Module: WSQ Programming Foundations (SF)

Module Project

Student Management System

HOÀNG MINH TRƯỜNG

Code: edus000452

Start Date:	25/09/2023	**End Date:**	01/10/2023
Submission Date:	01/10/2023	**Presentation Date:**	

Content

No.	Description
1	Project Background & Objective
2	Project Deliverables
3	Project Milestones & Tasks
4	Project Environment
5	Tools Used
6	Project Design
7	Classes & Methods
8	Steps from coding to execution
9	Coding Standards
10	Project Results
11	Proposed Improvements
12	References

1. Project Background & Objective

1.1 Problem statement

As a Junior Programmer at GreatLearning University, you are part of a team that designs innovative, and data and interactions through ArrayLists, the system aims to promote better communication user-friendly applications. You are assigned to develop a centralized platform for managing student, data accuracy, and informed decision-making.

1. Project Background & Objective

1.2 Project Objective

- **Create an application** to **manage student information**, this information includes personal information and corresponding information of students during the learning process. The application is **written in Java** language, allowing users to operate through the **console window**.
- The project is a great opportunity to help students **understand program structure** when written in Java, how to use o**Class, Object, Methods**, and **Contribute**. Practice programming in an **object-oriented programming** language, practice thinking about designing and deploying a specific application.
- Use knowledge about*: **Loops, Switch Case, Scanner, If-Else, Array List***.

1. Project Background & Objective

1.3 Feature Scope applications

- Ask user how many Student will be added
- Enter Student information in turn
- Student just can enroll course 101:106
- Student able to view student ballance
- Student able to choose method pay(Partially/Full)
- Student status can be viewed by using their ID
- Send Confirmation Email

2. Project Deliverables

Project Deliverables

- Project Plan
- Report

List of Evidences

- Report
- Presentation
- -Code

3. Project Milestones & Tasks

Project Task ID	Project Task Description	Project Milestone ID
01	Project analysis	1 - Planing
02	Define Objective & Scope	1 - Planing
03	Create a Use Case Diagram	2 - Design System
04	Create Class Diagram	2 - Design System
05	Create Activities Diagram	2 - Design System
06	Programming flow chart	2 - Design System
07	Design functions in the Student Class	3 - Implementation
08	Design functions in the Student List Class	3 - Implementation
09	Demo	3 - Implementation
10	Edit functions	3 - Implementation
11	Unit test design	3 - Implementation
12	Project evaluation	4 – Final Report
13	Make a report	4 – Final Report

3. Project Milestones & Tasks

Project Task ID	Project Task Description	Project Milestone ID	20/09	21/09	22/09	23/09	24/09	25/09	26/09	27/09	28/09	29/09	30/09	01/10	Document
1	Project analysis	1 - Planing													
2	Define Objective & Scope														
3	Create a Use Case Diagram	2 - Design System													Use Case Diagram
4	Create Class Diagram														Class Diagram
5	Create Activities Diagram														Activities Diagram
6	Programming flow chart														
7	Design functions in the Student Class	3 - Implementation													
8	Design functions in the Student List Class														
9	Demo														
10	Edit functions														
11	Unit test design														
12	Project evaluation	4 – Final Report													
13	Make a report														Report.Docx Presentation Code.zip

4. Project Environment

Technical Environment and Tools Used

5. Tools Used

IDE & Support Tool

Mainly used for coding and creating applications

- Simple, Compact, Easy to install
- Integrated static code analysis
- Excellent usability and performance
- Many Plugin support available.
- Supports diagramming, modeling, reporting as well as testing
- Can be used directly after installing the software

Draw.io

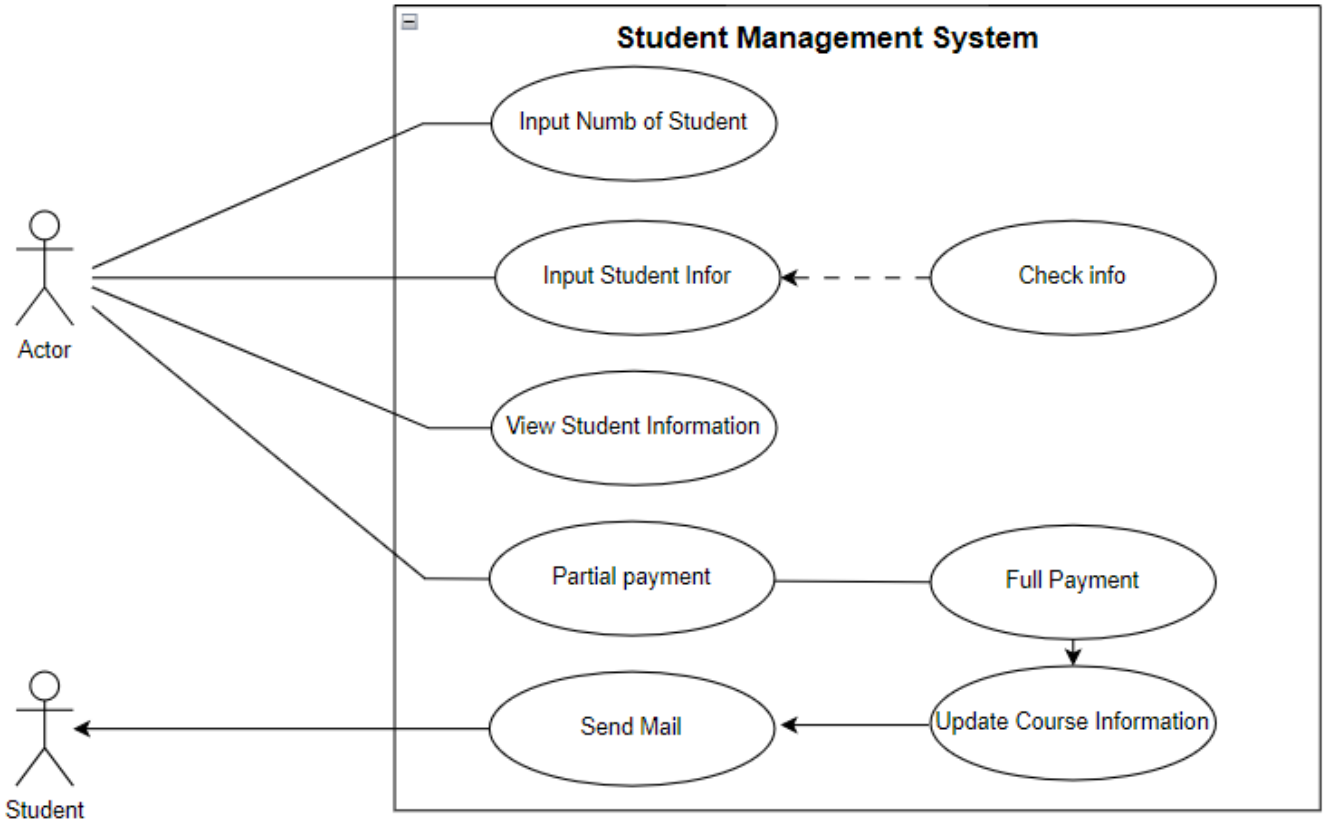
Mainly used to create flowcharts during component design

