A picture containing text, clipart

Description automatically generated**Project Report**

|  |  |
| --- | --- |
| Module Name | WSQ Programming Foundations (SF) |
| Course Name | Postgraduate Diploma in Software Engineering |
| Assignment Title | Student Management System |

|  |  |  |  |
| --- | --- | --- | --- |
| Learner Name | | Assessor Name | |
| HOÀNG MINH TRƯỜNG | |  | |
| Date Issued | Completion Date | | Submitted On |
| 25/09/2023 | 01/10/2023 | | 01/10/2023 |

**Plagiarism**

Plagiarism covers both copying or paraphrasing with the source without proper understanding and citation. The source includes but is not limited to:

* Books, Journal Articles, Online resources
* Others Assignment or project
* Software program
* Images

The learners must submit their work without plagiarism while producing the evidence against the assessment criteria. They must apply proper citation and Harvard referencing. Their work must go through the Turnitin software to check for the similarity index. Please consult the relevant module instructor or mentor if you need any further advice.

|  |
| --- |
| Learner declaration |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.  Student signature: Hoàng Minh Trường Date: 01/10/2023 |

Table of Contents

[1. Project Background 5](#_Toc147671350)

[2. Project Objective 5](#_Toc147671351)

[2.1 Objective: 5](#_Toc147671352)

[2.2 Feature scope applications: 5](#_Toc147671353)

[3. System description 6](#_Toc147671354)

[4. Environment Preparation 7](#_Toc147671355)

[4.1 IDE 7](#_Toc147671356)

[4.2 Support library. 8](#_Toc147671357)

[4.3 Diagram Support App 8](#_Toc147671358)

[5 Data Structure 9](#_Toc147671359)

[5.1 Over view: 9](#_Toc147671360)

[5.2 Class Course & Student 10](#_Toc147671361)

[5.3 Class StudenList 10](#_Toc147671362)

[6 Activities Diagram 11](#_Toc147671363)

[7 Method 15](#_Toc147671364)

[7.1 Method Add Student 15](#_Toc147671365)

[7.2 Method checkDupbyID(int ID) 15](#_Toc147671366)

[7.3 Add Student 16](#_Toc147671367)

[7.4 Method: findStudentByID(int id) 17](#_Toc147671368)

[7.5 Payment Implementation 18](#_Toc147671369)

[7.6 Method Send Mail 19](#_Toc147671370)

[8 Result 20](#_Toc147671371)

[8.1 Objective about knowledge 20](#_Toc147671372)

[8.2 Scope of Application Feature: 20](#_Toc147671373)

[9 Conclusion and suggestions for improvement. 21](#_Toc147671374)

[9.1 Data Base 21](#_Toc147671375)

[9.2 Programmability 21](#_Toc147671376)

[10 Reference 23](#_Toc147671377)

**Table of tables**

[Table 1:Course List Information 4](#_Toc147669811)

[Table 2: Compare Eclipse vs VS Code 6](#_Toc147669812)

[Table 3: Process Diagram Detail 11](#_Toc147669813)

[Table 4: Objective Knowledge Evaluate 19](#_Toc147669814)

[Table 5: Application Feature Evaluate 19](#_Toc147669815)

**Table of Figures**

[Figure 1 Use Case Diagram 4](#_Toc147669598)

[Figure 2 Javax Mail 6](#_Toc147669599)

[Figure 3 Tool Draw.io 7](#_Toc147669600)

[Figure 4 Overall Class Diagram 8](#_Toc147669601)

[Figure 5: Activities Diagram 9](#_Toc147669602)

[Figure 6: System request user input student's info in turn 14](#_Toc147669603)

[Figure 7: Method Add Student - input ID, Name, No of Course 14](#_Toc147669604)

[Figure 8: Use Switch Case in choosing course 15](#_Toc147669605)

[Figure 9: Payment Process 15](#_Toc147669606)

[Figure 10: Payment Code Block 16](#_Toc147669607)

[Figure 11: Switch Case - Paymethod In Detail 17](#_Toc147669608)

Acknowledgement

I would like to express my sincere gratitude to the following individuals and organizations who helped me with my project:

Dr. NguyenTrung Lap– for providing valuable guidance and feedback throughout the project.

