MỤC LỤC:

Contents

[7. BÀI THỰC HÀNH SỐ 05 – CÁC KHÁI NIỆM VÀ NGUYÊN LÝ THIẾT KẾ PHẦN MỀM 1](#_Toc91421493)

[7.1. NỘI dung chi tiết 1](#_Toc91421494)

[1.3.1. Một số yêu cầu mở rộng 1](#_Toc91421495)

[1.3.2. Coupling và Cohesion: các khái niệm cơ bản trong thiết kế 3](#_Toc91421496)

[1.3.3. Nguyên lý thiết kế SOLID 3](#_Toc91421497)

[1.4. BÀI TẬP 4](#_Toc91421498)

[1.4.1. Coupling và Cohesion 4](#_Toc91421499)

[1.4.2. Nguyên lý thiết kế SOLID 5](#_Toc91421500)

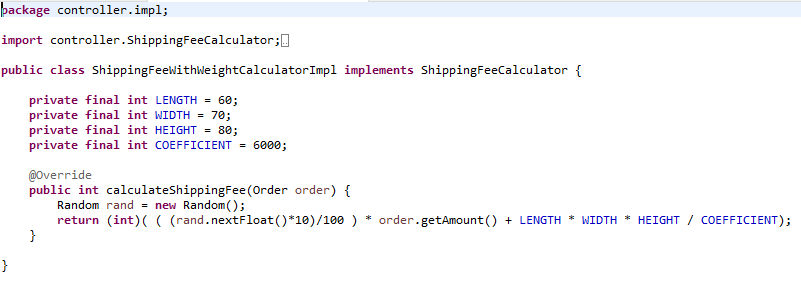
# 7. BÀI THỰC HÀNH SỐ 05 – CÁC KHÁI NIỆM VÀ NGUYÊN LÝ THIẾT KẾ PHẦN MỀM

## 7.1. NỘI dung chi tiết

### Một số yêu cầu mở rộng

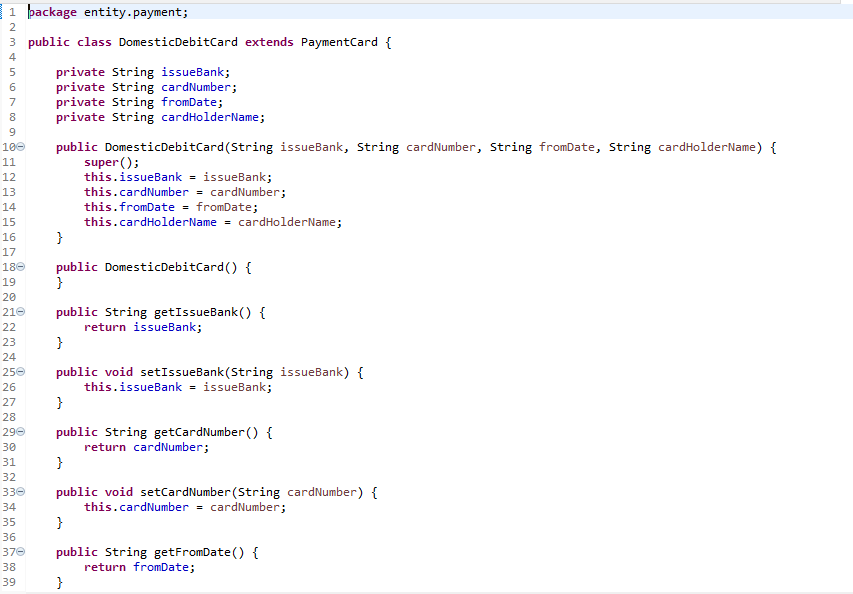
1. **Thay đổi phí ship:**

* Phí vận chuyển hiện nay cũng phụ thuộc vào trọng lượng thực tế, khoảng cách và độ cồng kềnh của sản phẩm (tức là kích thước của sản phẩm: chiều dài, chiều rộng, chiều cao)
* Công thức chuyển đổi được đưa ra như sau:
* Alternative weight (kg) =Length (cm) × Width (cm) × Height (cm) / 6000
* Trọng lượng của sản phẩm bằng trọng lượng thực tế cộng với trọng lượng thay thế.



1. **Thêm một cách tính tiền mới:**

* Domestic debit card:
* Issuing bank, e.g., VietinBank
* Card number: 16 digits
* Valid-from date, e.g., 12/33
* Cardholder’s name, e.g., VU DUY MANH
* Đối với các yêu cầu mở rộng này, giả sử API vẫn giữ nguyên, giả sử thông tin thẻ được thay đổi.



1. **Sử dụng interbank khác**

* Sử dụng một interbank khác với giao thức kết nối và API thay đổi.