- No installation required
- Simple, easy to use
- Powerful tool, supports many chart types
- Convenient. Can access, draw, adjust and export files online
- There is a supporting UML library
- There is no limit to the number of charts like many other tools or platforms.

6. Project Design

System Design – Use Case Diagram

Through the requirements given in the problem, along with the limitation of the scope of implementation, because there is no scope for user login or authorization, the actor can perform all functions of the system. Therefore, the project aims to design for the highest level user - admin. After the admin operates on the system, a confirmation email will be sent to all students.



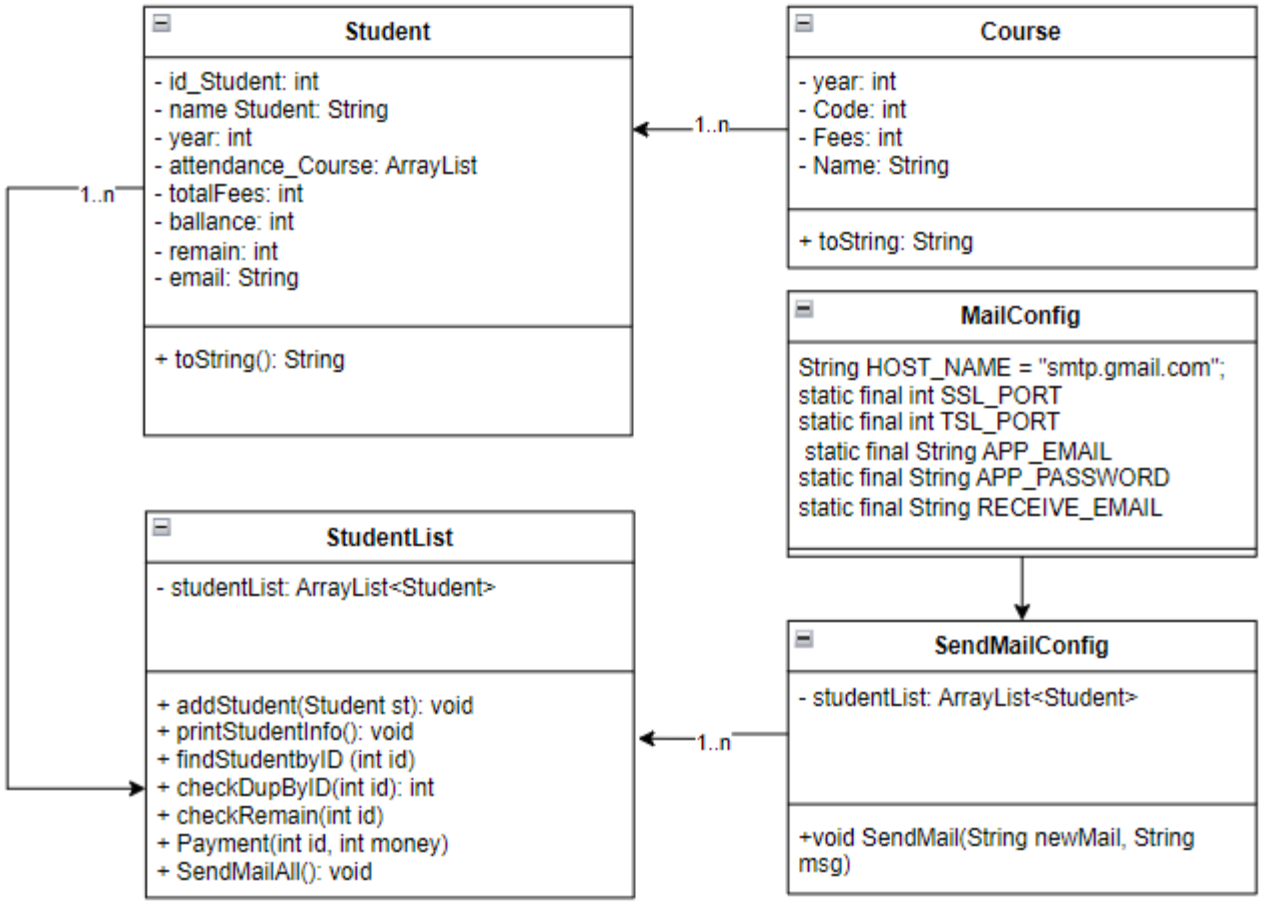
7. Classes & Methods

List of Classes & Methods used

The information used in the system is declared, managed and used according to the following diagram.

