

## Assignment 07

### Class Design

#### 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.

##### a) 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.

#### b) 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.





























#### c) 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 trách 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).

#### d) Nhóm các lớp thiết kế

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

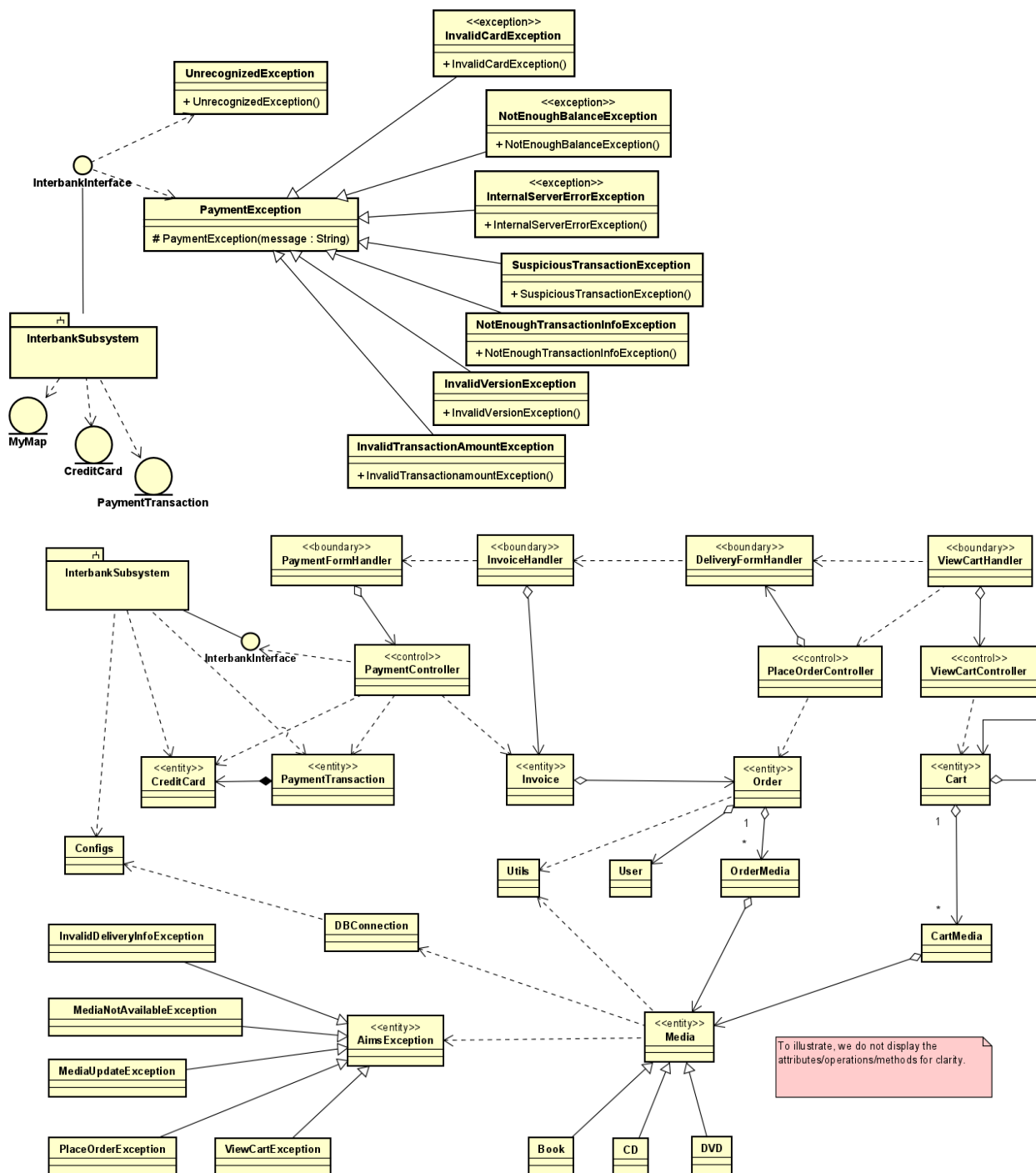
- ▼  > src
  - >  > (default package)
  - >  > common.exception
  - >  > controller
  - >  > entity.cart
  - >  > entity.db
  - >  > entity.media
  - >  > entity.order
  - >  > entity.payment
  - >  > entity.shipping
  - >  > entity.user
  - >  > subsystem
  - >  > subsystem.interbank
  - >  > utils
  - >  > views.fxml
  - >  > views.screen
  - >  > views.screen.cart
  - >  > views.screen.home
  - >  > views.screen.invoice
  - >  > views.screen.payment
  - >  > views.screen.shipping
- ▼  test
  - >  subsystem.interbank
  - >  utils

