NAME: SUDAIS MURAD **REG NO:SP23-BSE-040**

SDA ASSIGNMENGT 2

Student Fee Management System - Documentation

Project Title:

Process Payment

Project Overview:

This Java Swing-based GUI application allows accountants to process student fee payments, validate details, select a payment method, and generate a voucher receipt. It implements the fully dressed use case **"Process Payment (UC-003)"** by covering all preconditions, postconditions, and scenarios (like partial payment and invalid payment). The system uses Object-Oriented Programming concepts and GRASP principles for clean, modular design.

🖥️ Technologies Used:

- Java (JDK 17)
- Java Swing (for GUI)
- Java SE API (HashMap, LocalDate, ActionListeners)
- NetBeans or IntelliJ (IDE for development)

✅ Functional Requirements:

Use Case	Description
Process payment	Student enters ID, name, email, and amount paid. System processes payment and generates a voucher.
Search by ID	Search student records by ID to view payment status and details.
Select Payment Method	Accountant selects payment method (Bank, Cash, or Online).
Print/Download Receipt	System provides receipt in printable and downloadable format.
Login Access	System allows only authorized users (username: `admin`, password: `admin123`).

Non-Functional Requirements:

- Validates inputs (like ID and amount paid).
- Provides user-friendly and interactive GUI.
- Displays payment status and remaining fee instantly.
- Generates vouchers with unique IDs.
- GUI is responsive and logically arranged using `setBounds()`.

Class Overview:

Class Name	Responsibility
`FeeCalculator`	Returns the fixed total fee (60000).
`Payment`	Handles amount, method, calculates remaining amount and payment status.
`PaymentProcessor`	Receives payment and logs it to the console.
`VoucherGenerator`	Generates unique voucher using timestamp and student ID.
`Student`	Stores student information along with their payment object.
`StudentFeeController`	Processes payment, manages student records, and supports search by ID.
`Main`	Controls GUI flow, login authentication, and connects all components together.

Forward Engineering - Class Design for Code Generation:

Class Name	Key Attributes / Methods
`FeeCalculator`	`getTotalFee()`
`Payment`	`getRemainingAmount()`, `getStatusWithDueDate()`, `getMethod()`, etc.
`VoucherGenerator`	`generateVoucher()`
`StudentFeeController`	`processStudentFee()`, `searchStudentById()`
`Main`	GUI event handling, login page, button actions (`Submit`, `Print`, `Search`)

GUI Components (Visual Layout):

Component	Description
`JTextField`	For Student ID, Name, Email, Amount Paid
`JComboBox`	Dropdown to select payment method (Bank, Cash, Online)
`JButton`	Buttons for Login, Submit Payment, Print Receipt, Download, Search
`JTextArea`	Output box to display payment summary and messages
`JFrame` & `JPanel`	Windows and layout containers

GRASP Principles Applied:

GRASP Principle	Class / Justification
Controller	`Main` handles overall application flow and user interactions
Information Expert	`Payment` and `FeeCalculator` calculate payment status and fee details
Creator	`StudentFeeController` creates and stores `Student` and `Payment` objects
Low Coupling	Each class does one task and interacts via method calls
High Cohesion	Each class has a single responsibility (cohesion is maintained)

Design Pattern Used:

Pattern	Implementation in Code
Strategy Pattern	Different classes (`Payment`, `VoucherGenerator`, `FeeCalculator`) have distinct roles.
Future Ideas	You can later apply:

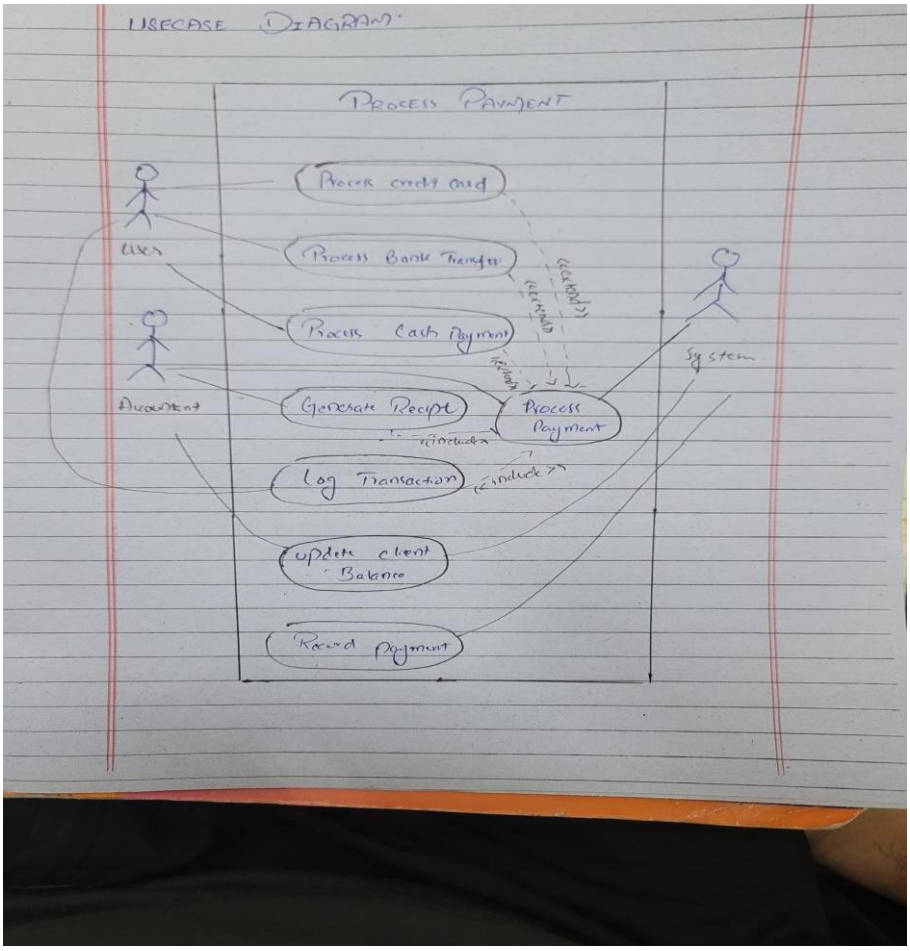
- Factory Pattern (to create various types of payments)
- Observer Pattern (for notifications)
- Decorator Pattern (for discounts, late fees)

Additional Diagrams

Use Case Diagram:

A use case diagram represents the interaction between the user (student) and the system. It includes the primary actor and their goals in using the system. In this system, the Student actor interacts with the system to perform:

- Submit Payment
- Issue Refund
- View Payment Status



Fully Dressed Use Case

Process Payment

Attribute	Details
Use Case Name	Process Payment
Use Case ID	UC-003
Primary Actor	Accountant, System (Automated)
Secondary Actors	Payment Gateway, User (Client)

Description	Records and validates a payment against an invoice, updates the client's balance, and generates a receipt.
Preconditions	1. Invoice exists and is approved. 2. Client account is active. 3. Payment details are valid.
Postconditions	1. Payment is recorded. 2. Client balance is updated. 3. Receipt is generated (if applicable).
Trigger	Accountant initiates payment or System detects a scheduled payment.
Main Success Scenario (Basic Flow)	1. Accountant/System selects the invoice for payment. 2. System validates invoice status and amount. 3. User/Payment Gateway provides payment details (credit card, bank transfer, etc.). 4. System processes payment and receives confirmation. 5. System records payment in the database. 6. System updates client balance. 7. System generates a receipt (optional). 8. Payment is marked as "Completed."
Alternative Flows	**E1: Invalid Payment** - 3a. Payment fails (e.g., insufficient funds). - 3b. System notifies Accountant/User and retries or cancels. **E2: Partial Payment** - 3a. User pays a partial amount. - 3b. System updates invoice status to "Partially Paid."

