**SOLID và Design Patterns** là những nguyên tắc và mẫu thiết kế giúp xây dựng phần mềm có cấu trúc tốt, dễ bảo trì, mở rộng và kiểm thử.

from abc import ABC, abstractmethod

class Notification(ABC):

@abstractmethod

def send(self, message):

pass

class EmailNotification(Notification):

def send(self, message):

print("Sending Email: " + message)

class SMSNotification(Notification):

def send(self, message):

print("Sending SMS: " + message)

class Order:

def \_\_init\_\_(self, items, notification: Notification):

self.items = items

self.notification = notification

def confirm\_order(self):

# Logic xác nhận đơn hàng

self.notification.send("Your order has been confirmed!")

# Sử dụng:

email\_notifier = EmailNotification()

order1 = Order(items=[], notification=email\_notifier)

order1.confirm\_order() # In ra: Sending Email: Your order has been confirmed!

sms\_notifier = SMSNotification()

order2 = Order(items=[], notification=sms\_notifier)

order2.confirm\_order() # In ra: Sending SMS: Your order has been confirmed!

*Giải thích:*  
Class Order không phụ thuộc vào một lớp thông báo cụ thể mà chỉ dựa vào giao diện Notification. Nhờ đó, ta có thể dễ dàng thay đổi cách gửi thông báo mà không cần sửa đổi logic của Order.

**Kết luận**

Việc áp dụng các nguyên tắc SOLID trong phát triển phần mềm giúp:

* **Dễ dàng thay đổi và mở rộng:** Mỗi class chỉ chịu trách nhiệm một nhiệm vụ
* **Giảm thiểu lỗi:** Các thay đổi ở một module không ảnh hưởng đến các module khác.
* **Linh hoạt:** Dễ dàng thêm tính năng mới mà không cần sửa đổi code cũ.