Assignment 08

Class Design

Chuẩn bị

* Người học cần cài đặt sẵn Eclipse và môi trường Java11 trên máy cá nhân
* Clone project từ: <https://github.com/leminhnguyen/AIMS-Student>

Mục đích và nội dung

### 5.3.1. Thiết kế lớp (Class Design)

#### 5.3.1.1. Bước đầu tạo các lớp thiết kế

Trong phần này, chúng ta sẽ ánh xạ các thành phần thiết kế (design elements, ví dụ: lớp – class, nhóm các lớp – group of classes, gói – package, subsystem) từ các lớp phân tích (analysis classes). Mỗi lớp thiết kế nên chỉ phục vụ tốt một mục đích duy nhất. Chúng ta sẽ xác định các lớp thiết kế dựa vào biểu đồ lớp kiến trúc và khuôn mẫu (stereotype) của lớp đấy. Lưu ý rằng chúng ta chưa ứng dụng các mẫu thiết kế (design patterns) trong bài thực hành này.

##### Thiết kế lớp boundary

###### Lớp boundary: Giao diện người dùng (User interface)

Trong Case study, chúng ta sử dụng JavaFX để xây dựng giao diện. Do đó, từ góc nhìn kiến trúc, mỗi lớp boundary giao diện người dùng tương đương với lớp thiết kế phụ trách xử lý sự kiện hoặc hành động của người dùng (được bắt bởi FXML tương ứng). Trong JavaFX, mặc dù các lớp thiết kế này thường được gọi là “controller” của các tệp FXML, chúng không thực sự đóng vai trò như lớp control trong UML. Do vậy, đa số các lớp xử lý sự kiện hiện tại đã khá đơn giản, và ánh xạ là 1-1.

###### Lớp boundary: Giao diện hệ thống (System/device boundary)

Trong bài thực hành trước, chúng ta đã “evolve”/ánh xạ lớp boundary cho liên ngân hàng Interbank thành một subsystem. Tuy nhiên, subsystem này thiết kế chưa được tốt: InterbankSubsystemController quá phức tạp, đồng thời một phần InterbankBoundary vẫn có thể tái sử dụng. Hiện tại trong mô tả của Case Study, chỉ có nhắc đến một hệ thống thông tin web bên ngoài (Interbank), nhưng thực tế có rất nhiều hệ thống như thế mà cần giao tiếp với REST APIs sử dụng các phương thức HTTP. Trong khi các hệ thống này có giao thức kết nối giống nhau, một trong số đó có thể là một hệ thống mà AIMS Software cần giao tiếp trong tương lai. Vì vậy, vì mục đích tái sử dụng, chúng ta có thể cần một lớp mới, ví dụ là API, để phụ trách giao tiếp API qua phương thức HTTP GET và HTTP POST. Ngoài ra, chúng ta sẽ nhìn nhận các vấn đề ở lớp control trong phần sau.

##### Thiết kế lớp entity

Với bản mô tả hiện tại và đối với 2 use case “Pay Order” và “Place Order”, đa số các lớp entity trong thiết kế kiến trúc đã đơn giản, và có thể ánh xạ 1-1 với lớp thiết kế.

##### Thiết kế lớp control

Tương tự, đa số các lớp control trong thiết kế kiến trúc đã đơn giản, và có thể ánh xạ 1-1 với lớp thiết kế. Tuy nhiên, InterbankSubsystemController đang phụ trác 2 tác vụ: (1) điều khiển luồng dữ liệu và (2) chuyển đổi dữ liệu (chuyển đổi dữ liệu sang định dạng yêu cầu và xử lý kết quả trả về). Do đó chúng ta cần ít nhất một lớp khác để phụ trách chuyển đổi dữ liệu, ví dụ JSON hoặc MyMap (tùy thuộc vào thiết kế mà lớp này có thể tái sử dụng cho các hệ thống thông tin web khác).

##### Nhóm các lớp thiết kế

Sau đây là một cách để nhóm các lớp thiết kế vào các package.

