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 02

Danh sách thành viên

|  |  |
| --- | --- |
| Hồ Đức Hân | 20194040 |
| Hà Mạnh Hùng | 20194066 |
| Trần Minh Hoàng | 20194061 |
| Trần Thành Long | 20194102 |

*Hà Nội, tháng 7 năm 2023*

Mục lục

Mục lục 1

1 Tổng quan 4

1.1 Mục tiêu 4

1.2 Phạm vi 4

*1.2.1* *Mô tả phần mềm* 4

*1.2.2* *Các chức năng chính của phần mềm* 4

*1.2.3* *Cấu trúc mã nguồn* 5

*1.2.4* *Các yêu cầu thêm cần cân nhắc cùng quá trình tái cấu trúc* 5

*1.2.5* *Các hoạt động thực thi trên mã nguồn* 6

*1.2.6* *Kết quả dự kiến* 6

1.3 Danh sách thuật ngữ 7

1.4 Danh sách tham khảo 7

2 Đánh giá thiết kế cũ 8

2.1 Nhận xét chung 8

2.2 Đánh giá các mức độ coupling và cohesion 8

2.2.1 Coupling 8

2.2.2 Cohesion 10

2.3 Đánh giá việc tuân theo SOLID 12

2.3.1 SRP 12

2.3.2 OCP 13

2.3.3 LIP 14

2.3.4 ISP 15

2.3.5 DIP 15

2.4 Các vấn đề về Clean Code 16

2.4.1 Clear Name 16

2.4.2 Clean Function/Method 16

2.4.3 Clean Class 17

3 Đề xuất cải tiến 18

3.1 Vấn đề Thêm mặt hàng Media mới: AudioBook (requirement 1) và giải pháp 18

3.1.1 *Vấn đề* 18

3.1.2 *Giải pháp* 18

3.2 Vấn đề Xem chi tiết sản phẩm (requirement 2) và giải pháp 19

*3.2.1* *Vấn đề* 19

*3.2.2* *Giải pháp* 20

3.3 Vấn đề Thay đổi thông báo lỗi (requirement 3) và giải pháp 21

*3.3.1* *Vấn đề* 21

*3.3.2* *Giải pháp* 21

3.4 Vấn đề Thay đổi cách tính khoảng cách (requirement 4) và giải pháp 22

*3.4.1* *Vấn đề* 22

*3.4.2* *Giải pháp* 22

3.5 Vấn đề Thêm hình thức thanh toán mới: Thẻ nội địa(Domestic Card) (requirement 5) và giải pháp 23

*3.5.1* *Vấn đề* 23

*3.5.2* *Giải pháp* 24

3.6 Vấn đề Thay đổi công thức tính phí vận chuyển (requirement 6) và giải pháp 25

*3.6.1* *Vấn đề* 25

*3.6.2* *Giải pháp* 26

3.7 Vấn đề Cập nhật lại chức năng hủy đơn hàng (requirement 7) và giải pháp 27

*3.7.1* *Vấn đề* 27

*3.7.2* *Giải pháp* 27

3.8 Vấn đề Common Coupling: SessionInformation, Cart và giải pháp 28

*3.8.1* *Vấn đề* 28

*3.8.2* *Giải pháp* 28

4 Tổng kết 30

4.1 Kết quả tổng quan 30

4.2 Các vấn đề tồn đọng 30

**Danh sách các minh họa**

**Error! No table of figures entries found.**

**Danh sách các bảng**

**Error! No table of figures entries found.**

# Tổng quan

## Mục tiêu

Mục đích của báo cáo mẫu thiết kế phần mềm là để đánh giá thiết kế cũ một cách tổng quan và chi tiết từng phần, bao gồm (i) đánh giá về mức độ coupling, cohesion, (ii) đánh giá việc tuân theo nguyên lý SOLID, (iii) đánh giá mức độ clean code. Đồng thời, trong bản báo cáo cũng đưa ra một số đề xuất cải tiến cho các vấn đề gặp phải liên quan đến các mục đánh giá ở trên.

Đối tượng hướng tới của báo cáo này là kỹ sư thiết kế, nhân viên phát triển và nhân viên bảo trì phần mềm, thông báo về những vấn đề gặp và đưa ra giải pháp sửa chữa.

## Phạm vi

### *Mô tả phần mềm*

Hệ thống thương mại điện tử AIMS, cho phép mua bán sản phẩm phương tiện truyền thông vật lý. 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*

* Chức năng Đăng nhập: Khi đăng nhập, hệ thống sẽ tự động lưu và lấy thông tin giao hàng mặc định, thông tin thẻ và sản phẩm hiện có trong giỏ hàng vào trong tài khoản khách hàng.
* Chức năng Hiển thị danh sách sản phẩm: Khi khởi động, hệ thống sẽ hiện ra danh sách của 20 sản phẩm bất kỳ. Mỗi sản phẩm trong danh sách bao gồm ảnh, tên sản phẩm, giá cả và số lượng sản phẩm hiện có trong kho.
* Chức năng tìm kiếm sản phẩm: cho phép người dùng tìm kiếm theo loại mặt hàng, bao gồm sách (Book), CD và DVD.
* Chức năng Đặt mua sản phẩm: Người dùng chọn số lượng và nhấn đặt mua để thêm mặt hàng vào giỏ hàng.
* Chức năng Xem giỏ hàng: Liệt kê danh sách các mặt hàng, số lượng đã đặt mua, chi phí cho mỗi mặt hàng. Hiển thị tổng giá tiền của toàn bộ giỏ hàng.
* Chức năng Hủy mua mặt hàng: Người dùng hủy mặt hàng sẽ loại bỏ mặt hàng đó khỏi giỏ hàng.
* Chức năng Thay đổi số lượng sản phẩm: Người dùng có thể thêm hoặc bớt số lượng sản phẩm định mua trong giới hạn nhất định, tối thiểu là 1 và tối đa không vượt quá số hàng trong kho.
* Chức năng Đặt hàng: Người dùng điền thông tin về tên, điện thoại, địa chỉ và các hướng dẫn thêm khi giao hàng. Hệ thống hiển thị hóa đơn và cho phép người dùng lựa chọn phương thức thanh toán phù hợp.

### *Cấu trúc mã nguồn*

Mã nguồn được chia thành 8 package lớn.

* Package default: gồm App, trong đó chứa hàm main(), là điểm bắt đầu của chương trình.
* Package common: khai báo các loại exceptions sử dụng trong chương trình, đồng thời chứa interface Observer và Observable dùng trong observer pattern.
* Package controller: bao gồm các controller tương ứng với các chức năng lớn của hệ thống.
* Package dao: chứa các data access object dùng để giao tiếp với cơ sở dữ liệu.
* Package entity: chứa các lớp đại diện cho các thực thế có trong hệ thống.
* Package subsystem: hệ thống con dùng để xử lý việc thanh toán bằng thẻ.
* Package utils: gồm các lớp cung cấp tiện ích như validate dữ liệu, cung cấp hàm hash, chuyển dữ liệu từ dạng json sang cấu trúc map.
* Package views: chứa các lớp xử lý giao diện.

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

Hệ thống có 7 yêu cầu thêm:

**Yêu cầu 1**: Thêm mặt hàng Media mới: AudioBook. AudioBook bao gồm các trường thông tin về author, format, language, accent, lengthInMinutes.

**Yêu cầu 2**: Thêm màn hình xem chi tiết sản phẩm

Tại màn hình chính (home screen), người dùng có thể xem danh sách các sản phẩm. Bên cạnh nút “Add to cart” tại mỗi sản phẩm, chương trình cần bổ sung thêm nút “View details”. Nếu khách hàng quan tâm sản phẩm nào, có thể chọn để xem chi tiết. Tại màn hình xem chi tiết, tất cả thông tin của sản phẩm được hiển thịđầy đủđể khách hàng xem xét, kèm theo lựa chọn số lượng và nút “Add to cart”.

