# 1. Đánh giá mã nguồn UC Place Rush Order, Pay Order, Place Order

***Chú ý: theo mã nguồn em thiết kế từ BTVN các lab trường, Place Rush Order và Place Order cùng sử dụng chung một lớp, chỉ có thêm một biến Boolean để điều khiển nên là control coupling***

## 1.1. Coupling

### 1.1.1. Content coupling

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| Không có |  |  |

### 1.1.2. Common coupling

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| Phương thức placeOrder trong PlaceOrderController và thuộc tính cart của Cart | Phương thức này sử dụng chung biến static cart với Cart. Lỗi có thể bị lan truyền từ Cart sang PlaceOrderController | Cần lập trình thêm tham số cart cho phương thức placeOrder và để ViewCartController quản lý cart |
| Phương thức getCartSubtotal trong ViewCartController | Phương thức này sử dụng chung biến static cart với Cart. Lỗi có thể bị lan truyền từ Cart sang ViewCartController | Có thể chuyển phương thức này sang lớp Cart và để lớp Cart phụ trách chức năng nàynm |

### 1.1.3. Control coupling

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| processDeliveryInfo trong PlaceOrderController (chú ý, đây là do em thiết kế BTVN từ Lab04) | Dựa vào Boolean isRush để phân luồng thực hiện giao hàng thông thường và giao hàng nhanh | Cần lập trình theo hướng tách hai chức năng này riêng biệt ra và sử dụng chỉ khi nào cần thiết |

### 1.1.4. Stamp coupling

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| calculateShippingFree trong PlaceOrderController với Order | Phương thức này yêu cầu pass toàn bộ tham số order, tuy nhiên điều này là thừa | Có thêm xem xét thay thế Order bằng List <int>, tuy nhiên không nên sửa vì về sau phương pháp tính tiền có thể thay đổi |

### 1.1.5. Data coupling

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| Các phương thức validate trong PlaceOrderController | Các phương thức validate này chỉ yêu cầu duy nhất trường mà nó sẽ validate nên là pass tham số vừa đủ |  |

## 1.2. Coupling

### 1.2.1. Coincidental cohesion

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| Utils | Các lớp này đặt chung package chỉ vì chúng không hợp lý nên chia thành các lớp khác | Tuy nhiên, điều này có thể chấp nhận được vì utils thông thường là các package mang tính thủ tục, ví dụ java.util thì có cả Random (sinh ngẫu nhiên) và ArrayList, không hề liên quan gì tới nhau. |

### 1.2.2. Logical cohesion

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
|  |  |  |

### 1.2.3. Temporal cohesion

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| Các phương thức validate và placeOrder trong PlaceOrderController | Các phương thức chỉ quan hệ với nhau trong lúc ban đầu cần validate thông tin đơn hàng còn về sau không hề sử dụng tới nhau | Có thể phân các phương thức validate thành một lớp khác |

### 1.2.4. Procedural cohesion

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| payOrder và refund trong InterbankSubsystemController | Các phương thức này có quan hệ chặt chẽ theo thời gian | Có thể tách làm hai lớp riêng, tuy nhiên cách làm này sẽ làm hệ thống nhiều lớp hơn |

### 1.2.5. Cohesion: Communicational cohesion

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| Các phương thức payOrder và refund | Cùng sử dụng chung bộ tham số đầu vào và đầu ra, tuy nhiên các thức thực hiện cũng như triển khai là khác nhau |  |

### 1.2.6. Cohesion: Functional cohesion

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| Lớp API | Các phương thức trong API đã được chia nhỏ và cùng hướng đến mục đích là kết nối tới HTTPS thực hiện POST/GET |  |

# 2. Đánh giá mã nguồn theo nguyên lý SOLID

## Single Responsibility Principle

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| PlaceOrderController | Chưa đạt được vì vẫn còn validate và đặt hàng | Có thể tách rời các chức năng, như là tách thêm một Controller phụ trách chức năng validate |

## Open/Closed Principle

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| PlaceOrderController | Chưa vì muốn thay đổi các phương pháp tính tiền vẫn phải thay đổi thêm | Có thể sử dụng strategy pattern, đó là tạo thêm interface ShippingFeeCalculator và các Strategy trong tương lai sẽ implement nó |

## Liskov Substitution Principle

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| Entity | Thiết kế CartMedia, Media, Book, CD, DVD thể hiện rất rõ tư tưởng Liskov | Vẫn còn phương thức getAllMedia khiến các lớp con không thay thế được lớp cha, tuy nhiên có thể xóa các đoạn code override đi và Liskov sẽ ngay lập tức thỏa mãn |
| BaseController, HomeController, ViewCartController, PlaceOrderController, PaymentController | Đã tuân thủ rất rõ nguyên tắc Liskov |  |

## Interface Segregation

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| payOrder và refund ở InterbankSubsystemController | Hai chức năng pay order và refund mặc dù có liên quan tới nhau về thứ tự trong nghiệp vụ, tuy nhiên vẫn nên cần tách biệt nhau vì ở giữa chúng còn rất nhiều nghiệp vụ khác như tạo PaymentTransaction, … | Có thể tách hai chức năng này riêng để tránh interface thực hiện quá nhiều việc |

## Dependency Inversion

|  |  |  |
| --- | --- | --- |
| Related Modules | Description | Improvement |
| PaymentTransaction và CreditCard | Hai lớp này đang phụ thuộc quá chặt vào nhau và theo yêu cầu mở rộng thì rất có thể tương lai sẽ có thêm các loại thẻ | Cần chuyển CreditCard về dạng abstract hoặc thêm mới một lớp abstract để CreditCard kế thừa |