ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Báo cáo Mẫu thiết kế phần mềm

Version 1.0

<Mã nhóm>

<Danh sách thành viên>

*Hà Nội,* *<tháng, năm>*

*<Tất cả chú thích trong dấu ngoặc nhọn không thuộc về nội dung của báo cáo cuối, chỉ sử dụng nhằm mục đích hướng dẫn. Khi sử dụng mẫu báo cáo này, cần loại bỏ hết tất cả chú thích và/hoặc thay thế bằng nội dung tương ứng với chú thích.>*

*<Mẫu báo cáo này thuộc về GV. Nguyễn Thị Thu Trang và GV. Bùi Thị Mai Anh, được sử dụng cho các sinh viên tham gia môn Mẫu thiết kế phần mềm. Bất kỳ chỉnh sửa và/hoặc sử dụng nằm ngoài sự đồng ý của tác giả đều hoàn toàn nghiêm cấm.>*

Mục lục

Mục lục 1

1 Tổng quan 3

1.1 Mục tiêu 3

1.2 Phạm vi 3

1.3 Danh sách thuật ngữ 3

1.4 Danh sách tham khảo 3

2 Đánh giá thiết kế cũ 4

2.1 Nhận xét chung 4

2.2 Đánh giá các mức độ coupling và cohesion 4

2.2.1 Coupling 4

2.2.2 Cohesion 4

2.3 Đánh giá việc tuân theo SOLID 4

2.3.1 SRP 5

2.3.2 OCP 5

2.3.3 ... 5

2.4 Các vấn đề về Clean Code 5

2.4.1 Clear Name 5

2.4.2 Clean Function/Method 5

2.4.3 Clean Class 5

2.5 Các vấn đề khác 6

3 Đề xuất cải tiến 7

3.1 Vấn đề xxx và giải pháp 7

3.2 Vấn đề yyy và giải pháp 7

4 Tổng kết 8

4.1 Kết quả tổng quan 8

4.2 Các vấn đề tồn đọng 8

**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 nội dung sau mô tả khái quát nội dung báo cáo>*

## Mục tiêu

<*Xác định mục đích sử dụng của báo cáo mẫu thiết kế phần mềm. Xác định đối tượng người đọc là ai trong quy trình xử lý phần mềm. Mô tả nội dung khái quát của báo cáo*>

## Phạm vi

<

*Trong mục này, cần làm rõ:*

1. *Mô tả khái quát phần mềm*
2. *Các chức năng chính của phần mềm*
3. *Cấu trúc mã nguồn*
4. *Các yêu cầu thêm cần cân nhắc cùng quá trình tái cấu trúc*
5. *Các hoạt động (Ví dụ: review, refactor) thực thi trên mã nguồn để đạt được mục tiêu kể trên*
6. *Kết quả dự kiến*

>

## Danh sách thuật ngữ

*<Danh sách và giải thích thuật ngữ sử dụng trong báo cáo. Bất kỳ giả định nào về kiến thức của người đọc cần cân nhắc thật kỹ>*

## 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

*<Danh sách các tài liệu sử dụng liên quan đến báo cáo. Bao gồm cả các tài liệu thêm cung cấp trong quá trình thực hiện tái cấu trúc>*

# Đánh giá thiết kế cũ

<*Mục này bao gồm kết quả code review trên mã nguồn case study>*

## Nhận xét chung

*<Đưa ra nhận xét chung về chất lượng mã nguồn case study hiện tại. Hiệu năng ứng biến của code đối với các kế hoạch update trong tương lai>*

## Đánh giá các mức độ coupling và cohesion

*<Sau quá trình học, đúc kết và giải thích ngắn gọn vai trò và ý nghĩa của coupling, cohesion.>*

### Coupling

*<Nhận xét tổng quát về mức độ phụ thuộc kết dính giữa các module>*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Coupling* | *Module* | *Mô tả* | *Lý do* |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

### Cohesion

*<Nhận xét tổng quát về mức độ liên kết chặt chẽ giữa các submodule>*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Các mức độ về Cohesion* | *Module* | *Mô tả* | *Lý do* |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

## Đánh giá việc tuân theo SOLID

*<Nếu phải chỉnh sửa mã nguồn theo các yêu cầu phát sinh thì bản thiết kế và mã nguồn ban đầu có tuân theo các nguyên lý thiết kế SOLID hay không?>*