Dr.Ho Kim Giau – for helping me with research and providing support when I needed it.

I would also like to thank EduClaas for supporting my project and giving me the opportunity to learn and grow.

Without the help and support of these individuals and organizations, this project would not have been possible. Thank you all for your time, effort, and contributions.

Sincerely,

Hoang Minh Truong

## Project Background

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.

## Project Objective

### 2.1 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 Class, Methods, and Distribution. 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.

### 2.2 Feature scope applications:

1. Ask the user how many students will be added
2. 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.
3. A student just can enroll in the following courses:

Table 1:Course List Information

|  |  |  |  |
| --- | --- | --- | --- |
| Year | Code | Name | Fees |
| 1 | 101 | Programming Foundations | $ 1200 |
| 1 | 102 | User Interface Design | $ 1800 |
| 2 | 103 | User Experience Design | $ 2000 |
| 2 | 104 | Database Design and Implementation | $ 1500 |
| 3 | 105 | Web Development Foundations | $ 2000 |
| 3 | 106 | Capstone Project | $ 2500 |

1. The student should have a 5-digit unique ID, with the prefix of the year – Don’t Duplicate.
2. The student should be able to view their balance and prompted to pay the outstanding fee.
3. The system should allow the student to pay partially or in full.
4. The student status can be viewed by using their ID.

ID, Name, courses enrolled, total fees, and balance should be displayed.

1. Send confirmation email upon enrolment

## System description

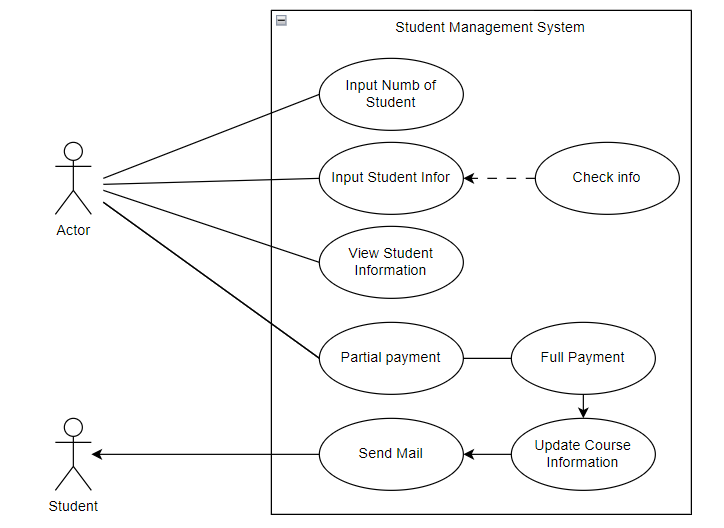


Figure 1 Use Case Diagram

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.

The system includes main functions:

* Enter the number of students.
* Enter student information according to the predefined structure: Will include some rules that apply to student information about ID, course... mentioned in the scope. In case it detects that a student is already on the list or has an incorrect year or course, the system will notify and ask the user to re-enter correctly.
* Search and view student information by ID: Through the student ID, the actor can view information about courses that students register for, personal information and tuition information.
* Choose to pay partially or full of the tuition fee: The system allows the actor to choose one of these two payment types. In case the amount the actor wants to pay does not match the payment conditions, the system will notify then request the user to re-enter the amount.
* The system will send updates to all students in StudentList().

Students are identified by a unique ID number, ensuring data uniqueness. This ID number plays the role of primary key in searching for student information and implementing methods.

## Environment Preparation

### IDE

An Integrated Development Environment (IDE) is an application that fully facilitates programming.

In general, it is designed to help programmers develop applications with an environment and necessary tools.

Some IDEs are recommended for use: Eclipse, InteliJ, Visual Studio Code….

