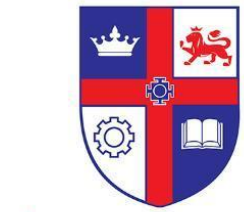
LONDON METROPOLITAN UNIVERSITY



islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework - 2

2023-24 Spring

Student Name: Shirshak Aryal

London Met ID: 22085620

College ID: NP01CP4S230122

Group: C21

Assignment Due Date: Friday, August 11, 2023

Assignment Submission Date: Friday, August 11, 2023

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

Table of Contents

Table of Figures

Table of Tables

1. Introduction

This project is the second coursework of the module CS4001NI Programming, which carries 30% of the final grade of the module. The coursework involves the use of Java to create a program that builds upon the previous coursework and creates a GUI (Graphical User Interface) along with buttons of varying functionalities.

The program consists of a GUI that allows users/admins to add Regular and/or Dropout Students, as well as calculating their attendance grade, granting them scholarships, recording that their bills have been paid, removing them, and finally displaying all the students currently added to the 'database'. It also has the convenience of allowing the user to clear all fields at once, should they want to do so.

The main goal of this project is to familiarize about more advanced concepts in programming and Java, and how they can be used to convenience various real-life problems. This is to be achieved through the following objectives:

a. Creating a class 'StudentGUI' inside the package that contains the previous coursework
b. Using the various JComponents in the Swing package to create a GUI
c. Using Listener Interfaces from the AWT (Abstract Window Toolkit) package to add functionality to the buttons
d. Applying the concepts of object upcasting and downcasting where appropriate
e. Applying the concepts of Array lists and the ways to iterate through them using 'for-each' loop

1.1. Tools Used

1.1.1. BlueJ

BlueJ is an IDE (Integrated Development Environment) that allows users to develop Java programs quickly and easily. It has a simple, minimalistic GUI specifically designed for beginners so that they do not get overwhelmed and start learning Java programming more easily. (Anon., 2021)

BlueJ was used to create and operate all of the Java programs in this project.

1.1.2. MS-Word

MS-Word is a word-processing software developed by Microsoft Corporation (Anon., 2023). It is the most widely used word-processing program in the world.

MS-Word was used to formulate this report for the project.

1.1.3. Draw.io

draw.io is a technology framework designed to create diagramming applications. It holds itself as the most-used browser-based diagramming software among end users worldwide. (Anon., 2023)

draw.io was used to create class diagrams that represent the structure of the programs in the project.

2. Class Diagrams

Class diagrams are static representations of the overview of the structure and relationship among the classes involved in a program. They include all the attributes and methods used in the classes.

Class diagrams for the classes in this project are shown below:

2.1. Student class

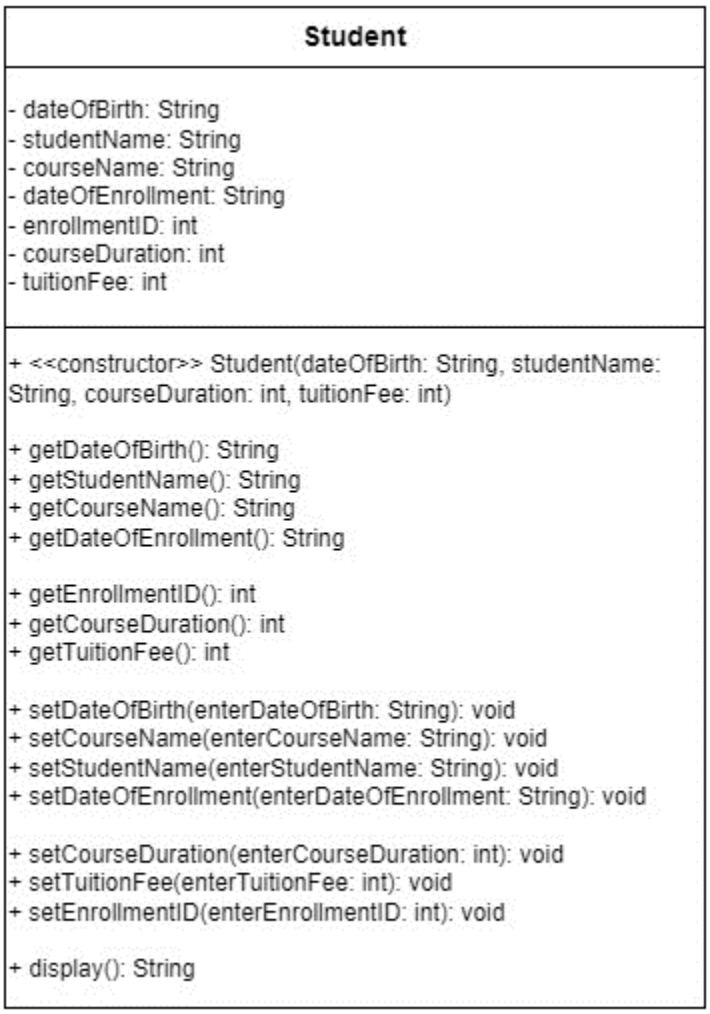


Figure 1: Class Diagram of Student class

2.2. Regular class

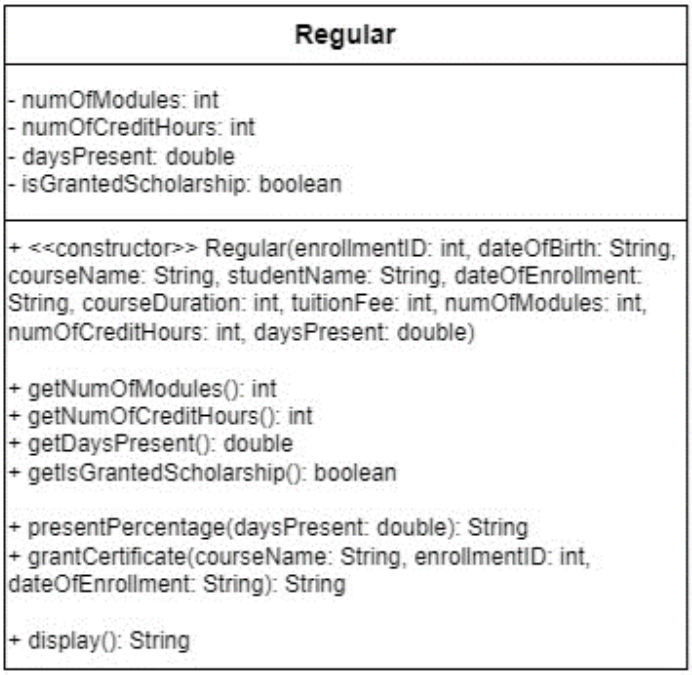


Figure 2: Class Diagram of Regular class

2.3. Dropout class

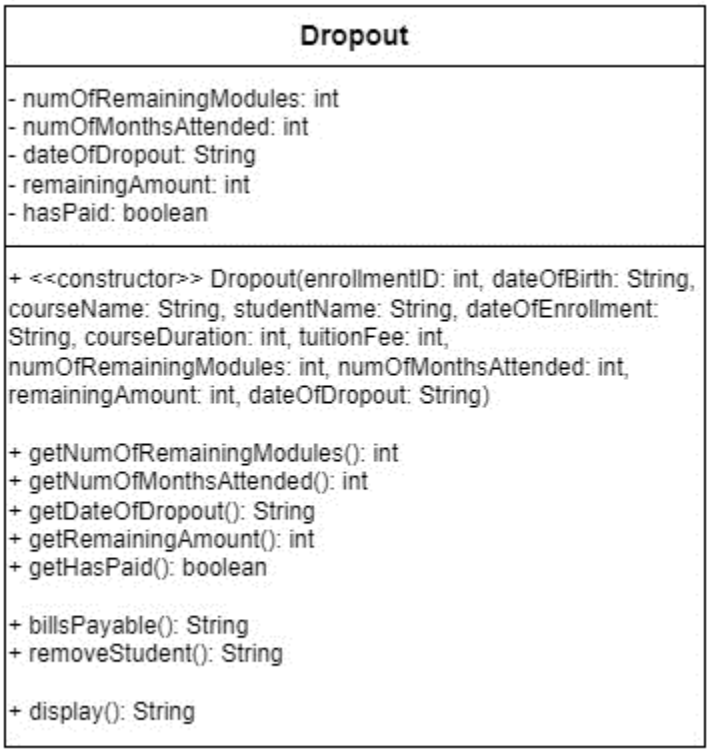


Figure 3: Class Diagram of Dropout class

2.4. StudentGUI class

Figure 4: Class Diagram of StudentGUI class

2.5. Combined class diagram

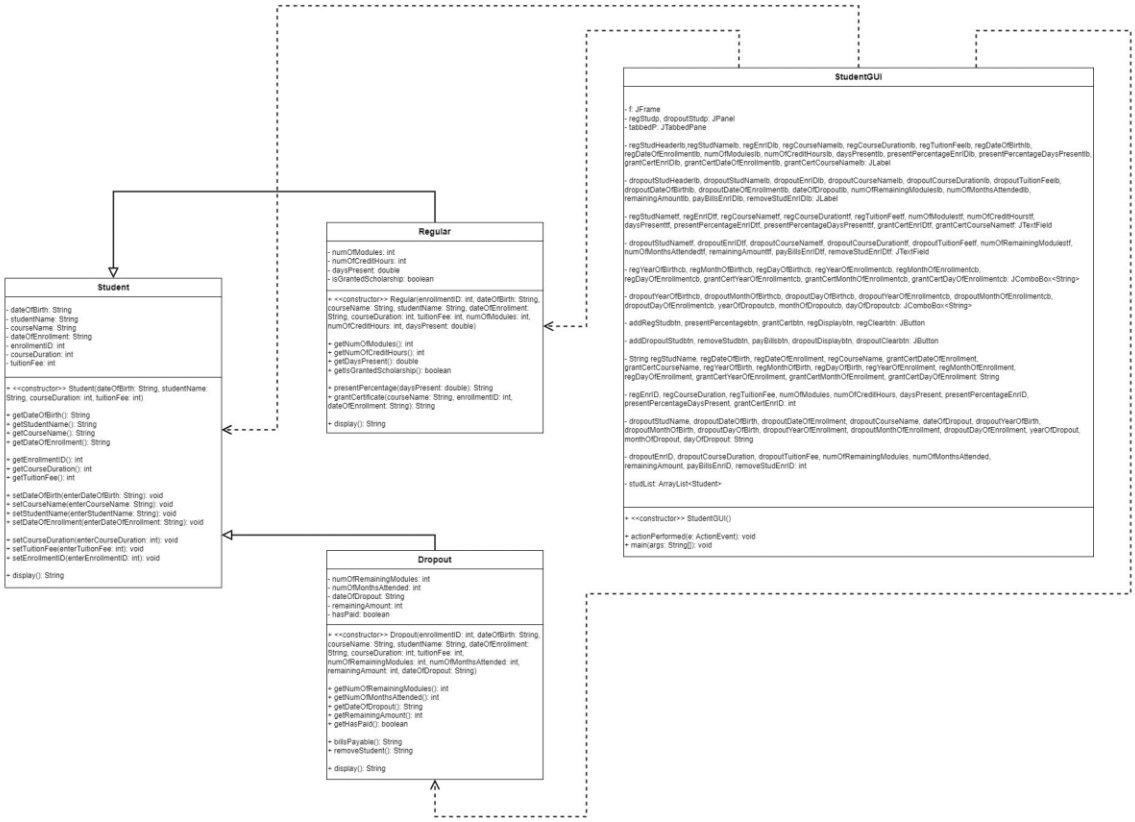


Figure 5: Combined Class Diagram

3. Pseudocode

3.1. StudentGUI class

IMPORT the required classes from AWT package

IMPORT the required classes from Swing package

IMPORT the ArrayIlst class from utility package

CREATE a class StudentGUI that **implements** the ActionListener class

DO

DECLARE f as a JFrame object with private access modifier

DECLARE regStudp and dropoutStudp as JPanel objects with private access modifiers

DECLARE tabbedP as a JTabbedPane object with private access modifier

DECLARE regStudHeaderlb, regStudNamelb, regEnrIDlb, regCourseNamelb, regCourseDurationlb, regTuitionFeelb, regDateOfBirthlb, regDateOfEnrollmentlb, numOfModuleslb, numOfCreditHourslb, daysPresentlb, presentPercentageEnrIDlb, presentPercentageDaysPresentlb, grantCertEnrIDlb, grantCertDateOfEnrollmentlb and grantCertCourseNamelb as JLabel objects with private access modifiers

DECLARE dropoutStudHeaderlb, dropoutStudNamelb, dropoutEnrIDlb, dropoutCourseNamelb, dropoutCourseDurationlb, dropoutTuitionFeelb, dropoutDateOfBirthlb, dropoutDateOfEnrollmentlb, dateOfDropoutlb, numOfRemainingModuleslb, numOfMonthsAttendedlb, remainingAmountlb, payBillsEnrIDlb, removeStudEnrIDlb as JLabel objects with private access modifiers

DECLARE regStudNametf, regEnrIDtf, regCourseNametf, regCourseDurationtf, regTuitionFeetf, numOfModulestf, numOfCreditHourstf, daysPresenttf,

presentPercentageEnrIDtf, presentPercentageDaysPresenttf, grantCertEnrIDtf, grantCertCourseNametf as JTextField objects with private access modifiers

DECLARE dropoutStudNametf, dropoutEnrIDtf, dropoutCourseNametf, dropoutCourseDurationtf, dropoutTuitionFeetf, numOfRemainingModulestf, numOfMonthsAttendedtf, remainingAmounttf, payBillsEnrIDtf, removeStudEnrIDtf as JTextField objects with private access modifiers

DECLARE regYearOfBirthcb, regMonthOfBirthcb, regDayOfBirthcb, regYearOfEnrollmentcb, regMonthOfEnrollmentcb, regDayOfEnrollmentcb, grantCertYearOfEnrollmentcb, grantCertMonthOfEnrollmentcb, grantCertDayOfEnrollmentcbf as JComboBox objects of String datatype with private access modifiers

DECLARE dropoutYearOfBirthcb, dropoutMonthOfBirthcb, dropoutDayOfBirthcb, dropoutYearOfEnrollmentcb, dropoutMonthOfEnrollmentcb, dropoutDayOfEnrollmentcb, yearOfDropoutcb, monthOfDropoutcb, dayOfDropoutcb as JComboBox objects of String datatype with private access modifiers

DECLARE addRegStudbtn, presentPercentagebtn, grantCertbtn,

regDisplaybtn, regClearbtn as JButton objects with private access modifiers

DECLARE addDropoutStudbtn, removeStudbtn, payBillsbtn,

dropoutDisplaybtn, dropoutClearbtn as JButton objects with private access modifiers

DECLARE regStudName, regDateOfBirth, regDateOfEnrollment, regCourseName, grantCertDateOfEnrollment, grantCertCourseName, regYearOfBirth, regMonthOfBirth, regDayOfBirth, regYearOfEnrollment, regMonthOfEnrollment, regDayOfEnrollment, grantCertYearOfEnrollment, grantCertMonthOfEnrollment, grantCertDayOfEnrollment as String variables with private access modifiers

DECLARE regEnrID, regCourseDuration, regTuitionFee, numOfModules, numOfCreditHours, daysPresent, presentPercentageEnrID, presentPercentageDaysPresent, grantCertEnrID as integer variables with private access modifiers

DECLARE dropoutStudName, dropoutDateOfBirth, dropoutDateOfEnrollment, dropoutCourseName, dateOfDropout, dropoutYearOfBirth, dropoutMonthOfBirth, dropoutDayOfBirth, dropoutYearOfEnrollment, dropoutMonthOfEnrollment, dropoutDayOfEnrollment, yearOfDropout, monthOfDropout, dayOfDropout as String variables with private access modifiers

DECLARE dropoutEnrID, dropoutCourseDuration, dropoutTuitionFee, numOfRemainingModules, numOfMonthsAttended, remainingAmount,

payBillsEnrID, removeStudEnrID as integer variables with private access modifiers

INITIALIZE studList as an ArrayList of Student datatype

END DO

CREATE the constructor method StudentGUI() without any parameters, with public access modifier

DO

INITIALIZE f with a new JFrame object

INITIALIZE the required JPanel objects

INITIALIZE the required JLabel objects

SET the font of regStudHeaderlb and dropoutStudHeaderlb

INITIALIZE the required JTextField objects

INITIALIZE the required JButton objects

INITIALIZE yearar as an array of String datatype of size fifty

INITIALIZE monthar as an array of String datatype of size twelve

INITIALIZE dayar as an array of String datatype of size thirty-one

INITIALIZE year as an integer variable with the value of nineteen-ninety

FOR i from zero to less than fifty, incremented by one

    SET the value at index ‘i’ of yearar with the String value of year

    INCREMENT year by one

END FOR

INITIALIZE month as an integer variable with the value of one

FOR i from zero to less than or equals to eleven, incremented by one

    SET the values at index ‘i’ of monthar with the String value of month

    INCREMENT month by one

END FOR

INITIALIZE day as an integer variable with the value of one

FOR i from zero to less than or equals to thirty, incremented by one

    SET the values at index ‘i’ of dayar with the String value of day

    INCREMENT day by one

END FOR

INITIALIZE the required JComboBox objects

SET the size and position of the required components

ADD action listeners to the JButton objects

ADD the required components to regStudp

ADD the required components to dropoutStudp

SET background colors of the JPanel and JButton objects

SET border colors of the JTextField and JButton objects

SET the size of regStudp, dropoutStudp

SET the layout of regStudp, dropoutStudp to null

INITIALIZE the required JTabbedPane object

ADD regStudp, dropoutStudp to the tabbed pane

SET the visibility of the tabbed pane to true

SET the size and position of the tabbed pane

ADD the tabbed pane to f

SET the size of f

SET the layout of f to null

SET the resizable property of f to false

SET the visibility of f to true

SET the default close operation of f to exit on close

END DO

CREATE a method actionPerformed() with ActionEvent object as parameter, with public access modifier and void return type

DO

IF the source of ActionEvent is equal to addRegStudbtn

DO

IF any of the required text fields is empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

GET the text from regStudNametf

GET the selected items from regYearOfBirthcb, regMonthOfBirthcb, regDayOfBirthcb and convert them into String values

GET the selected items from regYearOfEnrollmentcb, regMonthOfEnrollmentcb, regDayOfEnrollmentcb and convert them into String values

GET the text from regCourseNametf

TRY

DO

GET the text from regEnrIDtf, regCourseDurationtf, regTuitionFeetf, numOfModulestf, numOfCreditHourstf, daysPresenttf and convert them into integer values

CREATE a new Regular object and upcast it

IF regEnrID is a negative value

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

IF studList is empty

DO

ADD the Regular object to studList

DISPLAY a dialog box with the required message

END DO

ELSE

DO

INITIALIZE a boolean variable isDuplicate as false

FOR each student in studList

IF the student is an instance of Regular class

IF the student's enrollment ID is equal to that of the Regular object

DO

SET isDuplicate to true

break;

END DO

END IF

END IF

END FOR

IF isDuplicate is equal to false

DO

ADD the Regular object to studList

DISPLAY a dialog box with the required message

END DO

ELSE

DO

DISPLAY a dialog box with the required message

END DO

END IF

END DO

END IF

END DO

END DO

CATCH

DO

DISPLAY a dialog box with the required message

END DO

END TRY

END DO

END IF

END DO

ELSE IF the source of ActionEvent is equal to presentPercentagebtn

DO

IF any of the required text fields are empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

TRY

DO

GET the text from presentPercentageEnrIDtf and presentPercentageDaysPresenttf and convert them into integers

IF studList is empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

INITIALIZE a boolean variable calculatedPresentPercentage as false

FOR each student in studList

IF the student is an instance of Regular class

DO

DOWNCAST the student

IF the student's enrollment ID is equal to presentPercentageEnrID

DO

SET calculatedPresentPercentage to true

CALL the presentPercentage() method on the student

DISPLAY a dialog box with the required message

break;

END DO

END IF

END DO

END IF

END FOR

IF calculatedPresentPercentage is equal to false

DO

DISPLAY a dialog box with the required message

END DO

END IF

END DO

END IF

END DO

CATCH

DO

DISPLAY a dialog box with the required message

END DO

END TRY

END DO

END IF

END DO

ELSE IF the source of ActionEvent is equal to grantCertbtn

DO

IF any of the required text fields are empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

GET the text from grantCertCourseName

GET the text from grantCertYearOfEnrollment, grantCertMonthOfEnrollment, grantCertDayOfEnrollment and convert them into String values

TRY

DO

GET the text from grantCertEnrID and convert it into integer value

IF studList is empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

INITIALIZE a boolean variable grantedCertificate as false

FOR each student in studList

IF the student is an instance of Regular class

DO

DOWNCAST the student

IF the student's enrollment ID is equal to presentPercentageEnrID

DO

SET
grantedCertificate to
true

CALL the grantCertificate()
method on the student

DISPLAY a dialog box with the
required message

break;

END DO

END IF

END DO

END IF

END FOR

IF grantedCertificate is equal to false

DO

DISPLAY a dialog box with the required
message

END DO

END IF

END DO

END IF

END DO

CATCH

DO

DISPLAY a dialog box with the required message

END DO

END TRY

END DO

END IF

END DO

ELSE IF the source of ActionEvent is equal to regDisplaybtn

DO

INITIALIZE a boolean variable displayedStudent as false

FOR each student in studList

IF the student is an instance of Regular class

DO

SET displayedStudent to true

DOWNCAST the student

CALL the display() method on the student

DISPLAY a dialog box with the required message

END DO

END IF

END FOR

IF displayedStudent is equal to false

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

DISPLAY a dialog box with the required message

END DO

END IF

END DO

ELSE IF the source of ActionEvent is equal to regClearbtn

DO

DISPLAY a dialog box with the required warning message

INITIALIZE textfieldList as a new ArrayList of JTextField datatype

ADD the required text fields to textfieldList

FOR each text field in textfieldList

DO

SET the text of each text field as empty Strings

END DO

END FOR

END DO

ELSE IF the source of ActionEvent is equal to addDropoutStudbtn

DO

IF any of the required text fields is empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

GET the text from dropoutStudNametf

GET the selected items from dropoutYearOfBirthcb, dropoutMonthOfBirthcb, dropoutDayOfBirthcb and convert them into String values

GET the selected items from dropoutYearOfEnrollmentcb, dropoutMonthOfEnrollmentcb, dropoutDayOfEnrollmentcb and convert them into String values

GET the selected items from yearOfDropoutcb, monthOfDropoutcb, dayOfDropoutcb and convert them into String values

GET the text from dropoutCourseName

TRY

DO

GET the text from dropoutEnrIDtf, dropoutCourseDurationtf, dropoutTuitionFeetf, numOfRemainingModulestf, numOfMonthsAttendedtf, remainingAmounttf and convert them into integer values

CREATE a new Dropout object and upcast it

IF dropoutEnrID is a negative value

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

IF studList is empty

DO

ADD the Dropout object to studList

DISPLAY a dialog box with the required message

END DO

ELSE

DO

INITIALIZE a boolean variable isDuplicate as false

FOR each student in studList

IF the student is an instance of Dropout class

IF the student's enrollment ID is equal to that of the Dropout object

DO

SET isDuplicate to true

break;

END DO

END IF

END IF

END FOR

IF isDuplicate is equal to false

DO

ADD the Dropout object to studList

DISPLAY a dialog box with the required message

END DO

ELSE

DO

DISPLAY a dialog box with the required message

END DO

END IF

END DO

END IF

END DO

END IF

END DO

CATCH

DO

DISPLAY a dialog box with the required message

END DO

END TRY

END DO

END IF

END DO

ELSE IF the source of ActionEvent is equal to payBillsbtn

DO

IF any of the required text fields are empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

TRY

DO

GET the text from payBillsEnrIDtf and convert it into integer value

IF studList is empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

INITIALIZE a boolean variable paidBills as false

FOR each student in studList

IF the student is an instance of Dropout class

DO

DOWNCAST the student

IF the student's enrollment ID is equal to payBillsEnrID

DO

SET
paidBills to
true

CALL the billsPayable()
method on the student

DISPLAY a dialog box with the
required message

break;

END DO

END IF

END DO

END IF

END FOR

IF paidBills is equal to false

DO

DISPLAY a dialog box with the required
message

END DO

END IF

END DO

END IF

END DO

CATCH

DO

DISPLAY a dialog box with the required message

END DO

END TRY

END DO

END IF

END DO

ELSE IF the source of ActionEvent is equal to removeStudbtn

DO

IF any of the required text fields are empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

TRY

DO

GET the text from removeStudEnrIDtf and convert it into integer value

IF studList is empty

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

INITIALIZE a boolean variable isRemoved as false

FOR each student in studList

IF the student is an instance of Dropout class

DO

DOWNCAST the student

IF the student's enrollment ID is equal to removeStudEnrID

DO

SET
isRemoved to
true

CALL the removeStudent()
method on the student

DISPLAY a dialog box with the
required message

break;

END DO

END IF

END DO

END IF

END FOR

IF isRemoved is equal to false

DO

DISPLAY a dialog box with the required
message

END DO

END IF

END DO

END IF

END DO

CATCH

DO

DISPLAY a dialog box with the required message

END DO

END TRY

END DO

END IF

END DO

ELSE IF the source of ActionEvent is equal to dropoutDisplaybtn

DO

INITIALIZE a boolean variable displayedStudent as false

FOR each student in studList

IF the student is an instance of Dropout class

DO

SET displayedStudent to true

DOWNCAST the student

CALL the display() method on the student

DISPLAY a dialog box with the required message

END DO

END IF

END FOR

IF displayedStudent is equal to true

DO

DISPLAY a dialog box with the required message

END DO

ELSE

DO

DISPLAY a dialog box with the required message

END DO

END IF

END DO

ELSE

DO

DISPLAY a dialog box with the required warning message

INITIALIZE textfieldList as a new ArrayList of JTextField datatype

ADD the required text fields to textfieldList

FOR each text field in textfieldList

DO

SET the text of each text field as empty Strings

END DO

END FOR

END DO

END IF

END DO

CREATE a static method main() with String[] args as parameter, with public access modifier and void return type

DO

CALL StudentGUI() constructor

END DO

4. Method description

4.1. actionPerformed(ActionEvent e)

This method takes an ActionEvent object ‘e’ a a parameter, and contains the functionalities of all the JButtons, listed below:

4.1.1. Add Regular Student button

If this button is clicked, the method first checks if there are any empty text fields relevant to the Regular() constructor that might not have been filled. If so, it displays an error message. If not, it retrieves the text filled in each of those text fields and stores them in separate variables, implementing a try block to catch NumberFormatExceptions where the data retrieved needs to be converted into integer form. Then it calls the Regular() constructor and passes the variables in the required order, and upcasts the new Regular object as that of the Student class.

