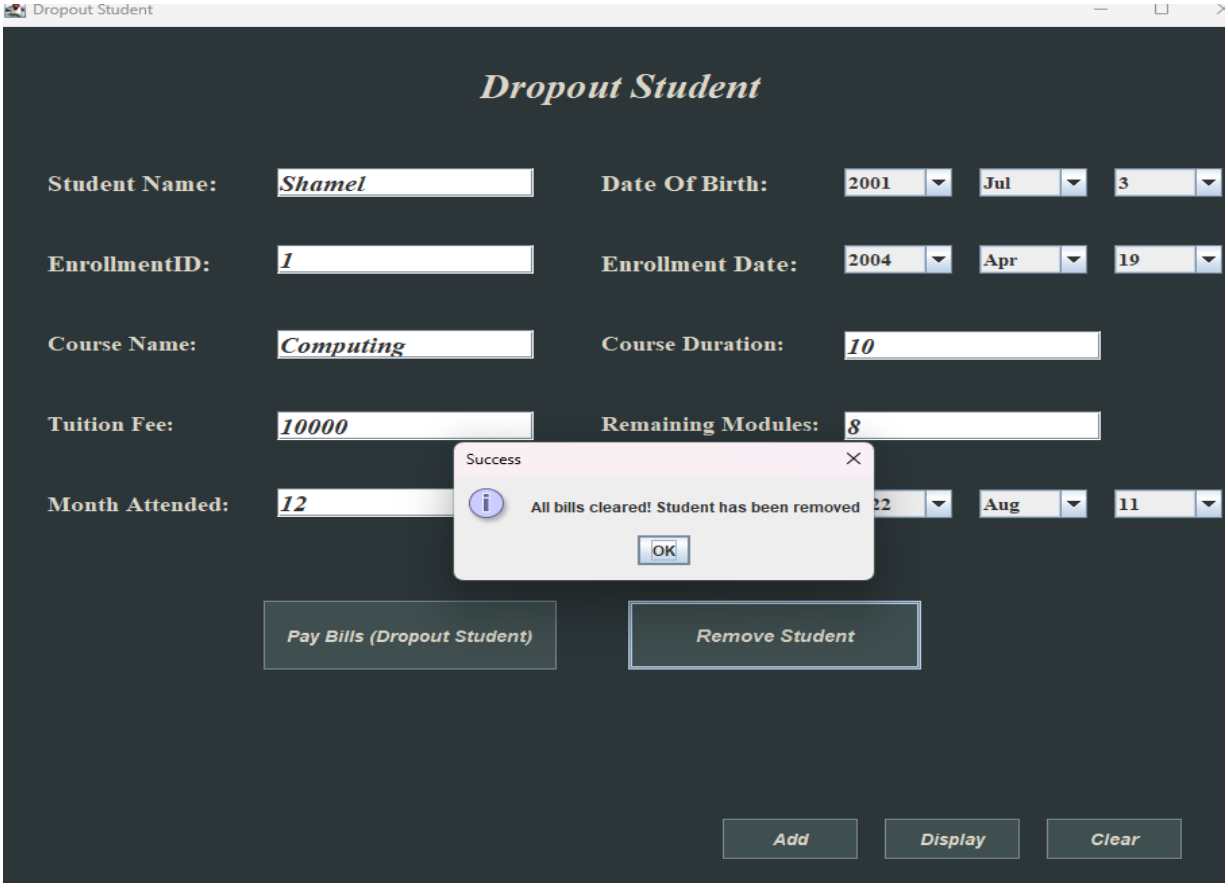

Figure 9:Remove Student

Test Three

Test No.	3
Objective	• To test when unsuitable values are entered for the Enrollment ID.
Action	• Open any frame regular/ dropout frame. • For the Enrollment ID field, enter an inappropriate value (for example, a value that is not numeric or out of range). • When the "Add Regular Student" button is clicked, the associated action is triggered. • Check that the dialog box showing an invalid Enrollment ID error message appears. • A screenshot should be taken of the error dialog box and the corresponding GUI interface.
Expective Result	• When the add button is pressed it would display “Invalid output! Please enter a valid integer.”.
Action Result	• When the add button was pressed it displayed “Invalid output! Please enter a valid integer.”.
Conclusion	Test run successfully without any error

Table 4:Test Three

Figure 10:Unsuitable value

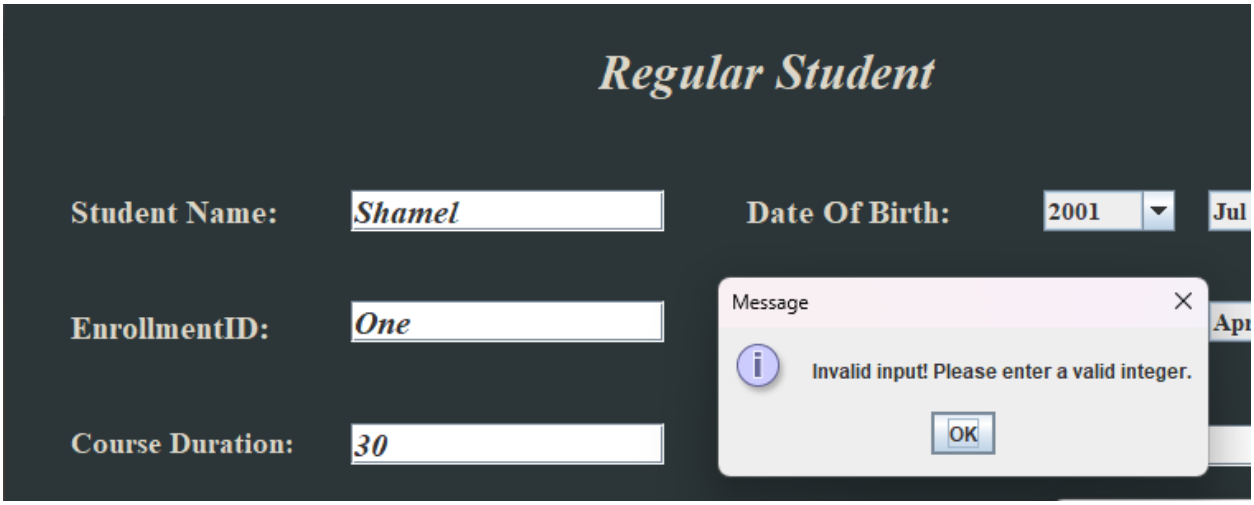

Figure 11: Unsuitable Value

Error detection and error correction

Error detection and error correction Error-correcting codes are algorithms that express sequences of numbers in such a way as to detect and correct (within certain limits) any errors introduced in the sequence. (MathWorld, n.d.). A popular error detecting code is a block code that separates messages into set sized blocks of bits and superfluous bits for error detection. The goal of error detection is to ascertain whether an error occurred. It makes no difference how many mistake bits are in a file or what kind they are. (Thakur, 2020).

Syntax Error

If a programmer is writing an incorrect line of code, he's making syntax errors. The most common syntax errors are Punctuations and Unnameable Names. The code becomes ineffective, whether it is compiled or interpreted, due to a programming error. (Acevedo, 2021).

```
regularFrame.ad(dateOfEnrollmentDays);
```

Figure 12: Syntax Error

```
regularFrame.add(dateOfEnrollmentDays);
```

Figure 13: Syntax Error Correction

Runtime Error

While the programme is running, you may encounter a Runtime Error, commonly known as an exception. They are not recorded by the compiler during compilation but will be identified at runtime. Runtime errors are typically caused by problems such as dividing by zero, accessing an out of bounds array index, or attempting to utilise a null object reference. (Techslang, n.d.)

```java
// Converting the input Strings to integer using the parseIntValue method
int enrollmentIDValue = Integer.parseInt(enrollmentStr);
int courseDurationValue = Integer.parseInt(courseDurationStr);
int tuitionFeeValue = Integer.parseInt(tuitionFeeStr);
int numberOfModulesValue = Integer.parseInt(numberOfModulesStr);
int daysPresentValue = Integer.parseInt(daysPresentStr);
int creditHoursValue = Integer.parseInt(creditHoursStr);

boolean found = false;
for (Regular student : regularStudents) {
        if (student.getEnrollmentId() == enrollmentIDValue) {
                found = true;
                break;
        }
}


if (found) {
        JOptionPane.showMessageDialog(null, "Student with that Enrollment ID already exist", "Error",
                JOptionPane.INFORMATION_MESSAGE);
} else {
        Regular newRegularStudent = new Regular(enrollmentIDValue, dateOfBirthCombined, courseName, studentName, dateOfE
        regularStudents.add(newRegularStudent);

        JOptionPane.showMessageDialog(null, "Student has been added", "Success",
                JOptionPane.INFORMATION_MESSAGE);
```

