LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2023-24 Spring

Student Name: Aasutosh Kumar Verma
London Met ID: 22085760
College ID: NP01AI4S230020
Group: AI3
Assignment Due Date: Friday, August 11, 2023
Assignment Submission Date: Thursday, August 10, 2023

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

Contents

List of Figures

1. INTRODUCTION

The goal of this course is to evaluate students' fundamental understanding of the Java programming language and the purpose of this coursework is to evaluate students' knowledge of Java's foundations while also assisting them in learning and mastering it. Producing reliable Java classes for Student an organization that creates and remove Student along with a user-friendly GUI is the aim of this project. The following are the tools that were used to guarantee the completion of the coursework:

1.2 BLUEJ

Figure 1: BlueJ Icon

I used BlueJ as a Java Development Environment. I created every class and coded the entire program in BlueJ. It is a tool with a graphical user interface (GUI) that is exceedingly simple to use. There is limit using it but it’s good for educational purpose It finds syntax mistakes, explains what's wrong, and gives a few correction suggestions. It is also easy to manage classes and express information visually. Also got tutorial videos in YouTube for solving troubleshoot or getting ideas while using it. BlueJ further simplifies the coding process by offering features like a built-in debugger which make complex codes easier for users to find errors & save time while solving it.

1.3 MS-Word

Figure 2: MS Word icon

The report component of this coursework is meticulously crafted using Microsoft Word, a software revered for its widespread utilization and user-friendly interface. With a multitude of applications, Word has garnered widespread acclaim for its exceptional attributes. These include a rich array of editing tools, sophisticated styling options, an automated content generator, an intuitive user interface, and a robust search functionality that spans across all aspects. Given its remarkable capabilities, Microsoft Word emerges as the quintessential choice for authoring reports, ensuring a seamless and efficient process.

1.4 Diagram.net

Figure 3: diagram.net

When it comes to crafting diverse graphical representations, diagrams.net (formerly known as draw.io) stands as a widely acknowledged and versatile program. Its capabilities span a broad spectrum, catering not only to uncomplicated endeavors like devising personal mind maps, but also extending to intricate engineering undertakings. In the context of this report, diagrams.net emerged as my preferred tool for constructing a pivotal class diagram. This preference was driven by its effortless accessibility, user-friendly convenience, and remarkable clarity in conveying complex concepts. The seamless synergy between its accessibility, ease of use, and clarity made diagrams.net an invaluable asset in succinctly illustrating the intricate relationships within the system architecture.

2. Class diagram

In the Unified Modeling Language, a class diagram shows how the relationships and source code dependencies between classes (UML). A class, which in this case refers to a particular item in a program or the chunk of code that corresponds to that thing, defines the methods and variables in an object. All types of object-oriented programming can benefit from class diagrams (OOP). Although the idea has been around for a while, it has been improved as OOP modeling paradigms have advanced. The following are the class diagrams of the classes.

2.1 Student

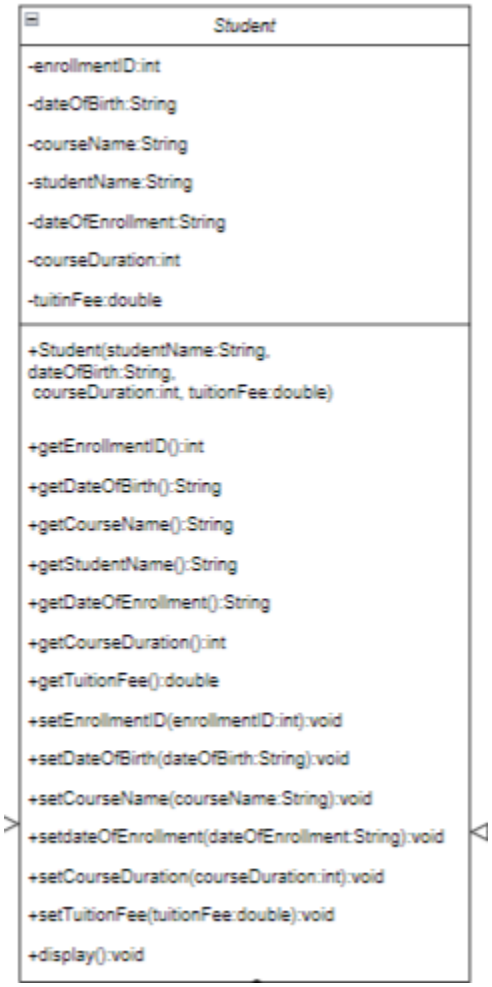

Figure 4: Student Diagram

2.2 Regular

Figure 5: Regular class Diagram

2.3 Dropout

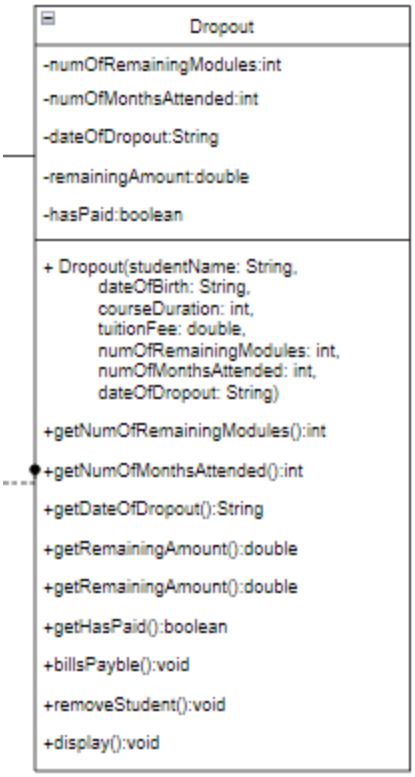

Figure 6: Droopout Class Diagram.

2.4 Student GUI

Figure 7: Student class Diagram.

2.5 Combined class diagram

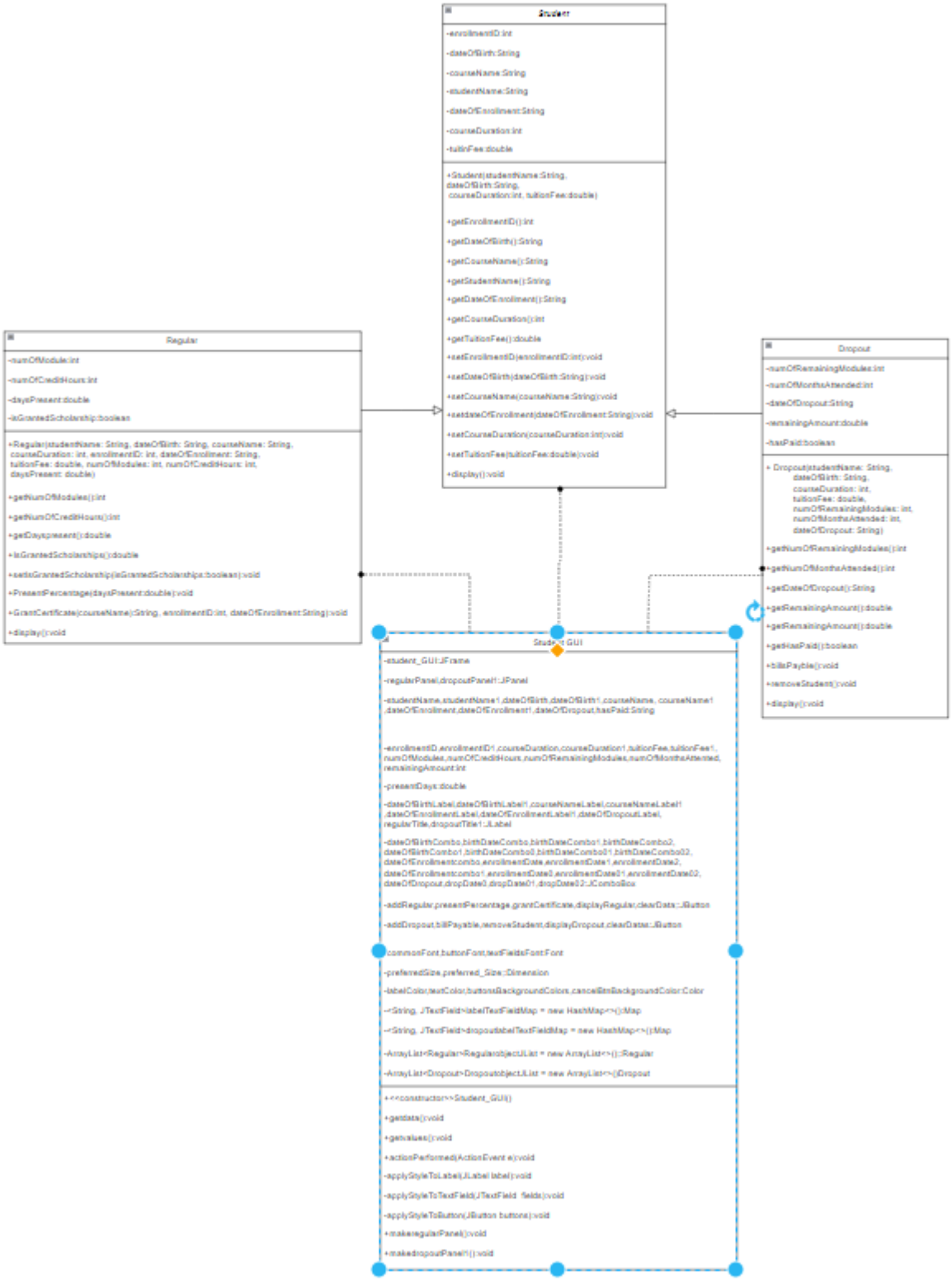

Figure 8: Combined class diagram

3 Pseudocode

(Not exactly code just a short summary)

CREATE public class Student_GUI which implements Action Listener

Declare a class named Student_GUI:

Declare instance variables for UI components:

- A main frame (main window for the application)
- Two panels: one for regular students and one for dropouts
- Labels for displaying static text for various data fields
- Text fields for collecting user input
- Dropdown lists (combo boxes) for date selection
- Buttons for various actions (e.g., adding students, clearing fields)
- Two lists for displaying regular and dropout students
- Variables for storing student data (e.g., student name, date of birth, course name, etc.)
- An ArrayList to store 'Student' objects

Define the constructor for Student_GUI:

Initialize the main frame:

- Set its title as "Student GUI"
- Specify what happens when the frame is closed (exit the application)
- Set its layout to grid layout (1 row, 2 columns)

Initialize the regular student panel:

- Set its layout manager to null (absolute positioning)

Initialize labels:

- Create a title label and other labels for student information like student name, enrollment ID, days present, etc.

Initialize text fields:

- Create text fields for each student data entry (like student name, enrollment ID, course duration, etc.)

Initialize combo boxes for date selection:

- Create dropdowns (combo boxes) for date selection: one each for day, month, and year for enrollment date and date of birth.

Initialize buttons:

- Create action buttons like "Create Regular Student", "Present Percentage", "Grant Certificate", etc.

Position UI components on the regular student panel using set bounds:

- Set the position and size of each label, text field, combo box, and button on the panel.

Note: The given code only initializes and sets the layout for the regular student panel. Presumably, there will be similar code for the dropout panel and then adding these panels to the main frame.

End class

Start

Initialize a JPanel named dropoutPanel1 with no layout manager (absolute positioning)

Create JLabel components:

- Create title label named "DROPOUT FORM"
- Create labels for student details like "Student Name", "Enrollment ID", "Tuition Fee", etc.

Create JTextField components:

- Fields for entering student name, enrollment ID, tuition fee, etc.

Create JComboBox components for dates:

- For date of enrollment, date of birth, and date of dropout.
- Each date consists of three combo boxes (for day, month, and year)

Create JButton components:

- Create buttons for actions like "Create Dropout Student", "Remove Student", "Bills check", "Display", and "Clear".

Position JLabel components on dropoutPanel1 using absolute coordinates.

Position JTextField and JComboBox components on dropoutPanel1 using absolute coordinates.

Position JButton components on dropoutPanel1 using absolute coordinates.

Set DocumentFilter to JTextField components to ensure that only integers are accepted.

Set placeholders for certain JTextFields for better user guidance.

Set fonts and styles for titles.

Set background colors for panels.

Add components to regularPanel (not shown in the provided code, but inferred):

- Add all the components like JLabel, JTextField, JComboBox, and JButton related to the regular student form.

Add components to dropoutPanel1:

- Add all the components like JLabel, JTextField, JComboBox, and JButton related to the dropout student form.

Attach action listeners to all buttons:

- For both regular and dropout panels, listen for button clicks to perform respective actions.

Add both panels (regular and dropout) to the main frame.

Set main frame properties:

- Position the main frame to the center of the screen.
- Make the main frame visible.

End

Function createRegularStudent:

TRY

Gather student name, date of birth, course details, and enrollment details from form fields

Construct full date of birth and date of enrollment from selected year, month, and day

Check if course name is empty, enrollment ID is non-positive, or days present is non-positive

IF any checks fail:

Alert "Failed to create object! Ensure course name, enrollment ID, and days present are valid."