It then checks if the enrollment ID is negative. If so, it displays an error message to the user. If not, it then checks if the Student array list is empty. If so, it adds the Regular object to that array list, and displays a message to the user indicating that the student has been added. If not, it iterates through the array list to search for Regular objects to check if any enrollment IDs match the one entered by the user. If the IDs match, it shows a message to the user, indicating that they have entered a duplicate ID. If the ID is unique, the student is added to the array list, with a message being displayed to indicate so.

If in case the user types in string values inside any text field where the required input is an integer, the catch block is executed, which shows a message to the user indicating the invalidity of the data.

4.1.2. Present Percentage button

If this button is clicked, the method first checks if the required text fields are empty, i.e., an enrollment ID text field and a ‘number of days present’ text field. If so, it displays an indicative message to the user. If not, a try block is opened, where the method retrieves the values entered in the text fields and stores them in separate variables.

Then the method checks if the Student array list is empty. If so, it shows a message to the user implying that the list is empty. Otherwise, it iterates through the array list to search for any Regular objects with an enrollment ID that matches the one entered by the user. If so, it first downcasts that object into that of the Regular class, calls the presentPercentage() method from the Regular class on that object, displays a message to show the attendance grade, and breaks the iteration loop.

If there are no Regular students in the Student array list, it shows a message to the user, conveying the situation to them.

If by any chance the user enters string values in any one of the two text fields, the method displays a message to the user, informing them to enter numbers instead of other values.

4.1.3. Grant Certificate button

If this button is clicked, the method firstly checks if the two required text fields, i.e., an ‘enrollment ID’ text field and a ‘course name’ text field, are empty, and informs the user if that is the case. If not, the method retrieves the values entered in those text fields as well as the selected date of enrollment and stores them in separate variables (the retrieval of the value in the enrollment ID text field is done inside a try block).

The method then checks if the Student array list is empty, and informs the user if so. If not, it iterates through the array list to search for Regular objects. If found, it then checks for that Regular object whose enrollment ID matches the one provided by the user. If a match is found, that object is downcasted into that of the Regular class, grantCertificate() method from that class is called on that object, and a message is shown to the user that displays the details of that Regular object, as well as a message that says the student has graduated. It also says if that student has been provided with a scholarship, depending upon their attendance grade.

If no Regular objects are found in the Student array list, a message indicative of the situation is displayed to the user.

If the user enters string values in the enrollment ID text field by any means, an error message is displayed that informs the user that the data is invalid.

4.1.4. Display button (Regular class)

If this button is clicked, the method first iterates through the Student array list to search for Regular objects. If found, it downcasts each of those Regular objects as those of the Regular class, and calls the display() method from that class on them. It then shows a message to the user that displays every detail of those students.

If in case no Regular objects are found in the array list, a message is displayed to the user to convey it.

4.1.5. Clear button (Regular class)

If this button is clicked, it first shows a warning message to the user that informs them that all of the textfields are going to be cleared. After the user clicks ‘Ok’, the method sets the values of all the textfields as empty strings.

4.1.6. Add Dropout Student button

If this button is clicked, the method first checks if there are any empty text fields relevant to the Dropout() constructor that might not have been filled. If so, it displays an error message. If not, it retrieves the text filled in each of those text fields and stores them in separate variables, implementing a try block to catch NumberFormatExceptions where the data retrieved needs to be converted into integer form. Then it calls the Dropout() constructor and passes the variables in the required order, and upcasts the new Dropout object as that of the Student class.

It then checks if the enrollment ID is negative. If so, it displays an error message to the user. If not, it then checks if the Student array list is empty. If so, it adds the Dropout object to that array list, and displays a message to the user indicating that the student has been added. If not, it iterates through the array list to search for Dropout objects to check

if any enrollment IDs match the one entered by the user. If the IDs match, it shows a message to the user, indicating that they have entered a duplicate ID. Otherwise if the ID is unique, the student is added to the array list, with a message being displayed to indicate so.

If in case the user types in string values inside any text field where the required input is an integer, the catch block is executed, which shows a message to the user indicating the invalidity of the data.

4.1.7. Pay Bills button

If this button is clicked, the method firstly checks if the required text field, i.e. an enrollmentID text field, is empty, and informs the user if that is the case. If not, the method retrieves the value entered in that text field and stores it in a variable, inside a try block.

The method then checks if the Student array list is empty, and informs the user if so. If not, it iterates through the array list to search for Dropout objects. If found, it then checks for that Dropout object whose enrollment ID matches the one provided by the user. If a match is found, that object is downcasted into that of the Dropout class, the billsPayable() method from that class is called on that object, and a message is shown to the user that displays the details of that Dropout object, as well as a message that displays the remaining amount of fees of that student.

If no Dropout objects are found in the Student array list, a message indicative of the situation is displayed to the user.

If the user enters string values in the enrollment ID text field by any means, an error message is displayed that informs the user that the data is invalid.

4.1.8. Remove Student button

If this button is clicked, the method firstly checks if the required text field, i.e. an enrollment ID text field, is empty, and informs the user if that is the case. If not, the method retrieves the value entered in that text field and stores it in a variable, inside a try block.

The method then checks if the Student array list is empty, and informs the user if so. If not, it iterates through the array list to search for Dropout objects. If found, it then checks for that Dropout object whose enrollment ID matches the one provided by the user. If a match is found, that object is downcasted into that of the Dropout class, the removeStudent() method from that class is called on that object, and a message is shown to the user that is dependent on whether the bills have already been paid for that student. If so, the message shows that all of the attributes of that student have been set to either empty strings or 0. If not, the message says that the bills have not been cleared.

If no Dropout objects are found in the Student array list, a message indicative of the situation is displayed to the user.

If the user enters string values in the enrollment ID text field by any means, an error message is displayed that informs the user that the data is invalid.

4.1.9. Display button (Dropout class)

If this button is clicked, the method first iterates through the Student array list to search for Dropout objects. If found, it downcasts each of those Dropout objects as those of the Dropout class, and calls the display() method from that class on them. It then shows a message to the user that displays every detail of those students.

If in case no Dropout objects are found in the array list, a message is displayed to the user to convey it.

4.1.10. Clear button (Dropout class)

If this button is clicked, it first shows a warning message to the user that informs them that all of the text fields are going to be cleared. After the user clicks ‘Ok’, the method sets the values of all the text fields as empty strings.

4.2. main (String[] args)

This method creates an object of StudentGUI class by calling its constructor.

5. Testing

5.1. Test 1: To test that the program can be compiled and run using the command prompt

Objective	To test that the program can be compiled and run using the command prompt
Actions Performed	1. The folder containing the StudentGUI.java file was opened. 2. The terminal was opened here by right clicking inside the folder and selecting the 'Open in Terminal' option. 3. The respective commands were written to compile and run the program.
Expected Result	The program should have been complied and the GUI should have been displayed.
Actual Result	The program was complied, and the GUI was displayed.
Conclusion	The test was successful.

Table 1: Test 1: To test that the program can be compiled and run using the command prompt

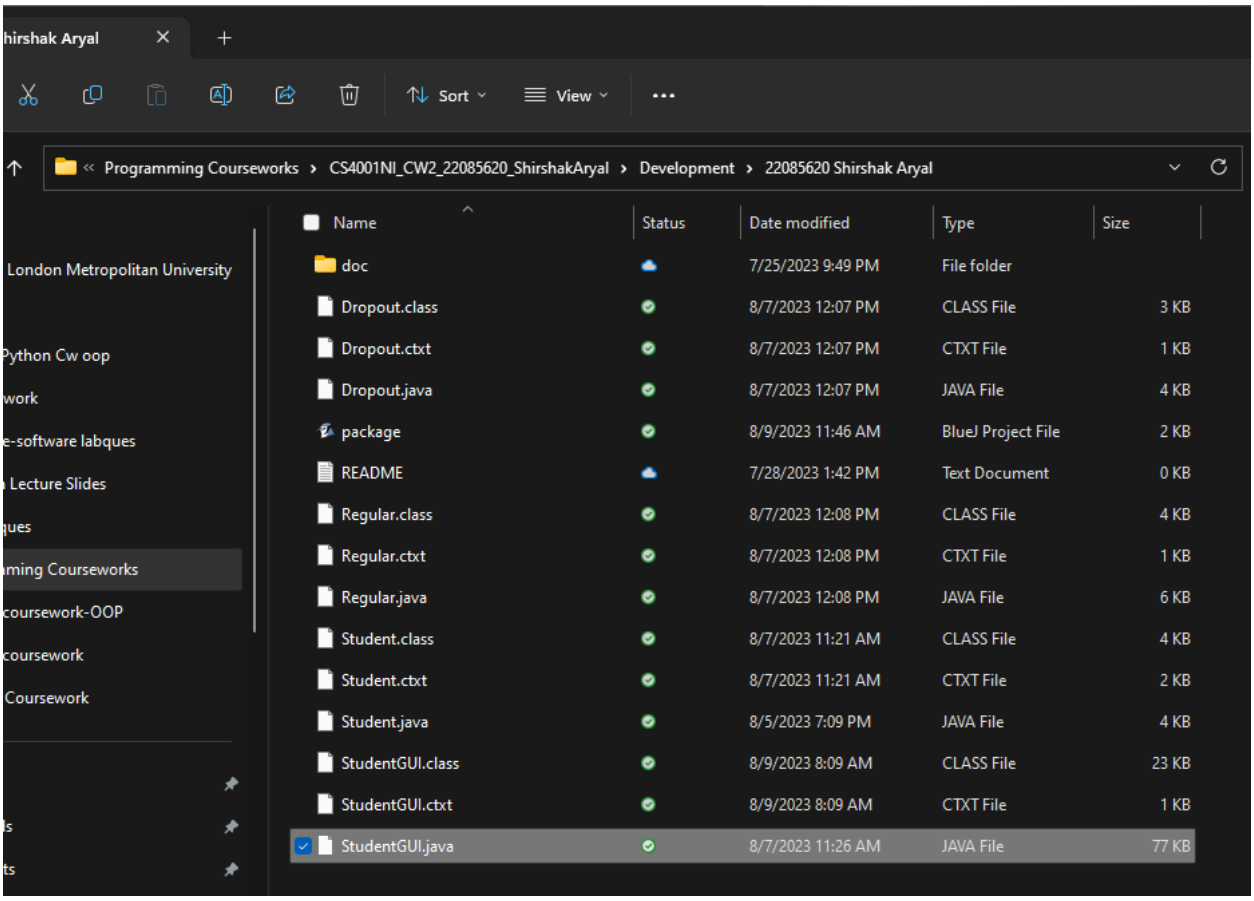


Figure 6: Opening the folder containing the StudentGUI.java file

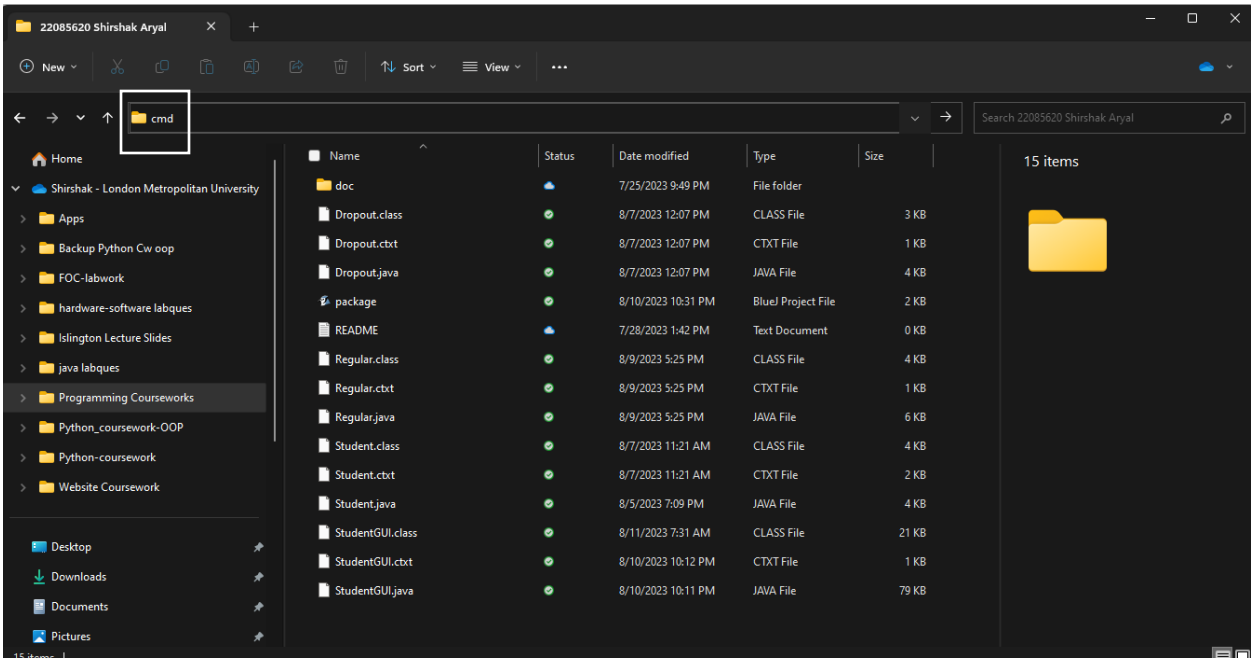


Figure 7: Typing 'cmd' in place of the file path

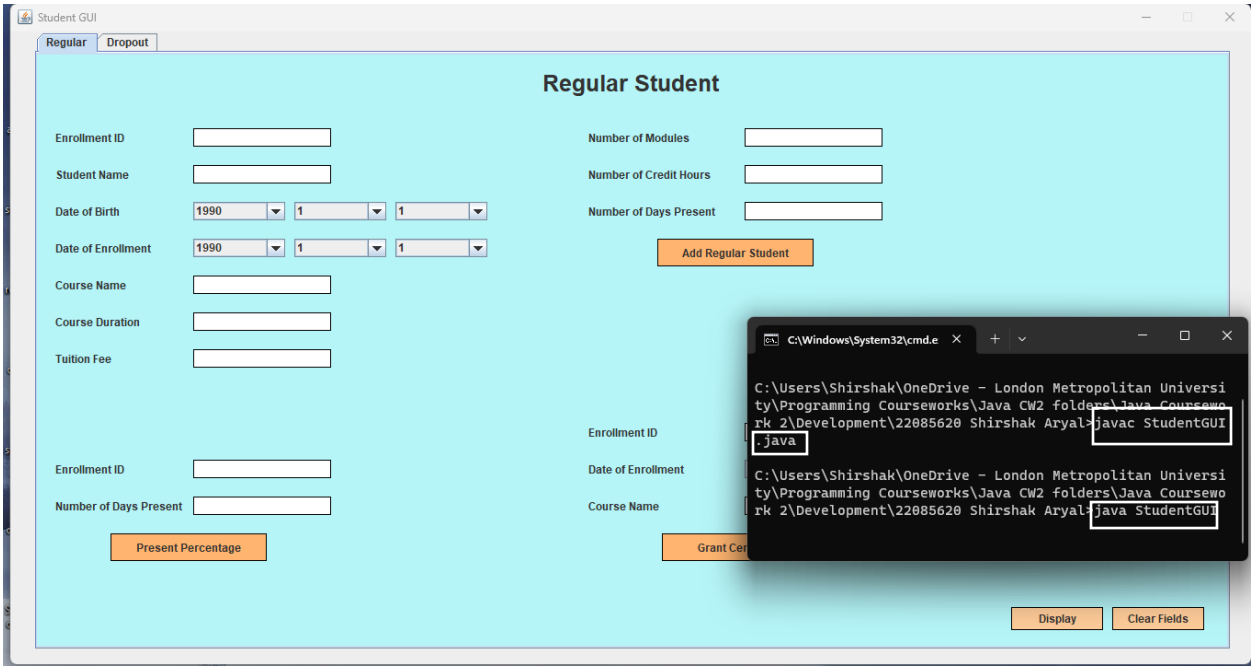


Figure 8: Writing the commands to compile and run the program, and the program running in the background

5.2. Test 2: To test the functionality of buttons

5.2.1. Test 2.a.: To test the ‘Add a Regular Student’ button

Objective	To test the ‘Add a Regular Student’ button
Actions Performed	1. The StudentGUI class was run. 2. The required text fields were filled with valid values. 3. The ‘Add Regular Student’ button was clicked.
Expected Result	A dialog box should have been displayed that said, ‘Successfully added regular student.’
Actual Result	A dialog box was displayed that said, ‘Successfully added regular student.’
Conclusion	The test was successful.

Table 2: Test 2.a.: To test the ‘Add a Regular Student’ button

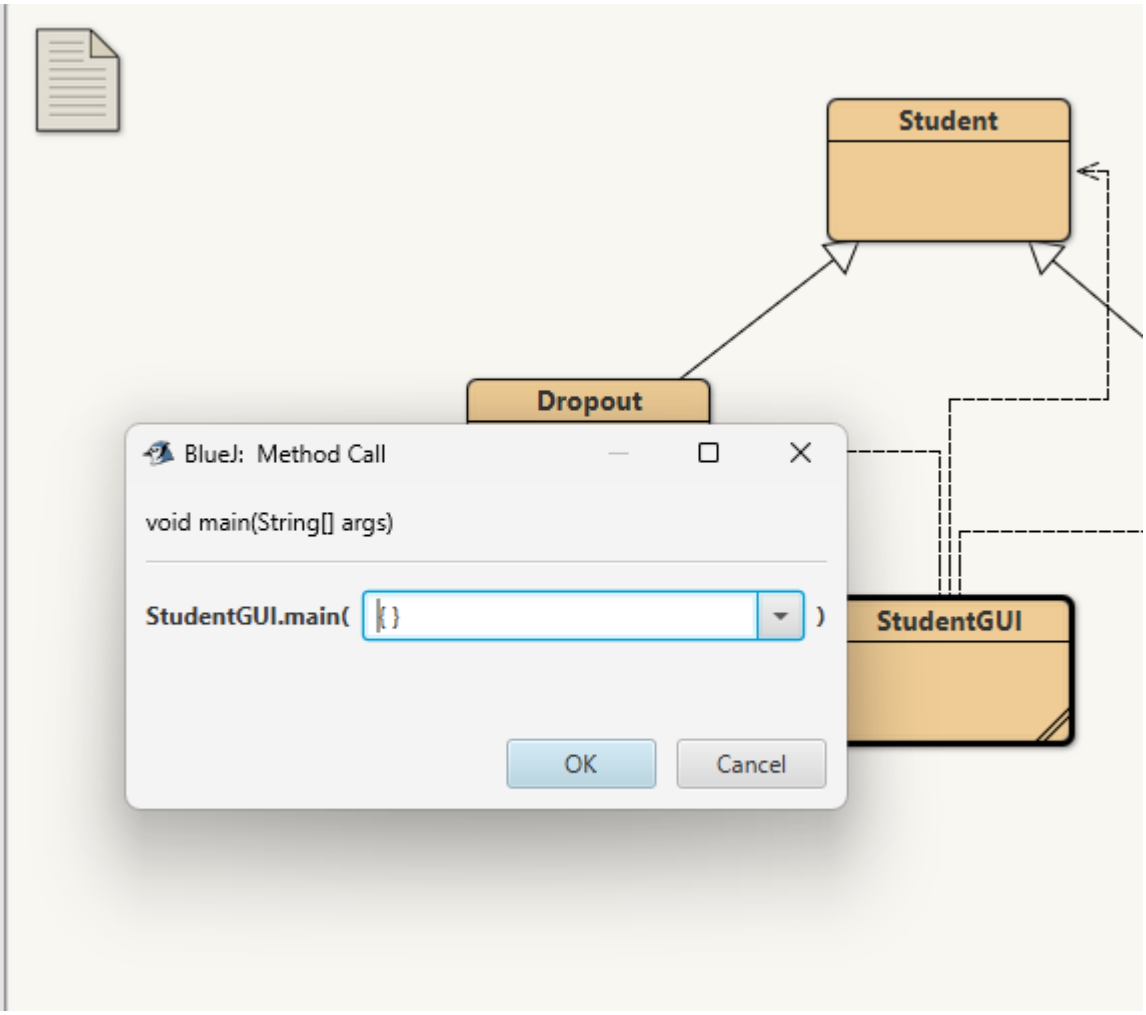


Figure 9: Running the StudentGUI class

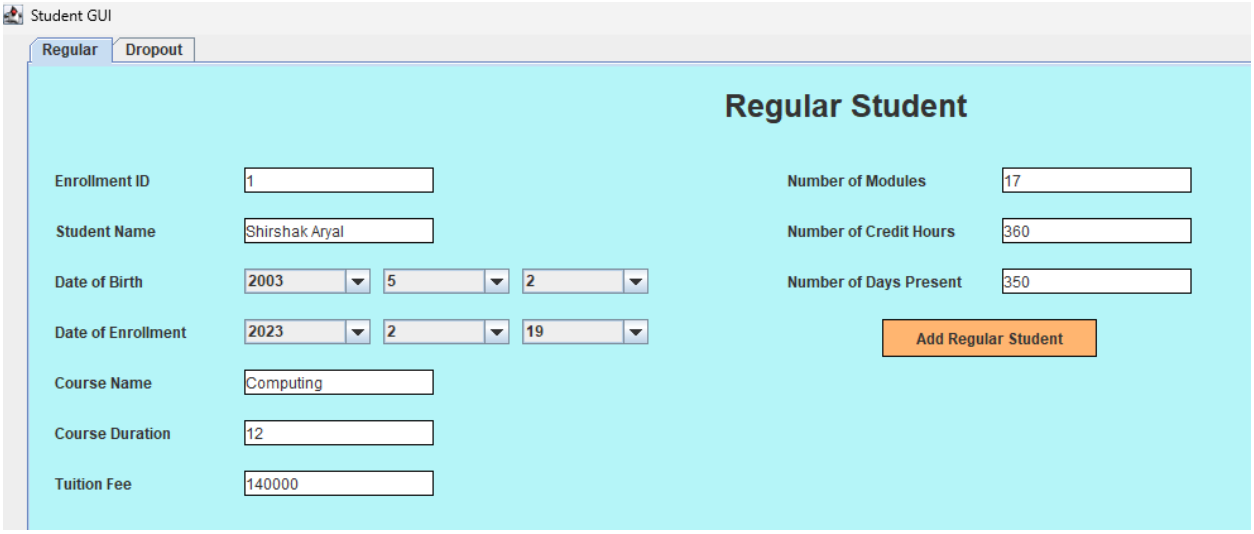


Figure 10: Filling the required text fields with valid values and clicking the 'Add Regular Student' button

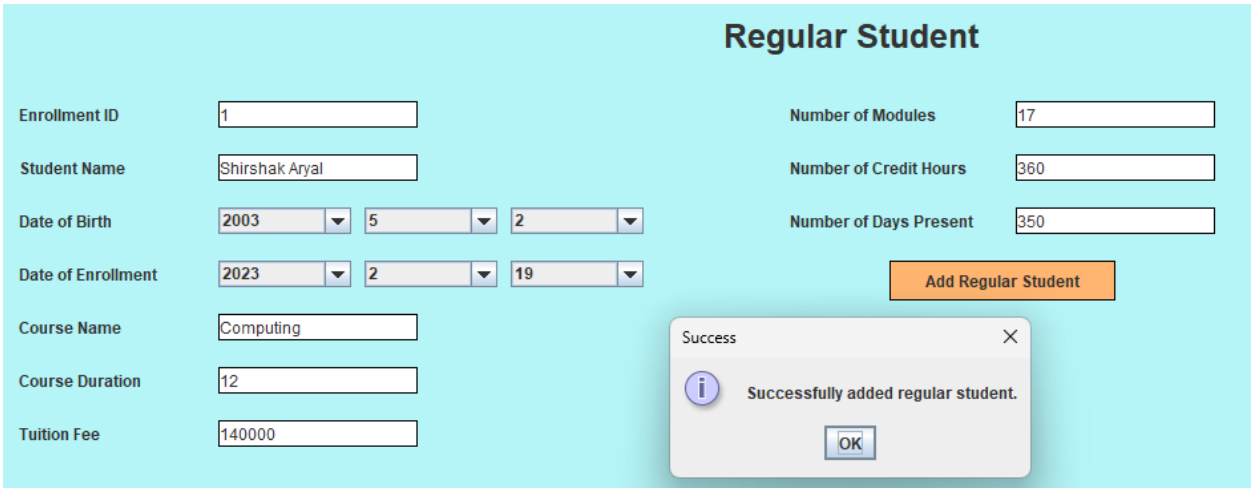


Figure 11: Dialog box with the desired message being displayed

5.2.2. Test 2.b.: To test the ‘Add a Dropout Student’ button

Objective	To test the ‘Add a Dropout Student’ button
Actions Performed	1. The tabbed pane for Dropout was clicked to navigate to the Dropout panel. 2. The required text fields were filled with valid values. 3. The ‘Add Dropout Student’ button was clicked.
Expected Result	A dialog box should have been displayed that said, ‘Successfully added dropout student.’
Actual Result	A dialog box was displayed that said, ‘Successfully added dropout student.’
Conclusion	The test was successful.

