ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Báo cáo Mẫu thiết kế phần mềm

**Nhóm 11**

|  |  |
| --- | --- |
| Họ và Tên | MSSV |
| Phạm Đình Thắng | 20173370 |
| Nguyễn Quang Tạo | 20173356 |
| Trần Đình Thắng | 20173363 |
| Trịnh Văn Thắng | 20183630 |

*Hà Nội, 06-06-2021*

**Mục lục**

[1 Tổng quan 3](#_Toc74079672)

[1.1 Mục tiêu 3](#_Toc74079673)

[1.2 Phạm vi 3](#_Toc74079674)

[1.3 Danh sách thuật ngữ 5](#_Toc74079675)

[1.4 Danh sách tham khảo 5](#_Toc74079676)

[2 Đánh giá thiết kế cũ 6](#_Toc74079677)

[2.1 Nhận xét chung 6](#_Toc74079678)

[2.2 Đánh giá các mức độ coupling và cohesion 6](#_Toc74079679)

[2.2.1 Coupling 6](#_Toc74079680)

[2.2.2 Cohesion 10](#_Toc74079681)

[2.3 Đánh giá việc tuân theo SOLID 14](#_Toc74079682)

[2.3.1 SRP 14](#_Toc74079683)

[2.3.2 OCP 15](#_Toc74079684)

[2.3.3 LSP 15](#_Toc74079685)

[2.3.4 ISP 15](#_Toc74079686)

[2.3.5 DIP 16](#_Toc74079687)

[2.4 Các vấn đề về Clean Code 16](#_Toc74079688)

[2.4.1 Clear Name 16](#_Toc74079689)

[2.4.2 Clean Function/Method 16](#_Toc74079690)

[2.4.3 Clean Class 17](#_Toc74079691)

[3 Đề xuất cải tiến 19](#_Toc74079692)

[3.1 Vấn đề “Thay đổi thư viện và cách tính phí vận chuyển” và giải pháp 19](#_Toc74079693)

[3.2 Vấn đề “Vi phạm SRP tại PlaceOrderController” và giải pháp 20](#_Toc74079694)

[3.3 Vấn đề “Thêm phương thức thanh toán” và giải pháp 20](#_Toc74079695)

[3.4 Vấn đề “Cập nhật chức năng hủy đơn hàng” và giải pháp 21](#_Toc74079696)

[4 Tổng kết 22](#_Toc74079697)

[4.1 Kết quả tổng quan 22](#_Toc74079698)

[4.2 Các vấn đề tồn đọng 22](#_Toc74079699)

[4.3 Đóng góp của các thành viên 22](#_Toc74079700)

# Tổng quan

## Mục tiêu

Tài liệu này mô tả, đánh giá về việc tuân thủ/vi phạm các nguyên lý thiết kế phần mềm cũng như các cải tiến cho AIMS project. Thiết kế của mã nguồn sau khi được chỉnh sửa và cải tiến cần đạt được low-coupling, high-cohesion, hạn chế vi phạm các nguyên lý SOLID. Developer của dự án cần đọc tài liệu này để cải tiến thiết kế của mình.

Mã nguồn của project: <https://github.com/trangntt-for-student/dp-case-study>

Tài liệu gồm 4 phần chính như sau:

1. Tổng quan: Mô tả khái quát nội dung của báo cáo
2. Đánh giá thiết kế cũ: Đánh giá về tình trạng codebase hiện tại và chỉ ra các vi phạm của codebase
3. Đề xuất cải tiến: Đưa ra những đề xuất cải tiến để khắc phục những vấn đề đã nêu ra và xử lý khi có thêm những yêu cầu phát sinh mới
4. Tổng kết: Mô tả kết quả của dự án và những vấn đề còn tồn đọng