### SRP

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

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

### OCP

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

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

### ...

*<Tương tự>*

## Các vấn đề về Clean Code

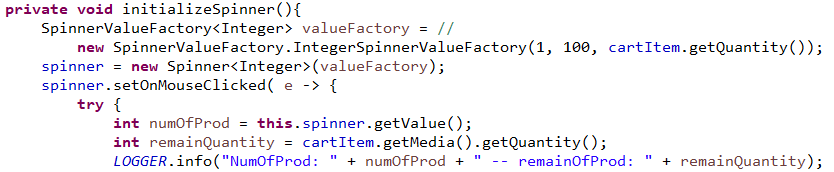
### Clear Name

*<Nhận xét tình trạng mã nguồn ban đầu có đáp ứng clear name hay không?>*

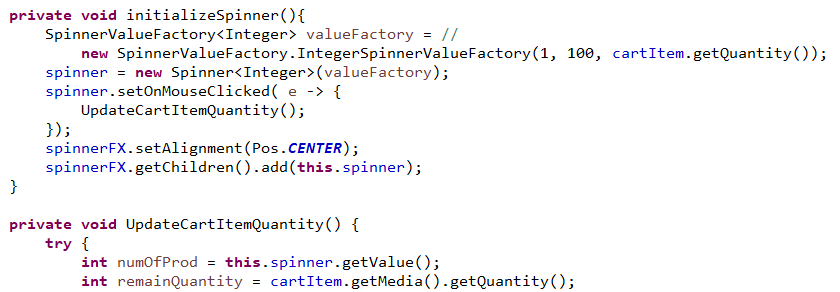
### Clean Function/Method

**case 1**

Class views.screen.cart.MediaHandler, method initializeSpinner(), có 2 nhiệm vụ, (i) tạo ra Spinner, (ii) xử lý sự kiện khi click vào spinner. Việc này làm giảm tính dễ đọc của code.

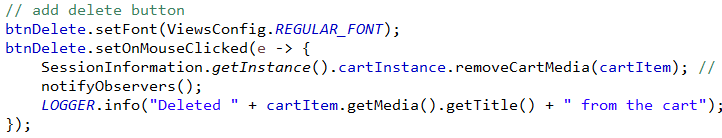


Cách xử lý: Xử lý ở method-level, bằng cách tách phần xử lý khi click spinner thành một private method riêng.

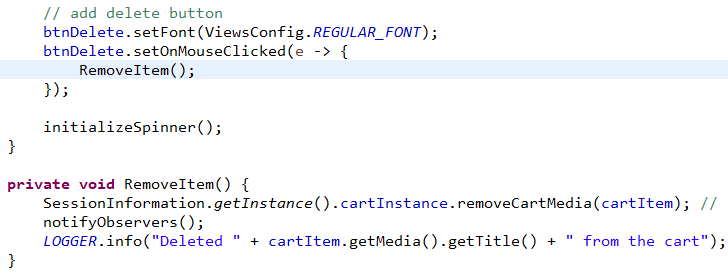


**case 2**

Class views.screen.cart.MediaHandler, method setMediaInfo() có 2 nhiệm vụ, vừa thiết lập các thông số về đồ họa, vừa xử lý sự kiện khi click vào nút Delete. Việc này làm giảm tính dễ đọc của code.

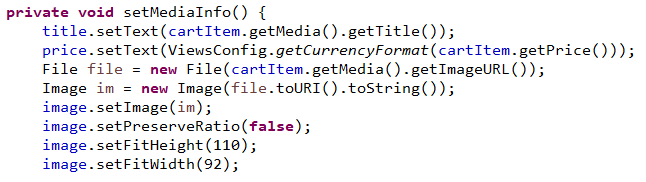
****

Cách xử lý: tách phần xử lý khi click nút Delete thành một phương thức riêng.