Table 3: Test 2.b.: To test the ‘Add a Dropout Student’ button

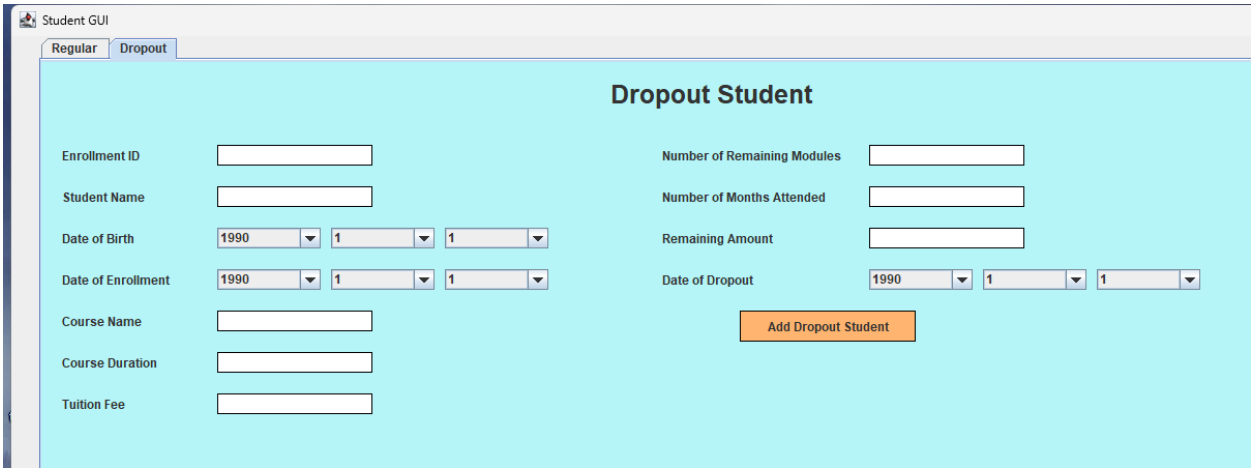


Figure 12: Navigating to the Dropout panel

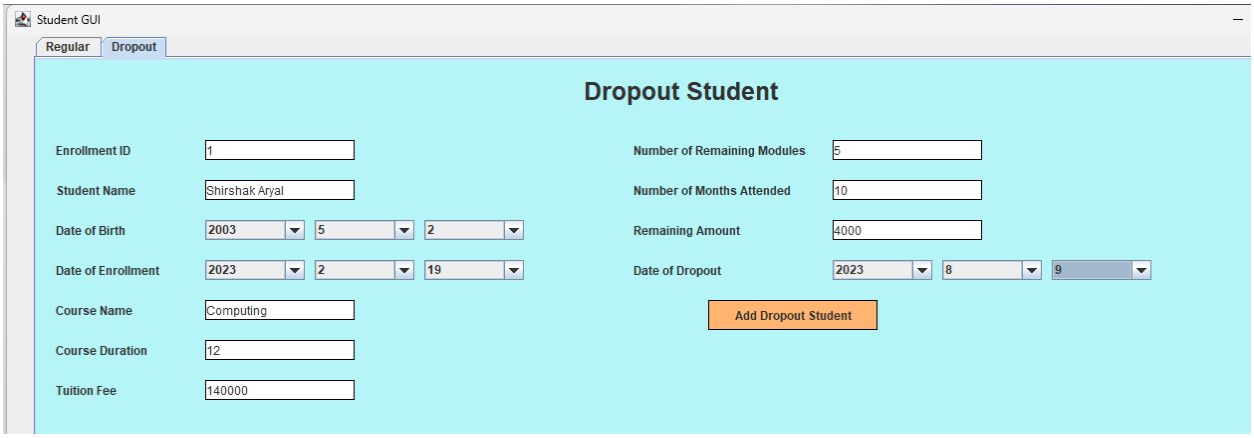


Figure 13: Filling the required text fields with valid values, and clicking the 'Add Dropout Student' button

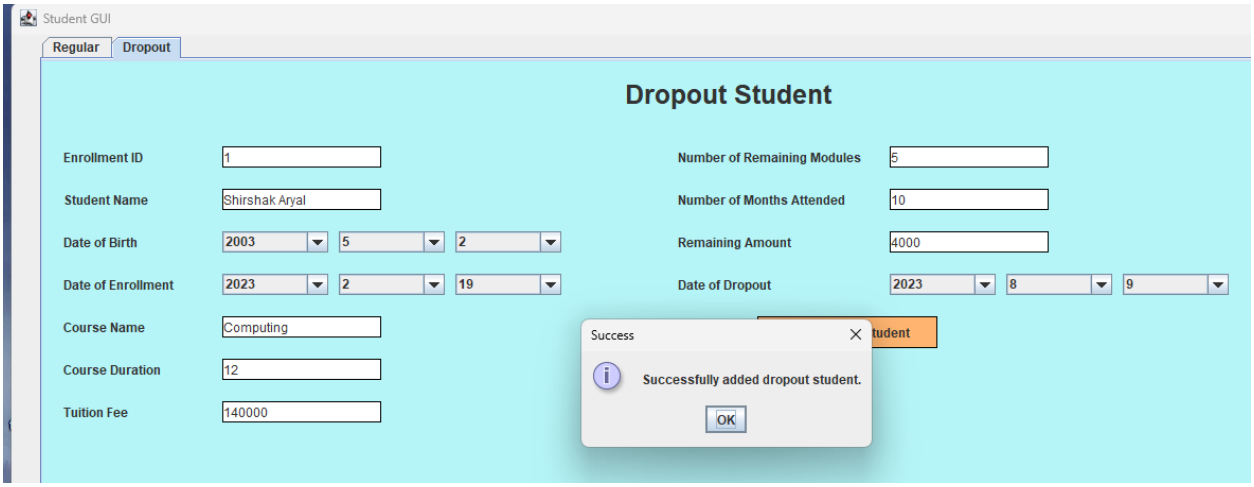


Figure 14: Dialog box with the desired message being displayed

5.2.3. Test 2.c.: To test the ‘Calculate Present Percentage of Regular Student’ button

Objective	To test the ‘Calculate Present Percentage of Regular Student’ button
Actions Performed	1. The tabbed pane for Regular was clicked to navigate to the Regular panel. 2. The ‘Enrollment ID’ and ‘Days Present’ text fields were filled with valid values. 3. The ‘Present Percentage’ button was clicked.
Expected Result	A dialog box should have been displayed that showed the attendance grade of the student.
Actual Result	A dialog box was displayed that showed the attendance grade of the student.
Conclusion	The test was successful.

Table 4: To test the ‘Calculate Present Percentage of Regular Student’ button

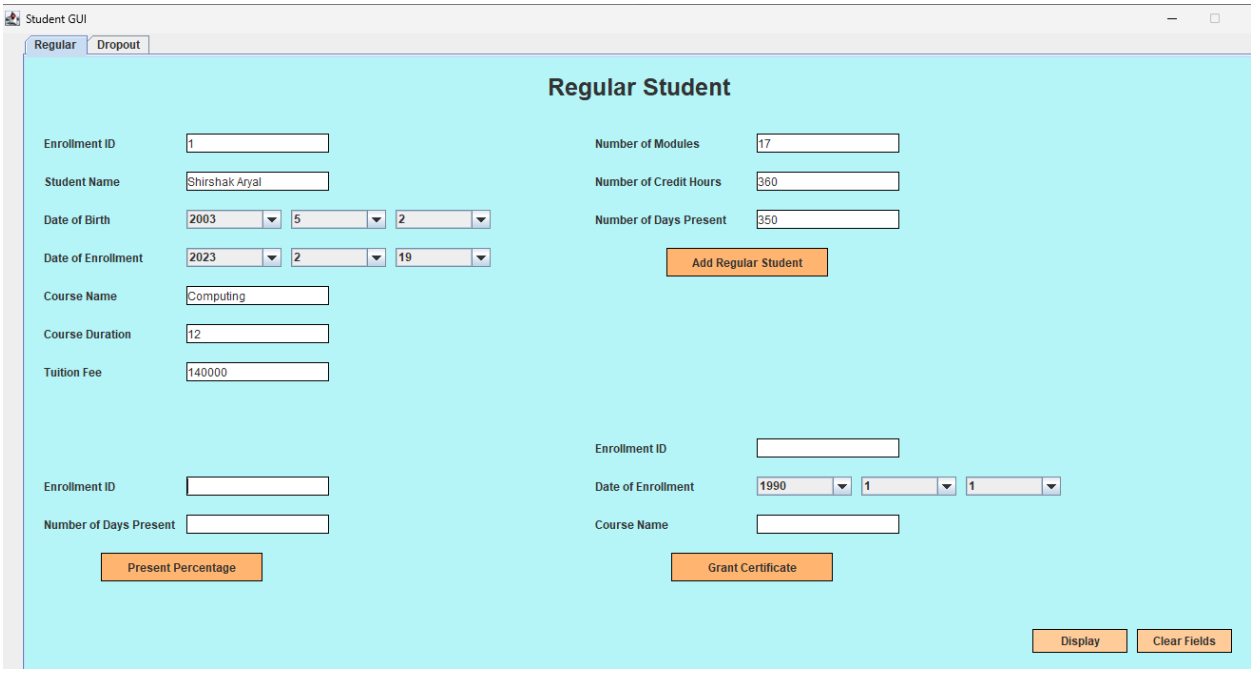


Figure 15: Navigating to the Regular panel

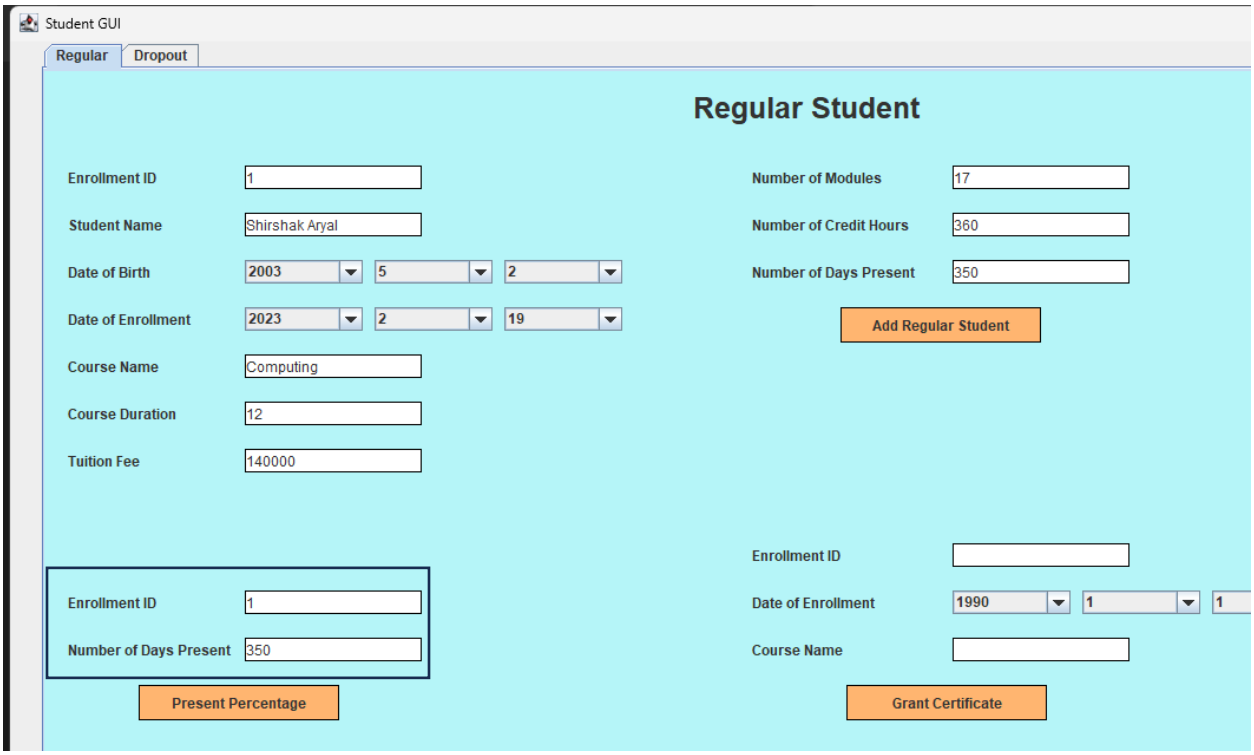


Figure 16: Filling the required text fields with valid values and clicking the 'Present Percentage' button

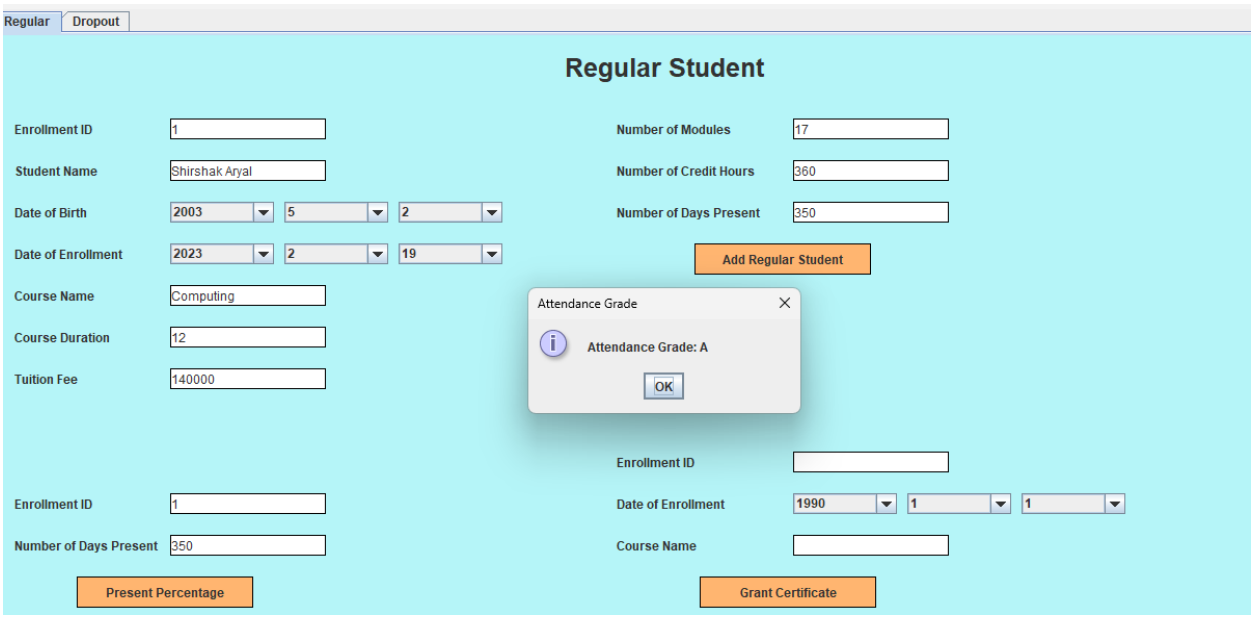


Figure 17: Dialog box with the attendance grade being displayed

5.2.4. Test 2.d.: To test the ‘Grant Certificate of Regular Student’ button

Objective	To test the ‘Grant Certificate of Regular Student’ button
Actions Performed	1. The required text fields, i.e., the ‘Enrollment ID’ and the ‘Course Name’ text fields were filled with the valid values. 2. The date of enrollment was selected from the combo boxes. 3. The ‘Grant Certificate’ button was clicked.
Expected Result	A dialog box should have been displayed that displayed the details of the student’s graduation.
Actual Result	A dialog box was displayed that displayed the details of the student’s graduation.
Conclusion	The test was successful.

Table 5: Test 2.d.: To test the ‘Grant Certificate of Regular Student’ button

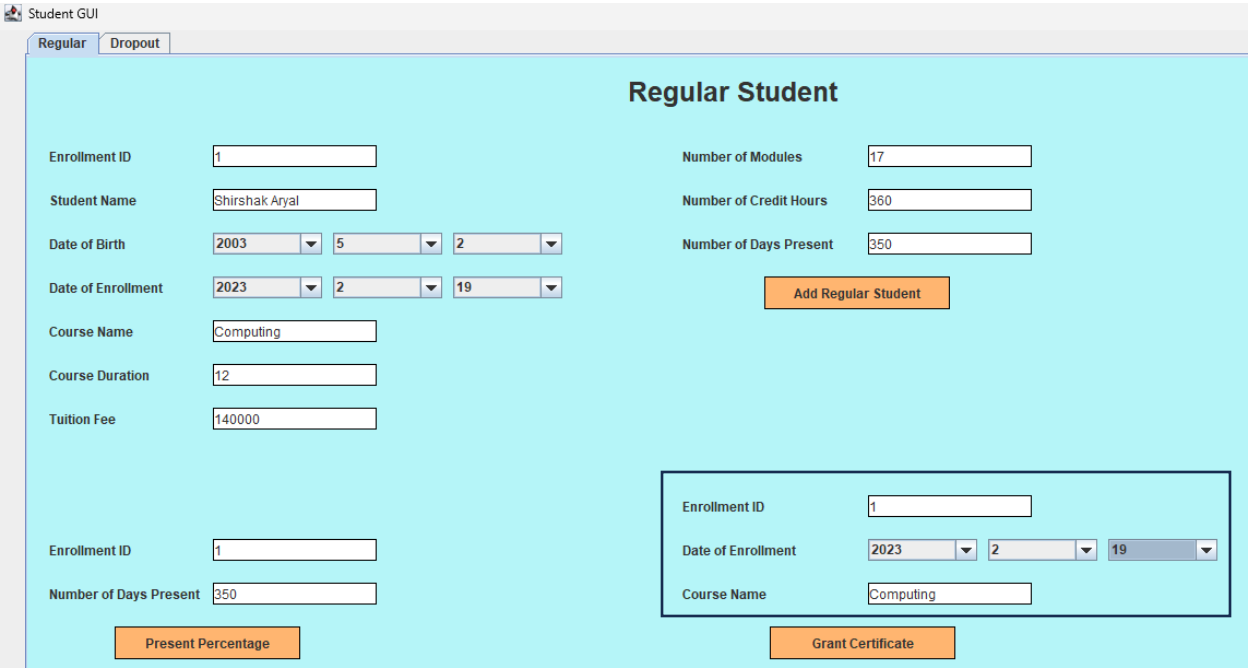


Figure 18: Filling the required text fields with valid values, selecting the date of enrollment from the combo boxes, and clicking the 'Grant Certificate' button

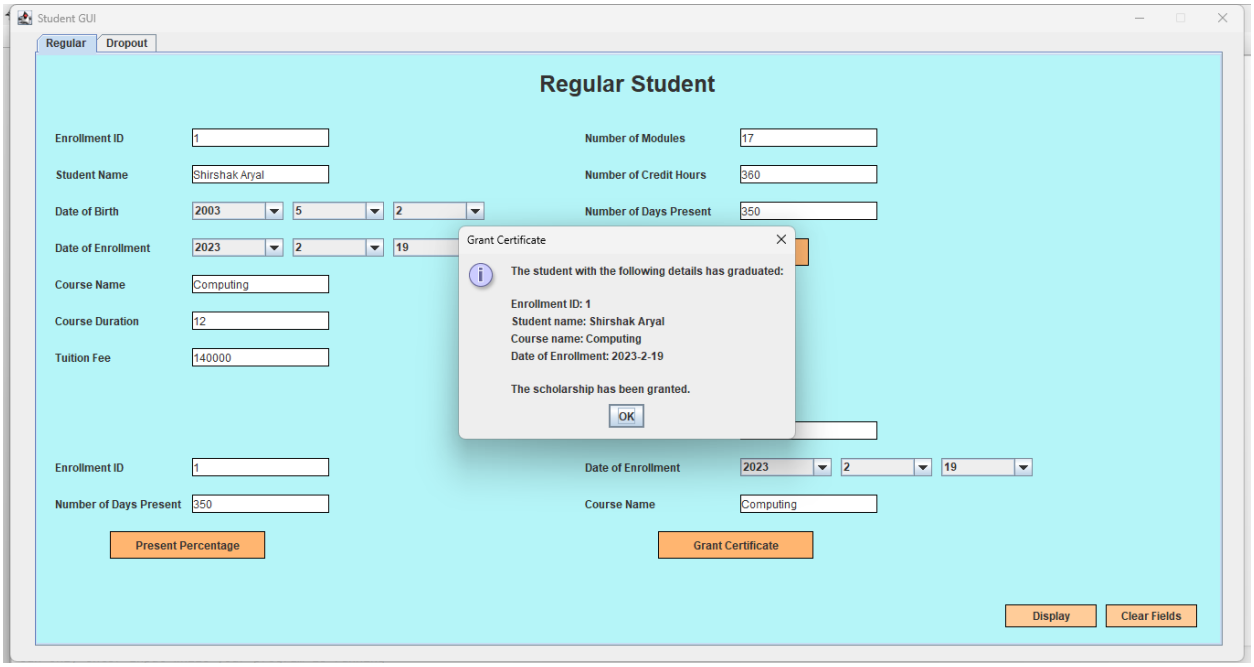


Figure 19: Dialog box with the details of the student's graduation being displayed

5.2.5. Test 2.e.: To test the ‘Pay Bills of Dropout Student’ button

Objective	To test the ‘Pay Bills of Dropout Student’ button
Actions Performed	1. The tabbed pane for Dropout was clicked to navigate to the Dropout panel. 2. The required text field, i.e., the ‘Enrollment ID’ text field, was filled with a valid value. 3. The ‘Pay Bills’ button was clicked.
Expected Result	A dialog box should have been displayed that said, ‘’.
Actual Result	A dialog box was displayed that said, ‘’.
Conclusion	The test was successful.

