ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Báo cáo Mẫu thiết kế phần mềm

Version 1.0

NHÓM 16

Danh sách thành viên

|  |  |
| --- | --- |
| Nguyễn Bắc Việt | 20173464 |
| Ma Việt Tùng | 20173453 |
| Nguyễn Văn Vũ | 20173470 |
| Nguyễn Công Vượng | 20173475 |

*Hà Nội, ngày 03 tháng 06 năm 2021*

Mục lục

Mục lục 1

1 Tổng quan 6

1.1 Mục tiêu 6

1.2 Phạm vi 6

1.2.1 Mô tả khái quát phần mềm 6

1.2.2 Các chức năng chính của phần mềm 7

1.2.3 Cấu trúc của mã nguồn 10

1.2.4 Các yêu cầu thêm cần cân nhắc cùng quá trình tái cấu trúc 11

1.2.5 Các hoạt động thực thi trên mã nguồn 11

1.2.6 Kết quả dự kiến 11

1.3 Danh sách thuật ngữ 12

1.4 Danh sách tham khảo 12

2 Đánh giá thiết kế cũ 13

2.1 Nhận xét chung 13

2.2 Đánh giá các mức độ coupling và cohesion 13

2.2.1 Coupling 13

2.2.2 Cohesion 16

2.3 Đánh giá việc tuân theo SOLID 18

2.3.1 SRP 18

2.3.2 OCP 19

2.3.3 LIP 20

2.3.4 ISP 20

2.3.5 DIP 20

2.4 Các vấn đề về Clean Code 21

2.4.1 Clear Name 21

2.4.2 Clean function/method 21

2.4.3 Clean Class 22

3 Đề xuất cải tiến 23

3.1 Vấn đề liên quan đến thanh toán và giải pháp 23

3.1.1 Vấn đề 23

3.1.2 Giải pháp 24

3.2 Vấn đề mở rộng chức năng cho việc thêm chức năng cho sản phẩm của modul MediaHandler và giải pháp 26

3.2.1 Vấn đề 26

3.2.2 Giải pháp 27

3.3 Vấn đề lặp code trong các lớp views ScreenHandler và giải pháp 29

3.3.1 Vấn đề 29

3.3.2 Giải pháp 30

3.4 Vấn đề thay đổi cách thức tính khoảng cách và giải pháp 31

3.4.1 Vấn đề 31

3.4.2 Giải pháp 32

3.5 Vấn đề thay đổi cách tính phí vận chuyển và giải pháp 33

3.5.1 Vấn đề 33

3.5.2 Giải pháp 34

3.6 Vấn đề thêm một mặt hàng mới AudioBook 35

3.6.1 Vấn đề 35

3.6.2 Giải pháp 35

3.7 Vấn đề xem chi tiết đơn hàng và xóa những đơn hàng mà mà quản trị viên chưa chấp nhận 38

3.7.1 Vấn đề 38

3.7.2 Giải pháp 38

3.8 Vấn đề vi phạm SRP, design concepts và giải pháp 41

3.8.1 Vấn đề 41

3.8.2 Giải pháp 42

3.9 Vấn đề chưa clean code trong class ApplicationProgrammingInterface và giải pháp 42

3.9.1 Vấn đề 42

3.9.2 Giải pháp 42

3.10 Vấn đề vi pham nguyên tắc infomation Hiding đối với lớp SessionInfomation 43

4 Tổng kết 43

4.1 Kết quả tổng quan 43

4.2 Các vấn đề tồn đọng 44

**Danh sách các hình minh họa**

[Hình 1‑1 Biểu đồ use case tổng quan 8](#_Toc74058921)

[Hình 1‑2 Biểu đồ usecase quản lý giỏ hàng 9](#_Toc74058922)

[Hình 1‑3 Biểu đồ usecase quản lý sản phẩm 10](#_Toc74058923)

[Hình 1‑4 Biểu đồ usecase quản lý người dùng 11](#_Toc74058924)

[Hình 3‑1 Biểu đồ lớp vấn đề thêm phương thức thanh toán trong code base 24](#_Toc74058925)

[Hình 3‑2 Biểu đồ lớp giải pháp vấn đề thêm phương thức thanh toán 25](#_Toc74058926)

[Hình 3‑3 Minh họa code giải pháp thêm phương thức thanh toán -1 26](#_Toc74058927)

[Hình 3‑4 Minh họa code giải pháp thêm phương thức thanh toán -2 26](#_Toc74058928)

[Hình 3‑5 Minh họa code giải pháp thêm phương thức thanh toán -3 27](#_Toc74058929)

[Hình 3‑6 Minh họa code giải pháp thêm phương thức thanh toán - 4 27](#_Toc74058930)

[Hình 3‑7 Vấn đề thêm chức năng cho sản phẩm trong code base 28](#_Toc74058931)

[Hình 3‑8 Giải pháp vấn đề thêm chức năng cho sản phẩm 29](#_Toc74058932)

[Hình 3‑9 Minh họa code cho chức năng thêm màn hình xem chi tiết 30](#_Toc74058933)

[Hình 3‑10 Vấn đề lặp code trong các lớp của module views 31](#_Toc74058934)

[Hình 3‑11 Giải pháp vấn đề lặp code trong module views và thay đổi yêu cầu load giao diện 32](#_Toc74058935)

[Hình 3‑12 Vấn đề thay đổi cách thức tính khoảng cách 33](#_Toc74058936)

[Hình 3‑13 Giải pháp vấn đề thay đổi cách thức tính khoảng cách 34](#_Toc74058937)

[Hình 3‑14 Vấn đề thay đổi cách tính chi phí vận chuyển 35](#_Toc74058938)

[Hình 3‑15 Giải pháp vấn đề thay đổi cách tính chi phí vận chuyển 36](#_Toc74058939)

[Hình 3‑16 Giải pháp vấn đề thêm một mặt hàng mới 37](#_Toc74058940)

[Hình 3‑17 Minh họa code thêm một mặt hàng mới -1 38](#_Toc74058941)

[Hình 3‑18 Minh họa code thêm một mặt hàng mới -2 38](#_Toc74058942)

[Hình 3‑19 Giải pháp vấn đề xử lý đơn hàng 40](#_Toc74058943)

[Hình 3‑20 Minh họa code vấn đề xử lý đơn hàng -1 41](#_Toc74058944)

[Hình 3‑21 Minh họa code vấn đề xử lý đơn hàng -2 41](#_Toc74058945)

[Hình 3‑22 Vấn đề vi phạm SRP 42](#_Toc74058946)

[Hình 3‑23 Giải pháp vấn đề vi phạm SRP 43](#_Toc74058947)

[Hình 3‑24 Vấn đề clean code 43](#_Toc74058948)

[Hình 3‑25 Giải pháp vấn đề clean code 44](#_Toc74058949)

**Danh sách các bảng**

[Bảng 2‑1 Đánh giá mức độ coupling 14](#_Toc74058950)

[Bảng 2‑2 Đánh giá mức độ cohesion 17](#_Toc74058951)

[Bảng 2‑3 Đánh giá việc tuân thủ SRP 19](#_Toc74058952)

[Bảng 2‑4 Đánh giá việc tuân thủ OCP 20](#_Toc74058953)

[Bảng 2‑5 Đánh giá việc tuân thủ DIP 21](#_Toc74058954)

[Bảng 3‑1 Các vấn đề coupling chưa giải quyết 45](#_Toc74058955)

[Bảng 3‑2 Các vấn đề cohesion chưa giải quyết 46](#_Toc74058956)

# Tổng quan

## Mục tiêu

Thiết kế là một bước không thể thiếu trong việc tiếp cận, lập trình và tích hợp các module vào một hệ thống thống nhất. Tài liệu này nhằm chỉ ra kiến trúc phần mềm và cách hoạt động của từng thành phần của phần mềm. Tái cấu trúc lại thiết kế của code base hiện tại.

Mục tiêu cụ thể:

* Xác định mô hình tổng quan phần mềm
* Đánh giá thiết kế cũ của code base
* Xác định các thành phần vi phạm Design concept, SOLID principles
* Áp dụng một số design pattern
* Đề xuất cài tiến các vấn đề vi phạm

Tài liệu cũng là cơ sở để các đội phát triển hệ thống phối hợp làm việc với nhau, lập trình và tích hợp các module vào một hệ thống thống nhất.

Tài liệu còn là cơ sở để người dùng có thể sử dụng vào việc mở rộng mô đun/ hệ thống.

## Phạm vi

Tài liệu này chỉ áp dụng cho việc tái cấu trúc mã nguồn để phù hợp với các yêu cầu phát sinh trong tương lai.

Mỗi yêu cầu thay đổi trong tài liệu này sẽ ảnh hưởng đến kết quả phân tích, thiết kế, lập trình và các kịch bản kiểm tra liên quan đến chương trình.  
Tài liệu này chỉ được phép lưu hành nội bộ giữa các thành viên trong nhóm, phục vụ cho việc báo cáo bài tập lớn môn học. Tài liệu cũng là cơ sở để các thành viên hệ thống phối hợp làm việc với nhau, lập trình và tích hợp các module vào một hệ thống thống nhất.

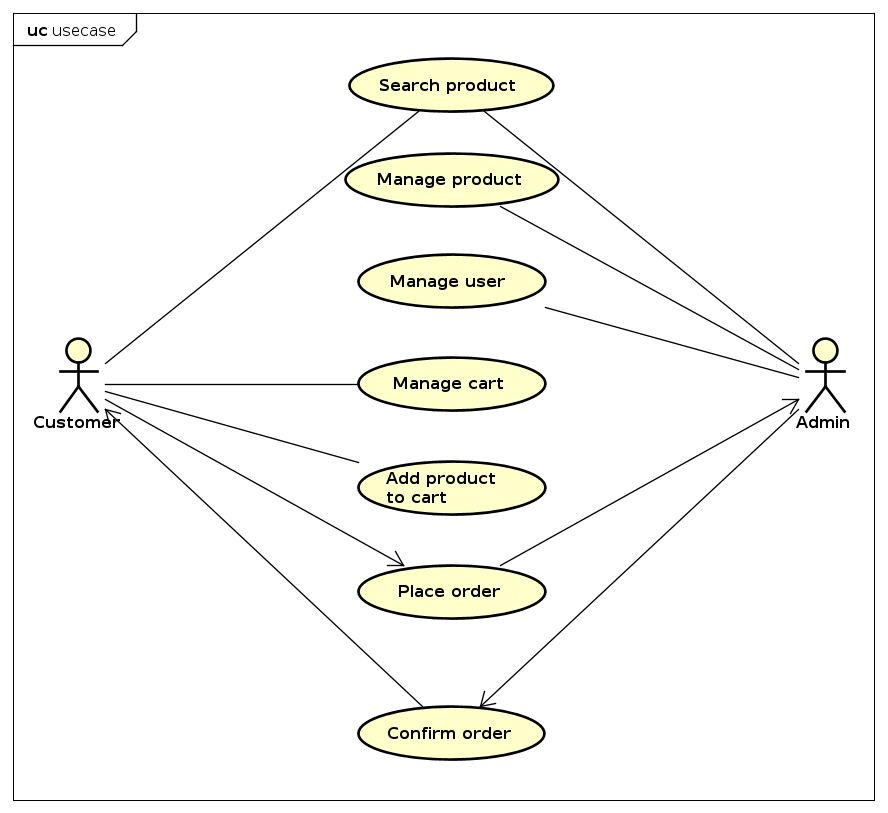
Tài liệu còn là cơ sở để người dùng có thể sử dụng vào việc mở rộng hệ thống. Các quy định về sử dụng tài liệu đối với bên thứ ba là hoàn toàn không có điều kiện về tài chính, bản quyền và tuân theo các quy tắc về tài liệu mở.

### Mô tả khái quát phần mềm

AIMS Project là một hệ thống đa nền tảng hoạt động 24/7, cho phép người dùng mới có thể làm quen dễ dàng. Hệ thống này có thể cho phép phục vụ 1000 khách hàng cùng lúc mà hiệu suất không bị giảm đáng kể, đồng thời có thể hoạt động 300 giờ liên tục không hỏng hóc. Ngoài ra, hệ thống có thể hoạt động trở lại bình thường trong vòng 1 giờ sau khi xảy ra lỗi. Thời gian đáp ứng của hệ thống tối đa là 1 giây khi bình thường hoặc 2 giây lúc cao điểm  
Trong phạm vi môn học này, thay vì các tính năng liên quan tới xác thực tài khoản hay quản lý người dùng, chúng ta sẽ tập trung vào các tính năng liên quan tới đặt hàng và thanh toán đơn đặt hàng của khách hàng trong AIMS Project.