Exit function

CATCH any error:

Alert "Failed to create object!"

```
Function DropoutgetValue:
   TRY
      Gather student name, date of birth, course details, enrollment details, and dropout
date from form fields for Dropout students
      Print retrieved student details for debugging purposes
   CATCH any error:
      Alert "Error!"

Function addDropout:
   TRY
      Extract values from Dropout student form using DropoutgetValue function
      Create a new Dropout student object using extracted values
      Add the new Dropout student to a list of students
      Alert "Dropout student created successfully!"
   CATCH any error:
      Alert "Error!"

Function addRegularStudentButton:
   Generate properties for regular student using createRegularStudent function
   Create a new Regular student object using generated properties
   Add the new Regular student to a list of students
   Alert "Regular student created successfully!"

Function clearRegularStudentFields:
   Clear all regular student form input fields
   Reset all date-related combo boxes to their default values

Function clearDropoutStudentFields:
   Clear all dropout student form input fields
   Reset all date-related combo boxes to their default values
WHEN action is performed DO
```

```
IF the action is from 'addaRegularStudentButton' THEN
    CALL the function to add a regular student

ELSE IF the action is from 'calculatepresentPercentageofRegularStudentButton' THEN
    PROMPT for the Enrollment ID
    SET flag to indicate student not found
    FOR each student in student list DO
        IF student is a Regular student THEN
            IF provided Enrollment ID matches student's Enrollment ID THEN
                SET flag to indicate student found
                PROMPT for number of days present
                IF days present is valid THEN
                    CALCULATE present percentage and DISPLAY the grade
                    EXIT the loop
            END IF
        END IF
    END FOR
    IF student not found THEN
        DISPLAY error message

ELSE IF the action is from 'grantCertificateofRegularStudentButton' THEN
    PROMPT for the Enrollment ID
    FOR each student in student list DO
        IF student is a Regular student THEN
            IF provided Enrollment ID matches student's Enrollment ID THEN
                PROMPT for certificate details
                GRANT certificate to the student
                DISPLAY success message and EXIT the loop
            ELSE
                DISPLAY error message
            END IF
```

```
        END IF
    END FOR

ELSE IF the action is from 'displayButton' THEN
    TRY
        PROMPT for the Enrollment ID
        SET flag to indicate student not found
        FOR each student in student list DO
            IF student is a Regular student AND matches the provided ID THEN
                DISPLAY student details
                SET flag to indicate student found and EXIT the loop
            END IF
        END FOR
        IF student not found THEN
            DISPLAY error message
    CATCH any input errors and DISPLAY an error message

ELSE IF the action is from 'clearButton' THEN
    CLEAR Regular Student input fields
    DISPLAY success message

ELSE IF the action is from 'addaDropoutStudentButton' THEN
    CALL the function to add a dropout student

ELSE IF the action is from 'paythebillsofDropoutStudentButton' THEN
    PROMPT for the Enrollment ID
    SET flag to indicate ID did not match
    FOR each student in student list DO
        IF student is a Dropout student AND matches the provided ID THEN
            CALCULATE bills payable for the student
            DISPLAY success message
```

```
            SET flag to indicate ID match found and EXIT the loop
        END IF
    END FOR
    IF ID did not match THEN
        DISPLAY error message

ELSE IF the action is from 'removeDropoutStudentButton' THEN
    PROMPT for the Enrollment ID
    SET flags to indicate student was not removed and bills are not cleared
    FOR each student in student list DO
        IF student is a Dropout student AND matches the provided ID THEN
            IF student has cleared the bills THEN
                REMOVE the student
                TRY to play a removal sound and DISPLAY success message based on
sound playback
            ELSE
                DISPLAY message that bills are not cleared
            END IF
            EXIT the loop
        END IF
    END FOR
    IF student was not removed THEN
        DISPLAY error message

ELSE IF the action is from 'displayButton1' THEN
    TRY
        PROMPT for Dropout Student Enrollment ID
        SET flag to indicate student not found
        FOR each student in student list DO
            IF student is a Dropout student AND matches the provided ID THEN
                DISPLAY student details
```

```
            SET flag to indicate student found and EXIT the loop
          END IF
        END FOR
        IF student not found THEN
          DISPLAY error message
      CATCH any input errors and DISPLAY an error message

  ELSE IF the action is from 'clearButton1' THEN
      CLEAR Dropout Student input fields
      DISPLAY success message
  END IF
END WHEN
```

4-Method description/Button description

Method name	Description
addaRegularButton	This is a button which takes value from users to create a regular student object and is added to Student Arraylist. Its requires enrollmentID ,Dayspresent, with non-zero value & Coursename that cant be empty to create regular object.
addaDropoutButton	This is a button which takes value from users to create a dropout student object and is added to Student Arraylist.To create dropout object :-

	enrollmentID,Courseduration,numofremaininigmodules, numofmonthsattended must be integer.
calculatepresentPercentage Button	This button is used to calculate present percentage of a regular student based on days presents in given months to check grade from A(Excellent) to E(poor) and to process for checking grantCertificate of student.
grantCertificateofRegularStudentButton	This button is used to grant certificates based on grade after calulating present percentage of regular student, If a student receive grade A in persentPercentage than output :- certificate granted with enrollmentId if there persentpercentage is below 80% than it show doesn't met requirement for certificates.
paythebillsofDropoutStudent	This button is used to calculate tuition fee to check wheather the student has paid all bills or not of dropout object. If the tuition fee is zero than bills is cleared but is there is value than its shows bills need to be paid. If Users input decimal values in tuiton fee & days present the double will be converted to int .
removeDropoutStudentButton	This button show wheather before removing student wheather there bills Is cleared or not if not the student wont be removed unless there bills is cleared.
displaButton	Its display the input JTextField values of users while creating regular object .
displayButton1	Its display the input JTextField values of users while creating Dropout object .
clearButton	Its used to clear JTextFields in Regular panel Replacing current value with default value define by users
clearButton1	Its used to clear JTextFields in Dropout panel. Replacing current value with default value define by users.

5 Testing

5.1 Test-1

Test No	**1**
Objective	To check the program can run using command prompt
Action	The command prompt is opened in the path containing java files. 'Javac Student_GUI java' command is used to compile the program, Java Student_GUI.java command is used to run the GUI
Expected result	The program will compile & run
Actual result	The program complied and GUI was executed
Conclusion	**The test is successful.**

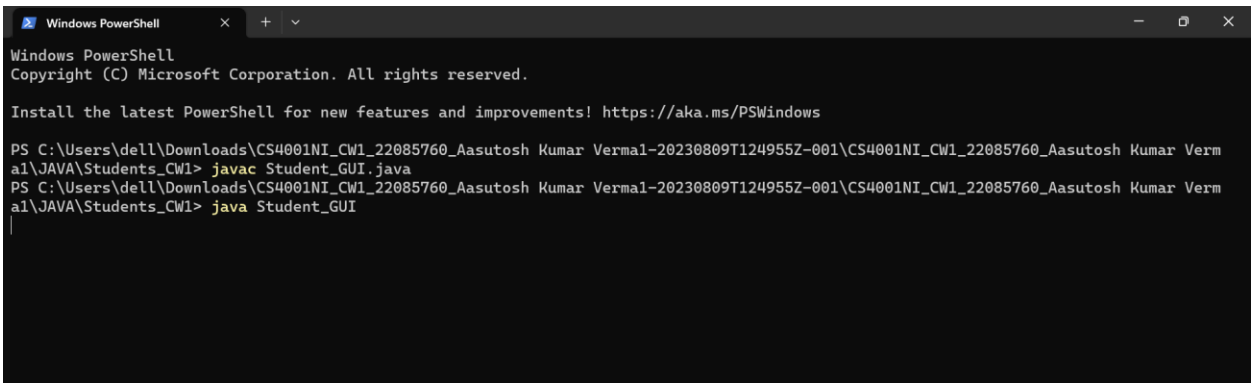

Figure 9: computer command prompt

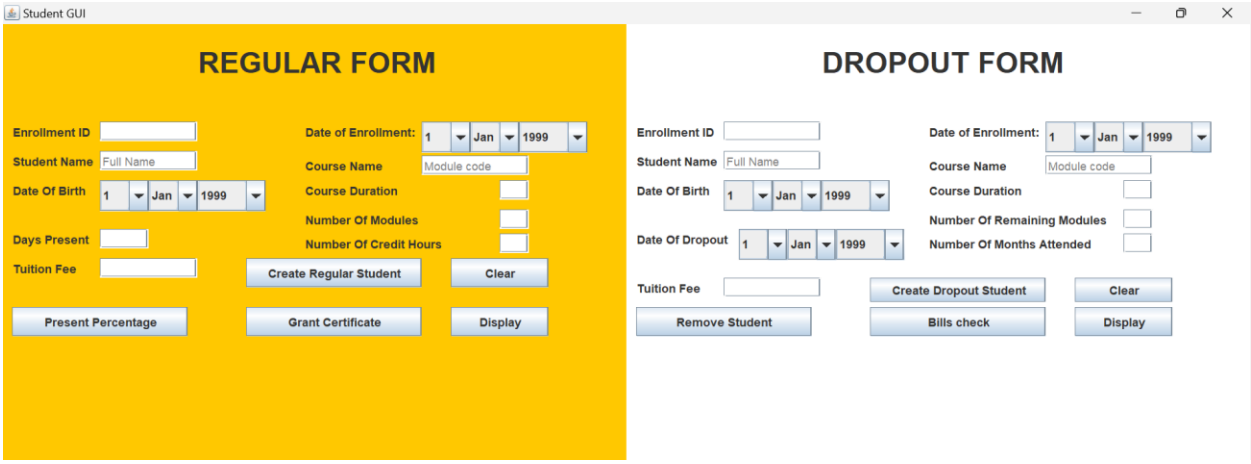

Figure 10: GUI

5.2 Test-2

5.2.1

Test No	**2.1**
Objective	To check the CreateRegularStudent Button is working or not.
Action	GUI is opened. Valid inputs are given to create regular object. Add button is clicked.
Expected result	The regular student will be created & will show appropriate message.
Actual result	The regular student created & showed appropriate message.
Conclusion	**This test is successful.**

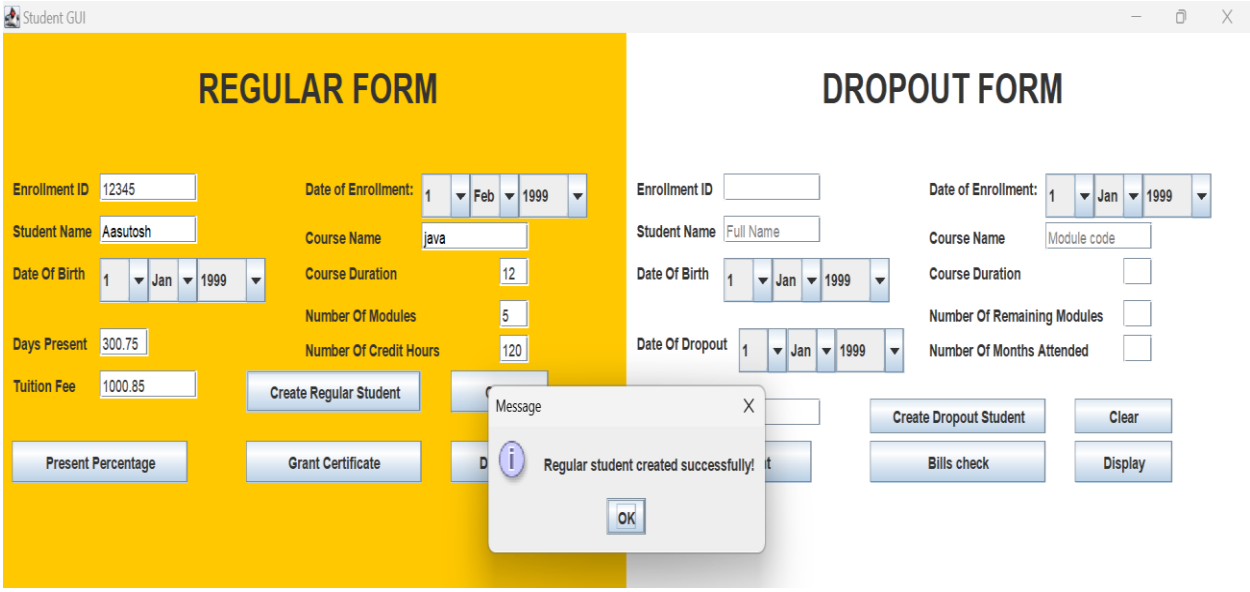

Figure 11: Regular object.