Text

Description automatically generatedApplication

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated Text

Description automatically generated with medium confidence

#### 5.3.1.2. Xác định mối quan hệ giữa các lớp

Để phục vụ mục đích minh họa mối quan hệ giữa các lớp được dễ nhìn và rõ ràng, các lớp ở các hình sau không bao gồm các attribute cũng như operation/method.

Diagram

Description automatically generatedDiagram

Description automatically generated

#### 5.3.1.3. Lớp thiết kế

Trong phần này, các bước thiết thiết kế lớp sẽ được minh họa theo từng bước.

##### Lớp “InterbankInterface”

A picture containing diagram

Description automatically generated

***Attribute***

Không

***Operation***

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu trả về* | *Mô tả (mục đích)* |
| 1 | payOrder | PaymentTransaction | Thanh toán đơn hàng và trả về giao dịch thanh toán |
| 2 | refund | PaymentTransaction | Hoàn tiền và trả về giao dịch thanh toán |

*Parameter:*

* card – thẻ tín dụng để giao dịch
* amount – số tiền giao dịch
* contents – nội dung giao dịch

*Exception:*

* PaymentException – nếu mã lỗi trả về đã biết
* UnrecognizedException – nếu không tìm thấy mã lỗi trả về hoặc có lỗi hệ thống

***Method***

Không

***State***

Không

Lưu ý khi lập trình, luôn cần chú thích các thành phần được công khai (public elements). Một ví dụ chú thích cho lớp này được thể hiện trong ảnh dưới đây.

Graphical user interface, text, application

Description automatically generated

Text, timeline

Description automatically generated

Một số đường dẫn tham khảo:

<https://users.soe.ucsc.edu/~eaugusti/archive/102-winter16/misc/howToWriteJavaDocs.html>

<https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>

##### Lớp “PaymentController”

Text

Description automatically generated

***Attribute***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu trả về* | *Giá trị mặc định* | *Mô tả* |
| 1 | card | CreditCard | NULL | Represent the card used for payment |
| 2 | interbank | InterbankInterface | NULL | Represent the Interbank subsystem |

***Operation***

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu trả về* | *Mô tả (mục đích)* |
| 1 | payOrder | Map<String,String> | Thanh toán đơn hàng và trả về giao dịch thanh toán |

*Parameter:*

* amount – số tiền giao dịch
* contents – nội dung giao dịch
* cardNumber – số thẻ
* cardHolderName – tên chủ sở hữu
* expirationDate – ngày hết hạn theo định dạng "mm/yy"
* securityCode - mã bảo mật cvv/cvc

*Exception:*

* Không

***Method***

* getExpirationDate: Chuyển dữ liệu ngày từ định dạng “mm/yy” sang “mmyy”.

Diagram

Description automatically generated

***State***

Không

##### Biểu đồ trạng thái (state machine) cho đối tượng “Order”

Chúng ta chỉ cần xác định các trạng thái được đặt tên (named states) của đối tượng, từ đó nắm được các trạng thái có thể có, và sự chuyển đổi giữa các trạng thái đó khi có các sự kiện tác động. Trước tiên, chúng ta tìm những lớp có các trạng thái được đặt tên. Sau đó, tìm các trạng thái có thể cho một đối tượng của lớp đó. Cuối cùng, mô hình hóa bằng biểu đồ trạng thái.

Dễ thấy, đối tượng “Order” là một trong những đối tượng có trạng thái được đặt tên. Dưới đây là một ví dụ minh họa biểu đồ trạng thái của đối tượng từ sau khi khách hàng đặt hàng cho tới khi một quản trị viên xử lý đơn hàng.

Diagram

Description automatically generated

#### 5.3.1.4. Biều đồ lớp thiết kế

Cuối cùng, đặt tất cả các lớp thiết kế vào một biểu đồ lớp thiết kế tổng quan. Lưu ý không thể hiện chi tiết của subsystem trong biểu đồ này. Ngoài ra, nếu có quá nhiều chi tiết trong biểu đồ, có thể làm tương tự với các package và tạo biểu đồ lớp thiết kế cho từng package.

