**Bài thực hành 05 – Tuần 13**

**Mục lục**

[I. Link đã nộp bài lên Git/GitHub 2](#_Toc91452830)

[II. Nội dung báo cáo kết quả thực hành 2](#_Toc91452831)

[1. Coupling và Cohesion 2](#_Toc91452832)

[1.1. Coupling 2](#_Toc91452833)

[1.2. Cohesion 2](#_Toc91452834)

[1.2.1. Procedural cohesion 2](#_Toc91452835)

[2. Nguyên lý thiết kế SOLID 3](#_Toc91452836)

[2.1. Single Responsibility Principle 3](#_Toc91452837)

[2.2. Open/Closed Principle 3](#_Toc91452838)

[2.3. Liskov Substitution 5](#_Toc91452839)

[2.4. Interface Segregation 6](#_Toc91452840)

[2.5. Dependency Inversion 7](#_Toc91452841)

# Link đã nộp bài lên Git/GitHub

<https://github.com/HieuPham2000/PhamTrungHieu-20183535-710808.git>

# Nội dung báo cáo kết quả thực hành

## Coupling và Cohesion

### Coupling

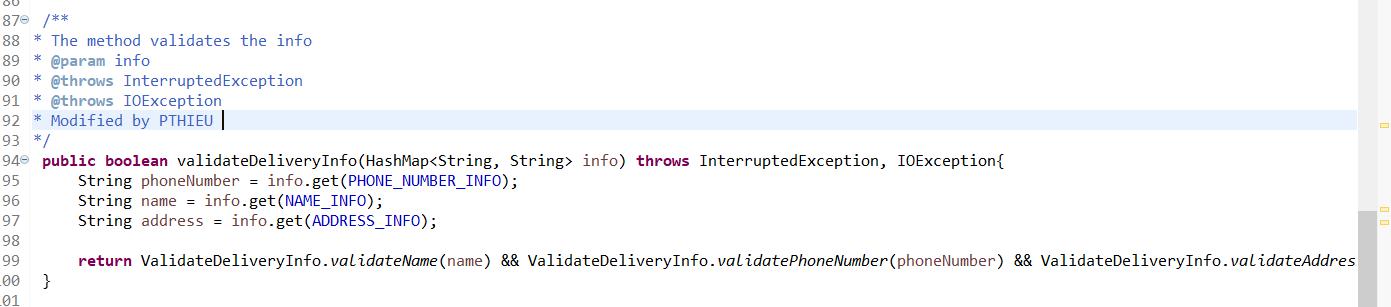
Không có cải tiến

### Cohesion

#### Procedural cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| Class PlaceOrderController trong package control | validate các trường dữ liệu từng bước một với các phương thức validation | Tách các phương thức validation ra class ValidateDeliveryInfo đặt trong package utils.validate |

* Trong lớp PlaceOrderController:



* Lớp ValidateDeliveryInfo



## Nguyên lý thiết kế SOLID

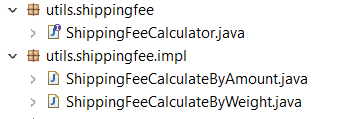
### Single Responsibility Principle

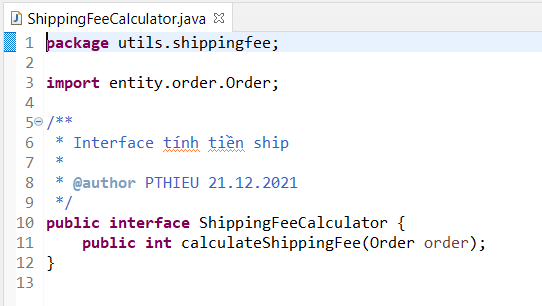
Không có cải tiến

### Open/Closed Principle

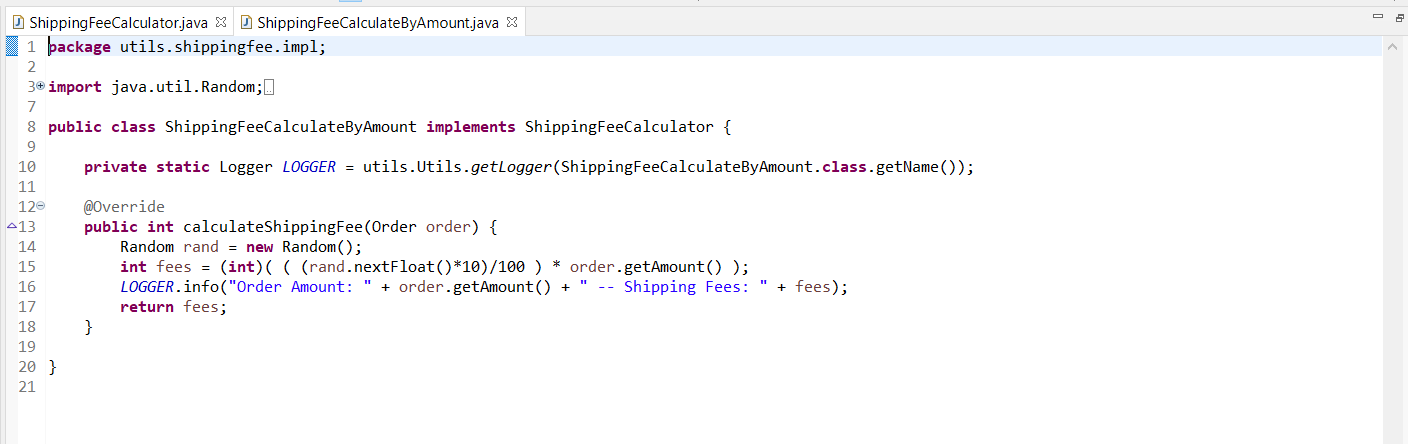
|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
| 1 | Lớp PlaceOrderController | Lớp PlaceOrderController có một phương thức là calculateShippingFee. Sau này, nếu ta muốn tính phí ship theo kiểu khác, hoặc cần lưu nhiều kiểu tính phí ship khác nhau, thì chúng ta sẽ phải sửa đoạn code trên bằng một đoạn code với cách tính hoàn toàn khác | Tạo ra một interface 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 ra một interface ShippingFeeCalculator với phương thức trừu tượng là calculateShippingFee.