In this project, I will compare and choose between the following two IDEs:

Table 2: Compare Eclipse vs VS Code

|  |  |
| --- | --- |
| **EClipse** | **Visual Studio Code** |
|  |  |
| * Simple, compact, easy to install * Helps you organize and edit code with quick validation, enhanced editing, cross-referencing, and code suggestions. * Integrated static code analysis * Smart code year calculation for code completion and quick editing. * Excellent usability and performance * Many Plugin support available. * • Supports diagramming, modeling, reporting as well as testing | * Simple, compact, easy to install * Intellisense support * Built-in Emmet * Integrate with Git * Very good bug fixing support * Extension can be installed from outside * Always run Terminal in the editor |
| * Can be used directly after installing the software | * Must install Java Extension Pack * Must install JDK |

Basically the above two IDEs are simple and easy to use, however Esclipe is more convenient, so in this project, I chose Eclipse.

**Version: Eclipse IDE 2023 09**

### Support library.

One of the important requirements of the project is the ability to send emails to students. This method requires a platform-independent and protocol-independent framework to build mail and messaging applications. The JavaMail API is available as an optional package for use with the Java SE platform and is also included in the Java EE platform.



Figure 2 Javax Mail

Version: August 29, 2018 - JavaMail 1.6.2 Final Release

### Diagram Support App

During the system analysis and design process, UML design is a mandatory requirement, helping to clearly define classes and methods. In this project, I chose Draw.io to draw UML diagrams for the following reasons:

* 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.