**Yêu cầu 3**: Thay đổi yêu cầu khi load giao diện lỗi. Thay vì hiển thị thông báo lỗi trên màn hình popup, chương trình hiển thị trực tiếp lỗi màu đỏ trên phía trên cùng của các trang. Thay vì thông báo “Error when loading resources”, cần bổ sung thêm lỗi khi thao tác với resource nào, và cụ thể lỗi (ví dụ: không tìm thấy,...).

**Yêu cầu 4**: Thay đổi cách tính khoảng cách, sử dụng thư viện mới. Thay vì sử dụng thư viện cũ DistanceCalculator, ta sử dụng một thư viện mới là alt-distance-api.jar có interface gần giống với distance-api.jar.

**Yêu cầu 5**: Thêm phương thức thanh toán mới: Thẻ nội địa (Domestic Card). Loại thẻ này bao gồm các trường thông tin về type, issuring bank, card number, valid-from date. cardholder’s name. Với thẻ nội địa, chương trình vẫn kết nối với API của Interbank, chỉ thay đổi thông tin của phương thức thanh toán.

**Yêu cầu 6**: Thay đổi công thức tính phí vận chuyển. Phí vận chuyển sẽđược tính dựa theo khối lượng thực tế, khoảng cách, và độcồng kềnh của kiện hàng.Ví dụ: Kích thước của kiện hàng: chiều dài, chiều rộng, chiều cao. Độ cồng kềnh của kiện hàng có thể lấy tượng trưng: Độ cồng kềnh (kg) = Chiều dài (cm) x Chiều rộng (cm) x Chiều cao (cm) / 6000. Khối lượng và độ cồng kềnh (đơn vị: kg) lấy hệ số là 10 (đ/kg); khoảng cách (đơn vị: km, mặc định trả về từ thư viện Distance-API) lấy hệ số là 1.

**Yêu cầu 7**: Cập nhật lại chức năng hủy đơn hàng. Chức năng hủy đơn hàng sẽ bị vô hiệu hóa sau khi đơn hàng đã được quản trị viên chấp nhận. Nếu đơn hàng vẫn đang ở trạng thái chờ phê duyệt, khách hàng vẫn có thể hủy đơn hàng như bình thường.

### *Các hoạt động thực thi trên mã nguồn*

Thực hiện việc đánh giá về mức độ coupling, cohesion, mức độ tuân theo nguyên lý SOLID của thiết kế trong code base, tìm kiếm những nơi xảy ra code smell, đánh giá mức độ sạch của mã nguồn theo 3 mức độ: clear name, clear method và clear class.

Đưa ra các giải pháp cho vấn đề coupling, cohesion, vi phạm SOLID và làm sạch code bằng cách refactor.

Đưa ra giải pháp về việc thay đổi theo yêu cầu thêm.

### *Kết quả dự kiến*

Mã nguồn hoạt động tốt, chương trình vận hành bình thường, không có lỗi phát sinh thêm.

Làm sạch code, giảm coupling về mức data coupling và tăng cohesion lên mức functional cohesion trên tất cả các lớp trong code base.

Áp dụng triệt để nguyên lý SOLID vào mã nguồn.

Tái cấu trúc mã nguồn sao cho các yêu cầu thêm có thể thực hiện dễ dàng, không gây ra thêm coupling, cohesion và vi phạm nguyên lý SOLID.

## Danh sách thuật ngữ

1. **SOLID**: là viết tắt của 5 chữ cái đầu trong 5 nguyên lý thiế kế SRP, OCP, LSP, ISP, DIP.
2. **SRP**: Single responsibility principle
3. **OCP**: Open/closed principle
4. **LSP**: Liskov substitution principle
5. **ISP**: Interface segregation principle
6. **DIP**: Dependency inversion principle

## 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

# Đánh giá thiết kế cũ

## Nhận xét chung

Mã nguồn hiện tại hoàn toàn có thể giúp hệ thống vận hành một cách bình thường. Tuy nhiên thiết kế cũ còn tồn đọng khá nhiều vi phạm về các nguyên lý thiết kế SOLID, Clean Code và chưa áp dụng nhiều design pattern dẫn đến trong tương lai khi bổ xung thêm các tính năng mới hoặc tái sử dụng lại sẽ gây khó khăn cho những người phát triển sau.

## Đánh giá các mức độ coupling và cohesion

Coupling xác định mức độ phụ thuộc giữa các thành phần trong hệ thống. Coupling thấp tạo ra tính linh hoạt, tái sử dụng và dễ bảo trì. Cohesion xác định mức độ tương tác và sự liên kết giữa các thành phần trong một nhóm. Cohesion cao tạo ra tính tổ chức, hiệu suất và dễ đọc của mã nguồn. Coupling và cohesion cùng đóng vai trò quan trọng trong việc xây dựng hệ thống hướng đối tượng mạnh mẽ và dễ quản lý.

### Coupling

Về mức độ package thì nhìn chung đảm bảo sự phụ thuộc giữa các package là thấp nhưng với mức độ class và method thì một số module vẫn còn phụ thuộc nhau.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Coupling* | *Module* | *Mô tả* | *Lý do* |
| 1 | Common coupling | Controller/AuthenticationController | Các phương thức trong class này như getMainUser, login, logout đều sử dụng, chia sẻ chung global data là expired Time, mainUser thuộc class sessionInformation | Common coupling sẽ làm ảnh hưởng tới việc kiểm soát dữ liệu, khó xác định được tất cả các nguồn ảnh hưởng tới dữ liệu, và khó trong việc tái sử dụng các phương thức bị common coupling |
| 2 | Common coupling | Controller/ BaseController | Các phương thức trong class này như checkMediaInCart(), getListCartMedia() đều sử dụng, chia sẻ chung global data là cartInstance thuộc class sessionInformation | Common coupling sẽ làm ảnh hưởng tới việc kiểm soát dữ liệu, khó xác định được tất cả các nguồn ảnh hưởng tới dữ liệu, và khó trong việc tái sử dụng các phương thức bị common coupling |
| 3 | Common coupling | Controller/ PlaceOrderController | Các phương thức trong class này như placeOrder(), createOrder() đều sử dụng, chia sẻ chung global data là cartInstance thuộc class sessionInformation | Common coupling sẽ làm ảnh hưởng tới việc kiểm soát dữ liệu, khó xác định được tất cả các nguồn ảnh hưởng tới dữ liệu, và khó trong việc tái sử dụng các phương thức bị common coupling |
| 4 | 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. |
| 5 | Stamp Coupling | Views/Screen/Home/ LoginScreenHandler | Phương thức setupData truyền vào dto nhưng không sử dụng | Stamp coupling sẽ gây ảnh hưởng đến phương thức khi mà dữ liệu truyền vào thay đổi cấu trúc dữ liệu thì sẽ ảnh hưởng tới phương thức đó truy cập dữ liệu |
| 6 | Stample Coupling | Order | phương thức khởi tại Order(Cart cart) vi phạm | Cả phương thức chỉ cần sử dụng kết quả của cart.calSubtotal() nhưng lại truyền tham chiếu cả đối tượng Cart |
| 7 | Stample Coupling | Cart | Phương thức checkMediaInCart(Media media) | Biến media chỉ sử dụng 1 lần để lấy mediaId nhưng lại truyền cả đối tượng media |
| 8 | Stample Coupling | DeliveryInfo | Phương thức calculateShippingFee(Order order) | Biến order truyền vào nhưng không được sử dụng |
| 9 | Common Coupling | SessionInformation |  | Các thuộc tính đều ở dạng public, tuy nhiên không cung cấp các phương thức get và set để thay đổi chúng nên các module sử dụng có thể truy cập trực tiếp vào thuộc tính để thay đổi. Gây ra cả content và common ở một số module |
| 10 | Content coupling | Cart | Phương thức getListMedia() { return lstCartItem; } | Việc sử dụng method này ở những nơi khác có thể trực tiếp thay đổi Cart.lstCartItem mà không cần thông qua Cart |