Figure 14:Runtime Error

```java
try {
        // Converting the input Strings to integer using the parseIntValue method
        int enrollmentIDValue = Integer.parseInt(enrollmentStr);
        int courseDurationValue = Integer.parseInt(courseDurationStr);
        int tuitionFeeValue = Integer.parseInt(tuitionFeeStr);
        int numberOfModulesValue = Integer.parseInt(numberOfModulesStr);
        int daysPresentValue = Integer.parseInt(daysPresentStr);
        int creditHoursValue = Integer.parseInt(creditHoursStr);

        boolean found = false;
        for (Regular student : regularStudents) {
                if (student.getEnrollmentId() == enrollmentIDValue) {
                        found = true;
                        break;
                }
        }


        if (found) {
                JOptionPane.showMessageDialog(null, "Student with that Enrollment ID already exist", "Error",
                        JOptionPane.INFORMATION_MESSAGE);
        } else {
                Regular newRegularStudent = new Regular(enrollmentIDValue, dateOfBirthCombined, courseName, studentName,
                regularStudents.add(newRegularStudent);

                JOptionPane.showMessageDialog(null, "Student has been added", "Success",
                        JOptionPane.INFORMATION_MESSAGE);
        }
}catch (NumberFormatException a) {
        // To Display the Error Message To the User
        JOptionPane.showMessageDialog(openningFrame, "Invalid input! Please enter a valid integer.");
}
```

Figure 15: Runtime Error

Logical Error

These are the programming mistakes that developers make. Despite running and producing the desired These programs are loaded with these errors. The user often receives an output consisting of two numbers, but the user expects it to be multiplied. (Edureka, 2021).

```
if (!found) {
        JOptionPane.showMessageDialog(null, "Congratulation your grade is: '
} else {
        JOptionPane.showMessageDialog(null, "Did not find student", "Error",
}
ch (NumberFormatException a) {
```

Figure 16: Logical Error

```
if (found) {
        JOptionPane.showMessageDialog(null, "Congratulation your grade is: " +
} else {
        JOptionPane.showMessageDialog(null, "Did not find student", "Error", JO
}
(NumberFormatException a) {
```

Figure 17: Logical Error

Conclusion

In conclusion, the creation of an engaging Graphic User Interface (GUI) for the student management system represents a significant stride towards enhanced user satisfaction and streamlined information management. The visually appealing design not only boosts functionality but also encourages active participation from staff and administrators, fostering improved decision-making and data handling.

The GUI's key strength lies in its real-time insights and analytics, empowering administrators with comprehensive student performance data and visualizations. This newfound capability empowers educational institutions to identify areas for improvement and implement targeted initiatives, ultimately driving academic success.

Moreover, the GUI's seamless communication capabilities across different university divisions underscore its collaborative prowess. From faculty tracking student progress to the finance department managing fees, this connectivity ensures a harmonious curriculum. The GUI's commitment to inclusiveness ensures accessibility for users of all skill levels, aligning with industry standards and promoting equal opportunities. Beyond functionality, rigorous testing and user feedback integration promise an intuitive and personalized experience, fostering ease of use.

As the project progresses, it's evident that the GUI's success relies on a collective effort involving developers, education experts, and users. The goal of creating a universally adaptable interface, combining technological know-how with practical insights, ensures alignment with diverse educational needs.

The addition of an account management dashboard enhances functionality, convenience, and appeal, serving as a valuable tool for nurturing student experiences through a blend of function, aesthetics, and immediate feedback.

In summary, several challenges were encountered while developing the GUI for the student management system. In addition, the interface was refined to serve the user's needs across diverse devices and screen sizes, addressing technical complexities in the visualization of real-time data, and taking into account user feedback to ensure optimal usability. Iterative design and testing was required to strike a balance between

functionality, aesthetics, and accessibility. To ensure effective communication and data sharing, seamless integration with various university departments required meticulous coordination. The GUI's evolution represents innovation, collaboration, and a commitment to advancing educational software, offering an enriching and motivating environment for users across the education landscape.

References

Acevedo, S., 2021. *Woz U.* [Online]
Available at: https://woz-u.com/blog/common-programming-syntax-errors-and-how-to-fix-them/#:~:text=A%20syntax%20error%20occurs%20when,the%20code%20won't%20work.
[Accessed 7 8 2023].

Edureka, 2021. *Introduction to Error in Java.* [Online]
Available at: https://www.edureka.co/blog/introduction-to-errors-in-java/#:~:text=Logical%20Errors%20%E2%80%93%20These%20are%20the,expected%20is%20multiplication%20of%20numbers.
[Accessed 7 8 2023].

GeeksforGeeks, 2023. *GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/methods-in-java/

MathWorld, W., n.d. *Error-Correcting Code.* [Online]
Available at: https://mathworld.wolfram.com/Error-CorrectingCode.html#:~:text=An%20error%2Dcorrecting%20code%20is,is%20known%20as%20coding%20theory.
[Accessed 10 5 2023].

Techslang, n.d. *What is a Runtime Error?.* [Online]
Available at: https://www.techslang.com/definition/what-is-a-runtime-error/#:~:text=A%20runtime%20error%20is%20an,or%20even%20your%20personal%20computer.
[Accessed 7 8 2023].

TechTarget, 1999. *TargetTech.* [Online]
Available at:
https://www.techtarget.com/whatis/definition/pseudocode#:~:text=Pseudocode%20(pronounced%20SOO%2Ddoh%2D,process%20of%20developing%20a%20program.
[Accessed 9 5 2023].

Thakur, A., 2020. *Tutorialspoint.* [Online]
Available at: https://www.tutorialspoint.com/what-are-error-detecting-codes
[Accessed 10 5 2023].

Appendix

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.security.DrbgParameters;

import java.util.ArrayList;


public class StudentGUI {

	ArrayList<Student> allStudents = new ArrayList<Student>();


	Dropout sharedDropoutStudent = null;


	private JFrame openningFrame, regularFrame, dropoutFrame;


	   // For Regular Student

	private JLabel introduction, catogory, studentName, enrollmentID, daysPresent,
courseName, courseDuration,

			tuitionFee,numberOfModules, numberOfCreditHours,
regularIntroduction, dateOfBirth, dateOfEnrollment;

	private JTextField studentTextField, enrollmentIDTextField, courseNameTextField,
courseDurationTextField,

			tuitionFeeTextField, numberOfModulesTextField,
numberOfCreditHoursTextField,
```

```
            daysPresentTextField;
    private JComboBox<String> dateOfBirthYears, dateOfBirthMonths,
dateOfBirthDays, dateOfEnrollmentYears,
            dateOfEnrollmentMonths, dateOfEnrollmentDays;
    private JButton percentage, add, certification, displayRegular, clearRegular;

    // For Dropout Student
    private JLabel studentNameDropout, dropoutIntroduction, enrollmentIDDropout,
courseNameDropout,
            dropoutDateOfBirth, dropoutDateOfEnrollment, courseDurationDropout,
tuitionFeeDropout,
            numberOfRemainingModulesDropout,
            numberOfMonthAttendedDropout, dateOfDropout;
    private JComboBox<String> dateOfEnrollmentYearsDropout,
dateOfEnrollmentMonthsDropout,
            dateOfEnrollmentDaysDropout,
            dateOfBirthYearsDropout, dateOfBirthMonthsDropout,
dateOfBirthDaysDropout, dropoutYear,
            dropoutMonth, dropoutDay;
    private JTextField dropoutStudentTextField, dropoutEnrollmentTextField,
dropoutCourseName,
            dropoutCourseDuration, dropoutTuitionFee, dropoutRemainingModules,
            dropoutMonthAttendedTextField;
    private JButton bills, removeStudent, displayDropout, clearDropout, adding;
```