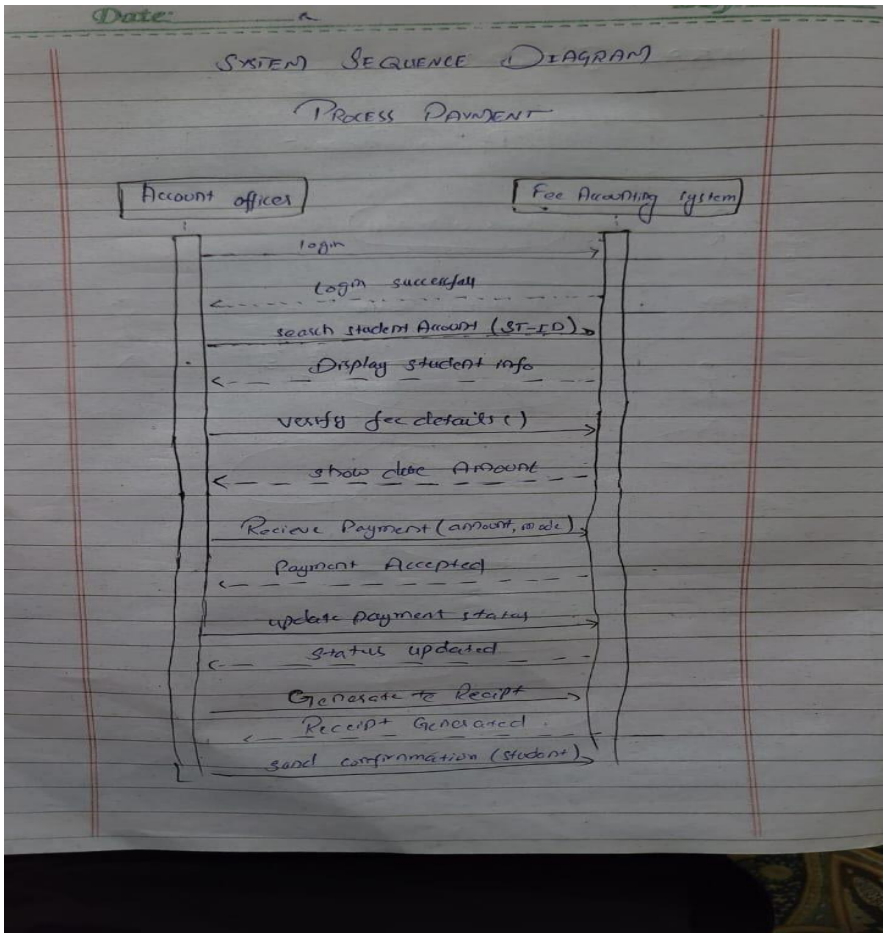
Attribute	Details
	E3: Offline Payment (Cash/Check) - 3a. Accountant manually marks payment as "Pending." - 3b. System updates status after manual verification.
Business Rules	1. Payments must match the invoice amount (± tolerance of 1%). 2. Overdue invoices may incur late fees. 3. Digital payments require gateway authentication.
Non-Functional Requirements	1. Payment processing time < 2 seconds. 2. PCI-DSS compliance for card payments. 3. Audit logs for all transactions.
Assumptions	1. Payment Gateway is always available. 2. Currency conversion is handled externally.

-

System Sequence Diagram :

A system sequence diagram shows the sequence of messages exchanged between the actor and the system. It covers one main use case.