5.2.2

Test No	**2.2**
Objective	To Check Present Percentage Button working or not.
Action	After regular object is created. Click presentPercentage button. Appropriate message will show that takes input :- Enter enrollmentID . Enter number of days present. Click ok
Expected result	PresentPercentage will calculate and will display in message after entering valid enrollment ID & days present.The result will show in appropriate message

Actual result	PresentPercentage calculated and displayed in message after entered valid enrollment ID & days present.The result shown in appropriate message
Conclusion	**This Test is successful.**

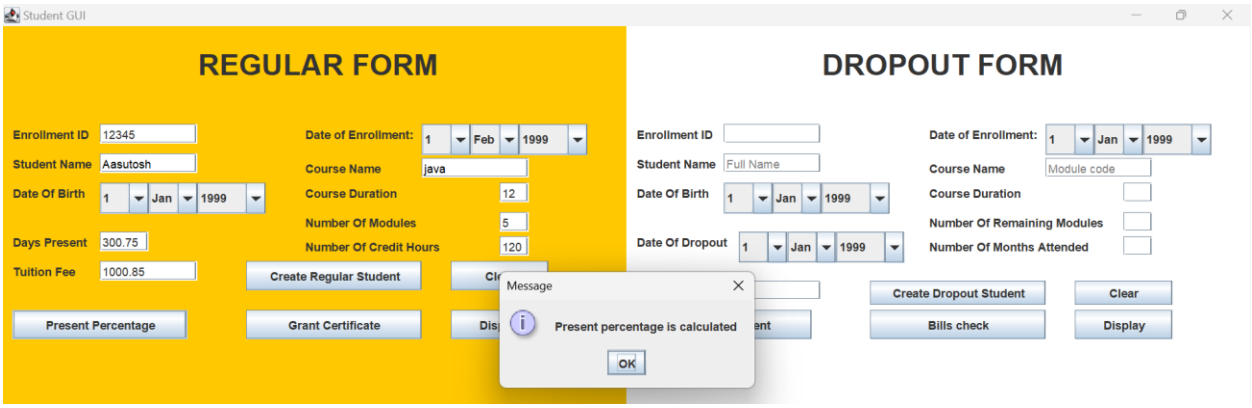

Figure 12: Present Percentage

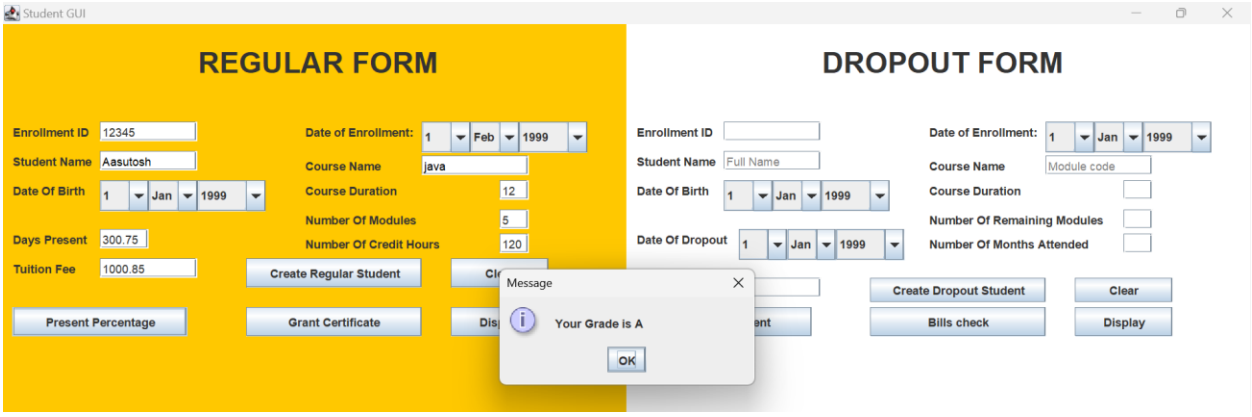

Figure 13: Grade Displayed

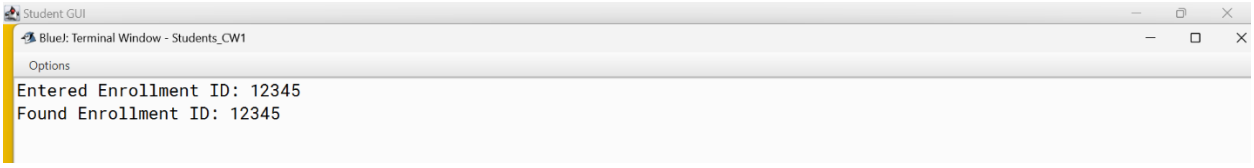

Figure 14: Java console

5.2.3

Test No	**2.3**
Objective	To check Grant Certificate Button is working or not.
Action	Afte calculating presentPercentage. Click grant certificates button. Enter enrollment Id. Enter Days Present. Click ok.
Expected result	A messge will display that takes input before showing result:- enrollment ID & days present. Click ok After enter all valid details A result message will display with student enrollment ID & name .
Actual result	A messge displayed that takes input before showing result:- enrollment ID & days present. Click ok After entered all valid details A result message displayed with student enrollment ID & name .
Conclusion	**This test is successful.**

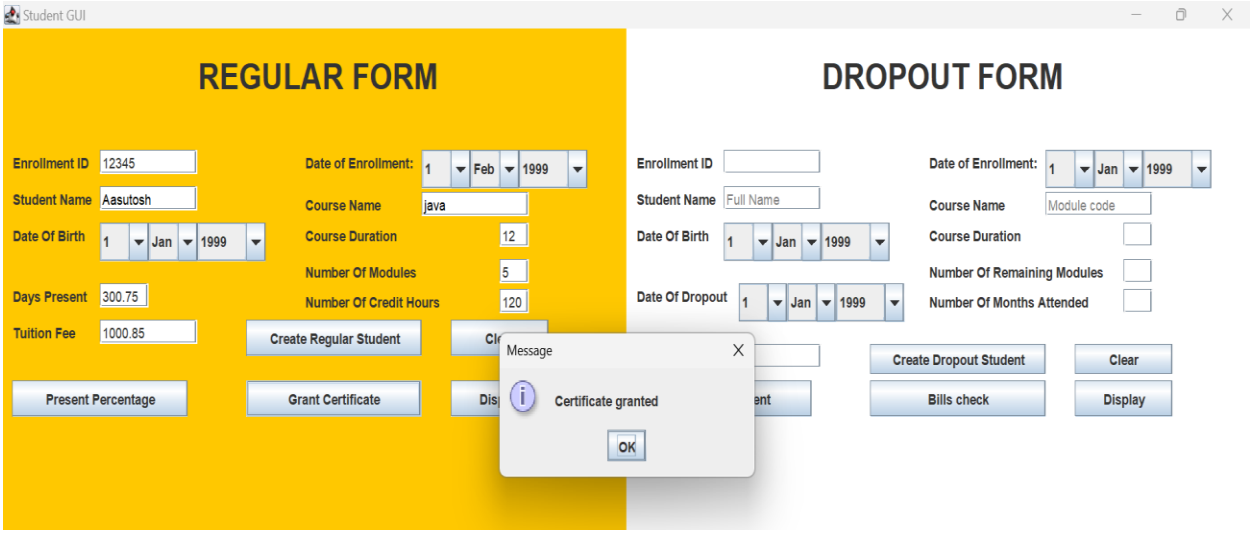

Figure 15: Certificates granted

```
Aasutosh has graduated from java with enrollment ID 12345 and enrollment date 1Feb1999. The scholarship has been granted.
Verfied by the school Administrative.
```

Figure 16: Java Console

5.2.4

Test No	**2.4**
Objective	To check Display Button working or not.
Action	Create regular object Enter all valid inputs. Calculate presentpercentage. Calculate grant certificates Click display button.
Expected result	The result will show in bluej console fields with appropriate message of inputs fields.

Actual result	The result will show in bluej console fields with appropriate message of inputs fields.
Conclusion	**This test is successful.**

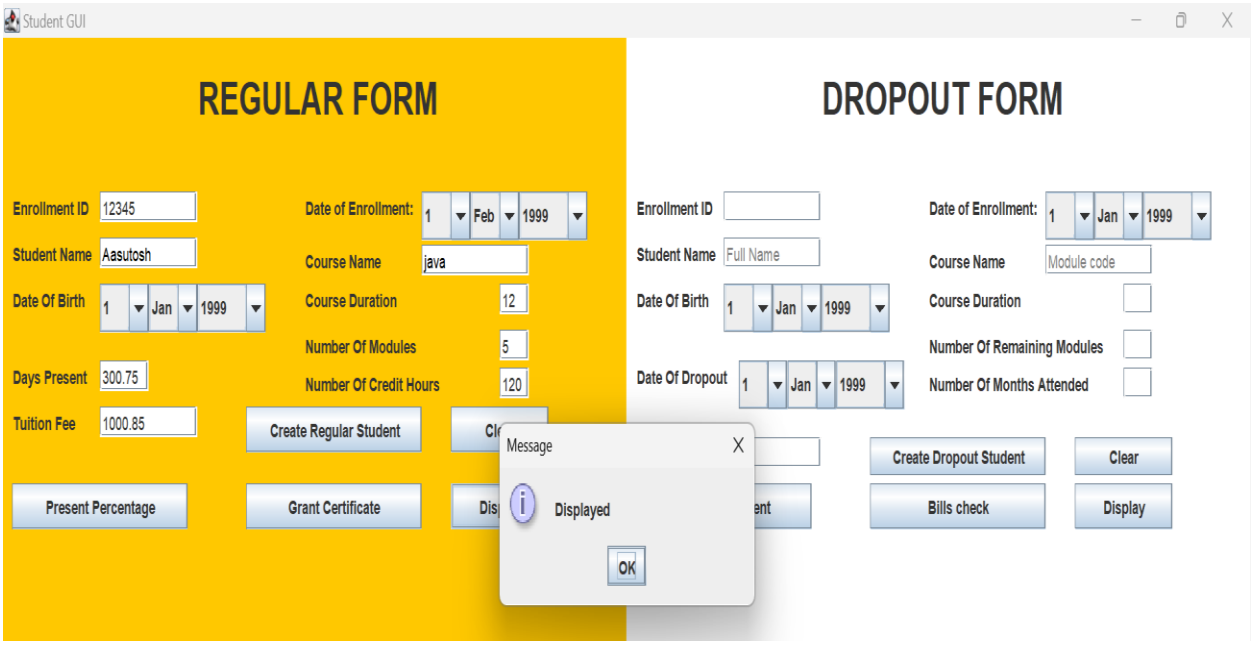

Figure 17: Displyed of regular

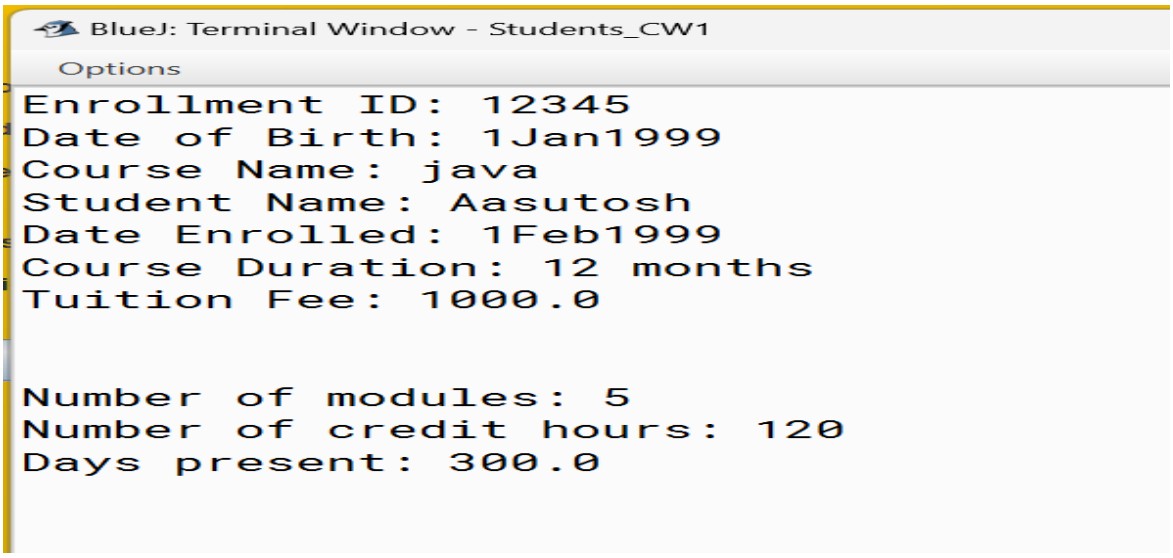

Figure 18:Displyed of regular in java console

5.2.5

Test No	**2.5**
Objective	To Check Clear Button Working or not.
Action	Input valid details while creating object. Click clear button.
Expected result	The input details will remove and default value will set in place of current value.
Actual result	The input details removed and default value set in place of current value.
Conclusion	**This test is successful.**

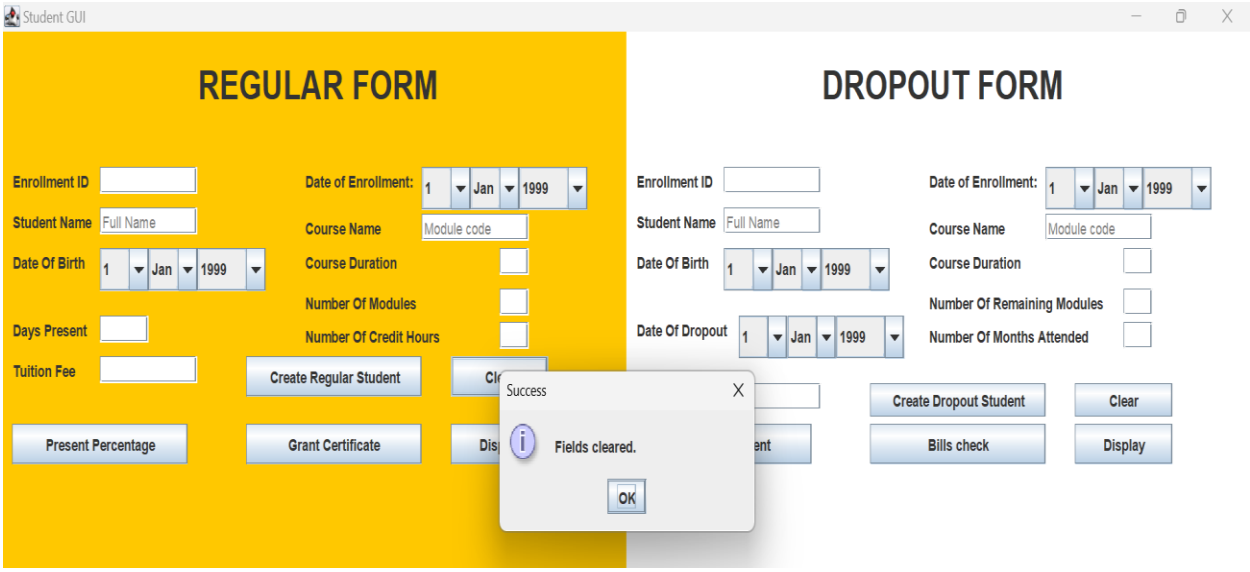

Figure 19: Clear button

5.2.6

Test No	**2.6**
Objective	To check Create dropout Student Button is working or not.
Action	Open GUI. Enter valid inputs required to create dropout object. Click create dropout button.
Expected result	The value will store in student arraylist.and a appropriate message will display.
Actual result	The value stored in student arraylist.and a appropriate message displayed.
Conclusion	**This test is successful.**