```
// buttons for the Oppening GUI
private JButton regularStudent, dropoutStudent;

public StudentGUI() {
    // For the Starter GUI
    openningFrame = new JFrame("Student Management System");

    introduction = new JLabel("Welcome to the Student Management System");
    catogory = new JLabel("Please Select A Catogry");

    regularStudent = new JButton("Regular Student");
    dropoutStudent = new JButton("Dropout Student");

    // Setting the x-axis, y-axis, height width of the Openning Frame components
    introduction.setBounds(110, 50, 500, 50);
    introduction.setFont(new Font("Serial", Font.ITALIC | Font.BOLD, 20));
    introduction.setForeground(new Color(220, 215, 201));
    catogory.setBounds(205, 120, 500, 50);
    catogory.setFont(new Font("Serial", Font.ITALIC | Font.BOLD, 20));
    catogory.setForeground(new Color(220, 215, 201));
    regularStudent.setBounds(150, 180, 160, 60);
    regularStudent.setBackground(new Color(63, 78, 79));
```

```
regularStudent.setForeground(new Color(220, 215, 201));
regularStudent.setFont(new Font("Serial", Font.ITALIC | Font.BOLD, 16));
dropoutStudent.setBounds(350, 180, 160, 60);
dropoutStudent.setBackground(new Color(63, 78, 79));
dropoutStudent.setFont(new Font("Serial", Font.ITALIC | Font.BOLD, 16));
dropoutStudent.setForeground(new Color(220, 215, 201));

// Adding the component in the openning Frame
openningFrame.add(introduction);
openningFrame.add(catogory);
openningFrame.add(regularStudent);
openningFrame.add(dropoutStudent);

// Setting the background of the openning Frame
openningFrame.getContentPane().setBackground(new Color(44, 54, 57));

openningFrame.setLayout(null);
openningFrame.setResizable(false);
openningFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
openningFrame.setSize(650, 400);
openningFrame.setVisible(true);
openningFrame.setLocationRelativeTo(null);
```

```
// For Regular Student
regularFrame = new JFrame("Regular Student");
regularIntroduction = new JLabel("Regular Student");

studentName = new JLabel("Student Name:");
studentTextField = new JTextField();

enrollmentID = new JLabel("EnrollmentID:");
enrollmentIDTextField = new JTextField();

courseDuration = new JLabel("Course Duration:");
courseDurationTextField = new JTextField();

tuitionFee = new JLabel("Tuition Fee:");
tuitionFeeTextField = new JTextField();

numberOfModules = new JLabel("Number of Modules:");
numberOfModulesTextField = new JTextField();

courseName = new JLabel("Course Name:");
courseNameTextField = new JTextField();

numberOfCreditHours = new JLabel("Credit Hours:");
```

```
        numberOfCreditHoursTextField = new JTextField();


        daysPresent = new JLabel("Days Present:");
        daysPresentTextField = new JTextField();


        dateOfBirth = new JLabel("Date Of Birth:");


        // Array for the date Of birth for the Regular Frame
        String yearValue[] = { "1995", "1996", "1997", "1998", "1999", "2000", "2001",
"2002", "2003", "2004",
                "2005",
                "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013",
"2014", "2015", "2016",
                "2017", "2018",
                "2019", "2020", "2021", "2022", "2023", "2024" };
        dateOfBirthYears = new JComboBox<String>(yearValue);


        String monthValue[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
"Sep", "Oct", "Nov",
                "Dec" };
        dateOfBirthMonths = new JComboBox<String>(monthValue);


        String dayValue[] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15",
```

```
                "16",
                "17",
                "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
"30", "31" };
        dateOfBirthDays = new JComboBox<String>(dayValue);


        // For Date of Enrollment
        dateOfEnrollment = new JLabel("Enrollment Date:");
        String enrollmentYear[] = { "1995", "1996", "1997", "1998", "1999", "2000",
"2001", "2002", "2003",
                "2004",
                "2005",
                "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013",
"2014", "2015", "2016",
                "2017", "2018",
                "2019", "2020", "2021", "2022", "2023", "2024" };
        dateOfEnrollmentYears = new JComboBox<String>(enrollmentYear);


        String enrollmentMonth[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
"Aug", "Sep", "Oct",
                "Nov",
                "Dec" };
        dateOfEnrollmentMonths = new JComboBox<String>(enrollmentMonth);
```

```
        String enrollmentDay[] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
"12", "13", "14",
                "15",
                "16",
                "17",
                "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
"30", "31" };
        dateOfEnrollmentDays = new JComboBox<String>(enrollmentDay);

        // JButton of the Regular Frame
        add = new JButton("Add");
        percentage = new JButton("Calculate Percentage");
        certification = new JButton("Grant Certificate");
        displayRegular = new JButton("Display");
        clearRegular = new JButton("Clear");

        // adding all the JLabels inside the Regular frame.
        regularFrame.add(regularIntroduction);
        regularFrame.add(studentName);
        regularFrame.add(enrollmentID);
        regularFrame.add(courseDuration);
        regularFrame.add(courseName);
        regularFrame.add(tuitionFee);
```

```
regularFrame.add(numberOfCreditHours);
regularFrame.add(numberOfModules);
regularFrame.add(daysPresent);
regularFrame.add(dateOfBirth);
regularFrame.add(dateOfEnrollment);

// Adding all the JButtons inside the Regular frame
regularFrame.add(add);
regularFrame.add(percentage);
regularFrame.add(certification);
regularFrame.add(displayRegular);
regularFrame.add(clearRegular);

// Adding all the JTextField inside the Regular Frame
regularFrame.add(studentTextField);
regularFrame.add(enrollmentIDTextField);
regularFrame.add(courseDurationTextField);
regularFrame.add(courseNameTextField);
regularFrame.add(tuitionFeeTextField);
regularFrame.add(numberOfCreditHoursTextField);
regularFrame.add(numberOfModulesTextField);
regularFrame.add(daysPresentTextField);
regularFrame.add(dateOfBirthYears);
```

```
regularFrame.add(dateOfBirthMonths);
regularFrame.add(dateOfBirthDays);
regularFrame.add(dateOfEnrollmentYears);
regularFrame.add(dateOfEnrollmentMonths);
regularFrame.add(dateOfEnrollmentDays);

// setting Bonds of the JButtons for the Regular Frame

add.setBounds(540, 690, 100, 35);
add.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 14));
add.setBackground(new Color(63, 78, 79));
add.setForeground(new Color(220, 215, 201));
percentage.setBounds(240, 500, 200, 60);
percentage.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 16));
percentage.setBackground(new Color(63, 78, 79));
percentage.setForeground(new Color(220, 215, 201));
certification.setBounds(490, 500, 200, 60);
certification.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 16));
certification.setBackground(new Color(63, 78, 79));
certification.setForeground(new Color(220, 215, 201));
displayRegular.setBounds(660, 690, 100, 35);
displayRegular.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 14));
displayRegular.setBackground(new Color(63, 78, 79));
```

```
displayRegular.setForeground(new Color(220, 215, 201));
clearRegular.setBounds(780, 690, 100, 35);
clearRegular.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 14));
clearRegular.setBackground(new Color(63, 78, 79));
clearRegular.setForeground(new Color(220, 215, 201));

regularIntroduction.setBounds(360, 30, 220, 40);
regularIntroduction.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 30));
regularIntroduction.setForeground(new Color(220, 215, 201));

studentName.setBounds(40, 100, 150, 70);
studentName.setFont(new Font("Serif", Font.BOLD, 20));
studentName.setForeground(new Color(220, 215, 201));
studentTextField.setBounds(210, 123, 190, 25);
studentTextField.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 20));

dateOfBirth.setBounds(450, 100, 150, 70);
dateOfBirth.setFont(new Font("Serif", Font.BOLD, 20));
dateOfBirth.setForeground(new Color(220, 215, 201));
dateOfBirthYears.setBounds(630, 123, 80, 25);
dateOfBirthYears.setFont(new Font("Serif", Font.BOLD, 15));
dateOfBirthMonths.setBounds(730, 123, 80, 25);
dateOfBirthMonths.setFont(new Font("Serif", Font.BOLD, 15));
```

