**Bài thực hành 05 – Tuần 13, ngày 21/12/2021**

**Link Github:** [**https://github.com/bahoang3105/NguyenBaHoang-20183543-710808**](https://github.com/bahoang3105/NguyenBaHoang-20183543-710808)

Contents

[I. Một số yêu cầu mở rộng 2](#_Toc91444106)

[1. Thay đổi phí ship: 2](#_Toc91444107)

[2. Thêm một cách tính tiền mới 3](#_Toc91444108)

[3. Sử dụng Interbank khác 5](#_Toc91444109)

[II. UC Place Rush Order 6](#_Toc91444110)

[1. Coupling và Cohesion 6](#_Toc91444111)

[2. Nguyên lý SOLID 6](#_Toc91444112)

1. Một số yêu cầu mở rộng
2. Thay đổi phí ship:

Lớp “PlaceRushOrderController” có phương thức “*calculateShippingFees*” để tính phí vận chuyển. Chúng ta có thể dễ dàng nhận thấy, sau này, nếu ta muốn tính phí ship theo kiểu khác, hoặc cần lưu nhiều kiểu tính phí ship khác nhau, thì chúng ta sẽ phải sửa đoạn code trên bằng một đoạn code với cách tính hoàn toàn khác. Điều này vi phạm nguyên tắc open/closed.

* Giải pháp: tạo ra Interface là ShippingFeeCalculator với phương thức trừu tượng là “*calculateShippingFee*”. Khi chúng ta muốn thêm một hay nhiều cách tính phí ship thì chúng ta chỉ cần viết thêm một lớp mới implement interface trên. Và tại PlaceOrderController chúng ta khởi tạo một đối tượng là interface ShippingFeeCalculator với instance là loại chiến lược tính phí ship mà chúng ta muốn.
* Tạo interface ShippingFeeCalculator với phương thức trừu tượng là calculateShippingFee

Graphical user interface, text, application, email

Description automatically generated

* Tại PlaceOrderController, tạo 1 phương thức khởi tạo với tham số là chiến lược tinh phí ship mà ta muốn sử dụng

Graphical user interface, text, application, email

Description automatically generated

Phương thức tinh phí ship sẽ được tinh theo chiến lược sử dụng

Graphical user interface, text, application, email

Description automatically generated

* Tính phí ship dựa theo cân nặng:

Graphical user interface, text, application

Description automatically generated

1. Thêm một cách tính tiền mới

Hiện tại, PaymentTransaction đang phụ thuộc chặt chẽ vào lớp CreditCard, sau này giả sử không sử dụng CreditCard để thanh toán mà sử dụng một loại phương thức thanh toán khác, ví dụ như domestic debit card… như vậy thiết kế hiện tại đã vi phạm nguyên lý D trong SOLID.

* Giải pháp: tạo một lớp abstract là PaymentCard và lớp PaymentTransaction chỉ quan tâm đến lớp PaymentMethod này.
* Tạo một lớp ảo PaymentCard trong entity.payment

Text

Description automatically generated with medium confidence

* Cho các lớp CreditCard, DomesticDebitCard kế thừa lớp này.
* DomesticDebitCard:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

* Tại lớp PaymentTransaction thay đổi phương thức khởi tạo cũ với tham số CreditCard thành PaymentCard

Graphical user interface, text, application, email

Description automatically generated

1. Sử dụng Interbank khác

* Tại API, phương thức readResponse sẽ chỉ trả về chuỗi kí tự đọc từ InputStreamReader, việc xử lí như thế nào sẽ do Interbank xác định.

Graphical user interface, text, application

Description automatically generated

* Tại InterbankSubsystemController, xử lí response trả về từ API.



Graphical user interface, text, application, email

Description automatically generated

1. UC Place Rush Order
2. Coupling và Cohesion

- Coupling: hiện tại lớp “*PlaceRushOrderController*” đang ở mức độ coupling là data coupling vì chỉ tương tác với các module khác qua tham số truyền vào hàm.

- Cohesion: hiện tại lớp “*PlaceRushOrderController*” đang ở mức độ cohesion là sequential cohesion vì input của thành phần này là output của thành phần kia.

|  |  |  |
| --- | --- | --- |
| Type | Description | Improvement |
| Open/close | Các phương thức validateReceiveTime, validateRushOrderInfo, validateRushOrderInstruction để kiểm tra dữ liệu có thỏa mãn hay không. Chúng ta có thể dễ dàng nhận thấy, sau này, nếu ta muốn kiểm tra dữ liệu theo kiểu khác hoặc muốn sử dụng nhiều kiểu xác thực thì sẽ phải sửa đoạn code trên bằng một đoạn code với các tinh hoàn toàn khác. | Tạo Interface RushOrderInputValidator |

1. Nguyên lý SOLID
   1. Single Responsibility: Lớp “PlaceRushOrderController” thực hiện chức năng chung là điều khiển luồng thực thi của usecase placerushorder.
   2. Open/Close: Các phương thức validateReceiveTime, validateRushOrderInfo, validateRushOrderInstruction để kiểm tra dữ liệu có thỏa mãn hay không. Chúng ta có thể dễ dàng nhận thấy, sau này, nếu ta muốn kiểm tra dữ liệu theo kiểu khác hoặc muốn sử dụng nhiều kiểu xác thực thì sẽ phải sửa đoạn code trên bằng một đoạn code với các tinh hoan toan khác. Điều này vi phạm nguyên tắc open/close.

* Giải pháp:
* Tạo Interface RushOrderInputValidator với các phương thức trừu tượng isValidReceiveTime, isValidRushOrderInfo, isValidRushOrderInstruction.

Graphical user interface, text, application

Description automatically generated

* Tại PlaceRushOrderController tạo một phương thức khởi tạo với tham số là RushOrderInputValidator muốn sử dụng

Graphical user interface, text, application

Description automatically generated

Các phương thức validateReceiveTime, validateRushOrderInfo, validateRushOrderInstruction sẽ sử dụng các phương thức được cung cấp bởi đối tượng RushOrderInputValidator.

Graphical user interface, text, application

Description automatically generated

* Tạo class RushOrderValidator thực thi RushOrderInputValidator:

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

* 1. Liskov Substitution: Sau khi sửa đổi code theo giải pháp trên thì lớp này đã thỏa mãn nguyên lý này.
  2. Interface Segregation: Thỏa mãn nguyên lý này
  3. Dependency Inversion: Thỏa mãn nguyên lý này