Table 6: Test 2.e.: To test the ‘Pay Bills of Dropout Student’ button

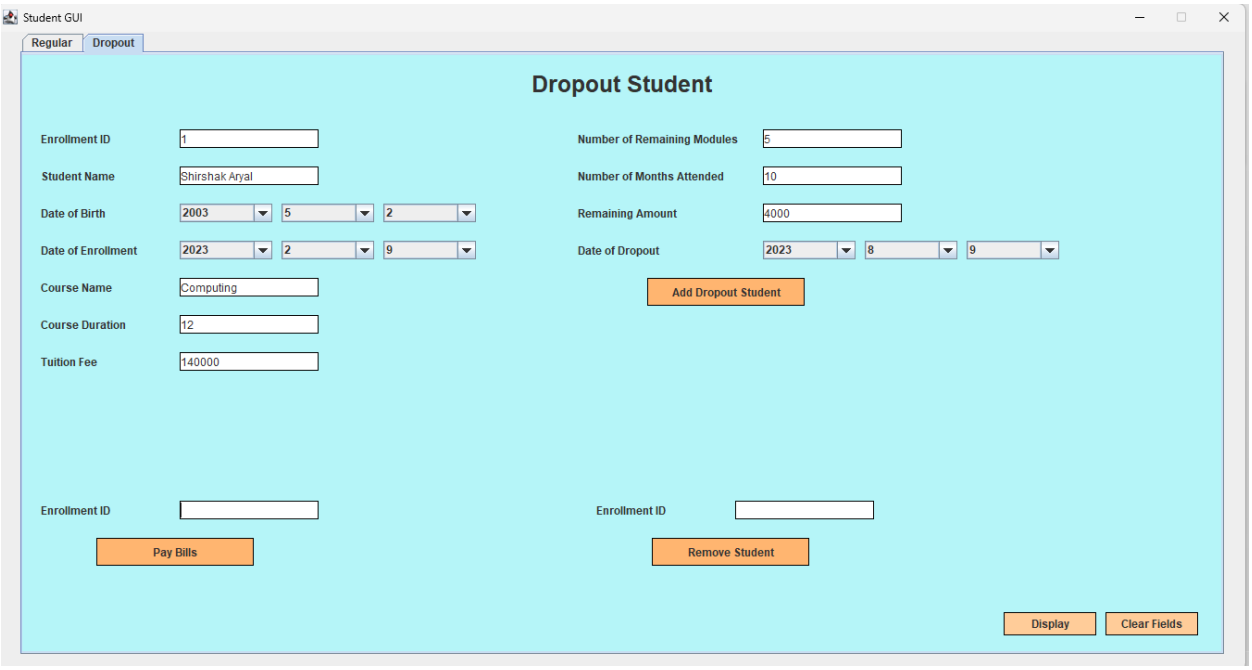


Figure 20: Navigating to the Dropout panel

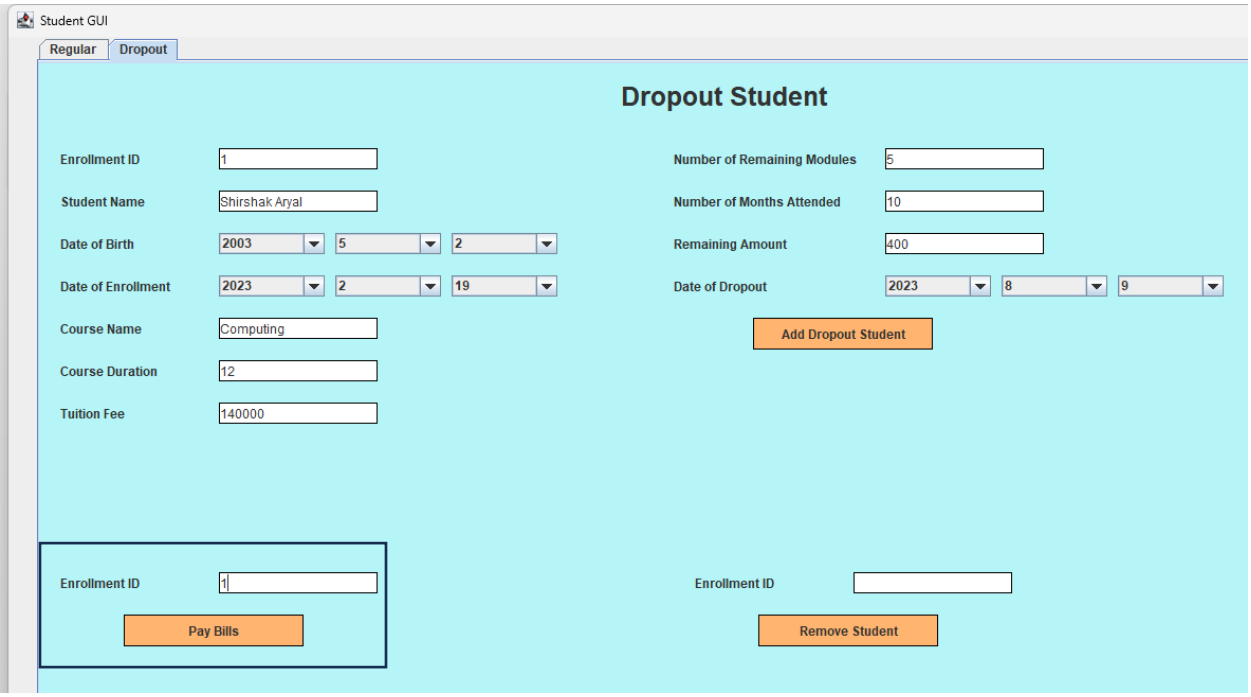


Figure 21: Filling the required text field with a valid value

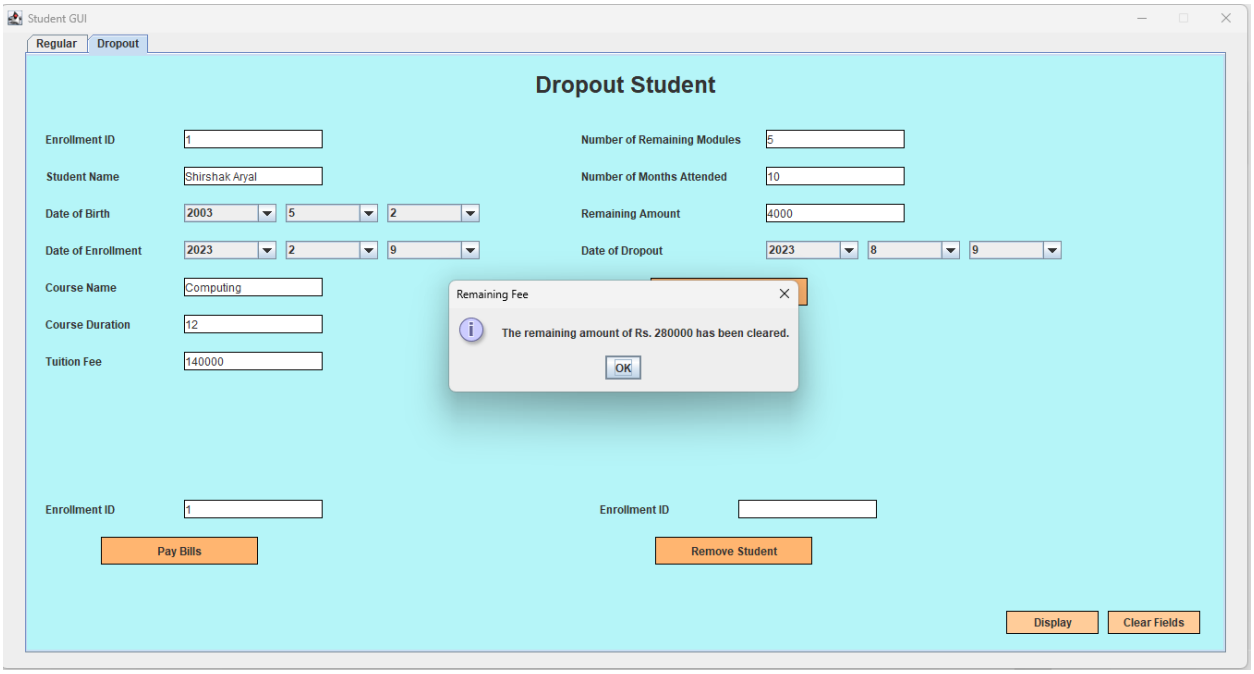


Figure 22: Dialog box with the remaining amount of fees being displayed

5.2.6. Test 2.f.: To test the ‘Remove Dropout Student’ button

Objective	To test the ‘Remove Dropout Student’ button
Actions Performed	1. The required text field, i.e., the ‘Enrollment ID’ text field was filled with a valid value. 2. The ‘Remove Student’ button was clicked.
Expected Result	A dialog box should have been displayed that said, ‘Successfully removed dropout student’.
Actual Result	A dialog box was displayed that said, ‘Successfully removed dropout student’.
Conclusion	The test was successful.

Table 7: Test 2.f.: To test the ‘Remove Dropout Student’ button

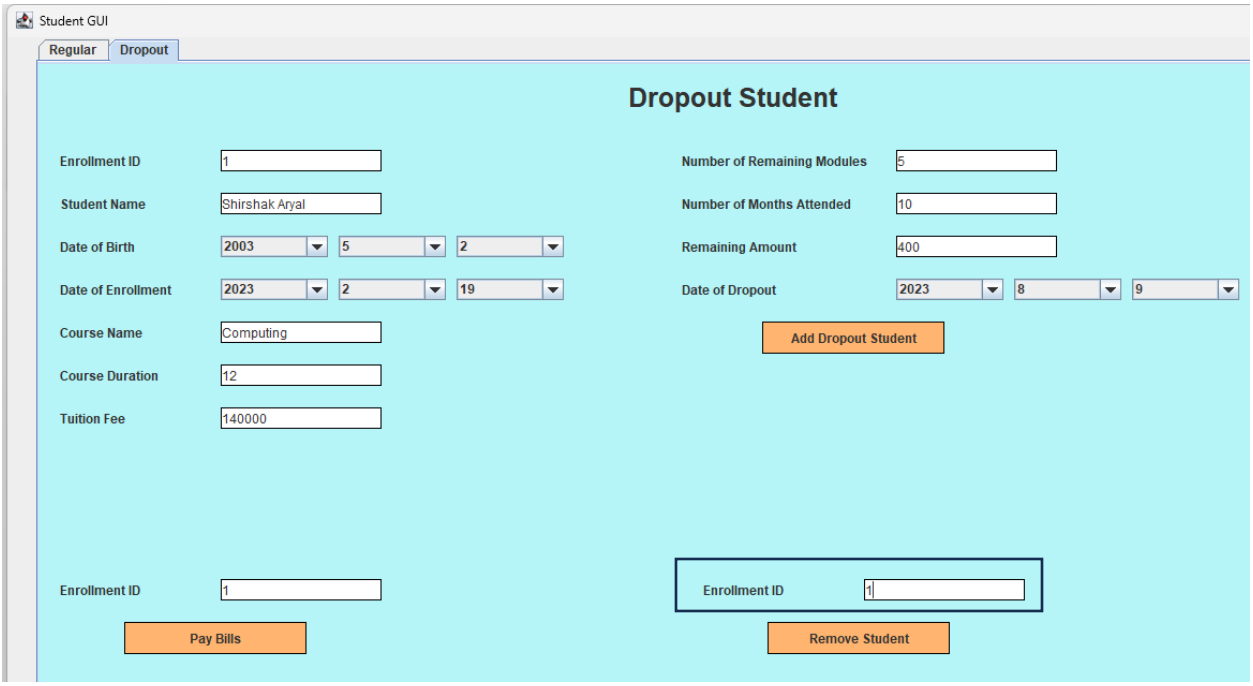


Figure 23: Filling the required text field with a valid value

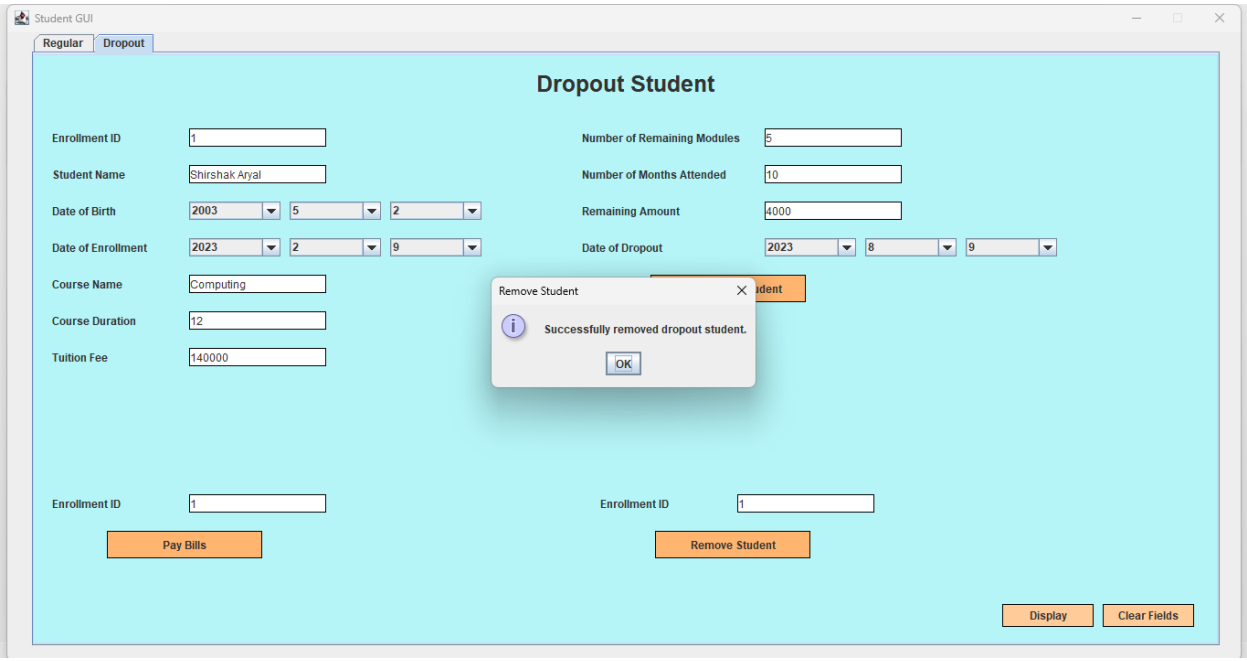


Figure 24: Dialog box with the desired message being displayed

5.3. Test 3: To test that the appropriate dialog boxes appear when unsuitable values for enrollment ID are entered

5.3.1. Test 3.a.: To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for adding a Regular student is left blank

Objective	To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for adding a Regular student is left blank
Actions Performed	1. In the Regular panel, all the text fields for adding a Regular student were filled with valid values, but the 'Enrollment ID' text field was left blank. 2. The 'Add Regular Student' button was clicked.
Expected Result	A dialog box should have been shown that informed the user that one or more text fields have been left empty.
Actual Result	A dialog box was shown that informed the user that one or more text fields have been left empty.
Conclusion	The test was successful.

Table 8: Test 3.a.: To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for adding a Regular student is left blank

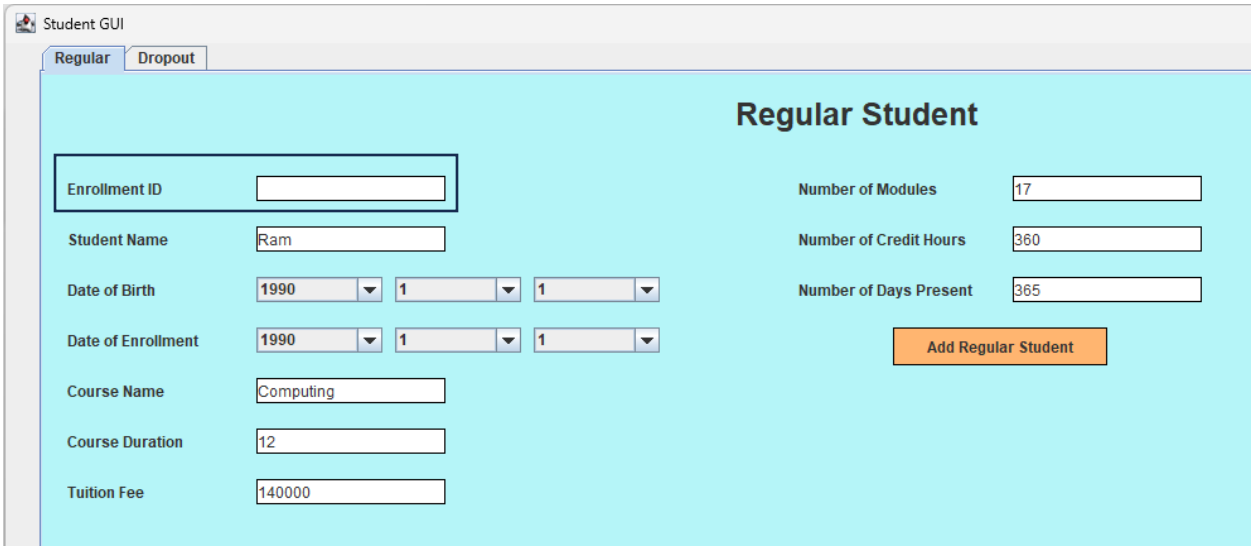


Figure 25: Leaving the 'Enrollment ID' text field blank while filling all other text fields with valid values, then clicking the 'Add Regular Student' button

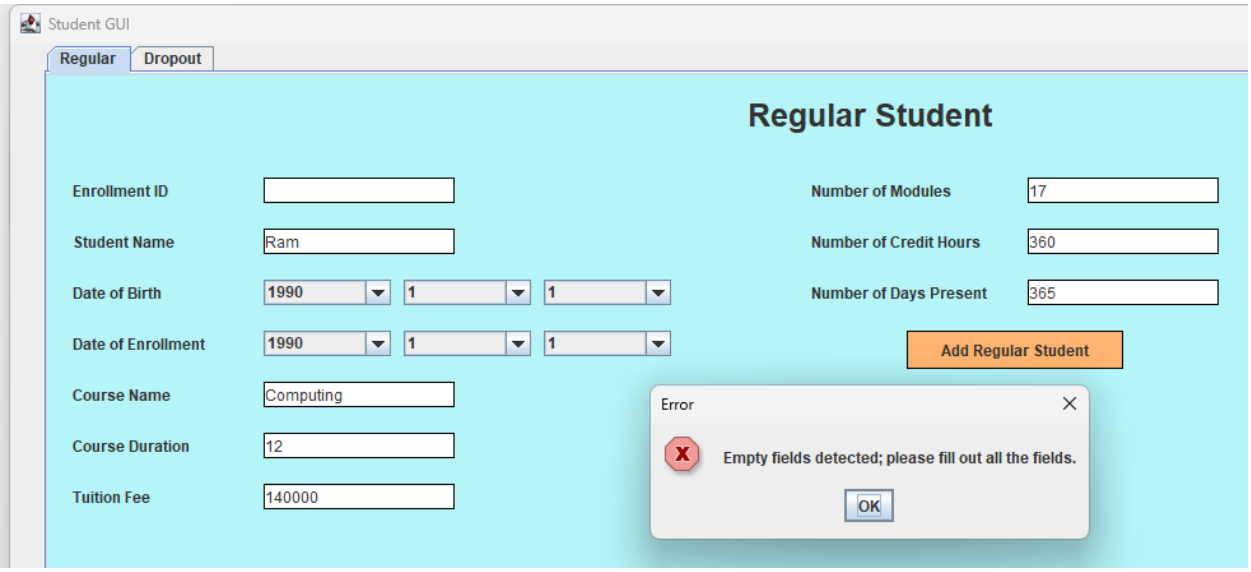


Figure 26: Error message being displayed that informs the user that one or more text fields have been left empty

5.3.2. Test 3.b.: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Regular student is filled with a string value

Objective	To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Regular student is filled with a string value
Actions Performed	1. In the Regular panel, all the text fields for adding a Regular student were filled with valid values, but the ‘Enrollment ID’ text field was filled with a string value. 2. The ‘Add Regular Student’ button was clicked.
Expected Result	A dialog box should have been shown that informed the user that the data entered in one or more text fields is invalid and reminds them to enter numbers.
Actual Result	A dialog box was shown that informed the user that the data entered in one or more text fields is invalid and reminds them to enter numbers.
Conclusion	The test was successful.

Table 9: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Regular student is filled with a string value

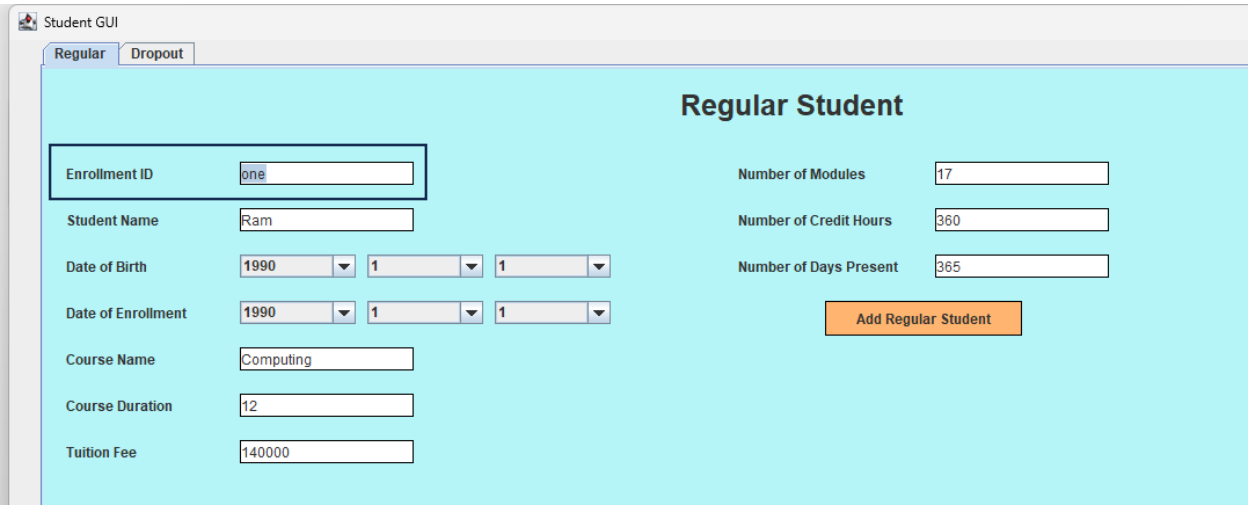


Figure 27: Filling the 'Enrollment ID' text field with a string value while filling all other text fields with valid values, then clicking the 'Add Regular Student' button

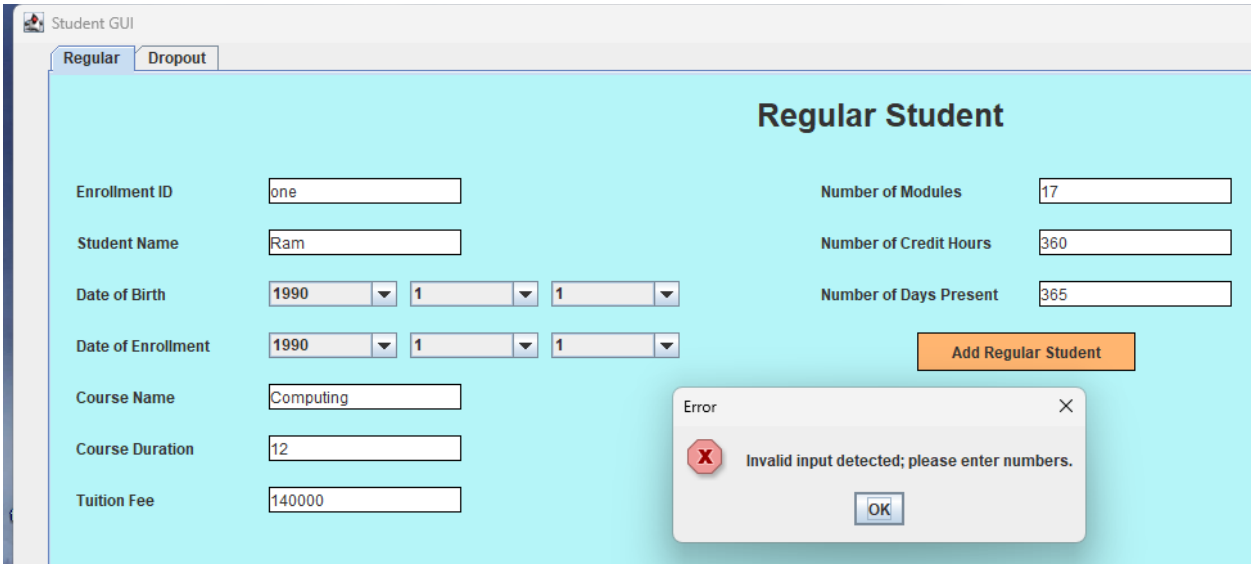


Figure 28: Error message being displayed that informs the user that one or more text fields have invalid input and reminds them to enter numbers

5.3.3. Test 3.c.: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Regular student is filled with a duplicate value

Objective	To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Regular student is filled with a duplicate value
Actions Performed	1. In the Regular panel, a Regular student was added to the Student array list. 2. The same enrollment ID was then used in an attempt to add another Regular student to the array list. 3. The ‘Add Regular Student’ button was clicked.
Expected Result	A dialog box should have been shown that informed the user that the enrollment ID entered is a duplicate one.
Actual Result	A dialog box was shown that informed the user that the enrollment ID entered is a duplicate one.
Conclusion	The test was successful.