```
dateOfBirthDays.setBounds(830, 123, 80, 25);
dateOfBirthDays.setFont(new Font("Serif", Font.BOLD, 15));

enrollmentID.setBounds(40, 170, 125, 70);
enrollmentID.setFont(new Font("Serif", Font.BOLD, 20));
enrollmentID.setForeground(new Color(220, 215, 201));
enrollmentIDTextField.setBounds(210, 190, 190, 25);
enrollmentIDTextField.setFont(new Font("Serif", Font.ITALIC | Font.BOLD,
20));

dateOfEnrollment.setBounds(450, 170, 150, 70);
dateOfEnrollment.setFont(new Font("Serif", Font.BOLD, 20));
dateOfEnrollment.setForeground(new Color(220, 215, 201));
dateOfEnrollmentYears.setBounds(630, 190, 80, 25);
dateOfEnrollmentYears.setFont(new Font("Serif", Font.BOLD, 15));
dateOfEnrollmentMonths.setBounds(730, 190, 80, 25);
dateOfEnrollmentMonths.setFont(new Font("Serif", Font.BOLD, 15));
dateOfEnrollmentDays.setBounds(830, 190, 80, 25);
dateOfEnrollmentDays.setFont(new Font("Serif", Font.BOLD, 15));

courseDuration.setBounds(40, 240, 150, 70);
courseDuration.setFont(new Font("Serif", Font.BOLD, 18));
courseDuration.setForeground(new Color(220, 215, 201));
```

```
        courseDurationTextField.setBounds(210, 264, 190, 25);
        courseDurationTextField.setFont(new Font("Serif", Font.ITALIC | Font.BOLD,
20));

        tuitionFee.setBounds(450, 240, 100, 70);
        tuitionFee.setFont(new Font("Serif", Font.BOLD, 18));
        tuitionFee.setForeground(new Color(220, 215, 201));
        tuitionFeeTextField.setBounds(590, 265, 190, 25);
        tuitionFeeTextField.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 20));

        numberOfModules.setBounds(40, 310, 180, 70);
        numberOfModules.setFont(new Font("Serif", Font.BOLD, 18));
        numberOfModules.setForeground(new Color(220, 215, 201));
        numberOfModulesTextField.setBounds(210, 335, 190, 25);
        numberOfModulesTextField.setFont(new Font("Serif", Font.ITALIC |
Font.BOLD, 20));

        courseName.setBounds(450, 310, 150, 70);
        courseName.setFont(new Font("Serif", Font.BOLD, 18));
        courseName.setForeground(new Color(220, 215, 201));
        courseNameTextField.setBounds(590, 335, 190, 25);
        courseNameTextField.setFont(new Font("Serif", Font.ITALIC | Font.BOLD,
20));
```

```
        numberOfCreditHours.setBounds(40, 380, 120, 70);
        numberOfCreditHours.setFont(new Font("Serif", Font.BOLD, 18));
        numberOfCreditHours.setForeground(new Color(220, 215, 201));
        numberOfCreditHoursTextField.setBounds(210, 402, 190, 25);
        numberOfCreditHoursTextField.setFont(new Font("Serif", Font.ITALIC |
Font.BOLD, 20));

        daysPresent.setBounds(450, 380, 110, 70);
        daysPresent.setFont(new Font("Serif", Font.BOLD, 18));
        daysPresent.setForeground(new Color(220, 215, 201));
        daysPresentTextField.setBounds(590, 405, 190, 25);
        daysPresentTextField.setFont(new Font("Serif", Font.ITALIC | Font.BOLD,
20));

        // Background color for the Regular Frame
        regularFrame.getContentPane().setBackground(new Color(44, 54, 57));

        regularFrame.setLayout(null);
        regularFrame.setSize(950, 800);
        regularFrame.setVisible(false);
        regularFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
// GUI for the Dropout
dropoutFrame = new JFrame("Dropout Student");
dropoutIntroduction = new JLabel("Dropout Student");

studentNameDropout = new JLabel("Student Name:");
dropoutStudentTextField = new JTextField();

enrollmentIDDropout = new JLabel("EnrollmentID:");
dropoutEnrollmentTextField = new JTextField();

courseNameDropout = new JLabel("Course Name:");
dropoutCourseName = new JTextField();

courseDurationDropout = new JLabel("Course Duration:");
dropoutCourseDuration = new JTextField();

tuitionFeeDropout = new JLabel("Tuition Fee:");
dropoutTuitionFee = new JTextField();

numberOfRemainingModulesDropout = new JLabel("Remaining Modules:");
dropoutRemainingModules = new JTextField();

numberOfMonthAttendedDropout = new JLabel("Month Attended: ");
```

```
dropoutMonthAttendedTextField = new JTextField();


dropoutDateOfBirth = new JLabel("Date Of Birth:");
dropoutDateOfEnrollment = new JLabel("Enrollment Date:");
dateOfDropout = new JLabel("Dropout Date:");


// Array for the date of birth for the dropout year
String dodBirthYear[] = { "1995", "1996", "1997", "1998", "1999", "2000",
        "2001", "2002", "2003",
        "2004",
        "2005",
        "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013",
"2014",
        "2015", "2016",
        "2017", "2018",
        "2019", "2020", "2021", "2022", "2023", "2024" };
dateOfBirthYearsDropout = new JComboBox<String>(dodBirthYear);


String dodBirthMonth[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
        "Aug", "Sep", "Oct", "Nov",
        "Dec" };
dateOfBirthMonthsDropout = new JComboBox<String>(dodBirthMonth);
```

```
String dodBirthDay[] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
        "11", "12", "13", "14",
        "15",
        "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27",
"28",
        "29",
        "30", "31" };
dateOfBirthDaysDropout = new JComboBox<String>(dodBirthDay);

String dodYears[] = { "1995", "1996", "1997", "1998", "1999", "2000", "2001",
        "2002", "2003", "2004",
        "2005",
        "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013",
"2014",
        "2015", "2016",
        "2017", "2018",
        "2019", "2020", "2021", "2022", "2023", "2024" };
dateOfEnrollmentYearsDropout = new JComboBox<String>(dodYears);

String dodMonths[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
        "Aug", "Sep", "Oct", "Nov",
        "Dec" };
dateOfEnrollmentMonthsDropout = new JComboBox<String>(dodMonths);
```

```
String dodDays[] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
        "12", "13", "14", "15",
        "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27",
"28",
        "29",
        "30", "31" };
dateOfEnrollmentDaysDropout = new JComboBox<String>(dodDays);

String dropoutYears[] = { "1995", "1996", "1997", "1998", "1999", "2000",
        "2001", "2002", "2003",
        "2004",
        "2005",
        "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013",
"2014",
        "2015", "2016",
        "2017", "2018",
        "2019", "2020", "2021", "2022", "2023", "2024" };
dropoutYear = new JComboBox<String>(dropoutYears);

String dropoutMonths[] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
        "Aug", "Sep", "Oct", "Nov",
        "Dec" };
```

```
dropoutMonth = new JComboBox<String>(dropoutMonths);


String dropoutDays[] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
        "11", "12", "13", "14",
        "15",
        "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27",
"28",
        "29",
        "30", "31" };
dropoutDay = new JComboBox<String>(dropoutDays);


bills = new JButton("Pay Bills (Dropout Student)");
removeStudent = new JButton("Remove Student");
displayDropout = new JButton("Display");
clearDropout = new JButton("Clear");
adding = new JButton("Add");


// Adding all the JLabel for the Dropout Frame
dropoutFrame.add(dropoutIntroduction);
dropoutFrame.add(studentNameDropout);
dropoutFrame.add(enrollmentIDDropout);
dropoutFrame.add(courseNameDropout);
dropoutFrame.add(courseDurationDropout);
```

```
dropoutFrame.add(tuitionFeeDropout);
dropoutFrame.add(numberOfRemainingModulesDropout);
dropoutFrame.add(numberOfMonthAttendedDropout);
dropoutFrame.add(dropoutDateOfBirth);
dropoutFrame.add(dropoutDateOfEnrollment);
dropoutFrame.add(dateOfDropout);