**case 3**

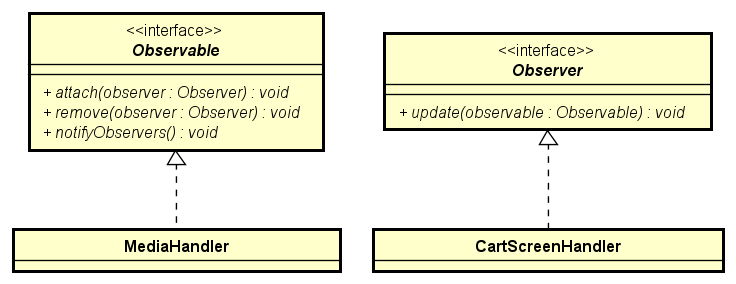
Class views.screen.cart.MediaHandler, method setMediaInfo() có chứa các đoạn mã thiết lập chiều dài và chiều rộng của ảnh với giá trị cố định. Điều này vi phạm nguyên lý OCP, vì các thông số này có khả năng cao sẽ thay đổi trong tương lai, nếu muốn thay đổi sẽ cần sửa lại code của class này. Đồng thời, các thông số này có thể thay đổi trong lúc thực thi chương trình, trong trường hợp ứng dụng muốn responsive cho nhiều loại kích cỡ màn hình.



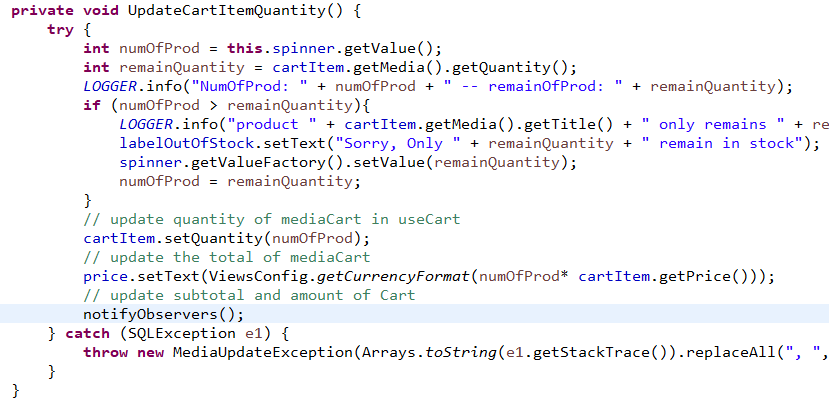
Case 4

Hai lớp views.screen.cart.MediaHandler, views.screen.cart.CartScreenHandler có sự phụ thuộc lẫn nhau. MediaHandler chứa reference đến CartScreenHandler và CartScreenHandler phụ trách tạo ra MediaHandler.

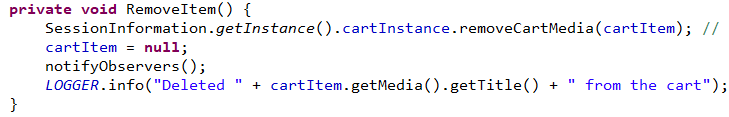
Cách xử lý: sử dụng Observer pattern, với MediaHandler implement Observable và CartScreenHandler implement Observer.



MediaHandler có hai sự kiện cần thông báo cho Observer: (i) khi người dùng xóa một mặt hàng và (ii) khi người dùng thay đổi số lượng một mặt hàng.