Table 10: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Regular student is filled with a duplicate value

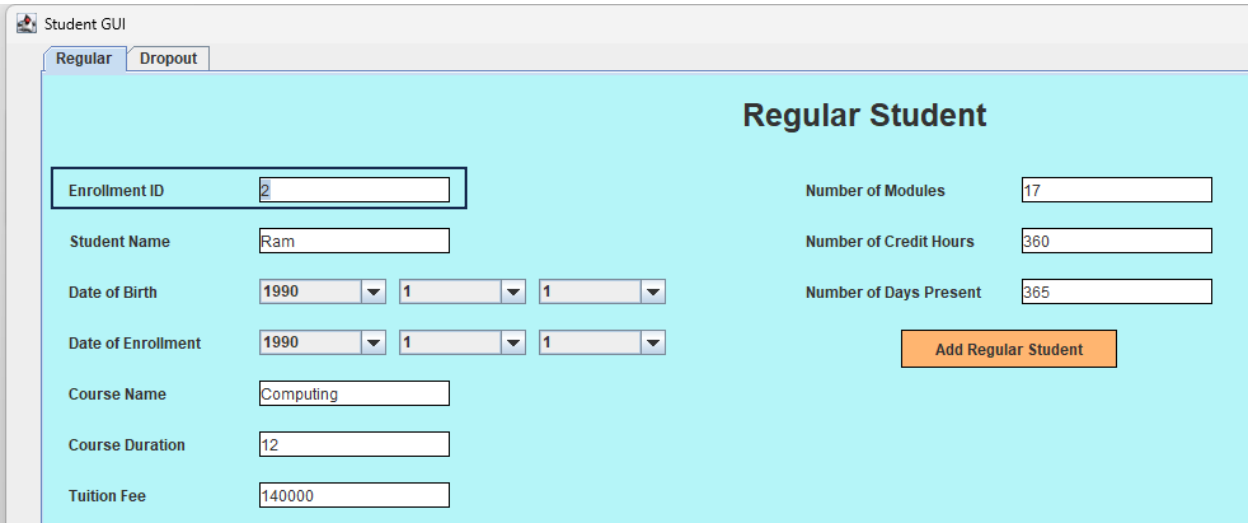


Figure 29: Setting the enrollment ID of a Regular student as 2, then clicking the 'Add Regular Student' button

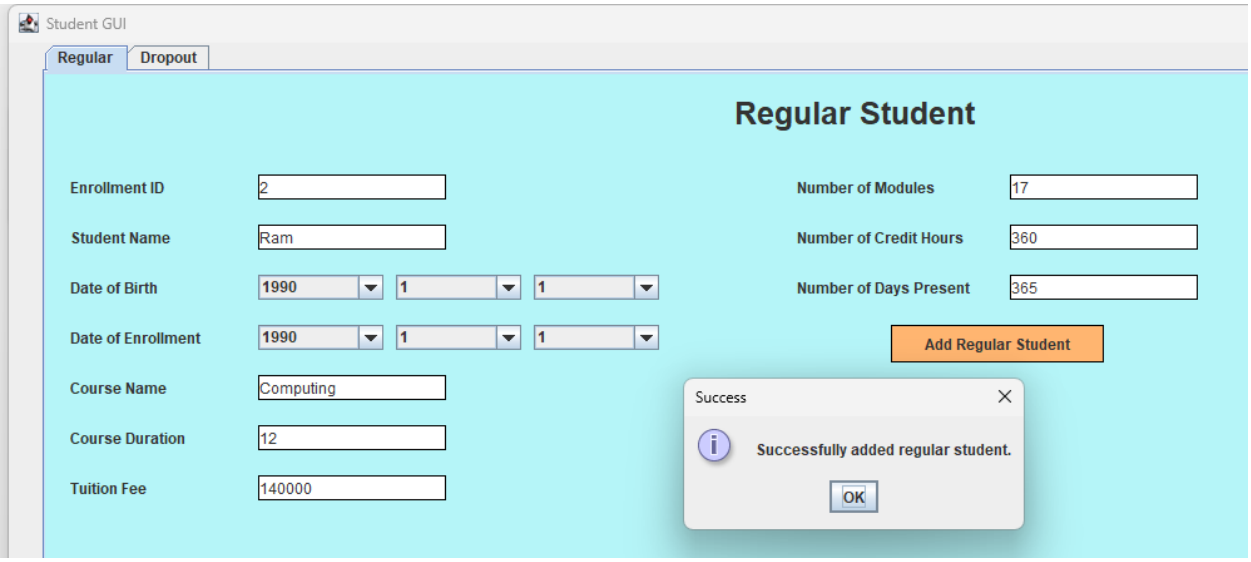


Figure 30: Successfully adding the Regular student to the Student array list

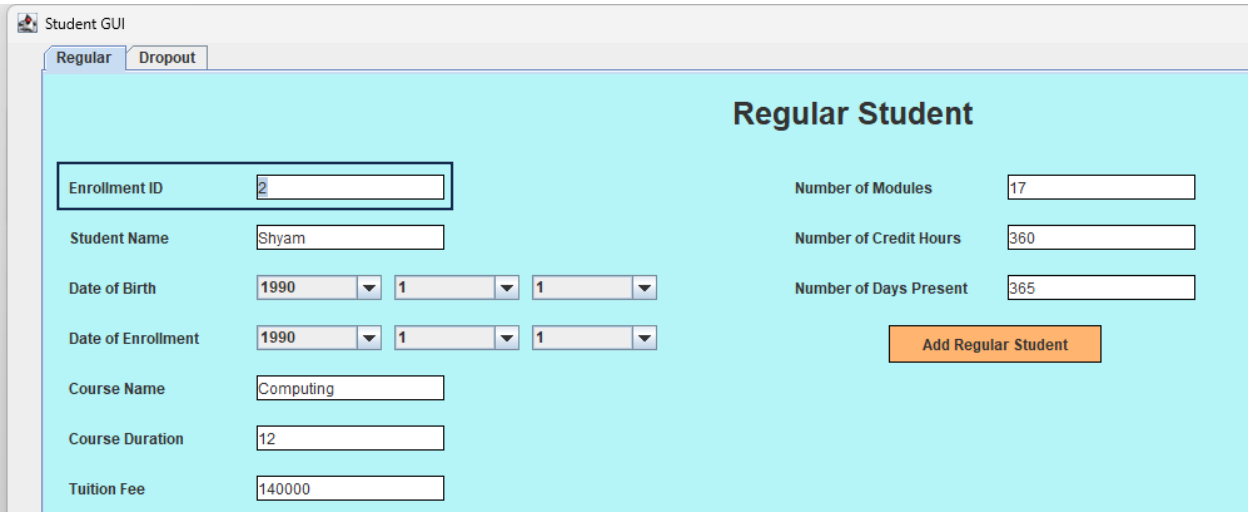


Figure 31: Using the same enrollment ID and attempting to add another Regular student

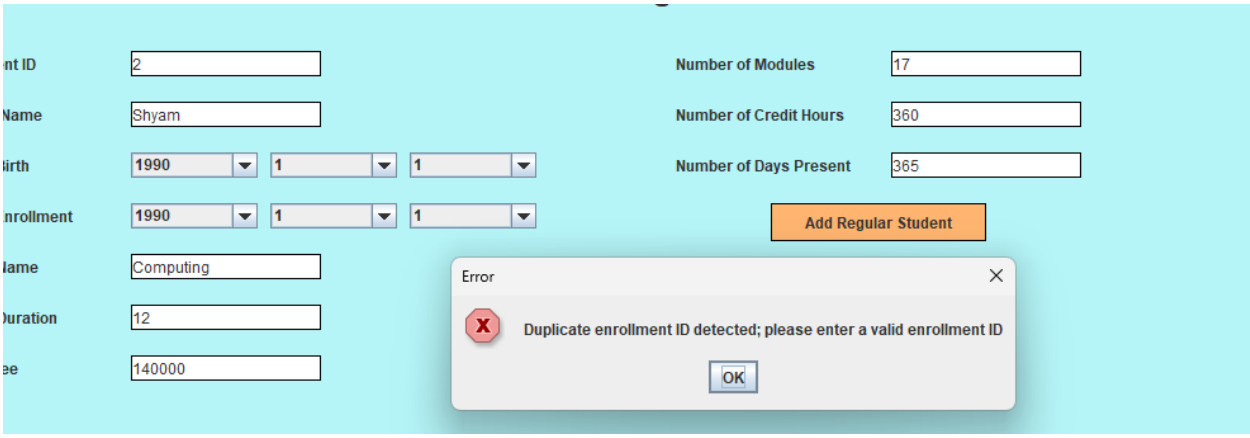


Figure 32: Error message being shown that informs the user that the enrollment ID entered is a duplicate one

5.3.4. Test 3.d.: To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for calculating the present percentage is filled with a non-existing value

Objective	To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for calculating the present percentage is filled with a non-existing value
Actions Performed	1. In the Regular panel, a Regular student was added to the Student array list. 2. A different enrollment ID was then used in an attempt to calculate the present percentage. 3. The 'Present Percentage' button was clicked.
Expected Result	A dialog box should have been shown that informed the user that the enrollment ID entered is non-existent.
Actual Result	A dialog box was shown that informed the user that the enrollment ID entered is non-existent.
Conclusion	The test was successful.

Table 11: To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for calculating the present percentage is filled with a non-existing value

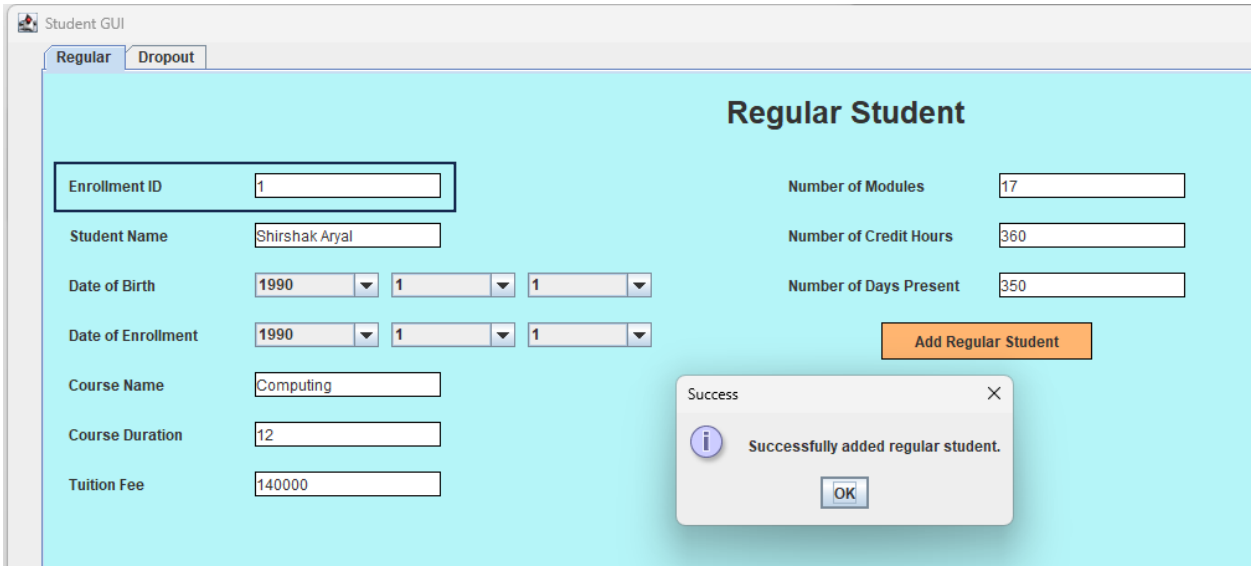


Figure 33: Successfully adding a Regular student with the enrollment ID as 1

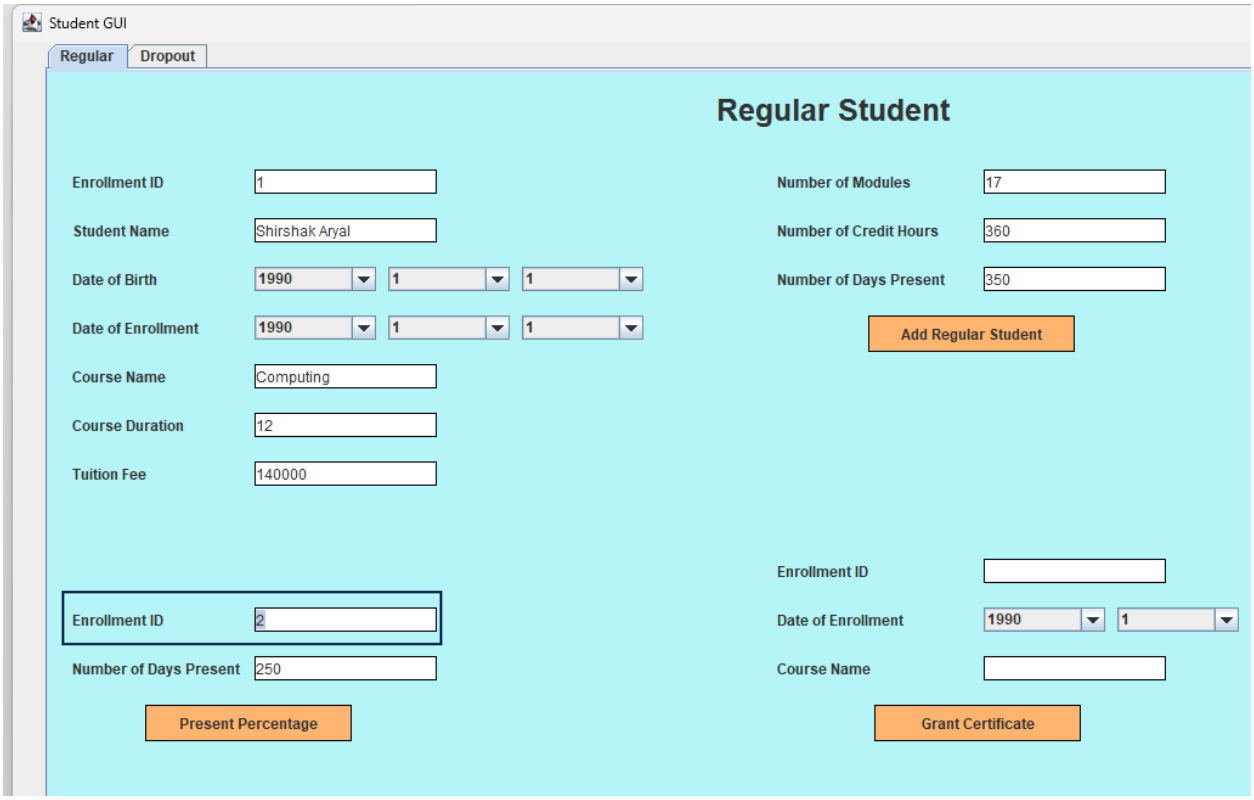


Figure 34: Attempting to calculate the present percentage using a non-existent enrollment ID

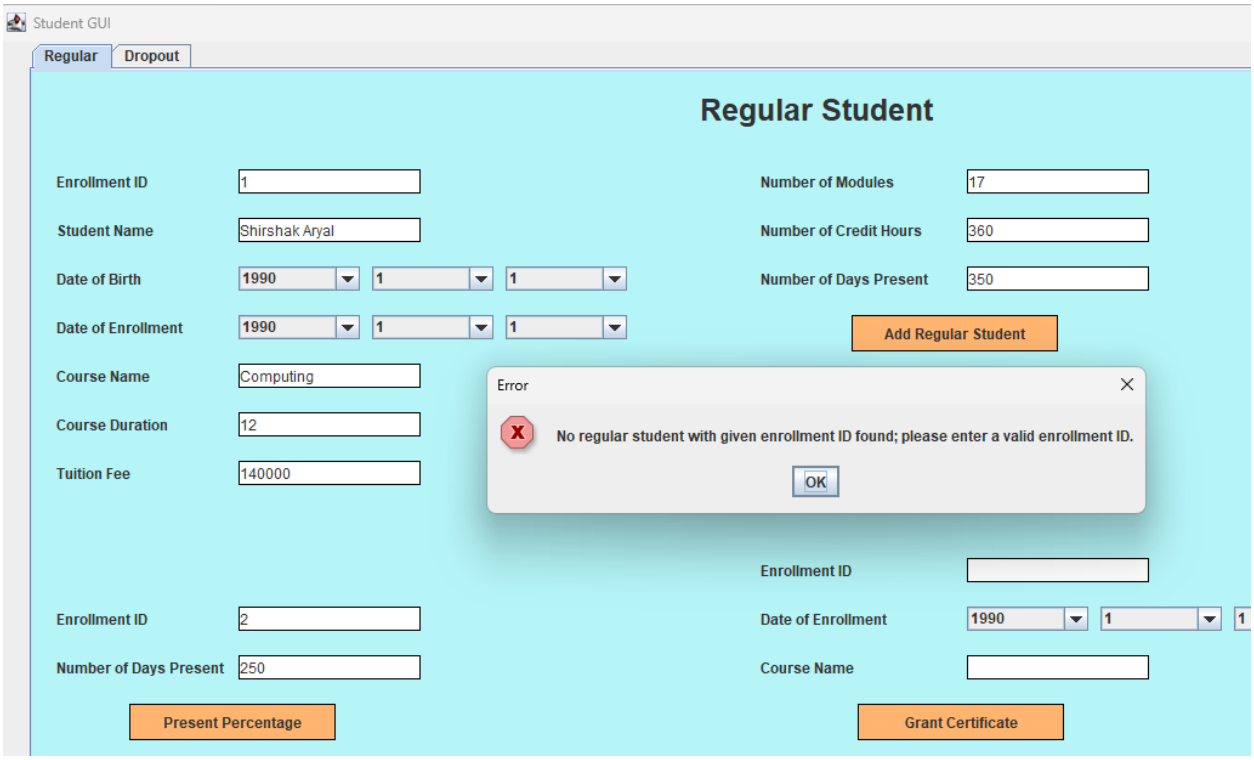


Figure 35: Error message being shown that informs the user that the enrollment ID entered is non-existent

5.3.5. Test 3.e.: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is left blank

Objective	To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is left blank
Actions Performed	4. In the Dropout panel, all the text fields for adding a Regular student were filled with valid values, but the ‘Enrollment ID’ text field was left blank. 5. The ‘Add Dropout Student’ button was clicked.
Expected Result	A dialog box should have been shown that informed the user that one or more text fields have been left empty.
Actual Result	A dialog box was shown that informed the user that one or more text fields have been left empty.
Conclusion	The test was successful.

Table 12: Test 3.e.: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is left blank

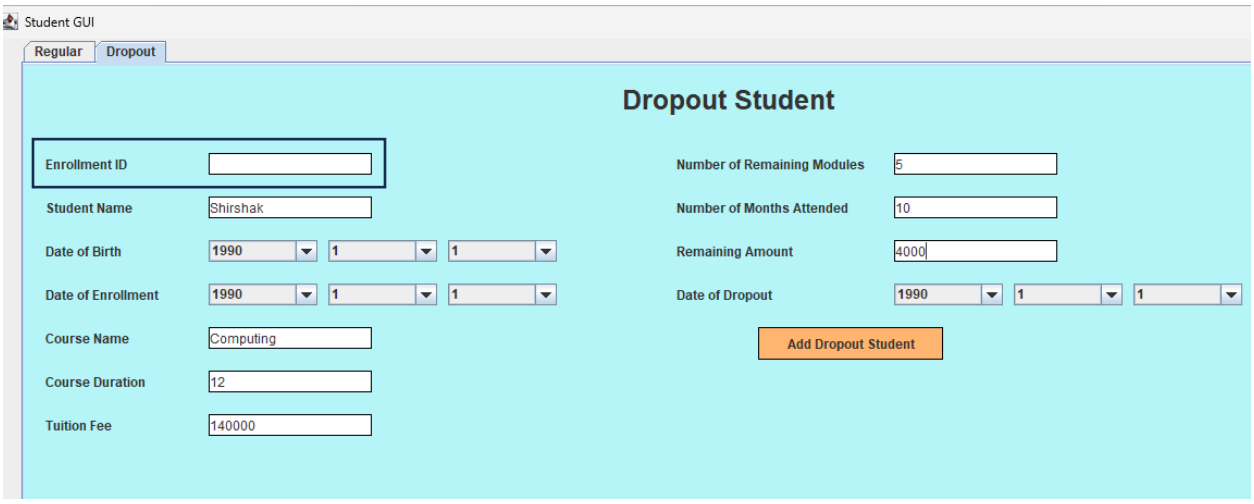


Figure 36: Leaving the 'Enrollment ID' text field blank while filling all other text fields with valid values, then clicking the 'Add Dropout Student' button

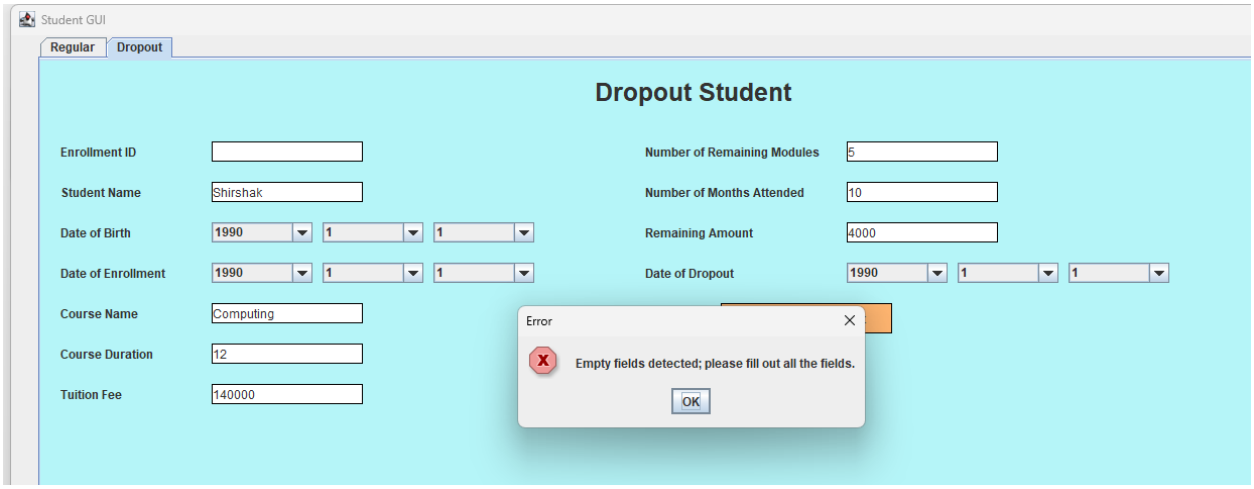


Figure 37: Error message being displayed that informs the user that one or more text fields have been left empty

5.3.6. Test 3.f.: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is filled with a string value

Objective	To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is filled with a string value
Actions Performed	3. In the Dropout panel, all the text fields for adding a Dropout student were filled with valid values, but the ‘Enrollment ID’ text field was filled with a string value. 4. The ‘Add Dropout Student’ button was clicked.
Expected Result	A dialog box should have been shown that informed the user that the data entered in one or more text fields is invalid and reminds them to enter numbers.
Actual Result	A dialog box was shown that informed the user that the data entered in one or more text fields is invalid and reminds them to enter numbers.
Conclusion	The test was successful.

Table 13: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is filled with a string value

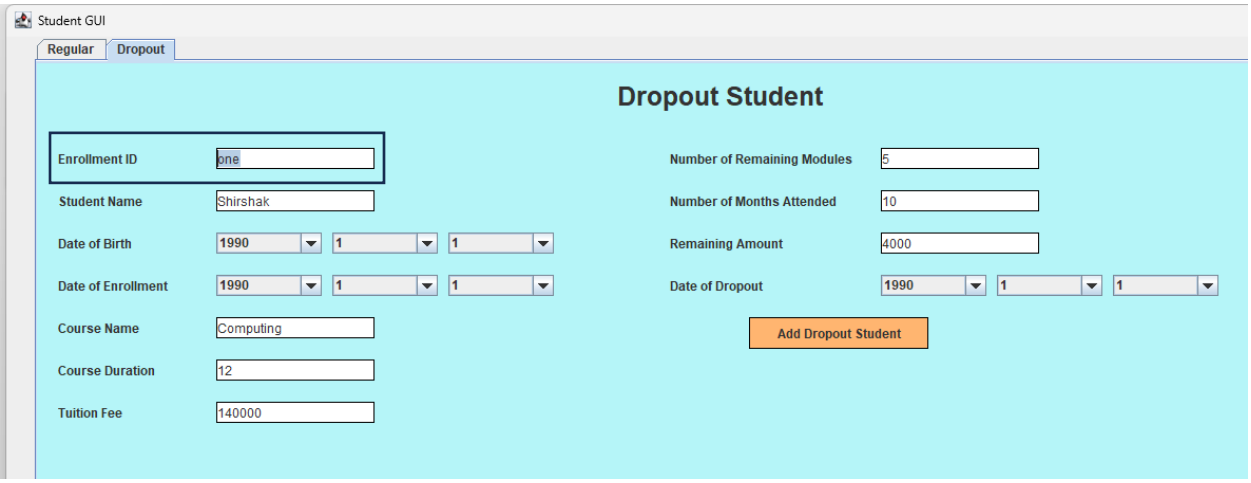


Figure 38: Filling the 'Enrollment ID' text field with a string value while filling all other text fields with valid values, then clicking the 'Add Dropout Student' button

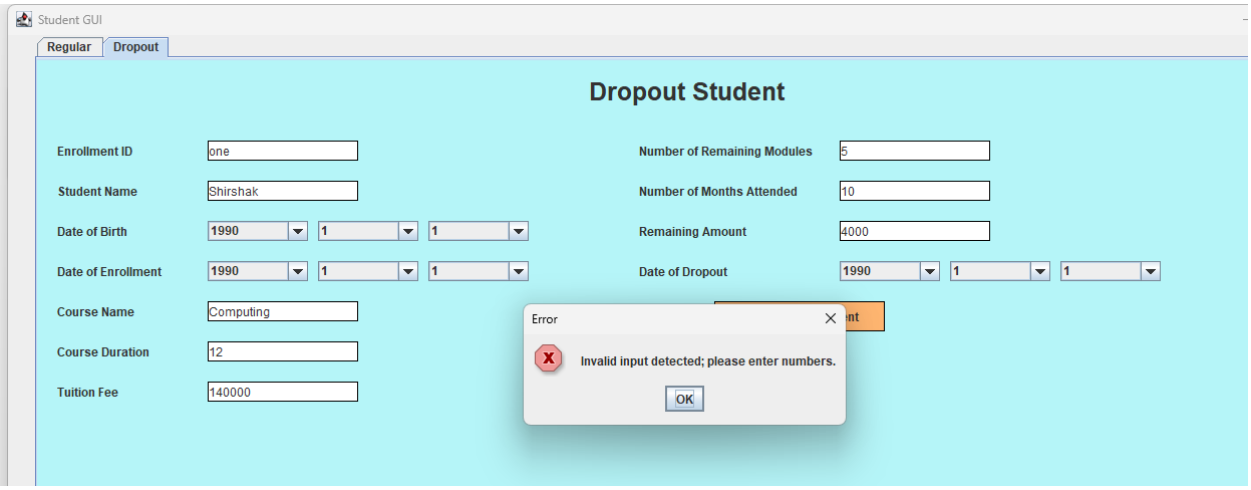


Figure 39: Error message being displayed that informs the user that one or more text fields have invalid input and reminds them to enter numbers

5.3.7. Test 3.g.: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is filled with a duplicate value

Objective	To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is filled with a duplicate value
Actions Performed	6. In the Dropout panel, a Dropout student was added to the Student array list. 7. The same enrollment ID was then used to add another Dropout student to the array list. 8. The ‘Add Dropout Student’ button was clicked.
Expected Result	A dialog box should have been shown that informed the user that the enrollment ID entered is a duplicate one.
Actual Result	A dialog box was shown that informed the user that the enrollment ID entered is a duplicate one.
Conclusion	The test was successful.