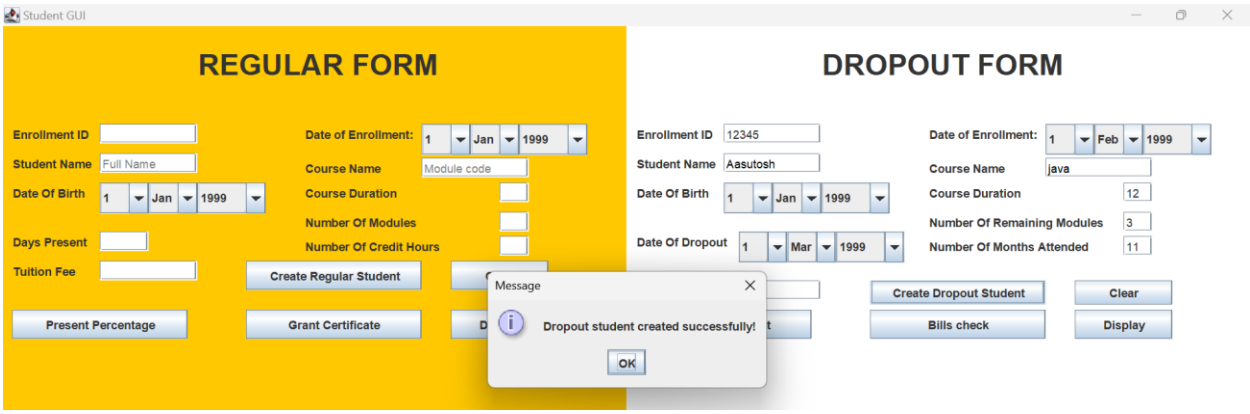

Figure 20: Dropout student form

5.2.7

Test No	**2.7**
Objective	To Check Bills Check Button is working or not.
Action	After dropout student object is created. Click bills check button. Enter enrollment ID Click ok
Expected result	An appropriate message will display with tuition fee. Paid or not .
Actual result	An appropriate message displayed with tuition fee. Paid or not .
Conclusion	**This test is successful.**

```
All bills cleared by the student
```

Figure 21: Java console

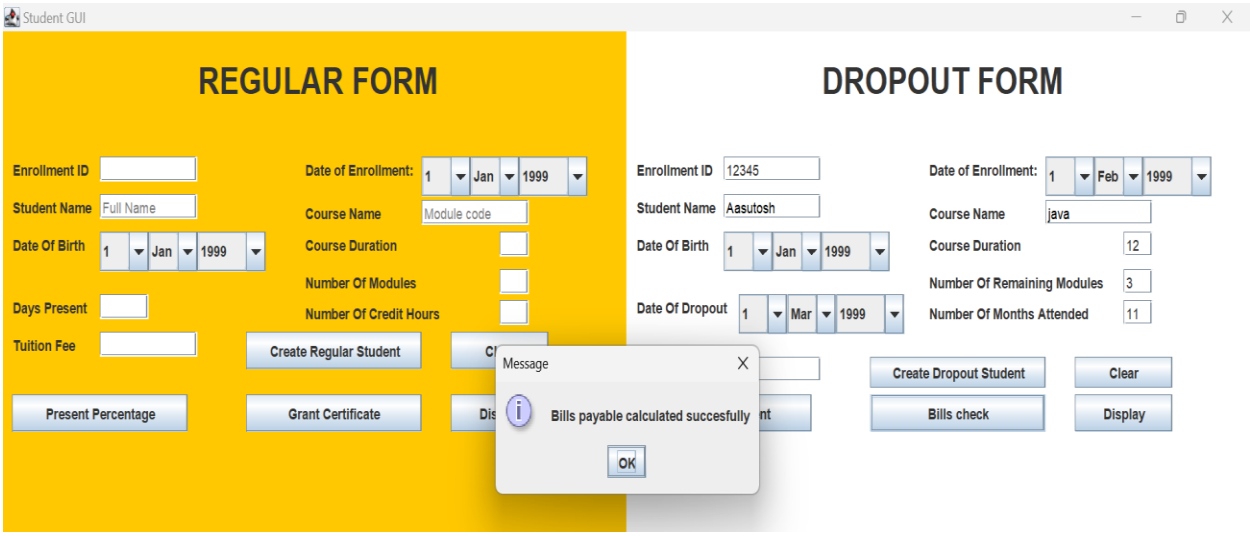

Figure 22: Bills Message displayed

5.2.8

Test No	**2.8**
Objective	To Check Remove Button is working or not.
Action	After Bills Check. Click remove student Button. Enter enrollment ID. Click ok.
Expected result	A message will display if bills is clear or not to be remove.
Actual result	A message displayed if bills is cleared or not to be removed.
Conclusion	**This test is successful.**

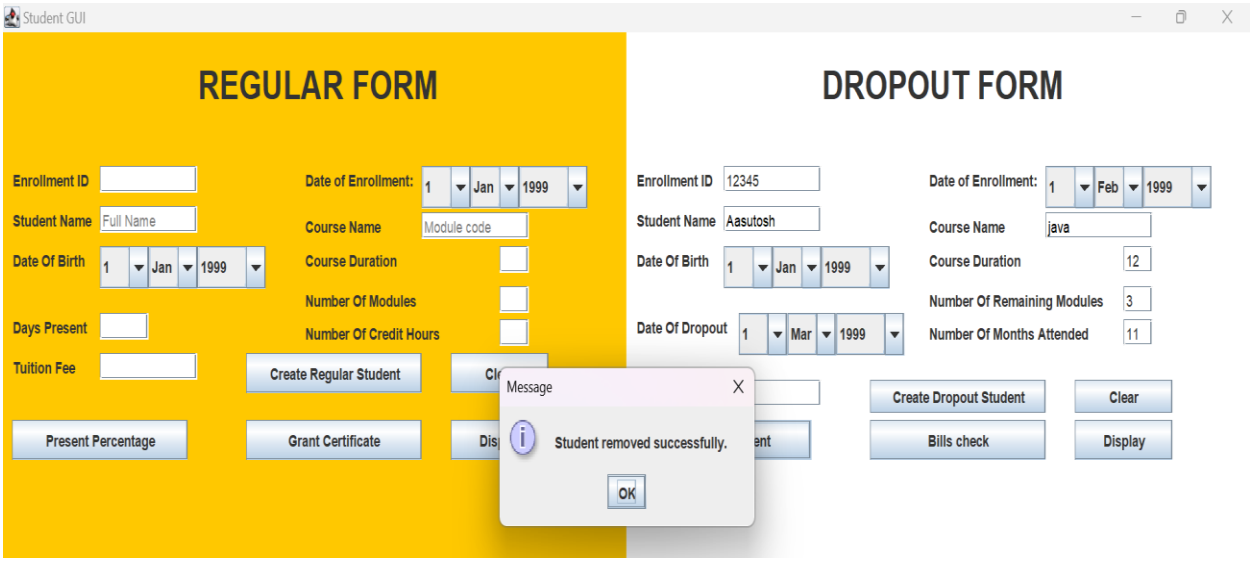

Figure 23: Dropout Student removed.

5.2.9

Test No	**2.9**
Objective	To Check Display Button is working or not.

Action	Enter valid inputs while creating dropout objects. Check bills left or paid. Click display button. Enter enrollment ID
Expected result	The inputs fields details will display in blueJ console.
Actual result	The inputs fields details displayed in blueJ console.
Conclusion	**This test is successful.**

```
Enrollment ID: 12345
Date of Birth: 1Jan1999
Course Name: java
Student Name: Aasutosh
Date Enrolled: 1Feb1999
Course Duration: 12 months
Tuition Fee: 0.0


Number of remaining modules: 3
Number of months attended: 11
Date of dropout: 1Mar1999
Remaining amount: Rs 0.0
```

Figure 24: Displayed details of Dropout student.

5.2.10

Test No	**2.10**
Objective	To Check clear Button is working or not.
Action	Enter valid inputs for creating dropout student objects. Click clear button.
Expected result	The currents inputs will replace with defaults value set by users.

Actual result	The currents inputs replaced with defaults value set by users.
Conclusion	**This test is successful.**

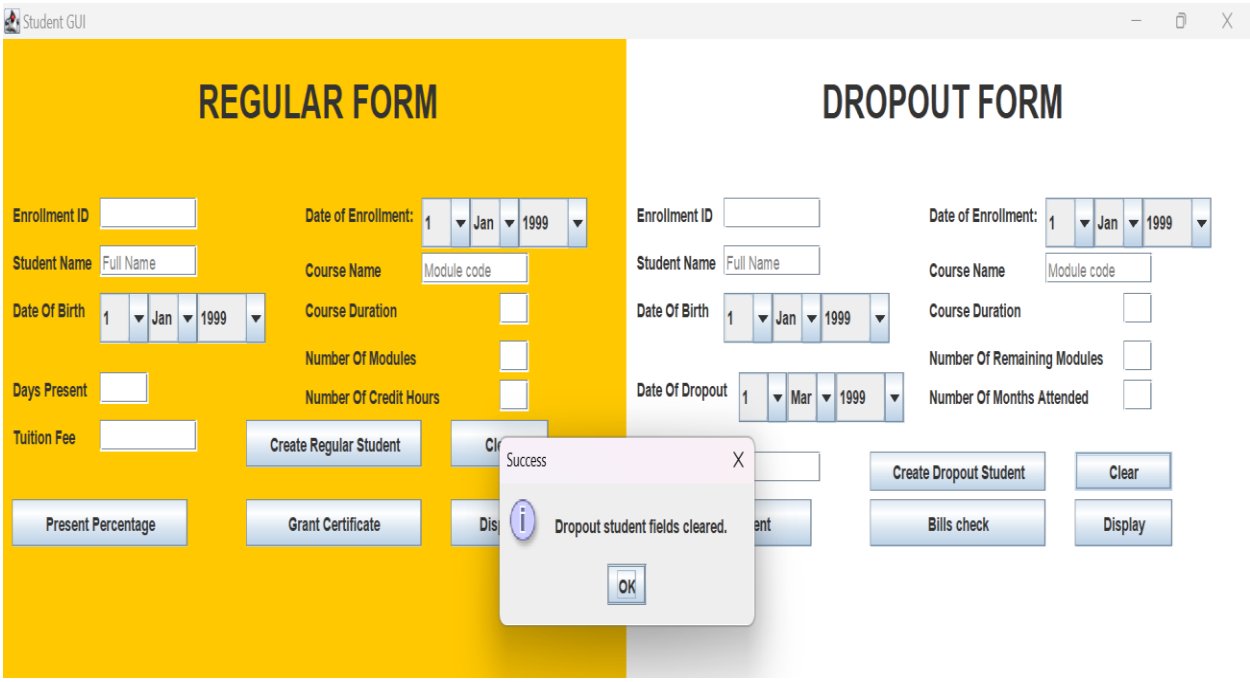

Figure 25: Dropout student input Fields cleared.

6 Test-3

6.1

6.1.1

Test No	**6.1.1**

Objective	To Check if number format exception is working correctly
Action	Open GUI. Click in Enrollment ID TextFields Enter non-integer value in enrollment Id For eg ;-A, ;, p, [, etc
Expected result	When users inputs non integer values in TextFields a error message will pop up.
Actual result	When users inputs non integer values in TextFields a error message shown.
Conclusion	**This test is successful.**

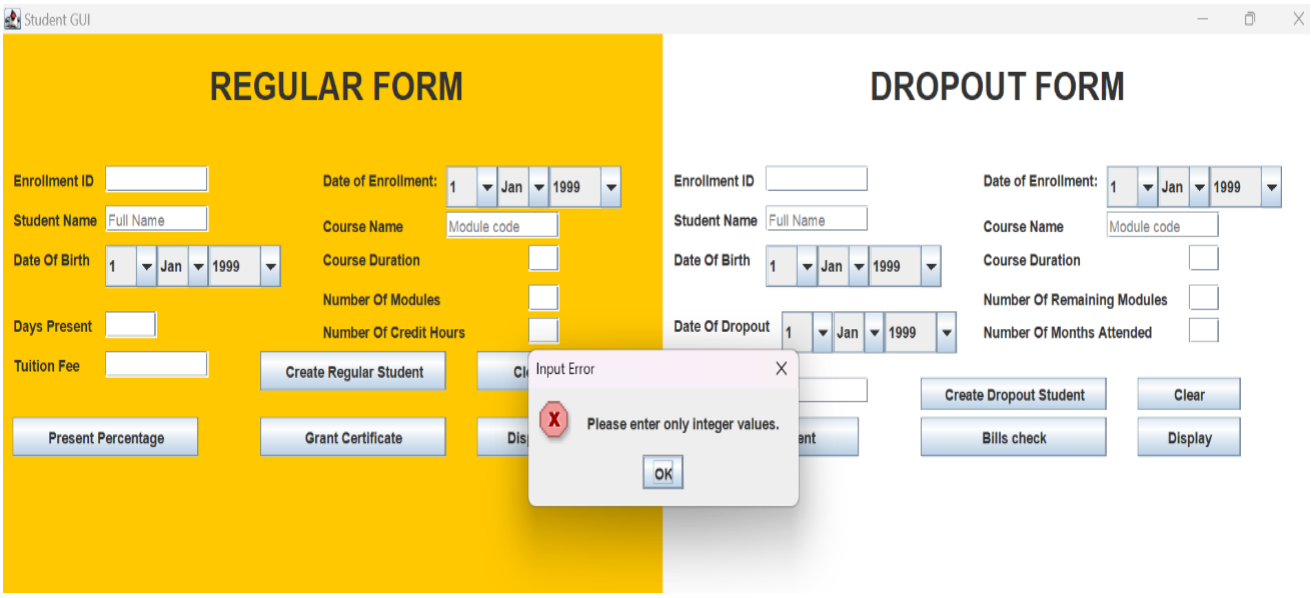

Figure 26: only int value.