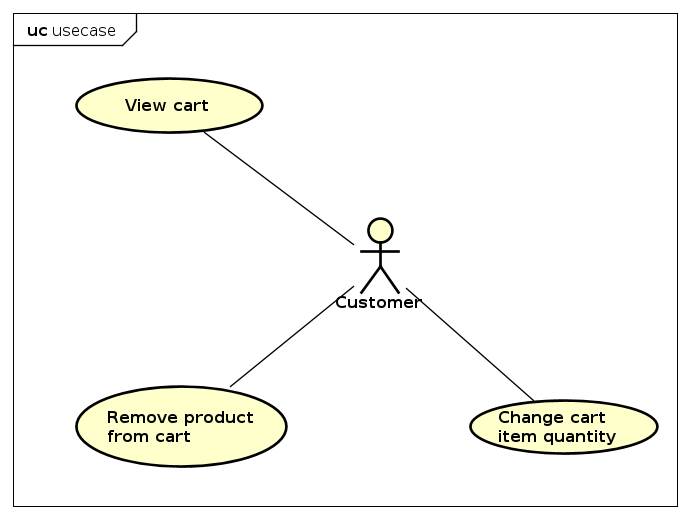
### Các chức năng chính của phần mềm

#### General use case



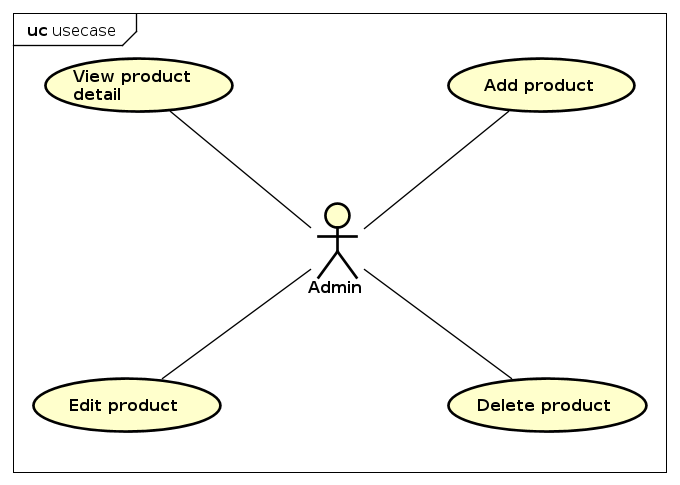
Hình ‑ Biểu đồ use case tổng quan

#### Manage cart



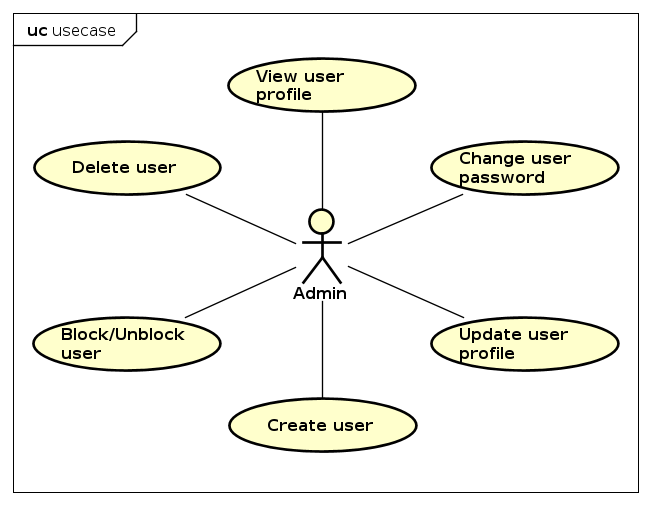
Hình ‑ Biểu đồ usecase quản lý giỏ hàng

#### Manage product



Hình ‑ Biểu đồ usecase quản lý sản phẩm

#### Manage user



Hình ‑ Biểu đồ usecase quản lý người dùng

### Cấu trúc của mã nguồn

Trong project chứa các folder:

* src: chứa mã nguồn cần tái cấu trúc
* lib: chứa các thư viện
* test: thực hiện cho unit test

**Trong src bao gồm:**

* common: chứa các lớp dùng chung cho toàn bộ hệ thống
* controller: chịu trách nhiệm liên kết data từ tầng model (trong lớp dao, subsystem) với tầng views
* dao: chứa các lớp để tương tác với dữ liệu trong database
* entity: chứa các lớp thực thể
* subsystem: chứa module để gọi các api thanh toán
* utils: Chứa các lớp hỗ trợ
* views: thành phần view của project chịu trách nhiệm binding dữ liệu từ controller vào view của file .fxml, xử lý các thao tác của người dùng và truyền cho controller
* App: thực thi chương trình

### Các yêu cầu thêm cần cân nhắc cùng quá trình tái cấu trúc

Sau khi học và tiếp xúc với một số design pattern cùng với việc practice trên code base nhóm nhận thấy về việc thêm mới một sản phẩm thì dựa vào code hiện tại đã dáp ứng được nguyên lý OCP

Nhóm tập trung vào một số yêu cầu thêm sau:

* Thêm màn hình xem chi tiết sản phẩm
* Thay đổi yêu cầu khi load giao diện
* Thay đổi cách tính khoảng cách sử dụng thư viện mới
* Thêm phương thức thanh toán mới
* Thay đổi công thức tính phí vận chuyển
* Cập nhật lại chức năng hủy đơn hàng

### Các hoạt động thực thi trên mã nguồn

Trong các tuần đều có bài tập để practice trên code base:

Nhóm có 4 thành viên chia làm 2 subteam:

* Subteam1: Ma Việt Tùng, Nguyễn Bắc Việt
* Subteam2: Nguyễn Văn Vũ, Nguyễn Công Vượng

Trong các tuần các subteam practice trên code base với các yêu cầu của mỗi tuần. Sau đó từng thành viên trong subteam review code chéo. Sau khi các subteam thống nhất thì push lên github và review chéo theo subtem rồi tiến hành merge code.

### Kết quả dự kiến

Tái cấu trúc mã nguồn để: clean code, áp dụng các design pattern, oop concepts giúp loại bỏ các thành phần vi phạm design concepts, solid principles

## Danh sách thuật ngữ

|  |  |
| --- | --- |
| SRP | Single responsibility priciple |
| OCP | Open/Closed principle |
| LIP | Liskov substitution principe |
| ISP | Interface segregation principle |
| DIP | Dependency inversion principle |
| OOP | Object Oriented Programming |

## Danh sách tham khảo

1. Centers for Medicare & Medicaid Services. (n.d.). *System Design Document Template.* Retrieved from Centers for Medicare & Medicaid Services: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SystemDesignDocument.docx

2. Cornell University How We Refactor and How We Document it? On the Use of Supervised Machine Learning Algorithms to Classify Refactoring Documentation

Retrived from www.elsevier.com/locate/eswa

3. JavaFX tutorial

https://www.javatpoint.com/javafx-tutorial

# Đánh giá thiết kế cũ

## Nhận xét chung

Về cơ bản code có cấu trúc phân cấp hợp lý, khả năng tái sử dụng cao. Tuy nhiên khả năng ứng biến của code đối với các kế hoạch update trong tương lai. Một số chức năng chưa có. Việc sử dụng thư viện khác thay thế hay thêm một phương thức thanh toán mới đều phải thay đổi rất nhiều trong code base

## Đánh giá các mức độ coupling và cohesion

* Coupling thể hiện sự phụ thuộc giữa các component với nhau. Low coupling, loose coupling có nghĩa là các component ít phụ thuộc vào nhau, sự thay đổi trong component này ít khi, hoặc không ảnh hưởng đến component kia. Ngược lại, high coupling và tight coupling cho thấy các component phụ thuộc nhiều vào nhau, khi thay đổi 1 component thì các component kia đều bị ảnh hưởng và có khả năng phải thay đổi theo. Low coupling là mục tiêu chúng ta cần hướng đến để đảm bảo cho hệ thống ít bị ảnh hưởng khi có thay đổi và do đó, tăng tốc độ thực hiện công việc và bảo trì.
* Cohesion thể hiện sự kết dính giữa các thành phần trong cùng một module. Cohesion đánh giá chất lượng mà một component / module tập trung vào một việc đơn lẻ. High cohesion là mục tiêu mà chúng ta hướng tới

### Coupling

Nhìn chung về mức độ package thì các package đảm bảo sự phụ thuộc giữa các package là thấp. Nhưng đối với mức độ class và method thì các module này vẫn còn phụ thuộc nhau khá nhiều.

Bảng ‑ Đánh giá mức độ coupling

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Coupling* | *Module* | *Mô tả* | *Lý do* |
| 1 | Common  Coupling | ViewCartController | Trong các phương thức checkAvailabilityOfProduct(), getCartSubtotal() sử dụng trực tiếp biến cartInstance | Sử dụng trực tiếp dữ liệu của lớp khác, có thể làm thay đổi dữ liệu của lớp sử dụng |
| 2 | Content Coupling | AuthenticationController | Trong các phương thức: getMainUser(), login(String email, String password), logout() đều sử dụng SessionInfomation.mainUser | Truy cập trực tiếp và Thay đổi dữ liêu static của lớp SessionInformation |
| 3 | Data Coupling | BookDAO | Trong phương thức getByID(int id), lấy thông tin về id để hoàn thiện truy vấn câu lệnh sql | Cần thông tin về id để hoàn thiện truy vấn câu lệnh sql |
| 4 | Stamp Coupling,Control Coupling | Cart | Phương thức checkMediaInCart nhận tham số là đối tượng Media. | Phương thức checkMediaInCart chỉ sử dụng thuộc tính id của lớp Media nhưng tham số truyền vào là cả đối tượng Media, truy cap truc tiep vao du lieu cua CartItem. |
| 5 | Data  Coupling | HomeController,  PaymentController | Các tham số của các phương thức được sử dụng và phục vụ cho luông thực thi | Không làm thay đổi dữ liệu của các module khác |
| 6 | Content Coupling | ApplicationProgrammingInterface | Phương thức allowMethods sử dụng các thuộc tính của lớp HttpUrlConnection | Phương thức allowMethods thay đổi trực tiếp giá trị của trường methods và modifiers của lớp HttpUrlConnection. |
| 7 | Control Coupling | InterbankPayloadConverter | extractPaymentTransaction(String responseText) có tham số truyền vào là responseText khi responseText là null thì hàm trả về null nếu không thực hiện tiếp công việc trích thông tin về transaction | Hàm phụ thuộc vào tham số truyền vào response và trả về các kết quả khác nhau. |
| 8 | Stamp Coupling | CartItem | Phương thức khởi tạo có tham số media, cart, quantity, price. | Không sử dụng tham số cart đươc truyền vào |
| 9 | Control  Coupling | PlaceOrderController | Phương thức validateDeliveryInfo và  ProcessDeliveryInfotruyền vào tham số là HashMap | Tham số là HashMap có thể ảnh hưởng đến luồng hoạt động của phương thức. Nếu trong tương lai lớp Delivery thay đổi thì cũng phải thì tham số là HashMap cũng phải thay đổi theo. Nếu bên sử dụng các phương thức này truyền vào HashMap có key không đúng có thể dẫn tới exception |
| 10 | Stamp Coupling | HomeScreenHandler | setupData(Object dto) có tham số truyền vào là dto nhưng trong luồng lại không dùng đến | Bị phí bộ nhớ truyền vào, tham số dto là không cần thiết |
| 13 | Content Coupling | HomeScreenHandler | Phương thức addToCartClick  (Media media, int requestQuantity) gọi phương thức của instance media.setQuanity(int)  và cart.addCartMedia(CartItem) | Làm thay đổi trực tiếp thuộc tính quanity của đối tượng media truyền vào, đối tượng cartInstance được lấy từ lớp SessionInfomation |
| 14 | Content Coupling | ShippingScreenHandler | phương thức preprocessDeliveryInfo() có gọi đến oder.setDeliveryInfo | làm thày đổi trạng thái của thuộc tính delivery bên của instance có class Order |

### Cohesion

Các submodule : view, dao, entity nhìn chung là đã đáp ứng từ tight cohesion trở lên