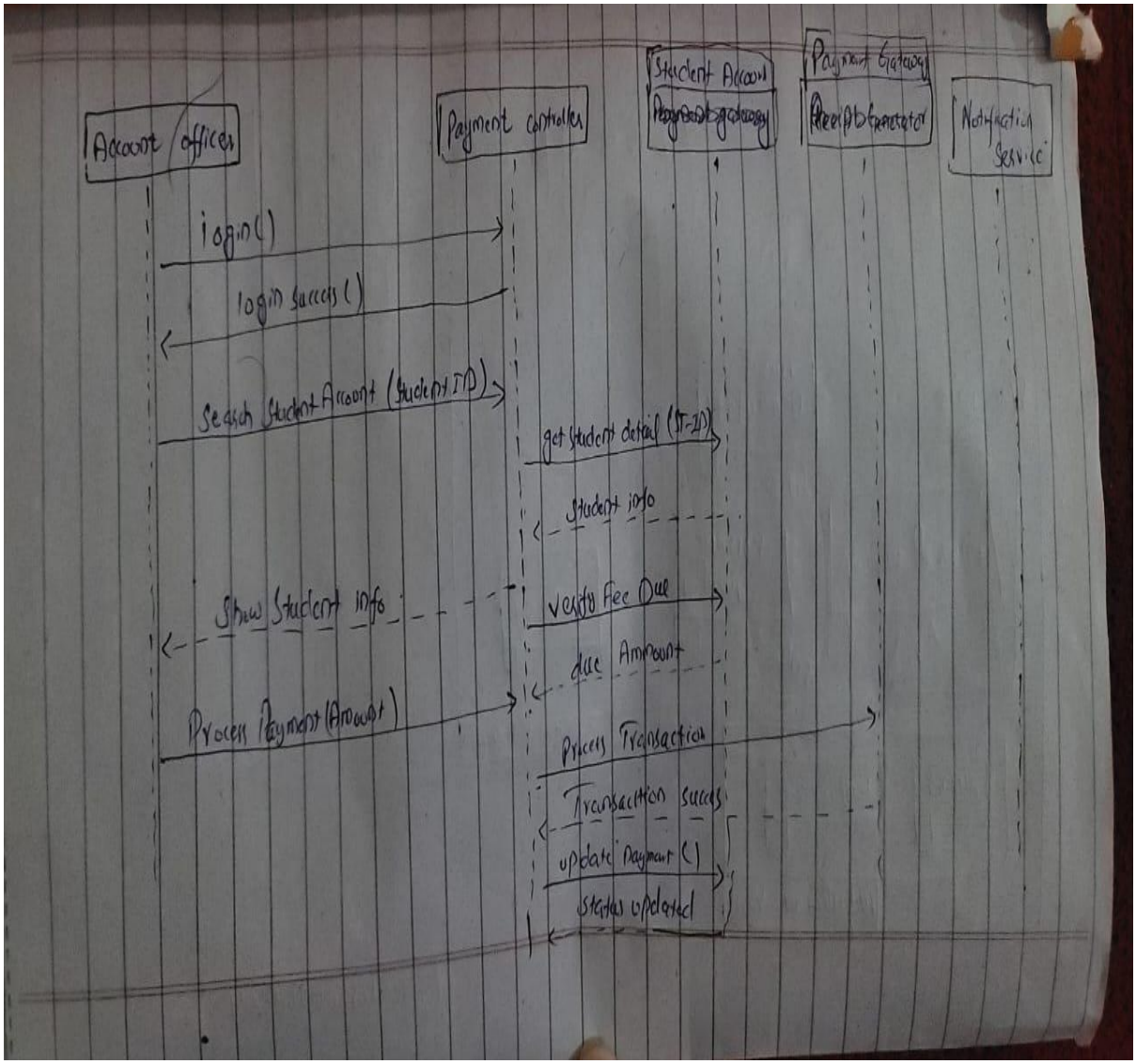
- The student fills the form in GUI
- GUI sends data to the system for processing
- The system responds with calculated results and voucher



Sequence Diagram :

A sequence diagram provides a more detailed view of object interactions over time:

1. Student interacts with GUI
2. GUI calls Main class methods
3. Main creates Payment and FeeCalculator objects
4. Calls relevant methods and displays results



Package Diagram :