here are 4 main Classes used including:

- Student: Contains basic information of students
- StudentList: Contains information about the list of all students & functions used for calculation
- MailConfig & Sendmail Config declare & install methods to support sending mail from the Javax.Mail library

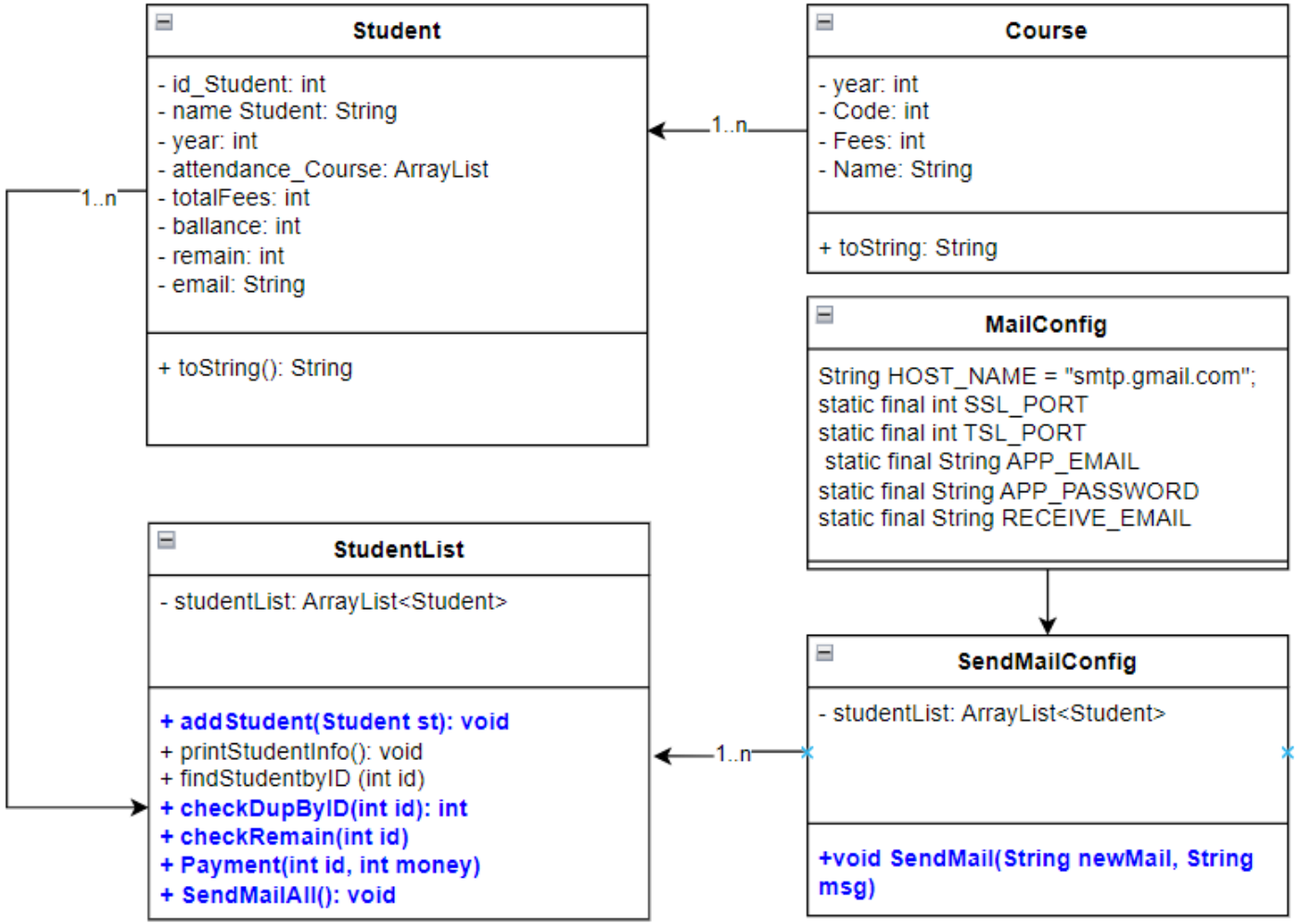


7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- Add Student
- findStudentByID(int id)
- Payment Implementation
- Method Send Mail



7. Classes & Methods

List of Classes & Methods used

Methods used:

- **Method Add Student**
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- Payment Implementation
- Method Send Mail

Input No of Student → Student ID → Student Name → Student's Email → Student Year → Amount of Course → Course in list

