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 6

Thành viên: Lù Mạnh Thắng 20194167

Nguyễn Tiến Tuấn 20194202

*Hà Nội,* *tháng 6/2024*

Mục lục

1 Tổng quan 2

1.1 Mục tiêu 3

1.2 Phạm vi 3

1.3 Danh sách thuật ngữ 6

1.4 Danh sách tham khảo 6

2 Đánh giá thiết kế cũ 6

2.1 Nhận xét chung 7

2.2 Đánh giá các mức độ coupling và cohesion 7

2.2.1 Coupling 7

2.2.2 Cohesion 10

2.3 Đánh giá việc tuân theo SOLID 12

2.3.1 SRP 12

2.3.2 OCP 14

2.3.3 LSP 14

2.4 Các vấn đề về Clean Code 16

2.4.1 Clean Name 16

2.4.2 Clean Function/Method 17

2.4.3 Clean Class 18

2.5 Các vấn đề khác 18

3 Đề xuất cải tiến 19

3.1 Sử dụng Singleton với Cart 20

3.2. Cải tiến vấn đề thay đổi phương thức tính khoảng cách sử dụng thư viện mới và thay đổi công thức tính phí vận chuyển 20

4 Tổng kết 22

4.1 Kết quả tổng quan 23

4.2 Các vấn đề tồn đọng 23

# Tổng quan

Nội dung báo cáo gồm:

* Đánh giá thiết kế cũ theo các nguyên tắc Coupling, Cohesion, nguyên lý SOLID, các vấn đề về Clean Code
* Đề xuất cải tiến cho các vấn đề được đưa ra dựa theo các Design Pattern được học và được mô tả thông qua biểu đồ lớp và minh họa mã nguồn.

## Mục tiêu

Tài liệu này được sử dụng làm báo cáo môn học Mẫu thiết kế phần mềm - IT4536 của Trường Công nghệ thông tin và truyền thông – Đại học Bách Khoa Hà Nội. Đối tượng người đọc là những người đã có kiến thức cơ sở về lập trình hướng đối người thiết kế hệ thống phần mềm, người có kiến thức cơ bản về công nghệ thông tin.

Báo cáo là kết quả làm việc của nhóm , nhằm tổng kết kết quả, cách thức thực hiện việc tái cấu trúc mã nguồn dựa trên một hệ thống phần mềm có sẵn được cung cấp bởi giảng viên, sử dụng các mẫu Design Pattern trong môn học áp dụng để cải tiến thiết kế hiện tại.

## Phạm vi

***1.2.1. Mô tả khái quát phần mềm***

Phần mềm AIMS là 1 web bán hàng với các sản phẩm: sách, đĩa CD và DVD. Người dùng muốn mua sản phẩm nào sẽ thêm sản phẩm đó vào giỏ hàng. Sau đó người dùng có thể thanh toán bằng hình thức credit card. Sau đó người dùng điền các thông tin vận chuyển để sản phẩm được giao đến đúng người.