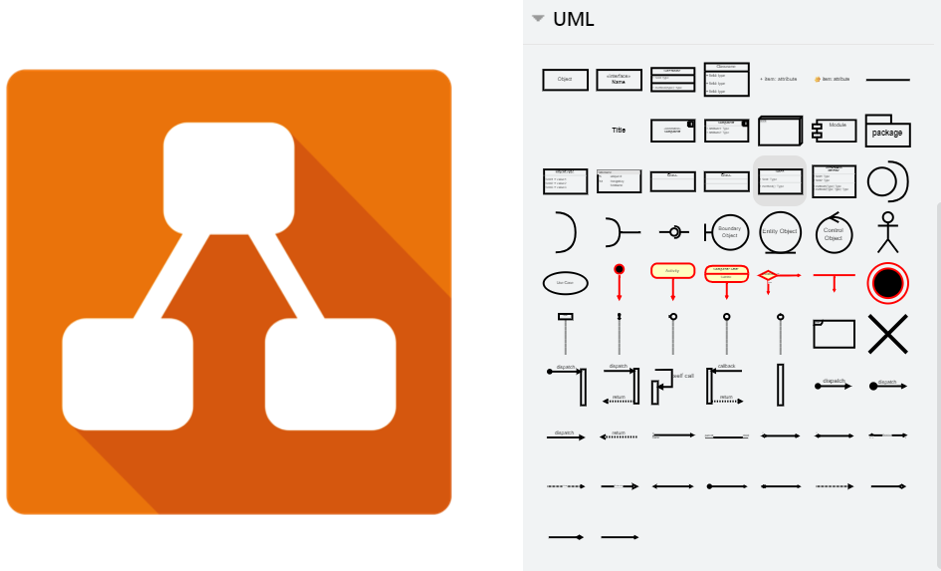


Figure 3 Tool Draw.io

## Data Structure

### Over view:

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

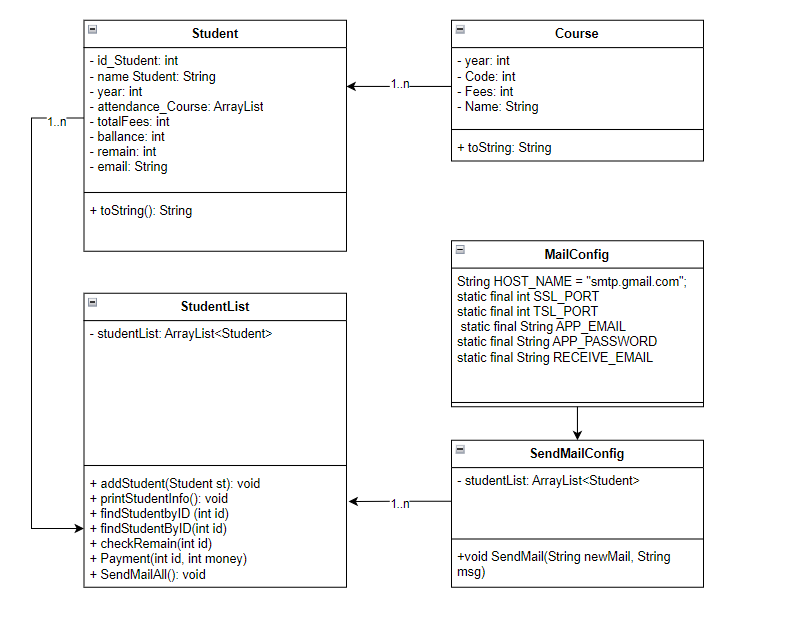


Figure 4 Overall Class Diagram

### Class Course & Student

Contains course information including year, code, fee, course\_name.

Contains basic student information

### Class StudenList

Contains information of the entire student list.

Contains many important methods such as:

+addStudent(Student st):void , checkDupByID(int id): int, checkRemain(int id), Payment(int id, int money), SendMailAll(): void

