Phạm Minh Trường 20215292

1. **SOLID**
2. **Single Responsibility Principle**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | PaymentController | Chịu trách nhiệm cho 2 nhiệm vụ: thanh toán và chuyển kiểu format ngày tháng | Tách ra thành 2 class |
|  | VnPayConfig | Lớp này chịu trách nhiệm quá nhiều về các chức năng như mã hóa, tạo URL và lấy địa chỉ IP | Tách ra các lớp riêng biệt cho mỗi nhiệm vụ, ví dụ HashUtil, UrlBuilder, IpAddressRetriever |
|  | PlaceOrderController | Chịu trách nhiệm cho nhiều nhiệm vụ: điều khiển luồng dữ liệu, validate dữ liệu, tính toán phí giao hàng | Tách ra thành các class con |

1. **Open/Closed Principle**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | calculateShippingFee | Khó mở rộng khi muốn tính phí ship theo cách khác | Tạo một interface calculateShippingFee |
|  | VnPaySubsystemController | Phương thức generatePayOrderUrl không dễ dàng mở rộng để hỗ trợ các phương thức thanh toán khác ngoài VNPay | Tạo các interface hoặc abstract class để mở rộng cho các hệ thống thanh toán khác |

1. **Liskov Substitution Principle**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | Media, Book, CD, DVD | Phương thức getAllMedia() trong Media trả về một List Media nhưng các lớp con override phương thức này trả về null | Xóa đoạn code override |
|  | VNPayConfig | Phương thức hmacSHA512 và hashAllFields đều có thể thay đổi các giá trị của các tham số nhưng không được xác định rõ trong thiết kế | Cần đảm bảo rằng việc thay thế các phương thức này không làm thay đổi hành vi của hệ thống. Tạo interface hoặc class abstract cho các phương thức mã hóa |

1. **Interface Segregation Principle**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | VnPaySubsystemController | Chứa các phương thức xử lý thanh toán như tạo URL thanh toán (generatePayOrderUrl), thực hiện thanh toán (payOrder), và xử lý hoàn tiền (refund). Các phương thức này đang được gộp chung trong một lớp duy nhất, làm cho lớp này có quá nhiều trách nhiệm | Tách các phương thức thành các giao diện nhỏ hơn: Tạo giao diện riêng biệt cho xử lý thanh toán |

1. **Dependency Inversion Principle**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | PaymentTransaction, CreditCard | PaymentTransaction đang phụ thuộc chặt chẽ vào CreditCard | Tạo một lớp abstract PaymentCard |
|  | VnPayConfig, VnPaySubsystemController | VnPayConfig đang trực tiếp phụ thuộc vào các lớp cụ thể và dữ liệu cấu hình, điều này gây khó khăn khi thay đổi cấu hình hoặc mở rộng | Sử dụng Dependency Injection để truyền các cấu hình vào các lớp sử dụng chúng, tránh việc hard-code cấu hình trong lớp VnPayConfig |