- ▼  > src
  - ▼  > (default package)
    - >  App.java
  - ▼  > common.exception
    - >  > InternalServerErrorException.java
    - >  > InvalidCardException.java
    - >  > InvalidDeliveryInfoException.java
    - >  > InvalidTransactionAmountException.java
    - >  > InvalidVersionException.java
    - >  > MediaNotAvailableException.java
    - >  > MediaUpdateException.java
    - >  > NotEnoughBalanceException.java
    - >  > NotEnoughTransactionInfoException.java
    - >  > PaymentException.java
    - >  > PlaceOrderException.java
    - >  > SuspiciousTransactionException.java
    - >  > UnrecognizedException.java
    - >  > ViewCartException.java
  - ▼  > controller
    - >  BaseController.java
    - >  > PaymentController.java
    - >  > PlaceOrderController.java
    - >  > ViewCartController.java
  - ▼  > entity.cart
    - >  > Cart.java
    - >  > CartMedia.java
  - ▼  > entity.db
    - >  > AIMSDB.java

- > entity.media
      - > Book.java
      - > CD.java
      - > DVD.java
      - > Media.java
    - entity.order
      - > Order.java
      - > OrderMedia.java
    - entity.payment
      - > CreditCard.java
      - > PaymentTransaction.java
    - entity.shipping
      - > Shipment.java
    - entity.user
      - > User.java
    - > subsystem
      - > InterbankInterface.java
      - > InterbankSubsystem.java
    - > subsystem.interbank
      - > InterbankBoundary.java
      - > InterbankSubsystemController.java
    - > utils
      - > API.java
      - > Configs.java
      - > MyMap.java
      - > Utils.java
  - views.fxml
    - cart.fxml
    - home.fxml
    - invoice\_rush\_order\_included.fxml
    - invoice\_rush\_order\_only.fxml
    - invoice.fxml
    - media\_cart.fxml
    - media\_home.fxml
    - media\_invoice.fxml
    - payment.fxml
    - result.fxml
    - rush\_order.fxml
    - shipping.fxml
    - > splash.fxml
  - views.screen
    - > BaseScreenHandler.java
    - > FXMLScreenHandler.java
    - > SplashScreenHandler.java
  - views.screen.cart
    - > CartScreenHandler.java
    - > MediaHandler.java
  - views.screen.home
    - > HomeScreenHandler.java
    - > MediaHandler.java
  - views.screen.invoice
    - > InvoiceScreenHandler.java
  - views.screen.payment
    - > PaymentScreenHandler.java
    - > ResultScreenHandler.java
  - views.screen.shipping
    - > ShippingScreenHandler.java

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

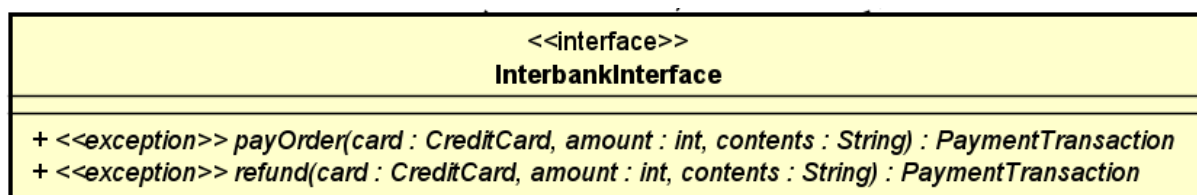
Để phục vụ mục đích minh họa mối quan hệ giữa các lớp được dễ nhìn và rõ ràng, các lớp ở các hình sau không bao gồm các attribute cũng như operation/method.