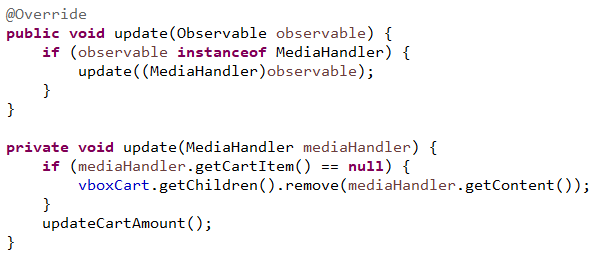


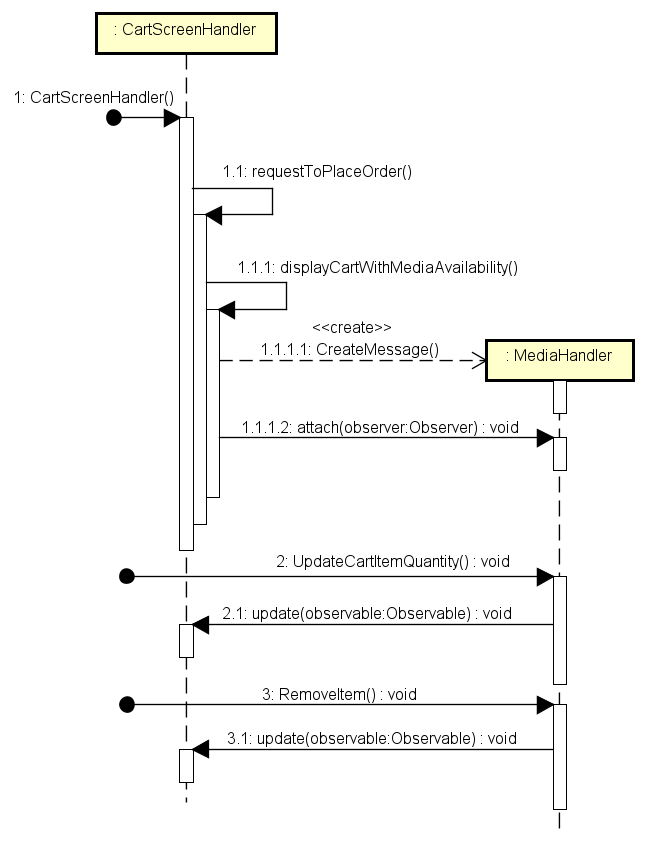
Cần có cách để nhận biết khi MediaHandler thông báo update() tới Observer, đó là cập nhật cho việc xóa hay thay đổi số lượng mặt hàng. Ta sẽ nhận biết thay đổi này thông qua thuộc tính cartItem. Khi xóa, ta set cartItem=null, đồng thời cung cấp phương thức get cho thuộc tính này.



Trong code base ban đầu, tại CartScreenHandler, mỗi lần delete một mặt hàng, ta phải tạo lại danh sách hiển thị các mặt hàng từ đầu, sử dụng hàm updateCart(). Điều này không cần thiết, làm giảm hiệu năng của chương trình. Thay vì gọi hàm updateCart(), ta chỉ cần xóa đối tượng hiển thị trong vboxCart.

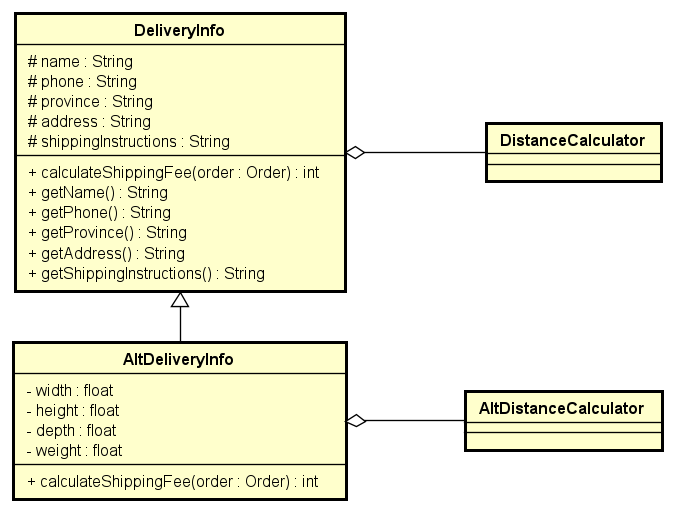
Cả hai sự kiện xóa và thay đổi số lượng mặt hàng đều cần cập nhật lại tổng giá tiền, do đó luôn gọi hàm updateCartAmount() trong hàm update().





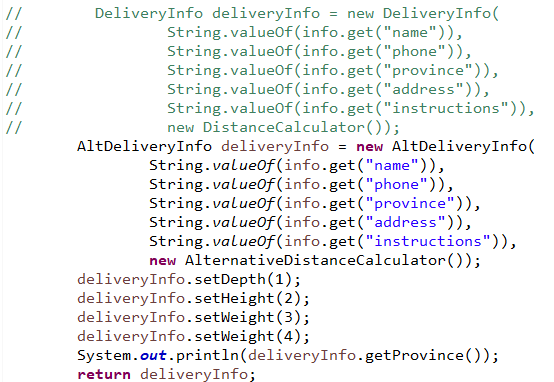
**Case 5**

Yêu cầu thay đổi công thức tính vận chuyển. Với việc sử dụng thư viện tính khoảng cách mới, nếu không muốn sửa lại các lớp hiện có, ta cần sử dụng adapter.



Ta tạo thêm lớp AltDeliveryInfo kế thừa từ DeliveryInfo, bổ sung thêm các thuộc tính cần thiết và override lại calculateShippingFee(). Bên trong phương thức này sẽ gọi tới thư viện mới để thực hiện tính toán.