Submodule controller: các class trong module này hầu như đều ở mức low cohesion

Bảng ‑ Đánh giá mức độ cohesion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Cohesion* | *Module* | *Mô tả* | *Lý do* |
| 1 | Temporal  Cohesion | Authentication  Controller | Định nghĩa phương thứ md5(String message) | Đặt phương thức này trong lớp chỉ vì nó có cùng thời gian thực hiện cho mục đích Authentication nhưng phương thức này không liên quan gì đến logic xử lý của lớp(Thao tác với data và view) |
| 2 | Procedural Cohesion | ApplicationProgrammingInterface | Các đoạn code trong phương thức get được thực hiện theo thứ tự: thiết lập request -> gửi request -> đọc dữ liệu từ response. | Các đoạn code chỉ có liên kết về thứ tự thực hiện, không có liên kết về chức năng. |
| 3 | Procedural Cohesion | ApplicationProgrammingInterface | Các đoạn code trong phương thức post được thực hiện theo thứ tự như đã mô tả ở phương thức get. | Tương tự phương thức get, các đoạn code trong phương thức post không có liên kết về chức năng. |
| 4 | Functional Cohesion | MediaDAO | Có các phương thức getAllMedia(), getMediaById(int id), updateMediaFieldById(String tbname, int id, String field, Object value). chứa các giao tiếp với cơ sở dữ liệu liên quan đến Media | Cùng thực hiện các nhiệm vụ trên chung một cơ sở dữ liệu |
| 5 | Temporal  Cohesion | PaymentController | Có phương thức getExpirationDate | Đặt phương thức này trong lớp chỉ vì nó có cùng thời gian thực hiện cho mục đích Payment nhưng phương thức này không liên quan gì đến logic xử lý của lớp(Thao tác với data và view) |
| 6 | Temporal  Cohesion | PlaceOrderController | Có các phương thức validate | Đặt phương thức này trong lớp chỉ vì nó có cùng thời gian thực hiện cho mục đích Place order nhưng phương thức này không liên quan gì đến logic xử lý của lớp(Thao tác với data và view) |
| 7 | Procedural Cohesion | MyMap | Các đoạn code trong phương thức toMyMap(String str, int idx) thực hiện theo thứ tự: check input -> get key -> get value | Các đoạn code liên quan với nhau về thứ tự thực hiện nhưng không liên quan về chức năng. |

## Đánh giá việc tuân theo SOLID

Nếu phải chỉnh sửa mã nguồn theo các yêu cầu phát sinh thì bản thiết kế và code base chưa tuân theo các nguyên lý thiết kế SOLID

### SRP

Một số class trong submodule controller chưa tuân thủ SRP

Các class trong submodule dao, view, interbank đã tuân thủ SRP

Bảng ‑ Đánh giá việc tuân thủ SRP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | Authentication  Controller | Là lớp controller thực hiện việc kết nối với data và bind dữ liệu vào trong view. Có chứa phương thức md5(String ssage) | Lớp này tồn tại nhiều hơn 1 lý do để thay đổi. Nếu thay đổi cách thức mã hóa( dùng base64, AES, SHA256...) sẽ phải thay đổi mã nguồn của lớp |
| 2 | PlaceOrderController | Là lớp controller thực hiện việc kết nối với data và bind dữ liệu vào trong view. Có chứa các phương thức validate | Lớp này tồn tại nhiều hơn 1 lý do để thay đổi. Nếu thay đổi logic validate hoặc logic place order sẽ phải thay đổi mã nguồn của lớp |
| 3 | PaymentController | Là lớp thực hiện việc xử lý thanh toán giỏ hàng | Tồn tại nhiều hơn 1 lý do để thay đổi. Có phương thức getExpridate(String date) sẽ phải thay đổi khi định dạng date thay đổi |
| 4 | Cart | Là lớp chứa dữ liệu của đơn hàng mà khách đặt trên hệ thống AIMS có phương thức tính giá calSubtotal() | Khi thay đổi cách tính giá thì ta phải thay đổi bên trong lớp này. |

### OCP

Các lớp DAO đã tuân thủ theo nguyên lí OCP.

Các lớp còn phải thay đổi code khi thay đổi yêu cầu thêm.

Bảng ‑ Đánh giá việc tuân thủ OCP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | Order | Lớp để chứa thông tin về đơn hàng như phí vận chuyển, giá cả,... | Khi thay đổi cách tính phí vận chuyển thì phải sửa trực tiếp trong code do nó phụ thuộc vào lớp DeliveryInfo. |
| 2 | DeliveryInfo | Dùng để xử lí dữ liệu về vận chuyển | Khi thay đổi phương thức tính toán DistanceCalculator phải sửa code trong class. |
| 3 | MediaHandler | Hiển thị thông tin chung của media | Chưa có các thao tác với media. Khi  Thực hiện yêu cầu thêm màn hình xem chi tiết thì phải chỉnh sửa code ở lớp này |
| 4 | PaymentControler | Kiểm soát giao dịch | Khi đổi thẻ ATM khi từ CreditCard-> Domestic Card thì phải thay đổi code. |
| 5 | PaymentTransaction | Thông tin giao dịch | Khi đổi thẻ ATM khi từ CreditCard-> Domestic Card thì phải thay đổi code. |
| 6 | InterBankSubSystem | Giao tiếp với interbank để thực hiện giao dịch | Khi đổi thẻ ATM khi từ CreditCard-> Domestic Card thì phải thay đổi code. |
| 8. | InterbankSubsystemController | KIểm soát interbanksubsystem | Khi đổi thẻ ATM khi từ CreditCard-> Domestic Card thì phải thay đổi code. |
| 9 | Các lớp ...ScreenHandler trong views.screen | Các màn hình giao diện | Khi thay đổi cách thức thông báo lỗi khi khởi tạo giao diện, phải thay đổi code của các màn hình. |
| 10 | MediaHandler | Hiển thị các sản phẩm trên màn hình Home | Sử lý thêm sản phầm vào giỏ hàng bằng ObserverPattern. Nếu trong tương lai cần xử lý thêm sự kiện nào đó của MediaHandler(ví dụ sự kiện xem chi tiết sản phẩm) cần thay đổi rất nhiều code |

### LIP

Các lớp con đã thực hiện kế thừa tốt, không xuất hiện các lớp con khác biệt với lớp cha

### ISP

Các interface đã thỏa mãn nguyên lý ISP, không có interface nào mà có phương thức thưa.

### DIP

Bảng ‑ Đánh giá việc tuân thủ DIP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | Order | Lớp để chứa thông tin về đơn hàng như phí vận chuyển, giá cả,... | Phí vận chuyển của nó phụ thuộc vào lớp DeliveryInfo. |
| 2 | DeliveryInfo | Dùng để xử lí dữ liệu về vận chuyển | Phụ thuộc vào lớp DistanceCalculator, nên khi thay đổi cách tính toán là phải thay đổi nhiều. |
| 3 | InterbankInterface | payOrder và refund hiện đang sử dụng lớp cụ thể CreditCard. | Các phương thức này đang phụ thuộc vào lớp cụ thể CreditCard và không thể tái sử dụng nếu trong tương lai sử dụng phương thức thanh toán mới. |
| 4 | PaymentControler | Kiểm soát giao dịch | Phụ thuộc CreaditCard |
| 5 | PaymentTransaction | Thông tin giao dịch | Phụ thuộc CreaditCard |
| 6 | InterBankSubSystem | Giao tiếp với interbank để thực hiện giao dịch | Phụ thuộc CreaditCard |
| 7. | InterbankSubsystemController | KIểm soát interbanksubsystem | Phụ thuộc CreditCard |

## Các vấn đề về Clean Code

### Clear Name

* *Một số phương thức còn dùng biến viết tắt, khó hiểu, ví dụ:*

*.* AuthenticationController: phương thức md5, tên biên md là tên viết tắt không nên dùng đổi thành messageDigest, tên biến sb nên đổi thành StringBuilder.

Cart: tên biến cm nên đổi thành cartItem.

### Clean function/method

Mã nguồn ban đầu chưa đáp ứng clean method.

* Các phương thức setupData(null), setupFunctionality() bị duplicate trong các class screenhandler

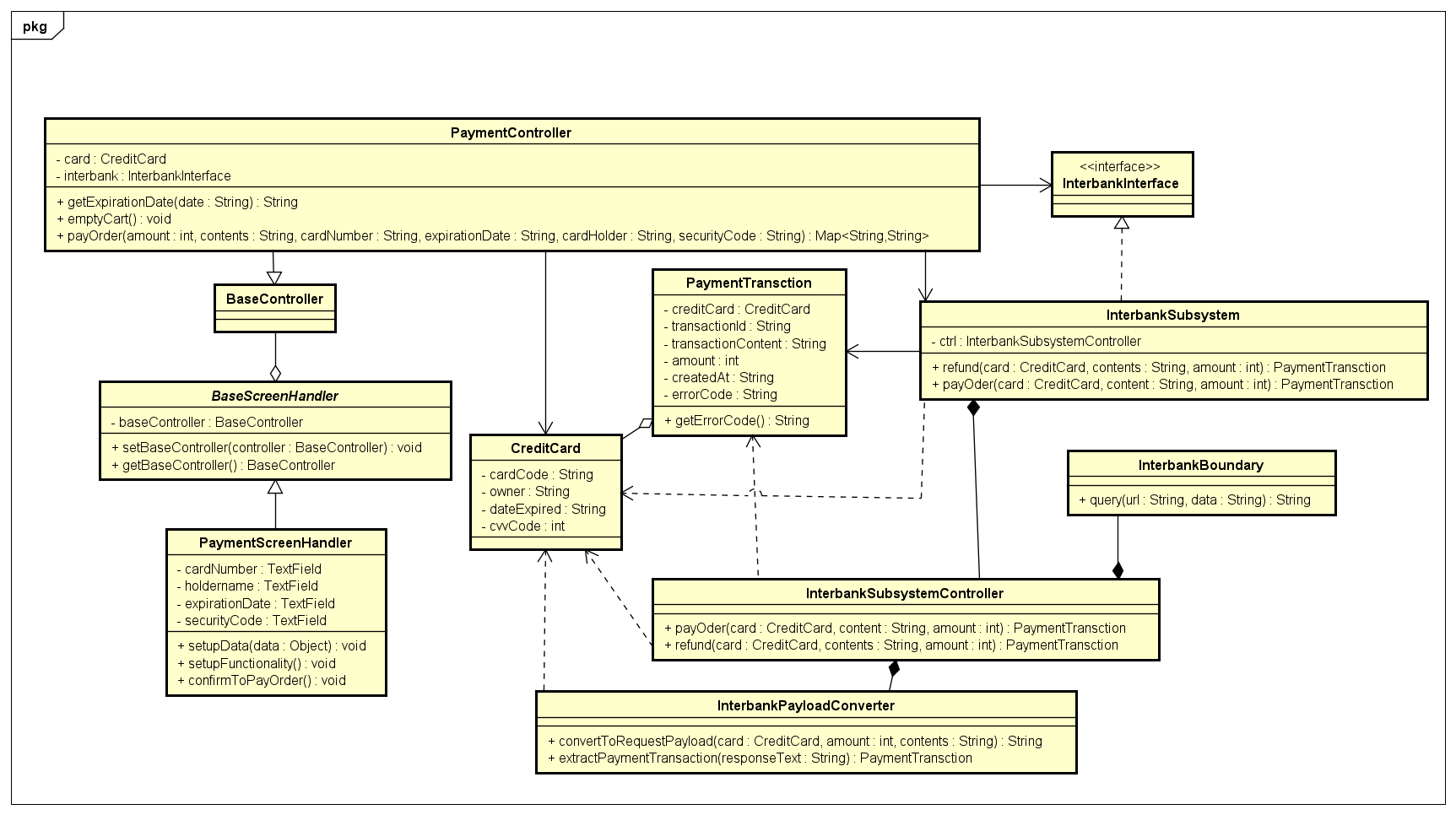
### Clean Class

* Một số class vẫn còn code smell chỉ một thay đổi nhỏ trong class dẫn tói phải thay đổi rất nhiều method: Đó là các class vi phạm nguyên lý OCP(Các class trong view.screen, DeliveryInfo, Các lớp trong module payment)
* Class Mediahandler bị duplicate