### Cohesion

Đánh giá chung, mức độ liên kết chặt chẽ giữa các submodule chưa cao.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Cohesion* | *Module* | *Mô tả* | *Lý do* |
| 1 | Coincidental Cohesion | Controller/Auth enticationContr oller | Phương thức md5 nên để vào class utils để controller gọi đến khi sử dụng | Vì các phương thức không liên quan được đặt vào trong class nên dẫn tới việc quản lý và bảo trì khó khăn |
| 2 | Coincidental Cohesion | Controller/Place OrderController | Phương thức processDeliveryInfo không liên quan đến nghiệp vụ của class, nên tách ra một module riêng | Vì các phương thức không liên quan được đặt vào trong class nên dẫn tới việc quản lý và bảo trì khó khăn |
| 3 | Coincidental Cohesion | Controller/Place OrderController | Vì các phương thức validate như validateDeliveryInfo, validatePhoneNumber, validateName, validateAddress cùng xử lý logic là validate nên ta cần tách ra một module riêng để xử lý và có thể tái sử dụng được các hàm validate | Vì các phương thức không liên quan được đặt vào trong class nên dẫn tới việc quản lý và bảo trì khó khăn |
| 4 | Coincidental Cohesion | subsystem/inter bank/InterbankP ayloadConverte r | phương thức getToday không liên quan tới mục đích của class | Khó bảo trì, khó tái sử dụng khi phương thức này nằm ở trong một class không thể hiện mục đích của nó |
| 5 | Logical Cohesion | views/screen/po pup/PopupScree n | 2 phương thức success và error chỉ cùng thực hiện 1 thể loại công việc | Khó bảo trì và tái sử dụng đối với mỗi phương thức riêng lẻ |
| 6 | 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) |
| 7 | Procedural Cohesion | Controller/Auth enticationContr oller | Phương thức logout được nhóm lại trong class này để theo tuần tự thực thi login->logout |  |
| 8 | Procedural Cohesion | entity/media/ Media | Các khối lệnh trong phương thức getQuantity chỉ liên kết với nhau vì nằm trong cùng 1 trình tự thực hiện cập nhật số lượng và lấy ra số lượng | Các hành động chưa liên kết chặt chẽ với nhau, tính tái sử dụng không cao khi chỉ muốn thực hiện lấy ra số lượng mà không cập nhật |
| 9 | Communicat ion Cohesion | entity/order/ Order | Các khối lệnh trong phương thức setDeliveryInfo chỉ liên kết với nhau bởi dữ liệu dùng chung | Tính tái sử dụng chưa cao khi chỉ muốn đặt thông tin vận chuyển mà không thực hiện tính phí vận chuyển |

## 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

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | 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 |
| 2 | Controller/Authent icationController | Class AuthenticationController thực hiện nhiều hơn một nhiệm vụ như là vừa xác thực vừa mã hóa dữ liệu | Vì các nhiệm vụ trong class này không có sự gắn kết liên quan đến nhau nên độ cohesion kém vì mong muốn chúng ta cần thiết kế mã nguồn sao cho độ cohesion trong 1 module càng cao càng tốt |
| 3 | Controller/PlaceOr derController | Class PlaceOrderController thực hiện nhiều hơn một nhiệm vụ như là vừa điều khiển luồng dữ liệu: tạo đơn hàng, tạo hóa đơn, xử lý thông tin đơn, vừa validate dữ liệu | Vì có nhiều nhiệm vụ trong một class nên khi cần thay đổi hoặc phát triển thì code trong class càng ngày càng rắc rối dẫn tới khó quản lý và bảo trì hơn |
| 4 | Controller/Paymen tController | Class này thực hiện nhiều hơn một nhiệm vụ vừa thực hiện validate dữ liệu date input, vừa thực hiện thanh toán qua phương thức pay Order | Khó tái sử dụng những hàm validate dữ liệu, giảm độ cohesion của module |
| 5 | Utils/ApplicationP rogrammingInterfa ce | Class thực hiện đến nhiều hơn một nhiệm vụ, vừa thực hiện get(), post() vừa tạo ra kết nối HTTP thông qua phương thức setupConnection(), đồng thời cũng sử dụng method allowMethod() cho việc điều khiển truy cập | Vì các nhiệm vụ trong class này không có sự gắn kết liên quan đến nhau nên độ cohesion kém vì mong muốn chúng ta cần thiết kế mã nguồn sao cho độ cohesion trong 1 module càng cao càng tốt đồng thời rất khó tái sử dụng code nếu sau này có các class khác mới cũng cần các phương thức connection() hay allowMethod() thì lại phải viết lại các method đó |

### OCP

*<Nhận xét chung về tình trạng thiết kế và mã nguồn trên nguyên lý SRP>*

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | PaymentControler | Kiểm soát giao dịch | Khi đổi thẻ ATM khi từ CreditCard-> Domestic Card thì phải thay đổi code. |
| 2 | PaymentTransaction | Thông tin giao dịch | Khi đổi thẻ ATM khi từ CreditCard-> Domestic Card thì phải thay đổi code. |
| 3 | InterBankSubSystem | Giao tiếp với interbank để thực hiện giao dịch | Khi đổi thẻ ATM khi từ CreditCard-> Domestic Card thì phải thay đổi code. |
| 4 | InterbankSubsystemController | Kiểm soát interbanksubsystem | Khi đổi thẻ ATM khi từ CreditCard-> Domestic Card thì phải thay đổi code. |
| 5 | DeliveryInfo | Xử lý dữ liệu vận chuyển | Khi thay đổi phương thức tính toán DistanceCalculator hoặc cách tính phí vận chuyển phải sửa code trong class. |

### LIP

*<Nhận xét chung về tình trạng thiết kế trên nguyên lý LIP>*

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | Controller/Authe nticationControll er | Class AuthenticationController kế thừa BaseController nhưng lại không thực hiện các hành vi phương thức của BaseController | AuthenticationController không thể thay thế hoàn toàn BaseController, không thể thực hiện được toàn bộ các yêu cầu trong BaseController |
| 2 | Controller/Paym entController | Class PaymentController kế thừa BaseController nhưng lại không thực hiện các hành vi phương thức của BaseController | PaymentController không thể thay thế hoàn toàn BaseController, không thể thực hiện được toàn bộ các yêu cầu trong BaseControlle |

### ISP

Mã nguồn đã tuân thủ nguyên lý thiết kế ISP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 |  |  |  |
| 2 |  |  |  |

### DIP

*<Nhận xét chung về tình trạng thiết kế trên nguyên lý DIP>*

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | 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. |
| 2 | PaymentControler | Kiểm soát giao dịch | Phụ thuộc CreaditCard |
| 3 | PaymentTransaction | Thông tin giao dịch | Phụ thuộc CreaditCard |
| 4 | InterBankSubSystem | Giao tiếp với interbank để thực hiện giao dịch | Phụ thuộc CreaditCard |
| 5 | InterbankSubsystemController | Kiểm soát interbanksubsystem | Phụ thuộc CreditCard |
| 6 | DeliveryInfo | Xử lý dữ liệu vận chuyển | Phụ thuộc vào DistanceCalculator là 1 lớp cấp thấp |
| 7 | ShippingConfigs | Thông tin các tỉnh thành | Lớp phụ thuộc vào các giá trị cụ thể, thông tin tỉnh thành cần được trừu tượng hóa |

## Các vấn đề về Clean Code

### Clear Name

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Giải pháp* |
| 1 | InterbankSybsytem | Sử dụng từ viết tắt để đặt tên biến | Thay thế tên biến *ctrl* thành *interbankSubsustemControler* |
| 2 | BaseScreenHandler | Sử dụng tên viết tắt để đặt tên biến, tên hàm | Thay thế tên biến bControler thành baseControler cùng các hàm getter setter của nỏ |
| 3 | MediaHandler | Đặt tên class trong thư mục view trùng nhau ở cart, home | Sửa tên class MediaHomeHandler |

### Clean Function/Method

|  |  |  |
| --- | --- | --- |
| *Module* | *Mô tả* | *Giải pháp* |
| PaymentController | GetExprirationDate ở mức 2 level of abstraction | Tách thành hàm isValidDatetime(int month, int year) để kiểm tra thánh năm hợp lệ |
| InterbankPayloadConverter | Vi phạm Data-Level Refactoring | Thay1.2 thành hằng số DINSTANCE\_FACTORY |
| PopPupScreen | Vi phạm Data-Level Refactoring | Thay 0.8 thành hàng số CLOSE\_TIME |
| InterbankPayloadConverter | Statement-Level Refactoring | Thay các điều kiện trong switch thành hàng số |