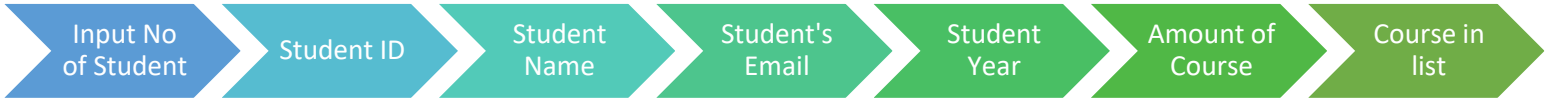
```
if(luaChon ==1) {
    System.out.println("How Many Student you want to add?"); int n = input.nextInt();
    for(int i=1;i<=n;i++) {
        System.out.println("Input Student ID No:"+i); int studentID = input.nextInt();
        //Ràng buộc điều kiện, không để ID ngoài đk đề, và k trùng.
        // Sử dụng hàm phụ check Dup để ràng buộc
        while(studentID<10000 ||studentID>39999||studentList.checkDupbyID(studentID) == 1 ) {
            if(studentID<10000 ||studentID>39999) {
                System.out.println("Wrong Student ID, must a 5-digit unique ID,start with year:");
            } else if(studentList.checkDupbyID(studentID) == 1) {
                System.out.println("This student already appears in the list, please re-input:");
            }
            studentID = input.nextInt();
        }
        input.nextLine();
        System.out.println("Input Student Name:"+i); String studentName = input.nextLine();
        System.out.println("Input Student's Email:"); String studentMail = input.nextLine();
        System.out.println("Input Student Year:"+i); int year = input.nextInt();
        while (year<1 || year >3) {
            System.out.println("Wrong year, year in range [1,3]. try again:");
            year = input.nextInt();
        }
        Student st = new Student(studentID,studentName,year,studentMail);
        System.out.println("Input amount of course:"); int k = input.nextInt();
        while (k <0 || k>6) {
            if(k<0) {
                System.out.println("Students must register for at least 1 course, Try again:");
            }else {
                System.out.println("Students can only register for a maximum of 6 courses, Try Again:");
            }
        }
```

7. Classes & Methods

List of Classes & Methods used

Methods used:

- **Method Add Student**
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- Payment Implementation
- Method Send Mail




```
switch(course_ID) {
case 101:
    Course course1 = new Course(1,101,"Programming Foundations",1200);
    st.getAttendance_Course().add(course1);
    int a= st.getTotalFees();
    a += 1200;
    st.setTotalFees(a);
    st.setRemain(a);
    break;
/*
 *Similar for the courses in the list (Course 102:105)
 */
case 106:
    Course course6 = new Course(1,106,"Capstone Project",2500);
    st.getAttendance_Course().add(course6);
    //st.getBallance()+=2500;
    int f= st.getTotalFees();
    f += 2500;
    st.setTotalFees(f);
    st.setRemain(f);
    break;
}
```

7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- **Method checkDupbyID(int ID)**
- findStudentByID(int id)
- Payment Implementation
- Method Send Mail

```
//Check Dupt by ID
public int checkDupbyID(int id) {
                int is_Dup=0;
                for (Student student : studentList) {
                        int a = student.getId_Student();
                if(id == a) {
                is_Dup =1;
                }
        }
                return is_Dup;
        }
```

7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- **findStudentByID(int id)**
- Payment Implementation
- Method Send Mail

The method allows users to search for student information by parameter ID. If the ID is not found, the system will return message that not found student. Otherwise, if found, the method will print out the corresponding student information.

```
public void findStudentByID(int id) {
        int checkempty = 0;
        for (Student student : studentList) {
        int a = student.getId_Student();
        if(id == a) {
                checkempty +=1;
                System.out.println(student);
                        }
        }
        if( checkempty == 0) {
        System.out.println("There are no student macth ID No: "+ id);
                }
        }
```

7. Classes & Methods

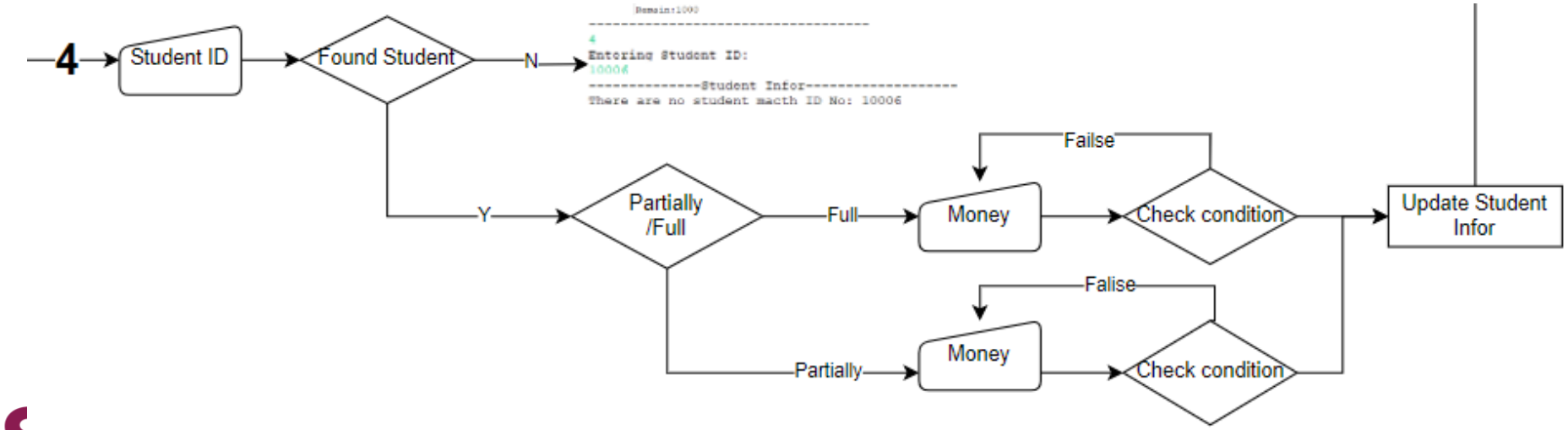
List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- **Payment Implementation**
- Method Send Mail

When initiating the payment process, the user must enter a Student ID. The system will return the student's current information, from which the user can choose the appropriate payment method.

After finding student information, the switch case structure gives the user two options: partial payment or full payment.