Diagram, engineering drawing, schematic

Description automatically generatedDiagram

Description automatically generated

Diagram

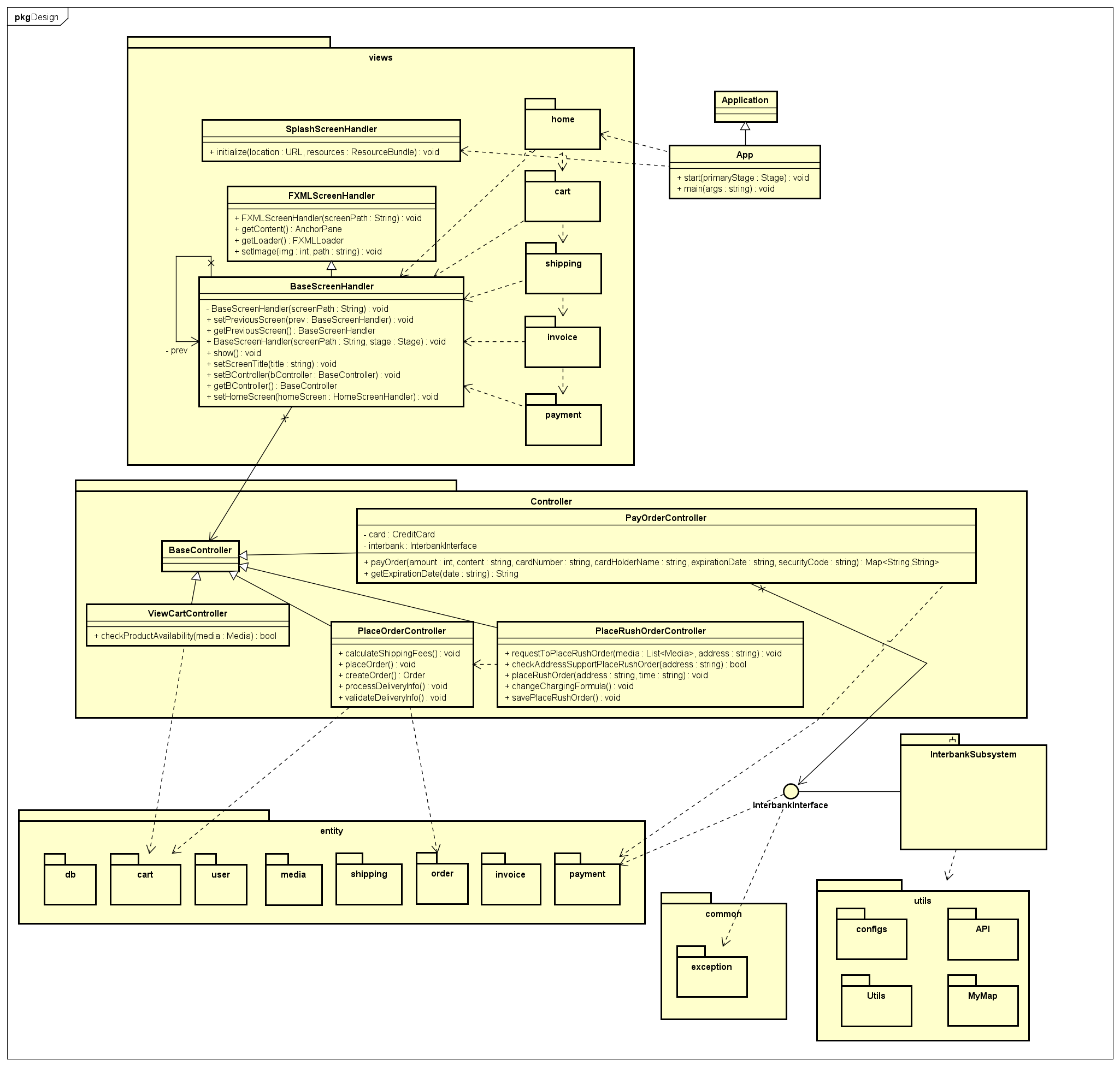
Description automatically generated with medium confidence

Bài tập cá nhân:

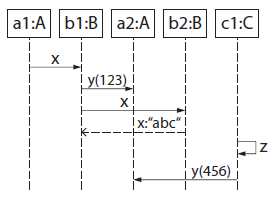
**Câu 1.** Bài tập AIMS Project:

**Hãy thiết kế lớp chi tiết cho Use case “Place Rush Order”.**

Trả lời:



**Câu 2.** Cho biểu đồ tuần tự (sequence diagram) như hình dưới đây, theo nội dung các thông điệp trên biểu đồ hãy vẽ biểu đồ cho 3 lớp A, B, C. Mỗi lớp phải thực hiện các phương thức nào?