# Đề xuất cải tiến

## Vấn đề liên quan đến thanh toán và giải pháp

### Vấn đề



Hình ‑ Biểu đồ lớp vấn đề thêm phương thức thanh toán trong code base

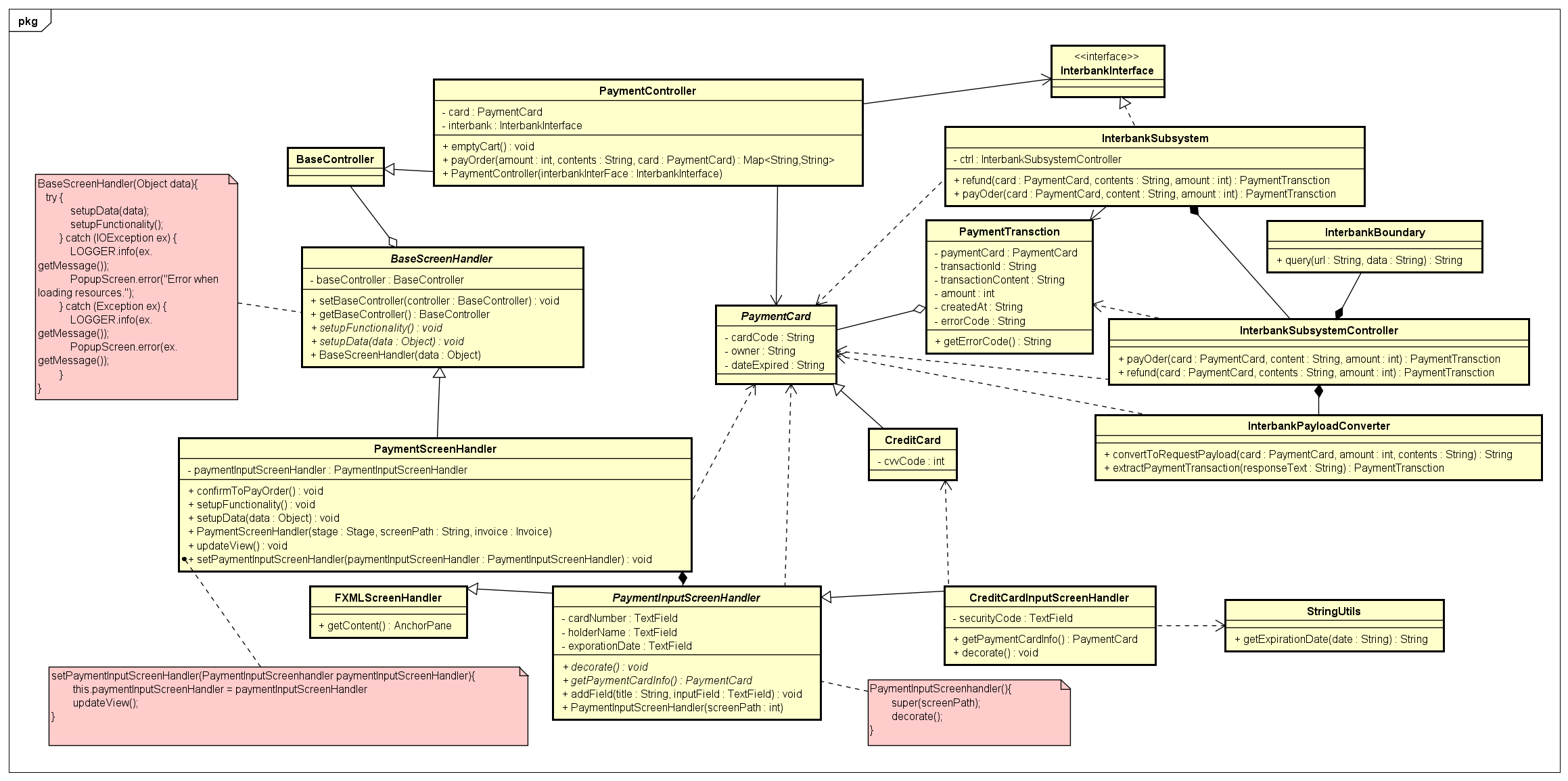
Từ biểu đồ lớp ta có thể thấy một số vấn đề sau:

* Trong PaymentController chứa phương thức getExpiredate. Phương thức này được thêm vào lớp chỉ là vì trong lớp có sử dụng phương thức này, nhưng phương thức này không liên quan đến nghiệp vụ thanh toán của lớp Controller. Lớp vi phạm SRP.
* Các thành phần trong module thanh toán này đều phụ thuộc vào lớp chi tiết là CreditCard vì vậy vi phạm DIP, khi thêm một phương thức thanh toán mới sẽ cần modify toàn bộ code vì vậy vi phạm OCP.
* Lớp PaymentScreenhandler chỉ chứa các thành phần view của CreditCard vì vậy khi thêm phương thức thanh toán mới lớp này không đáp ứng được yêu cầu, chỉ có cách viết lại lớp này để thêm được phương thức thanh toán mới.

### Giải pháp

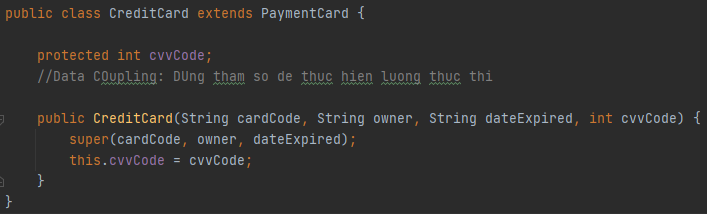
**Áp dụng strategy pattern:**

* Cần tạo lớp abstract PaymentCard và các thành phần trong module sẽ phụ thuộc vào PaymentCard. Khi thêm phương thức mới chỉ cần tạo lớp kế thừa PaymentCard
* Tạo thêm lớp abstract PaymentCardInputScreenHandler chứa các textfield của PaymentCard. Khi thêm phương thức thanh toán mới sẽ tạo lớp kế thừa từ PaymentCardInputScreenHandler và thêm các textfield riêng của lớp đó
* Khi người dùng chọn phương thức thanh toán từ giao diện sẽ setPaymentCardInputScreenHandler, updateView của PaymentScreenHandler và khi người dùng xác nhận thanh toán thì sẽ lấy thông tin thanh toán từ getPaymentInfo từ PaymentCardInputScreenHandler thay vì lấy trực tiếp từ PaymentScreenhandler

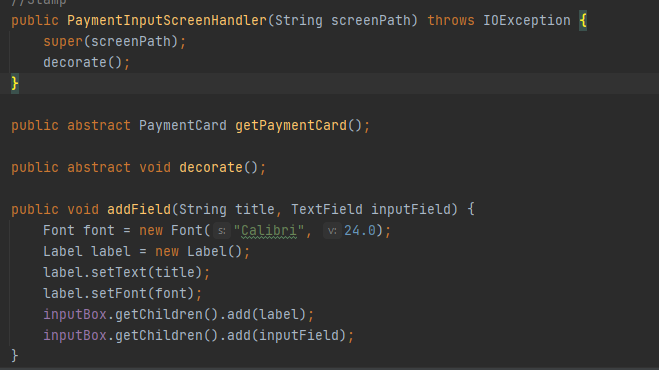


Hình ‑ Biểu đồ lớp giải pháp vấn đề thêm phương thức thanh toán

**Minh họa code:**



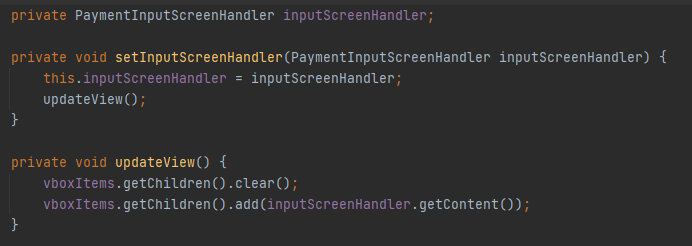
Hình ‑ Minh họa code giải pháp thêm phương thức thanh toán -1



Hình ‑ Minh họa code giải pháp thêm phương thức thanh toán -2



Hình ‑ Minh họa code giải pháp thêm phương thức thanh toán -3

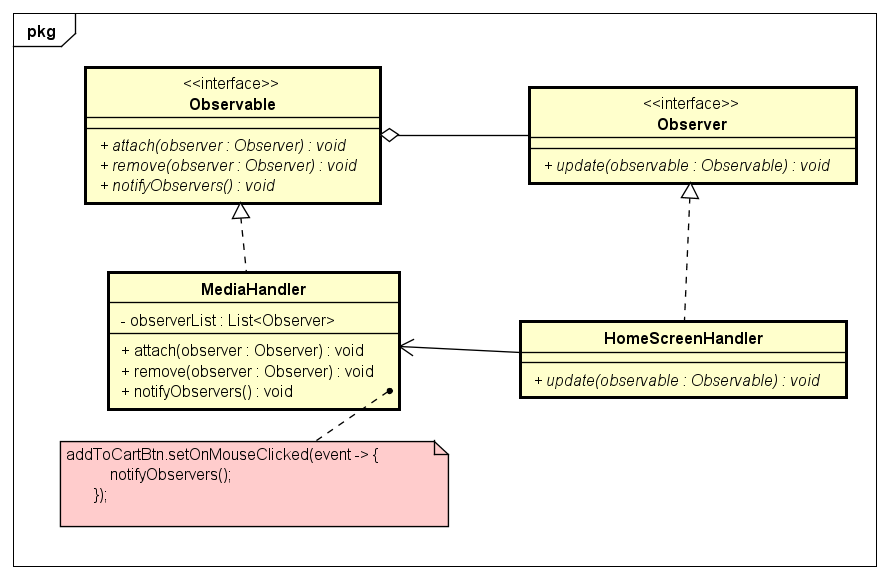


Hình ‑ Minh họa code giải pháp thêm phương thức thanh toán - 4

## Vấn đề mở rộng chức năng cho việc thêm chức năng cho sản phẩm của modul MediaHandler và giải pháp

### Vấn đề

* Hiện tại trong code còn có sự lặp lại tên của lớp MediaHandler -> đổi tên lớp MediaHandler trong module cart thành CartMediaHandler



Hình ‑ Vấn đề thêm chức năng cho sản phẩm trong code base

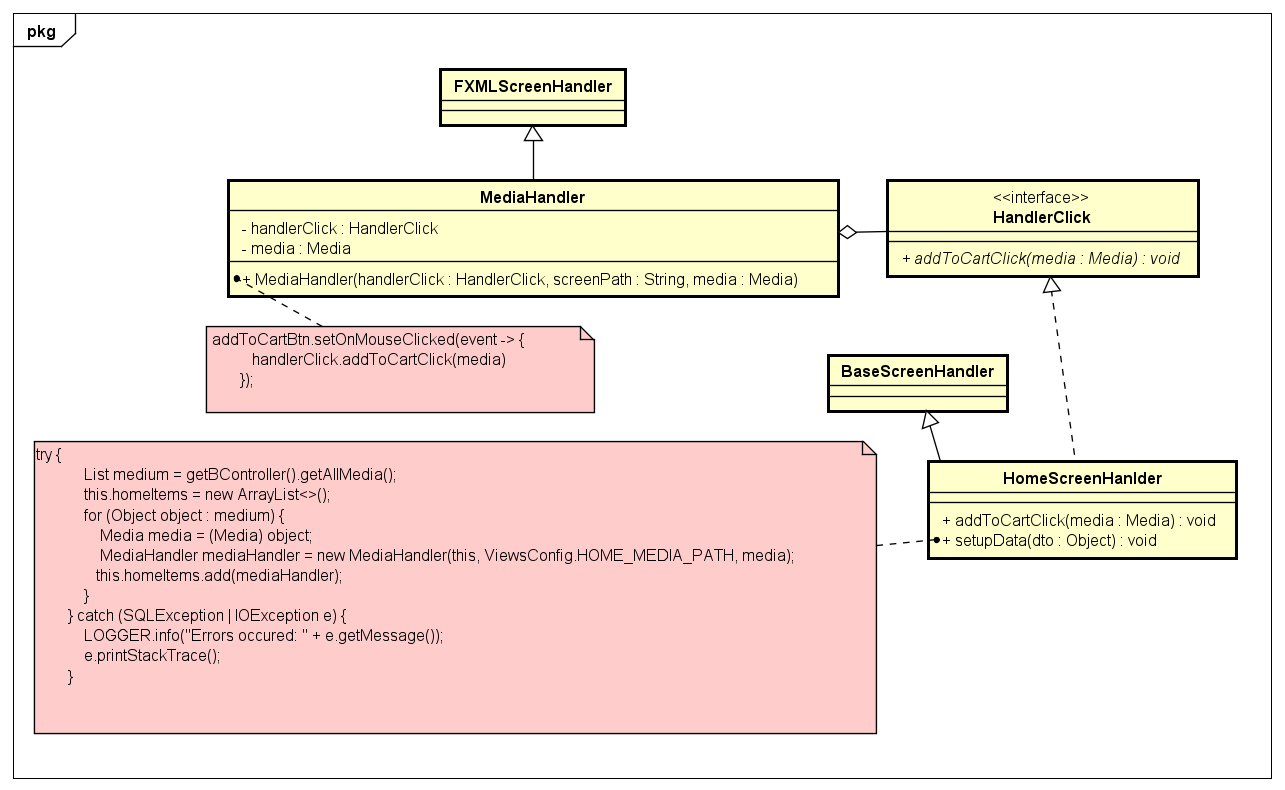
**Các vấn đề trong thiết kế trên:**

* Áp dụng mẫu thiết kế Observer pattern là chưa hợp lý vì chỉ có lớp HomeScreenHanlder attach MediaHandler vì vậy không cần thiết phải áp dụng observer. Hơn nữa MediaHandler hiện tại chỉ có thể notifyObservers() khi click vào button addToCartBtn. Trong tương lai nếu có yêu cầu thêm về việc click vi dụ như thêm button toDetailBtn thì không thể handle được việc click button và thông báo cho HomeScreenHandler
* Để giải quyết việc thêm button toDetailBtn mà vẫn áp dụng ObserverPattern không phải là không được chứ ta có thể thêm tham số(tag, interface HandleClick) vào hàm update của Observer và notifyObservers() của Observable để HomeScreenHandler biết sự thay đổi là do việc click nào tạo ra rồi xử lý. Nhưng nếu thiết kế như vậy thì hàm update() có thể dẫn tới control coupling