Tại client, cụ thể là PlaceOrderController, tại phương thức processDeliveryInfo(), ta sửa lại cách khởi tạo DeliveryInfo:



Phương thức này vẫn giữ nguyên giá trị trả về là DeliveryInfo, do đó các thành phần khác của hệ thống sẽ không bị ảnh hưởng.

**case 6:**

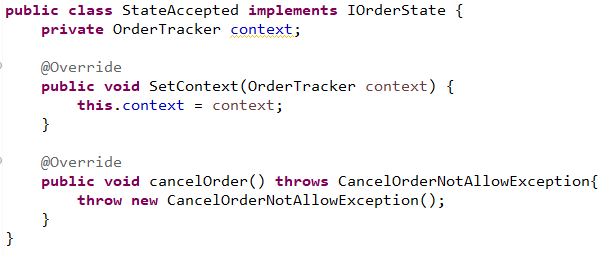
Yêu cầu cập nhật chức năng hủy đơn hàng.

Đây là chức năng mới, chưa có trong hệ thống và có đặc điểm khác với các use case đã có. Do đó, ta cần tạo một controller mới là OrderManagementController.

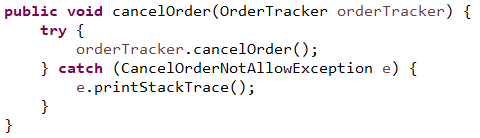
Để nhận biết được đơn hàng đã được quản trị viên chấp nhận hay chưa, ta cần thêm một trường thông tin về trạng thái đơn hàng. Để không phải chỉnh sửa lại lớp Order, ta tạo thêm một lớp OrderTracker có chứa đối tượng Order bên trong.

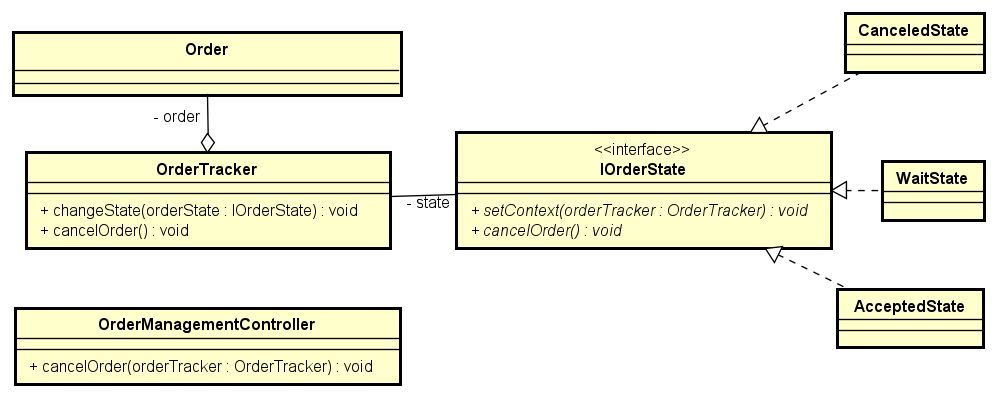
Ta sử dụng state pattern với OrderTracker là context và tạo thêm các state kế thừa từ một interface là IOrderState.

OrderTracker cung cấp phương thức cancelOrder() cho phép người dùng hủy đơn hàng, tùy thuộc vào trạng thái đơn hàng hiện tại sẽ có cách xử lý lệnh cancelOrder() khác nhau. Trường hợp không cho phép hủy đơn hàng, hàm cancelOrder() tại các state này sẽ trả về exception.



Controller sẽ bắt các exception này để xử lý, đưa thông báo cho người dùng.





### Clean Class

*<Tương tự>*

## Các vấn đề khác

*<Ngoài các vấn đề bám theo nội dung lý thuyết kể trên, các vấn đề khác được liệt kê ở đây, theo cấu trúc bảng tương tự. Cần làm rõ các vấn đề cần xem xét trước khi đưa ra ví dụ.>*

# 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 xxx và giải pháp

## Vấn đề yyy và giải pháp

# Tổng kết

***<Đúc kết lại hiệu năng của thiết kế sau khi tái cấu trúc. Mô tả các vấn đề còn sót lại hoặc chưa kịp thực hiện. >***

## Kết quả tổng quan

*<So sánh với dự kiến ban đầu, kết quả đã thực hiện được hoàn thiện được bao nhiêu kỳ vọng*>

## Các vấn đề tồn đọng

*<Nêu ra các vấn đề còn sót lại trong mã nguồn. Đánh giá khả năng xử lý*>