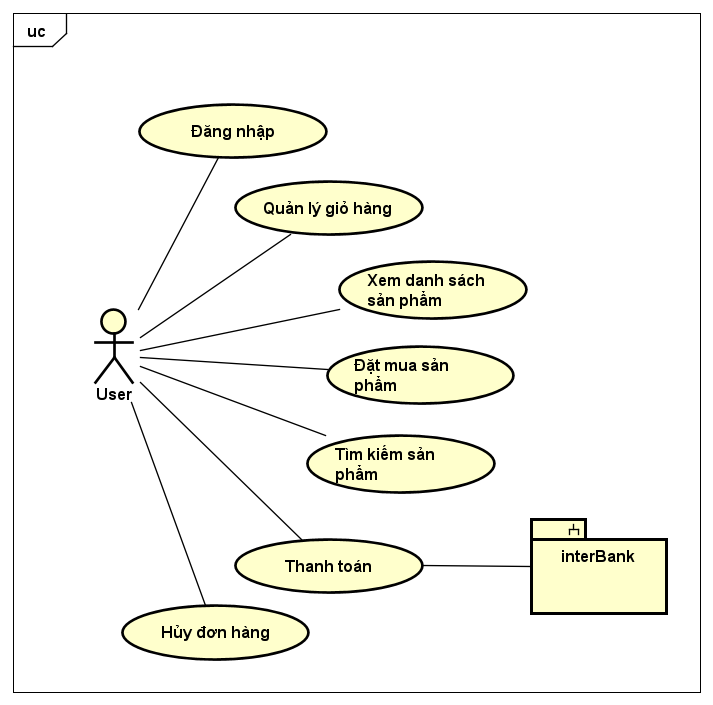
## Phạm vi

1. Mô tả khái quát phần mềm

AIMS project là một ứng dụng để khách hàng mua sắm Media online, sử dụng trên Desktop. Khách hàng có thể mua các loại sản phẩm như Book, DVD và CD. Hệ thống 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

1. Các chức năng chính

* Đăng nhập
* Tìm kiếm sản phẩm
* Xem danh sách sản phẩm
* Quản lý giỏ hàng
* Đặt mua sản phẩm
* Thanh toán
* Hủy đơn hàng



1. Cấu trúc mã nguồn

Cấu trúc project sử dụng mô hình MVC, mã nguồn của project được chia thành 7 package chính:

* common: Gồm các lớp thường dùng (vd: các exception tự định nghĩa )
* controller: Gồm các lớp đóng vai trò là controller trong MVC
* entity: Gồm các class đóng vai trò là model trong MVC
* views: Gồm các lớp định nghĩa giao diện của ứng dụng, view trong MVC
* dao: Gồm các class có chức năng truy vấn đến CSDL cho các đối tượng model
* subsystem: các subsystem được sử dụng trong phần mềm (vd: api thanh toán)
* utils: các lớp tiện ích

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

* Thêm mặt hàng Media mới: AudioBook
* 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: Thẻ nội địa (Domestic Card)
* Thay đổi công thức tính phí vận chuyển
* Cập nhật lại chức năng hủy đơn hàng

1. Các hoạt động

* Đánh giá mức độ tuân thủ coupling, cohesion
* Đánh giá mức độ tuân thủ SOLID
* Đánh giá về Clean Code
* Refactor code để tuân thủ các yêu cầu trên
* Refactor code để đáp ứng các yêu cầu phát triển trong tương lai

1. Kết quả dự kiến

* Đánh giá được tổng thể về thiết kế của project
* Cải tiến mã nguồn đáp ứng Clean Code, SOLID và đáp ứng một số yêu cầu phát triển trong tương lai

## Danh sách thuật ngữ

## 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

Retrieved from www.elsevier.com/locate/eswa

# Đánh giá thiết kế cũ

## Nhận xét chung

Mã nguồn hiện tại được thiết kế có cấu trúc tốt, tuân thủ tương đối tốt Clean Code. Nhưng chưa đáp ứng tốt với các yêu cầu thay đổi trong tương lai.

## Đánh giá các mức độ coupling và cohesion

* Coupling là mức độ phụ thuộc lẫn nhau giữa các mô-đun phần mềm. Low-coupling là một yêu cầu để phần mềm có cấu trúc và thiết kế tốt.
* Cohesion là mức độ mà các phần tử bên trong mô-đun liên quan đến nhau. Một thiết kế đảm bảo high-cohesion sẽ có độ tin cậy, khả năng tái sử dụng và tính dễ hiểu cao.
* Coupling thường tương phản với cohesion. Một thế kế tốt cần đảm bảo low-coupling và high-cohesion.