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

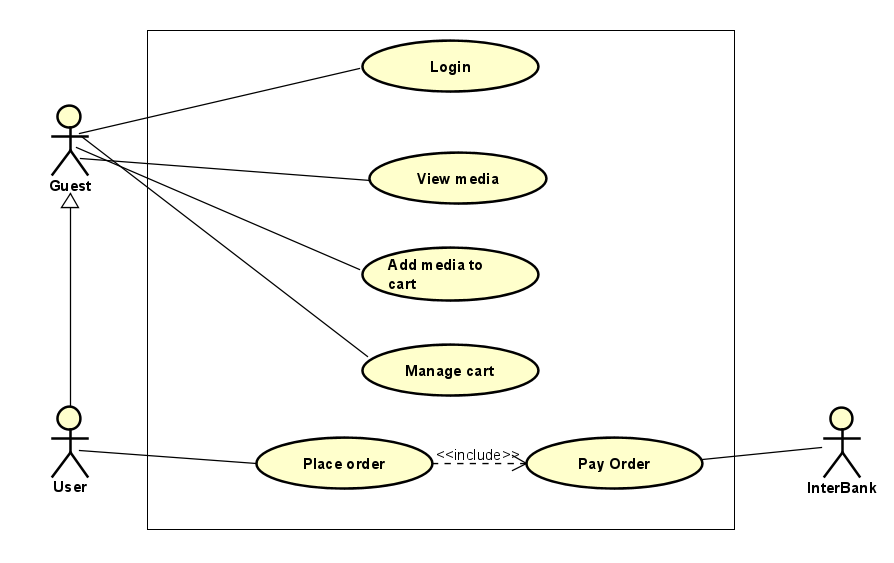
- Đăng nhập

- Thêm sản phẩm, xem danh sách sản phẩm

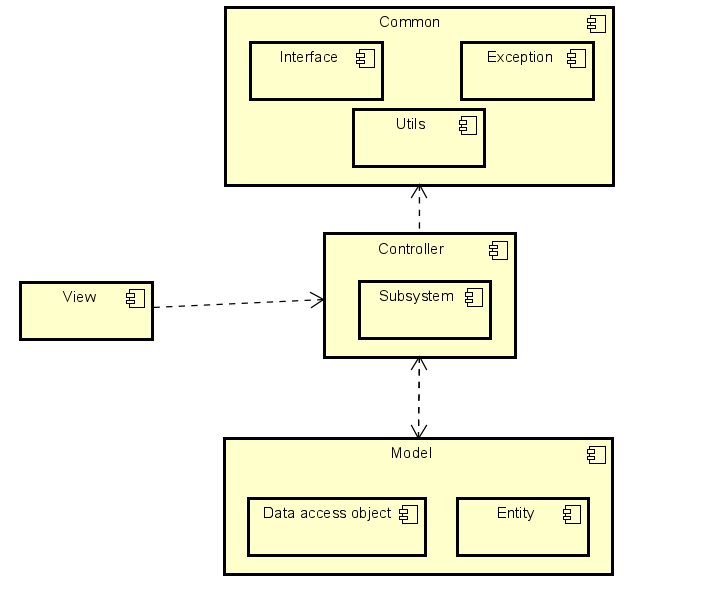
- Xem giỏ hàng

- Thanh toán

- Điền thông tin vận chuyển



***1.2.3. Cấu trúc mã nguồn***



Mã nguồn của phần mềm hiện tại đang được thiết kế theo mô hình MVC, gồm ba tầng chính là Model, View và Controller. Trong đó, tầng View bao gồm các xử lý logic ở các màn hình, lấy dữ liệu từ người người dùng và chuyển cho tầng Controller và thực hiện điều hướng giữa các màn hình với nhau; tầng Controller bao gồm việc xử lý logic của luồng dữ liệu, xác thực dữ liệu người dùng và tương tác với hệ thống subsystem để thực hiện thanh toán tiền cho đơn hàng của người dùng; tầng Model sẽ thực hiện lưu dữ liệu của hệ thống và bao gồm hai modules chính là Data Access Object và Entity. Ngoài ra hệ thống còn có một các thành phần dùng chung, gọi chung là Common, bao gồm các Interfaces, các Exceptions và Utils.

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

Cần đảm bảo các nguyên lý trong SOLID. Việc tái cấu trúc phải đảm bảo các tính năng của hệ thống hiện tại không bị thay đổi về mặt chức năng và phi chức năng.

***1.2.5. Các hoạt động (Ví dụ: review, refactor) thực thi trên mã nguồn để đạt được mục tiêu kể trên***

Các hoạt động Review mã nguồn:

* Xác định các mức độ coupling và cohesion

- Xác định các vi phạm nguyên lý SOLID

Các hoạt động Refactor mã nguồn:

- methods, class để đảm bảo các tiêu chí về clean code

- Tái cấu những vi phạm nguyên lý SOLID đã phát hiện được

- Áp dụng các Design Pattern đã học vào hệ thống.

**1.2.6. *Kết quả dự kiến***

Kết quả dự kiến sẽ là một mã nguồn dễ đọc, dễ hiểu, dễ kiểm thử, dễ tái sử dụng và dễ mở rộng. Các tên biến, tên hàm và tên lớp được đặt một cách rõ ràng và mô tả chính xác mục đích và nhiệm vụ của chúng. Cấu trúc code được sắp xếp một cách hợp lý và dễ theo dõi, giúp giảm độ phức tạp và tăng tính tái sử dụng của mã nguồn.

## 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

Retrived from www.elsevier.com/locate/eswa

# Đánh giá thiết kế cũ

## Nhận xét chung

Mã nguồn đáp ứng được những chức năng cơ bản của phần mềm. Tuy nhiên mã nguồn hiện tại khó để mở rộng, nếu cần phát triển tính năng mới thì rất khó khăn do vi phạm các quy tắc SOLID, clean code, chưa áp dụng nhiều design pattern và vi phạm các quy tắc về coupling, cohesion.

## Đánh giá các mức độ coupling và cohesion

Trong lĩnh vực thiết kế hệ thống phần mềm, mức độ coupling của một thiết kế được xác định bằng sự phụ thuộc, gắn kết giữa hai hay nhiều classes, modules. Một thiết kế được coi là low coupling khi một class, module này bị thay đổi, sẽ không gây nhiều ảnh hưởng tới các class, module khác mà nó liên kết tới. Ngược lại, một thiết kế được coi là high coupling khi các class, module bị kết dính quá chặt chẽ với nhau, một thay đổi của thành phần này gây ra nhiều sự thay đổi của các thành phần liên quan tới nó, dẫn tới việc hệ thống rất khó thay đổi và bảo trì.

Không giống với coupling, mức độ cohesion của một class, module, package được xác định bằng độ liên kết chặt chẽ của các thành phần bên trong một class, module, package đó. Một thiết kế được coi là low cohesion khi module đó thực hiện quá nhiều công việc, mà không tập trung vào một công việc cụ thể nào, ví dụ như các package Utils trong thực tế thường được sử dụng để chứa nhiều hàm tiện ích của hệ thống mà có ít liên quan với nhau. Và ngược lại, một thiết kế được coi là high cohesion khi class, module, package chỉ tập trung vào việc thực hiện một nhiệm vụ, mục tiêu cụ thể và duy nhất.

Tổng kết lại, một thiết kế phần mềm tốt là một thiết kế có mức độ cohesion cao và coupling thấp (high cohesion và low coupling).