6.1.2

Test No	**6.1.2**
Objective	To Check Empty field is restricted
Action	Open GUI . No inputs in textfields by users Click Create regular button.
Expected result	A error message will display.
Actual result	A error message displayed.
Conclusion	**This test is successful.**

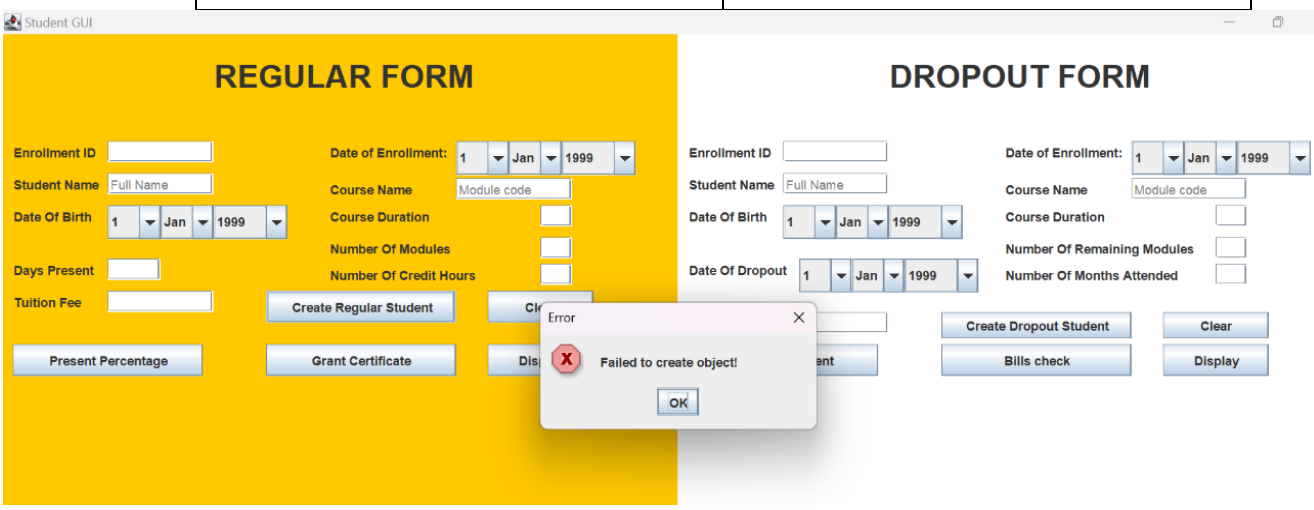

Figure 27: Empty Fields Error

6.2
6.2.1

Test No	**6.2.1**

Objective	To Check if Bills not paid student will remove or not .
Action	Open GUI Create dropout students objects. Enter Tuition fee value not equal to zero. Click remove buttons. Enter enrollment ID
Expected result	A warning message will show with appropriate message including enrollment id.
Actual result	A warning message shown with appropriated message included enrollment id.
Conclusion	**This test is successful.**

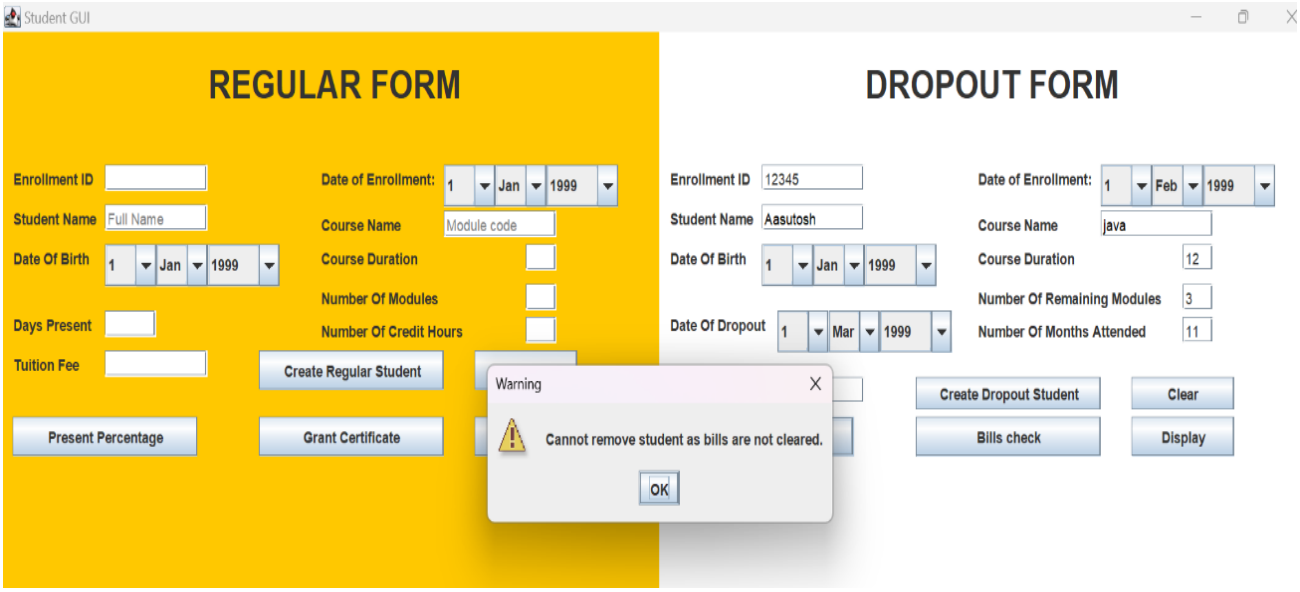

Figure 28: warning message .

7 Error Detection & Correction

7.1 Synatx Error

These are the type of error that occurs when there is mistake in the code itself. In this code, JfextField is not recognized by Java. This is fixed to JTextField to use JTextFields in java.

Before.

```
Student_GUI - Students_CW1
lass  Edit  Tools  Options
dent_GUI ×
mpile  Undo  Cut  Copy  Paste  Find...  Close

//Text Field for Regular
/* A class in the Java Swing library that represents a single-line text input field. */
private JTextField studentNameText;
private JTextField enrollmentIDText;
private JTextField courseNameText;
private JTextField courseDurationText;
private JTextField tuitionFeeText;
private JfextField numOfModulesText;
private JTextField numOfCreditHoursText;
private JTextField daysPresentText;
private int numOfMonthsAttended, numOfRemainingModules;
```

Figure 29: Syntax Error

After Correction

```
Class  Edit  Tools  Options
udent_GUI ×
Compile  Undo  Cut  Copy  Paste  Find...  Close

//Text Field for Regular
/* A class in the Java Swing library that represents a single-line text input field. */
private JTextField studentNameText;
private JTextField enrollmentIDText;
private JTextField courseNameText;
private JTextField courseDurationText;
private JTextField tuitionFeeText;
private JTextField numOfModulesText;
private JTextField numOfCreditHoursText;
private JTextField daysPresentText;
private int numOfMonthsAttended, numOfRemainingModules;
```

Figure 30: Syntax Error correction

7.2 Semantic error

These are the error that occurs when there is incorrect initialization of data with its datatype or vice versa. In this program, bills cleared is initialized as a Boolean datatype To check whether students bills are cleared or not but the value is initialized as int. To fix this, bills cleared is initialized as Boolean in given condition.

Before

```
}

        boolean studentRemoved = false;  // To check if a studer
        int billsCleared = false;    // To check if student's b:

        for(Student s:arraylist)
        {
            if (s instanceof Dropout) {
```

Figure 31: Semantic error

After Correction

```
        boolean studentRemoved = false;  // To check
        boolean billsCleared = false;    // To check

        for(Student s:arraylist)
        {
            if (s instanceof Dropout) {
                Dropout drop = (Dropout) s;
                if (sdid==drop.getEnrollmentID())
```

Figure 32: Semantic error correction

7.3 Logical Error

These are the type of error that compiler cannot detect when a program executes. However, after execution, there is undesired output due to the commands given by the

user. In this code, dateOfBirth = dateOfBirth2 + dateOfBirth1 + dateOfBirthyear; was arranged in wrong parsing in a row due to which date would be displayed incorrectly. This is fixed by placing correct combo box in right order.

Before

```
Undo  Cut  Copy  Paste  Find...  Close
{
    try{
        // Retrieve data from the form components for Dropout students
        studentName = studentNameText1.getText();
        // For debugging purposes: Print the retrieved data
        System.out.println(studentName);
        String dateOfBirthyear = String.valueOf(birthDateComboBox0.getSelectedItem());
        String dateOfBirth1=String.valueOf(birthDateComboBox01.getSelectedItem());
        String dateOfBirth2=String.valueOf(birthDateComboBox02.getSelectedItem());
        dateOfBirth=dateOfBirth2 + dateOfBirth1 + dateOfBirthyear;
        System.out.println(dateOfBirth);
        courseName = courseNameText1.getText();
        System.out.println(courseName);
        courseDuration = Integer.parseInt(courseDurationText1.getText());
        System.out.println(courseDuration );
        enrollmentID = Integer.parseInt(enrollmentIDText1.getText());
        System.out.println(enrollmentID);
        String dateOfEnrollmenty = String.valueOf(enrollmentDateComboBox0.getSelectedItem())
```

Figure 33: Logical error

After Correction

```
{
    try{
        // Retrieve data from the form components for Dropout students
        studentName = studentNameText1.getText();
        // For debugging purposes: Print the retrieved data
        System.out.println(studentName);
        String dateOfBirthyear = String.valueOf(birthDateComboBox0.getSelectedItem());
        String dateOfBirth1=String.valueOf(birthDateComboBox01.getSelectedItem());
        String dateOfBirth2=String.valueOf(birthDateComboBox02.getSelectedItem());
        dateOfBirth=dateOfBirthyear+dateOfBirth1+dateOfBirth2;
        System.out.println(dateOfBirth);
        courseName = courseNameText1.getText();
        System.out.println(courseName);
        courseDuration = Integer.parseInt(courseDurationText1.getText());
        System.out.println(courseDuration );
        enrollmentID = Integer.parseInt(enrollmentIDText1.getText());
        System.out.println(enrollmentID);
```

Figure 34: Semantic error correction.

8 Conclusion

"During this course, I discovered that Java is an immensely powerful programming language. Its applications and purposes extend from simple arithmetic tasks to full-fledged software for daily record-keeping and beyond. Even those new to programming can quickly create professional applications, thanks to Java's ease of learning and user-friendly syntax. Its adaptability, extensive community support, and wide-ranging capabilities make it an asset in the field of software development. Whether for a beginner aiming to gain a clear understanding of fundamental concepts or an experienced developer looking to build robust and scalable systems, Java offers the tools and flexibility to achieve those goals, solidifying its place as a cornerstone in modern programming."

For this coursework, I constructed a graphical user interface (GUI). This achievement was made possible by creating a specific class dedicated to the GUI and its functions. Building upon our prior coursework using pre-existing classes, this project added to the sense of continuous progression. Honestly, this wasn't the most challenging task I've ever faced, thanks to a team of highly skilled lecturers who are exceptional educators. If obstacles arise, there's a community of passionate peers always ready to work collaboratively to find solutions. This supportive environment fosters a relentless determination to persevere and complete the task at hand.
I got few IOT based projects Idea while doing it which can be implemented in real based scenarios. Hope After this coursework I would learn backend of java programing to take my research further.

Overall, this coursework has deepened my understanding of Java and broadened my awareness of its extensive applicability. I've come to appreciate that specializing in Java can be a promising career path. While developing the classes and objects, I discerned that Java is a more straightforward and accommodating programming language compared to traditional C or C++. This distinction doesn't make Java superior in all respects, but it certainly adds a precious dimension to its appeal, particularly for those looking to master modern programming techniques.