### Coupling

Các module chủ yếu có mức độ kết dính trung bình. Content-coupling, common-coupling vẫn còn xuất hiện thường xuyên, cần sửa đổi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Coupling* | *Module* | *Mô tả* | *Ảnh hưởng* |
| 1 | Content Coupling | -Controller.AuthenticationController.login  -Controller.BaseController.getListCartMedia  -dao.media.MediaDAO  -Các module xảy ra Content Coupling như views.screen.cart.CartScreenHandler , views.screen.intro.IntroScreenHandler , | - Sử dụng và thay đổi các thuộc tính mainUser, expiredTime của module SessionInformation  - Sử dụng phương thức static getListMedia của module SessionInformation  -Sử dụng phương thức getConnection().createStatement() của module entity.db.Aimsdb | -Module này có thể truy cập và thay đổi dữ liệu của module khác, làm ảnh hưởng tới tính đóng gói thông tin và khiến không kiểm soát được các dữ liệu hiện có trong module  - Ảnh hưởng tới tính đóng gói thông tin và kiến không kiểm soát được các dữ liệu hiện có trong module  -Module này có thể truy cập và thay đổi dữ liệu của module khác ảnh hưởng tới tính đóng gói và sự kiểm soát dữ liệu một cách hiệu quả |
| 2 | Common Coupling | -Controller.AuthenticationController.getMainUser  - Controller.BaseController.GetlistCartMedia  -Entity.media (Book, CD, DVD)  -Controller.AuthenticationController.logout  -Các module khác xảy ra common coupling là Controller. HomeController.getAllMedia, views.screen.cart.MediaHandler, views.screen.cart.CartScreenHandler, subsystem.interbank.InterBankConfigs | -Truy cập trực tiếp vào module SessionInformation và thay đổi giá trị mainUser, expiredTime  - Truy cập vào biến toàn cục SessionInformation  -Các biến để ở dạng public chứ không ở dạng private  -Truy cập trực tiếp vào module SessionInformation và thay đổi giá trị mainUser, expiredTime | -Module này thay đổi dữ liệu của module khác khiến không kiểm soát được các dữ liệu hiện có trong module  - Việc này dẫn đến việc không kiểm soát được lỗi và sẽ xảy ra lỗi không mong muốn  -Dễ bị module khác chỉnh sửa giá trị mà không biết  -Module này thay đổi dữ liệu của module khác khiến không kiểm soát được các dữ liệu hiện có trong module |
| 3 | Control Coupling | -Subsystem.interbank.InterbankPayloadConverter | -Phương thức PaymentTransaction điểu khiển luồng hoạt động của module exception thông qua switch case | -Khi module này điều khiển luồng hoạt động của module exception sẽ dẫn tới trường hợp một vài tham số đầu vào không đúng |
| 4 | Stamp Coupling | -entity.order.Order.Order  -entity.shipping.DeliveryInfo.calculateShippingFee  -Các số module khác xảy ra stamp coupling là subsystem.interbank.InterbankSubsystemController.PaymentTransaction và , subsystem.InterbankSubsystem.PaymentTransaction và views.screen.invoice.InvoiceScreenHandler.InvoiceScreenHandler | -Phương thức Order() sử dụng tham số là đối tượng cart của class Cart  -Phương thức calculateShippingFee() sử dụng tham số truyền vào là đối tượng order của class Order | -khi class Cart thay đổi thuộc tính hoặc phương thức nào đó sẽ ảnh hưởng đến thể hiện của class này là cart được gọi đến ở phương thức Order()  - khi class Order thay đổi thuộc tính hoặc phương thức thì sẽ ảnh hưởng đến tham số truyền vào của phương thức calculateShippingFee() |
| 5 | Data Coupling | -controller.BaseController.checkMediaInCart  -controller.PaymentController.getExpirationDate  -Các module khác xảy ra data coupling là  controller.PaymentController.payOrder và  controller.PlaceOrderController.validatePhoneNumber/Name/Address và dao.media.BookDAO.getMediaByID và  dao.media.CDDAO.getMediaById ... | -Các module khác khi tương tác với phương thức này phải qua tham số đầu vào là kiểu dữ liệu nguyên thuỷ integer  -Các module khác khi tương tác với phương thức này phải qua tham số đầu vào là kiểu dữ liệu dạng String | -Module tương đối độc lập với module khác do tham số truyền vào là dạng integer nên khi tương tác không ảnh hưởng nhiều tới cấu trúc module  -Module tương đối độc lập với module khác |