### Coupling

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Coupling* | *Module* | *Mô tả* | *Lý do* |
| 1 | Content Coupling | Cart.java | Tại phương thức getListMediaCart(), phương thức trả về một List các đối tượng CartItem. | Vi phạm việc truy cập trái phép tới các đối tượng CartItem, vì nó trả về tham chiếu các đối tượng lớp CartItem nên ta hoàn toàn có thể sử dụng và sửa đổi trực tiếp dữ liệu của các đối tượng lớp CartItem |
| 2 | Content Coupling | Order.java | Tại phương thức getListOrderMedia(), phương thức trả về một List các đối tượng OrderItem | Vi phạm việc truy cập trái phép tới các đối tượng Order, vì nó trả về tham chiếu các đối tượng lớp OrderItem nên ta hoàn toàn có thể sử dụng và sửa đổi trực tiếp dữ liệu của các đối tượng lớp OrderItem |
| 3 | Content Coupling | ApplicationProgrammingInterface.java | Một lớp trừu tượng có chức năng cung cấp một giao diện để thực thi các yêu cầu API (Application Programming Interface) | Lớp đã sử dụng chỉ định truy cập setAccessible để get thuộc tính lớp khác. Việc đó đã vi phạm content coupling |
| 4 | Content Coupling | MyMap.java | Lớp này cung cấp các phương thức để chuyển đổi giữa MyMap và chuỗi JSON, cũng như chuyển đổi giữa đối tượng Java và MyMap | Thay đổi phạm vi truy cập của đối tượng lớp Field sẽ làm vi phạm tính đóng gói |
| 5 | Data Coupling | HomeController.java, PaymentController.java, PlaceOrderController.java, ViewCartController.java | Các lớp quản lý lần lượt các luồng sự kiện của trang home, payment, placeorder, viewcart | Các lớp kế thừa từ lớp BaseController, do đó có một phụ thuộc dữ liệu vào lớp cha. |
| 6 | Common coupling | Utils.java | Lớp quản lý việc hiển thị ngày tháng và trả về đối tượng Logger | Các lớp có thể truy cập vào dữ liệu global là DateFormat, vi phạm common coupling |
| 7 | Common Coupling | SessionInformation.java | Lớp quản lý các thông tin của 1 session, bao gồm: user, cartInstance và thời gian truy cập | Các dữ liệu User, cartInstance và localDateTime đều là dữ liệu global và có thể được truy cập bởi các module khác, dẫn đến common coupling. |
| 8 | Stamp Coupling | DeliveryInfo.java | Phương thức calculateShippingFee(Order order) | Mặc dù truyền tham số order vào nhưng hiện tại phương thức này không sử dụng đến |
| 9 | Stamp Coupling | InterbankSubsystemController.java | Phương thức refund(CreditCard card, int amount, String contents) | Mặc dù truyền các tham số như mô tả nhưng không dùng, chỉ trả về giá trị null |
| 10 | Stamp Coupling | IntroScreenHandler.java | Phương thức setupData có Tham số dto không được sử dụng | Truyền thừa thông tin dẫn đến vi phạm. |
| 11 | Control coupling | PlaceOrderController.java | Tùy thuộc vào giá trị của info mà điểu chỉnh luồng hoạt động | Hệ thống trở nên phức tạp hơn và khó khăn trong việc hiểu và bảo trì. |
| 12 | Control coupling | AuthencationController.java | Tùy thuộc vào giá trị của mainUser mà điểu chỉnh luồng hoạt động | Hệ thống trở nên phức tạp hơn và khó khăn trong việc hiểu và bảo trì. |

### Cohesion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Cohesion* | *Module* | *Mô tả* | *Lý do* |
| 1 | Communicational | Cart.java | Đại diện cho 1 entity cụ thể, và cung cấp các phương thức để quản lý và truy xuất các thuộc tính của entity đó | Các method trong class đều sử dụng dữ liệu từ cartItem |
| 2 | Communicational Cohesion | AuthenticationController.java | Các phương thức getMainUser(), login(), logout() đều làm việc trên dữ liệu người dùng được lưu ở lớp SesssionInformation | Các phương thức trên được gom vào 1 lớp vì lý do cùng làm việc trên dữ liệu về thông tin mainUser, cartInstance, expiredTime của lớp SessionInformaiton, do đó vi phạm Communicational Cohesion |
| 3 | Procedural  Cohesion | PlaceOrderController.java | Có các method placeOrder(), createOrder(), processDeliveryInfo() và các method validate, createInvoice() | Các methods trên thực hiện theo quy trình xử lý khi đặt hàng |
| 4 | Coincidental cohesion | Utils.java | Lớp quản lý việc hiển thị ngày tháng và trả về đối tượng Logger | Các phương thức định dạng ngày tháng và trả về đối tượng Logger hoàn toàn không liên quan đến nhau |
| 5 | Coincidental  Cohesion | \utils\Utils.java | Gồm 2 biến DATE\_FORMATTER và Logger | DATE\_FORMATTER và Logger không hề liên quan gì đến nhau |
| 6 | Logical cohesion | PlaceOrderController.java | Lớp quản lý các phương thức xác nhận đơn hàng, cùng với đó là tạo order và hóa đơn | Các phương thức validate không thực sự liên quan đến nhau |
| 7 | Logical cohesion | PopupScreen.java | Lớp quản lý các giao diện popup | Các phương thức hiển thị message popup không liên quan đến nhau |
| 8 | Temporal cohesion | Các class screenHandler | Các lớp quản lý việc hiển thị các giao diện | Các phương thức liên quan đến trình tự thực hiện |