### 5.3.1.3. Lớp thiết kế

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

#### a) Lớp “InterbankInterface”



**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.

```

1 package subsystem;
2
3 import common.exception.PaymentException;
4 import common.exception.UnrecognizedException;
5 import entity.payment.CreditCard;
6 import entity.payment.PaymentTransaction;
7
8 /**
9  * The {@code InterbankInterface} class is used to communicate with the
10  * {@link subsystem.InterbankSubsystem InterbankSubsystem} to make transaction
11  *
12  * @author hieud
13  *
14  */
15 public interface InterbankInterface {
16
17     /**
18      * Pay order, and then return the payment transaction
19      *
20      * @param card - the credit card used for payment
21      * @param amount - the amount to pay
22      * @param contents - the transaction contents
23      * @return {@link entity.payment.PaymentTransaction PaymentTransaction} - if the
24      *         payment is successful
25      * @throws PaymentException if responded with a pre-defined error code
26      * @throws UnrecognizedException - if responded with an unknown error code or
27      *         something goes wrong
28      */
29     public abstract PaymentTransaction payOrder(CreditCard card, int amount, String contents)
30         throws PaymentException, UnrecognizedException;
31
32     /**
33      * Refund, and then return the payment transaction
34      *
35      * @param card - the credit card which would be refunded to
36      * @param amount - the amount to refund
37      * @param contents - the transaction contents
38      * @return {@link entity.payment.PaymentTransaction PaymentTransaction} - if the
39      *         payment is successful
40      * @throws PaymentException if responded with a pre-defined error code
41      * @throws UnrecognizedException - if responded with an unknown error code or
42      *         something goes wrong
43      */
44     public abstract PaymentTransaction refund(CreditCard card, int amount, String contents)
45         throws PaymentException, UnrecognizedException;
46
47 }
48

```

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>

## b) Lớp “PaymentController”

<<control>> PaymentController	
- card : CreditCard	
- interbank : InterbankInterface	
+ payOrder(amount : Int, contents : String, cardNumber : String, cardHolderName : String, expirationDate : String, securityCode : String) : Map<String,String>	
- getExpirationDate(date : String) : String	