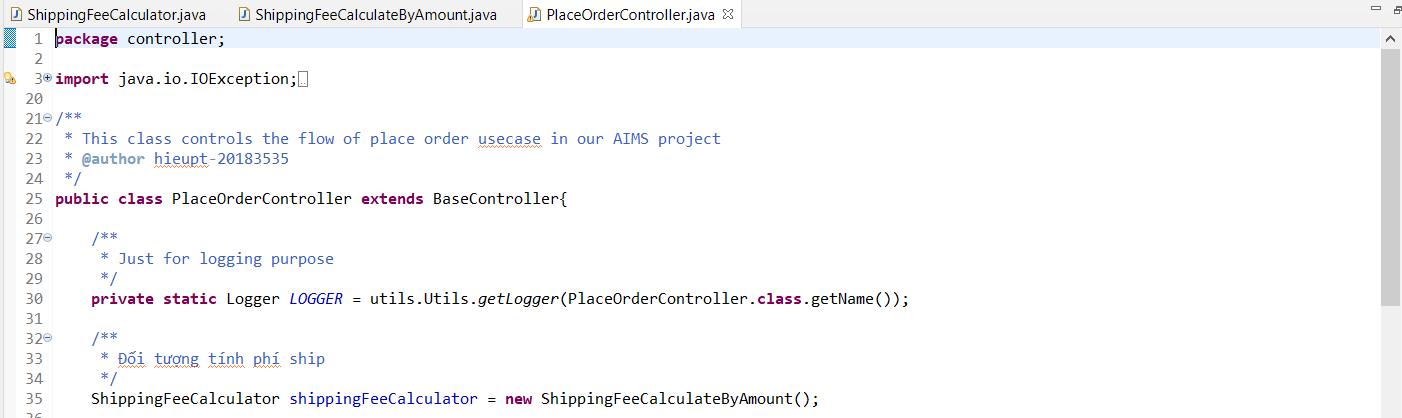


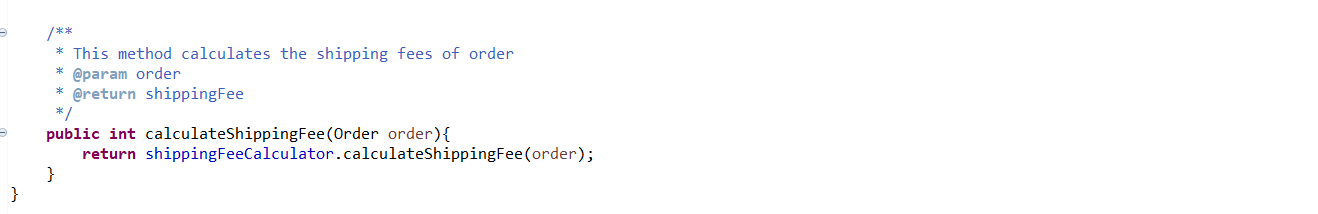


* Viết thêm một lớp mới implement interface trên



* Tại lớp PlaceOrderController





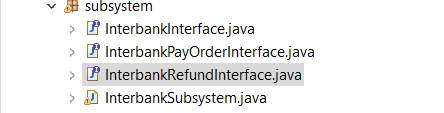
### Liskov Substitution

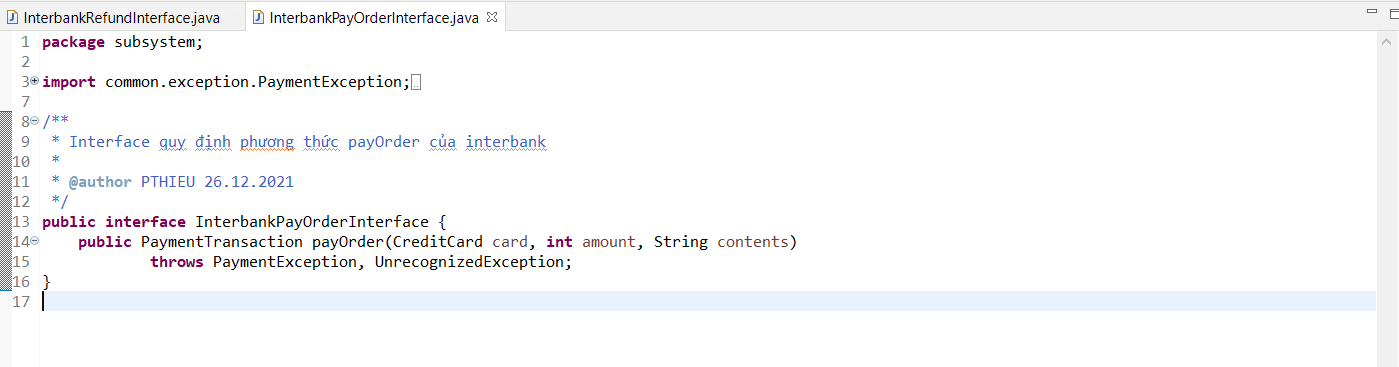
|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
| 1 | Các lớp trong package entity.media | Hệ thống phân cấp cây kế thừa của Media có vi phạm nguyên tắc. Phương thức Media.getAllMedia() được kì vọng trả về một List, tất cả các class con override lại phương thức này nhưng lại trả về null. | Xoá đoạn code Override đi. Chương trình sau đó vẫn chạy bình thường vì phương thức getAllMedia() của lớp cha Media không phải phương thức abstract, không bắt buộc phải override. |