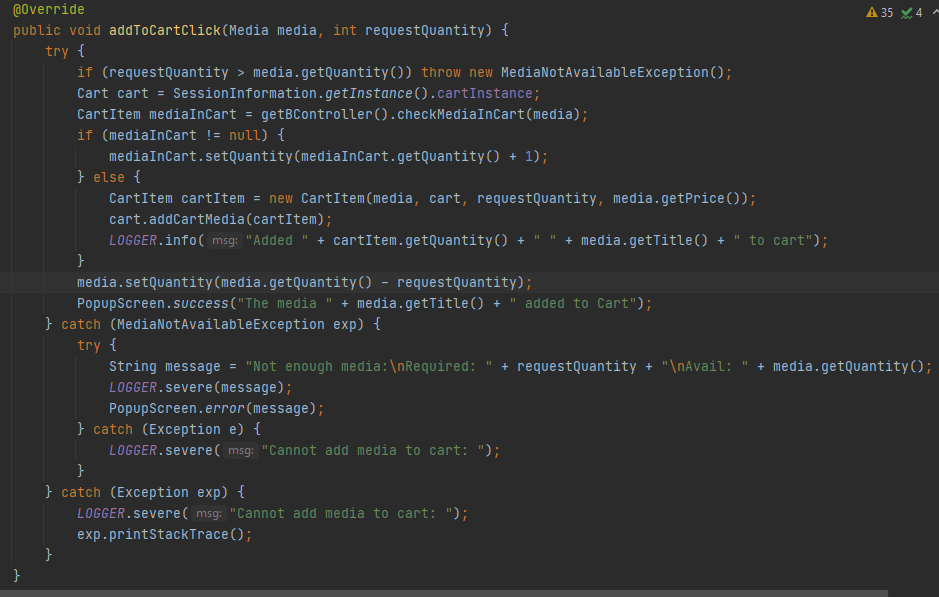
### Giải pháp

Thay vì áp dụng observer pattern để lắng nghe việc click button thì sử dụng interface HandleClick để HomeScreenHandler giao tiếp với MediaHandler:

* HomeScreenHandler sẽ implement HandlerClick.
* Khi xảy ra sự kiện click bên MediaHandler thì chỉ cần gọi phương thức tương ứng từ HandlerClick



Hình ‑ Giải pháp vấn đề thêm chức năng cho sản phẩm



Hình ‑ Minh họa code cho chức năng thêm màn hình xem chi tiết

## Vấn đề lặp code trong các lớp views ScreenHandler và giải pháp

### Vấn đề

* Đoạn code dưới đây bị lặp trong phương thức khởi dựng của các lớp HomeScreenHandler, CartScreenHandler, IntroScreenHandler,...:

try {

setupData(data);

setupFunctionlity();

} catch (IOException ex) {

LOGGER.info(ex.getMessage());

PopupScreen.error("Error when loading resources.");

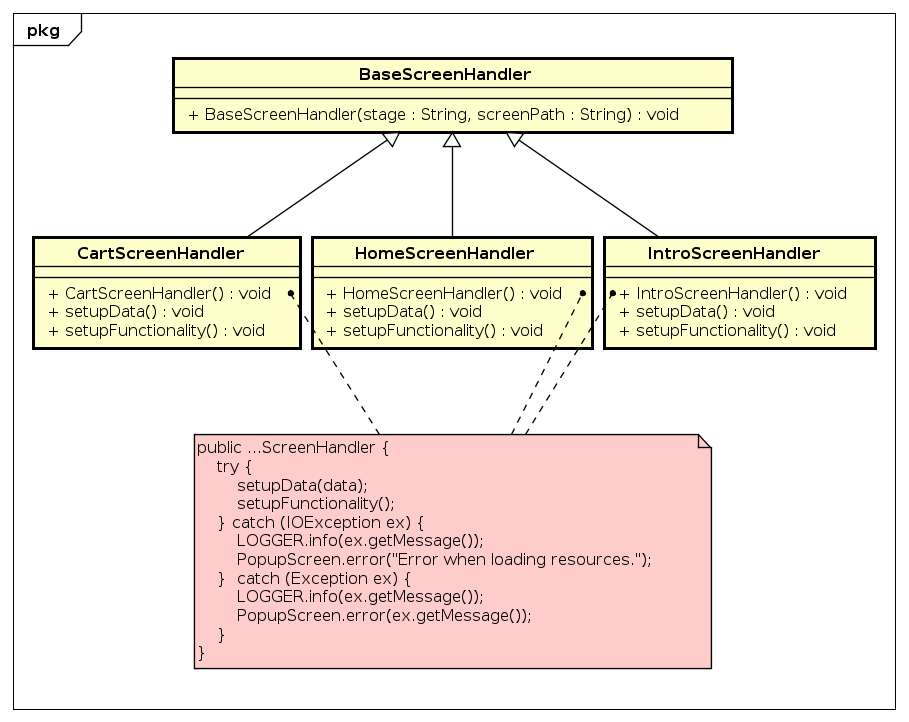
} catch (Exception ex) {

LOGGER.info(ex.getMessage());

PopupScreen.error(ex.getMessage());

}

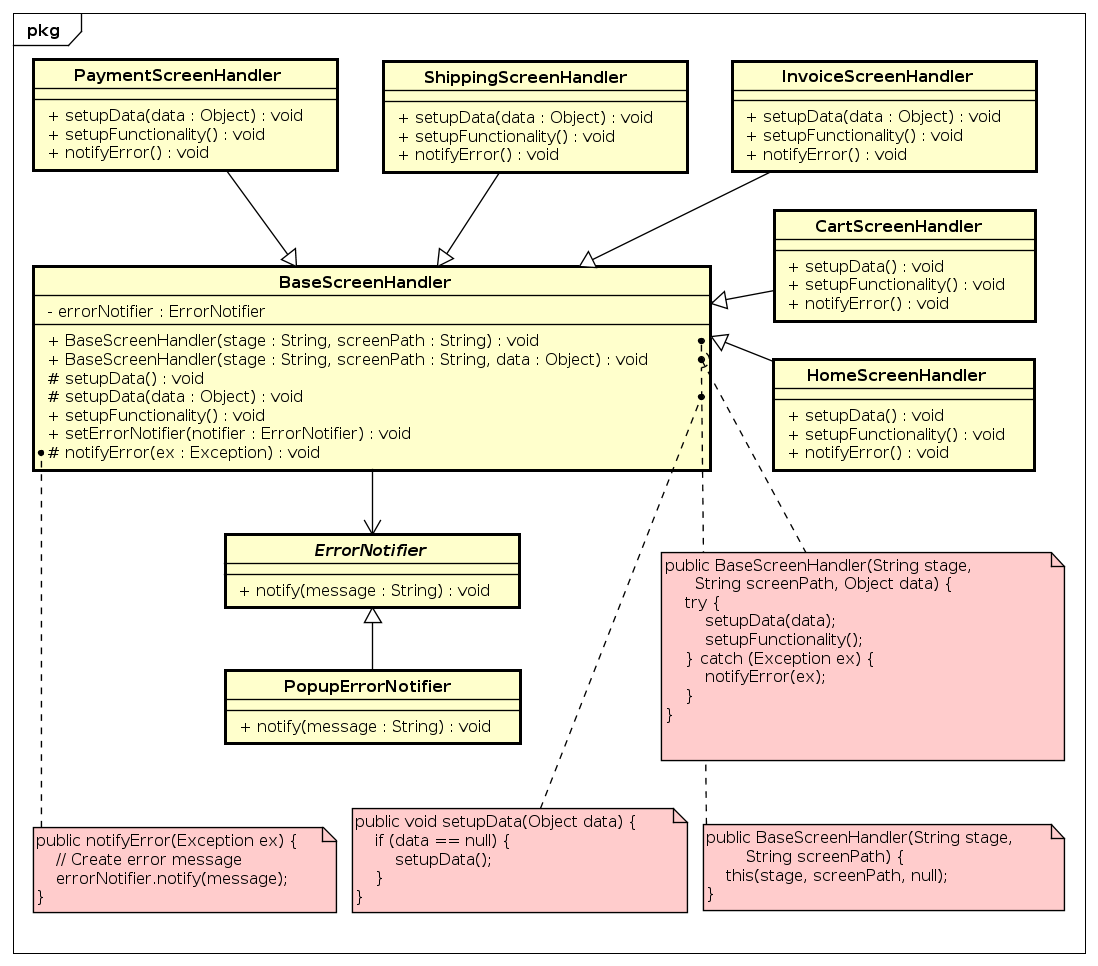
* Bên cạnh đó, trong tương lai khi có yêu cầu thay đổi cách thức thông báo lỗi khi khởi tạo màn hình, cần phải thay đổi code của tất cả các lớp ScreenHandler (vi phạm nguyên lý OCP).



Hình ‑ Vấn đề lặp code trong các lớp của module views

### Giải pháp

* Vấn đề lặp code: áp dụng Template method
  + Đưa đoạn code bị lặp lên phương thức khởi dựng của lớp cha BaseScreenHandler.
  + Lớp BaseScreenHandler có các phương thức setupData và setupFunctionality và các lớp con sẽ ghi đè các phương thức này.
* Vấn đề vi phạm nguyên lý OCP khi có thay đổi: áp dụng Strategy Pattern
  + Lớp BaseScreenHandler có thêm thuộc tính errorNotifier là đối tượng của lớp abstract ErrorNotifier. Lớp này có phương thức notify thực hiện thông báo lỗi.
  + Việc thông báo lỗi khi khởi tạo các màn hình được thực hiện bằng cách gọi errorNotifier.notify().
  + Khi cần thay đổi cách thức thông báo lỗi, chỉ cần thay đổi thuộc tính errorNotifier.



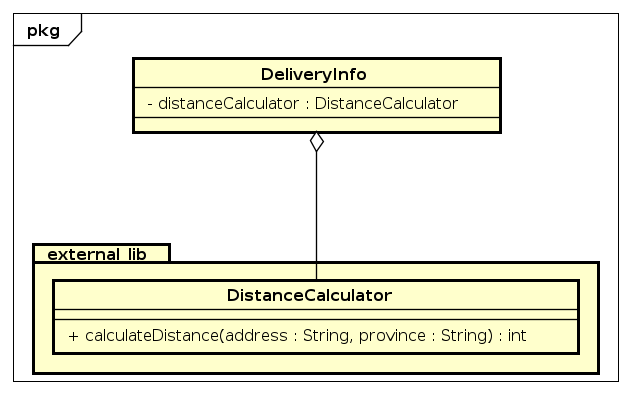
Hình ‑ Giải pháp vấn đề lặp code trong module views và thay đổi yêu cầu load giao diện

Lớp PopupErrorNotifier thực hiện thông báo lỗi bằng hình thức popup.

