Đánh giá mã nguồn theo nguyên lý SOLID

1. Single Responsibility Principle

Related modules	Description	Improvement Direction
Class Cart	Lớp vừa quản lý instance, vừa quản lý các item Các phương thức quản lý cart và instance không liên quan đến nhau	Phân thành 2 lớp CartManager (quản lý singleton) Cart (quản lý cart)
Class BaseController	Lớp BaseController không hoàn toàn là "base" theo nghĩa thuần túy của một lớp cha dùng chung. Nó chứa logic cụ thể liên quan đến giỏ hàng, không phù hợp nếu có các controller khác không liên quan đến giỏ hàng.	Tách các phương thức liên quan đến giỏ hàng (e.g., checkMediaInCart, getListCartMedia) vào một lớp tiện ích (e.g., CartService). Giữ BaseController thuần túy như một lớp cha để các controller khác kế thừa nếu cần chia sẻ chung các chức năng như logging hoặc xử lý request cơ bản.
Class PlaceOrderController	Lớp đảm nhận quá nhiều trách nhiệm, vi phạm nguyên lý SRP: Kiểm tra tính khả dụng của sản phẩm. Tạo đơn hàng (createOrder) và hóa đơn (createInvoice). Xử lý và xác thực thông tin giao hàng. Tính phí vận chuyển.	Tạo các lớp hoặc dịch vụ riêng như: OrderService để quản lý logic đặt hàng. InvoiceService để quản lý logic hóa đơn. ShippingService để xử lý tính phí vận chuyển. DeliveryInfoValidator để xác thực thông tin giao hàng. Controller: Chỉ nên điều phối luồng xử lý và gọi các dịch vụ trên.
Class PaymentController	1, Xử lý quá nhiều trách nhiệm: Điều phối luồng thanh toán (phương thức payOrder). Lưu kết quả giao dịch vào cơ sở dữ liệu (onTransactionCompleted). Làm trống giỏ hàng (emptyCart). 2, Controller tự tạo đối tượng VnPaySubsystemControllertrách nhiệm này không cần thiết, sẽ làm giảm khả năng mở rộng.	1, Tạo thêm các dịch vụ cụ thể như: PaymentService: Chịu trách nhiệm quản lý logic thanh toán. CartService: Quản lý logic liên quan đến giỏ hàng, bao gồm làm trống giỏ. TransactionService: Chịu trách nhiệm lưu giao dịch vào cơ sở dữ liệu. 2, Inject VnPaySubsystemController hoặc IPayment vào constructor. Controller chỉ nên điều phối logic mà không chứa các xử lý chi tiết.

2. Open-Closed Principle		
Related modules	Description	Improvement Direction
BaseController	BaseController hiện phụ thuộc trực tiếp vào lớp Cart. Nếu cần thay thế hoặc mở rộng sang các hệ thống giỏ hàng khác, việc này yêu cầu sửa đổi logic trong BaseController.	Tách biệt lớp Cart khỏi BaseController bằng cách sử dụng interface
HomeController	getAllMedia đang gọi trực tiếp phương thức của Media. Nếu sau này cần thêm các phương thức lọc hoặc xử lý đặc thù cho từng loại media (ví dụ: lấy chỉ Book hoặc DVD), sẽ cần sửa đổi logic trong HomeController	Tạo một service hoặc DAO để quản lý logic truy vấn, tách biệt việc lấy dữ liệu từ lớp HomeController
ViewCartController	ViewCartController phụ thuộc trực tiếp vào lớp Cart qua các phương thức tĩnh (getCart).	Tách biệt lớp Cart khỏi BaseController bằng cách sử dụng interface
DBConnection	DBConnection chỉ hỗ trợ SQLite với cấu hình cố định. Nếu cần hỗ trợ nhiều loại cơ sở dữ liệu (MySQL, PostgreSQL), phải thay đổi logic trong lớp này. Chuỗi kết nối (connectionString) được viết cứng trong mã, không dễ cấu hình.	Trừu tượng hóa logic kết nối cơ sở dữ liệu bằng cách sử dụng một interface
Media, Book, CD, DVD	Phương thức getMediaById và getAllMedia chứa logic truy vấn SQL trực tiếp trong lớp, khiến việc mở rộng cách thức truy vấn hoặc thay đổi cấu trúc dữ liệu khó khăn.	Trích xuất logic truy vấn cơ sở dữ liệu ra một lớp DAO (Data Access Object)
Order	Phương thức getAmount có logic tính toán trực tiếp trong lớp. Nếu cần thay đổi cách tính (ví dụ: thêm chiết khấu, phí dịch vụ), phải sửa đổi mã trong lớp này. Dữ liệu giao hàng (deliveryInfo) sử dụng HashMap, khiến việc mở rộng thêm các trường thông tin cụ thể (vd: số điện thoại, mã bưu chính) khó khăn.	Tách logic tính toán tổng số tiền (getAmount) sang một lớp service riêng để dễ mở rộng Thay thế HashMap trong deliveryInfo bằng một lớp cụ thể