// Adding the JButtons in the DropoutFrame
dropoutFrame.add(bills);
dropoutFrame.add(removeStudent);
dropoutFrame.add(displayDropout);
dropoutFrame.add(clearDropout);
dropoutFrame.add(adding);


// Adding the textField and the JComboBox in the Dropout Frame
dropoutFrame.add(dropoutStudentTextField);
dropoutFrame.add(dropoutEnrollmentTextField);
dropoutFrame.add(dropoutCourseName);
dropoutFrame.add(dropoutCourseDuration);
dropoutFrame.add(dropoutTuitionFee);
dropoutFrame.add(dropoutRemainingModules);
dropoutFrame.add(dropoutMonthAttendedTextField);
```

```
dropoutFrame.add(dateOfBirthYearsDropout);
dropoutFrame.add(dateOfBirthMonthsDropout);
dropoutFrame.add(dateOfBirthDaysDropout);
dropoutFrame.add(dateOfEnrollmentYearsDropout);
dropoutFrame.add(dateOfEnrollmentMonthsDropout);
dropoutFrame.add(dateOfEnrollmentDaysDropout);
dropoutFrame.add(dropoutYear);
dropoutFrame.add(dropoutMonth);
dropoutFrame.add(dropoutDay);

//
dropoutIntroduction.setBounds(360, 30, 210, 40);
dropoutIntroduction.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 30));
dropoutIntroduction.setForeground(new Color(220, 215, 201));

studentNameDropout.setBounds(40, 100, 150, 70);
studentNameDropout.setFont(new Font("Serif", Font.BOLD, 20));
studentNameDropout.setForeground(new Color(220, 215, 201));
dropoutStudentTextField.setBounds(210, 123, 190, 25);
dropoutStudentTextField.setFont(new Font("Serif", Font.ITALIC | Font.BOLD,
20));

enrollmentIDDropout.setBounds(40, 170, 125, 70);
```

```
enrollmentIDDropout.setFont(new Font("Serif", Font.BOLD, 20));
enrollmentIDDropout.setForeground(new Color(220, 215, 201));
dropoutEnrollmentTextField.setBounds(210, 190, 190, 25);
dropoutEnrollmentTextField.setFont(new Font("Serif", Font.ITALIC |
Font.BOLD, 20));

dropoutDateOfBirth.setBounds(450, 100, 150, 70);
dropoutDateOfBirth.setFont(new Font("Serif", Font.BOLD, 20));
dropoutDateOfBirth.setForeground(new Color(220, 215, 201));
dateOfBirthYearsDropout.setBounds(630, 123, 80, 25);
dateOfBirthYearsDropout.setFont(new Font("Serif", Font.BOLD, 15));
dateOfBirthMonthsDropout.setBounds(730, 123, 80, 25);
dateOfBirthMonthsDropout.setFont(new Font("Serif", Font.BOLD, 15));
dateOfBirthDaysDropout.setBounds(830, 123, 80, 25);
dateOfBirthDaysDropout.setFont(new Font("Serif", Font.BOLD, 15));

dropoutDateOfEnrollment.setBounds(450, 170, 150, 70);
dropoutDateOfEnrollment.setFont(new Font("Serif", Font.BOLD, 20));
dropoutDateOfEnrollment.setForeground(new Color(220, 215, 201));
dateOfEnrollmentYearsDropout.setBounds(630, 190, 80, 25);
dateOfEnrollmentYearsDropout.setFont(new Font("Serif", Font.BOLD, 15));
dateOfEnrollmentMonthsDropout.setBounds(730, 190, 80, 25);
dateOfEnrollmentMonthsDropout.setFont(new Font("Serif", Font.BOLD, 15));
```

```
dateOfEnrollmentDaysDropout.setBounds(830, 190, 80, 25);
dateOfEnrollmentDaysDropout.setFont(new Font("Serif", Font.BOLD, 15));

courseNameDropout.setBounds(40, 240, 150, 70);
courseNameDropout.setFont(new Font("Serif", Font.BOLD, 18));
courseNameDropout.setForeground(new Color(220, 215, 201));
dropoutCourseName.setBounds(210, 264, 190, 25);
dropoutCourseName.setFont(new Font("Serif", Font.ITALIC | Font.BOLD,
20));

courseDurationDropout.setBounds(450, 240, 150, 70);
courseDurationDropout.setFont(new Font("Serif", Font.BOLD, 18));
courseDurationDropout.setForeground(new Color(220, 215, 201));
dropoutCourseDuration.setBounds(630, 265, 190, 25);
dropoutCourseDuration.setFont(new Font("Serif", Font.ITALIC | Font.BOLD,
20));

tuitionFeeDropout.setBounds(40, 310, 180, 70);
tuitionFeeDropout.setFont(new Font("Serif", Font.BOLD, 18));
tuitionFeeDropout.setForeground(new Color(220, 215, 201));
dropoutTuitionFee.setBounds(210, 335, 190, 25);
dropoutTuitionFee.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 20));
```

```
        numberOfRemainingModulesDropout.setBounds(450, 310, 180, 70);
        numberOfRemainingModulesDropout.setFont(new Font("Serif", Font.BOLD,
18));
        numberOfRemainingModulesDropout.setForeground(new Color(220, 215,
201));
        dropoutRemainingModules.setBounds(630, 335, 190, 25);
        dropoutRemainingModules.setFont(new Font("Serif", Font.ITALIC |
Font.BOLD, 20));

        numberOfMonthAttendedDropout.setBounds(40, 380, 139, 70);
        numberOfMonthAttendedDropout.setFont(new Font("Serif", Font.BOLD, 18));
        numberOfMonthAttendedDropout.setForeground(new Color(220, 215, 201));
        dropoutMonthAttendedTextField.setBounds(210, 402, 190, 25);
        dropoutMonthAttendedTextField.setFont(new Font("Serif", Font.ITALIC |
Font.BOLD, 20));

        dateOfDropout.setBounds(450, 380, 150, 70);
        dateOfDropout.setForeground(new Color(220, 215, 201));
        dateOfDropout.setFont(new Font("Serif", Font.BOLD, 20));
        dropoutYear.setBounds(630, 403, 80, 25);
        dropoutYear.setFont(new Font("Serif", Font.BOLD, 15));
        dropoutMonth.setBounds(730, 403, 80, 25);
        dropoutMonth.setFont(new Font("Serif", Font.BOLD, 15));
        dropoutDay.setBounds(830, 403, 80, 25);
```

```
dropoutDay.setFont(new Font("Serif", Font.BOLD, 15));


bills.setBounds(200, 500, 217, 60);
bills.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 14));
bills.setBackground(new Color(63, 78, 79));
bills.setForeground(new Color(220, 215, 201));
removeStudent.setBounds(470, 500, 217, 60);
removeStudent.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 15));
removeStudent.setBackground(new Color(63, 78, 79));
removeStudent.setForeground(new Color(220, 215, 201));
displayDropout.setBounds(660, 690, 100, 35);
displayDropout.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 14));
displayDropout.setBackground(new Color(63, 78, 79));
displayDropout.setForeground(new Color(220, 215, 201));
clearDropout.setBounds(780, 690, 100, 35);
clearDropout.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 14));
clearDropout.setBackground(new Color(63, 78, 79));
clearDropout.setForeground(new Color(220, 215, 201));
adding.setBounds(540, 690, 100, 35);
adding.setFont(new Font("Serial", Font.BOLD | Font.ITALIC, 14));
adding.setBackground(new Color(63, 78, 79));
adding.setForeground(new Color(220, 215, 201));
```

```
        dropoutFrame.getContentPane().setBackground(new Color(44, 54, 57));

        dropoutFrame.setLayout(null);
        dropoutFrame.setVisible(false);
        dropoutFrame.setSize(950, 800);
        dropoutFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // calling the Event Handling Methods
        addActionListeners();

    }

    // Method for all the Event Handling
    public void addActionListeners() {
        // ActionListener for "regularStudent" button
        regularStudent.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                openningFrame.setVisible(true); // Hides the openingFrame
                regularFrame.setVisible(true); // Opens the regularFrame
            }
        });
```