### Cohesion

Các submodule đã có sự liên kết khá chặt chẽ. Vẫn có một số hàm tiện ích được đặt ở trong các class một cách rời rạc.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Cohesion* | *Module* | *Mô tả* | *Ảnh hưởng* |
| 1 | Coincidental Cohesion | - controller.PaymentController.emptyCart  - subsystem..interbank.InterbankPayloadConverter.getToday  - views.creen.home.LoginScreenHandler.backToHomeScreen  - controller.AuthenticationController | - Phương thức emptyCart không liên quan đến lớp PaymentController  - Phương thức getToday không tách biệt với lớp => cho vào lớp Utils ( chung )  - Phương thức backToHomeScreen nên khộng được tách biệt với lớp  - hàm md5() nên để lại class utils để controller gọi đến khi sử dụng |  |
| 2 | Logical cohesion | - entity.cart.CartItem  - controller.PlaceOrderController | - trong lớp các thành phần media, price, quantity khác nhau về bản chất  - các phương thức validate như validateDeliveryInfo , validatePhoneNumber, validateName, validateAddress cần được tách riêng vào một lớp |  |
| 3 | Temporal cohesion | - entity.order.Order.setDeliveryInfo  - controller.AuthenticationController  - views.screen.home.HomeScreenHandler  - views.screen.cart.CartScreenHandler  - views.screen.intro.IntroScreenHandler | - thực hiện 2 việc cùng 1 lúc  - Các phương thức thực hiện theo thứ tự khi login  - các hàm không liên quan đến nhau  - setOnMouseClicked va setOnMouseClicked doc lap  - Các phương thức được gom lại theo thời gian xử lý |  |
| 4 | Procedural Cohesion | - entity.media.Media.getQuantity  - controller.AuthenticationController | - 2 khối lệnh chỉ liên kết với nhau vì năm trong 1 trình tự thực hiện.  - phương thức logout() được nhóm lại trong Class này để tuân theo trình tự thực thi từ login -> Logout |  |
| 5 | Communicational Cohesion | - utils.HttpClient  - controller.BaseController  - entity.order.Order  - .entity.shipping.DeliveryInfo.calculateShippingFee | - Các phương thức ( get, getString, post ) thực hiện việc chia sẻ dữ liệu của thuộc tín Logger và các mục đích thực hiện liên quan đến nhau  - Các phương thức checkMediaInCart() cùng truy cập đến dữ liệu giỏ hàng ( SessionInformation.getInstance().getCartInstance() )  - Các phương thức của Class Order sử dụng các thuộc tính chung  - Phương thức calculateShippingFee() không liên quan đến Class DeliveryInfo này |  |
| 6 | Functional Cohesion | - utils.MyMap  - entity.cart.Cart | - Tất cả các phương thức đều đóng góp nhất quán vào nhiệm vụ  - Các phương thức đều thực hiện cùng một mục đích có quan hệ mật thiết với nhau |  |

## Đánh giá việc tuân theo SOLID

Với thiết kế hiện tại, khi có những yêu cầu phát sinh thì đa phần chưa đáp ứng được, cần cải tiến thiết kế.

### SRP

Phần lớp các class controller khá lớn, phức tạp, làm nhiều việc với các mục đích khác nhau, chưa tuân thủ nguyên lý SRP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Ảnh hưởng* |
| 1 | controller.PlaceOrderController | Class PlaceOrderController chứa nhiều hàm validate như validateAddress, validateName, validateNumber, ... | Class PlaceOrderController có hơn 1 lý do để thay đổi, do chứa nhiều hàm validate nên class sẽ trở nên cồng kềnh, khó đọc hiểu và sửa đổi thậm chí có thể nhầm các phương thức validate với nhau |
| 2 | controller.AuthenticationControlelr | Class AuthenticationController vừa có nhiện vụ login logout vừa có nhiệm vụ mã hóa mật khẩu (md5) | Class AuthenticationController có hơn 1 lý do để thay đổi, do có nhiều chức năng không liên quan đến nhau trong class nên khi đọc hiểu và sửa đổi sẽ trở nên khó khăn |
| 3 | controller.PaymentController | Class PaymentController vừa có nhiệm vụ thanh toán vừa có nhiệm vụ lấy ngày hết hạn (getExpirationDate) | Class PaymentController có hơn 1 lý do để thay đổi, do có nhiều chức năng không liên quan đến nhau trong class nên khi đọc hiểu và sửa đổi sẽ trở nên khó khăn |

### OCP

Với những yêu cầu phát sinh thì các class liên quan hầu như chưa đáp ứng được nguyên lý OCP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Ảnh hưởng* |
| 1 | controller.PaymentController | Phương thức payOrder sẽ phải thay đổi khi có thêm các kiểu thanh toán mới | Khi có thêm kiểu thanh toán mới với thiết kế hiện tại sẽ phải viết phần code mới vào class payOrder dẫn đến vi phạm nguyên lý OCP. |
| 2 | controller.PlaceOrderController | Phương thức validateInfo sẽ thay đổi khi thêm hoặc bớt các dữ liệu cần validate | Khi có thay đổi về số trường cần validate trong thông tin người dùng thì phương thức validateInfo phải được viết lại vào phần code cũ. |
| 3 | entity.Shipping.DeliveryInfo | Phương thức calculateShippingFee sẽ phải thay đổi khi thay đổi phương thức tính phí vận chuyển | Khi có thay đổi về cách phí vận chuyển thì với code hiện tại thì code mới sẽ phải viết lại vào phần code cũ là phần tính phương thức vận chuyển cũ |