Table 14: To test if the appropriate dialog box is shown when the ‘Enrollment ID’ text field for adding a Dropout student is filled with a duplicate value

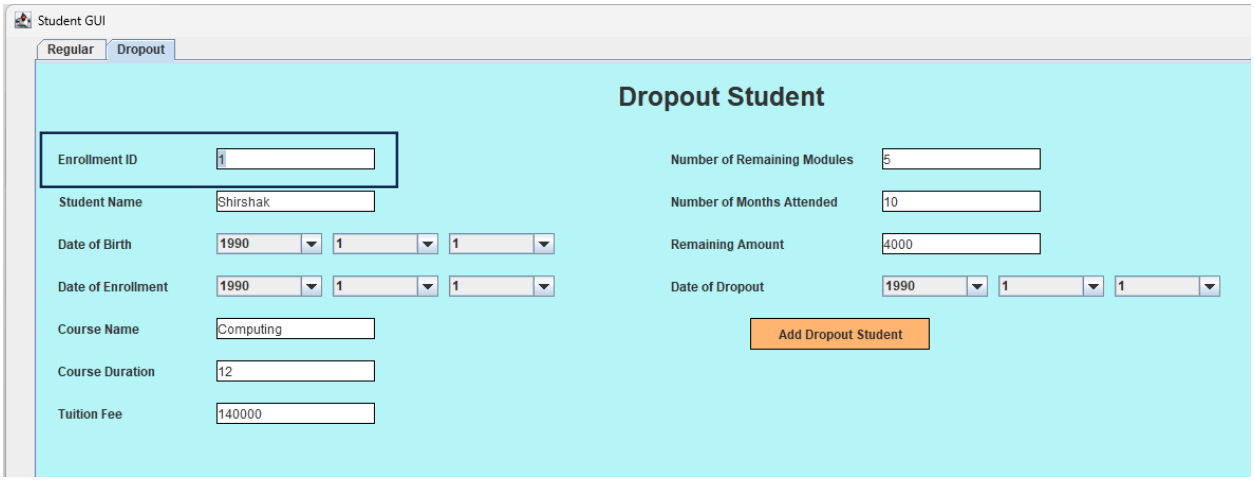


Figure 40: Setting the enrollment ID of a Dropout student as 1, then clicking the 'Add Dropout Student' button

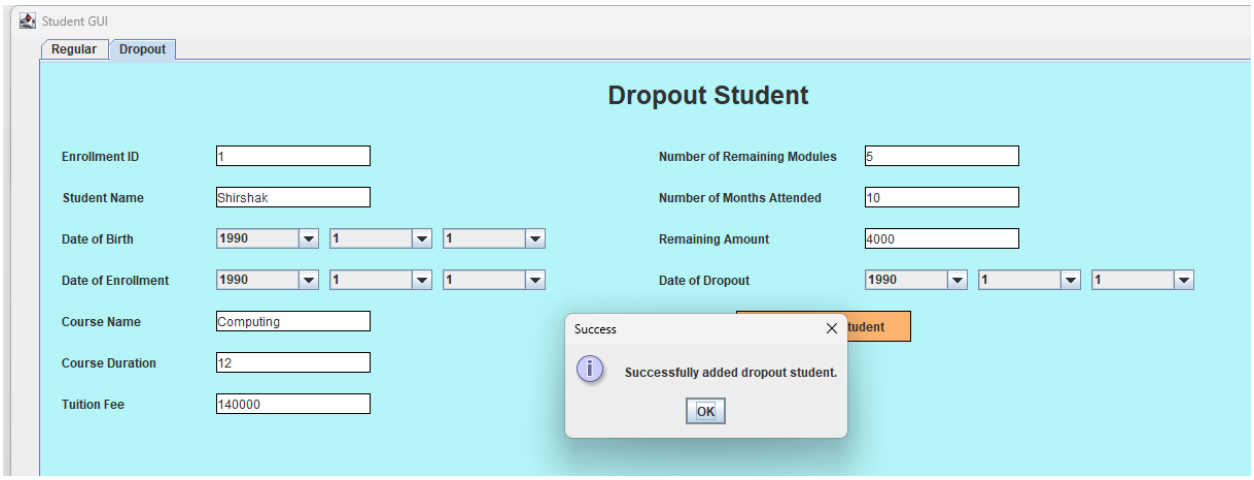


Figure 41: Successfully adding the Dropout student to the Student array list

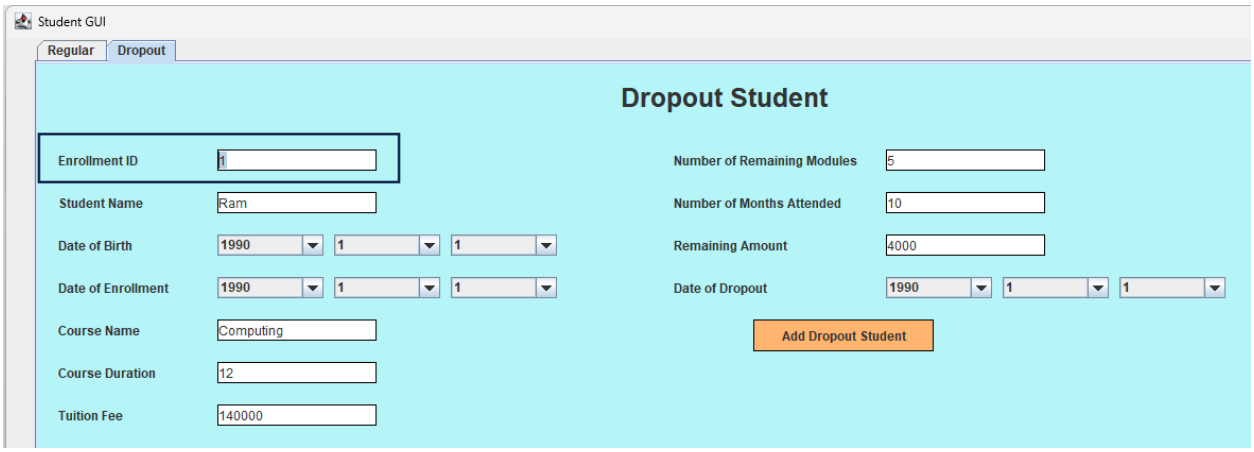


Figure 42: Using the same enrollment ID and attempting to add another Dropout student

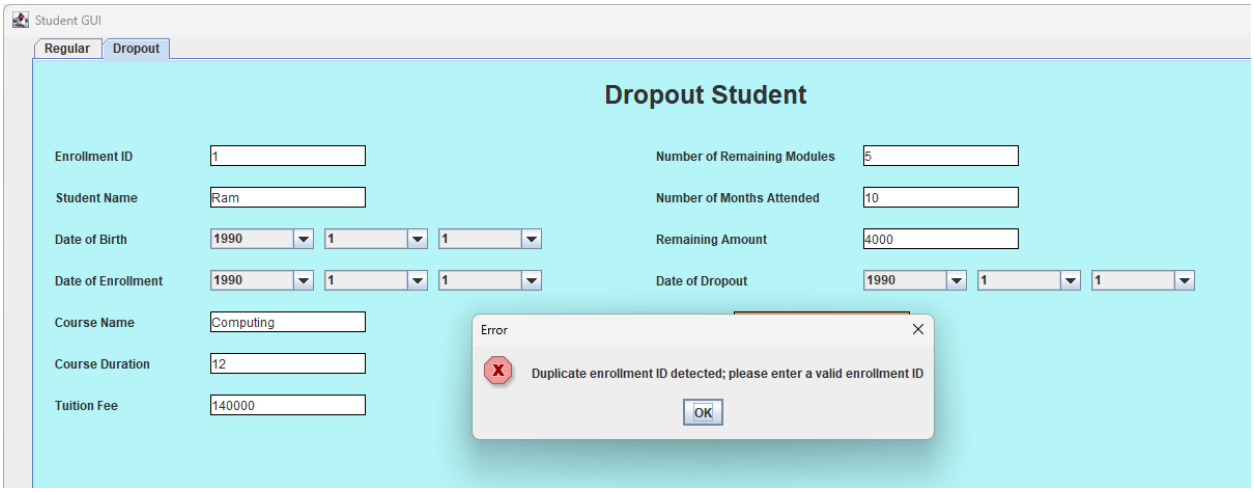


Figure 43: Error message being shown that informs the user that the enrollment ID entered is a duplicate one

5.3.8. Test 3.h.: To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for calculating the remaining amount of fees is filled with a non-existing value

Objective	To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for calculating the remaining amount of fees is filled with a non-existing value
Actions Performed	4. In the Dropout panel, a Dropout student was added to the Student array list. 5. A different enrollment ID was then used to calculate the present remaining amount of fees. 6. The 'Pay Bills' button was clicked.
Expected Result	A dialog box should have been shown that informed the user that the enrollment ID entered is non-existent.
Actual Result	A dialog box was shown that informed the user that the enrollment ID entered is non-existent.
Conclusion	The test was successful.

Table 15: To test if the appropriate dialog box is shown when the 'Enrollment ID' text field for calculating the remaining amount of fees is filled with a non-existing value

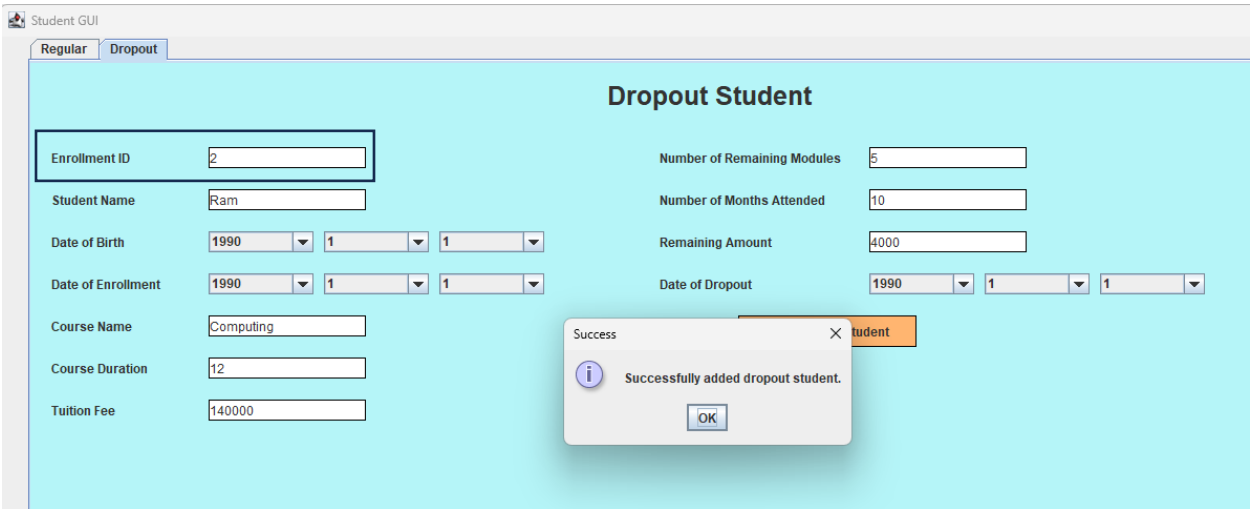


Figure 44: Successfully adding a Dropout student with the enrollment ID as 2

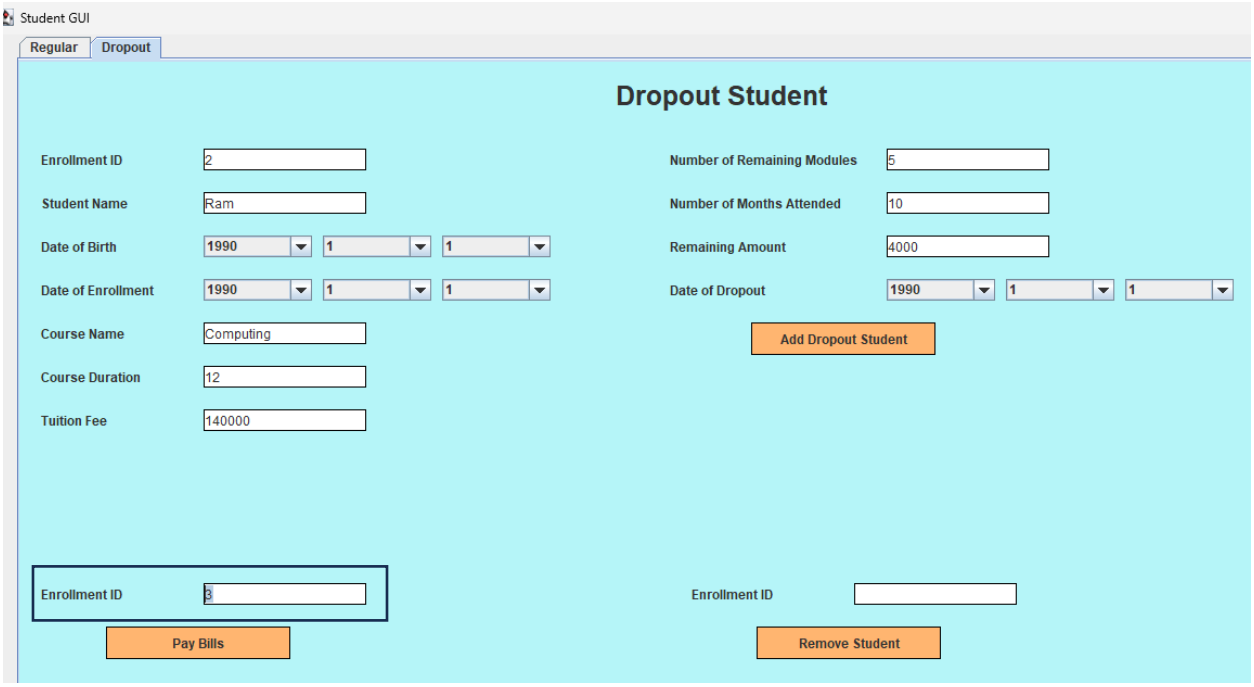


Figure 45: Attempting to calculate the remaining amount of fees using a non-existent enrollment ID

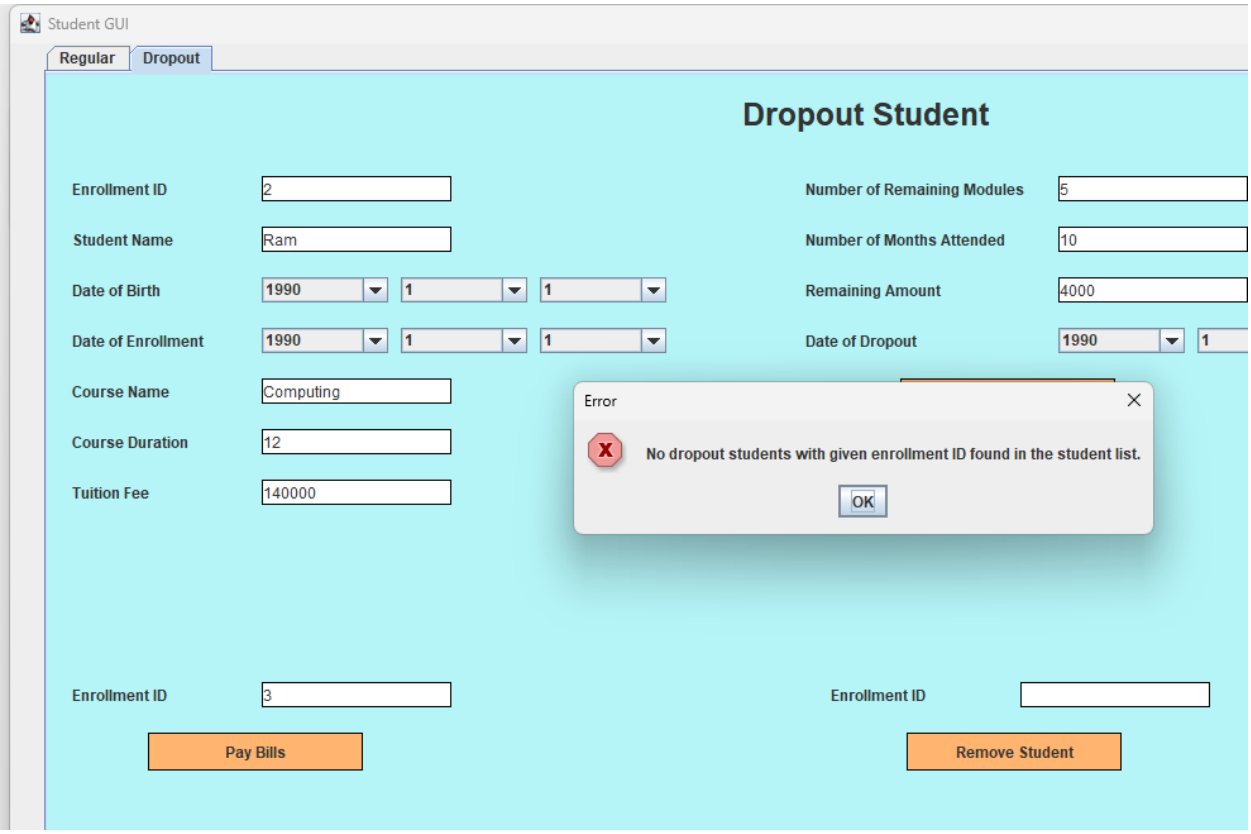


Figure 46: Error message being shown that informs the user that the enrollment ID entered is non-existent

6. Error Detection

6.1. Syntax Error

6.1.1. Error

A syntax error occurred in one of the lines of code in the ‘Grant Certificate Button’ section of the actionPerformed(ActionEvent e) method. The reason behind this was a missing semicolon.

```
boolean grantedCertificate = false;//setting a flag

for(Student stud:studList){

    //IF THE STUDENT IS OF Regular CLASS
    if(stud instanceof Regular){
        Regular regStud = (Regular) stud;//downcasting stud as an object of Regular class

        //IF THE USER-ENTERED ENROLLMENT ID MATCHES THAT OF A PRE-EXISTING Regular STUDENT
        if(regStud.getEnrollmentID() == grantCertEnrID){
            grantedCertificate = true;//setting the flag to true

            //granting certificate to regStud
            String grantedCertificateMessage = regStud.grantCertificate(grantCertCourseName,grantCertEnrID,grantCertDateOfEnrollment);

            //displaying message to notify the user
            JOptionPane.showMessageDialog(f,grantedCertificateMessage,"Grant Certificate",JOptionPane.INFORMATION_MESSAGE)
            ';' expected
            break;
        }
    }
}
```

Figure 47: Missing semicolon in the code

6.1.2. Correction

The error was corrected simply by adding the missing semicolon to that code.

```
boolean grantedCertificate = false;//setting a flag

for(Student stud:studList){

    //IF THE STUDENT IS OF Regular CLASS
    if(stud instanceof Regular){
        Regular regStud = (Regular) stud;//downcasting stud as an object of Regular class

        //IF THE USER-ENTERED ENROLLMENT ID MATCHES THAT OF A PRE-EXISTING Regular STUDENT
        if(regStud.getEnrollmentID() == grantCertEnrID){
            grantedCertificate = true;//setting the flag to true

            //granting certificate to regStud
            String grantedCertificateMessage = regStud.grantCertificate(grantCertCourseName,grantCertEnrID,grantCertDateOfEnrollment);

            //displaying message to notify the user
            JOptionPane.showMessageDialog(f,grantedCertificateMessage,"Grant Certificate",JOptionPane.INFORMATION_MESSAGE);

            break;
        }
    }
}
```

Figure 48: No syntax error seen after adding the missing semicolon

6.2. Semantic Error

6.2.1. Error

A semantic error occurred in the 'Add Dropout Student Button' section of the actionPerformed (ActionEvent e) method, caused due to the arguments passed into the Dropout() constructor not matching with the expected parameters.

```
dropoutEnrID = Integer.parseInt(dropoutEnrIDtf.getText());//enrollment ID

dropoutCourseDuration = Integer.parseInt(dropoutCourseDurationtf.getText());//course duration
dropoutTuitionFee = Integer.parseInt(dropoutTuitionFeetf.getText());//tuition fee

numOfRemainingModules = Integer.parseInt(numOfRemainingModulestf.getText());//number of remaining modules
numOfMonthsAttended = Integer.parseInt(numOfMonthsAttendedtf.getText());//number of months attended
remainingAmount = Integer.parseInt(remainingAmounttf.getText());//remaining amount


//CREATING AND UPCASTING A NEW Dropout OBJECT
Student dropoutStud = new Dropout(dropoutEnrID,dropoutDateOfBirth,dropoutStudName,dropoutDateOfEnrollment,
                      dropoutCourseDuration,dropoutTuitionFee,numOfR
                      dateOfDropout);

constructor Dropout in class Dropout cannot be applied to given types;
  required: int,java.lang.String,java.lang.String,java.lang.String,java.lang.String,int,int,int,int,int,java.lang.String
  found:    int,java.lang.String,java.lang.String,java.lang.String,int,int,int,int,int,java.lang.String
  reason: actual and formal argument lists differ in length

//CHECKING IF STUDENT ARRAYLIST IS EMPTY
if (studList.isEmpty()){
    studList.add(dropoutStud);//adding student to student arraylist

    //displaying message to notify the user
    JOptionPane.showMessageDialog(f,"Successfully added dropout student.","Success",JOptionPane.INFORMATION_MESSAGE);
}
```

Figure 49: Semantic error caused due to mismatching of arguments with parameters in the Dropout() constructor

6.2.2. Correction

This error was corrected by adding the missing argument in the Dropout() constructor, which was the variable for the course name of Dropout student.

```
try{
    dropoutEnrID = Integer.parseInt(dropoutEnrIDtf.getText());//enrollment ID

    dropoutCourseDuration = Integer.parseInt(dropoutCourseDurationtf.getText());//course duration
    dropoutTuitionFee = Integer.parseInt(dropoutTuitionFeetf.getText());//tuition fee

    numOfRemainingModules = Integer.parseInt(numOfRemainingModulestf.getText());//number of remaining modules
    numOfMonthsAttended = Integer.parseInt(numOfMonthsAttendedtf.getText());//number of months attended
    remainingAmount = Integer.parseInt(remainingAmounttf.getText());//remaining amount


    //CREATING AND UPCASTING A NEW Dropout OBJECT
    Student dropoutStud = new Dropout(dropoutEnrID,dropoutDateOfBirth,dropoutCourseName,dropoutStudName,dropoutDateOfEnrollment,
                          dropoutCourseDuration,dropoutTuitionFee,numOfRemainingModules,numOfMonthsAttended,remainingAmount,
                          dateOfDropout);


    //CHECKING IF STUDENT ARRAYLIST IS EMPTY
```

Figure 50: No semantic error seen after adding the missing argument in the Dropout() constructor

6.3. Logical Error

6.3.1. Error

A logical error occurred in the program that displayed an error message that informed the user to enter non-negative values, even when the value of the enrollment ID entered for adding a Regular student was non-negative. This was caused due to the wrong comparison operator being used in the code for the ‘Add Regular Student’ button, i.e. ‘>=’ was used instead of ‘<=’.