```
// ActionListener for "dropoutStudent" button
dropoutStudent.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        openningFrame.setVisible(true); // Hides the openingFrame
        dropoutFrame.setVisible(true); // Opens the dropoutFrame
    }
});

// ActionListener for "displayDropout" button
displayDropout.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        displayDropoutAction();
    }
});


removeStudent.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String dropoutStudentName = dropoutStudentTextField.getText();
```

```
String dropoutEnrollmentID = dropoutEnrollmentTextField.getText();

String dropoutCourseDurationText =
dropoutCourseDuration.getText();

String dropoutTuitionFeeText = dropoutTuitionFee.getText();

String dropoutRemainingModulesText =
dropoutRemainingModules.getText();

String dropoutMonthAttendedText =
dropoutMonthAttendedTextField.getText();


// Getting the selected date values for Date of Birth

String dropoutDateOfBirthYear = (String)
dateOfBirthYearsDropout.getSelectedItem();

String dropoutDateOfBirthMonth = (String)
dateOfBirthMonthsDropout.getSelectedItem();

String dropoutDateOfBirthDay = (String)
dateOfBirthDaysDropout.getSelectedItem();

String dateOfBirthCombined = dropoutDateOfBirthYear + "-" +
dropoutDateOfBirthMonth + "-" + dropoutDateOfBirthDay;


// Getting the selected date values for Date of Birth

String dropoutDateOfEnrollmentYear = (String)
dateOfEnrollmentYearsDropout.getSelectedItem();

String dropoutDateOfEnrollmentMonth = (String)
dateOfEnrollmentMonthsDropout.getSelectedItem();
```

```
String dropoutDateOfEnrollmentDay = (String) dateOfEnrollmentDaysDropout.getSelectedItem();

String dateOfEnrollmentCombined = dropoutDateOfEnrollmentYear + "-" + dropoutDateOfEnrollmentMonth + "-" + dropoutDateOfEnrollmentDay;


// Getting the selected date values for Date of Birth

String dropoutYears = (String) dropoutYear.getSelectedItem();

String dropoutMonths = (String) dropoutMonth.getSelectedItem();

String dropoutDays = (String) dropoutDay.getSelectedItem();

String dateOfDropoutCombined = dropoutYears + "-" + dropoutMonths + "-" + dropoutDays;


try {

    // Converting the input Strings to integer using the parseIntValue method

    int dropoutEnrollmentIDValue = Integer.parseInt(dropoutEnrollmentID);

    int dropoutCourseDurationValue = Integer.parseInt(dropoutCourseDurationText);

    int dropoutTuitionFeeValue = Integer.parseInt(dropoutTuitionFeeText);

    int dropoutRemainingModules = Integer.parseInt(dropoutRemainingModulesText);

    int dropoutMonthAttended = Integer.parseInt(dropoutMonthAttendedText);
```

```
for (Student student : allStudents) {
    if (student instanceof Regular &&
student.getEnrollmentId() == dropoutEnrollmentIDValue) {
        if (sharedDropoutStudent == null) {
            sharedDropoutStudent = new
Dropout(dateOfBirthCombined, dropoutStudentName, dropoutCourseDurationValue,
dropoutTuitionFeeValue, dropoutRemainingModules, dropoutMonthAttended,
dateOfDropoutCombined);
        }
        if (sharedDropoutStudent.getHasPaid()) { // bills
cleared
            JOptionPane.showMessageDialog(null, "All bills
cleared! Student has been removed", "Success",

JOptionPane.INFORMATION_MESSAGE);
            allStudents.remove(student);
            allStudents.add(sharedDropoutStudent);
        } else {
            JOptionPane.showMessageDialog(null, "Bills
not cleared! Please clear all bills", "Error",

JOptionPane.INFORMATION_MESSAGE);
        }
```

```
                    }

                }
            }  catch (NumberFormatException a) {
                // To Display the Error Message To the User
                JOptionPane.showMessageDialog(openningFrame, "Invalid input! Please enter a valid integer.");
            }
        }
    });


    // ActionListener for "displayRegular" button
    displayRegular.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            displayRegularStudentAction();
        }
    });
```

```
        // student add button
        add.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Getting the input values as Strings
                String studentName = studentTextField.getText();
                String courseName = courseNameTextField.getText();
                String enrollmentStr = enrollmentIDTextField.getText();
                String courseDurationStr = courseDurationTextField.getText();
                String tuitionFeeStr = tuitionFeeTextField.getText();
                String numberOfModulesStr =
numberOfModulesTextField.getText();
                String daysPresentStr = daysPresentTextField.getText();
                String creditHoursStr = numberOfCreditHoursTextField.getText();


                // Getting the selected date values for Date of Birth
                String dateOfBirthYear = (String)
dateOfBirthYears.getSelectedItem();
                String dateOfBirthMonth = (String)
dateOfBirthMonths.getSelectedItem();
                String dateOfBirthDay = (String)
dateOfBirthDays.getSelectedItem();
                String dateOfBirthCombined = dateOfBirthYear + "-" +
dateOfBirthMonth + "-" + dateOfBirthDay;
```

```
                // Getting the selected date values for Date of Enrollment
                String dateOfEnrollmentYear = (String)
dateOfEnrollmentYears.getSelectedItem();
                String dateOfEnrollmentMonth = (String)
dateOfEnrollmentMonths.getSelectedItem();
                String dateOfEnrollmentDay = (String)
dateOfEnrollmentDays.getSelectedItem();
                String dateOfEnrollmentCombined = dateOfEnrollmentYear + "-" +
dateOfEnrollmentMonth + "-" + dateOfEnrollmentDay;


                try {
                    // Converting the input Strings to integer using the
parseIntValue method
                    int enrollmentIDValue = Integer.parseInt(enrollmentStr);
                    int courseDurationValue = Integer.parseInt(courseDurationStr);
                    int tuitionFeeValue = Integer.parseInt(tuitionFeeStr);
                    int numberOfModulesValue =
Integer.parseInt(numberOfModulesStr);
                    int daysPresentValue = Integer.parseInt(daysPresentStr);
                    int creditHoursValue = Integer.parseInt(creditHoursStr);


                    boolean found = false;
                    for (Student student : allStudents) {
                        if (student instanceof Regular &&
student.getEnrollmentId() == enrollmentIDValue) {
```

```
                found = true;
                break;
            }
        }



        if (found) {
            JOptionPane.showMessageDialog(null, "Student with that
Enrollment ID already exist", "Error",
                JOptionPane.INFORMATION_MESSAGE);
        } else {
            Regular newRegularStudent = new
Regular(enrollmentIDValue, dateOfBirthCombined, courseName, studentName,
dateOfEnrollmentCombined, courseDurationValue, tuitionFeeValue,
numberOfModulesValue, creditHoursValue, daysPresentValue);
            allStudents.add(newRegularStudent);


            JOptionPane.showMessageDialog(null, "Student has
been added", "Success",
                JOptionPane.INFORMATION_MESSAGE);
        }
    }catch (NumberFormatException a) {
        // To Display the Error Message To the User
```

```
                    JOptionPane.showMessageDialog(openningFrame, "Invalid
input! Please enter a valid integer.");
                }
            }
        });

        // pay bills
        bills.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String dropoutStudentName = dropoutStudentTextField.getText();
                String dropoutEnrollmentID = dropoutEnrollmentTextField.getText();
                String dropoutCourseDurationText =
dropoutCourseDuration.getText();
                String dropoutTuitionFeeText = dropoutTuitionFee.getText();
                String dropoutRemainingModulesText =
dropoutRemainingModules.getText();
                String dropoutMonthAttendedText =
dropoutMonthAttendedTextField.getText();

                // Getting the selected date values for Date of Birth
                String dropoutDateOfBirthYear = (String)
dateOfBirthYearsDropout.getSelectedItem();
```

```
String dropoutDateOfBirthMonth = (String) dateOfBirthMonthsDropout.getSelectedItem();

String dropoutDateOfBirthDay = (String) dateOfBirthDaysDropout.getSelectedItem();

String dateOfBirthCombined = dropoutDateOfBirthYear + "-" + dropoutDateOfBirthMonth + "-" + dropoutDateOfBirthDay;