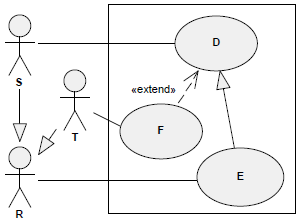
Trả lời:

A: Y

B: X

C: Z

**Câu 3.** Cho biểu đồ ca sử dụng dưới đây, tác nhân S có thể tương tác với usecase nào?



Trả lời: tác nhân S có thể tương tác với các use case D, E, F

**Câu 4.** Hãy giải thích mỗi hình sau đây ứng với loại kết hợp nào và ý nghĩa của chúng là gì.

Diagram

Description automatically generated

Trả lời:

i) Đây là quan hệ Aggregation. Hành khách là một phần của chuyến bay, chuyến bay chứa hành khách. Một hành khách chỉ đi được một chuyến bay, một chuyến bay có thể chở 0 hoặc nhiều hành khách. Chuyến bay bị hủy thì hành khách vẫn còn.

ii) Đây là quan hệ Composition. Section là một phần của Document, Document chứa và kiểm soát sự tồn tại của Section, nếu Document bị hủy thì Section cũng mất, nhưng Section bị mất thì Document ko bị mất theo. Một Section chỉ thuộc một Document, một Document có thể chứa 0 hoặc nhiều section.

iii) Đây là quan hệ Inheritance, Thư ký (lớp con)cũng là một nhân viên, kế thừa đầy đủ các thuộc tính và phương thức của Nhân viên (lớp cha)

**Câu 5.** Hãy mô tả thông tin trong mỗi sơ đồ sau.

Graphical user interface

Description automatically generated

Trả lời:

+ Sơ đồ 1: “Giao hàng” có 2 usecase kế thừa là ”Giao ở cửa hàng” và “Mang đến tận nhà”.

+ Sơ đồ 2: Khi thực hiện usecase“Bán hàng” bắt buộc phải thực hiện 2 usecase là “In hóa đơn” và “Xuất hàng”

+ Sơ đồ 3: Khi thực hiện usecase “Bán hàng”, thì đôi lúc sẽ có trường hợp cần thực hiện usecase “Nhập hàng từ nhà cung cấp” hoặc usecase “Hết hàng”. Khi thực hiện usecase “Hết hàng” thì có trường hợp cần thực hiện usecase “Nhập hàng từ nhà cung cấp”

**Câu 6.** Hãy giải thích các biểu đồ lớp sau đây?

Graphical user interface, application

Description automatically generated

Trả lời:

+ Biểu đồ 1: HangMua kế thừa Hang

HangMua là một phần của HoaDon, HoaDon chứa HangMua. Trong HoaDon có thể có từ 0 đến tối đa 20 HangMua, một HangMua thì chỉ thuộc một HoaDon

+ Biểu đồ 2: TaiKhoan có 1 No và 1 Co, kiểm soát sự tồn tại của No và Co. TaiKhoan mất thì No và Co mất

Bài tập nhóm:

Thực hiện Thiết kế lớp (Class Design) cho bài tập lớn môn học

* Các bước thực hiện:
  + 1. Xác định các phần tử thiết kế
  + 2. Thiết kế chi tiết cho từng lớp hoặc hệ thống con
  + 3. Thiết kế quan hệ giữa các lớp và xây dựng sơ đồ lớp
* Yêu cầu nộp bài:
  + Nộp bài vào thư mục 03-DetailedDesign trên thư mục Google Drive mà thầy đã tạo ([**https://bit.ly/3MMlJbT**](https://bit.ly/3MMlJbT))
  + Tổng hợp các kết quả thiết kế bước đầu ở trên viết tiếp vào tài liệu SRS
  + Các tệp tin Astah

**HẾT**