BÀI THỰC HÀNH SỐ 05 – CÁC KHÁI NIỆM VÀ NGUYÊN LÝ THIẾT KẾ PHẦN MỀM

# Coupling

## Content Coupling

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Không có |  |  |

## Common Coupling

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Không có |  |  |

## Control Coupling

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Không có |  |  |

## Stamp Coupling

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Phương thức calculateShippingFee trong lớp PlaceOrderController | Phương thức truyền vào một đối tượng Order trong khi chỉ thực hiện tính toán trên một vài thuộc tính | Có thể thay đổi bằng cách chỉ truyền những tham số quan trọng, tuy nhiên điều này làm giảm đi khả năng mở rộng của hệ thống |

## Data Coupling

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Lớp PaymentController | Các phương thức trong lớp này nhận vào số lượng tham số cần thiết và chỉ trao đổi với các lớp khác thông qua việc trung chuyển các tham số này. | Đây là một thiết kế tốt |

## Uncoupled

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
|  |  |  |

# Cohesion

## Coincidental Cohesion

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Trong lớp Configs và lớp Utils | Trong các class này có nhiều thuộc tính và phương thức được đặt vào mà không có sự liên quan đến nhau (ví dụ: getToday() và md5()) | Tách riêng các thuộc tính và phương thức này vào các module liên quan |

## Logical Cohesion

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Các thuộc tính về URL ở trong lớp Configs | Các thuộc tính liên kết với nhau về mặt logic là đều cung cấp đường dẫn đến với file fxml tương ứng với màn hình cần hiển thị, tuy nhiên mỗi thuộc tính lại có bản chất là tham chiếu đến một màn hình riêng. | Tách các thuộc tính này vào các lớp xử lý tương ứng |

## Temporal Cohesion

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
|  |  |  |

## Procedural Cohesion

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Các phưuong thức validate trong lớp PlaceOrderController | Các phương thức này chỉ quan hệ với nhau theo một trình tự chứ không liên hệ với nhau về mặt chức năng |  |

## Communicational Cohesion

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Các phương thức payOrder và refund trong lớp interbankSubsystemController | Các phương thức này thao tác trên cùng một dữ liệu đầu vào tuy nhiên cách thức thực hiện lại khác nhau | Cohesion ở mức độ này là tạm chấp nhận được |

## Sequential Cohesion

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Không có |  |  |

## Infomational Cohesion

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Các phương thức trong lớp entity Media và Order | Mỗi phương thức đều có một đầu vào và đầu ra riêng biệt nhưng đều thao tác trên các thuộc tính của lớp | Đây là một hướng thiết kế tốt |

## Functional Cohesion

|  |  |  |
| --- | --- | --- |
| Related Module | Description | Improvement |
| Các phương thức ở lớp API | Mỗi phương thức thực hiện một công việc riêng và đều nhắm tới một mục đích chung là xử lý các thông điệp HTTP | Đây là một hướng thiết kế tốt |

# SOLID Principles

## Single Responsibility

Class InterbankSubsystemController chịu trách nhiệm cho hai nhiệm vụ:

* Điều khiển luồng dữ liệu
* Chuyển đổi dữ liệu

Cần thay đổi lớp khi mà luồng dữ liệu thay đổi

## Open/Closed

Khi thiết kế mở rộng cho subsystem interbank để có thể thực hiện thanh toán với nhiều ngân hàng khác nhau:

* Ta viết một subsystem mới implements InterbankInterface và cài đặt các phương thức như payOrder và Refund, từ đó không phải cài đặt lại thiết kế của lớp cũ

Mở rộng đối với nhiều phương thức thanh toán tiền khác nhau:

* Ta viết một Interface đại diện cho các phương thức tính tiền và ở trong đó chỉ có một phương thức calculateShippingFee() duy nhất. Sau đó lần lượt tạo các lớp mới tương ứng với các chiến lược tính phí ship mới. Sau đó muốn tính phí theo chiến lược nào thì ta chỉ cần khởi tạo một đối tượng của Interface là instance của chiến lược tính tiền tương ứng

## Liskov Substitution

Trong lớp Media có các lớp con kế thừa nó là DVD, CD và Book. Tuy nhiên trong lớp Media có phương thức getAllMedia() đã được cài đặt sẵn, trong khi đó các lớp con Override lại phương thức lại trả về null. Vậy khi các đối tượng của lớp DVD, CD hay Book thay thế cho đối tượng Media và thực hiện phương thức getAllMedia() thì chương trình sẽ gặp lỗi logic.