### LSP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Ảnh hưởng* |
| 1 | controller | Các lớp trong module này đều kế thừa lớp BaseController mà không sử dụng đến 2 phương thức đã được kế thừa | Lớp BaseController có 2 phương thức checkMediaInCart và getListCartMedia tuy nhiên các lớp kế thừa nó lại không có chức năng liên quan đến giỏ hàng. Dẫn đến vi phạm nguyên lý LSP. |

### ISP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Ảnh hưởng* |
|  |  |  |  |

### DIP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Ảnh hưởng* |
| 1 | controller.PaymentController | Thuộc tính card trong PaymentController chứa các thông tin về thẻ dùng để thanh toán nên khi thay đổi phương thức thanh toán thì sẽ phải sửa lại code | -Sẽ phải sửa lại phần code cũ khi có thêm phương thức thanh toán mới. |

### 

## Các vấn đề về Clean Code

### Clear Name

Về cơ bản, mã nguồn đã đáp ứng Clear Name. Một số chỗ chưa clear được chỉ ra dưới đây

|  |  |  |  |
| --- | --- | --- | --- |
| # | Module | Mô tả | Lý do |
| 1 | entity.cart.Cart | private List<CartItem> lstCartItem | Nên để là listCartItem |
| 2 | entity.media.Media | protected Statement stm | Tên biến nên là statement |
| 3 | subsystem.interbank.InterbankSubsystemController | private InterbankSubsystemController ctrl | Tên biến là controller sẽ dễ hiểu hơn |

### Clean Function/Method

Phần lớn các Function đáp ứng Clean Function. Tuy nhiên vẫn còn 1 số hàm còn chưa Clean như dưới đây

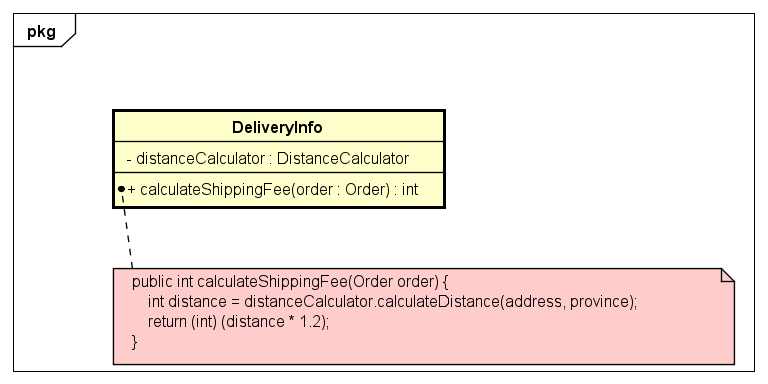
|  |  |  |  |
| --- | --- | --- | --- |
| # | Module | Mô tả | Lý do |
| 1 | Controller.AuthenicationController.getMainUser() | -Thêm function isLogin để sử dụng trong câu lệnh if kiểm tra người dùng đã đăng nhập hay chưa, nếu chưa đăng nhập sẽ logout | -Khiến cho code dễ đọc hiểu và chỉnh sửa hơn |
| 2 | Controller.PlaceController.validateDeliveryInfo() | -Thêm các biến boolean để validate số điện thoại, tên, địa chỉ của người dùng | -Khiến cho code dễ đọc hiểu và chỉnh sửa hơn |
| 3 | Controller.PlaceController.validatePhoneNumber | -Kết hợp 2 câu lệnh kiểm tra độ dài của số điện thoại và số điện thoại bắt đầu bằng 0 vào cùng trong 1 câu lệnh if | -Hai câu lệnh kiểm tra cùng muốn xác thực số điện thoại nhưng lại tách ra làm 2 câu lệnh if khiến code dài dòng |
| 4 | entity.media.Media | -Thêm phương thức getBasicInformation để các Class DVD, CD, Book có thể kế thừa và sử dụng lại, tránh trùng lặp code | -Phần code của hàm toString() bị trùng lặp lại 1 phần ở class DVD, CD và Book |
| 5 | entity.shipping.DeliveryInfo | -Thêm biến final CACULATE\_SHIPPING\_FEE = 1.2 để sử dụng lúc tính giá vận chuyển | -Tạo thêm các biến có nghĩa để sau này dễ sử dụng và dễ sửa đổi code |
| 6 | views.screen.cart.MediaHandle | -Thêm biến final HEIGHT\_MEDIA\_INFO = 110 và WIDTH\_MEDIA\_INFO = 92 để sử dụng lúc set chiều rộng và chiều cao cho Media | -Tạo thêm các biến có nghĩa để sau này dễ sử dụng và dễ sửa đổi code |