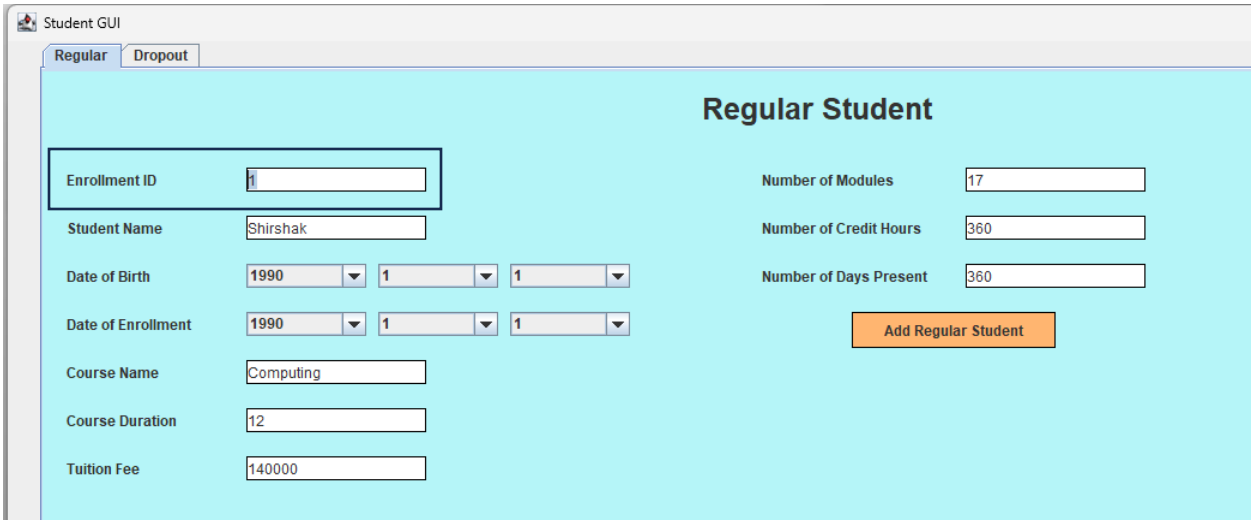


Figure 51: Attempting to add a Regular student with all valid values, including non-negative value for enrollment ID

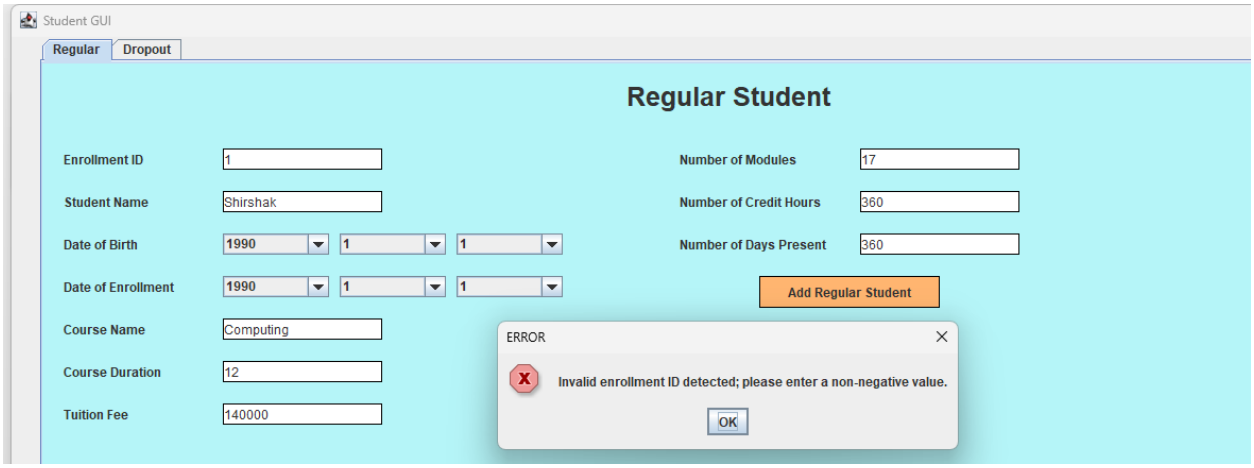


Figure 52: Error message being displayed that informs the user to enter non-negative value for enrollment ID

```
        regTuitionFee = Integer.parseInt(regTuitionFeetf.getText());//tuition fee

        numOfModules = Integer.parseInt(numOfModulestf.getText());//number of modules
        numOfCreditHours = Integer.parseInt(numOfCreditHourstf.getText());//number of credit hours
        daysPresent = Integer.parseInt(daysPresenttf.getText());//number of days present


        //CREATING AND UPCASTING A NEW Regular STUDENT OBJECT
        Student regStud = new Regular(regEnrID,regDateOfBirth,regCourseName,regStudName,regDateOfEnrollment,regCourseDuration,
                          regTuitionFee,numOfModules,numOfCreditHours,daysPresent);

        //CHECKING IF THE USER ENTERS A NEGATIVE ENROLLMENT ID
        if(regEnrID >= 0){
            //displaying message to notify the user
            JOptionPane.showMessageDialog(f,"Invalid enrollment ID detected; please enter a non-negative value.","ERROR",JOptionPane.ERROR_MESSAGE);
        }

        else{
            //CHECKING IF STUDENT ARRAYLIST IS EMPTY
            if (studList.isEmpty()){
                studList.add(regStud);//adding student to student arraylist

                //displaying message to notify the user
                JOptionPane.showMessageDialog(f,"Successfully added regular student.","Success",JOptionPane.INFORMATION_MESSAGE);
            }

            //IF STUDENT LIST IS NOT EMPTY
            else{
                boolean isDuplicate = false;//setting a flag
```

Figure 53: Incorrect comparison operator, '>=' being used in the code

6.3.2. Correction

The correction was done by using the correct comparison operator ‘<=’ in the code.

```
    Student regStud = new Regular(regEnrID,regDateOfBirth,regCourseName,regStudName,regDateOfEnrollment,regCourseDuration,
                      regTuitionFee,numOfModules,numOfCreditHours,daysPresent);

    //CHECKING IF THE USER ENTERS A NEGATIVE ENROLLMENT ID
    if(regEnrID <= 0){
        //displaying message to notify the user
        JOptionPane.showMessageDialog(f,"Invalid enrollment ID detected; please enter a non-negative value.","ERROR",JOptionPane.ERROR_MESSAGE);
    }

    else{
        //CHECKING IF STUDENT ARRAYLIST IS EMPTY
        if (studList.isEmpty()){
            studList.add(regStud);//adding student to student arraylist

            //displaying message to notify the user
            JOptionPane.showMessageDialog(f,"Successfully added regular student.","Success",JOptionPane.INFORMATION_MESSAGE);
        }

        //IF STUDENT LIST IS NOT EMPTY
        else{
            boolean isDuplicate = false;//setting a flag

            for(Student stud:studList){
```

Figure 54: Using the correct comparison operator, '<=' in the code

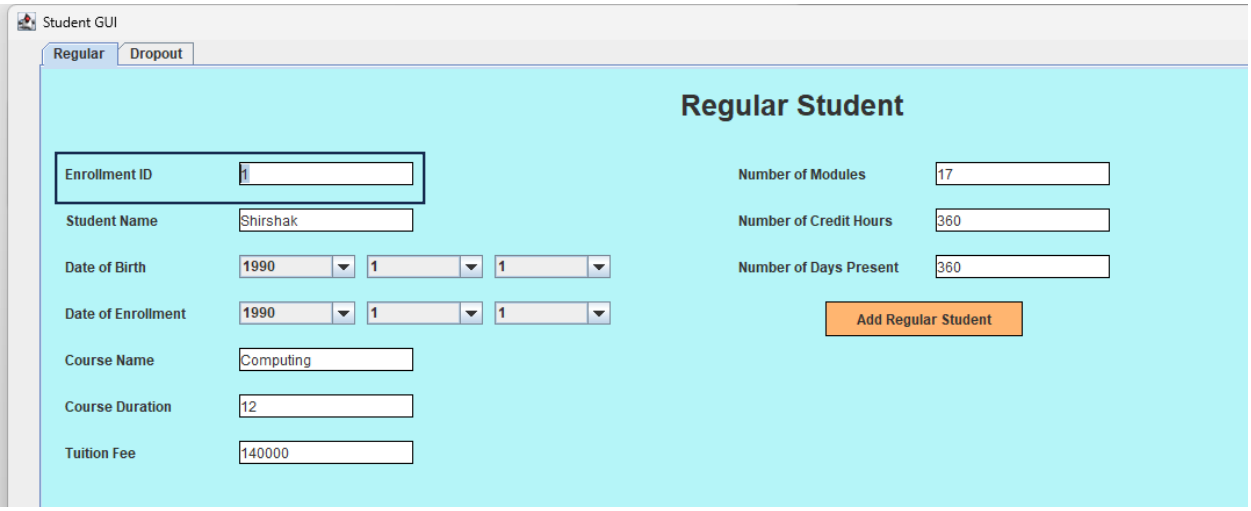


Figure 55: Attempting to add a Regular student again, with all valid values including a non-negative enrollment ID

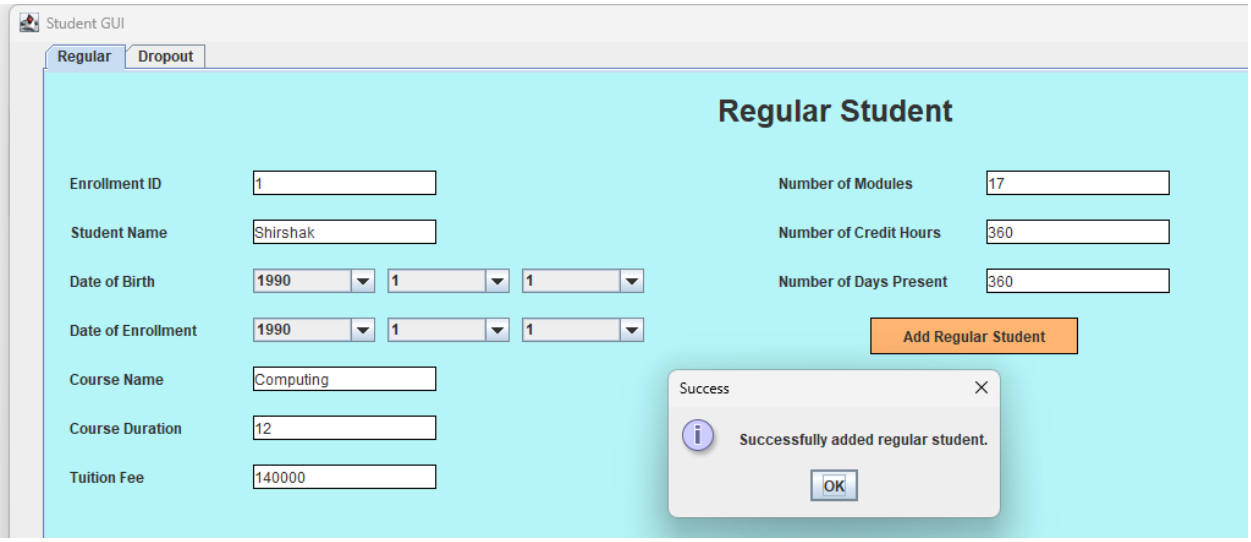


Figure 56: Successfully adding the Regular student

7. Conclusion

7.1. Things I learned

Creating this project has made me realize and learn some very essential things regarding programming (which can be used in other types of work as well). They are listed below in a rough order of priority:

a. **Pre-planning your work:** I have come to realize that just jumping into coding is a bad idea, and almost always leads to hassles, confusion, and difficulties as the volume of the code increases. Planning your work in advance helps to avoid unnecessary struggle in coding, even if the planning is a few moments of thinking right before typing the code.

b. **Comments:** Adding comments to your code is another important aspect of coding that helps avoid confusion in the long run. Adding comments to your code and making their meaning as clear as possible is important, especially because there are many instances where one does not understand his own program just by looking at the code. It is also important to comment to clarify the code to other people who may read it.

c. **Organizing sections of the code:** Unorganized code looks messy, which then leads to confusion later. It is essential to organize one's code to make it appear de-cluttered and non-congested as possible. Since whitespace does not affect the execution of the code, they can be added in any number in between different sections of the code to organize it. Even comments can be used as a separator between the different sections of the code.

Learning how to create a GUI using Java has been an entertaining task along with it being educational. I learned a lot about the different components of a GUI, how they can be put together to make the GUI look organized and intuitive to use, and how to decorate the components as needed by altering different properties such as their border style, background color, font style, etc.

In a broader sense, this project has helped me familiarize myself with more advanced concepts of Java, and Object-Oriented Programming in general.

7.2. Difficulties faced and their solutions

As mentioned above, avoiding pre-planning my code led me to face several issues in my program. The issues I faced were mostly logical errors, with a few of them throwing out a few kinds of exceptions like NumberFormatException, NullPointerException, ConcurrentModificationException, etc. The causes of these issues ranged from minor logical inaccuracies to insufficient understanding of things like Array lists, access modifiers, the working of instance methods, etc. These issues led to even more hassles later in debugging as well as maintaining the code.

The issues were fortunately solved through help from various sources. The most important one of those sources were the lecturers and tutors in our college, who eagerly helped in clarifying misconceptions and hinting at ideas to solve problems in the code. They have also led a helping hand in the formulation of this report. I express my gratitude towards them for all their assistance.

Discussion among friends, although not directly about the coursework itself, also helped me think of possible ideas to solve problems in our code, and served as reminders that there are various resources from which I can obtain help on the internet as well, for e.g., Google, ChatGPT, Bard, etc. I express my thankfulness towards their developers for making such resources and services available to the public, and free of charge.

To sum up, various difficulties arose during coding this project, which caused a lot of confusion, frustration, and hassles. They were solved by the help of various resources, especially from the lecturers and tutors in our college. I learnt a few, very important lessons in coding, and hope that this learning process continues without stopping.

8. References

8.1. Bibliography

Anon., 2021. *BlueJ.* [Online] Available at: https://www.bluej.org/about.html [Accessed 6 August 2023].

Anon., 2023. *draw.io.* [Online] Available at: https://www.drawio.com/about [Accessed 6 August 2023].

Anon., 2023. *Microsoft.* [Online] Available at: https://www.microsoft.com [Accessed 6 August 2023].

9. Appendix

9.1. StudentGUI code

```
import javax.swing.*;

import java.util.ArrayList;

import java.awt.event.*;

import java.awt.Font;

import java.awt.Color;


public class StudentGUI implements ActionListener

{



    private JFrame f;



    private JPanel regStudp, dropoutStudp;
```

```
    private JTabbedPane tabbedP;









    private JLabel regStudHeaderlb,

    regStudNamelb, regEnrIDlb, regCourseNamelb, regCourseDurationlb,
    regTuitionFeelb, regDateOfBirthlb, regDateOfEnrollmentlb,

    numOfModuleslb, numOfCreditHourslb, daysPresentlb,

    presentPercentageEnrIDlb, presentPercentageDaysPresentlb,
    grantCertEnrIDlb, grantCertDateOfEnrollmentlb, grantCertCourseNamelb;



    private JLabel dropoutStudHeaderlb,
```

dropoutStudNamelb, dropoutEnrIDlb, dropoutCourseNamelb, dropoutCourseDurationlb,

dropoutTuitionFeelb, dropoutDateOfBirthlb, dropoutDateOfEnrollmentlb,

dateOfDropoutlb, numOfRemainingModuleslb, numOfMonthsAttendedlb, remainingAmountlb,

payBillsEnrIDlb, removeStudEnrIDlb;

private JTextField regStudNametf, regEnrIDtf, regCourseNametf,

regCourseDurationtf, regTuitionFeetf,

numOfModulestf, numOfCreditHourstf, daysPresenttf,

presentPercentageEnrIDtf, presentPercentageDaysPresenttf,

grantCertEnrIDtf, grantCertCourseNametf;

private JTextField dropoutStudNametf, dropoutEnrIDtf, dropoutCourseNametf, dropoutCourseDurationtf, dropoutTuitionFeetf,

numOfRemainingModulestf, numOfMonthsAttendedtf, remainingAmounttf,

payBillsEnrIDtf, removeStudEnrIDtf;

private JComboBox<String> regYearOfBirthcb, regMonthOfBirthcb, regDayOfBirthcb,

regYearOfEnrollmentcb, regMonthOfEnrollmentcb, regDayOfEnrollmentcb,

grantCertYearOfEnrollmentcb, grantCertMonthOfEnrollmentcb, grantCertDayOfEnrollmentcb;

private JComboBox<String> dropoutYearOfBirthcb, dropoutMonthOfBirthcb, dropoutDayOfBirthcb,

dropoutYearOfEnrollmentcb, dropoutMonthOfEnrollmentcb, dropoutDayOfEnrollmentcb,

yearOfDropoutcb, monthOfDropoutcb, dayOfDropoutcb;

```
    private JButton addRegStudbtn, presentPercentagebtn, grantCertbtn,
    regDisplaybtn, regClearbtn;

    private JButton addDropoutStudbtn, removeStudbtn, payBillsbtn,
    dropoutDisplaybtn, dropoutClearbtn;

    private String regStudName, regDateOfBirth, regDateOfEnrollment, regCourseName,
    grantCertDateOfEnrollment, grantCertCourseName,

    regYearOfBirth, regMonthOfBirth, regDayOfBirth,
    regYearOfEnrollment, regMonthOfEnrollment, regDayOfEnrollment,

    grantCertYearOfEnrollment, grantCertMonthOfEnrollment, grantCertDayOfEnrollment;
```

private int regEnrID, regCourseDuration, regTuitionFee, numOfModules, numOfCreditHours, daysPresent,

presentPercentageEnrID, presentPercentageDaysPresent, grantCertEnrID;

private String dropoutStudName, dropoutDateOfBirth, dropoutDateOfEnrollment, dropoutCourseName, dateOfDropout,

dropoutYearOfBirth, dropoutMonthOfBirth, dropoutDayOfBirth,

dropoutYearOfEnrollment, dropoutMonthOfEnrollment, dropoutDayOfEnrollment,

yearOfDropout, monthOfDropout, dayOfDropout;

private int dropoutEnrID, dropoutCourseDuration, dropoutTuitionFee, numOfRemainingModules, numOfMonthsAttended, remainingAmount,