### Clean Class

*<Tương tự>*

# Đề xuất cải tiến

<*Mục này đưa ra những đề xuất cần cải tiến để khắc phục những vấn đề trên nếu có những yêu cầu phát sinh. Lưu ý, chỉ là cải tiến cho thiết kế và mã nguồn ban đầu để trong tương lai nếu có những yêu cầu phát sinh đưa ra thì không vi phạm hoặc ít vi phạm nhất các nguyên lý thiết kế SOLID đã nêu ở trên, đồng thời đảm bảo thiết kế đạt Low Coupling và High Cohesion*>

*<Phần này có thể gom một số vấn đề liên quan đã nêu ở phần 2 để đưa ra giải pháp giải quyết các vấn đề này. Mỗi mục con trong phần này là 1 vấn đề đã gom lại và đưa ra giải pháp tương ứng>*

*<Lưu ý, nhóm khi đưa ra đề xuất cần thể hiện các ý tưởng thiết kế dưới dạng biểu đồ lớp, biểu đồ tương tác và minh hoạ mã nguồn/ý tưởng về mã nguồn trong báo cáo; đồng thời, nhóm cần tái cấu trúc mã nguồn trên github tương ứng với giải pháp thiết kế đã đề xuất>.*

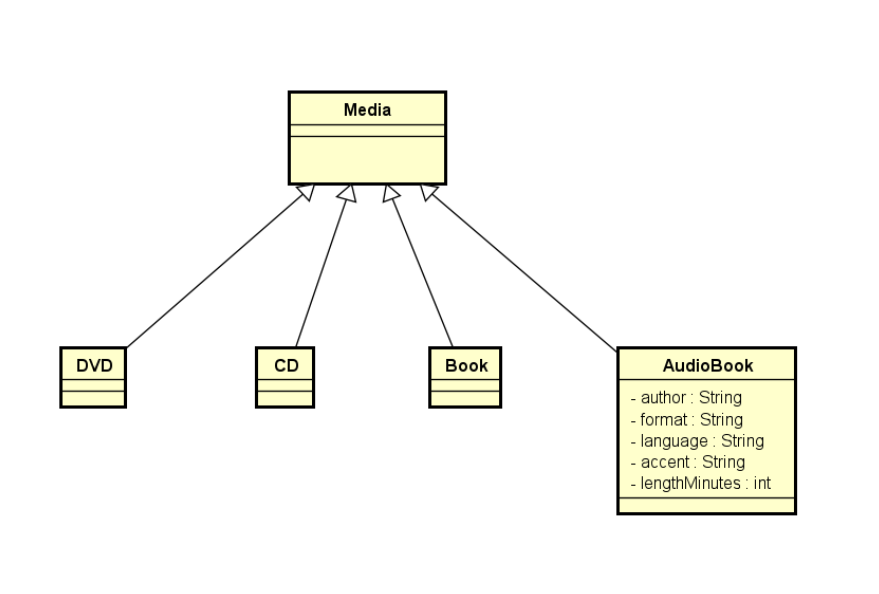
## Vấn đề Thêm mặt hàng Media mới: AudioBook (requirement 1) và giải pháp

### *Vấn đề*

Thêm mặt hàng Media mới: AudioBook. AudioBook bao gồm các trường thông tin về author, format, language, accent, lengthInMinutes.

### *Giải pháp*

Tạo ra lớp mới AudioBook kế thừa từ lớp Media đã có sẵn.



Mô tả mã nguồn:

public class AudioBook extends Media{

private String author;

private String format;

private String language;

private String accent;

private int lengthMinutes;

// contructor

// getter and setter

}

## Vấn đề Xem chi tiết sản phẩm (requirement 2) và giải pháp

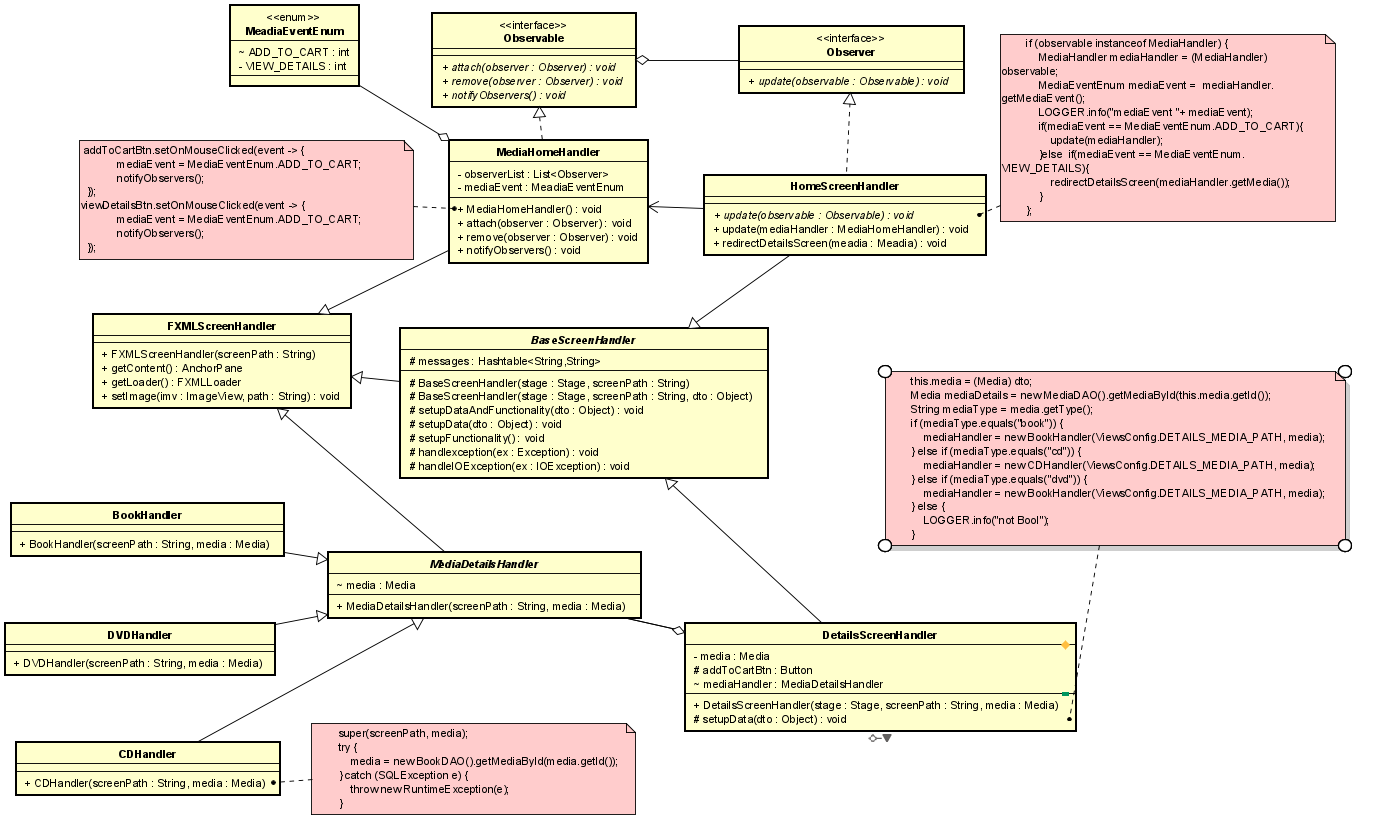
### *Vấn đề*

Việc sử dụng Observer pattern trong thiết kế hiện tại thì **home.MediaHandler** trong **HomeScreenHandler** hiện tại chỉ có thể sử dụng cho một mục đích là **addToCart** thêm hàng vào Cartnếu theo thiết kế ban đầu ta sẽ không triển khai thêm chức năng **viewDetails**  để xem chi tiết màn hình.

