

42 PYTHON OOP BOSS CHALLENGE
PAYMENT SYSTEM EDITION

* PYTHON OOP PROJECT *
* PAYMENT SYSTEM EDITION *

This subject follows the 42 style and focuses on designing a complete payment-processing system using Object-Oriented Programming principles.

Your mission: implement a scalable, extensible PAYMENT ENGINE with:

- ✓ Multiple payment methods
- ✓ Payment validation rules
- ✓ Exception handling
- ✓ Transaction logs
- ✓ Strategy design pattern
- ✓ Optional fraud detection system

=====

=====

1. PROJECT DESCRIPTION

=====

=====

You must implement a complete OOP payment processing module used by an online platform. The system must process different types of payments dynamically and securely.

Your code must demonstrate:

- Inheritance
- Polymorphism
- Encapsulation
- Abstract design (Strategy Pattern)
- Exception hierarchy
- Clean architectural separation

NO external libraries allowed (except json for saving logs).

=====

=====

2. MANDATORY PART

=====

=====

You must implement the following classes:

A. PaymentStrategy (ABSTRACT CLASS)

Defines the interface all payment methods must implement:

- validate()
- execute()
- generate_receipt()

You CANNOT instantiate this class.

B. Concrete Payment Methods

1. CreditCardPayment

Requires:

- card_number
- card_holder
- expiration_date
- cvv

Validations to implement:

- card number must be 16 digits
- expiration must not be in the past
- cvv must be 3 digits

2. PayPalPayment

Requires:

- email
- password/token

Validations to implement:

- email must contain "@"
- password length >= 6

3. CryptoPayment

Requires:

- wallet_address
- network (ex: "BTC", "ETH")

Validations to implement:

- wallet address must be alphanumeric
- network must be supported (BTC, ETH, USDT)

C. PaymentProcessor

Handles:

- selecting payment method
- running validation
- execution
- error handling
- logging

Methods:

- set_strategy(payment_strategy)

- process(amount)
- write_log(transaction_data)

D. Exceptions

You must implement:

- PaymentError (base)
- ValidationError
- PaymentExecutionError

E. Logging

Every transaction must be logged into a JSON file:
logs/payment_logs.json

The file must contain:

- timestamp
- payment method
- amount
- status (success/failure)
- message

3. BONUS PART

✓ Fraud Detection Module

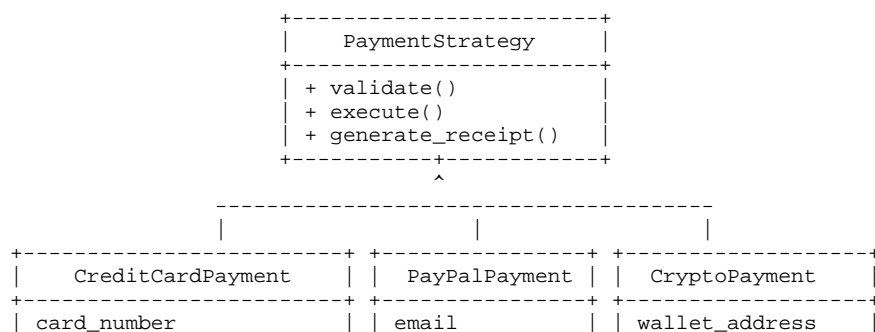
- Flags transactions > 2000\$
- Flags repeated failed transactions
- Admin can review fraud log

✓ Retry Mechanism (max 3 attempts)

✓ ASCII payment summary screen

✓ Unit-test style test suite (your own implementation)

4. ASCII UML DIAGRAM



expiration	password	network
cvv		

PaymentProcessor
strategy
process(amount)
write_log(data)

5. SUBMISSION REQUIREMENTS

Your repo must contain:

- payment_strategy.py
- credit_card_payment.py
- paypal_payment.py
- crypto_payment.py
- payment_processor.py
- exceptions.py
- main.py (demo program)
- logs/payment_logs.json
- README.md (explaining architecture)

Your program must run:
python3 main.py

Good luck, Cadet.
Make it modular, elegant, and worthy of a 42 project.