Student Management & Payment System

Detailed Class List

1 Person (Base Class)

Attributes:

- name
- email
- phone
- address

Methods:

- is_valid_email() — checks email format validity
- is_valid_phone() — validates 10-digit phone number
- update_contact(email=None, phone=None, address=None) — updates contact info
- __str__() — string representation

Purpose:

The base class for all human entities in the system (Students, Faculty).
It ensures consistent personal and contact information handling.

2 Student (inherits Person)

Attributes:

- student_id
- department
- year_of_study
- enrollments : list of Enrollment
- cgpa
- attendance : dict of course_code → attended_count
- account : BankAccount (optional link)
- date_joined

Methods:

- enroll(course) — enrolls in a course and creates an Enrollment object
- mark_attendance(course_code, present=True) — tracks attendance
- update_cgpa(new_grade_point) — recalculates CGPA dynamically
- calculate_total_fee() — sums up all enrolled course fees
- display_enrollments() — prints enrolled courses & payment status
- link_bank_account(account) — associates a BankAccount object

- __str__() — displays student profile

Purpose:

Models real-world student behavior including enrollment, academic progress, and payment.
Serves as a central entity linking multiple system components (Department, Enrollment, Bank, Payment).

3 Faculty (inherits Person)

Attributes:

- emp_id
- subject
- designation
- salary
- courses_taught : list of Course

Methods:

- assign_course(course) — links a faculty to a course
- calculate_bonus(percentage) — computes performance-based bonus
- __str__() — formatted faculty info

Purpose:

Represents teaching staff.
Used for assigning instructors to courses and tracking subject expertise.

4 Department

Attributes:

- dept_code
- name
- hod : Faculty
- courses : list of Course
- students : list of Student

Methods:

- add_course(course) — adds a course under this department
- add_student(student) — registers a student
- remove_student(student_id) — removes a student by ID
- display_info() — prints department summary

Purpose:

Acts as an aggregation root connecting courses, students, and faculty under one academic unit.

5 Course

Attributes:

- course_code
- course_name
- credits
- fee
- faculty : Faculty
- max_seats
- enrolled_students

Methods:

- seat_available() — checks if seats are open
- register_student() — increments seat count if available
- __str__() — displays course details

Purpose:

Models academic courses with credits, fee, and instructor.
Used for enrollment, billing, and progress tracking.

6 Enrollment

Attributes:

- student : Student
- course : Course
- enroll_date
- payment_status
- grade

Methods:

- mark_paid() — marks fee as paid
- assign_grade(grade) — updates student's CGPA and stores grade
- __str__() — readable enrollment info

Purpose:

A composition relationship that ties a Student and a Course.
Tracks payment and academic progress at the course level.

7 BankAccount (Encapsulation Demo)

Attributes:

- owner

- __balance *(private)*

Methods:

- balance *(getter)* — retrieves balance
- balance(amount) *(setter)* — updates balance with validation
- deposit(amount) — adds funds
- withdraw(amount) — subtracts funds with check
- transfer_to(other_account, amount) — inter-account transfer
- __str__() — formatted bank summary

Purpose:

Encapsulation in action — hides sensitive financial data.
Used by students or system accounts to simulate real payment logic.

8 PaymentGateway (Abstract Class)

Attributes:

- *(abstract, no fields)*

Methods:

- process_payment(enrollment, amount) *(abstract)* — to be overridden

Purpose:

Defines the structure for all payment mechanisms.
Ensures uniform interface for processing student payments.

9 UPIPayment (inherits PaymentGateway)

Attributes:

- transaction_id *(generated dynamically)*

Methods:

- process_payment(enrollment, amount) — simulates a UPI transaction and marks enrollment as paid

Purpose:

Concrete payment gateway for digital UPI transactions.
Demonstrates polymorphism by implementing process_payment.

10 CardPayment (inherits PaymentGateway)

Attributes:

- card_last4
- transaction_id

Methods:

- process_payment(enrollment, amount, card_last4) — simulates a card charge

Purpose:

Implements abstract process_payment() with credit/debit card context.
Showcases polymorphism and method overriding.

1️⃣1️⃣ Logger (Utility Class)

Attributes:

- log_count *(class variable)*
- logs *(list)*

Methods:

- format_log(msg) *(static)* — formats with timestamp
- log(msg) *(class)* — appends to log and increments count
- dump_logs(filepath) — writes logs to file

Purpose:

A supporting utility demonstrating static and class methods.
Used to record system actions, events, and flow.

Relationships Overview

Relationship	Description
Person → Student / Faculty	Inheritance (is-a)
Department → Course / Student	Aggregation (has-a)
Student → Enrollment → Course	Composition (strong ownership)
Student → BankAccount	Association (optional link)
PaymentGateway → UPIPayment / CardPayment	Abstraction & Polymorphism
Logger	Independent utility used system-wide

OOPs Concepts Illustrated

Concept	Demonstrated In	Example
Class & Object	Student, Course, Department	Creating student and course objects
Encapsulation	BankAccount	Hidden balance with getters/setters
Inheritance	Student, Faculty ← Person	Common properties
Polymorphism	PaymentGateway subclasses	process_payment() overriding
Abstraction	PaymentGateway	ABC enforcing method definition
Static/Class Methods	Logger	Global log counter
Composition	Student ↔ Enrollment	Tight link lifecycle
Aggregation	Department ↔ Course	Loose ownership
Exception Handling	BankAccount, Payment	ValueError handling

Real-Time Flow Example

1. Student “Dinesh” created under CSE Department.
2. Faculty “Dr. Meena” assigned as course instructor.
3. Dinesh enrolls in “Python Programming”.
4. Enrollment created with payment pending.
5. Payment processed via UPI → status updated to Paid.
6. Grade assigned → CGPA auto-updated.
7. Logger records all activities → export to log file.