## Đánh giá việc tuân theo SOLID

Bản thân mã nguồn cũ hiện đang vi phạm các nguyên tắc của tính SOLID nên dĩ nhiên khi thêm các yêu cầu mới, việc chỉnh sửa mã nguồn sẽ làm xuất hiện thêm các vi phạm

### SRP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | DeliveryInfo.java | Class DeliveryInfo đang có chứa một method dùng để tính toán chi phí vận chuyển. | Class DeliveryInfo là một entity vậy nên việc tính toán chi phí vận chuyển không phải là nhiệm vụ của nó.Nên tách hàm tính toán chi phí vận chuyển ra một class riêng và truyền vào DeliveryInfo. |
| 2 | AuthenticationController.java | Chịu trách nhiệm quản lý quá trình xác thực và phiên đăng nhập của người dùng trong hệ thống. Nó cung cấp các phương thức để kiểm tra trạng thái phiên đăng nhập, đăng nhập, đăng xuất và mã hóa mật khẩu. | Ngoài các method liên quan đến Authen(Login,logout, getMainUser, isAnonymousSession) thì method md5 lại không liên quan. Nên tách hàm sinh mã băm ra một class riêng và truyền vào AuthenticationController. |
| 3 | PlaceOrderController.java | Chịu trách nhiệm điều khiển quy trình đặt hàng trong hệ thống. Nó cung cấp các phương thức để kiểm tra tính khả dụng của sản phẩm, tạo đơn hàng và hóa đơn, xử lý thông tin vận chuyển và xác thực thông tin. | Class nên chỉ xử các phương thức liên quan đến placeOrder, các method validate nên tách thành một class riêng |
| 4 | HomeScreenHandler.java | Xử lý màn hình trang chủ của ứng dụng | Thực hiện quá nhiều chức năng khác nhau như: Lớp này quản lý giao diện màn hình chính; Xử lý logic liên quan đến hiển thị danh sách media, xử lý sự kiện click, xử lý đăng nhập, quản lý giỏ hàng, cập nhật dữ liệu và nhiều hơn nữa.  Ngoài ra phương thức “update” thực hiện nhiều chức năng, bao gồm kiểm tra số lượng media, xử lý giỏ hàng và hiển thị thông báo |
| 5 | MediaDAO | Module này đảm nhiệm các hoạt động chung liên quan đến việc truy xuất và cung cấp thông tin về các đối tượng media từ cơ sở dữ liệu(Vi phạm SRP) | Thực hiện quá nhiều chức năng khác nhau như: Lấy danh sách tất cả các media từ cơ sở dữ liệu (getAllMedia); Lấy thông tin của một media cụ thể bằng ID từ cơ sở dữ liệu(getMediaById); Cập nhật một trường cụ thể của media bằng ID trong cơ sở dữ liệu(updateMediaFieldById) |

### OCP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | BookDAO | Lớp vi phạm OCP | Nếu muốn thêm 1 sản phẩm thì cần thêm 1 hàm getMediaById() |
| 2 | AuthenticationController.java | Lớp chứa các phương thức xác thực người dùng, chứa phương thức login() | Phương thức login(): ví dụ sau này muốn thay đổi cách xác thực (có thể qua vân tay hoặc giọng nói thì cần thay đổi trực tiếp mã nguồn) |
| 3 | DeliveryInfo.java | Chứa thông tin của delivery và phương thức tính phí ship calculateShippingFee() | Hiện tại calculateShippingFee() chỉ tính phí ship theo khoảng cách, sau này nếu thêm hình thức tính phí ship như cân nặng thì phải modify mã nguồn |
| 2 | CDDAO | Lớp vi phạm OCP | Nếu muốn thêm 1 sản phẩm thì cần thêm 1 hàm getMediaById() |

### LSP

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | BaseController.java | Các lớp controller AuthenticationController, HomeController, PaymentController kế thừa BaseController có các phương thức checkMediaInCart(), getListCartMedia() | Các lớp trên kế thừa BaseController là không hợp lý vì nó không sử dụng hay liên quan tới việc check và lấy dữ liệu từ giỏ hàng |
| 2 | PopupScreen.java | Kế thừa BaseScreenHandler | Không thể sử dụng các phương thức của BaseScreenHandler như getPreviousScreen(), getBController()... |
| 3 | HomeScreenHandler. | Chịu trách nhiệm quản lý giao diện và xử lý logic liên quan đến màn hình chính của ứng dụng | Lớp không dùng 2 phương thức setPreviousScreen và getPreviousScreen của lớp cha BaseScreenHandler |

***2.3.4 ISP***

Không phát hiện vi phạm

***2.3.5 DIP***

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | AuthenticationController.java | Có phương thức login gọi đối tượng của UserDAO để thực hiện authenticate thông tin email và password | Phụ thuộc vào phương thức của lớp cụ thể, khi muốn login theo thông tin khác sẽ dẫn đến thay đổi mã nguồn.ên phụ thuộc vào 1 lớp abstract |
| 2 | Order.java | Phương thức setDeliveryInfo() đang tính phí ship phụ thuộc vào phương thức cụ thể calculateShippingFee của deliveryInfo | Phụ thuộc vào phương thức tính phí ship của 1 lớp cụ thể là tính theo khoảng cách, khi thêm phương thức tính phí mới sẽ phải modify mã nguồn Nên phụ thuộc vào 1 phương thức abstract có nhiệm vụ tính phí ship |
| 3 | InterbankInterface,java | Phương thức payOrder chỉ nhận các thông tin của CreditCard | Phụ thuộc vào Credit card mà không phải 1 abstract class |