7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- **Payment Implementation**
- Method Send Mail

```
System.out.println("Entering Student ID:"); int stID= input.nextInt();
System.out.println("--------------Student Infor--------------------");
studentList.findStudentByID(stID);
System.out.println(
        "Choose Payment Method\r\n"
        + "1. Charge Partially\r\n"
        + "2. Charge Full\r\n"
        + "0. Exit\r\n");
int paymentMethod = input.nextInt();

while (paymentMethod<0 || paymentMethod >2) {
    System.out.println("Wrong Method, try again:");
    paymentMethod = input.nextInt();
}
switch(paymentMethod) {
case 1:
    /*
     * Charge Partially
     */
    break;
case 2:
    /*
     * Charge full
     */
    break;
default:
    System.out.println("---------Exit Payment---------\r\n");
    break;
}
```

7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- **Payment Implementation**
  - Check Remain(int ID)
  - Payment(int ID, int Money)
- Method Send Mail

```
// Check Remain by ID
public int checkRemain(int id) {
        int b=0;
        for (Student student : studentList) {
                int a = student.getId_Student();
                if(id == a) {
                        b = student.getRemain();
                }
        }
        return b;
}
//Payment - Input money of Student
public void Payment(int id, int money) {
        for (Student student : studentList) {
                int a = student.getId_Student();
                if(id == a) {
                        int b = student.getRemain();
                        int b1 = student.getBallance();
                        student.setRemain(b-money)
student.setBallance(b1+money);
        }
}
}
```

7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- **Payment Implementation**
- Method Send Mail

```
switch(paymentMethod) {
case 1:
    //Charge Partially
    System.out.println("Input Amount of Money"); int money = input.nextInt();
    while (money < 0 || money >= studentList.checkRemain(stID)) {
        System.out.println("Wrong money, amount must less than or equal the remain");
        money = input.nextInt();
        if(money ==0 ) {
            break;
        }
    }
    studentList.Payment(stID, money);
    System.out.println("---------------Successful payment, new information:----------\n");
    studentList.findStudentByID(stID);
    break;
case 2:
    //Charge Full
    System.out.println("Input Amount of Money"); int money2 = input.nextInt();
    while (money2 != studentList.checkRemain(stID)) {
        System.out.println("Wrong money, amount must equal the remain");
        money2 = input.nextInt();
    }
    studentList.Payment(stID, money2);
    System.out.println("---------------Successful payment, new information:----------\n");
    studentList.findStudentByID(stID);
    break;
default:
    System.out.println("---------Exit Payment---------\r\n");
    break;
}
```

7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- Payment Implementation
- **Method Send Mail**

Setup Class Mail Config

```
public class MailConfig {
public static final String HOST_NAME =
"smtp.gmail.com";
public static final int SSL_PORT = 465;
public static final int TSL_PORT= 587;
public static final String APP_EMAIL
="pgsepfshoangminhtruong@gmail.com";
public static final String APP_PASSWORD = "jlea
ikyw qtcs gnym"; // your password
public static final String RECEIVE_EMAIL =
"pgsepfshoangminhtruong@gmail.com";
//public String RECEIVE_EMAIL;
}
```

7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- Payment Implementation
- **Method Send Mail**

Setup Class SendmailConfig

```
public void SendMail(String newMail, String msg) {
// Get properties object
Properties props = new Properties();
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.host", MailConfig.HOST_NAME);
props.put("mail.smtp.socketFactory.port", MailConfig.SSL_PORT);
props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
props.put("mail.smtp.port", MailConfig.SSL_PORT);
// get Session
Session session = Session.getDefaultInstance(props, new javax.mail.Authenticator() {
protected PasswordAuthentication getPasswordAuthentication() {
return new PasswordAuthentication(MailConfig.APP_EMAIL, MailConfig.APP_PASSWORD);
}
});
// compose message
try {
MimeMessage message = new MimeMessage(session);
message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(newMail));
message.setSubject("Congratulations on your successful registration - Student Management
System");
message.setText(msg);
// send message
Transport.send(message);
System.out.println("Message sent successfully");
} catch (MessagingException e) {
throw new RuntimeException(e);
```

7. Classes & Methods

List of Classes & Methods used

Methods used:

- Method Add Student
- Method checkDupbyID(int ID)
- findStudentByID(int id)
- Payment Implementation
- **Method Send Mail**