Ngoài ra class home.MediaHandler có cùng tên với class cart.MediaHandler dễ ngây nhầm lẫn

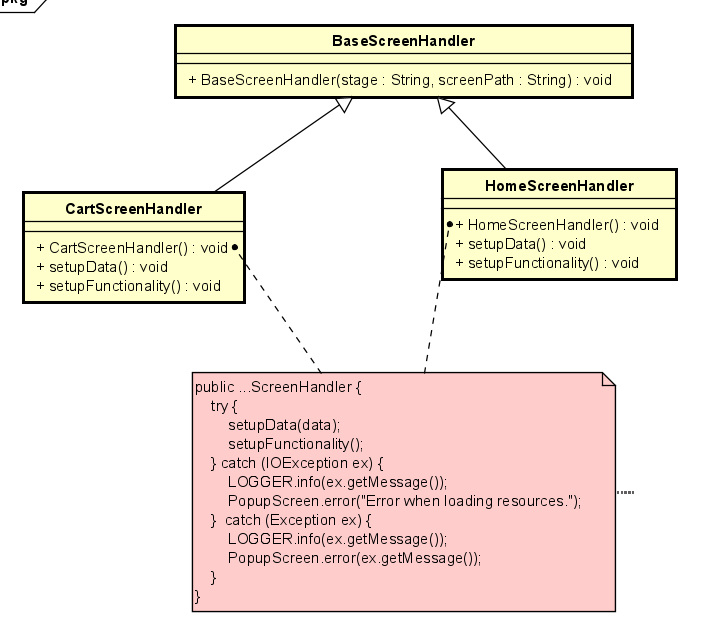
### *Giải pháp*

* Clean Name Code: đổi tên class **home.MediaHandler** thành **MediaHomeHandler**
* Thêm 1 class **MediaEvenEnum** là thuộc tính trong **MediaHomeHandler** nhằm để có thể xác định các ý định của **MediaHomeHandler** quan sát cung cấp cho **HomeScreenHandler**
* Sử dụng FactoryMethod để triển khai các màn hình chi tiết cho các loại như DVD, Book, CD...



## Vấn đề Thay đổi thông báo lỗi (requirement 3) và giải pháp

### *Vấn đề*



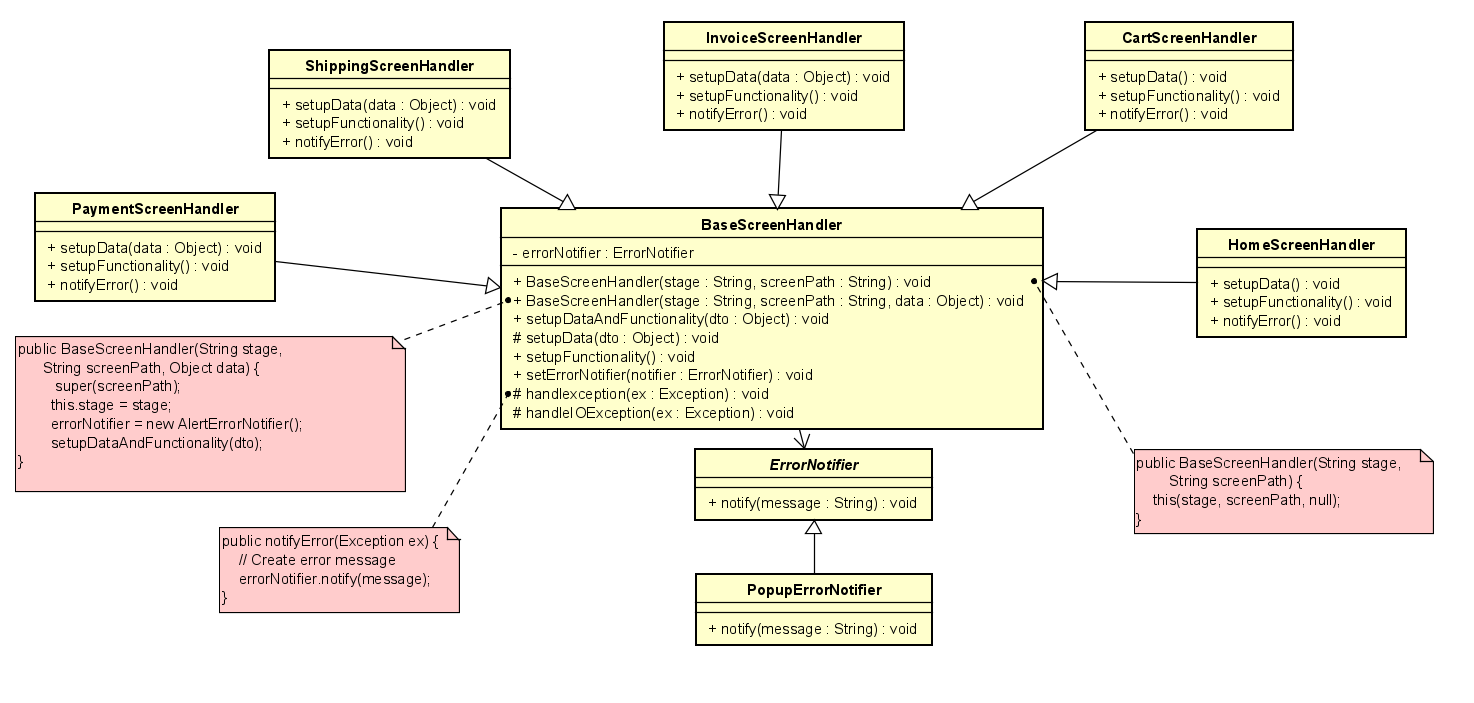
Các lớp con của **BaseScreenHandler** có sự lặp code, xử lý code về qui trình trong phương thức khởi tạo. Những **ScreenHandler** có thể không cần sử dụng phương thức đó nhưng vẫn được triển khai trong mã nguồn

Khi muốn thay đổi cách hiển thị lỗi PopupScreen ta phải trực tiếp sửa đổi cho từng đoạn code.

### *Giải pháp*

* Tạo 1 Class cha ErrorNotififier để kế thừa làm thuộc tính của lớp **BaseScreenHandler** để xác định kiểu thông báo lỗi khi có sự cố
* Sử dụng template method kết hợp trên lớp **BaseScreenHandler** tối ưu mã nguồn hơn khi triển khai mở rộng.

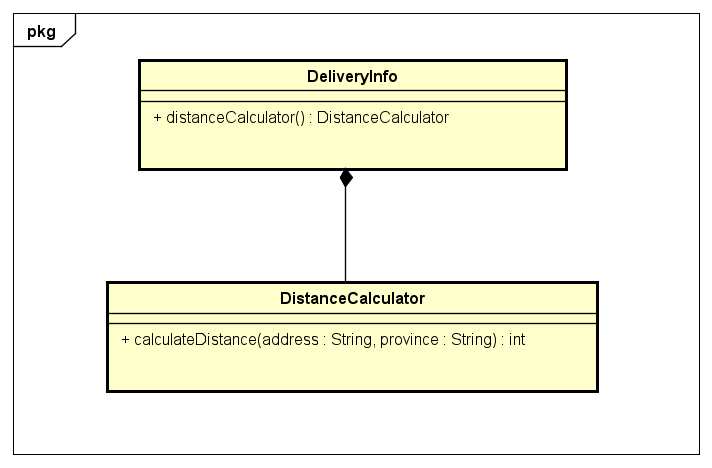
Với thiết kế mới, các phương thức setupData, setupFunctionality sẽ được gọi ở lớp cha. Lớp con chỉ làm nhiệm vụ là override các phương thức setup trên nếu cần. Với mỗi lớp con, khi cần hiển thị những lỗi riêng thì cần phải override lại phương thức getErrorMessage