Khắc phục: không cần Override lại phương thức này mà sử dụng luôn phương thức cung cấp bởi lớp cha Media

## Interface Segregation

Trong tương lai khi mở rộng ra các hệ thống thanh toán tiền mới, interface của Interbank hiện tại có thể không phù hợp vì có ngân hàng sẽ không cho phép hoàn tiền. Do vậy nếu trong trường hợp cần thiết cần tách InterbankInterface thành 2 Interface mới chuyên cho việc thanh toán và hoàn tiền.

## Dependency Inversion

Nếu mở rộng thêm các phương thức thanh toán mới thay cho thẻ tín dụng thì thiết kế hiện tại sẽ vi phạm nguyên lý Dependency Inversion do lớp PaymentTransaction đang phụ thuộc chặt vào lớp Card.

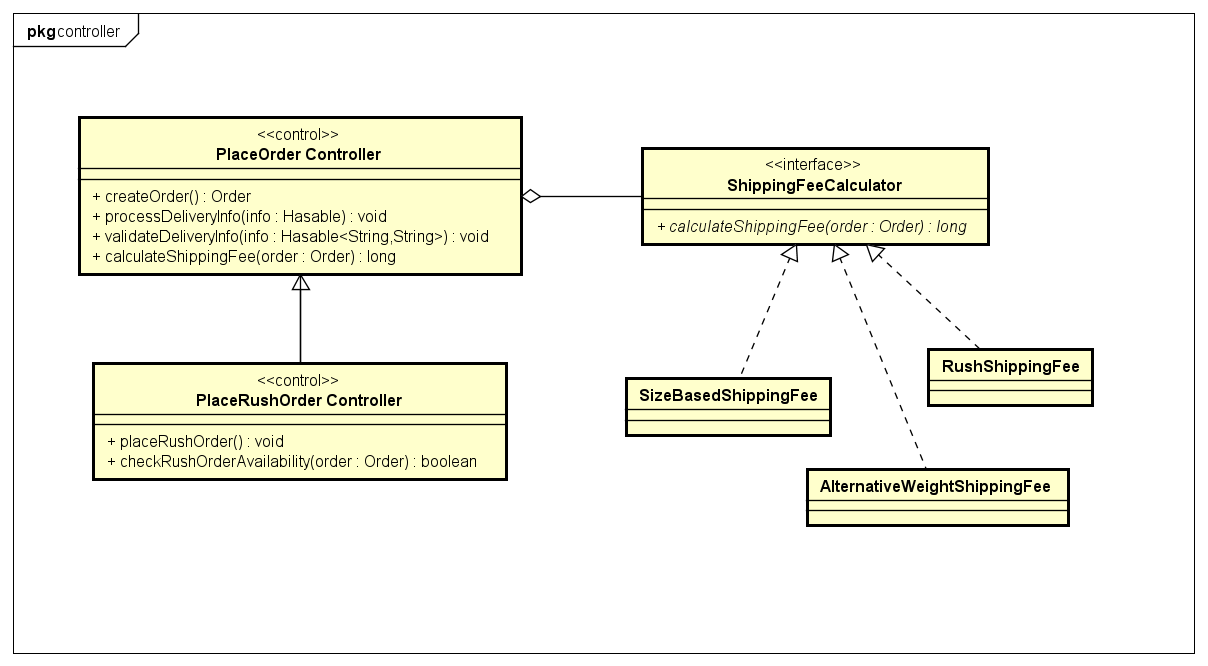
Khắc phục: tạo thêm 1 interface mới cho phương thức thanh toán, và lần lượt cho các lớp thanh toán (vd: CreditCard, COD) implements interface này

# Các biểu đồ lớp cho các phương pháp cải tiến

Diagram

Description automatically generated

Hình 1: Tách interface cho interbank và thêm interface PaymentMethod



Hình 2: Thêm interface cho các phương thức thanh toán