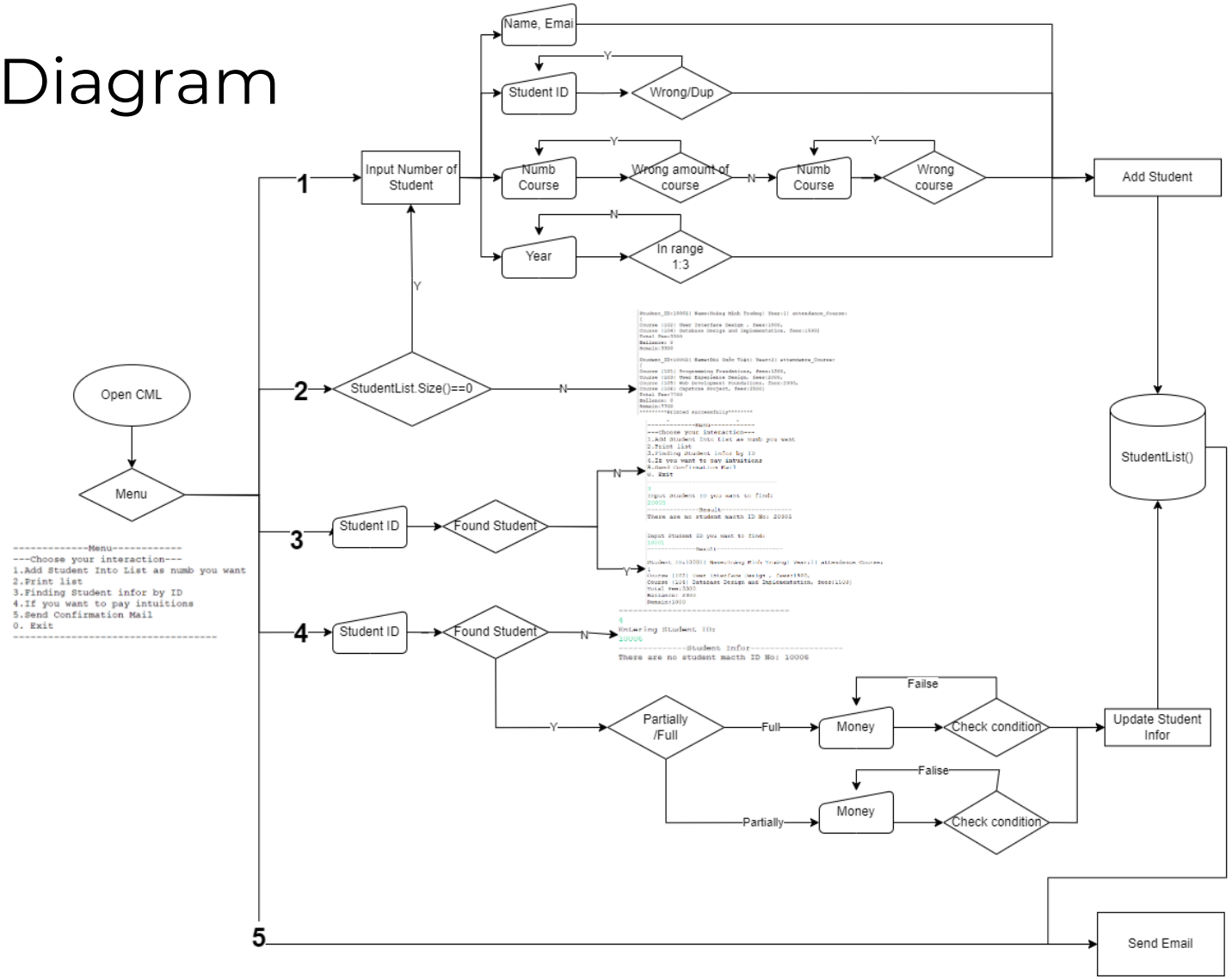
Method SendMailAll()

```
//Send Mail
public void SendMailAll() {
SendMailConfig sm = new SendMailConfig();
for (Student student : studentList) {
sm.SendMail(student.getEmail(),"Dear
Candidate,\r\n"+"Congratulations on your successful
registration, below is your information:\r\n" +
student.toString());
}
}
public int size() {
// TODO Auto-generated method stub
return 0;
}
```

8. Steps from coding to execution

Application execution methods

Process Diagram



8. Steps from coding to execution

Application execution methods

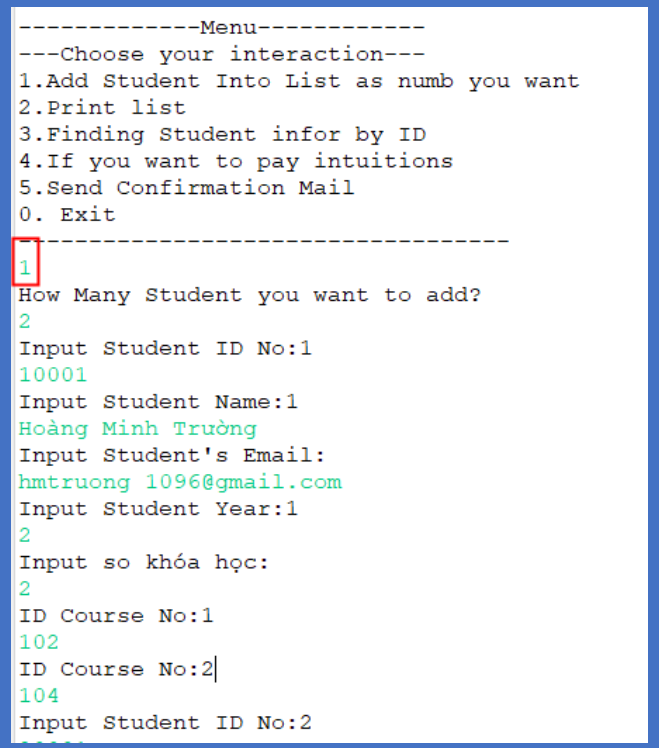
Main Loop

Activities	UI Page	Descriptions
Start/Main loop	----------Menu---------- ---Choose your interaction--- 1.Add Student Into List as numb you want 2.Print list 3.Finding Student infor by ID 4.If you want to pay intuitions 5.Send Confirmation Mail 0. Exit ----------	The main loop presents the main options for the user to choose from. The loop is arranged in switch cases, branching out into different subprograms.

8. Steps from coding to execution

Application execution methods

Adding Student Information

Adding student Information		**When the user selects option 1. The user enters the number of students they wants to add. Fill in information about:** • **Student ID** • **Student Name** • **Year** • **Amount of Courses** • **Courses** **When the information entered does not meet the requirements, the system will notify you requesting re-entry. Some errors are as follows:**

8. Steps from coding to execution

Application execution methods

Adding Student Information

	Wrong Student ID, must a 5-digit unique ID,start with year: 20001 This student already appears in the list, please re-input: 1003	• **Student ID is in wrong format** • **Student ID is duplicated**
	Input Student Year:1 6 Wrong year, year in range [1,3]. try again: 2	• Student Year not in range [1:3]
	ID Course No:1 1009 Error! Course_ID must in range 101-106. Try Again : 102	• The person entered the wrong course number, course_id must be in the range [101:106]