9 Bibliography

https://www.pluralsight.com/

https://www.w3schools.com/

https://www.java.com/en/

https://www.techopedia.com/definition/29530/bluej

10 Appendix

Student GUI code

```
import javax.swing.*;           // Provides classes for creating graphical user interfaces (GUIs) in Java.
import java.awt.event.ActionEvent;      // Represents an action event in a GUI, like a button click.
import java.awt.event.ActionListener;     // Interface for classes that respond to action events.
import java.text.ParseException;          // Represents errors that occur while parsing text to a different
format, such as dates.
import java.awt.Dimension;              // Represents the width and height dimensions of components in a
GUI.
import javax.swing.JButton;             // Represents a GUI button that users can click on.
import java.awt.Font;                   // Represents fonts that can be used in GUI components.
import java.awt.*;                      // Core AWT (Abstract Window Toolkit) package, provides basic GUI
components and layouts.
import java.awt.event.*;                // Provides interfaces and classes for handling different types of GUI
events.
import java.util.*;                     // Provides various utility classes and interfaces (e.g., lists, sets, maps).
import javax.swing.border.*;            // Provides borders that can be applied to GUI components.
import javax.swing.JLabel;              // Represents a text label in a GUI, which can display text or icons.
```

```
import javax.swing.text.*;              // Provides classes for advanced text display and manipulation in
GUIs.
import java.awt.event.FocusEvent;        // Represents an event where a GUI component gains or loses
focus.
import java.awt.event.FocusListener;     // Interface for classes that respond to focus events.
import java.util.ArrayList;              // A resizable array, part of the Java Collections Framework.
import java.awt.Toolkit;                 //for beep sound



//importing required packages
/**
 * Write a description of class Student_GUI here.
 *
 * @author (Aasutosh verma)
 * @version (1.0.0)
 */

public class Student_GUI implements ActionListener  //main class implements action listener
{
   // instance variables - replace the example below with your own
   private JFrame mainFrame;//Top level Container(Main window or Frame).
   //JPanel-A class in the Java Swing library that represents a container for holding and organizing other
GUI components.
   private JPanel regularPanel, dropoutPanel1;
   //JLabel- It is used to display a single line of non-editable text or an image on a Java graphical user
interface (GUI).
   private JLabel studentNameLabel,enrollmentIDLabel, dateOfEnrollmentLabel, courseNameLabel,
courseDurationLabel, dateOfBirthLabel, tuitionFeeLabel, numOfModulesLabel, numOfCreditHoursLabel,
daysPresentLabel, studentNameLabel1,enrollmentIDLabel1, dateOfEnrollmentLabel1,
courseNameLabel1, courseDurationLabel1, dateOfBirthLabel1, tuitionFeeLabel1,
numOfRemainingModulesLabel ,numOfMonthsAttendedLabel, dateOfDropoutLabel;

   //Text Field for Regular
   /* A class in the Java Swing library that represents a single-line text input field. */
   private JTextField studentNameText;
   private JTextField enrollmentIDText;
   private JTextField courseNameText;
   private JTextField courseDurationText;
   private JTextField tuitionFeeText;
   private JTextField numOfModulesText;
   private JTextField numOfCreditHoursText;
   private JTextField daysPresentText;
   private int numOfMonthsAttended, numOfRemainingModules;


   //Text Field for Dropout
   private JTextField studentNameText1;
```

```
    private JTextField enrollmentIDText1;
    private JTextField courseNameText1;
    private JTextField courseDurationText1;
    private JTextField tuitionFeeText1;
    private JTextField numOfRemainingModulesText;
    private JTextField numOfMonthsAttendedText;


    //JComboBox-A class in the Java Swing library that represents a drop-down list or combo box
component in a graphical user interface (GUI).
    private JComboBox<String> birthDateComboBox, birthDateComboBox1, birthDateComboBox2,
birthDateComboBox0, birthDateComboBox01, birthDateComboBox02;
    private JComboBox<String> enrollmentDateComboBox, enrollmentDateComboBox1,
enrollmentDateComboBox2, enrollmentDateComboBox0, enrollmentDateComboBox01,
enrollmentDateComboBox02;
    private JComboBox<String> dropoutDateComboBox0, dropoutDateComboBox01,
dropoutDateComboBox02;

    //JButton-It used to create clickable buttons that can trigger actions when pressed.
    private JButton addaRegularStudentButton, calculatepresentPercentageofRegularStudentButton,
grantCertificateofRegularStudentButton, displayButton, clearButton;
    private JButton addaDropoutStudentButton, paythebillsofDropoutStudentButton,
removeDropoutStudentButton, displayButton1, clearButton1;

    private JList<String> regularStudentList;
    private JList<String> dropoutStudentList;
    private double dayspresent;
    private  String studentName, dateOfBirth, courseName, dateOfEnrollment,dateOfDropout;;
    private int courseDuration,enrollmentID,numOfCreditHours,numOfModules;
    private double tuitionFee,daysPresent;

    //This is the declaration of the ArrayList.
    private ArrayList <Student> arraylist = new ArrayList <Student>(); //The ArrayList<Student>() is called a
constructor, which initializes a new ArrayList. The <Student> in the constructor indicates that this
ArrayList will store Student objects.

    public Student_GUI()
    {
        mainFrame = new JFrame("Student GUI");
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(new GridLayout(1, 2));

        /*----------------------------PANEL-1--------------------------------------*/
        regularPanel = new JPanel(null);

        /*----------------------------LABEL---------------------------------------*/
        JLabel titleLabel = new JLabel("REGULAR FORM");
        studentNameLabel = new JLabel("Student Name");
```

```
        enrollmentIDLabel = new JLabel("Enrollment ID");
        daysPresentLabel = new JLabel("Days Present");
        tuitionFeeLabel = new JLabel("Tuition Fee");
        courseNameLabel = new JLabel("Course Name");
        courseDurationLabel = new JLabel("Course Duration");
        numOfModulesLabel = new JLabel("Number Of Modules");
        numOfCreditHoursLabel = new JLabel("Number Of Credit Hours");
        dateOfEnrollmentLabel = new JLabel("Date of Enrollment: ");
        String days[] =
{"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","2
3","24","25","26","27","28","29","30","31"};
        enrollmentDateComboBox = new JComboBox<String>(days);
        String month[] = {"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
        enrollmentDateComboBox1 = new JComboBox<String>(month);
        String year[] = {"1999", "2000", "2001", "2002", "2003", "2004", "2005"};
        enrollmentDateComboBox2 = new JComboBox<String>(year);
        dateOfBirthLabel = new JLabel("Date Of Birth");

        //TEXT FIELD for Regular components
        studentNameText = new JTextField();
        enrollmentIDText = new JTextField();
        daysPresentText = new JTextField();
        tuitionFeeText = new JTextField();
        courseNameText = new JTextField();
        courseDurationText = new JTextField();
        numOfModulesText = new JTextField();
        numOfCreditHoursText = new JTextField();
        //COMBOBOX for Regular components

        enrollmentDateComboBox = new JComboBox<String>(days);
        enrollmentDateComboBox1 = new JComboBox<String>(month);
        enrollmentDateComboBox2 = new JComboBox<String>(year);
        birthDateComboBox = new JComboBox<String>(days);
        birthDateComboBox1 = new JComboBox<String>(month);
        birthDateComboBox2 = new JComboBox<String>(year);

        //Buttons
        addaRegularStudentButton = new JButton("Create Regular Student");
        calculatepresentPercentageofRegularStudentButton = new JButton("Present Percentage");
        grantCertificateofRegularStudentButton = new JButton("Grant Certificate");
        displayButton = new JButton("Display");
        clearButton = new JButton("Clear");

        //Label Set Bounds For Regular components
        titleLabel.setBounds(200, 15, 450, 50);
        studentNameLabel.setBounds(10, 130, 100, 20);
        enrollmentIDLabel.setBounds(10, 100, 100, 20);
        daysPresentLabel.setBounds(10, 210, 100, 20);
```

```
        tuitionFeeLabel.setBounds(10, 240, 100, 20);
        courseNameLabel.setBounds(310, 135, 100, 20);
        courseDurationLabel.setBounds(310, 160, 100, 20);
        numOfModulesLabel.setBounds(310, 190, 160, 20);
        numOfCreditHoursLabel.setBounds(310, 215, 160, 20);
        dateOfEnrollmentLabel.setBounds(310, 100, 118, 20);
        dateOfBirthLabel.setBounds(10, 160, 100, 20);

        //Text Set Bounds for Regular components
        studentNameText.setBounds(100, 130, 100, 20);
        enrollmentIDText.setBounds(100, 100, 100, 20);
        daysPresentText.setBounds(100, 210, 50, 20);
        tuitionFeeText.setBounds(100, 240, 100, 20);
        courseNameText.setBounds(430, 135, 110, 20);
        courseDurationText.setBounds(510, 160, 30, 20);
        numOfModulesText.setBounds(510, 190, 30, 20);
        numOfCreditHoursText.setBounds(510, 215, 30, 20);
        enrollmentDateComboBox.setBounds(430, 100, 50, 32);
        enrollmentDateComboBox1.setBounds(480, 100, 50, 32);
        enrollmentDateComboBox2.setBounds(530, 100, 70, 32);
        birthDateComboBox.setBounds(100, 160, 50, 32);
        birthDateComboBox1.setBounds(150, 160, 50, 32);
        birthDateComboBox2.setBounds(200, 160, 70, 32);
        //SET BOUNDS FOR BUTTONS
        addaRegularStudentButton.setBounds(250, 240, 180, 30);
        calculatepresentPercentageofRegularStudentButton.setBounds(10, 290, 180, 30);
        grantCertificateofRegularStudentButton.setBounds(250, 290,180, 30);
        displayButton.setBounds(460, 290, 100, 30);
        clearButton.setBounds(460, 240, 100, 30);



////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////////
        //PANEL - 2
        dropoutPanel1= new JPanel(null);

        //JLABEL
        JLabel titleLabel1 = new JLabel("DROPOUT FORM");
        studentNameLabel1 = new JLabel("Student Name");
        enrollmentIDLabel1 = new JLabel("Enrollment ID");
        tuitionFeeLabel1 = new JLabel("Tuition Fee");
        courseNameLabel1 = new JLabel("Course Name");
        courseDurationLabel1 = new JLabel("Course Duration");
        numOfRemainingModulesLabel = new JLabel("Number Of Remaining Modules");
        numOfMonthsAttendedLabel = new JLabel("Number Of Months Attended");
        dateOfEnrollmentLabel1 = new JLabel("Date of Enrollment: ");
```

```
        String days1[] =
{"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","2
3","24","25","26","27","28","29","30","31"};
        enrollmentDateComboBox0 = new JComboBox<String>(days1);
        String month1[] = {"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
        enrollmentDateComboBox01 = new JComboBox<String>(month1);
        String year1[] = {"1999", "2000", "2001", "2002", "2003", "2004", "2005"};
        enrollmentDateComboBox02 = new JComboBox<String>(year1);
        dateOfBirthLabel1 = new JLabel("Date Of Birth");
        dateOfDropoutLabel = new JLabel("Date Of Dropout");

        //JTEXT FIELD
        studentNameText1 = new JTextField();
        enrollmentIDText1 = new JTextField();
        tuitionFeeText1 = new JTextField();
        courseNameText1 = new JTextField();
        courseDurationText1 = new JTextField();
        numOfRemainingModulesText = new JTextField();
        numOfMonthsAttendedText = new JTextField();

        //JCOMBOBOX
        enrollmentDateComboBox0 = new JComboBox<String>(days1);
        enrollmentDateComboBox01 = new JComboBox<String>(month1);
        enrollmentDateComboBox02 = new JComboBox<String>(year1);
        birthDateComboBox0 = new JComboBox<String>(days1);
        birthDateComboBox01 = new JComboBox<String>(month1);
        birthDateComboBox02 = new JComboBox<String>(year1);
        dropoutDateComboBox0 = new JComboBox<String>(days1);
        dropoutDateComboBox01 = new JComboBox<String>(month1);
        dropoutDateComboBox02 = new JComboBox<String>(year1);

        //Buttons
        addaDropoutStudentButton = new JButton("Create Dropout Student");
        removeDropoutStudentButton = new JButton("Remove Student");
        paythebillsofDropoutStudentButton = new JButton("Bills check");
        displayButton1 = new JButton("Display");
        clearButton1 = new JButton("Clear");

        //JLabel Set Bounds For DROPOUT components
        titleLabel1.setBounds(200, 15, 450, 50);
        studentNameLabel1.setBounds(10, 130, 100, 20);
        enrollmentIDLabel1.setBounds(10, 100, 100, 20);
        dateOfDropoutLabel.setBounds(10, 210, 100, 20);
        tuitionFeeLabel1.setBounds(10, 260, 100, 20);
        courseNameLabel1.setBounds(310, 135, 100, 20);
        courseDurationLabel1.setBounds(310, 160, 100, 20);
        numOfRemainingModulesLabel.setBounds(310, 190, 250, 20);
        numOfMonthsAttendedLabel.setBounds(310, 215, 250, 20);
```

```
dateOfEnrollmentLabel1.setBounds(310, 100, 118, 20);
dateOfBirthLabel1.setBounds(10, 160, 100, 20);

//Text Set Bounds for DROPOUT components
studentNameText1.setBounds(100, 130, 100, 20);
enrollmentIDText1.setBounds(100, 100, 100, 20);
dropoutDateComboBox0.setBounds(115, 210, 50, 32);
dropoutDateComboBox01.setBounds(165, 210, 50, 32);
dropoutDateComboBox02.setBounds(215, 210, 70, 32);
tuitionFeeText1.setBounds(100, 260, 100, 20);
courseNameText1.setBounds(430, 135, 110, 20);
courseDurationText1.setBounds(510, 160, 30, 20);
numOfRemainingModulesText.setBounds(510, 190, 30, 20);
numOfMonthsAttendedText.setBounds(510, 215, 30, 20);
enrollmentDateComboBox0.setBounds(430, 100, 50, 32);
enrollmentDateComboBox01.setBounds(480, 100, 50, 32);
enrollmentDateComboBox02.setBounds(530, 100, 70, 32);
birthDateComboBox0.setBounds(100, 160, 50, 32);
birthDateComboBox01.setBounds(150, 160, 50, 32);
birthDateComboBox02.setBounds(200, 160, 70, 32);

//SET BOUNDS FOR BUTTONS
addaDropoutStudentButton.setBounds(250, 260, 180, 25);
removeDropoutStudentButton.setBounds(10, 290, 180, 30);
paythebillsofDropoutStudentButton.setBounds(250, 290,180, 30);
displayButton1.setBounds(460, 290, 100, 30);
clearButton1.setBounds(460, 260, 100, 25);

/*--------------------Set the DocumentFilter  to accept only integers------*/