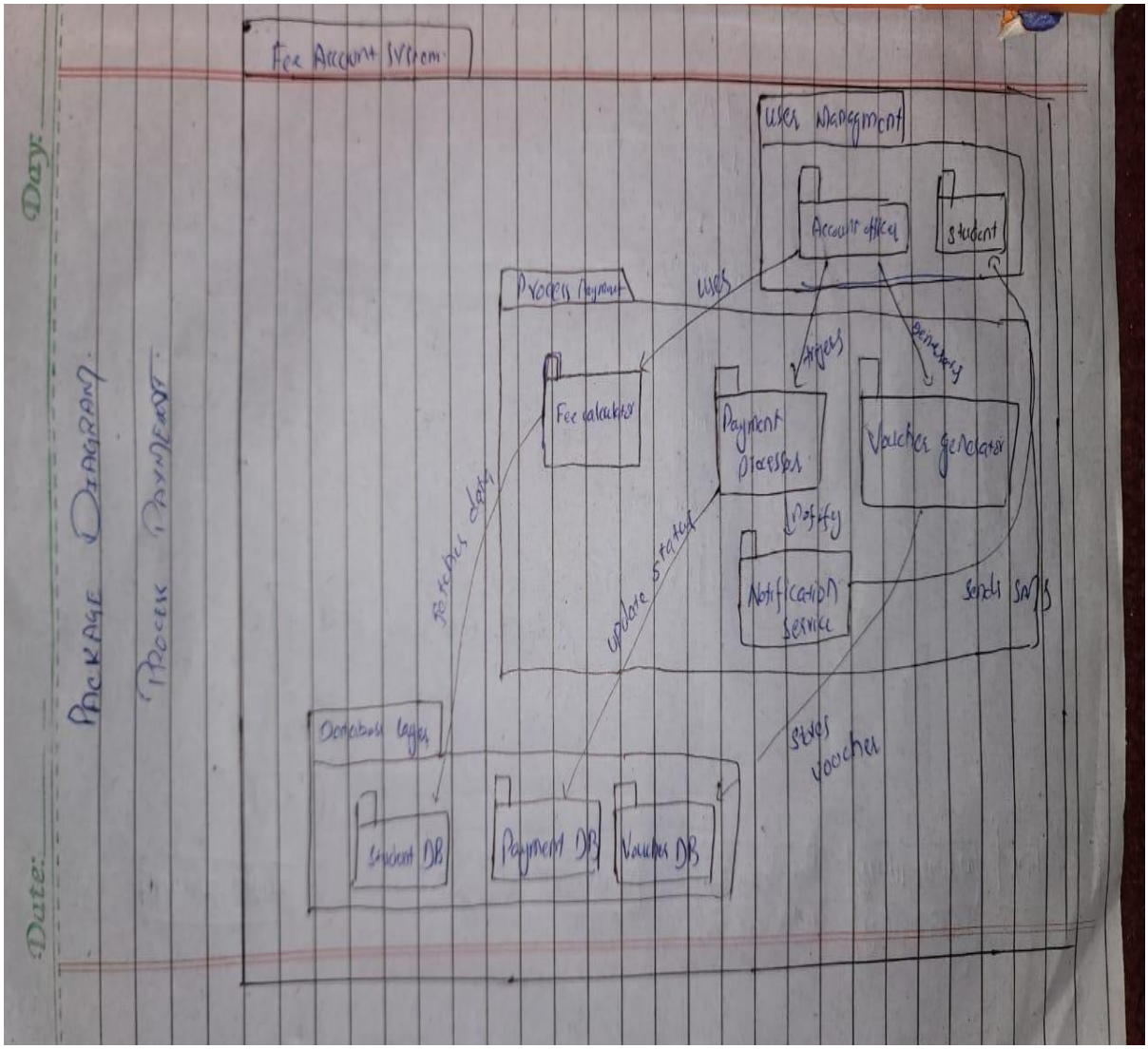
A package diagram shows how classes are grouped. Here, all classes (`Main`, `Payment`, `FeeCalculator`, `VoucherGenerator`) belong to the same package `sdalabassignment2`. It visually represents organization and dependency of class files within the project.