8. Steps from coding to execution

Application execution methods

Check Student List

Check Student List	Student_ID:10001| Name:Hoàng Minh Trường| Year:1| attendance_Course: [Course |102| User Interface Design , fees:1800, Course |104| Database Design and Implementation, fees:1500] Total Fee:3300 Ballance: 0 Remain:3300 Student_ID:10002| Name:Bùi Quốc Việt| Year:2| attendance_Course: [Course |101| Programming Foundations, fees:1200, Course |103| User Experience Design, fees:2000, Course |105| Web Development Foundations, fees:2000, Course |106| Capstone Project, fees:2500] Total Fee:7700 Ballance: 0 Remain:7700 *********Printed successfully********	**When selecting is 2. The system will print out all current student information.**

8. Steps from coding to execution

Application execution methods

Find Student

Find Student

```
-----------------------------------
3
Input Student ID you want to find:
10001
--------------Result-------------------

Student_ID:10001| Name:Hoàng Minh Trường| Year:1| attendance_Course:
[
Course |102| User Interface Design , fees:1800,
Course |104| Database Design and Implementation, fees:1500]
Total Fee:3300
Ballance: 0
Remain:3300
```

```
-------------Menu------------
---Choose your interaction---
1.Add Student Into List as numb you want
2.Print list
3.Finding Student infor by ID
4.If you want to pay intuitions
5.Send Confirmation Mail
0. Exit
-----------------------------------
3
Input Student ID you want to find:
20001
--------------Result-------------------
There are no student macth ID No: 20001
```

Users search for student information using Student ID. When found, the system prints all information of that student. If not found, the system will notify.

8. Steps from coding to execution

Application execution methods

Payment

Payment

```
4
Entering Student ID:
10001
--------------Student Infor--------------------

Student_ID:10001| Name:Hoàng Minh Trường| Year:1| attendance_Course:
[
Course |102| User Interface Design , fees:1800,
Course |104| Database Design and Implementation, fees:1500]
Total Fee:3300
Ballance: 0
Remain:3300
Choose Payment Method
1. Charge Partially
2. Charge Full
0. Exit

1
Input Amount of Money
1300
---------------Successful payment, new information:----------


Student_ID:10001| Name:Hoàng Minh Trường| Year:1| attendance_Course:
[
Course |102| User Interface Design , fees:1800,
Course |104| Database Design and Implementation, fees:1500]
Total Fee:3300
Ballance: 1300
Remain:2000
-------------Menu------------
```

```
--------------Student Infor--------------------

Student_ID:10001| Name:Hoàng Minh Trường| Year:1| attendance_Course:
[
Course |102| User Interface Design , fees:1800,
Course |104| Database Design and Implementation, fees:1500]
Total Fee:3300
Ballance: 1300
Remain:2000
Choose Payment Method
1. Charge Partially
2. Charge Full
0. Exit

1
Input Amount of Money
3000
Wrong money, amount must less than or equal the remain
```

When the user wants to make a payment, the user must enter the student ID.

If the student is not found, the user can select 0 to exit the payment method.

When a student is found, the system will give the user two options.

- Partial payment.
- Full payment.

In there:

Total Fee: Total tuition fee of the courses.

Balance: The amount the student has paid.

Remain: The remaining amount to be paid.

If selecting 1, the user needs to enter 0 < money < Remain. If not satisfied, the system will report an error.

8. Steps from coding to execution

Application execution methods

Payment

```
--------------Student Infor-------------------

Student_ID:10002| Name:Bùi Quốc Việt| Year:2| attendance_Course:
[
Course |101| Programming Foundations, fees:1200,
Course |103| User Experience Design, fees:2000,
Course |105| Web Development Foundations, fees:2000,
Course |106| Capstone Project, fees:2500]
Total Fee:7700
Ballance: 0
Remain:7700
Choose Payment Method
1. Charge Partially
2. Charge Full
0. Exit

2
Input Amount of Money
1000
Wrong money, amount must equal the remain
7700
---------------Successful payment, new information:----------


Student_ID:10002| Name:Bùi Quốc Việt| Year:2| attendance_Course:
[
Course |101| Programming Foundations, fees:1200,
Course |103| User Experience Design, fees:2000,
Course |105| Web Development Foundations, fees:2000,
Course |106| Capstone Project, fees:2500]
Total Fee:7700
Ballance: 7700
Remain:0
-------------Menu------------
```

In case the user wants to pay in full. The system requires the correct Remain amount to be entered, otherwise an error will be reported.

8. Steps from coding to execution

Application execution methods

StudentList

Check Student List

```
-------------Menu------------
---Choose your interaction---
1.Add Student Into List as numb you want
2.Print list
3.Finding Student infor by ID
4.If you want to pay intuitions
5.Send Confirmation Mail
0. Exit
-----------------------------------
2

Student_ID:10001| Name:Hoàng Minh Trường| Year:1| attendance_Course:
[
Course |102| User Interface Design , fees:1800,
Course |104| Database Design and Implementation, fees:1500]
Total Fee:3300
Ballance: 2300
Remain:1000

Student_ID:10002| Name:Bùi Quốc Việt| Year:2| attendance_Course:
[
Course |101| Programming Foundations, fees:1200,
Course |103| User Experience Design, fees:2000,
Course |105| Web Development Foundations, fees:2000,
Course |106| Capstone Project, fees:2500]
Total Fee:7700
Ballance: 7700
Remain:0
*********Printed successfully********
```

After the payment process is completed, if the user selects option 2, the system will return the student's information. These have been updated with payment information.

8. Steps from coding to execution

Application execution methods

Send Mail

Send Email Confirmation		When selecting 5. The system will send confirmation emails to student emails in turn. With each successful send, the system will print a notification to the screen.