// Getting the selected date values for Date of Birth

String dropoutDateOfEnrollmentYear = (String) dateOfEnrollmentYearsDropout.getSelectedItem();

String dropoutDateOfEnrollmentMonth = (String) dateOfEnrollmentMonthsDropout.getSelectedItem();

String dropoutDateOfEnrollmentDay = (String) dateOfEnrollmentDaysDropout.getSelectedItem();

String dateOfEnrollmentCombined = dropoutDateOfEnrollmentYear + "-" + dropoutDateOfEnrollmentMonth + "-" + dropoutDateOfEnrollmentDay;


// Getting the selected date values for Date of Birth

String dropoutYears = (String) dropoutYear.getSelectedItem();

String dropoutMonths = (String) dropoutMonth.getSelectedItem();

String dropoutDays = (String) dropoutDay.getSelectedItem();

String dateOfDropoutCombined = dropoutYears + "-" + dropoutMonths + "-" + dropoutDays;


try {
```

```
// Converting the input Strings to integer using the
parseIntValue method
int dropoutEnrollmentIDValue =
Integer.parseInt(dropoutEnrollmentID);
int dropoutCourseDurationValue =
Integer.parseInt(dropoutCourseDurationText);
int dropoutTuitionFeeValue =
Integer.parseInt(dropoutTuitionFeeText);
int dropoutRemainingModules =
Integer.parseInt(dropoutRemainingModulesText);
int dropoutMonthAttended =
Integer.parseInt(dropoutMonthAttendedText);



boolean found = false;
for (Student student : allStudents) {
    if (student instanceof Regular &&
student.getEnrollmentId() == dropoutEnrollmentIDValue) {
        found = true;
        if (sharedDropoutStudent == null) {
            sharedDropoutStudent = new
Dropout(dateOfBirthCombined, dropoutStudentName, dropoutCourseDurationValue,
dropoutTuitionFeeValue, dropoutRemainingModules, dropoutMonthAttended,
dateOfDropoutCombined);
        }
```

```
                    sharedDropoutStudent.setHasPaid(true);
                    JOptionPane.showMessageDialog(null, "All bills cleared!", "Success", JOptionPane.INFORMATION_MESSAGE);
                }
            }

            if (!found) {
                JOptionPane.showMessageDialog(null, "Did not find student", "Error", JOptionPane.INFORMATION_MESSAGE);
            }
        }catch (NumberFormatException a) {
            // To Display the Error Message To the User
            JOptionPane.showMessageDialog(openningFrame, "Invalid input! Please enter a valid integer.");
        }
    }
});

// percentage calc button
percentage.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Getting the input values as Strings
```

```
String enrollmentStr = enrollmentIDTextField.getText();

String courseDurationStr = courseDurationTextField.getText();

String tuitionFeeStr = tuitionFeeTextField.getText();

String numberOfModulesStr =
numberOfModulesTextField.getText();

String daysPresentStr = daysPresentTextField.getText();

String creditHoursStr = numberOfCreditHoursTextField.getText();


// Getting the selected date values for Date of Birth

String dateOfBirthYear = (String)
dateOfBirthYears.getSelectedItem();

String dateOfBirthMonth = (String)
dateOfBirthMonths.getSelectedItem();

String dateOfBirthDay = (String)
dateOfBirthDays.getSelectedItem();

String dateOfBirthCombined = dateOfBirthYear + "-" +
dateOfBirthMonth + "-" + dateOfBirthDay;


// Getting the selected date values for Date of Enrollment

String dateOfEnrollmentYear = (String)
dateOfEnrollmentYears.getSelectedItem();

String dateOfEnrollmentMonth = (String)
dateOfEnrollmentMonths.getSelectedItem();

String dateOfEnrollmentDay = (String)
dateOfEnrollmentDays.getSelectedItem();
```

```
                    String dateOfEnrollmentCombined = dateOfEnrollmentYear + "-" +
dateOfEnrollmentMonth + "-" + dateOfEnrollmentDay;


                    try {
                        // Converting the input Strings to integer using the
parseIntValue method
                        int enrollmentIDValue = Integer.parseInt(enrollmentStr);
                        int daysPresentValue = Integer.parseInt(daysPresentStr);



                        boolean found = false;
                        char grade = 0;
                        for (Student student : allStudents) {
                            if (student instanceof Regular &&
student.getEnrollmentId() == enrollmentIDValue) {
                                Regular regularStudent = (Regular) student;
                                found = true;
                                grade =
regularStudent.presentPrecentage(daysPresentValue);
                            }
                        }


                        if (found) {
```

```
                    JOptionPane.showMessageDialog(null, "Congratulation
your grade is: " + grade, "Success", JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(null, "Did not find
student", "Error", JOptionPane.INFORMATION_MESSAGE);
                }
            }catch (NumberFormatException a) {
                // To Display the Error Message To the User
                JOptionPane.showMessageDialog(openningFrame, "Invalid
input! Please enter a valid integer.");
            }
        }
    });

    // certificate
    certification.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Getting the input values as Strings
            String enrollmentStr = enrollmentIDTextField.getText();
            String daysPresentStr = daysPresentTextField.getText();

            // Getting the selected date values for Date of Birth
```

```
String dateOfBirthYear = (String) dateOfBirthYears.getSelectedItem();

String dateOfBirthMonth = (String) dateOfBirthMonths.getSelectedItem();

String dateOfBirthDay = (String) dateOfBirthDays.getSelectedItem();

String dateOfBirthCombined = dateOfBirthYear + "-" + dateOfBirthMonth + "-" + dateOfBirthDay;


// Getting the selected date values for Date of Enrollment

String dateOfEnrollmentYear = (String) dateOfEnrollmentYears.getSelectedItem();

String dateOfEnrollmentMonth = (String) dateOfEnrollmentMonths.getSelectedItem();

String dateOfEnrollmentDay = (String) dateOfEnrollmentDays.getSelectedItem();

String dateOfEnrollmentCombined = dateOfEnrollmentYear + "-" + dateOfEnrollmentMonth + "-" + dateOfEnrollmentDay;


try {

    // Converting the input Strings to integer using the parseIntValue method

    int enrollmentIDValue = Integer.parseInt(enrollmentStr);

    int daysPresentValue = Integer.parseInt(daysPresentStr);
```

```
                    boolean found = false;
                    boolean isgranted = false;
                    for (Student student : allStudents) {
                        if (student instanceof Regular &&
student.getEnrollmentId() == enrollmentIDValue) {
                            Regular regularStudent = (Regular) student;
                            found = true;
                            char grade =
regularStudent.presentPrecentage(daysPresentValue);
                            if (grade == 'A') {
                                isgranted = true;
                            }
                        }
                    }

                    if (found) {
                        if (isgranted) {
                            JOptionPane.showMessageDialog(null, "Certificate
Granted!", "Success", JOptionPane.INFORMATION_MESSAGE);
                        } else {
                            JOptionPane.showMessageDialog(null, "Certificate
not Granted!", "Error", JOptionPane.INFORMATION_MESSAGE);
                        }
                    } else {
```

```
                    JOptionPane.showMessageDialog(null, "Did not find
student", "Error", JOptionPane.INFORMATION_MESSAGE);
                }
            }catch (NumberFormatException a) {
                // To Display the Error Message To the User
                JOptionPane.showMessageDialog(openningFrame, "Invalid
input! Please enter a valid integer.");
            }
        }
    });
    adding.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String dropoutStudent = dropoutStudentTextField.getText();
            String dropoutEnrollmentID = dropoutEnrollmentTextField.getText();
            String dropoutCourseDurationText =
dropoutCourseDuration.getText();
            String dropoutTuitionFeeText = dropoutTuitionFee.getText();
            String dropoutRemainingModulesText =
dropoutRemainingModules.getText();
            String dropoutMonthAttendedText =
dropoutMonthAttendedTextField.getText();


            // Getting the selected date values for Date of Birth
```

```
String dropoutDateOfBirthYear = (String) dateOfBirthYearsDropout.getSelectedItem();

String dropoutDateOfBirthMonth = (String) dateOfBirthMonthsDropout.getSelectedItem();