Day:

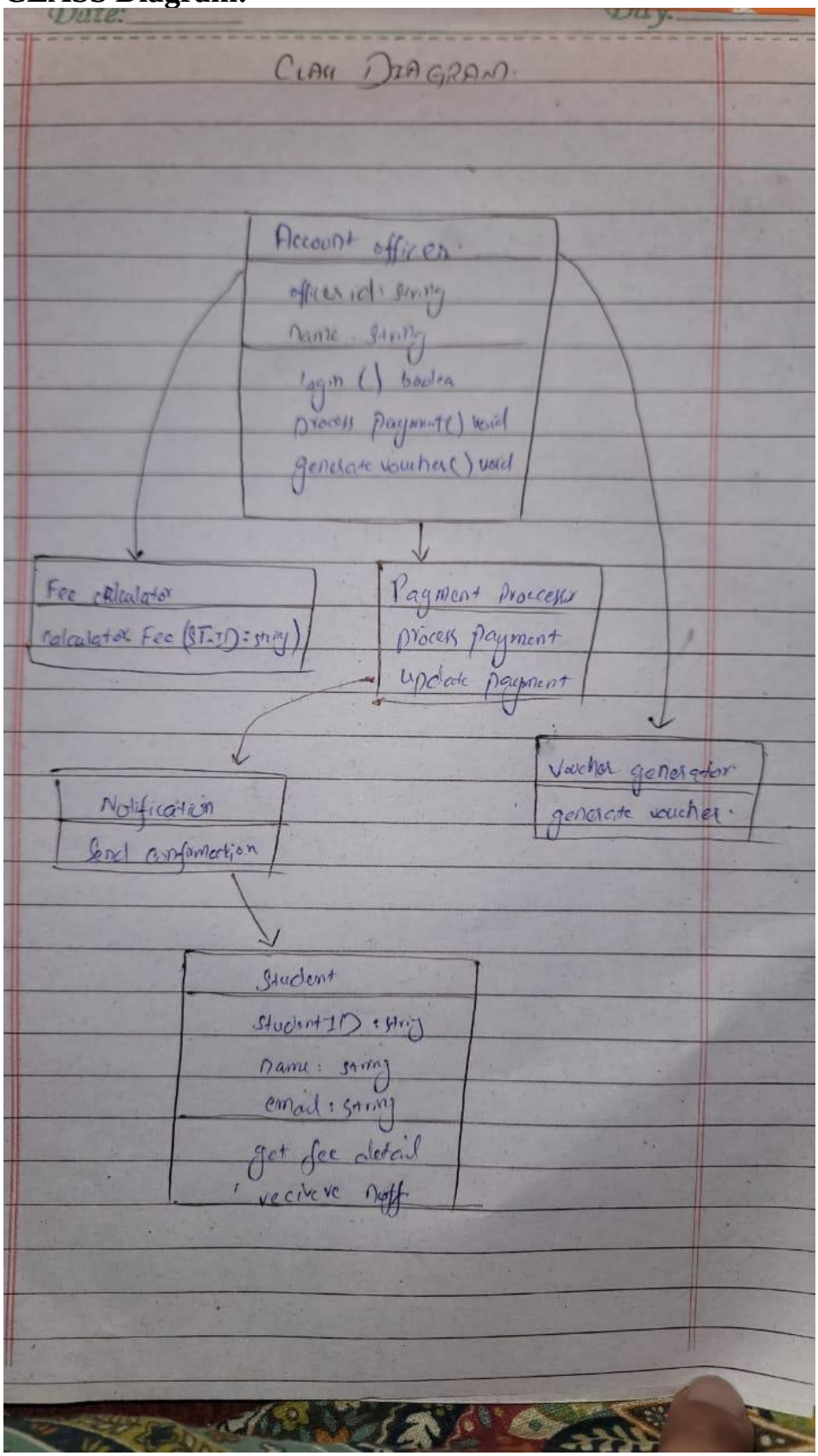
Date:

PACKAGE DIAGRAM.

(BLOCK DIAGRAM)



CLASS Diagram:



Future Scope:

- Add file or database saving of payment records.
- Implement login system for admin/students.
- Send receipts via email using SMTP.
- Export to PDF or CSV.

How to Run:

1. Open the project in NetBeans/IntelliJ
2. Run `Main.java`
3. Enter student info, payment amount, and press Submit.
4. If overpaid, enter refund amount and press Issue Refund.