**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

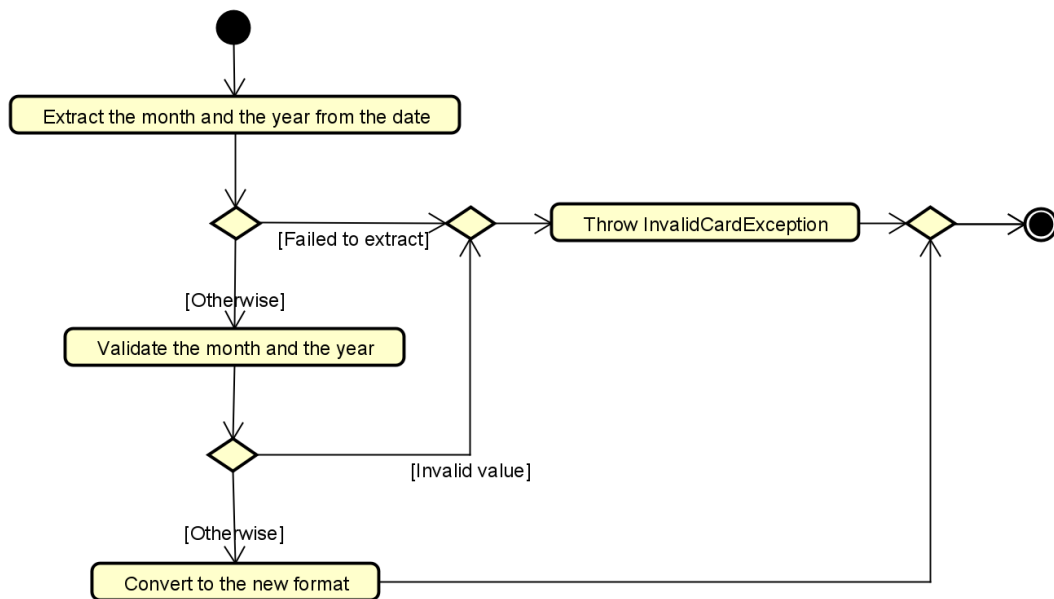
**Exception:**

- Không

**Method**

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





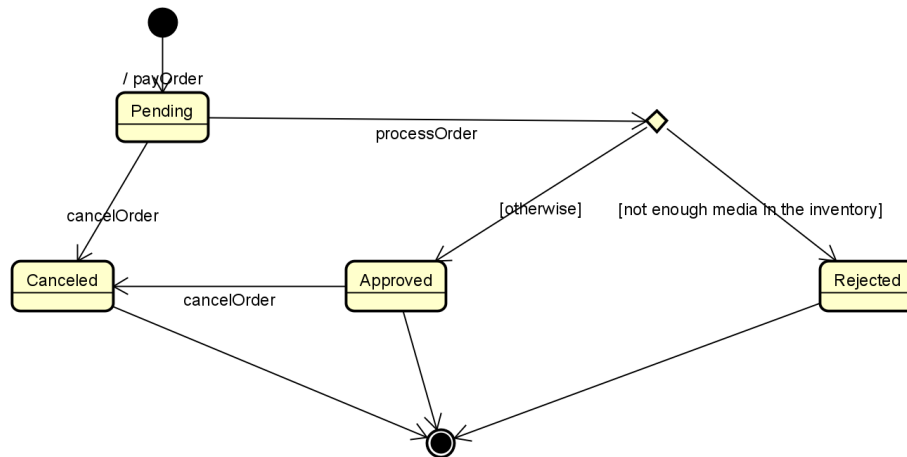
## State

Không

### c) 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.

Để 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.



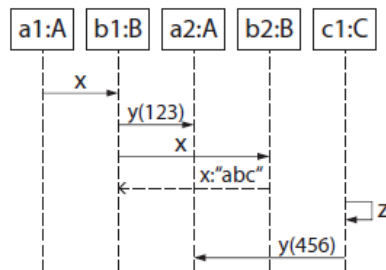
#### 5.3.1.4. Biểu đồ lớp thiết kế

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.



## Bài tập cá nhân:

Câu 1. 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?



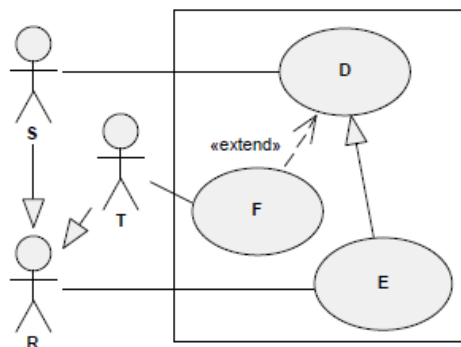
Trả lời:

A : y(123), y(456)