## Các vấn đề về Clean Code

### Clean Name

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | AuthenticationController.java | Tên biến chưa thể hiện rõ chức năng | md5 -> md5Encryption |
| 2 | HomeScreenHandler.java | Phương thức setupData() | Có biến medium bị sai chính tả. Nên đổi tên biến medium thành mediaList |
| 3 | CartScreenHandler.java | Phương thức setupFunctionality() | Có biến im được đặt tên không có ý nghĩa rõ ràng. Nên đổi im ⇒ image |
| 4 | InvoiceScreenHandler.java | Phương thức setupData() | Có n mis được đặt tên không có ý nghĩa rõ ràng, nên đổi mis ⇒ mediaInvoiceScreenHandler |
| 5 | PaymentScreenHandler.java | Phương thức confirmToPayOrder() | Có biến ctrl được đặt tên chưa có ý nghĩa rõ ràng, nên đổi ctrl ⇒ paymentController |
| 6 | InterbankSubsystem.java | Biến viết tắt chưa rõ nghĩa | ctrl -> interbankSubsystemController |

### Clean Function/Method

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Module* | *Mô tả* | *Lý do* |
| 1 | PlaceOrderController.java | Cả hai method validateName() và validateAddress() đều có cùng logic xử lí. | Bị duplicate code vì vậy cần viết lại thành 1 hàm |
| 2 | AuthenticationController.java | Method md5 | Do md5 là tên một thuật toán dùng để mã hóa, nên cần sửa tên method md5 thành md5Encryption để thể hiện rõ ràng nhiệm vụ của hàm này là sinh một message-digest sử dụng hàm md5. |
| 3 | HomeScreenHandler.java | Method update có hai vị trí log ra cùng message giống nhau, nhưng không khai báo biến để dùng chung mà sử dụng hard-code | Tạo biến errorMessage để lưu các giá trị log giống nhau |
| 4 | ViewCartController. | Method getCartSubtotal | Sửa phần cài đặt của method thành inline function |

### Clean Class

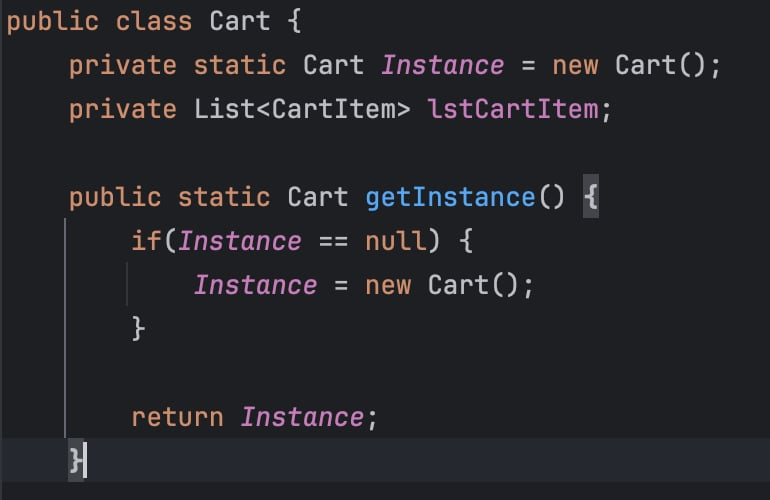
|  |  |  |
| --- | --- | --- |
| # | Vấn đề tồn tại | Danh sách class |
| 1 | Có các attribute được khai báo nhưng không được sử dụng | Utils  ApplicationProgrammingInterface |
| 2 | Có các method được khai báo nhưng không được sử dụng | Invoice  MediaDAO  PopupScreen  ShippingScreenHandler |
| 3 | Các lớp không được sử dụng | OrderDAO  OrderItemDAO  InvoiceDAO |

## Các vấn đề khác

# Đề xuất cải tiến

## Sử dụng Singleton với Cart

Nên dùng Singleton với Cart vì tránh tạo nhiều giỏ hàng : đảm bảo rằng chỉ có một instance duy nhất của giỏ hàng tồn tại giúp tránh việc tạo nhiều giỏ hàng không cần. Sử dụng Singleton đảm bảo rằng giỏ hàng được chia sẻ và cập nhật nhất quán.



## 3.2. Cải tiến vấn đề thay đổi phương thức tính khoảng cách sử dụng thư viện mới và thay đổi công thức tính phí vận chuyển