## Activities Diagram

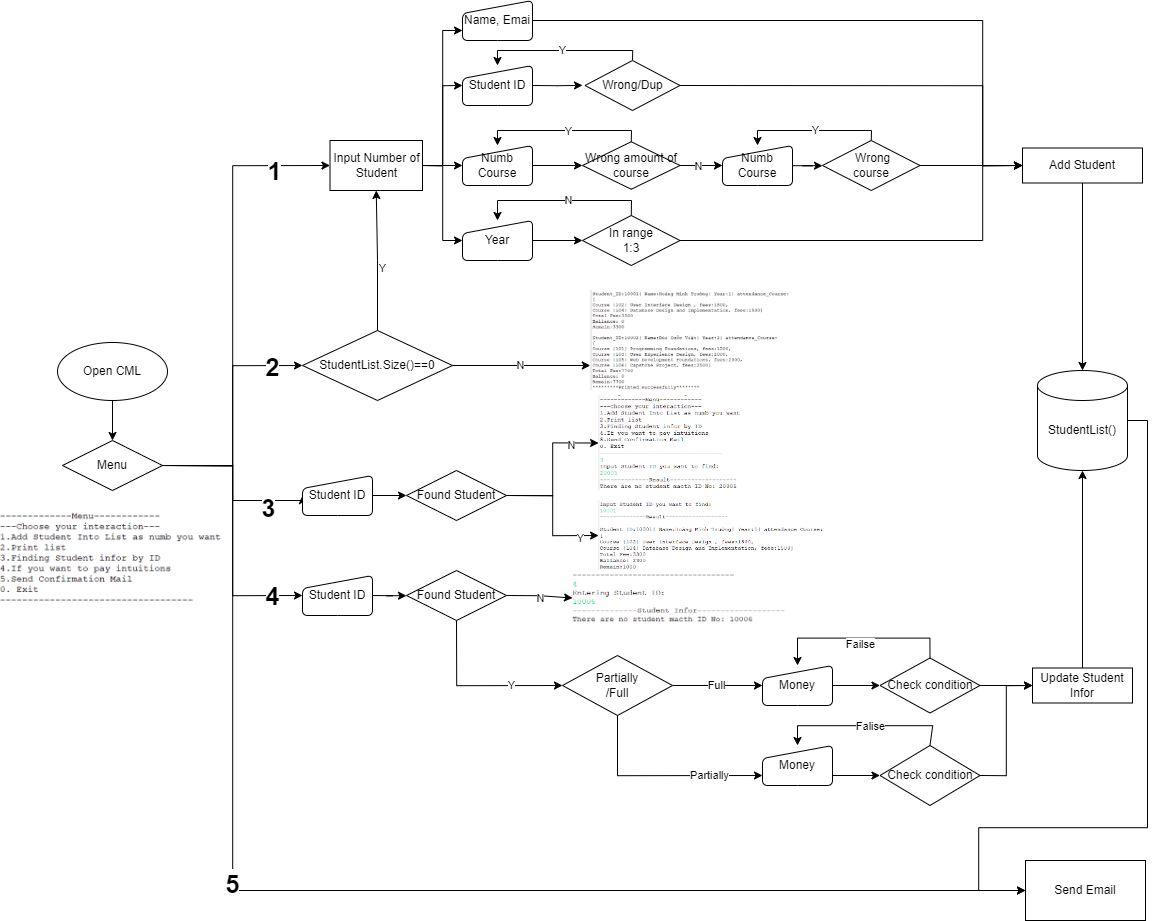


Figure 5: Activities Diagram

Activity Diagram describes activities and processing flows within the system. It can be used to describe business processes in the system, specific processes are described as follows:

Table 3: Process Diagram Detail

|  |  |  |
| --- | --- | --- |
| **Activities** | **UI Page** | **Descriptions** |
| Start/Main loop |  | 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. |
| 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: |
|  | * Student ID is in wrong format * Student ID is duplicated |
|  | * Student Year not in range [1:3] |
|  | * The person entered the wrong course number, course\_id must be in the range [101:106] |
| Check Student List |  | When selecting is 2. The system will print out all current student information. |
| Find Student |  | 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. |
| Payment | ----------------------------------------------------------------- | 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. |
|  |  | 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. |
| Check Student List |  | 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. |
| 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. |
| Result |  | After the actor sends the email. Students who have previously declared their email will receive an email informing them of the corresponding information. |

## Method

Methods are declared within project's class, and that they are used to perform many different actions.

Some key methods that need our attention in the project:

* Method: Add Student
* Method: FindStudentByID(int ID)
* Method: Payment(int ID, int money)
* Method: SendMailAll()

### Method Add Student

Allows users to add students according to a known number. After entering the assigned number, the user takes turns entering student information. The method uses a number of sub-methods to check for duplicate Student\_ID:

### Method checkDupbyID(int ID)

Returns variable is\_Dup in integer data type. If is\_Dup = 1, the system know that id is duplicated.

**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;

}

### Add Student

Users in turn pass parameters of Class student.

Figure 6: System request user input student's info in turn

Method will raise notion whenever have any errors bellow:

* Student\_ID: wrong format (must a 5-digit unique ID, start with year)
* Student\_ID: is duplicated
* Student year: not in range from 1 to 3
* Amount of course is wrong (<0 || >6)