## Vấn đề thay đổi cách thức tính khoảng cách và giải pháp

### Vấn đề

Lớp DeliveryInfo hiện tại đang sử dụng thư viện distance-api.jar, trong tương lai có thể sử dụng thư viện mới alt-distance-api.jar với interface khác với thư viện cũ nên cần phải sửa code nếu muốn sử dụng thư viện mới này (vi phạm OCP).

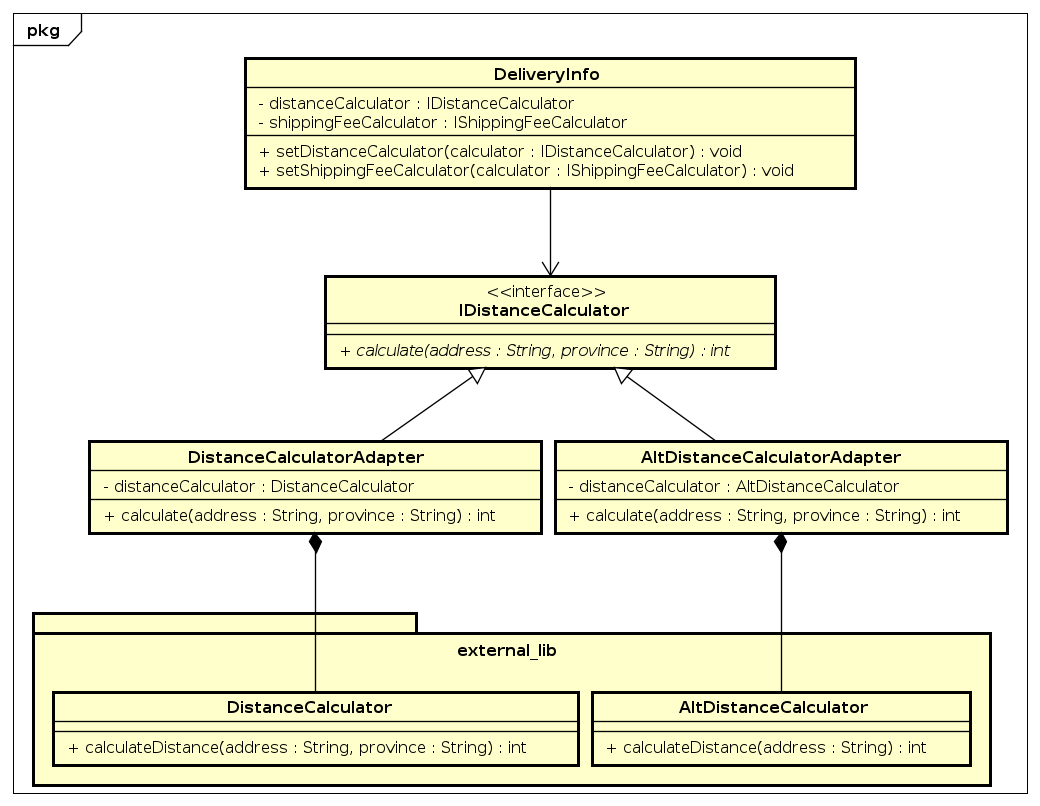


Hình ‑ Vấn đề thay đổi cách thức tính khoảng cách

### Giải pháp

Áp dụng Strategy và Adapter Pattern:

* Lớp DeliveryInfo thêm một thuộc tính distanceCalculator là đối tượng của interface IDistanceCalculator với phương thức calculate.
* Tạo lớp DistanceCalculatorAdapter thực thi IDistanceCalculator. Lớp DistanceCalculatorAdapter sẽ sử dụng thư viện distance-api.jar để tính khoảng cách trong phương thức calculate.
* Trong tương lai nếu cần thay đổi thư viện tính khoảng cách cần viết thêm adapter cho thư viện mới (adapter này sẽ thực thi IDistanceCalculator) và thay đổi thuộc tính distanceCalculator của lớp DeliveryInfo thành đối tượng của lớp adapter mới này.

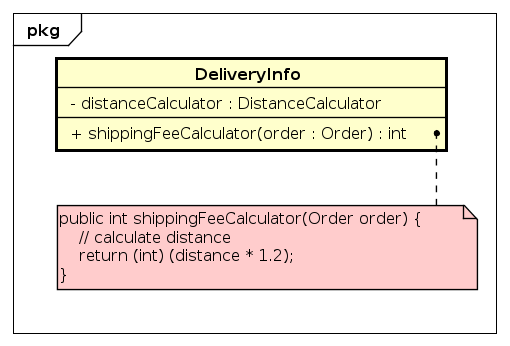


Hình ‑ Giải pháp vấn đề thay đổi cách thức tính khoảng cách

## Vấn đề thay đổi cách tính phí vận chuyển và giải pháp

### Vấn đề

Lớp DeliveryInfo hiện tại có phương thức calculateShippingFee để tính phí vận chuyển. Phương thức này đang sử dụng công thức: phí vận chuyển = (khoảng cách\*1.2). Trong tương lai khi thay đổi công thức tính phí vận chuyển sẽ phải thay đổi phương thức (vi phạm OCP).

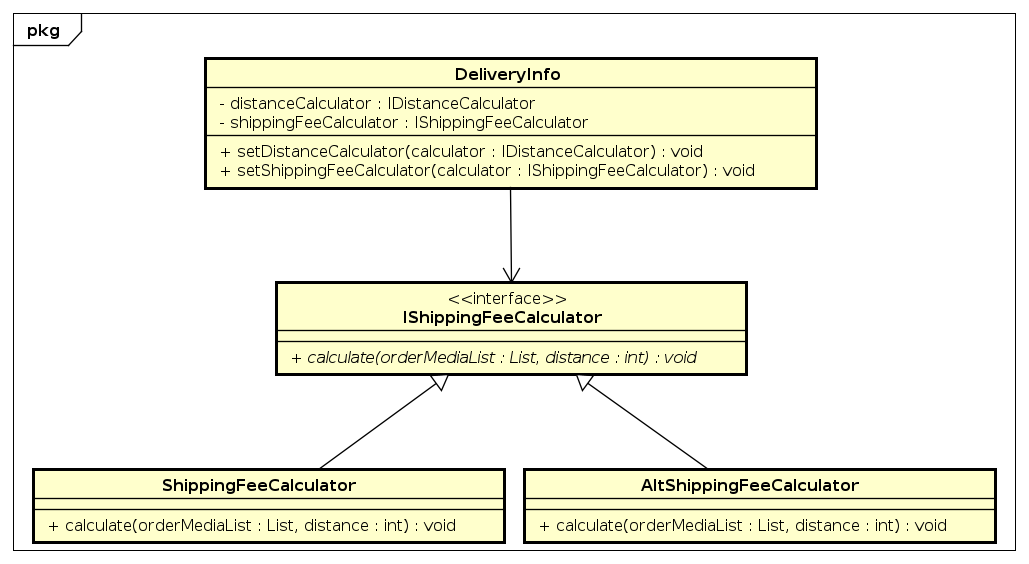


Hình ‑ Vấn đề thay đổi cách tính chi phí vận chuyển

### Giải pháp

Áp dụng Strategy Pattern:

* Lớp DeliveryInfo thêm thuộc tính shippingFeeCalculator là đối tượng của interface IShippingFeeCalculator với phương thức calculate.
* Tạo lớp ShippingFeeCalculator thực thi IShippingFeeCalculator. Lớp này sử dụng công thức cũ để tính phí trong phương thức calculate.
* Lớp DeliveryInfo tính phí vận chuyển bằng cách gọi shippingFeeCalculator.calculate()
* Trong tương lai khi thêm công thức tính phí vận chuyển chỉ cần viết lớp mới thực thi IShippingFeeCalculator. Nếu muốn thay đổi cách tính phí chỉ cần thay đổi thuộc tính shippingFeeCalculator của lớp DeliveryInfo.



Hình ‑ Giải pháp vấn đề thay đổi cách tính chi phí vận chuyển

## Vấn đề thêm một mặt hàng mới AudioBook

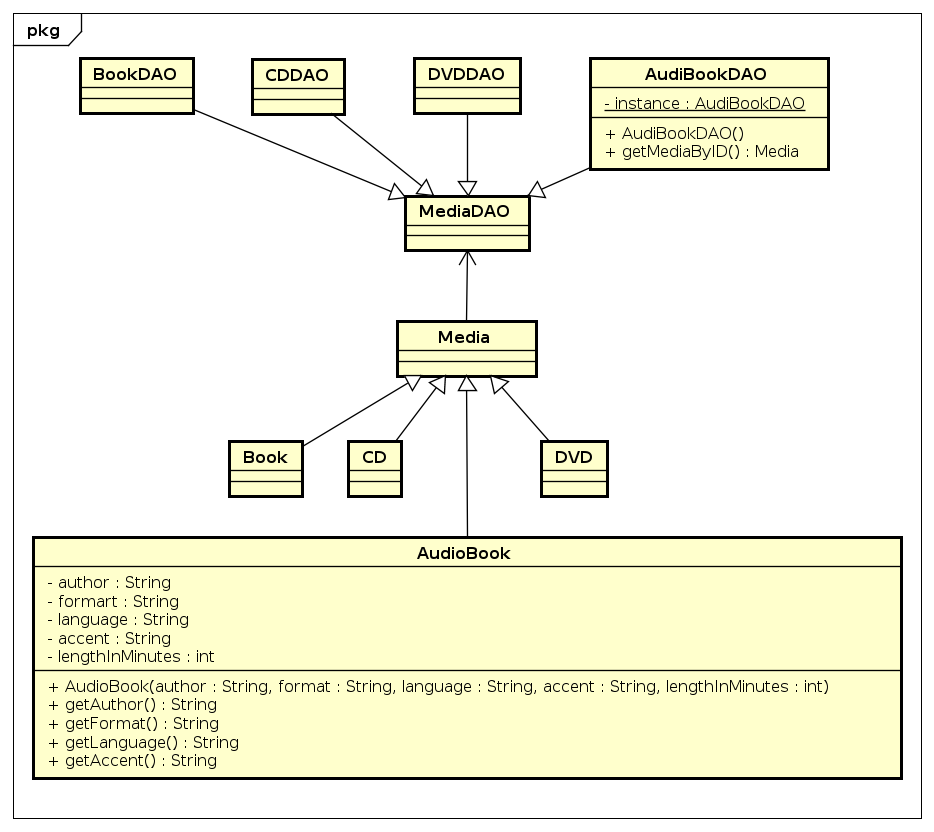
### Vấn đề

* Trong package entity/media dành cho các lớp media không có lớp đại diện cho mặt hàng AudioBook.
* Trong package dao/media thiếu lớp DAO dành cho việc truy vấn mặt hàng AudioBook trong cơ sở dữ liệu.
* Nếu sửa đổi các lớp khác trong hai package entity/media và dao/media để phù hợp việc xử lí thông tin AudioBook, nếu thay đổi code các lớp khác để xử lí thông tin lớp này ta dễ bị vi phạm nguyên lí OCP và SRP.

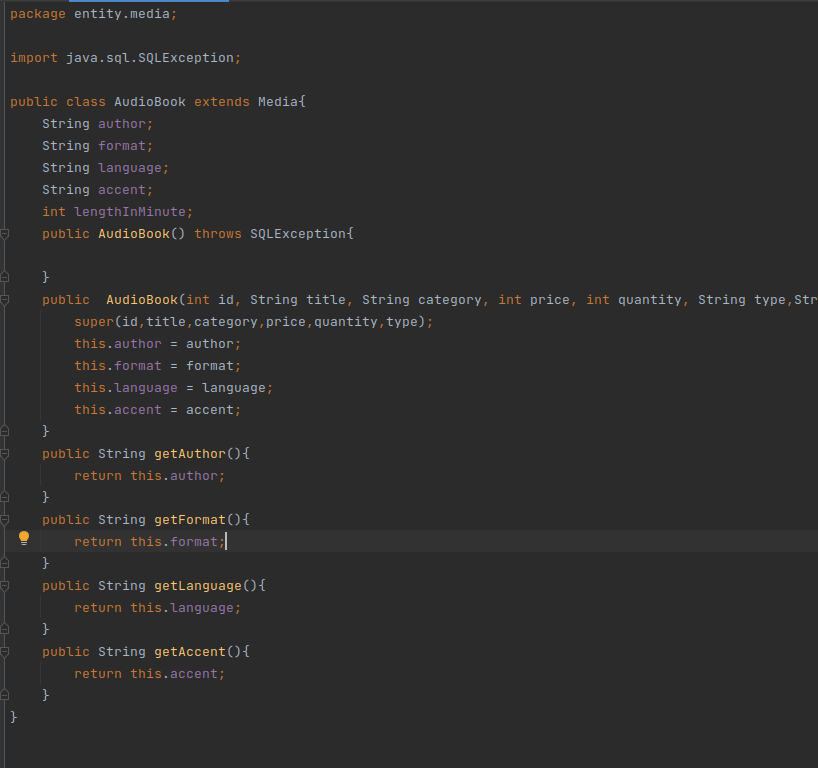
### Giải pháp

- Áp dụng mẫu thiết kế Singleton.