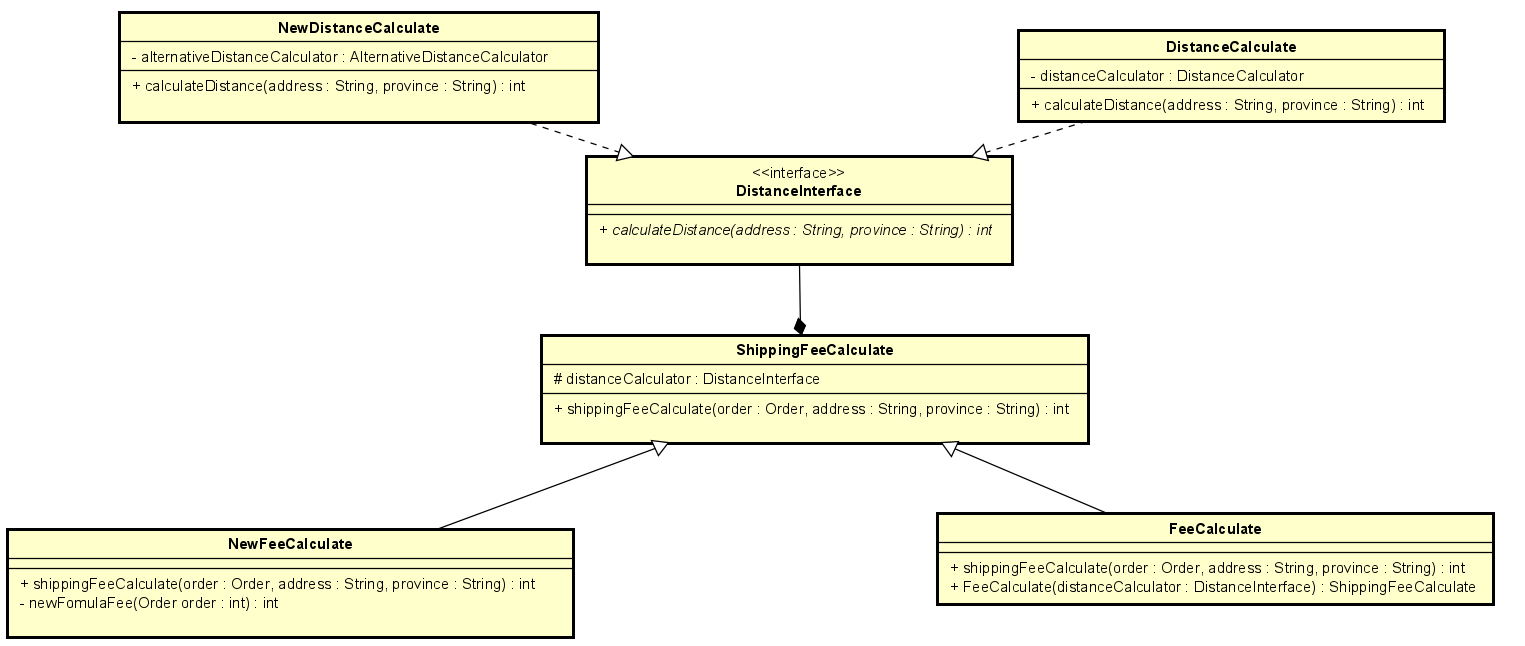
Để sử dụng thư viện tính khoảng cách mới Alt-Distance-API, ta phải sửa class DeliveryInfo do Class DeliveryInfo có một thuộc tính có kiểu dữ liệu là DistanceCalculator thuộc thư viện cũ Distance-API.

Công thức tính phí mới dựa trên khối lượng, độ cồng kềnh và khoảng cách nằm ở trong class Order nên ta chuyển việc tính phí và khoảng cách cho class Order sẽ dễ dàng hơn thay vì thực hiện ở class Delivery Info như hiện tại.

Bằng cách áp dụng Adapter Pattern, chúng ta thiết lập một interface cho phương thức tính khoảng cách, và bất kỳ class nào muốn sử dụng phương thức này sẽ phải implement interface đó. Việc tách riêng thành interface sẽ giúp chúng ta dễ dàng đáp ứng các yêu cầu mới chỉ bằng cách tạo một class mới implement interface này, mà không làm ảnh hưởng đến các class đã implement trước đây.

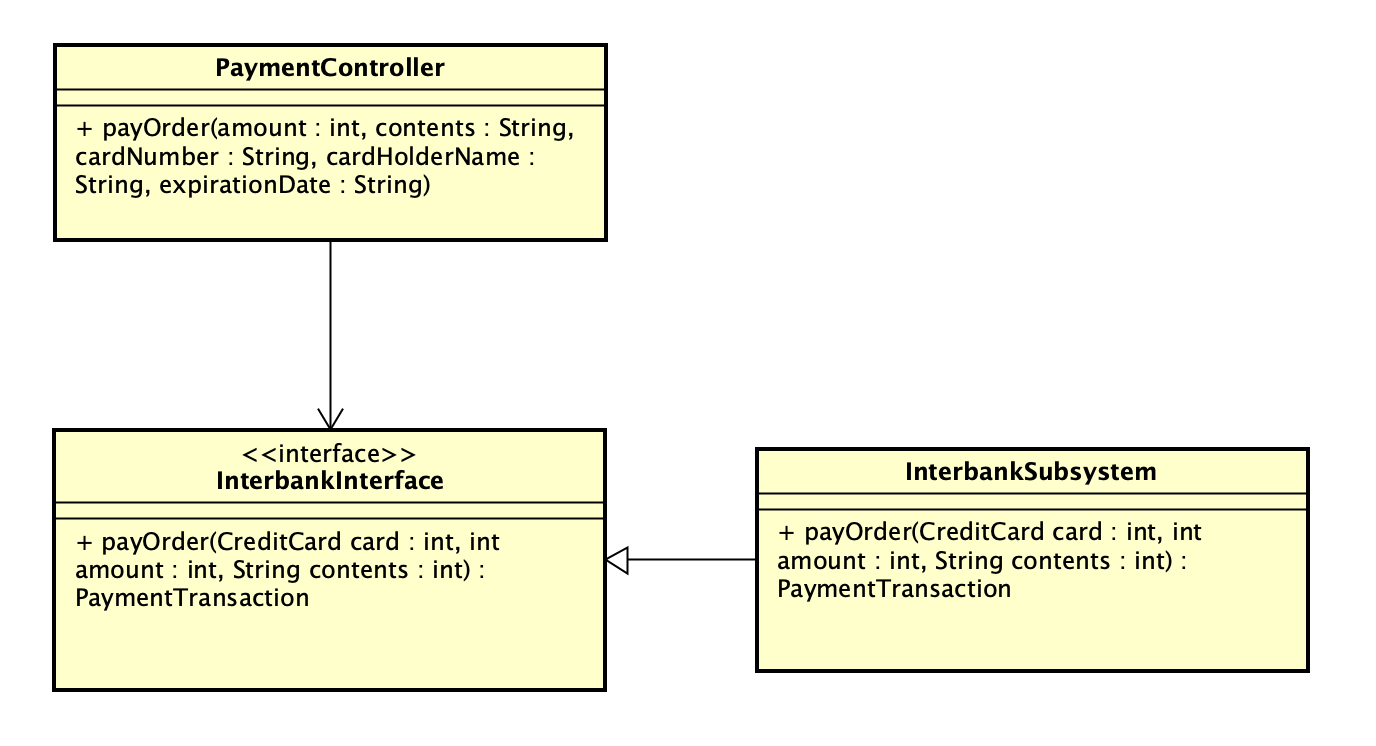
Để xử lí yêu cầu tiếp theo là thay đổi công thức tính phí vận chuyển ta áp dụng Strategy Pattern, tạo một abstract class ShippingFeeCalculate. Khi xuất hiện yêu cầu thay đổi ta chỉ cần tạo class extends abstract class này.

