Abstract Class

☑ 1. **Online Payment System**

Scenario: You are building a digital payment system that supports multiple payment modes like **Credit Card**, **Debit Card**, **UPI**, and **Net Banking**.

Problem Statement: Create an abstract class `PaymentMethod` with pure virtual functions like:

- `authenticateUser()`
- `processPayment()`
- `generateReceipt()`

Implement concrete classes for each payment method and simulate making a payment using abstraction.

☑ 2. **Transport Booking System**

Scenario: A travel booking app offers different transportation modes: **Bus**, **Train**, and **Flight**.

Problem Statement: Design an abstract class `Transport` with virtual methods like:

- `bookTicket()`
- `getDepartureTime()`
- `getSeatAvailability()`

Each mode of transport implements its own logic. Use polymorphism to show how booking differs by transport type.

☑ 3. **Restaurant Ordering System**

Scenario: A restaurant serves **Dine-in**, **Takeaway**, and **Online Delivery** orders.

Problem Statement: Define an abstract class `Order` with functions like:

- `selectItems()`
- `placeOrder()`
- `generateBill()`

Create separate classes for each order type. Use abstraction to allow the customer to interact only with the interface, not the backend processes.

☑ 4. **Smart Home Device Controller**

Scenario: A smart home supports different devices like **Smart Light**, **Smart Thermostat**, and **Smart Door Lock**.

Problem Statement: Create an abstract class `SmartDevice` with virtual functions:

- `turnOn()`
- `turnOff()`
- `statusReport()`

Each device has unique implementations. Write a controller that manages devices through the abstract interface.

☑ 5. **University Enrollment System**

Scenario: Different types of students (Undergraduate, Graduate, PhD) enroll in courses.

Problem Statement: Define an abstract class `Student` with methods like:

- `enrollCourse()`
- `submitAssignment()`
- `viewGrades()`

Each student type implements course enrollment and grading rules differently. Demonstrate the system using abstraction and polymorphism.