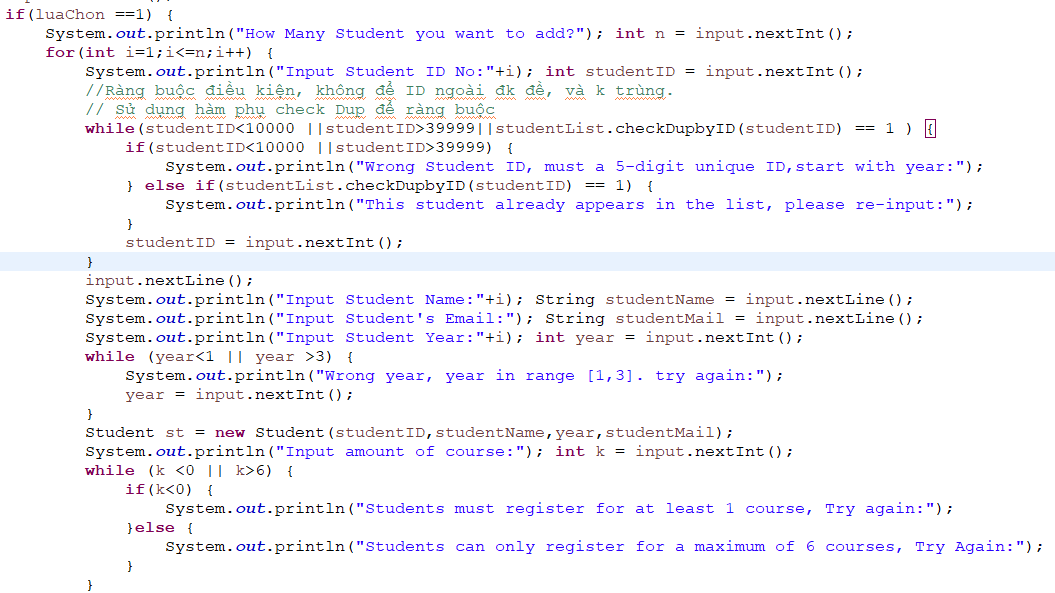


Figure 7: Method Add Student - input ID, Name, No of Course

The method uses a switch case structure for users to enter predefined course information.

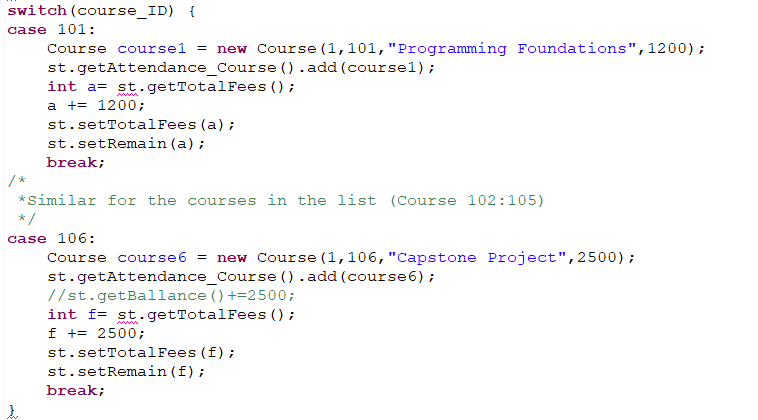


Figure 8: Use Switch Case in choosing course

### Method: findStudentByID(int id)

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) {

StudentList studentList2 = **new** StudentList();

**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);

}

}

### Payment Implementation

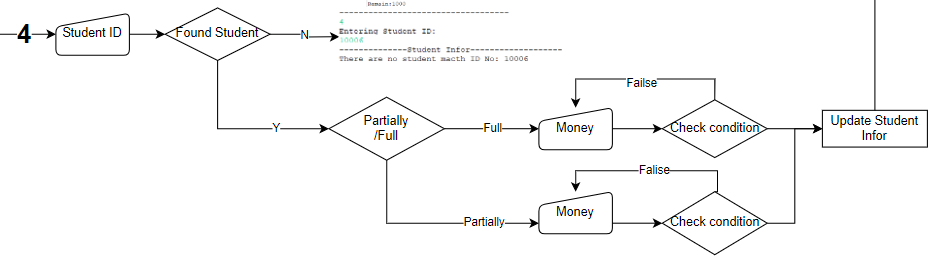


Figure 9: Payment Process

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.