payBillsEnrID, removeStudEnrID;

```
    private ArrayList<Student> studList = new ArrayList<Student>();



    public StudentGUI()
    {



        f = new JFrame("Student GUI");









        regStudp = new JPanel();
        dropoutStudp = new JPanel();
```

```
regStudHeaderlb = new JLabel("Regular Student");
regStudHeaderlb.setFont(new Font("Arial",Font.BOLD,25));

regStudNamelb = new JLabel("Student Name");
regEnrIDlb = new JLabel("Enrollment ID");
presentPercentageEnrIDlb = new JLabel("Enrollment ID");
regCourseNamelb = new JLabel("Course Name");
regCourseDurationlb = new JLabel("Course Duration");
regTuitionFeelb = new JLabel("Tuition Fee");
regDateOfBirthlb = new JLabel("Date of Birth");
regDateOfEnrollmentlb = new JLabel("Date of Enrollment");

grantCertEnrIDlb = new JLabel("Enrollment ID");
grantCertDateOfEnrollmentlb = new JLabel("Date of Enrollment");
grantCertCourseNamelb = new JLabel("Course Name");
```

```
numOfModuleslb = new JLabel("Number of Modules");
numOfCreditHourslb = new JLabel("Number of Credit Hours");
daysPresentlb = new JLabel("Number of Days Present");
presentPercentageDaysPresentlb = new JLabel("Number of Days Present");



dropoutStudHeaderlb = new JLabel("Dropout Student");
dropoutStudHeaderlb.setFont(new Font("Arial",Font.BOLD,25));

dropoutStudNamelb = new JLabel("Student Name");
dropoutEnrIDlb = new JLabel("Enrollment ID");
dropoutCourseNamelb = new JLabel("Course Name");
dropoutCourseDurationlb = new JLabel("Course Duration");
dropoutTuitionFeelb = new JLabel("Tuition Fee");
dropoutDateOfBirthlb = new JLabel("Date of Birth");
dropoutDateOfEnrollmentlb = new JLabel("Date of Enrollment");

dateOfDropoutlb = new JLabel("Date of Dropout");
numOfRemainingModuleslb = new JLabel("Number of Remaining Modules");
numOfMonthsAttendedlb = new JLabel("Number of Months Attended");
remainingAmountlb = new JLabel("Remaining Amount");
```

```
payBillsEnrIDlb = new JLabel("Enrollment ID");
removeStudEnrIDlb = new JLabel("Enrollment ID");









regStudNametf = new JTextField();
regEnrIDtf = new JTextField();
presentPercentageEnrIDtf = new JTextField();
regCourseNametf = new JTextField();
regCourseDurationtf = new JTextField();
regTuitionFeetf = new JTextField();

grantCertEnrIDtf = new JTextField();
grantCertCourseNametf = new JTextField();

numOfModulestf = new JTextField();
numOfCreditHourstf = new JTextField();
```

```
daysPresenttf = new JTextField();


presentPercentageDaysPresenttf = new JTextField();



dropoutStudNametf = new JTextField();
dropoutEnrIDtf = new JTextField();
dropoutCourseNametf = new JTextField();
dropoutCourseDurationtf = new JTextField();
dropoutTuitionFeetf = new JTextField();

numOfRemainingModulestf = new JTextField();
numOfMonthsAttendedtf = new JTextField();
remainingAmounttf = new JTextField();

payBillsEnrIDtf = new JTextField();
removeStudEnrIDtf = new JTextField();
```

```
addRegStudbtn = new JButton("Add Regular Student");
presentPercentagebtn = new JButton("Present Percentage");
grantCertbtn = new JButton("Grant Certificate");

regDisplaybtn = new JButton("Display");
regClearbtn = new JButton("Clear Fields");


addDropoutStudbtn = new JButton("Add Dropout Student");
payBillsbtn = new JButton("Pay Bills");
removeStudbtn = new JButton("Remove Student");

dropoutDisplaybtn = new JButton("Display");
dropoutClearbtn = new JButton("Clear Fields");
```

```
String[] yearar = new String[50];//years
String[] monthar = new String[12];//months
String[] dayar = new String[31];//days




int year = 1990;
for(int i=0; i<50; i++){
    yearar[i] = String.valueOf(year);
    year++;
}

int month = 1;
for(int i=0; i<=11; i++){
```

```
        monthar[i] = String.valueOf(month);
        month++;
    }


    int day = 1;
    for(int i=0; i<=30; i++){
        dayar[i] = String.valueOf(day);
        day++;
    }




    regYearOfBirthcb = new JComboBox<String>(yearar);
    regMonthOfBirthcb = new JComboBox<String>(monthar);
    regDayOfBirthcb = new JComboBox<String>(dayar);

    regYearOfEnrollmentcb = new JComboBox<String>(yearar);
    regMonthOfEnrollmentcb = new JComboBox<String>(monthar);
    regDayOfEnrollmentcb = new JComboBox<String>(dayar);
```

```
grantCertYearOfEnrollmentcb = new JComboBox<String>(yearar);
grantCertMonthOfEnrollmentcb = new JComboBox<String>(monthar);
grantCertDayOfEnrollmentcb = new JComboBox<String>(dayar);



//DROPOUT PANEL
dropoutYearOfBirthcb = new JComboBox<String>(yearar);
dropoutMonthOfBirthcb = new JComboBox<String>(monthar);
dropoutDayOfBirthcb = new JComboBox<String>(dayar);

dropoutYearOfEnrollmentcb = new JComboBox<String>(yearar);
dropoutMonthOfEnrollmentcb = new JComboBox<String>(monthar);
dropoutDayOfEnrollmentcb = new JComboBox<String>(dayar);

yearOfDropoutcb = new JComboBox<String>(yearar);
monthOfDropoutcb = new JComboBox<String>(monthar);
dayOfDropoutcb = new JComboBox<String>(dayar);
```

```
regStudHeaderlb.setBounds(550,20,200,25);

regEnrIDlb.setBounds(20,80,100,20);
regEnrIDtf.setBounds(170,80,150,20);

regStudNamelb.setBounds(20,120,100,20);
regStudNametf.setBounds(170,120,150,20);

regDateOfBirthlb.setBounds(20,160,100,20);
regYearOfBirthcb.setBounds(170,160,100,20);
regMonthOfBirthcb.setBounds(280,160,100,20);
regDayOfBirthcb.setBounds(390,160,100,20);

regDateOfEnrollmentlb.setBounds(20,200,150,20);
regYearOfEnrollmentcb.setBounds(170,200,100,20);
regMonthOfEnrollmentcb.setBounds(280,200,100,20);
regDayOfEnrollmentcb.setBounds(390,200,100,20);
```

```
regCourseNamelb.setBounds(20,240,100,20);
regCourseNametf.setBounds(170,240,150,20);

regCourseDurationlb.setBounds(20,280,100,20);
regCourseDurationtf.setBounds(170,280,150,20);

regTuitionFeelb.setBounds(20,320,100,20);
regTuitionFeetf.setBounds(170,320,150,20);

numOfModuleslb.setBounds(600,80,150,20);
numOfModulestf.setBounds(770,80,150,20);

numOfCreditHourslb.setBounds(600,120,150,20);
numOfCreditHourstf.setBounds(770,120,150,20);

daysPresentlb.setBounds(600,160,150,20);
daysPresenttf.setBounds(770,160,150,20);

addRegStudbtn.setBounds(675,200,170,30);

presentPercentageEnrIDlb.setBounds(20,440,150,20);
presentPercentageEnrIDtf.setBounds(170,440,150,20);
```

```
presentPercentageDaysPresentlb.setBounds(20,480,150,20);
presentPercentageDaysPresenttf.setBounds(170,480,150,20);

presentPercentagebtn.setBounds(80,520,170,30);


grantCertEnrIDlb.setBounds(600,400,100,20);
grantCertEnrIDtf.setBounds(770,400,150,20);

grantCertDateOfEnrollmentlb.setBounds(600,440,150,20);
grantCertYearOfEnrollmentcb.setBounds(770,440,100,20);
grantCertMonthOfEnrollmentcb.setBounds(880,440,100,20);
grantCertDayOfEnrollmentcb.setBounds(990,440,100,20);

grantCertCourseNamelb.setBounds(600,480,100,20);
grantCertCourseNametf.setBounds(770,480,150,20);

grantCertbtn.setBounds(680,520,170,30);

regDisplaybtn.setBounds(1060,600,100,25);
regClearbtn.setBounds(1170,600,100,25);
```

```
dropoutStudHeaderlb.setBounds(550,20,200,25);

dropoutEnrIDlb.setBounds(20,80,100,20);
dropoutEnrIDtf.setBounds(170,80,150,20);

dropoutStudNamelb.setBounds(20,120,100,20);
dropoutStudNametf.setBounds(170,120,150,20);

dropoutDateOfBirthlb.setBounds(20,160,100,20);
dropoutYearOfBirthcb.setBounds(170,160,100,20);
dropoutMonthOfBirthcb.setBounds(280,160,100,20);
dropoutDayOfBirthcb.setBounds(390,160,100,20);

dropoutDateOfEnrollmentlb.setBounds(20,200,150,20);
dropoutYearOfEnrollmentcb.setBounds(170,200,100,20);
dropoutMonthOfEnrollmentcb.setBounds(280,200,100,20);
dropoutDayOfEnrollmentcb.setBounds(390,200,100,20);

dropoutCourseNamelb.setBounds(20,240,100,20);
dropoutCourseNametf.setBounds(170,240,150,20);

dropoutCourseDurationlb.setBounds(20,280,100,20);
```

```
dropoutCourseDurationtf.setBounds(170,280,150,20);

dropoutTuitionFeelb.setBounds(20,320,100,20);
dropoutTuitionFeetf.setBounds(170,320,150,20);

numOfRemainingModuleslb.setBounds(600,80,200,20);
numOfRemainingModulestf.setBounds(800,80,150,20);

numOfMonthsAttendedlb.setBounds(600,120,200,20);
numOfMonthsAttendedtf.setBounds(800,120,150,20);

remainingAmountlb.setBounds(600,160,150,20);
remainingAmounttf.setBounds(800,160,150,20);

dateOfDropoutlb.setBounds(600,200,100,20);
yearOfDropoutcb.setBounds(800,200,100,20);
monthOfDropoutcb.setBounds(910,200,100,20);
dayOfDropoutcb.setBounds(1020,200,100,20);

addDropoutStudbtn.setBounds(675,240,170,30);

payBillsEnrIDlb.setBounds(20,480,100,20);
payBillsEnrIDtf.setBounds(170,480,150,20);
```

```
payBillsbtn.setBounds(80,520,170,30);


removeStudEnrIDlb.setBounds(620,480,100,20);
removeStudEnrIDtf.setBounds(770,480,150,20);
removeStudbtn.setBounds(680,520,170,30);


dropoutDisplaybtn.setBounds(1060,600,100,25);
dropoutClearbtn.setBounds(1170,600,100,25);









addRegStudbtn.addActionListener(this);
presentPercentagebtn.addActionListener(this);
grantCertbtn.addActionListener(this);


regDisplaybtn.addActionListener(this);
regClearbtn.addActionListener(this);
```

```
addDropoutStudbtn.addActionListener(this);
payBillsbtn.addActionListener(this);
removeStudbtn.addActionListener(this);

dropoutDisplaybtn.addActionListener(this);
dropoutClearbtn.addActionListener(this);









regStudp.add(regStudHeaderlb);

regStudp.add(regEnrIDlb);
regStudp.add(regEnrIDtf);

regStudp.add(regStudNamelb);
regStudp.add(regStudNametf);
```

```
        regStudp.add(regDateOfBirthlb);
        regStudp.add(regYearOfBirthcb);
        regStudp.add(regMonthOfBirthcb);
        regStudp.add(regDayOfBirthcb);

        regStudp.add(regDateOfEnrollmentlb);
        regStudp.add(regYearOfEnrollmentcb);
        regStudp.add(regMonthOfEnrollmentcb);
        regStudp.add(regDayOfEnrollmentcb);

        regStudp.add(regCourseNamelb);
        regStudp.add(regCourseNametf);

        regStudp.add(regCourseDurationlb);
        regStudp.add(regCourseDurationtf);

        regStudp.add(numOfModuleslb);
        regStudp.add(numOfModulestf);

        regStudp.add(numOfCreditHourslb);
        regStudp.add(numOfCreditHourstf);

        regStudp.add(regTuitionFeelb);
```

```
regStudp.add(regTuitionFeetf);


regStudp.add(daysPresentlb);
regStudp.add(daysPresenttf);


regStudp.add(addRegStudbtn);


regStudp.add(presentPercentageEnrIDlb);
regStudp.add(presentPercentageEnrIDtf);
regStudp.add(presentPercentageDaysPresentlb);
regStudp.add(presentPercentageDaysPresenttf);
regStudp.add(presentPercentagebtn);


regStudp.add(grantCertEnrIDlb);
regStudp.add(grantCertEnrIDtf);
regStudp.add(grantCertDateOfEnrollmentlb);
regStudp.add(grantCertYearOfEnrollmentcb);
regStudp.add(grantCertMonthOfEnrollmentcb);
regStudp.add(grantCertDayOfEnrollmentcb);
regStudp.add(grantCertCourseNamelb);
regStudp.add(grantCertCourseNametf);
regStudp.add(grantCertbtn);
```

```
regStudp.add(regDisplaybtn);
regStudp.add(regClearbtn);



dropoutStudp.add(dropoutStudHeaderlb);

dropoutStudp.add(dropoutEnrIDlb);
dropoutStudp.add(dropoutEnrIDtf);

dropoutStudp.add(dropoutStudNamelb);
dropoutStudp.add(dropoutStudNametf);

dropoutStudp.add(dropoutDateOfBirthlb);
dropoutStudp.add(dropoutYearOfBirthcb);
dropoutStudp.add(dropoutMonthOfBirthcb);
dropoutStudp.add(dropoutDayOfBirthcb);

dropoutStudp.add(dropoutDateOfEnrollmentlb);
dropoutStudp.add(dropoutYearOfEnrollmentcb);
dropoutStudp.add(dropoutMonthOfEnrollmentcb);
dropoutStudp.add(dropoutDayOfEnrollmentcb);
```

```
dropoutStudp.add(dropoutCourseNamelb);
dropoutStudp.add(dropoutCourseNametf);

dropoutStudp.add(dropoutCourseDurationlb);
dropoutStudp.add(dropoutCourseDurationtf);

dropoutStudp.add(numOfRemainingModuleslb);
dropoutStudp.add(numOfRemainingModulestf);

dropoutStudp.add(numOfMonthsAttendedlb);
dropoutStudp.add(numOfMonthsAttendedtf);

dropoutStudp.add(dropoutTuitionFeelb);
dropoutStudp.add(dropoutTuitionFeetf);

dropoutStudp.add(remainingAmountlb);
dropoutStudp.add(remainingAmounttf);

dropoutStudp.add(dateOfDropoutlb);
dropoutStudp.add(yearOfDropoutcb);
dropoutStudp.add(monthOfDropoutcb);
dropoutStudp.add(dayOfDropoutcb);
```

```
dropoutStudp.add(addDropoutStudbtn);


dropoutStudp.add(payBillsEnrIDlb);
dropoutStudp.add(payBillsEnrIDtf);
dropoutStudp.add(payBillsbtn);


dropoutStudp.add(removeStudEnrIDlb);
dropoutStudp.add(removeStudEnrIDtf);
dropoutStudp.add(removeStudbtn);


dropoutStudp.add(dropoutDisplaybtn);
dropoutStudp.add(dropoutClearbtn);









regStudp.setBackground(new Color(181,245,248));
dropoutStudp.setBackground(new Color(181,245,248));
```

```
addRegStudbtn.setBackground(new Color(255,181,112));
presentPercentagebtn.setBackground(new Color(255,181,112));
grantCertbtn.setBackground(new Color(255,181,112));

regDisplaybtn.setBackground(new Color(255,204,153));
regClearbtn.setBackground(new Color(255,204,153));

addDropoutStudbtn.setBackground(new Color(255,181,112));
payBillsbtn.setBackground(new Color(255,181,112));
removeStudbtn.setBackground(new Color(255,181,112));

dropoutDisplaybtn.setBackground(new Color(255,204,153));
dropoutClearbtn.setBackground(new Color(255,204,153));
```

```
        regEnrIDtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        regStudNametf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        regCourseNametf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        regCourseDurationtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        regTuitionFeetf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        numOfModulestf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        numOfCreditHourstf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        daysPresenttf.setBorder(BorderFactory.createLineBorder(Color.BLACK));


presentPercentageEnrIDtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));

presentPercentageDaysPresenttf.setBorder(BorderFactory.createLineBorder(Color.BLA
CK));

grantCertEnrIDtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));

grantCertCourseNametf.setBorder(BorderFactory.createLineBorder(Color.BLACK));


        dropoutEnrIDtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
```

```
        dropoutStudNametf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        dropoutCourseNametf.setBorder(BorderFactory.createLineBorder(Color.BLACK));

dropoutCourseDurationtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
dropoutTuitionFeetf.setBorder(BorderFactory.createLineBorder(Color.BLACK));

numOfRemainingModulestf.setBorder(BorderFactory.createLineBorder(Color.BLACK));

numOfMonthsAttendedtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        remainingAmounttf.setBorder(BorderFactory.createLineBorder(Color.BLACK));

        payBillsEnrIDtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        removeStudEnrIDtf.setBorder(BorderFactory.createLineBorder(Color.BLACK));




        addRegStudbtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));

        presentPercentagebtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        grantCertbtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));
```

```
regDisplaybtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));
regClearbtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));



addDropoutStudbtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));
payBillsbtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));
removeStudbtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));

dropoutDisplaybtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));
dropoutClearbtn.setBorder(BorderFactory.createLineBorder(Color.BLACK));









regStudp.setSize(1300,720);
```

```
        regStudp.setLayout(null);




        dropoutStudp.setSize(1300,720);
        dropoutStudp.setLayout(null);









        tabbedP = new JTabbedPane(JTabbedPane.TOP);


        tabbedP.addTab("Regular",regStudp);
        tabbedP.addTab("Dropout",dropoutStudp);
```

```
        tabbedP.setVisible(true);
        tabbedP.setBounds(27,0,1300,670);




        f.add(tabbedP);



        f.setSize(1366,726);
        f.setLayout(null);
        f.setResizable(false);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


    }


    @Override
```

```
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == addRegStudbtn){

        boolean emptyTextfields = regEnrIDtf.getText().isEmpty() || regStudNametf.getText().isEmpty() || regCourseNametf.getText().isEmpty()
        || regCourseDurationtf.getText().isEmpty() || regTuitionFeetf.getText().isEmpty() || numOfModulestf.getText().isEmpty()
        || numOfCreditHourstf.getText().isEmpty() || daysPresenttf.getText().isEmpty();

        if(emptyTextfields){

            JOptionPane.showMessageDialog(f, "Empty fields detected; please fill out all the fields.","Error",JOptionPane.ERROR_MESSAGE);
        }
        else{
            regStudName = regStudNametf.getText();

            regYearOfBirth = (String) regYearOfBirthcb.getSelectedItem();
            regMonthOfBirth = (String) regMonthOfBirthcb.getSelectedItem();
            regDayOfBirth = (String) regDayOfBirthcb.getSelectedItem();
            regDateOfBirth = regYearOfBirth + "-" + regMonthOfBirth + "-" + regDayOfBirth;
```

```
            regYearOfEnrollment = (String) regYearOfEnrollmentcb.getSelectedItem();
            regMonthOfEnrollment = (String) regMonthOfEnrollmentcb.getSelectedItem();
            regDayOfEnrollment = (String) regDayOfEnrollmentcb.getSelectedItem();
            regDateOfEnrollment = regYearOfEnrollment + "-" + regMonthOfEnrollment +
"-" + regDayOfEnrollment;

            regCourseName = regCourseNametf.getText();



            try{
               regEnrID = Integer.parseInt(regEnrIDtf.getText());

               regCourseDuration = Integer.parseInt(regCourseDurationtf.getText());
               regTuitionFee = Integer.parseInt(regTuitionFeetf.getText());

               numOfModules = Integer.parseInt(numOfModulestf.getText());
               numOfCreditHours = Integer.parseInt(numOfCreditHourstf.getText());
               daysPresent = Integer.parseInt(daysPresenttf.getText());


               Student regStud = new
Regular(regEnrID,regDateOfBirth,regCourseName,regStudName,regDateOfEnrollment,
regCourseDuration,
```

```
                    regTuitionFee,numOfModules,numOfCreditHours,daysPresent);

            if(regEnrID <= 0){

                JOptionPane.showMessageDialog(f,"Invalid enrollment ID detected; please enter a non-negative value.","ERROR",JOptionPane.ERROR_MESSAGE);
            }

            else{
                if (studList.isEmpty()){
                    studList.add(regStud);

                    JOptionPane.showMessageDialog(f,"Successfully added regular student.","Success",JOptionPane.INFORMATION_MESSAGE);
                }

                else{
                    boolean isDuplicate = false;

                    for(Student stud:studList){
```

```
                    if(stud instanceof Regular){


                        if(stud.getEnrollmentID() == regStud.getEnrollmentID()){
                            isDuplicate = true;
                            break;
                        }
                    }
                }

                if(isDuplicate == false){
                    studList.add(regStud);



                    JOptionPane.showMessageDialog(f,"Successfully added regular student.","Success",JOptionPane.INFORMATION_MESSAGE);
                }
                else{


                    JOptionPane.showMessageDialog(f,"Duplicate enrollment ID detected; please enter a valid enrollment ID",
                                                  "Error",JOptionPane.ERROR_MESSAGE);
                }
            }
```

```
                }
            }catch(NumberFormatException regularEx){

                JOptionPane.showMessageDialog(f, "Invalid input detected; please enter numbers.","Error",JOptionPane.ERROR_MESSAGE);
            }
        }
    }
```

```
    else if(e.getSource() == presentPercentagebtn){
```

```
boolean emptyTextfields = presentPercentageEnrIDtf.getText().isEmpty() || presentPercentageDaysPresenttf.getText().isEmpty();
if(emptyTextfields){

   JOptionPane.showMessageDialog(f,"Empty fields detected; please fill all the required fields.","Error",JOptionPane.ERROR_MESSAGE);
}
else{
   try{
      presentPercentageEnrID = Integer.parseInt(presentPercentageEnrIDtf.getText());
      presentPercentageDaysPresent = Integer.parseInt(presentPercentageDaysPresenttf.getText());

      if(studList.isEmpty()){

         JOptionPane.showMessageDialog(f,"List of students is empty.","Error",JOptionPane.ERROR_MESSAGE);
      }
      else{
         boolean calculatedPresentPercentage = false;

         for(Student stud:studList){
```

```
                if(stud instanceof Regular){
                    Regular regStud = (Regular) stud;



                    if(stud.getEnrollmentID() == presentPercentageEnrID){
                        calculatedPresentPercentage = true;

                        String presentPercentageMessage =
String.valueOf(regStud.presentPercentage(presentPercentageDaysPresent));

JOptionPane.showMessageDialog(f,presentPercentageMessage,"Attendance
Grade",JOptionPane.INFORMATION_MESSAGE);
                        break;
                    }
                }
            }


            if(calculatedPresentPercentage == false){

                JOptionPane.showMessageDialog(f,"No regular student with given
enrollment ID found; please enter a valid enrollment ID.",
                                              "Error",JOptionPane.ERROR_MESSAGE);
```

```
                }
            }
        }catch(NumberFormatException presentPercentageEx){

            JOptionPane.showMessageDialog(f,"Invalid input detected; please enter numbers.","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

```
else if(e.getSource() == grantCertbtn){
```

```
boolean emptyTextfields = grantCertCourseNametf.getText().isEmpty() || grantCertEnrIDtf.getText().isEmpty();


if(emptyTextfields){
   JOptionPane.showMessageDialog(f,"Empty textfields detected; please fill all the required fields.","Error",JOptionPane.ERROR_MESSAGE);
}
else{

   grantCertCourseName = grantCertCourseNametf.getText();


   grantCertYearOfEnrollment = (String) grantCertYearOfEnrollmentcb.getSelectedItem();
   grantCertMonthOfEnrollment = (String) grantCertMonthOfEnrollmentcb.getSelectedItem();
   grantCertDayOfEnrollment = (String) grantCertDayOfEnrollmentcb.getSelectedItem();


   grantCertDateOfEnrollment = grantCertYearOfEnrollment + "-" + grantCertMonthOfEnrollment + "-" + grantCertDayOfEnrollment;


   try{
      grantCertEnrID = Integer.parseInt(grantCertEnrIDtf.getText());
```

```
            if(studList.isEmpty()){

                JOptionPane.showMessageDialog(f,"List of students is empty.","Error",JOptionPane.ERROR_MESSAGE);
            }
            else{
                boolean grantedCertificate = false;

                for(Student stud:studList){

                    if(stud instanceof Regular){
                        Regular regStud = (Regular) stud;


                        if(regStud.getEnrollmentID() == grantCertEnrID){
                            grantedCertificate = true;


                            String grantedCertificateMessage = regStud.grantCertificate(grantCertCourseName,grantCertEnrID,grantCertDateOfEnrollment);
```

```
JOptionPane.showMessageDialog(f,grantedCertificateMessage,"Grant
Certificate",JOptionPane.INFORMATION_MESSAGE);

                    break;
                }
            }
        }


        if(grantedCertificate == false){

            JOptionPane.showMessageDialog(f,"No regular student with given
enrollment ID found; please enter a valid enrollment ID.",
                                    "Error",JOptionPane.ERROR_MESSAGE);

        }
    }
}catch(NumberFormatException grantCertEx){

    JOptionPane.showMessageDialog(f,"Invalid input detected; please enter
numbers.","Error",JOptionPane.ERROR_MESSAGE);
    }
}
```

```
    }
```

```
else if(e.getSource() == regDisplaybtn){
    boolean displayedStudent = false;

    for(Student stud:studList){


        if(stud instanceof Regular){
            displayedStudent = true;

            Regular regStud = (Regular) stud;
```

```
                String displayInfo = regStud.display();



                JOptionPane.showMessageDialog(f,displayInfo,"Student
Info",JOptionPane.INFORMATION_MESSAGE);

            }
        }



        if(displayedStudent == true){

            JOptionPane.showMessageDialog(f,"Successfully displayed information of
Regular student.","Success",JOptionPane.INFORMATION_MESSAGE);
        }



        else{

            JOptionPane.showMessageDialog(f,"No regular students found in the student
list.","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
else if(e.getSource()==regClearbtn){
    JOptionPane.showMessageDialog(f,"All fields will be cleared.","Warning",JOptionPane.WARNING_MESSAGE);

    ArrayList<JTextField> textfieldList = new ArrayList<JTextField>();


    textfieldList.add(regEnrIDtf);
    textfieldList.add(regStudNametf);
    textfieldList.add(regCourseNametf);
    textfieldList.add(regCourseDurationtf);
```

```
        textfieldList.add(regTuitionFeetf);
        textfieldList.add(numOfModulestf);
        textfieldList.add(numOfCreditHourstf);
        textfieldList.add(daysPresenttf);

        textfieldList.add(presentPercentageEnrIDtf);
        textfieldList.add(presentPercentageDaysPresenttf);
        textfieldList.add(grantCertEnrIDtf);
        textfieldList.add(grantCertCourseNametf);

        for(JTextField eachField:textfieldList){
            eachField.setText("");
        }
    }
```

```
else if(e.getSource() == addDropoutStudbtn){

    boolean emptyTextfields = dropoutEnrIDtf.getText().isEmpty() ||
dropoutStudNametf.getText().isEmpty() || dropoutCourseNametf.getText().isEmpty()
                    || dropoutCourseDurationtf.getText().isEmpty() ||
dropoutTuitionFeetf.getText().isEmpty()
                    || numOfRemainingModulestf.getText().isEmpty() ||
numOfMonthsAttendedtf.getText().isEmpty()
                    || remainingAmounttf.getText().isEmpty();


    if(emptyTextfields){

      JOptionPane.showMessageDialog(f,"Empty fields detected; please fill out all
the fields.","Error",JOptionPane.ERROR_MESSAGE);
    }
    else{
```

```
        dropoutStudName = dropoutStudNametf.getText();


        dropoutYearOfBirth = (String) dropoutYearOfBirthcb.getSelectedItem();
        dropoutMonthOfBirth = (String) dropoutMonthOfBirthcb.getSelectedItem();
        dropoutDayOfBirth = (String) dropoutDayOfBirthcb.getSelectedItem();
        dropoutDateOfBirth = dropoutYearOfBirth + "-" + dropoutMonthOfBirth + "-" +
dropoutDayOfBirth;


        dropoutYearOfEnrollment = (String)
dropoutYearOfEnrollmentcb.getSelectedItem();
        dropoutMonthOfEnrollment = (String)
dropoutMonthOfEnrollmentcb.getSelectedItem();
        dropoutDayOfEnrollment = (String)
dropoutDayOfEnrollmentcb.getSelectedItem();
        dropoutDateOfEnrollment = dropoutYearOfEnrollment + "-" +
dropoutMonthOfEnrollment + "-" + dropoutDayOfEnrollment;


        yearOfDropout = (String) yearOfDropoutcb.getSelectedItem();
        monthOfDropout = (String) monthOfDropoutcb.getSelectedItem();
        dayOfDropout = (String) dayOfDropoutcb.getSelectedItem();
        dateOfDropout = yearOfDropout + "-" + monthOfDropout + "-" + dayOfDropout;


        dropoutCourseName = dropoutCourseNametf.getText();
```

```
        try{
            dropoutEnrID = Integer.parseInt(dropoutEnrIDtf.getText());

            dropoutCourseDuration =
Integer.parseInt(dropoutCourseDurationtf.getText());
            dropoutTuitionFee = Integer.parseInt(dropoutTuitionFeetf.getText());

            numOfRemainingModules =
Integer.parseInt(numOfRemainingModulestf.getText());
            numOfMonthsAttended =
Integer.parseInt(numOfMonthsAttendedtf.getText());
            remainingAmount = Integer.parseInt(remainingAmounttf.getText());

            Student dropoutStud = new
Dropout(dropoutEnrID,dropoutDateOfBirth,dropoutCourseName,dropoutStudName,drop
outDateOfEnrollment,

dropoutCourseDuration,dropoutTuitionFee,numOfRemainingModules,numOfMonthsAtte
nded,remainingAmount,
                    dateOfDropout);
```

```
            if(dropoutEnrID <= 0){

                JOptionPane.showMessageDialog(f,"Invalid enrollment ID detected; please enter a non-negative value.","ERROR",JOptionPane.ERROR_MESSAGE);
            }

            else{

                if (studList.isEmpty()){
                    studList.add(dropoutStud);

                    JOptionPane.showMessageDialog(f,"Successfully added dropout student.","Success",JOptionPane.INFORMATION_MESSAGE);
                }

                else{
                    boolean isDuplicate = false;

                    for(Student stud:studList){
```

```
                if(stud instanceof Dropout){



                    if(stud.getEnrollmentID() == dropoutStud.getEnrollmentID()){
                        isDuplicate = true;
                        break;
                    }
                }
            }



            if(isDuplicate == false){
                studList.add(dropoutStud);



                JOptionPane.showMessageDialog(f,"Successfully added dropout student.","Success",JOptionPane.INFORMATION_MESSAGE);
            }
            else{


                JOptionPane.showMessageDialog(f,"Duplicate enrollment ID detected; please enter a valid enrollment ID",
                                              "Error",JOptionPane.ERROR_MESSAGE);
```

```
                    }
                }
            }
        }catch(NumberFormatException dropoutEx){

            JOptionPane.showMessageDialog(f, "Invalid input detected; please enter numbers.","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

```
else if(e.getSource() == payBillsbtn){
```

```
        boolean emptyTextfields = payBillsEnrIDtf.getText().isEmpty();



        if(emptyTextfields){

            JOptionPane.showMessageDialog(f,"Empty textfields detected; please fill all the required fields.","Error",JOptionPane.ERROR_MESSAGE);
        }
        else{

            try{
                payBillsEnrID = Integer.parseInt(payBillsEnrIDtf.getText());



                if(studList.isEmpty()){

                    JOptionPane.showMessageDialog(f,"List of students is empty.","Error",JOptionPane.ERROR_MESSAGE);
                }
                else{
                    boolean paidBills = false;

                    for(Student stud:studList){
```

```
                if(stud instanceof Dropout){
                    Dropout dropoutStud = (Dropout) stud;


                    if(dropoutStud.getEnrollmentID() == payBillsEnrID){
                        paidBills = true;

                        String remainingFeeMessage =
dropoutStud.billsPayable();//paying bills of dropoutStud


JOptionPane.showMessageDialog(f,remainingFeeMessage,"Remaining
Fee",JOptionPane.INFORMATION_MESSAGE);

                        break;
                    }
                }
            }
```

```
                if(paidBills == false){


                    JOptionPane.showMessageDialog(f,"No dropout students with given enrollment ID found in the student list.",
                                                  "Error",JOptionPane.ERROR_MESSAGE);
                }
            }
        }catch(NumberFormatException payBillsEx){


            JOptionPane.showMessageDialog(f,"Invalid input detected in dropout; please enter numbers.","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

```
else if(e.getSource() == removeStudbtn){
    boolean emptyTextfields = removeStudEnrIDtf.getText().isEmpty();



    if(emptyTextfields){

        JOptionPane.showMessageDialog(f,"Empty textfields detected; please fill all the required fields.","Error",JOptionPane.ERROR_MESSAGE);
    }
    else{

        try{
            removeStudEnrID = Integer.parseInt(removeStudEnrIDtf.getText());



            if(studList.isEmpty()){

                JOptionPane.showMessageDialog(f,"List of students is empty.","Error",JOptionPane.ERROR_MESSAGE);
            }
```

```
            else{
                boolean isRemoved = false;

                for(Student stud:studList){


                    if(stud instanceof Dropout){
                        Dropout dropoutStud = (Dropout) stud;



                        if(stud.getEnrollmentID() == removeStudEnrID){
                            isRemoved = true;//setting the flag to true

                            String removedStudentMessage = dropoutStud.removeStudent();



JOptionPane.showMessageDialog(f,removedStudentMessage,"Remove
Student",JOptionPane.INFORMATION_MESSAGE);
                            break;
                        }
                    }
                }
```

```
                if(isRemoved == false){

                    JOptionPane.showMessageDialog(f,"No dropout students with the given enrollment ID found in the student list.",
                                                  "Error",JOptionPane.ERROR_MESSAGE);
                }
            }
        }catch(NumberFormatException payBillsEx){

            JOptionPane.showMessageDialog(f,"Invalid input detected in dropout; please enter numbers.","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

```
        else if(e.getSource() == dropoutDisplaybtn){
            boolean displayedStudent = false;

            for(Student stud:studList){


                if(stud instanceof Dropout){
                    displayedStudent = true;

                    Dropout dropoutStud = (Dropout) stud;
                    String displayInfo = dropoutStud.display();


                    JOptionPane.showMessageDialog(f,displayInfo,"Student
Info",JOptionPane.INFORMATION_MESSAGE);

                }
            }
```

```
        if(displayedStudent == true){

            JOptionPane.showMessageDialog(f,"Successfully displayed information of Dropout student.","Success",JOptionPane.INFORMATION_MESSAGE);
        }



        else{

            JOptionPane.showMessageDialog(f,"No dropout students found in the student list.","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
        else{
            JOptionPane.showMessageDialog(f,"All fields will be
cleared.","Warning",JOptionPane.WARNING_MESSAGE);

            ArrayList<JTextField> textfieldList = new ArrayList<JTextField>();



            textfieldList.add(dropoutEnrIDtf);
            textfieldList.add(dropoutStudNametf;
            textfieldList.add(dropoutCourseNametf);
            textfieldList.add(dropoutCourseDurationtf);
            textfieldList.add(dropoutTuitionFeetf);
            textfieldList.add(numOfRemainingModulestf);
            textfieldList.add(numOfMonthsAttendedtf);
            textfieldList.add(remainingAmounttf);

            textfieldList.add(payBillsEnrIDtf);
            textfieldList.add(removeStudEnrIDtf);
```

```
            for(JTextField eachField:textfieldList){
                eachField.setText("");
            }
        }
    }


    public static void main(String[] args)
    {
        new StudentGUI();
    }
}
```