PaymentTransaction	Logic truy vấn SQL nằm trực tiếp trong lớp, khiến việc thay đổi hoặc mở rộng cách lưu trữ dữ liệu (vd: chuyển từ SQL sang NoSQL) phải sửa đổi mã trong lớp này. Phương thức save sử dụng câu truy vấn SQL cứng, và một lỗi xảy ra trong truy vấn có thể làm ảnh hưởng đến lớp.	Trích xuất logic truy vấn dữ liệu ra một lớp DAO (Data Access Object) riêng
Request và VnPaySubsystemController	Một số logic trùng lặp giữa Request và VnPaySubsystemController (vd: xây dựng URL thanh toán). Điều này khiến việc sửa đổi logic khó khăn vì cần thay đổi ở nhiều nơi.	Sử dụng một lớp builder để xây dựng URL hoặc xử lý phản hồi một cách độc lập
API	Chưa hoàn toàn tuân thủ, vì logic xử lý HTTP lặp lại trong các phương thức get và post.	Tạo một phương thức riêng để thiết lập và gửi yêu cầu HTTP, dùng chung cho cả get và post
Configi	Lớp Configs không tách biệt các loại cấu hình	Tách lớp Configs thành các nhóm cấu hình cụ thể hơn
Utils	Lớp chứa các phương thức tiện ích hỗn hợp mà không được tổ chức theo nhóm chức năng rõ ràng (vd: xử lý chuỗi, ngày giờ, mã hóa). Nếu cần thêm nhiều loại tiện ích mới, lớp có thể trở nên quá tải và vi phạm OCP.	Tách các tiện ích thành các lớp nhỏ hơn, chuyên biệt hơn
SplashForm	Đường dẫn hình ảnh logo (src/main/resources/...) được mã hóa cứng trong phương thức initialize. Nếu cần thay đổi vị trí hoặc quản lý hình ảnh khác, phải chỉnh sửa trực tiếp lớp, vi phạm OCP.	Trích xuất đường dẫn hình ảnh thành một cấu hình hoặc tham số để dễ dàng mở rộng mà không cần thay đổi mã
3. Liskov Substitution Principle		
Related modules	Description	Improvement Direction
PaymentTransaction	Nếu trong tương lai, PaymentTransaction được mở rộng (ví dụ: thêm các loại giao dịch khác nhau như RefundTransaction), cần đảm bảo các lớp con giữ nguyên hành vi của các phương thức như isSuccess.	Sử dụng abstraction để định nghĩa các hành vi chung
4. Interface Segregation Principle		
Related modules	Description	Improvement Direction

Interface IPayment	Một hoạt động thanh toán bao gồm ba quy trình chính: thanh toán, hoàn tiền và xử lý kết quả. Những quy trình này khác nhau giữa các cổng thanh toán, vì vậy giao diện cần được tách biệt.	Tách giao diện thành ba interface riêng biệt như sau: IPaymentProcessor: Xử lý các hoạt động liên quan đến thanh toán. IRefundProcessor: Xử lý các hoạt động hoàn tiền. ITransactionResult: Xử lý kết quả giao dịch.
Thêm interface cho quản lý media trong cart	Có 3 logic cần xử lý trong phần giỏ hàng liên quan đến media: quản lý media trong giỏ hàng, tính tổng phí, tổng số lượng media, và xác thực dữ liệu.	Chia thành 3 Interface với các method ICartMediaManager (add, remove, getlist, empty), ICartCalculator (getTotalMedia, calSubtotal) ICartValidator (checkAvailabilityOfProduct, checkMediaInCart)
Thêm interface cho quản lý order	Một đơn hàng có 3 quy trình: xử lý thông tin giao hàng, tạo đơn hàng và tính toán phí vận chuyể	Chia thành 3 Interface với các menthod: IDeliveryInfoProcessor (validateDeliveryInfo, processDeliveryInfo), IOrderCreator (createOrder, createInvoice), IShippingCalculator (calculateShippingFee)
Triển khai Factory design pattern cho việc lựa chọn cổng thanh toán	Trong tương lai, chúng ta có thể sẽ phải tích hợp nhiều phương thức thanh toán khác nhau.	IPaymentMethodFactory (Nơi chọn phương thức thanh toán), IPaymentMethod (Khi tích hợp phương thức thanh toán mới, chỉ cần triển khai interface này và thực hiện logic thanh toán bằng cách ghi đè phương thức)

5. Dependency Inversion Principle		
Related modules	Description	Improvement Direction
PaymentController	Phương thức payOrder gọi đến phương thức payOrder của lớp VnPaySubsystemController thay vì phương thức payOrder của interface IPayment. Trong tương lai nếu ta muốn sử dụng phương thức thanh toán khác, ta sẽ phải chỉnh sửa code	Gọi đến phương thức payOrder của interface IPayment. Ta có thể xác định sử dụng phương thức thanh toán nào trong constructor
Media	Constructor Media gọi đến phương thức getConnection của lớp DBConnection. Trong tương lại nếu ta muốn sử dụng cơ sở dữ liệu khác, ta sẽ phải chỉnh sửa code	Tạo một interface IDatabase và làm cho lớp DBConnection implement nó. Trong constructor Media, ta chỉ cần phải gọi phương thức getConnection của interface IDatabase và ta có thể xác định sử dụng cơ sở dữ liệu nào ở constructor
Book	Phương thức getMediaById gọi đến phương thức getConnection của lớp DBConnection. Trong tương lai nếu ta muốn sử dụng cơ sở dữ liệu khác, ta sẽ phải chỉnh sửa code	Tạo một interface IDatabase và làm cho lớp DBConnection implement nó. Trong phương thức getMediaById, ta chỉ cần phải gọi phương thức getConnection của interface IDatabase và ta có thể xác định sử dụng cơ sở dữ liệu nào ở constructor