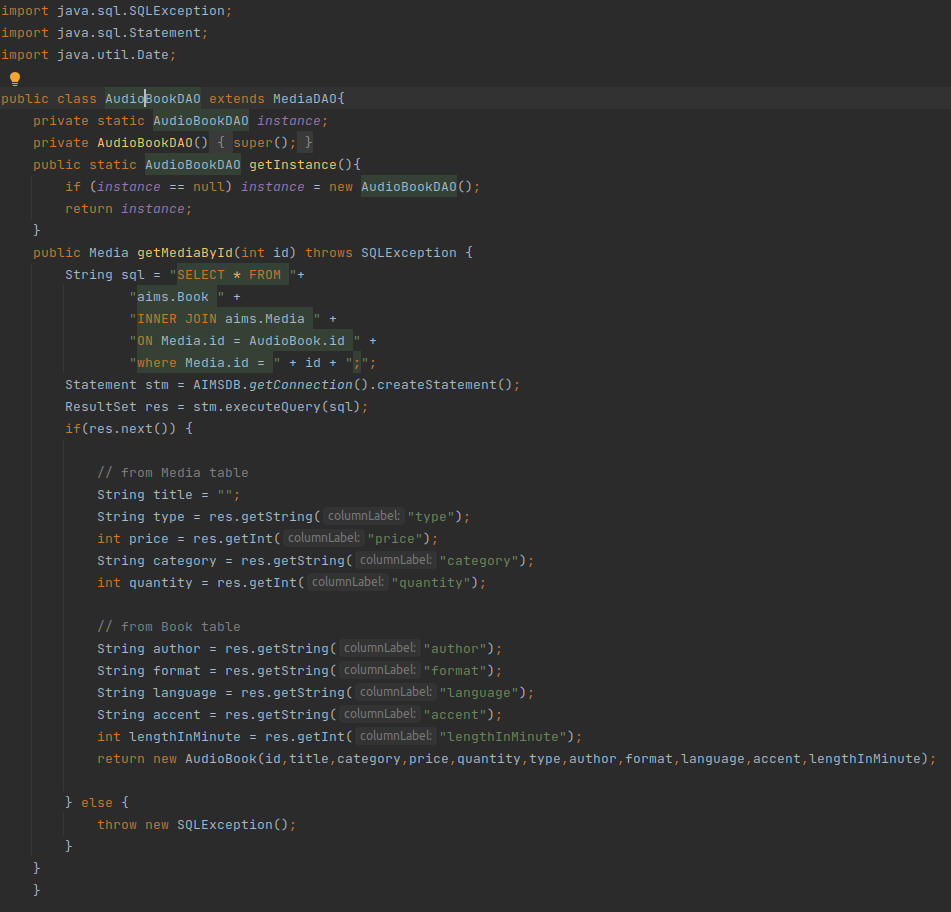
- Do mặt hàng AudioBook là dạng mặt hàng Media và thông tin có thể lưu trong bảng Medi trong cơ sở dữ liệu do cùng có một số thuộc tính như: id, title, price, quanity,… nên ta có thể xây dựng lớp AudioBook kế thừa lớp Media, lớp AudioBookDAO kế thừa lớp MediaDAO sử dụng thiết kế Singleton. Với cách thiết kế này, ta sẽ đảm bảo khi xây dựng lại hệ thống sẽ không phát sinh thêm các vi phạm nguyên tắc SOLID.



Hình ‑ Giải pháp vấn đề thêm một mặt hàng mới



Hình ‑ Minh họa code thêm một mặt hàng mới -1



Hình ‑ Minh họa code thêm một mặt hàng mới -2

## Vấn đề xem chi tiết đơn hàng và xóa những đơn hàng mà mà quản trị viên chưa chấp nhận

### Vấn đề

* Tại cơ sở dữ liệu cần có một trường thuộc tính để xác định xem đơn hàng đã được quản trị viên xác nhận hay chưa. Lớp Oder trong codebase chưa có thuộc tính thể hiện việc xác nhận này, Nếu sửa trực tiếp trong lớp Order ta dễ bị vi phạm OCP.
* Cần có lớp DAO,Controller trong codebase mà instance của nó để truy vấn thông tin trong bảng oder của database. Nếu xử lí bằng cách sửa các lớp DAO khác thì ta dễ vi phạm OCP, SRP.
* Cần có các lớp Screen và Handler để xử lí giao diện của xem đơn hàng, xử lí thao tác xóa đơn hàng để việc vi phạm nguyên tắc OCP,SRP là ít nhất trong codebase.

### Giải pháp

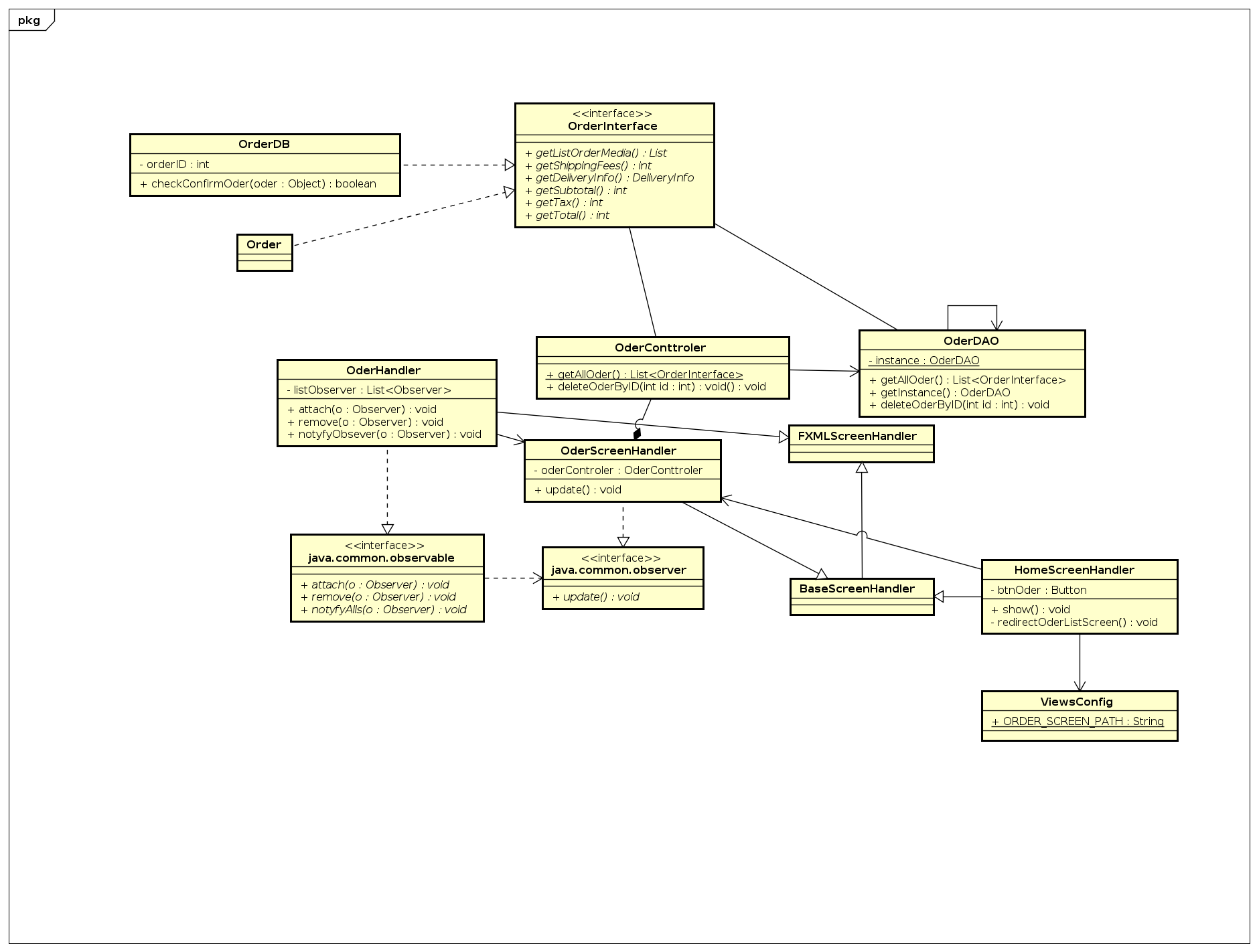
- Sử dụng các mẫu thiết kế như: Singleton, Observer.

- Đối với việc xây dựng một lớp Oder phù hợp cho việc lưu trữ thuộc tính xác định cho việc quản trị viên có xác nhận đơn hàng hay chưa, ta tạo một lớp OderInterface để các lớp khác giáo tiếp qua lớp OderInterface này mà không cần phân biệt các loại lớp Oder, giảm thiểu vi phạm SOLID.

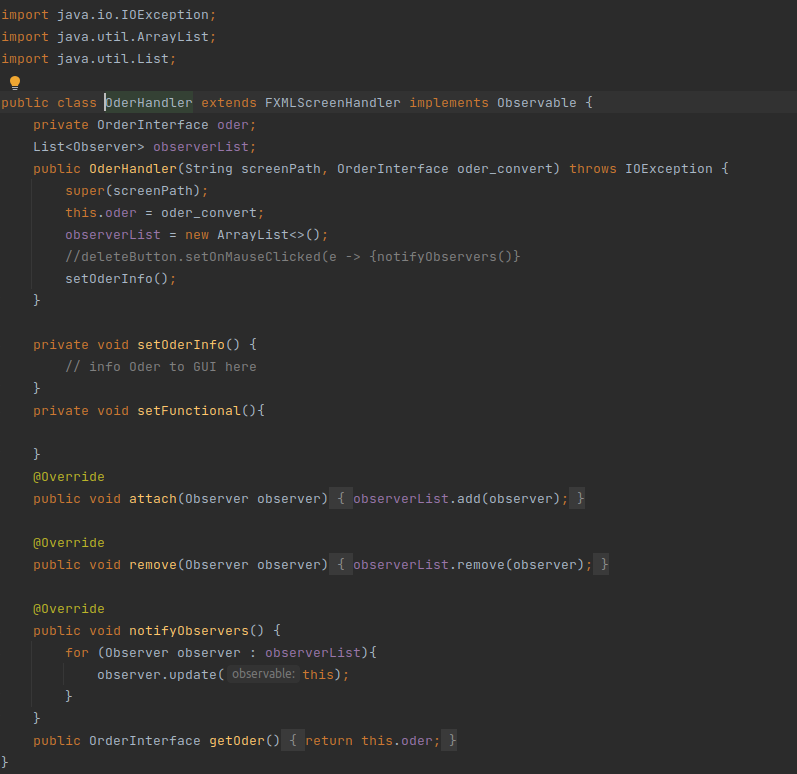
- Xây dựng lớp OderDAO với thiết kế Singleton để truy vấn câu lệnh sql Oder từ database.

- Lớp OderControler để kiểm soát xử lí với giao diện Oder.

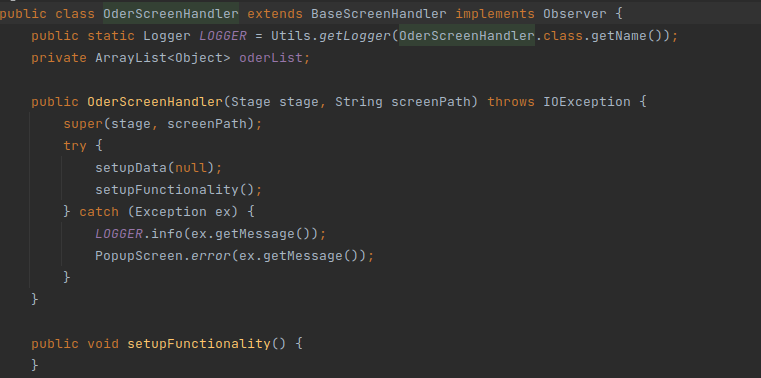
- Áp dụng mẫu Observer với hai lớp OderScreenHandler và lớp OderHandler để tạo giao diện xem chi tiết Oder, thực hiện xử lí tương tác với chức năng xóa Oder. Trong hàm update() của OderScreenHandler thì ta check điều kiện xem có xóa được hay không, nếu không thì bỏ qua.