### Coupling và Cohesion: các khái niệm cơ bản trong thiết kế

#### Coupling

1. **Content coupling**
2. **Common Coupling**
3. **Control Coupling**
4. **Stamp coupling**
5. **Data coupling**
6. **Uncoupled**

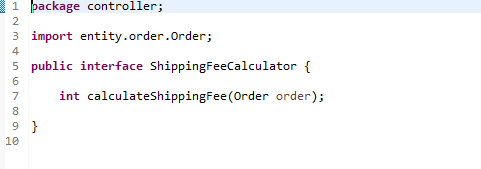
#### Cohesion

1. **Coincidental cohesion**
2. **Logical cohesion**
3. **Temporal cohesion**
4. **Procedural cohesion**
5. **Communicational cohesion**
6. **Sequential cohesion**
7. **Informational cohesion**
8. **Functional cohesion**

### Nguyên lý thiết kế SOLID

#### Single Responsibility

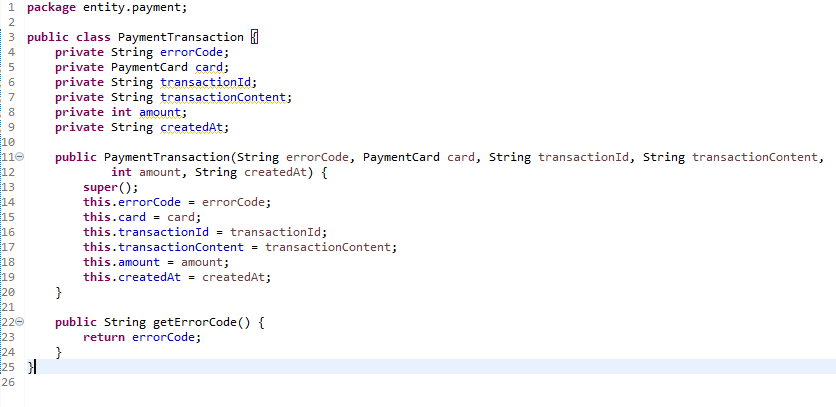
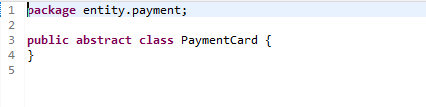
#### Open/Closed



#### Liskov Substitution

#### Interface Segregation

#### Dependency Inversion



## BÀI TẬP

### Coupling và Cohesion

**Trong phần này bạn cần:**

**- Review lại các mức coupling và cohesion và xác định các mức độ này trong modules của thiết kế cho“UC Place Rush Order”. Nếu thiết kế chưa tốt, hãy đề xuất giải pháp cải thiện.**

**- Viết báo cáo cho các vấn đề trên và sửa lại thiết kế và source code tương ứng với đề xuất.**

1. Coupling
   1. Content coupling

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
|  |  |  |

* 1. Common coupling
  2. Control coupling
  3. Stamp coupling
  4. Data coupling
  5. Uncoupled

1. Cohesion
   1. Coincidental cohesion
   2. Logical cohesion
   3. Temporal cohesion
   4. Procedural cohesion
   5. Communicational cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| InterbankSubsystemController | phương thức payOrder và refund có tham số và kiểu trả về giống nhau |  |

* 1. Sequential cohesion
  2. Informational cohesion

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| Hầu hết các lớp entity | Thao tác trên attribute của lớp |  |

* 1. Functional cohesion

### Nguyên lý thiết kế SOLID

**Trong phần này, bận cần thực hiện:**

**- Kiểm tra xem design của bạn sau khi đã thêm UC “Place Rush Order” xem có đáp ứng nguyên lý SOLID hay không trước và sau khi thêm các yêu cầu mở rộng. Nếu thiết kế chưa tốt, hãy đề xuất giải pháp giúp cho thiết kế tốt hơn để đáp ứng được với 5 nguyên lý SOLID.**

**- Viết báo cáo về các vấn đề trên và chỉnh sửa code để implement đề xuất của bạn. Chú ý rằng bạn không cần phải implement các yêu cầu mở rộng. Bạn cần cân nhắc việc có cần thiết kế để đảm bảo có các yêu cầu mở rộng thì không vi phạm các nguyên lý trong SOLID.**

1. Single Responsibility Principle

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  |  |  |  |

1. Open/Closed Principle

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | PlaceRushOrderController | Nếu ta muốn thay đổi cách validate các trường của place rush order, chúng ta sẽ cần phải sửa đoạn code bằng một đoạn code với các validate khác. Điều này vi phạm nguyên tắc open/closed | Tạo interface RushOrderInputValidator với các phương thức trừ tượng isValidReceiveTime, isValidRushOrderInfo,  isValidRushOrderInstruction.Tạo lớp SimpleRushOrderValidator thực thi interface trên. Trong class PlaceRushOrderController, khởi tạo một đối tượng là interface ShippingFeeCalculator với instance là validator chúng ta muốn. |
|  | ShippingFeeCalculator | Interface có phương thức tính phí. Khi muốn thêm một cách tính mới chỉ implement lại |  |
| 2.3 | Related modules | Description | Improvement |
| 2.4 |  |  |  |

1. Liskov Substitution

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | BaseController, PlaceOrderController, PlaceRushOrderController, HomeController, | Kế thừa từ BaseController. Mỗi khi cần thêm controller mới ta viết lớp mới kế thừa. Có thể thay thế BaseController tại mọi nơi sử dụng bằng các lớp con |  |
|  | Media, Book, CD, DVD | Kề thừa từ Media. Có thể thay thế Media tại mọi nơi sử dụng | Đã xoá phương thức getAllMedia khi override các lớp con |

1. Interface Segregation

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | InterbankInterface | Gồm hai phương thức payOrder và refun |  |

1. Dependency Inversion

|  |  |  |  |
| --- | --- | --- | --- |
| # | Related modules | Description | Improvement |
|  | PaymentTransaction, PaymentCard, CreditCard | PaymentTransaction đang phụ thuộc vào CreditCard | Tạo ra abstract class là PaymentCard |