### Clean Class

|  |  |  |  |
| --- | --- | --- | --- |
| # | Module | Mô tả | Lý do |
| 1 | controller.PlaceOrderController | Lớp này phụ trách việc đặt hàng nhưng chứa khá nhiều hàm validate | Nên tạo một lớp chuyên phụ trách việc validate để ý nghĩa của các lớp rõ ràng hơn |
| 2 | controller.AuthenticationController | Lớp này chứa những thao tác liên quan đến việc đăng nhập, đăng xuất của người dùng.  Chứa hàm mã hóa md5 | Các hàm tiện ích nên gom thành class riêng biệt thuận tiện cho việc tái sử dụng và khiến các class rõ ràng hơn |
| 3 | controller.PaymentController | Lớp xử lý liên quan đến thanh toán  Có hàm tiện ích: getExpirationDate() | Các hàm tiện ích nên gom thành class riêng biệt thuận tiện cho việc tái sử dụng và khiến các class rõ ràng hơn |

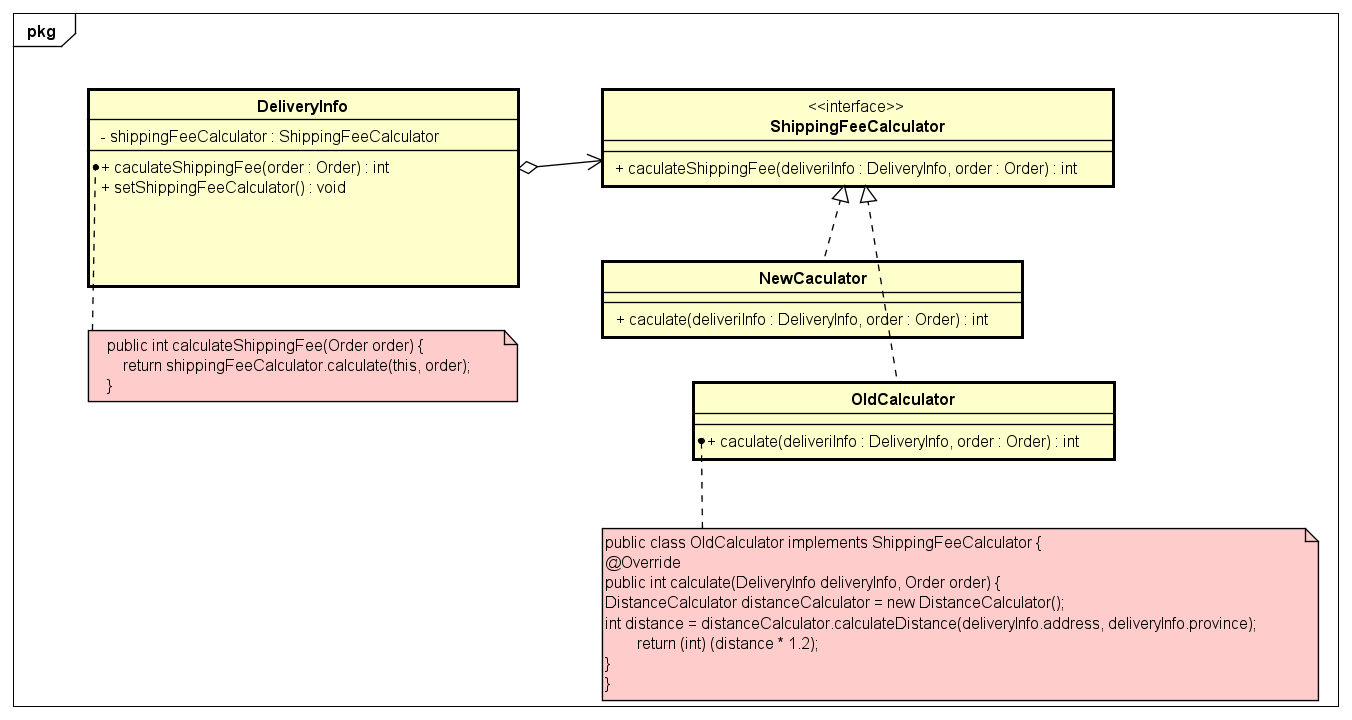
# Đề xuất cải tiến

## Vấn đề “Thay đổi thư viện và cách tính phí vận chuyển” và giải pháp



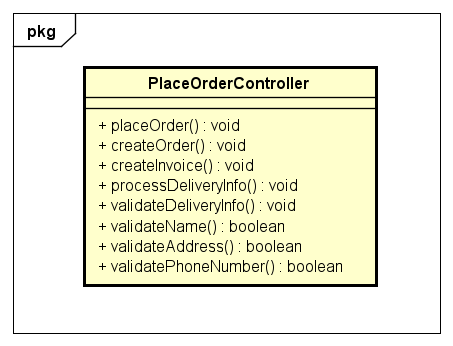
Nhìn vào biểu đồ trên, có thể thấy cách tính phí vận chuyển đang phụ thuộc vào một thư viện cố định. Trong tương lai, nếu muốn thay đổi cách tính phí cũng như sử dụng thư viện khác thì sẽ phải sửa đổi ở trong lớp DeliveryInfo, vi phạm tính đóng mở.