((AbstractDocument) courseDurationText.getDocument()).setDocumentFilter(new
IntegerOnlyFilter());
((AbstractDocument) enrollmentIDText.getDocument()).setDocumentFilter(new IntegerOnlyFilter());
((AbstractDocument) numOfModulesText.getDocument()).setDocumentFilter(new
IntegerOnlyFilter());
((AbstractDocument) numOfCreditHoursText.getDocument()).setDocumentFilter(new
IntegerOnlyFilter());
((AbstractDocument) courseDurationText1.getDocument()).setDocumentFilter(new
IntegerOnlyFilter());
((AbstractDocument) enrollmentIDText1.getDocument()).setDocumentFilter(new
IntegerOnlyFilter());
((AbstractDocument) numOfRemainingModulesText.getDocument()).setDocumentFilter(new
IntegerOnlyFilter());
((AbstractDocument) numOfMonthsAttendedText.getDocument()).setDocumentFilter(new
IntegerOnlyFilter());

/*------------------Placeholder for JTextField-------------------------------------------------------*/
setPlaceholder(studentNameText, "Full Name");
```

```
setPlaceholder(courseNameText, "Module code");

setPlaceholder(studentNameText1, "Full Name");
setPlaceholder(courseNameText1, "Module code");

/*-----------------SET FONT,SIZE,TYPE FOR COMPONENTS INSIDE REGULAR PANEL--*/
titleLabel.setFont(new Font("Arial", Font.BOLD, 30));
titleLabel1.setFont(new Font("Arial", Font.BOLD, 30));

/*-----------------SET COLORS FOR COMPONENTS INSIDE REGULAR PANEL----------*/
regularPanel.setBackground(Color.ORANGE);
dropoutPanel1.setBackground(Color.WHITE);

/*-------------------------COMPONENTS OF REGULAR PANEL---------------------*/
regularPanel.add(titleLabel);
regularPanel.add(studentNameLabel);
regularPanel.add(dateOfEnrollmentLabel);
regularPanel.add(enrollmentDateComboBox);
regularPanel.add(enrollmentDateComboBox1);
regularPanel.add(enrollmentDateComboBox2);
regularPanel.add(dateOfBirthLabel);
regularPanel.add(birthDateComboBox);
regularPanel.add(birthDateComboBox1);
regularPanel.add(birthDateComboBox2);
regularPanel.add(enrollmentIDLabel);
regularPanel.add(courseNameLabel);
regularPanel.add(courseDurationLabel);
regularPanel.add(tuitionFeeLabel);
regularPanel.add(numOfModulesLabel);
regularPanel.add(numOfCreditHoursLabel);
regularPanel.add(daysPresentLabel);
//TEXT ADD
regularPanel.add(enrollmentIDText);
regularPanel.add(studentNameText);
regularPanel.add(daysPresentText);
regularPanel.add(tuitionFeeText);
regularPanel.add(numOfModulesText);
regularPanel.add(numOfCreditHoursText);
regularPanel.add(courseNameText);
regularPanel.add(courseDurationText);
//BUTTONS ADD
regularPanel.add(addaRegularStudentButton);
regularPanel.add(calculatepresentPercentageofRegularStudentButton);
regularPanel.add(grantCertificateofRegularStudentButton);
regularPanel.add(displayButton);
regularPanel.add(clearButton);

/*-----------------------COMPONENTS OF DROPOUT----------------------------*/
```

```
dropoutPanel1.add(titleLabel1);
dropoutPanel1.add(studentNameLabel1);
dropoutPanel1.add(dateOfEnrollmentLabel1);
dropoutPanel1.add(enrollmentDateComboBox0);
dropoutPanel1.add(enrollmentDateComboBox01);
dropoutPanel1.add(enrollmentDateComboBox02);
dropoutPanel1.add(dateOfBirthLabel1);
dropoutPanel1.add(birthDateComboBox0);
dropoutPanel1.add(birthDateComboBox01);
dropoutPanel1.add(birthDateComboBox02);
dropoutPanel1.add(enrollmentIDLabel1);
dropoutPanel1.add(courseNameLabel1);
dropoutPanel1.add(courseDurationLabel1);
dropoutPanel1.add(tuitionFeeLabel1);
dropoutPanel1.add(numOfRemainingModulesLabel);
dropoutPanel1.add(numOfMonthsAttendedLabel);
dropoutPanel1.add(dateOfDropoutLabel);
dropoutPanel1.add(dropoutDateComboBox0);
dropoutPanel1.add(dropoutDateComboBox01);
dropoutPanel1.add(dropoutDateComboBox02);
//TEXT ADD
dropoutPanel1.add(enrollmentIDText1);
dropoutPanel1.add(studentNameText1);
dropoutPanel1.add(tuitionFeeText1);
dropoutPanel1.add(numOfRemainingModulesText);
dropoutPanel1.add(numOfMonthsAttendedText);
dropoutPanel1.add(courseNameText1);
dropoutPanel1.add(courseDurationText1);
//BUTTONS ADD
dropoutPanel1.add(addaDropoutStudentButton);
dropoutPanel1.add(removeDropoutStudentButton);
dropoutPanel1.add(paythebillsofDropoutStudentButton);
dropoutPanel1.add(displayButton1);
dropoutPanel1.add(clearButton1);

//adding action listener to buttons
//Dropout
addaDropoutStudentButton.addActionListener(this);
removeDropoutStudentButton.addActionListener(this);
paythebillsofDropoutStudentButton.addActionListener(this);
displayButton1.addActionListener(this);
clearButton1.addActionListener(this);
//Regular
addaRegularStudentButton.addActionListener(this);
calculatepresentPercentageofRegularStudentButton.addActionListener(this);
grantCertificateofRegularStudentButton.addActionListener(this);
displayButton.addActionListener(this);
clearButton.addActionListener(this);
```

```
        //mainFrame.pack();
        mainFrame.setLocationRelativeTo(null); // Center the mainFrame in the middle of the screen
        mainFrame.add(regularPanel); // Add the regular and dropout panels to the main frame
        mainFrame.add(dropoutPanel1);
        mainFrame.setVisible(true); // Make the main frame visible to the user
    }

    // Custom DocumentFilter to accept only integer input
    /*
 * IntegerOnlyFilter is a DocumentFilter that ensures only integer values
 * are allowed in the text components where it's applied.
 */
    private static class IntegerOnlyFilter extends DocumentFilter {
        /**
     * Called when new text is being inserted into the document.
     * This method checks each character of the inserted string and removes
     * non-digit characters.
     *
     * @param fb FilterBypass that can be used to mutate Document
     * @param offset the offset into the document to insert the content >= 0
     * @param text the text to insert
     * @param attr the attributes to associate with the inserted content
     */
        @Override
        public void insertString(FilterBypass fb, int offset, String text, AttributeSet attr) throws
BadLocationException {
            StringBuilder builder = new StringBuilder(text);
            for (int i = builder.length() - 1; i >= 0; i--) {
                char ch = builder.charAt(i);
                if (!Character.isDigit(ch)) {
                    builder.deleteCharAt(i);
                }
            }
            super.insertString(fb, offset, builder.toString(), attr);
        }
         /**
     * Called when a part of the document's text is being replaced.
     * This method checks each character of the replacement string and
     * removes non-digit characters. If any non-digit characters are found,
     * it raises an alert notifying the user to input only integer values.
     *
     * @param fb FilterBypass that can be used to mutate Document
     * @param offset the offset into the document to insert the content >= 0
     * @param length length of text to remove
     * @param text the text to insert
     * @param attrs the attributes to associate with the inserted content
     */
```

```
    @Override
    public void replace(FilterBypass fb, int offset, int length, String text, AttributeSet attrs)
    throws BadLocationException {
      if (text == null) {
        return;
      }
      StringBuilder builder = new StringBuilder(text);
      for (int i = builder.length() - 1; i >= 0; i--) {
        char ch = builder.charAt(i);
        if (!Character.isDigit(ch)) {
          builder.deleteCharAt(i);
        }
      }
      super.replace(fb, offset, length, builder.toString(), attrs);
      if (!builder.toString().equals(text)) {
        Toolkit.getDefaultToolkit().beep(); // Play beep sound when a non-digit character is found
        JOptionPane.showMessageDialog(null, "Please enter only integer values.", "Input Error",
JOptionPane.ERROR_MESSAGE);
      }
    }
  }

  // Method to set the placeholder for the JTextField
  /**
 * Sets a placeholder text for a JTextField. When the field is unfocused and empty,
 * the placeholder text will be shown in a gray color. When the field gains focus, the
 * placeholder text disappears and the text color becomes black.
 *
 * @param textField The JTextField for which the placeholder needs to be set.
 * @param placeholder The placeholder text to be displayed when the JTextField is empty.
 */
  private void setPlaceholder(JTextField textField, String placeholder) {
     // Initially set the JTextField with the placeholder and a gray color
    textField.setText(placeholder);
    textField.setForeground(Color.GRAY);
     // Add a focus listener to the JTextField to handle the appearance and disappearance of the
placeholder
    textField.addFocusListener(new FocusListener() {
      /**
     * This method is called when the JTextField gains focus. If the current text
     * in the JTextField is the placeholder, it clears the text and sets the
     * color to black.
     *
     * @param e The focus event
     */
        @Override
        public void focusGained(FocusEvent e) {
          if (textField.getText().equals(placeholder)) {
```

```
                textField.setText("");
                textField.setForeground(Color.BLACK);
            }
        }
    /**
    * This method is called when the JTextField loses focus. If the JTextField
    * is empty, it sets the text to the placeholder and changes the color to gray.
    *
    * @param e The focus event
    */
        @Override
        public void focusLost(FocusEvent e) {
            if (textField.getText().isEmpty()) {
                textField.setText(placeholder);
                textField.setForeground(Color.GRAY);
            }
        }
      });
  }
  /**
* This method is responsible for creating a RegularStudent object by extracting and
* processing information from various UI elements like JTextFields and JComboBoxes.
*/
  private void createRegularStudent() {
      boolean canCreateStudent = true;
     try{
        // Fetching the student name from the JTextField
        studentName = studentNameText.getText();
        //System.out.println(studentName);
        // Extracting the selected year, month, and day from the date of birth JComboBoxes
        String dateOfBirthyear = String.valueOf(birthDateComboBox.getSelectedItem());
        String dateOfBirth1=String.valueOf(birthDateComboBox1.getSelectedItem());
        String dateOfBirth2=String.valueOf(birthDateComboBox2.getSelectedItem());
        // Constructing the full date of birth string
        dateOfBirth=dateOfBirthyear+dateOfBirth1+dateOfBirth2;
        // Fetching the course name and duration from their respective JTextFields
        courseName = courseNameText.getText();
        courseDuration = Integer.parseInt(courseDurationText.getText());
        // Fetching the enrollment ID from the JTextField
        enrollmentID = Integer.parseInt(enrollmentIDText.getText());
        //System.out.println(enrollmentID);
        // Extracting the selected year, month, and day from the date of enrollment JComboBoxes
        String dateOfEnrollmenty = String.valueOf(enrollmentDateComboBox.getSelectedItem());
        String a=String.valueOf(enrollmentDateComboBox1.getSelectedItem());
        String b=String.valueOf(enrollmentDateComboBox2.getSelectedItem());
        // Constructing the full date of enrollment string
        dateOfEnrollment=dateOfEnrollmenty+a+b;
        // Fetching tuition fee, number of modules, number of credit hours, and days present
```

```
        // from their respective JTextFields
        tuitionFee = (int)Double.parseDouble(tuitionFeeText.getText());
        numOfModules = Integer.parseInt(numOfModulesText.getText());
        numOfCreditHours = Integer.parseInt(numOfCreditHoursText.getText());
        daysPresent = (int)Double.parseDouble(daysPresentText.getText());

        // Check if course name is vacant, enrollment ID is vacant, or days present is 0
      if(courseName.trim().isEmpty() || enrollmentID <= 0 || daysPresent <= 0) {
        Toolkit.getDefaultToolkit().beep();
        throw new IllegalArgumentException("Failed to create object! Ensure course name, enrollment ID,
and days present are valid.");

      }
       // If we reach this point, the inputs are valid, and we can create the object
      Regular regularStudent = new Regular(studentName, dateOfBirth, courseName, courseDuration,
enrollmentID, dateOfEnrollment, tuitionFee, numOfModules, numOfCreditHours, daysPresent);
      arraylist.add(regularStudent); // Assuming arraylist is the collection to hold students

      // Success message
      JOptionPane.showMessageDialog(mainFrame, "Regular Student created successfully.");
      } catch(Exception e) {
      // Displaying an error message if there is an exception (like incorrect input formats)
      Toolkit.getDefaultToolkit().beep();
      JOptionPane.showMessageDialog(mainFrame, "Failed to create object!" ,"Error",
JOptionPane.ERROR_MESSAGE);
      return; // Exit the method without proceeding to create the object
    }




}

   /**
 * This method extracts values entered by the user in the Dropout student form fields.
 * It retrieves the user inputs from the form components and sets them to class properties.
 * If any error occurs during the extraction of data, a simple error message is displayed to the user.
 */
   public void Dropoutgetvalue()
   {
      try{
        // Retrieve data from the form components for Dropout students
        studentName = studentNameText1.getText();
        // For debugging purposes: Print the retrieved data
        System.out.println(studentName);
        String dateOfBirthyear = String.valueOf(birthDateComboBox0.getSelectedItem());
        String dateOfBirth1=String.valueOf(birthDateComboBox01.getSelectedItem());
        String dateOfBirth2=String.valueOf(birthDateComboBox02.getSelectedItem());
```

```
        dateOfBirth=dateOfBirthyear+dateOfBirth1+dateOfBirth2;
        System.out.println(dateOfBirth);
        courseName = courseNameText1.getText();
        System.out.println(courseName);
        courseDuration = Integer.parseInt(courseDurationText1.getText());
        System.out.println(courseDuration );
        enrollmentID = Integer.parseInt(enrollmentIDText1.getText());
        System.out.println(enrollmentID);
        String dateOfEnrollmenty = String.valueOf(enrollmentDateComboBox0.getSelectedItem());
        String a=String.valueOf(enrollmentDateComboBox01.getSelectedItem());
        String b=String.valueOf(enrollmentDateComboBox02.getSelectedItem());
        dateOfEnrollment=dateOfEnrollmenty+a+b;
        System.out.println( dateOfEnrollment );
        tuitionFee = (int)Double.parseDouble(tuitionFeeText1.getText());
        System.out.println( tuitionFee );
        String dateofDropout=String.valueOf(dropoutDateComboBox0.getSelectedItem());