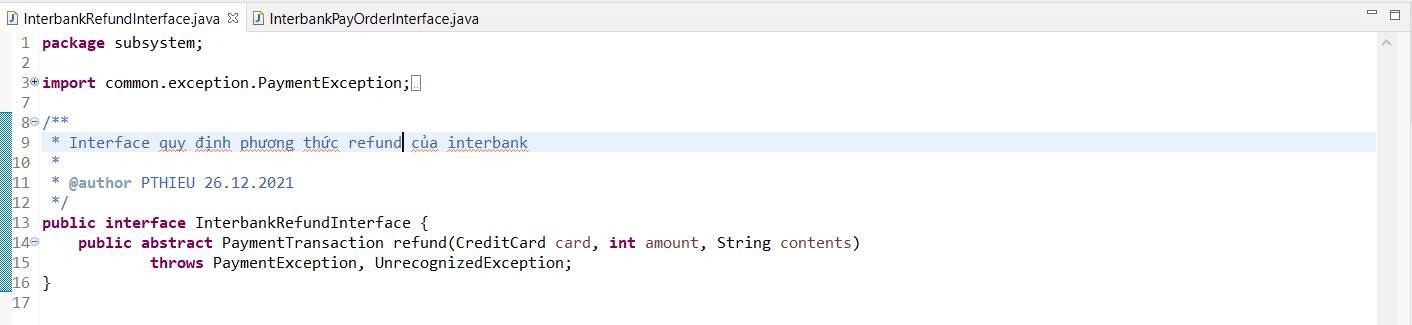


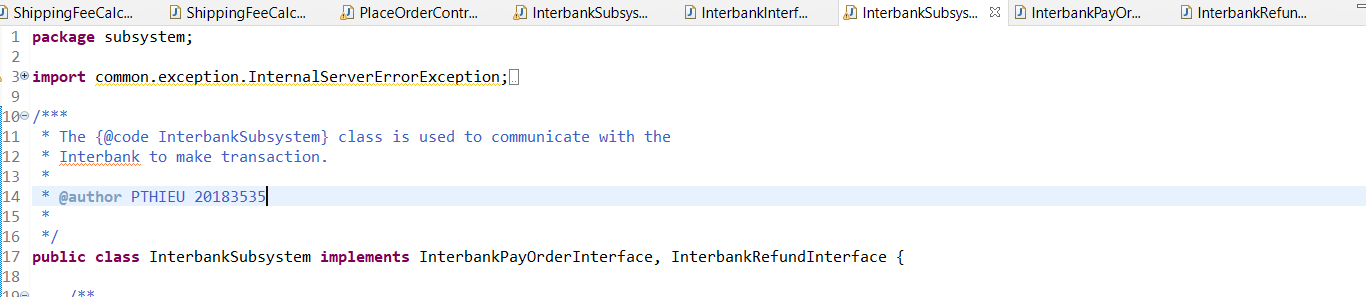
### Interface Segregation

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
| 1 | Package subsystem | Trong tương lai, có thể có một số hệ thống interbank khác không hoàn tiền cho khách hàng mà chỉ thanh toán, lúc này phương thức refund của InterbankInterface trở nên dư thừa đối với interbanksubsystem đó. | Viết thêm interface InterbankPayOrderInterface và InterbankRefundInterface. |









### Dependency Inversion

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
| 1 | Package entity.payment | 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. | Tạo một lớp abstract là PaymentCard và lớp PaymentTransaction chỉ quan tâm đến lớp PaymentMethod này |