Đề xuất cải tiến:



Ở đây chúng em áp dụng mẫu thiết kế Strategy, khi cần thay đổi thư viện-cách tính phí vận chuyển chỉ cần tạo một lớp Calculator mới thực thi Interface ShippingFeeCalculator và định nghĩa cách tính phí mới trong hàm calculate. Không cần sửa đổi mã nguồn của lớp DeliveryInfo.

## Vấn đề “Vi phạm SRP tại PlaceOrderController” và giải pháp



-Nhìn vào biểu đồ lớp, có thể thấy lớp PlaceOrderController đang chứa nhiều phương thức validate khác nhau dẫn đến vi phạm nguyên lý SRP làm cho hàm trở nên cồng kềnh, khó để đọc hiểu và sửa đổi sau này thậm chí có thể gây nhầm lẫn khi sử dụng.

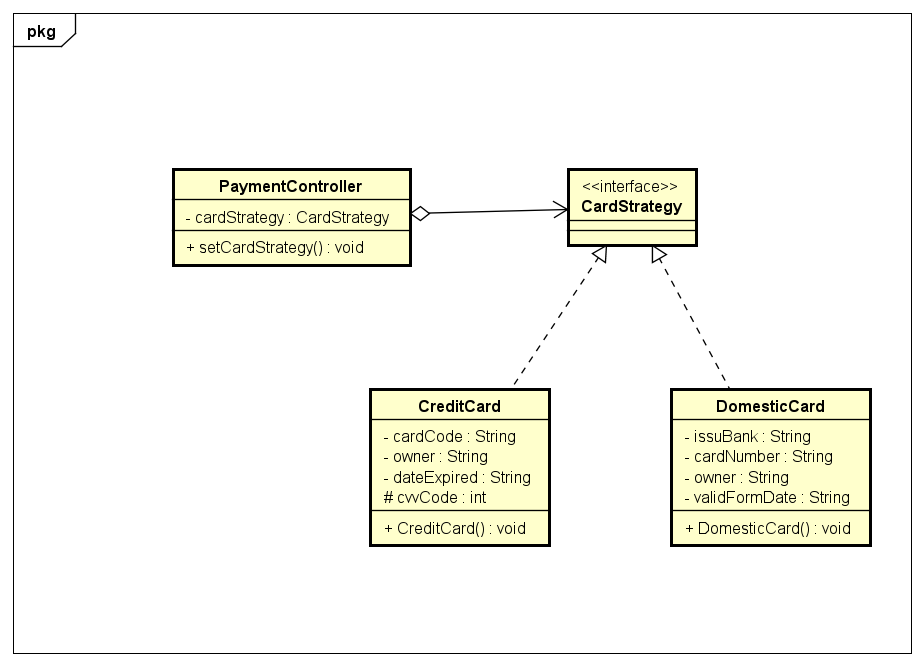
-Giải pháp: Tách các hàm validateName(), validateAddress(), validatePhoneNumber() vào class validate nằm trong mục utils để sử dụng cho hàm validateDeliveryInfo hoặc có thể sử dụng cho các hàm khác sau này nếu cần

-Giải quyết tương tự cho class AuthenticationController, tách function md5() vào class encryption nằm trong mục utils.