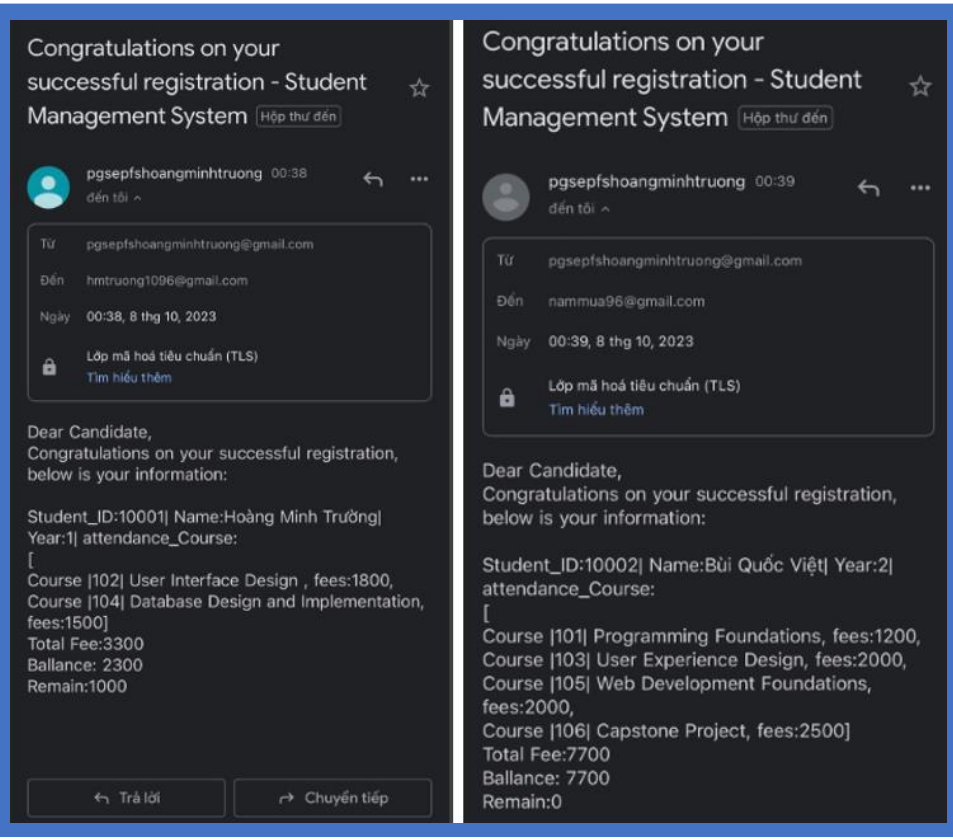
```
-------------Menu------------
---Choose your interaction---
1.Add Student Into List as numb you want
2.Print list
3.Finding Student infor by ID
4.If you want to pay intuitions
5.Send Confirmation Mail
0. Exit
---------------------------------

5
Message sent successfully
Message sent successfully
```

8. Steps from coding to execution

Application execution methods

Result

Result		After the actor sends the email. Students who have previously declared their email will receive an email informing them of the corresponding information.



9. Coding Standards

Application of coding standards

- Naming rules
- Student Management
- Course Management
- Binding conditions

10. Project Results

Objective about knowledge

Knowledge	Application
Loops	Use some loops: While, For
Switch Case	Used the Switch Case structure in several methods: • Course selection • Choose payment method
If – Else	Used if-else in design Interface Menu and several mothods.
Scanner	Use Scanner to enter data directly, enter important information • Make choices • Enter student information • Enter payment information
Array List	Used in managing object information in Classes

10. Project Results

Scope of Application Feature

Feature	Level of perfection
Ask the user how many students will be added	• Done
User should be promoted to enter the Student ID, Name, Course, Year for each student. In particular, Student ID is not allowed to be duplicate, Number of years ranges from 1 to 3.	• Make sure the student's year number must be within the ranges from 1 to 3 • Student IDs are not duplicated
A student just can enroll course in the table 1	• Students can only register for courses according to the listed list • Students can only register for 1 to 6 courses.
The student should have a 5-digit unique ID, with the prefix of the year	• Each Student have unique ID • The ID is limited to the range 10000 to 39999
The student should be able to view their balance and prompted to pay the outstanding fee.	• Student can view their balance • User can choose to pay partially or in full.
The student status can be viewed by using their ID.	• Done
Send confirmation email upon enrolment.	• Done

11. Proposed Improvements

a) Data Base

- Student & course information is being entered directly from the keyboard. This **manual** input method proves **to be ineffective** when working with many students.
- Recommendation: You can learn more about **transferring .csv and .xlsx files directly into the application**. **Combined** with **data query languages** such as SQL. More suitable for large data management.

b) Programmability

- **Many sections** are written in **sequential programming** style, making it difficult to edit the code. The proposed solution is to write detailed functions in the relevant class, in more detail in the Class Diagram design.
- Some methods that can be further improved:

- **Apply rules** for naming variables, method names, class names, and package names
- Practice repeatedly to master the **OOP** programming method
- Add **some binding conditions** so that the registration course does not **overlap**
- **Add constraints** to the input data format. For example, the entered email must match the predetermined email format...etc

15. References

- (Oracle, n.d.) https://docs.oracle.com/javase/tutorial/
- (NỘI, 2022) https://niithanoi.edu.vn/ide-phan-mem-lap-trinh-java-tot-nhat.html
- (Overflow, 2011) https://stackoverflow.com/questions/4746341/do-i-need-to-install-java-sdk-if-i-have-eclipse/4746497#4746497
- (Github.io, n.d.) https://javaee.github.io/javamail/
- (geeksforgeeks.org, n.d.) https://www.geeksforgeeks.org/scanner-class-in-java/

THANK YOU