Hình ‑ Giải pháp vấn đề xử lý đơn hàng



Hình ‑ Minh họa code vấn đề xử lý đơn hàng -1

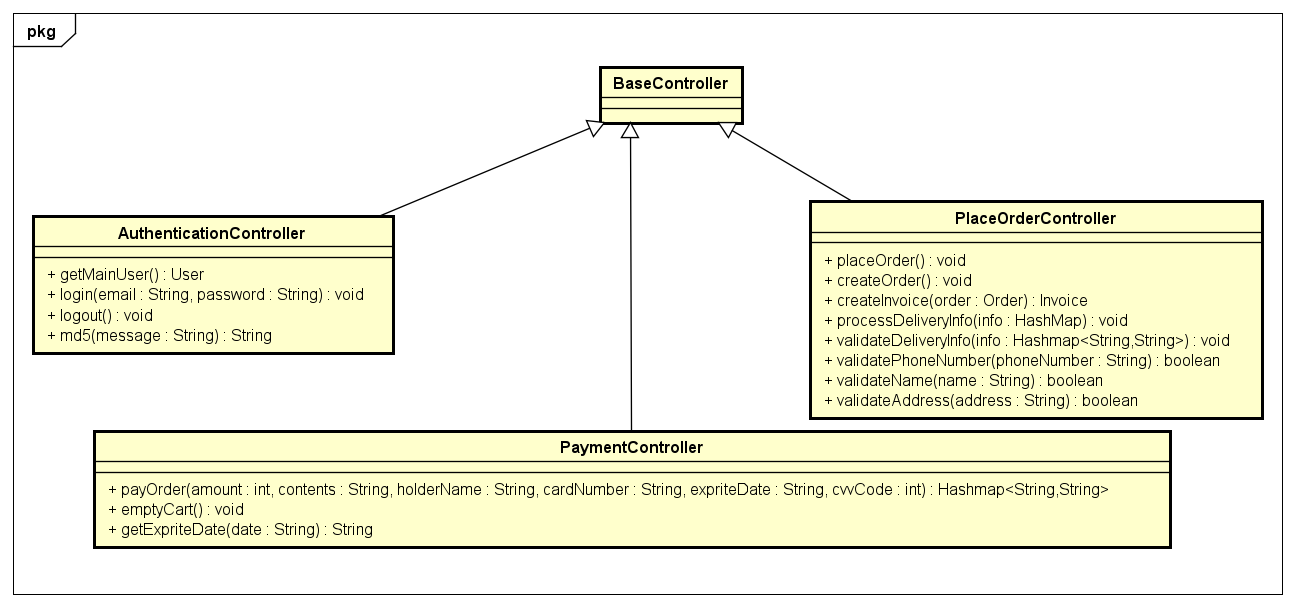


Hình ‑ Minh họa code vấn đề xử lý đơn hàng -2

## Vấn đề vi phạm SRP, design concepts và giải pháp

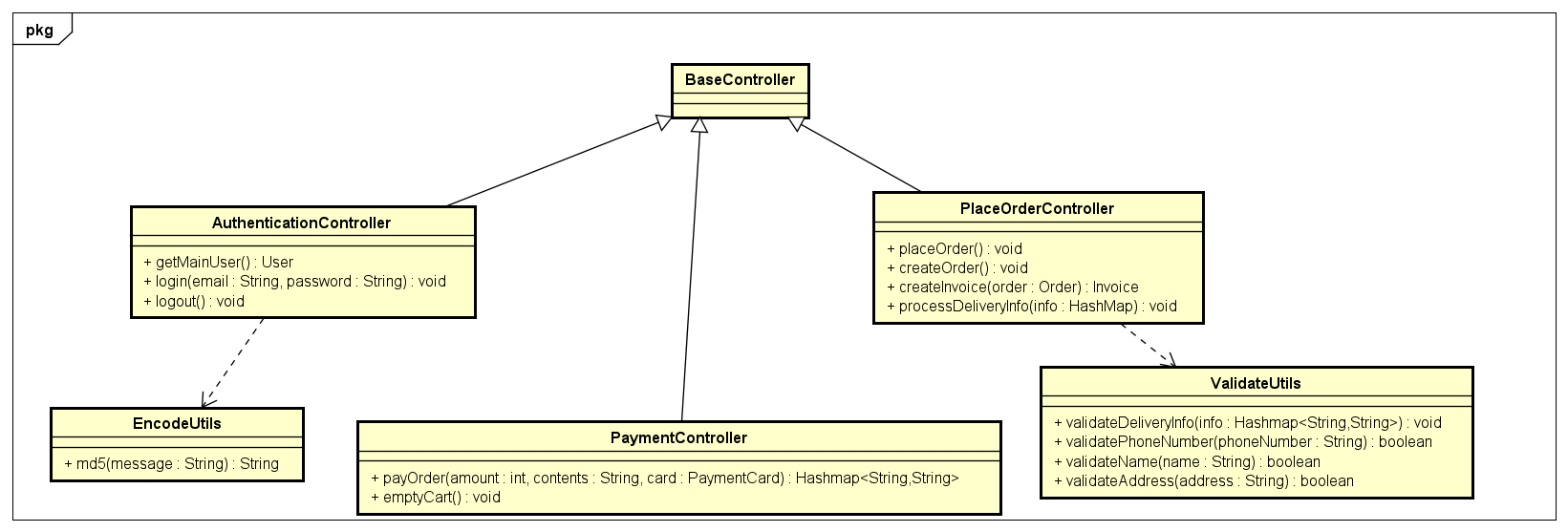
### Vấn đề

* AuthenticationController: Lớp này tồn tại nhiều hơn 1 lý do để thay đổi. Nếu thay đổi cách thức mã hóa( dùng base64, AES, SHA256...) sẽ phải thay đổi mã nguồn của lớp
* PlaceOrderController: Lớp này tồn tại nhiều hơn 1 lý do để thay đổi. Nếu thay đổi logic validate hoặc logic place order sẽ phải thay đổi mã nguồn của lớp
* PaymentController: Tồn tại nhiều hơn 1 lý do để thay đổi. Có phương thức getExpridate(String date) sẽ phải thay đổi khi định dạng date thay đổ
* Phương thức checkMediaInCart(Media media) vi phạm stamp coupling vì chỉ sử dụng id của media -> thay tham số Media media -> int mediaId



Hình ‑ Vấn đề vi phạm SRP

### Giải pháp

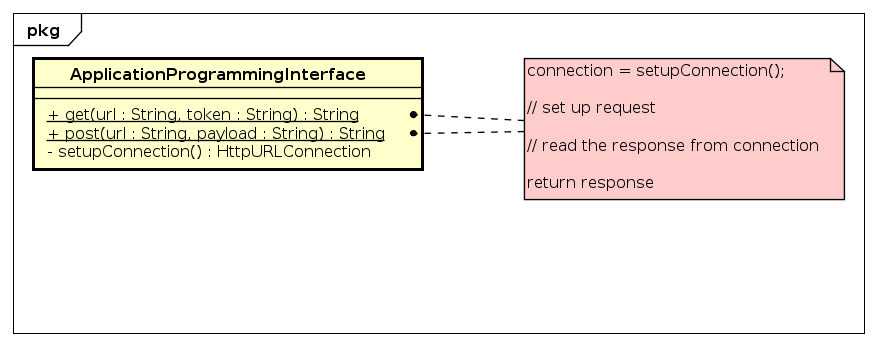
Nhóm các các phương thức validate, check, mã hóa ra thành các lớp utils riêng 

Hình ‑ Giải pháp vấn đề vi phạm SRP

## Vấn đề chưa clean code trong class ApplicationProgrammingInterface và giải pháp

### Vấn đề

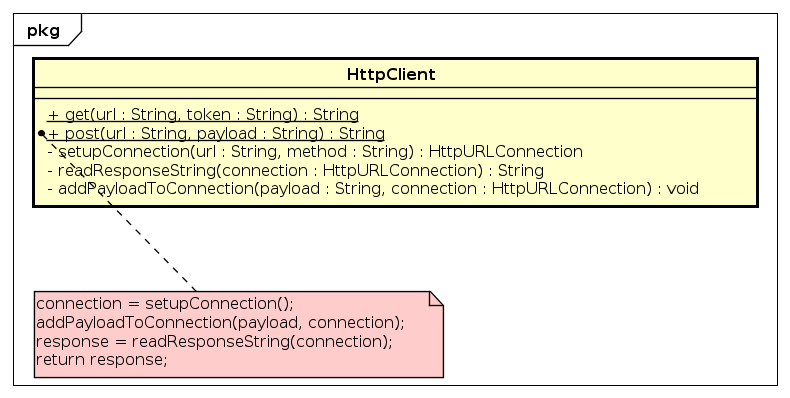
* Tên class không có tính gợi nhớ đến chức năng của lớp.
* Các phương thức get và post thực hiện nhiều hơn một nhiệm vụ (tạo request và đọc response).



Hình ‑ Vấn đề clean code

### Giải pháp

* Đổi tên class thành HttpClient.
* Tách đoạn code thực hiện tạo request trong phương thức post (thêm payload vào connection) thành phương thức addPayloadToConneciton.
* Tách đoạn code thực hiện đọc response thành phương thức readResponseString.



Hình ‑ Giải pháp vấn đề clean code

## Vấn đề vi pham nguyên tắc infomation Hiding đối với lớp SessionInfomation

**3.10.1 Vấn đề**

**.** Các thuộc tính của lớp SessionInfomation đang trong trạng thái mở với từ khóa public, việc để quyền public như nay vi phạm quy tắc infomation Hiding, các lớp khác có thể dễ dàng truy cập vào thuộc tính lớp SessionInfomation.

**3.10.2 Giải pháp**

**.** Xây dựng các hàm get và set cho lớp SessionInfomation, chuyển các thuộc tính khác là private.

4 Tổng kết

## 4.1 Kết quả tổng quan

So với dự kiến ban đầu nhóm hoàn thiện được 90% các yêu cầu dự kiến. Đã áp dụng được các OOP concepts, design pattern(singleton, strategy, adapter, observer) vào trong việc tái cấu trúc mã nguồn

## 4.2 Các vấn đề tồn đọng

Chưa thể áp dụng được nhiều design pattern vào trong project.

Một số vấn đề nhóm tìm ra trong design concepts nhưng chưa thể xử lý được:

* Coupling

Bảng ‑ Các vấn đề coupling chưa giải quyết

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Coupling* | *Module* | *Mô tả* | *Lý do* |
| 1 | Content  Coupling | ViewCartController | Trong các phương thức checkAvailabilityOfProduct(), getCartSubtotal() sử dụng trực tiếp biến cartInstance | Sử dụng trực tiếp dữ liệu của lớp khác, có thể làm thay đổi dữ liệu của lớp sử dụng |
| 2 | Control Coupling | InterbankPayloadConverter | extractPaymentTransaction(String responseText) có tham số truyền vào là responseText khi responseText là null thì hàm trả về null nếu không thực hiện tiếp công việc trích thông tin về transaction | Hàm phụ thuộc vào tham số truyền vào response và trả về các kết quả khác nhau. |
| 9 | Control  Coupling | PlaceOrderController | Phương thức validateDeliveryInfo và  ProcessDeliveryInfotruyền vào tham số là HashMap | Tham số là HashMap có thể ảnh hưởng đến luồng hoạt động của phương thức. Nếu trong tương lai lớp Delivery thay đổi thì cũng phải thì tham số là HashMap cũng phải thay đổi theo. Nếu bên sử dụng các phương thức này truyền vào HashMap có key không đúng có thể dẫn tới exception |

* Cohesion

Bảng ‑ Các vấn đề cohesion chưa giải quyết

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Cohesion* | *Module* | *Mô tả* | *Lý do* |
| 8 | Procedural Cohesion | MyMap | Các đoạn code trong phương thức toMyMap(String str, int idx) thực hiện theo thứ tự: check input -> get key -> get value | Các đoạn code liên quan với nhau về thứ tự thực hiện nhưng không liên quan về chức năng. |