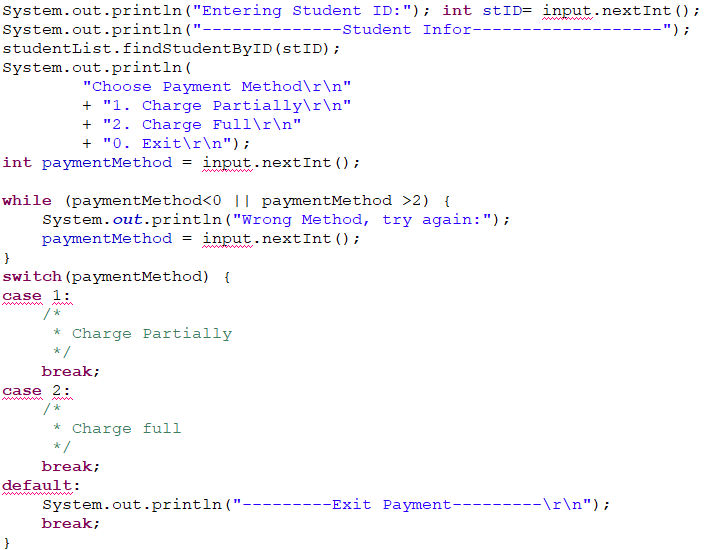


Figure 10: Payment Code Block

In the payment method, there are two sub-methods used:

* CheckRemain(int ID)
* Payment(int ID, int Money)

// 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);

}

}

}

Full payment and partial payment cases are relatively similar, the only difference is the binding conditions for the amount entered in each case.

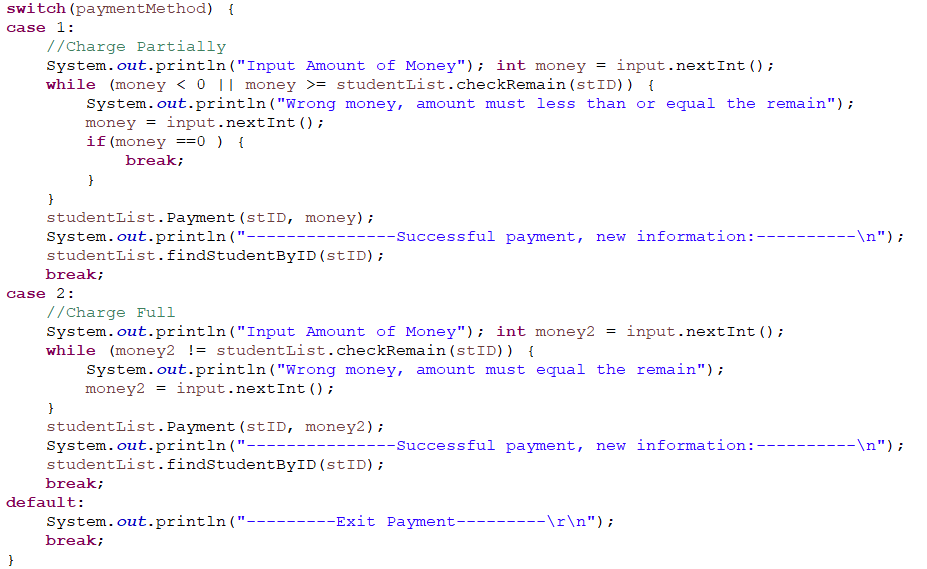


Figure 11: Switch Case - Paymethod In Detail

### Method Send Mail

* + 1. Mail Config

Set up constants related to email information. Because we send from email, we will set SSL port 465

**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;

}

## Result

The project was designed and met the requirements from the beginning of the project:

### Objective about knowledge

Table 4: Objective Knowledge Evaluate

|  |  |
| --- | --- |
| 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 |

### Scope of Application Feature:

All proposed features have met specific requirements:

Table 5: Application Feature Evaluate

|  |  |
| --- | --- |
| **Feature** | **Level of perfection** |
| 1. Ask the user how many students will be added | * Done |
| 1. 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 |
| 1. 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. |
| 1. 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 |
| 1. 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. |
| 1. The student status can be viewed by using their ID. | * Done |
| 1. Send confirmation email upon enrolment. | * Done |

## Conclusion and suggestions for improvement.

Overall, the project has accomplished the goals set out from the beginning. Apply the knowledge learned to create an application that allows users to manage student information, be able to observe an overview of all student information, and thereby make the right decisions.

During the process of running the application, I realized that the system encountered some exceptions, some features and options were proposed to upgrade the project in later stages. Some factors that can be improved:

### 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.

### 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

## Reference

(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/>