## Vấn đề Thay đổi cách tính khoảng cách (requirement 4) và giải pháp

### *Vấn đề*

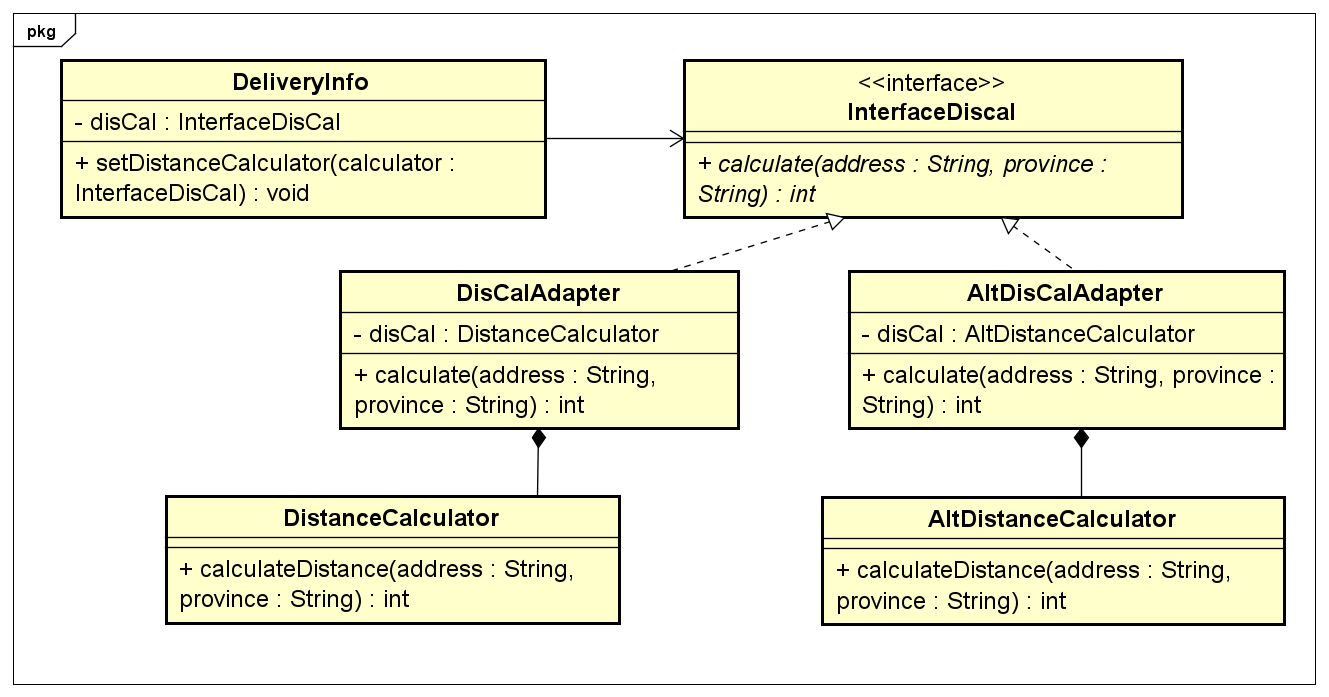
Lớp DeliveryInfo hiện chỉ đang sử dụng thư viện cũ là distance-api.jar với lớp DistanceCalculator. Khi muốn thêm cách tính khoảng cách mới, cần thêm thư viện mới alt-distance-api.jar với interface khác với thư viện cũ thì cần phải sửa lại code nên thiết kế cũ đã vi phạm nguyên lý OCP.



### *Giải pháp*

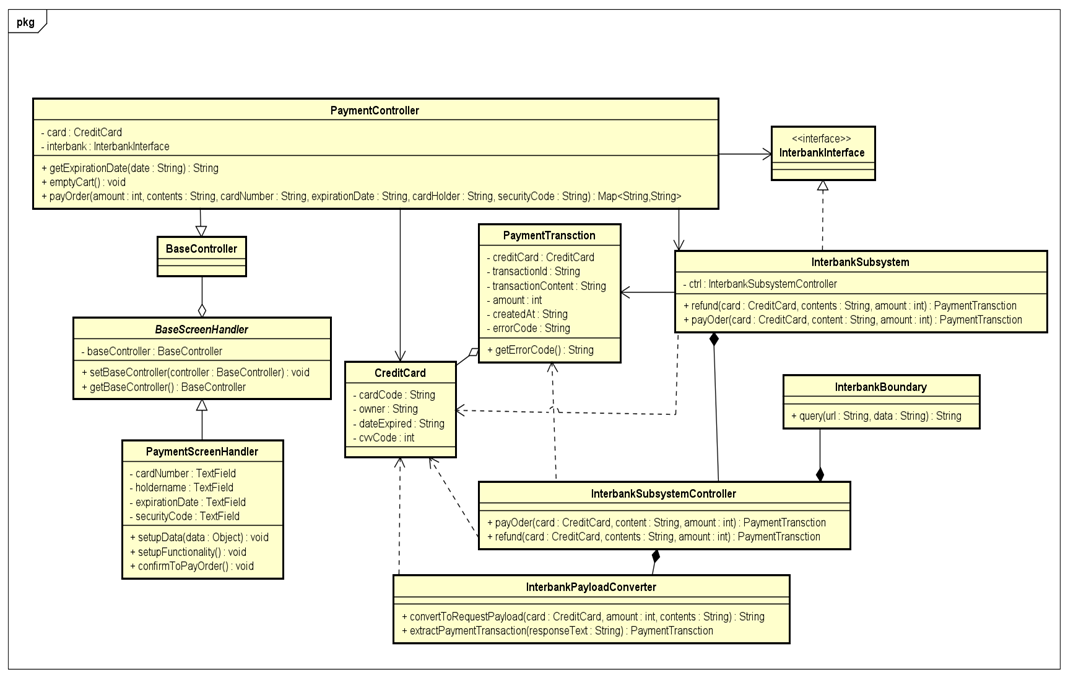
Áp dụng Strategy pattern:

* Trong lớp DeliveryInfo, thêm thuộc tính disCal là đối tượng của giao diện InterfaceDisCal và phương thức setDistanceCalculator.
* Tạo lớp DisCalAdapter thực thi giao diện InterfaceDisCal, lớp này sẽ sử dụng thư viện distance-api.jar để tính khoảng cách trong phương thức calculate.
* Khi thêm thư viện tính khoảng cách mới, chỉ cần viết thêm lớp AltDisCalAdapter cũng thực thi giao diện InterfaceDisCal cho thư viện mới.
* Khi muốn thay đổi cách tính khoảng cách, chỉ cần thay đổi thuộc tính disCal của lớp DeliveryInfo thành đối tượng của lớp mới thông qua phương thức setDistanceCalculator.



## Vấn đề Thêm hình thức thanh toán mới: Thẻ nội địa(Domestic Card) (requirement 5) và giải pháp

### *Vấn đề*



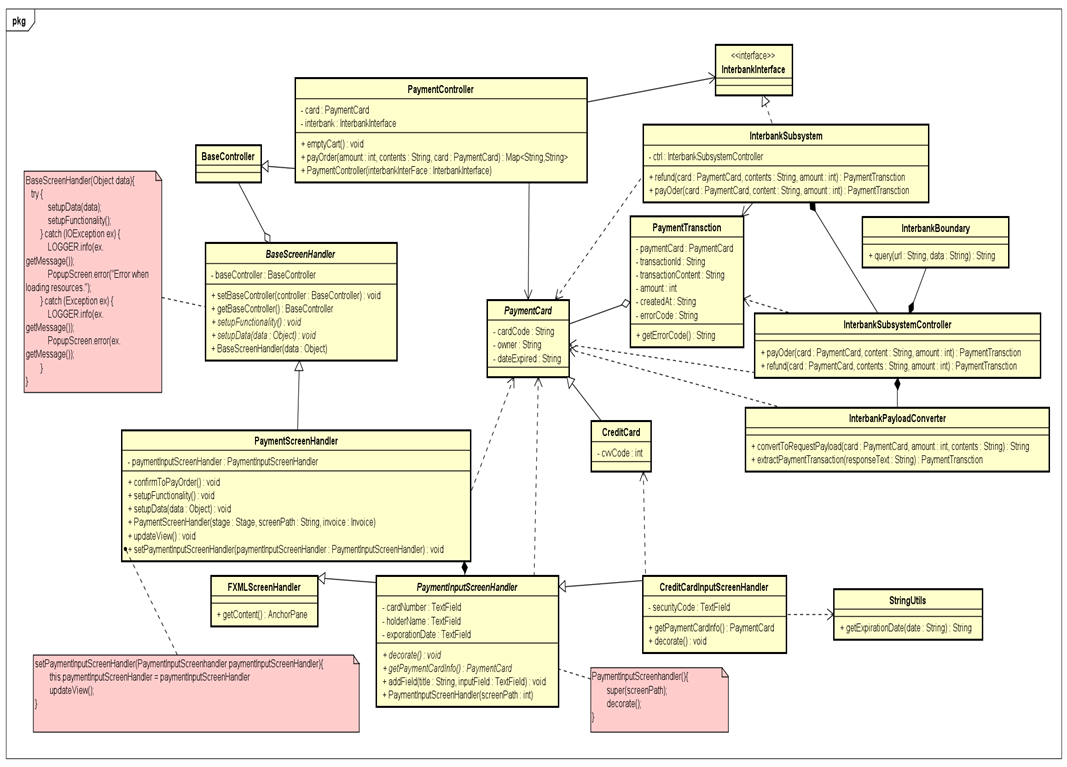
Từ biểu đồ lớp ta có thể thấy một số vấn đề sau:

* Trong PaymentController chứa phương thức getExpiredate. Phương thức này được thêm vào lớp vì trong lớp có sử dụng phương thức này, nhưng phương thức này không liên quan đến nghiệp vụ thanh toán của lớp Controller, vì vậy 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:

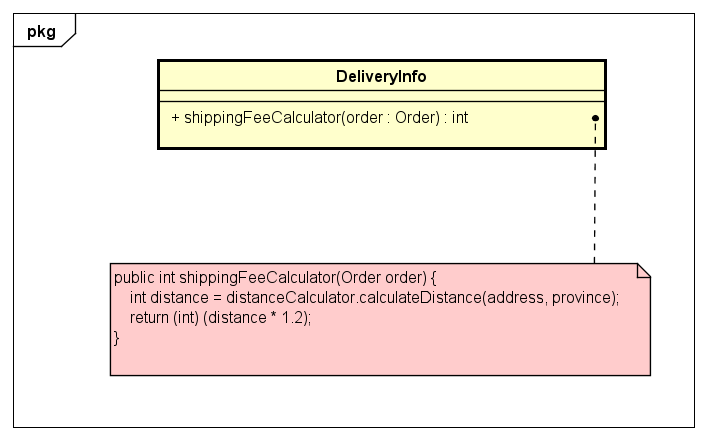
* Tạo lớp abstract PaymentCard và các thành phần trong module sẽ phụ thuộc vào PaymentCard. Khi thêm phương thức mới chỉ cần tạo lớp kế thừa PaymentCard
* Tạo thêm lớp abstract PaymentCardInputScreenHandler chứa các textfield của PaymentCard. Khi thêm phương thức thanh toán mới sẽ tạo lớp kế thừa từ 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



## Vấn đề Thay đổi công thức tính phí vận chuyển (requirement 6) và giải pháp

### *Vấn đề*

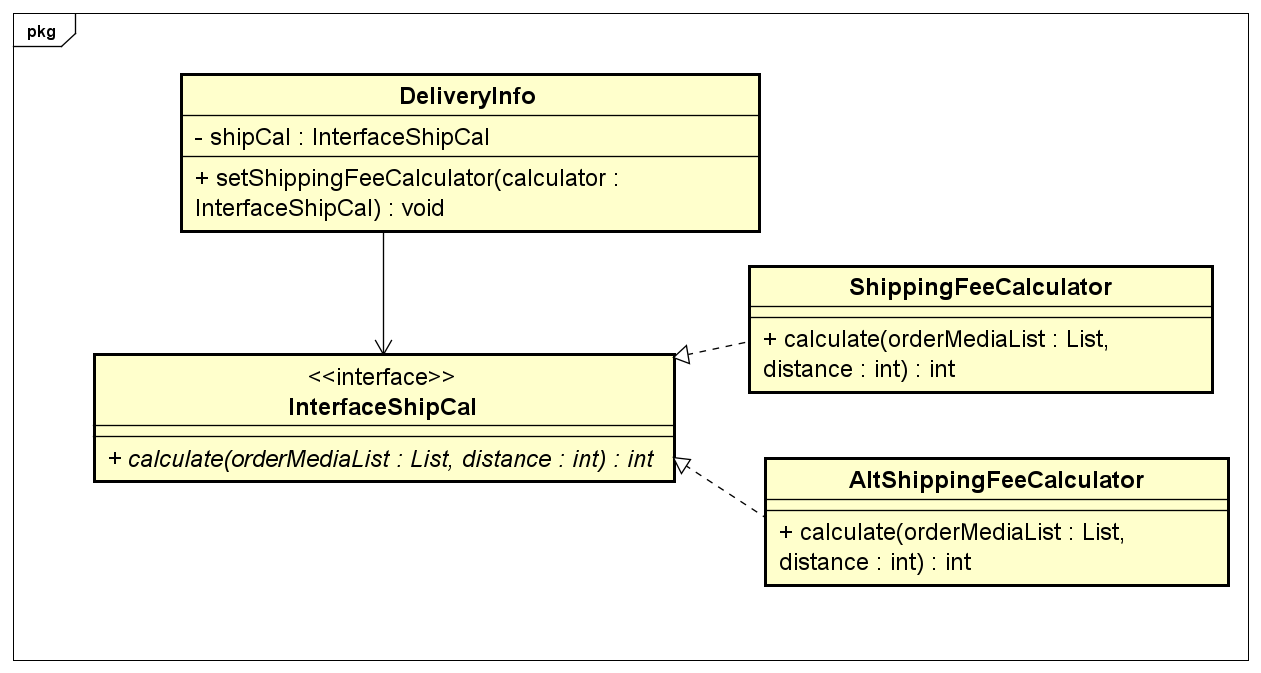
Lớp DeliveryInfo hiện đang dùng phương thức calculateShippingFee để tính phí vận chuyển với công thức cố định là ShippingFee = (Distance \* 1.2). Khi muốn thay đổi công thức tính phí vận chuyển trong tương lai, ta phải trực tiếp sửa phương thức calculateShippingFee, điều này là vi phạm nguyên lý OCP.



### *Giải pháp*

Áp dụng Strategy pattern:

* Trong lớp DeliveryInfo, thêm thuộc tính shipCal là đối tượng của giao diện InterfaceShipCal và phương thức setShippingFeeCalculator.
* Tạo lớp ShippingFeeCalculator thực thi giao diện InterfaceShipCal, lớp này sẽ sử dụng công thức tính khoảng cách cũ với phương thức calculate.
* Khi muốn tính phí vận chuyển, lớp DeliveryInfo sẽ sử dụng phương thức calculateShippingFee với giá trị trả về là shipCal.calculate(orderMediaList, distance).
* Khi muốn thêm cách tính phí vận chuyển mới, chỉ cần viết thêm lớp AltShippingFeeCalculator cũng thực thi giao diện InterfaceShipCal với phương thức calculate sử dụng công thức mới.
* Khi thay đổi cách tính phí vận chuyển, chỉ cần thay đổi thuộc tính shipCal của lớp DeliveryInfo thành đối tượng của lớp mới thông qua phương thức setShippingFeeCalculator.



## Vấn đề Cập nhật lại chức năng hủy đơn hàng (requirement 7) và giải pháp

### *Vấn đề*

Với mã nguồn cũ, sau khi đặt hàng và thanh toán, khách hàng có thể huỷ đơn hàng bất cứ lúc nào và được hoàn tiền. Do đó vấn đề này đã vi phạm OCP.