Biểu đồ lớp sau khi áp dụng thiết kế mới:



## 3.3. Thêm phương thức thanh toán mới

Thiết kế cũ:

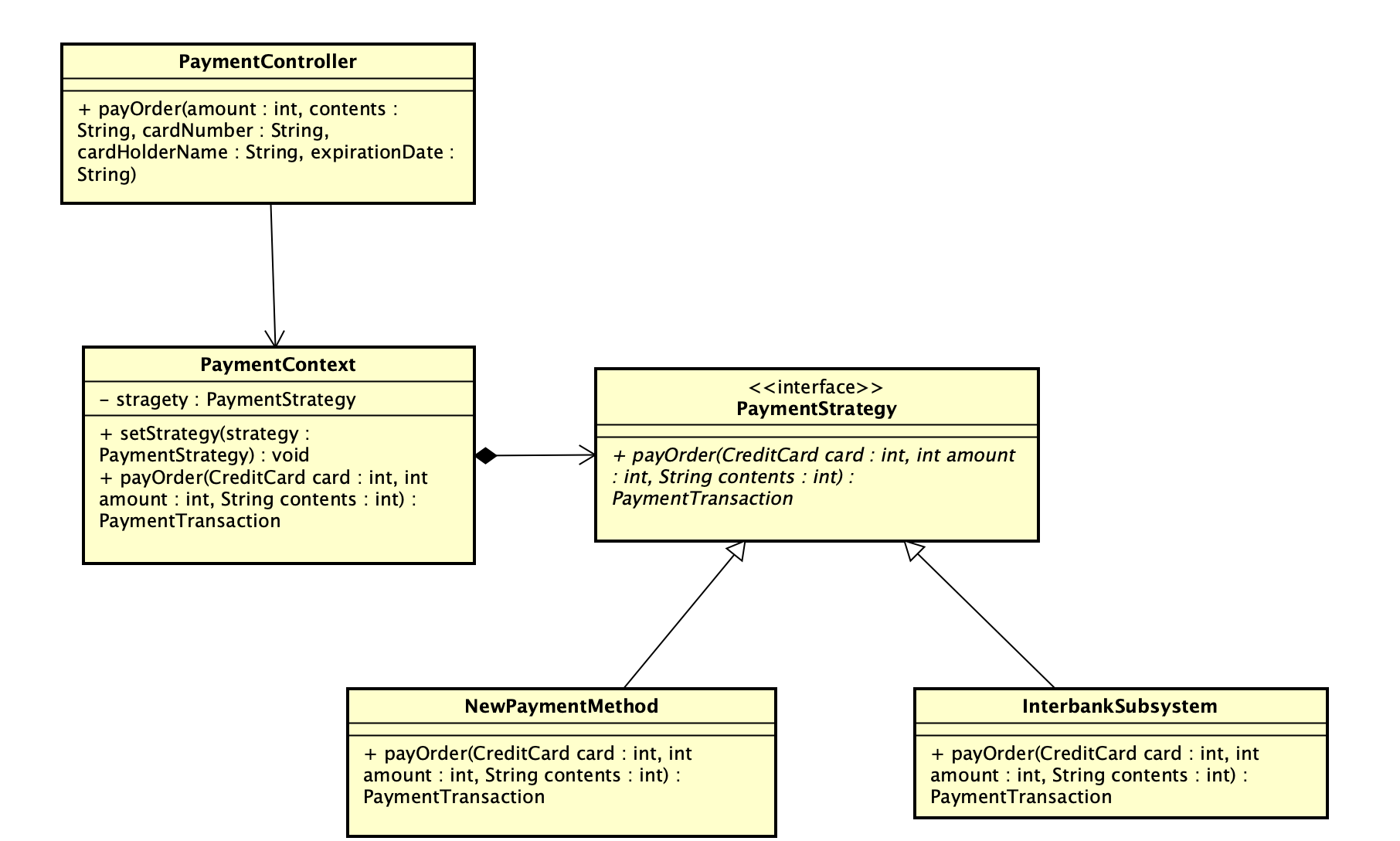


Sử dụng Strategy Pattern để thực hiện thanh toán, giúp dễ mở rộng hơn khi có thêm phương thức thanh toán mới.