String dropoutDateOfBirthDay = (String) dateOfBirthDaysDropout.getSelectedItem();

String dateOfBirthCombined = dropoutDateOfBirthYear + "-" + dropoutDateOfBirthMonth + "-" + dropoutDateOfBirthDay;


// Getting the selected date values for Date of Birth

String dropoutDateOfEnrollmentYear = (String) dateOfEnrollmentYearsDropout.getSelectedItem();

String dropoutDateOfEnrollmentMonth = (String) dateOfEnrollmentMonthsDropout.getSelectedItem();

String dropoutDateOfEnrollmentDay = (String) dateOfEnrollmentDaysDropout.getSelectedItem();

String dateOfEnrollmentCombined = dropoutDateOfEnrollmentYear + "-" + dropoutDateOfEnrollmentMonth + "-" + dropoutDateOfEnrollmentDay;


// Getting the selected date values for Date of Birth

String dropoutYears = (String) dropoutYear.getSelectedItem();

String dropoutMonths = (String) dropoutMonth.getSelectedItem();

String dropoutDays = (String) dropoutDay.getSelectedItem();

String dateOfDropoutCombined = dropoutYears + "-" + dropoutMonths + "-" + dropoutDays;
```

```
                    try {
                        // Converting the input Strings to integer using the
parseIntValue method
                        int dropoutEnrollmentIDValue =
Integer.parseInt(dropoutEnrollmentID);
                        int dropoutCourseDurationValue =
Integer.parseInt(dropoutCourseDurationText);
                        int dropoutTuitionFeeValue =
Integer.parseInt(dropoutTuitionFeeText);
                        int dropoutRemainingModules =
Integer.parseInt(dropoutRemainingModulesText);
                        int dropoutMonthAttended =
Integer.parseInt(dropoutMonthAttendedText);


                        boolean found = false;
                        for (Student student : allStudents) {
                            if (student instanceof Dropout &&
student.getEnrollmentId() == dropoutEnrollmentIDValue) {
                                found = true;
                                break;
                            }
                        }
```

```
                    if (found) {
                        JOptionPane.showMessageDialog(null, "Student with that
Enrollment ID already exist", "Error",
                            JOptionPane.INFORMATION_MESSAGE);
                    } else {
                        Dropout newDropoutStudent = new
Dropout(dateOfBirthCombined, dropoutStudent, dropoutCourseDurationValue,
dropoutTuitionFeeValue, dropoutRemainingModules, dropoutMonthAttended,
dateOfDropoutCombined);
                        allStudents.add(newDropoutStudent);


                        JOptionPane.showMessageDialog(null, "Student has
been added", "Success",
                            JOptionPane.INFORMATION_MESSAGE);
                    }
                }catch (NumberFormatException a) {
                    // To Display the Error Message To the User
                    JOptionPane.showMessageDialog(openningFrame, "Invalid
input! Please enter a valid integer.");
                }
            }
        });


        // ActionListener for "clearRegular" button
```

```
        clearRegular.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                clearRegularData();
            }
        });

        // ActionListener for "clearDropout" button
        clearDropout.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                clearDropoutData();
            }
        });
    }

    // Method to clear the Data of the Regular Student Data
    public void clearRegularData() {
        // Clearing the text fields
        studentTextField.setText("");
        enrollmentIDTextField.setText("");
```

```
        courseDurationTextField.setText("");
        tuitionFeeTextField.setText("");
        numberOfModulesTextField.setText("");
        courseNameTextField.setText("");
        numberOfCreditHoursTextField.setText("");
        daysPresentTextField.setText("");
        // Resetting the combo boxes
        dateOfBirthYears.setSelectedIndex(0);
        dateOfBirthMonths.setSelectedIndex(0);
        dateOfBirthDays.setSelectedIndex(0);
        dateOfEnrollmentYears.setSelectedIndex(0);
        dateOfEnrollmentMonths.setSelectedIndex(0);
        dateOfEnrollmentDays.setSelectedIndex(0);
    }

    // Method to clear the data for the Dropout Student Data
    public void clearDropoutData() {
        // Clearing the text fields
        dropoutStudentTextField.setText("");
        dropoutEnrollmentTextField.setText("");
        dropoutCourseName.setText("");
        dropoutCourseDuration.setText("");
        dropoutTuitionFee.setText("");
```

```
        dropoutRemainingModules.setText("");
        dropoutMonthAttendedTextField.setText("");

        // Resetting the combo boxes
        dateOfBirthYearsDropout.setSelectedIndex(0);
        dateOfBirthMonthsDropout.setSelectedIndex(0);
        dateOfBirthDaysDropout.setSelectedIndex(0);
        dateOfEnrollmentYearsDropout.setSelectedIndex(0);
        dateOfEnrollmentMonthsDropout.setSelectedIndex(0);
        dateOfEnrollmentDaysDropout.setSelectedIndex(0);
        dropoutYear.setSelectedIndex(0);
        dropoutMonth.setSelectedIndex(0);
        dropoutDay.setSelectedIndex(0);

    }

    // Method to handle actions for Regular Student frame
    public void displayRegularStudentAction() {
        StringBuilder message = new StringBuilder();

        boolean hasChildClassObject = false;
        for (Student student : allStudents) {
            if (student instanceof Regular) {
```

```
                hasChildClassObject = true;
                break;
            }
        }

        if (!hasChildClassObject) {
            JOptionPane.showMessageDialog(regularFrame, "No regular students
found!");
            return;
        }

        for (Student student : allStudents) {
            if (student instanceof Regular) {
                Regular regularStudent = (Regular) student;
                message.append("Student Name:
").append(regularStudent.getStudentName())
                        .append("\nCourse name:
").append(regularStudent.getCourseName())
                        .append("\nEnrollment ID:
").append(regularStudent.getEnrollmentId())
                        .append("\nCourse Duration:
").append(regularStudent.getcourseDuration())
                        .append("\nNumber of Modules:
").append(regularStudent.getNumOfModules())
```

```
                    .append("\nDays Present:
").append(regularStudent.getDaysPresent())
                    .append("\nCredit Hours:
").append(regularStudent.getNumOfCreditHour())
                    .append("\nTuition Fee:
").append(regularStudent.getTuitionFee())
                    .append("\nDate of Birth:
").append(regularStudent.getDateofBirth())
                    .append("\nDate of Enrollment:
").append(regularStudent.getdateOfEnrollment())
                    .append("\n");
            }
        }

        String resultMessage = message.toString();

        JOptionPane.showMessageDialog(regularFrame, resultMessage);
    }

    public void displayDropoutAction() {
        StringBuilder message = new StringBuilder();

        boolean hasChildClassObject = false;
```

```
        for (Student student : allStudents) {
            if (student instanceof Dropout) {
                hasChildClassObject = true;
                break;
            }
        }

        if (!hasChildClassObject) {
            JOptionPane.showMessageDialog(regularFrame, "No dropout students
found!");
            return;
        }

        for (Student student : allStudents) {
            if (student instanceof Dropout) {
                Dropout dropoutStudent = (Dropout) student;
                message.append("Student Name:
").append(dropoutStudent.getStudentName())
                        .append("\nCourse Name:
").append(dropoutStudent.getCourseName())
                        .append("\nDate of Birth:
").append(dropoutStudent.getDateofBirth())
                        .append("\nEnrollment ID:
").append(dropoutStudent.getEnrollmentId())
```

```
                                .append("\nDate of Enrollment:
").append(dropoutStudent.getdateOfEnrollment())

                                .append("\nCourse Duration:
").append(dropoutStudent.getcourseDuration())

                                .append("\nTuition Fee:
").append(dropoutStudent.getTuitionFee())

                                .append("\nRemaining Modules:
").append(dropoutStudent.getNumOfRemainingModules())

                                .append("\nMonth Attended:
").append(dropoutStudent.getNumofMonthAttended())

                                .append("\nDate of Dropout:
").append(dropoutStudent.getDateOfDropout());

                        }

                }

                String resultMessage = message.toString();

                JOptionPane.showMessageDialog(dropoutFrame, resultMessage);

        }


        public static void main(String[] args) {

                new StudentGUI();

        }

}
```