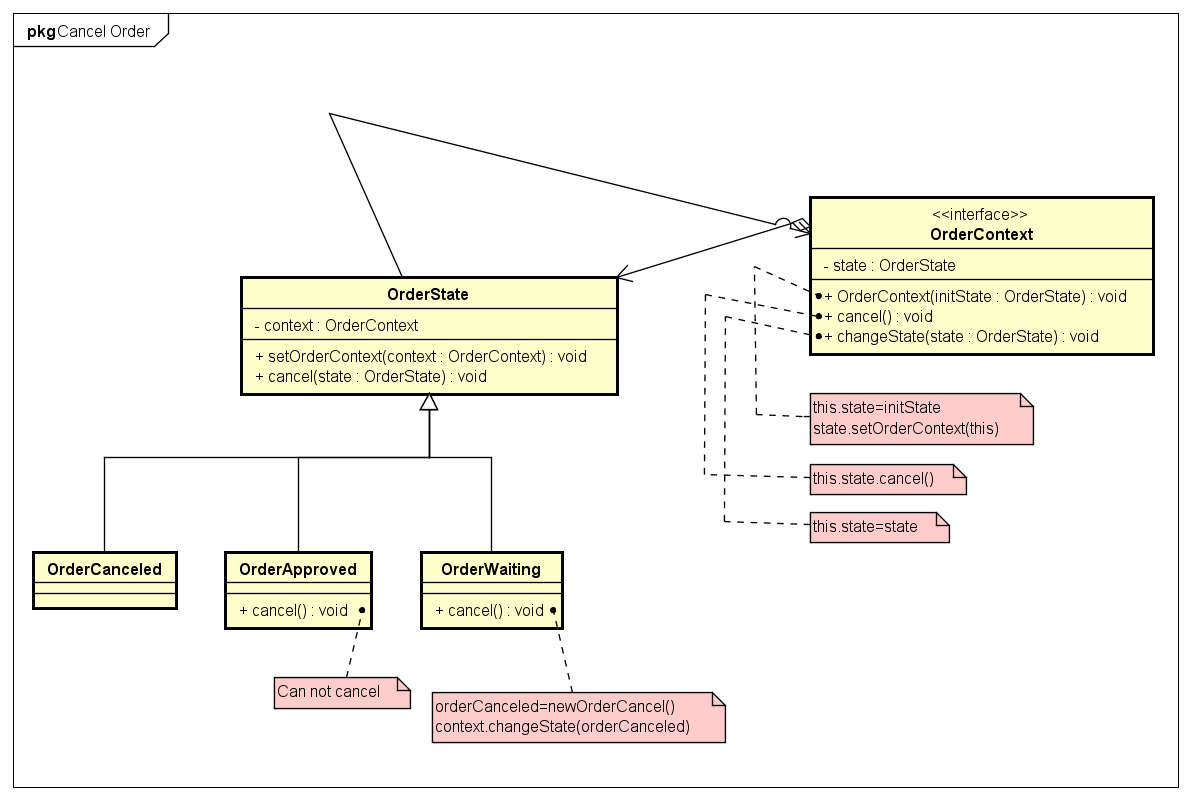
Sau khi refactor lại code, tính năng huỷ đơn hàng chỉ thực hiện được khi admin chưa duyệt đơn hàng

### *Giải pháp*

Giải pháp đưa ra: khi khách hàng tiến hàng hủy đơn hàng, hệ thống sẽ check xem đơn hàng đang ở trạng thái nào và phản hồi thông báo tới user.

→ Sử dụng State Pattern:

* Tạo class OrderContext để lưu trạng thái hiện tại của đơn và có chứa method cancelOrder
* Tạo Interface OrderState là trạng thái của đơn hàng, có method cancel để huỷ đơn hàng.
* Ngoài ra các class khác OrderWaiting, OrderApproved, OrderCanceled implements OrderState
* Lúc đó trạng thái đơn hàng sẽ chuyển sang OrderCanceled → thay đổi state ở OrderContext sang OrderCanceled.



## Vấn đề Common Coupling: SessionInformation, Cart và giải pháp

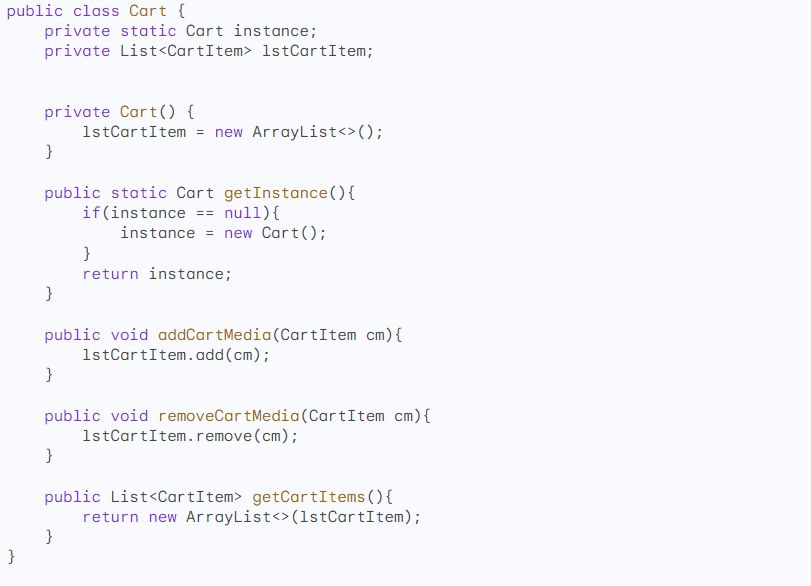
### *Vấn đề*

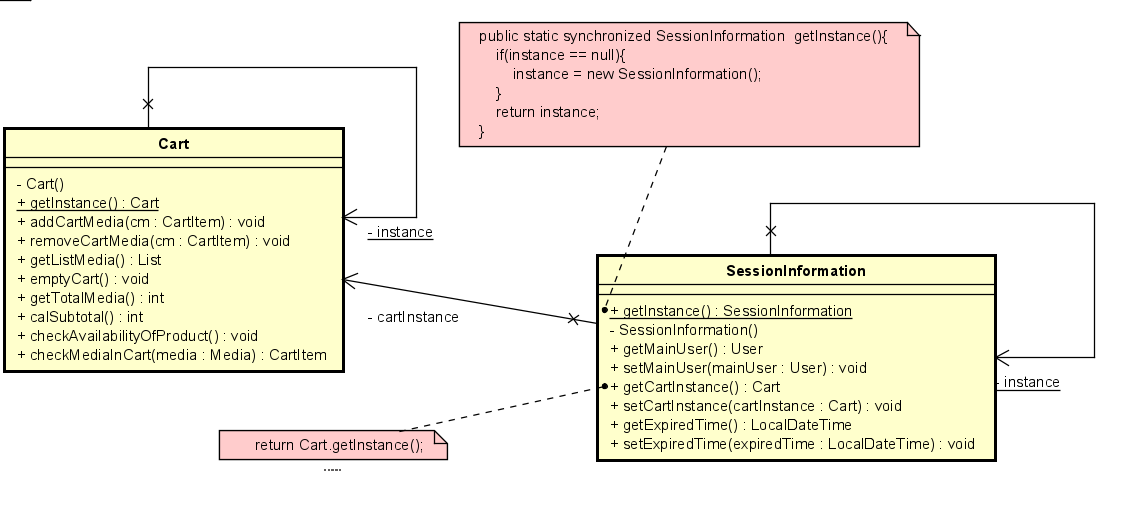
Các Class như **SessionInformation** và **Cart** chỉ cần khởi tạo 1 lần trong trương trình.

Ngoài ra Cart còn vi phạm content coupling.

### *Giải pháp*

Sử dụng Singleton Desgin code





# Tổng kết

## Kết quả tổng quan

Đánh giá được một cách tổng quan về các mức độ coupling, cohesion, clean code và nguyên lý SOLID. Tìm kiếm và sửa chữa được những một số phần gây ra high coupling, low cohesion, code smell và một số phần vi phạm SOLID.

Thực hiện tái cấu trúc lại mã nguồn để thỏa mãn hầu hết các yêu cầu thêm của hệ thống.

## Các vấn đề tồn đọng

Một số lỗi về việc không tuân thủ nguyên lý SOLID, gây ra high coupling và low cohesion vẫn chưa được sửa chữa một cách hoàn thiện.