Bằng cách thêm class mới từ implement PaymentStrategy, khi thực hiện đổi phương thức thanh toán. Gọi hàm SetStrategy() để chuyển khi muốn đổi phương thức thanh toán khác.

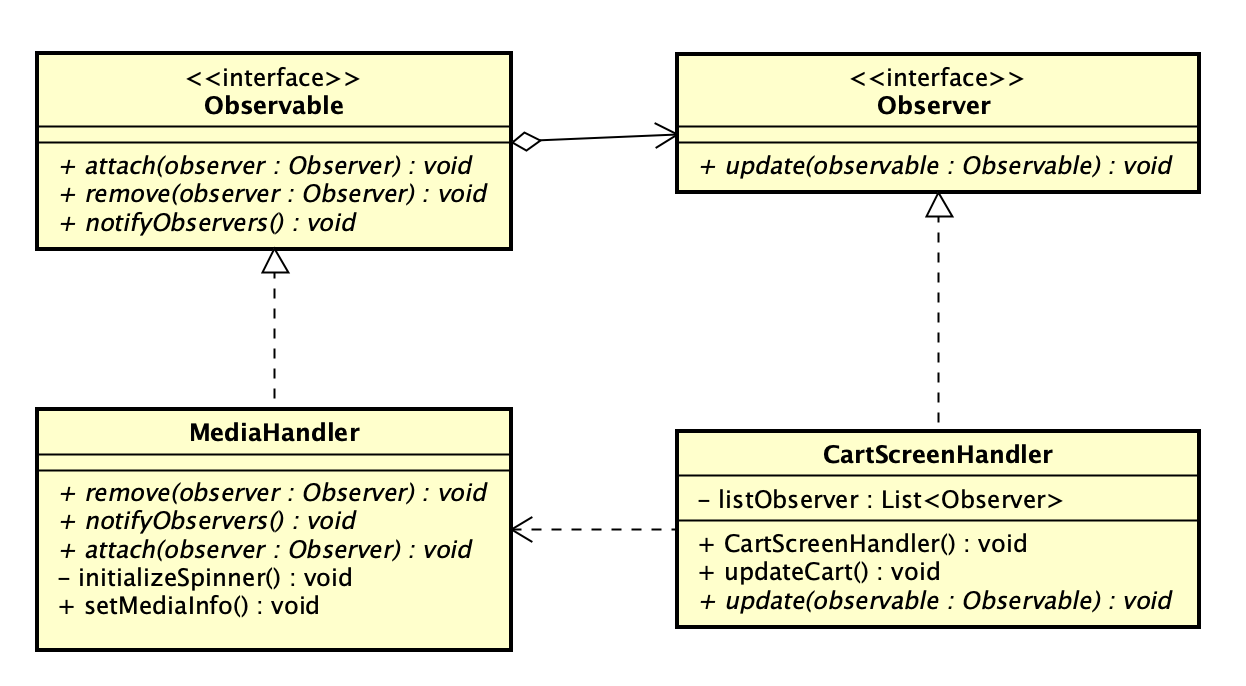
Ưu điểm di dùng là dễ mở rộng, bảo trì, tái sử dụng mã nguồn và dễ thêm phương thức thanh toán mới.

Thiết kế mới:



## 3.4. Sử dụng Observer Pattern để update Cart

Sử dụng Observer pattern cho CartScreenHandler là lớp Observer và MediaHandler là lớp Subject giúp cho việc update Cart khi MediaHandler thay đổi



# Tổng kết

## Kết quả tổng quan

Sau khi hoàn thành bài tập lớn lần này, dự án cửa hàng online AIMS đã có thiết kế tốt hơn. Mã nguồn về cơ bản đã đáp ứng được các nguyên lý SOLID, từ đây trong tương lai nếu có những yêu cầu mới hay cần sửa đổi mã nguồn thì những việc phát sinh đó sẽ được xử lý dễ dàng hơn trước. Thêm vào đó các vấn đề clean code như clean name, clean method, clean class cũng đã được khắc phục. Do đã sửa đổi theo nguyên lý SOLID mà các vấn đề coupling cohesion trước đó cũng không còn nữa.

Qua việc làm bài tập lớn lần này đã giúp các thành viên trong nhóm cải thiện kiến thức một cách rõ rệt, mỗi thành viên đều đã biết vận dụng kiến thức trong việc tái cấu trúc mã nguồn, thiết kế mã nguồn làm sao cho hợp lý và hiệu quả nhất.

## Các vấn đề tồn đọng

Như đã nói, vẫn còn một số vấn đề, hạn chế mà nhóm chưa thực hiện được. Một số pattern nhóm chưa thể tìm ra được giải pháp và đưa vào việc cải tiến mã nguồn. Ngoài ra, quá trình sửa đổi mã nguồn, cũng có những phát sinh, tồn đọng mà nhóm đã bỏ sót hoặc chưa phát hiện, kiểm soát, kính mong cô và mọi người thông cảm