## Vấn đề “Thêm phương thức thanh toán” và giải pháp

-Vấn đề: Khi có thêm phương thức thanh toán mới là thẻ nội địa thì ta sẽ phải sửa trực tiếp vào phần code cũ ở function payOrder()

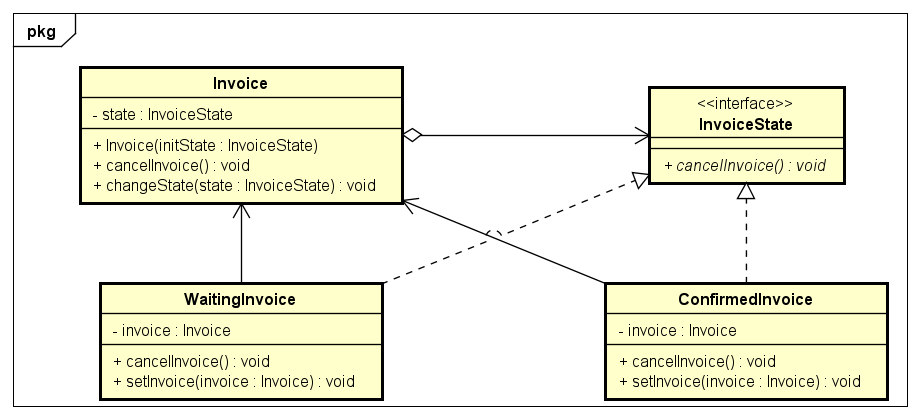
-Giải pháp: Sử dụng mẫu thiết kế Strategy, khi có thêm phương thức thanh toán mới là thẻ nội địa thì ta chỉ cần tạo thêm class DomesticCard và implement CardStrategy mà không cần phải thay đổi code cũ



## Vấn đề “Cập nhật chức năng hủy đơn hàng” và giải pháp

Chức năng hủy đơn hàng có thể thực hiện sau khi khách hàng đặt hàng thành công. Trong tương lai nếu muốn vô hiệu hóa chức năng hủy đơn hàng khi đơn hàng đã được admin xác nhận sẽ cần phải sửa đổi trực tiếp vào class Invoice.

Giải pháp chúng em đưa ra là sử dụng mẫu thiết kế State



Với mẫu thiết kế này, đơn hàng có 2 trạng thái Waiting và Confirmed tương ứng với việc hàm cancelInvoice() (hủy đơn hàng) sẽ dùng được hoặc vô hiệu hóa. State khởi tạo của đơn hàng sẽ là Waiting. Sau này, khi admin xác nhận đơn hàng của admin chỉ cần set State thành Confirmed, không sửa đổi mã nguồn class Invoice, đảm bảo nguyên lý OCP.

# Tổng kết

## Kết quả tổng quan

-Tìm được hầu hết các vi phạm về coupling, cohesion của codebase hiện tại

-Tim ra các vi phạm nguyên lý SOLID và giải quyết được 1 phần.

-Clean codebase làm cho code dễ đọc và chỉnh sửa hơn.

-Nhóm đã áp dụng các design pattern được học để xử lý các vi phạm trên, tuy nhiên do thời gian có hạn nên nhóm không thể giải quyết được hết những vi phạm đã tìm ra. Một số yêu cầu trong tương lai nhóm chưa tìm được phương pháp giải quyết.

-Tuy còn 1 số thiếu sót nhưng mã nguồn hiện tại cũng đã đáp ứng được 1 phần yêu cầu của môn học và dễ dàng để quản lý, sửa đổi sau này.

## Các vấn đề tồn đọng

|  |  |  |
| --- | --- | --- |
| **#** | **Vấn đề còn tồn động** | **Khả năng xử lý** |
| *1* | Thêm mặt hàng media mới (Audio Book) | Sẽ xử lý trong tương lai |
| *2* | Thêm màn hình: Xem chi tiết sản phẩm | Sẽ xử lý trong tương lai |
| *3* | Thay đổi yêu cầu load giao diện | Sẽ xử lý trong tương lai |

## Đóng góp của các thành viên

|  |  |
| --- | --- |
| Thành viên | Công việc |
| Phạm Đình Thắng | Tìm các lỗi liên quan đến Clear Name, Clean Class.  Cải tiến thiết kế đáp ứng các yêu cầu số 4,6,7 |
| Nguyễn Quang Tạo | Tìm các lỗi liên quan đến Clean Fucntion, SOLID  Cải tiến yêu cầu số 5, cải tiến vi phạm SRP |
| Trần Đình Thắng | Tìm module tương ứng với các cấp độ Cohesion |
| Trịnh Văn Thắng | Tìm module tương ứng với các cấp độ Coupling |