B: x , x:"abc" ,

C: z

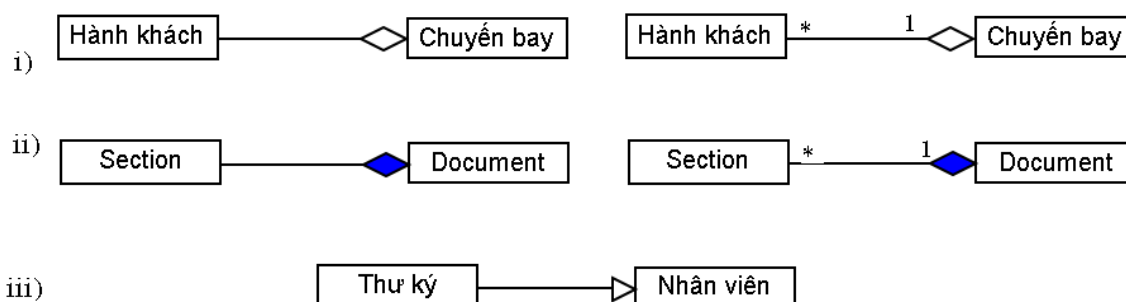
Câu 2. Cho biểu đồ ca sử dụng dưới đây, tác nhân S có thể tương tác với usecase nào?



Trả lời:

Tác nhân S có thể tương tác với D E F

Câu 3. Hãy giải thích mỗi hình sau đây ứng với loại kết hợp nào và ý nghĩa của chúng là gì.



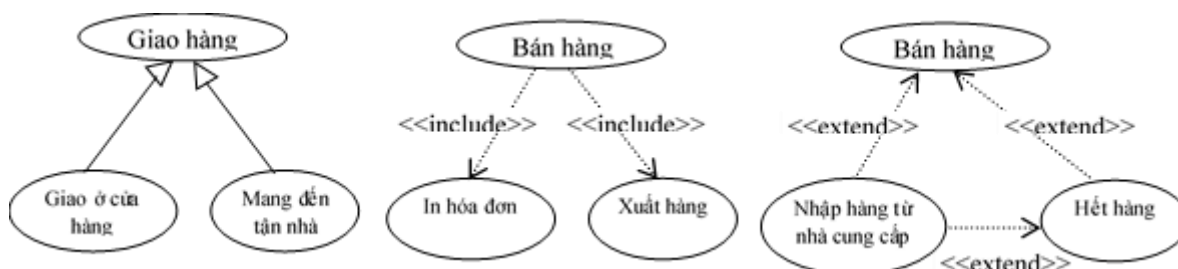
Trả lời:

i, Loại kết hợp là kết tập, khách hàng là một phần của chuyến bay, nhiều khách hàng có thể ở trong 1 chuyến bay

ii, Loại kết hợp là hợp thành, section là thành phần của document, nhiều section có thể nằm trong 1 document

iii, Loại kết hợp là tổng quát hóa, thư ký cũng là một nhân viên

Câu 4. Hãy mô tả thông tin trong mỗi sơ đồ sau.



Trả lời:

hình 1: UC giao hàng ở cửa hàng và UC mang đến tận nhà kế thừa UC giao hàng

hình 2: Khi thực hiện UC bán hàng thì bắt buộc phải in hóa đơn và xuất hàng

hình 3: Khi thực hiện UC bán hàng có thể hết hàng hoặc nhập hàng từ nhà cung cấp, nếu hết hàng sẽ có thể nhập hàng từ nhà cung cấp.

Câu 5. Hãy giải thích các biểu đồ lớp sau đây?



Trả lời:

HangMua kế thừa Hang và HangMua sẽ là 1 phần trong HoaDon với bội số  $0..n$  với 1

TaiKhoan sẽ luôn tồn tại No và Co, quan hệ là: hợp thành

## 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://drive.google.com/drive/folders/18or3uP5H3xr69Pkuidj\\_5eLQo\\_\\_X1hew?usp=sharing](https://drive.google.com/drive/folders/18or3uP5H3xr69Pkuidj_5eLQo__X1hew?usp=sharing))
  - Tổng hợp các kết quả thiết kế bước đầu ở trên viết tiếp vào tài liệu SRS
  - Các tệp tin Astah

---

# HẾT