        String dateofDropout1=String.valueOf(dropoutDateComboBox01.getSelectedItem());
        String dateofDropout2=String.valueOf(dropoutDateComboBox02.getSelectedItem());
        dateOfDropout=dateofDropout+dateofDropout1+dateofDropout2;
        System.out.println( dateOfDropout );
        numOfMonthsAttended=Integer.parseInt(numOfMonthsAttendedText.getText());
        System.out.println( numOfMonthsAttended);
        numOfRemainingModules=Integer.parseInt(numOfRemainingModulesText.getText());
        System.out.println( numOfRemainingModules);

      }
      catch(Exception e)
      {
          // Display error message if there's any exception during data retrieval
          Toolkit.getDefaultToolkit().beep();
        JOptionPane.showMessageDialog(mainFrame, "Error!");
      }
   }
   /**
 * This method is invoked to create a Dropout student instance and add it to the ArrayList.
 * It first retrieves values from the Dropout form fields, then creates a Dropout student
 * object with the retrieved values, and finally adds the object to the ArrayList.
 */
   public void adddropout()
   {
      try{
        // Extract values from the Dropout form fields
        this.Dropoutgetvalue();
        // Create a new instance of Dropout student with the gathered information
        Dropout d=new
Dropout(courseName,dateOfEnrollment,dateOfBirth,studentName,courseDuration,tuitionFee,
numOfRemainingModules, numOfMonthsAttended,dateOfDropout,enrollmentID );
        // Add the newly created Dropout student instance to the arraylist
```

```
        arraylist.add(d);
        // Notify the user about the successful creation of the Dropout student
        JOptionPane.showMessageDialog(mainFrame, "Dropout student created successfully!");
      }
      catch(Exception e)
      {
        // Display error message if there's any exception during the creation process
        Toolkit.getDefaultToolkit().beep();
        JOptionPane.showMessageDialog(mainFrame, "Error!");
      }
    }
    /**
  * This method is invoked when the "Add Regular Student" button is pressed.
  * It initiates the creation of a Regular student, encapsulates the student
  * information into a Regular object, and adds the object to an ArrayList.
  */
    public void addaRegularStudentButton()
    {
       // Generate the properties for the regular student from user input
      this.createRegularStudent();

    }
    /**
  * This method clears all input fields and resets the combo boxes related to
  * the RegularStudent data entry section of the UI.
  */
    private void clearRegularStudentFields() {
      // Clear the text fields and reset combo boxes
      setPlaceholder(studentNameText, "Full Name");
      setPlaceholder(courseNameText, "Module code");
      enrollmentIDText.setText("");
      daysPresentText.setText("");
      tuitionFeeText.setText("");
      courseDurationText.setText("");
      numOfModulesText.setText("");
      numOfCreditHoursText.setText("");

       // Resetting JComboBoxes for date fields to their default values
       // (assuming the first index, 0, is a default or placeholder value)
      enrollmentDateComboBox.setSelectedIndex(0);
      enrollmentDateComboBox1.setSelectedIndex(0);
      enrollmentDateComboBox2.setSelectedIndex(0);
      birthDateComboBox.setSelectedIndex(0);
      birthDateComboBox1.setSelectedIndex(0);
      birthDateComboBox2.setSelectedIndex(0);
    }
    /**
  * This method clears all input fields and resets the combo boxes related to
```

```
 * the DropoutStudent data entry section of the UI.
 */
   private void clearDropoutStudentFields() {
      // Clear the text fields and reset combo boxes
      setPlaceholder(studentNameText1, "Full Name");
      setPlaceholder(courseNameText1, "Module code");
      enrollmentIDText1.setText("");
      numOfRemainingModulesText.setText("");
      tuitionFeeText1.setText("");
      courseDurationText1.setText("");
      numOfMonthsAttendedText.setText("");

       // Resetting JComboBoxes for date fields to their default values
      // (assuming the first index, 0, is a default or placeholder value)
      dropoutDateComboBox0.setSelectedIndex(0);
      birthDateComboBox01.setSelectedIndex(0);
      dropoutDateComboBox02.setSelectedIndex(0);
      enrollmentDateComboBox0.setSelectedIndex(0);
      enrollmentDateComboBox01.setSelectedIndex(0);
      enrollmentDateComboBox02.setSelectedIndex(0);
      birthDateComboBox0.setSelectedIndex(0);
      birthDateComboBox01.setSelectedIndex(0);
      birthDateComboBox02.setSelectedIndex(0);
   }

    @Override
     public void actionPerformed(ActionEvent e)  {
       // Check if the action is triggered by the 'addaRegularStudentButton' button
      if (e.getSource() == addaRegularStudentButton)
      {
         this.addaRegularStudentButton();
      }
      // Check if the action is triggered by the 'calculatepresentPercentageofRegularStudentButton'
button
      if (e.getSource() == calculatepresentPercentageofRegularStudentButton)
      {
         // Calculate the present percentage for a Regular Student based on the given enrollment ID
         int id=Integer.parseInt(JOptionPane.showInputDialog("What is Enrollment"));
         boolean isFound = false;
         for (Student
s:arraylist)//calculatepresentPercentageofRegularStudentButton.addActionListener(this);
         {
            if(s instanceof Regular)
            {
               Regular r=(Regular)s;

               if (id==r.getEnrollmentID()) {
                System.out.println("Entered Enrollment ID: " + id);
```

```
            System.out.println("Found Enrollment ID: " + r.getEnrollmentID());
                isFound = true; // Found a matching ID
            // Input number of days present and calculate the grade

                // Prompt for the number of days present
                int dayspresent=(int)Double.parseDouble(JOptionPane.showInputDialog("Enter number  of
dayspresent"));
                if ((r.getEnrollmentID())*30 >= dayspresent)
                {
                    char grade=r.PresentPercentage(dayspresent);
                    JOptionPane.showMessageDialog(mainFrame, "Present percentage is calculated");
                    JOptionPane.showMessageDialog(mainFrame, "Your Grade is " +  grade);

                }
                break;// Exit loop once a match is found
            }
        }
    }
            // If the ID was not found in the list, show the error message
            if (!isFound) {
                Toolkit.getDefaultToolkit().beep();
                throw new IllegalArgumentException("enrollment ID not found");

            }



    }
    // Grant certificate for the Regular Student
if (e.getSource() == grantCertificateofRegularStudentButton) {
String enrollmentInput = JOptionPane.showInputDialog("What is Enrollment");
if (enrollmentInput == null || enrollmentInput.trim().isEmpty()) {
    JOptionPane.showMessageDialog(mainFrame, "Enrollment ID must be provided");
    return; // Exit the method if there is no input
}
int id;
try {
    id = Integer.parseInt(enrollmentInput);
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(mainFrame, "Enrollment ID must be a valid number");
    return; // Exit the method if the input is not a valid integer
}

boolean isFound = false; // Add a flag to check if the enrollment ID is found
for (Student s : arraylist) {
    if (s instanceof Regular) {
        Regular r = (Regular) s;
        if (id == r.getEnrollmentID()) {
```

```
            String cs = JOptionPane.showInputDialog("Enter number of days present");
            if (cs == null || cs.trim().isEmpty()) {
            JOptionPane.showMessageDialog(mainFrame, "Number of days must be provided");
            return; // Exit the method if there is no input
        }

        // Continue with granting the certificate
            r.GrantCertificate(cs, id, dateOfEnrollment);
            JOptionPane.showMessageDialog(mainFrame, "Certificate granted");
            System.out.println("\nVerfied by the school Administrative."); // Print to console
            isFound = true; // Mark the enrollment ID as found
            break; // Exit loop once a match is found
        }
    }
  }

  if (!isFound) { // If the ID was not found in the list, show the error message
    Toolkit.getDefaultToolkit().beep();
    JOptionPane.showMessageDialog(mainFrame, "Enrollment ID not found");
  }
}

    // Display the details of a Regular Student based on enrollment ID
    if (e.getSource() == displayButton)
  {
  try
  {
    int id = Integer.parseInt(JOptionPane.showInputDialog("What is Enrollment"));
    boolean studentFound = false;

    for (Student s : arraylist)
    {
        if (s instanceof Regular)
        {
            Regular r = (Regular) s;
            if (id == r.getEnrollmentID())
            {
                r.display();
                JOptionPane.showMessageDialog(mainFrame, "Displayed");
                studentFound = true;
                break; // exit the loop once the student is found
            }
        }
    }

    if (!studentFound)
    {
        Toolkit.getDefaultToolkit().beep();
```

```
            JOptionPane.showMessageDialog(mainFrame, "Enrollment ID not found");
        }
    }
    catch (NumberFormatException nfe)
    {
        Toolkit.getDefaultToolkit().beep();
        JOptionPane.showMessageDialog(mainFrame, "Please enter a valid enrollment number.");
    }
    // You might also want to catch other potential exceptions
    catch (Exception ex)
    {
        Toolkit.getDefaultToolkit().beep();
        JOptionPane.showMessageDialog(mainFrame, "An error occurred: " + ex.getMessage());
    }
}


   // Clear fields for Regular Student input
        if (e.getSource() == clearButton)
        {
            clearRegularStudentFields();
            JOptionPane.showMessageDialog(mainFrame, "Fields cleared.", "Success",
JOptionPane.INFORMATION_MESSAGE);
        }
// Add a Dropout Student
        if(e.getSource()==addaDropoutStudentButton)
        {
            this.adddropout();
        }
         // Calculate bills payable for a Dropout Student
        if(e.getSource()==paythebillsofDropoutStudentButton)
        {
            int sdid= Integer.parseInt(JOptionPane.showInputDialog("Enter your EnrollmentID"));
             boolean idMatchFound = false;
            for(Student s:arraylist)
            {
                if (s instanceof Dropout) {
                    Dropout drop = (Dropout) s;
                    if (sdid==drop.getEnrollmentID())
                    {
                        drop.billsPayble();
                        JOptionPane.showMessageDialog(mainFrame,"Bills payable calculated succesfully");
                        idMatchFound = true;
                        break;
                    }
                }
            }
```

```
            if (!idMatchFound) {
                Toolkit.getDefaultToolkit().beep();
               JOptionPane.showMessageDialog(mainFrame, "Enrollment ID did not match any records");
            }


      }
// Remove a Dropout Student if they have paid the bills
      if(e.getSource()==removeDropoutStudentButton)
      {
          String enrollmentIdInput = JOptionPane.showInputDialog("Enter your EnrollmentID");
   if (enrollmentIdInput == null || enrollmentIdInput.trim().isEmpty()) {
      JOptionPane.showMessageDialog(mainFrame, "Enrollment ID must be provided");
      return; // Exit the method if there is no input
   }

   int sdid;
   try {
      sdid = Integer.parseInt(enrollmentIdInput);
   } catch (NumberFormatException ex) {
      JOptionPane.showMessageDialog(mainFrame, "Enrollment ID must be a valid number");
      return; // Exit the method if the input is not a valid integer
   }

         boolean studentRemoved = false;  // To check if a student was actually removed
         boolean billsCleared = false;    // To check if student's bills are cleared

         for(Student s:arraylist)
         {
            if (s instanceof Dropout) {
               Dropout drop = (Dropout) s;
               if (sdid==drop.getEnrollmentID())
               {
                  if(drop.getHasPaid())
                  {
                     drop.removeStudent();
                  studentRemoved = true;

               Toolkit.getDefaultToolkit().beep();
               JOptionPane.showMessageDialog(mainFrame, "Student removed successfully.");
            } else {
               Toolkit.getDefaultToolkit().beep();
               JOptionPane.showMessageDialog(mainFrame, "Cannot remove student as bills are not
cleared.", "Warning", JOptionPane.WARNING_MESSAGE);

            }
            break; // Exit the loop once a student's condition is checked
         }
```

```
        }
    }

    // If no matching student was found, show an error message
    if (!studentRemoved) {
        Toolkit.getDefaultToolkit().beep();

    }

        }
        // Display attributes of all Dropout Students
 if (e.getSource() == displayButton1) {
    try {
        int id = Integer.parseInt(JOptionPane.showInputDialog("Enter Dropout Student Enrollment ID"));
        boolean studentFound = false;

        for (Student s : arraylist) {
            if (s instanceof Dropout) {
                Dropout drop = (Dropout) s;
                if (id == drop.getEnrollmentID()) {
                    drop.display();
                    JOptionPane.showMessageDialog(mainFrame, "Dropout student details displayed.");
                    studentFound = true;
                    break; // exit the loop once the student is found
                }
            }
        }

        if (!studentFound) {
            Toolkit.getDefaultToolkit().beep();
            JOptionPane.showMessageDialog(mainFrame, "Enrollment ID not found for Dropout student.");
        }
    } catch (NumberFormatException nfe) {
        Toolkit.getDefaultToolkit().beep();
        JOptionPane.showMessageDialog(mainFrame, "Please enter a valid enrollment number for Dropout
student.");
    } catch (Exception ex)// You might also want to catch other potential exceptions
    {
        Toolkit.getDefaultToolkit().beep();
        JOptionPane.showMessageDialog(mainFrame, "An error occurred: " + ex.getMessage());
    }
}


        if (e.getSource() == clearButton1) {
         clearDropoutStudentFields();
         JOptionPane.showMessageDialog(mainFrame, "Dropout student fields cleared.", "Success",
JOptionPane.INFORMATION_MESSAGE);
```

```
    }
}

public static void main(String[]args)
{
    Student_GUI student=new Student_GUI();
}

}
```