# Báp cáo bài thực hành số 03

Họ và tên: Dương Hồng Tuấn

MSSV: 2073439

# 5. BÀI THỰC HÀNH SỐ 03 – THIẾT KẾ LỚP VÀ MÔ HÌNH HÓA DỮ LIỆU

## 5.1. MỤC ĐÍCH VÀ NỘI DUNG

Trong bài thực hành này, chúng ta sẽ làm quen với 02 bước còn lại trong thiết kế chi tiết (detailed design process), gồm có thiết kế lớp (class design) và mô hình hóa dữ liệu (data modeling). Kết thúc bài thực hành, người học có thể đạt được đủ kỹ nâng để hoàn thiện Software Design Document (SDD).

## 5.2. CHUẨN BỊ

Người học cần tự hoàn thiện trước buổi học: bước thiết kế kiến trúc của phần mềm (Architectural Design) và thiết kế giao diện của bước đầu thiết kế chi tiết. Kết quả của bước thiết kế kiến trúc phần mềm sẽ là đầu vào cho bước thiết kế chi tiết. Đồng thời kết quả của thiết kế giao diện sẽ cần thiết để hoàn thiện thiết kế chi tiết.

## 5.3. NỘI DUNG CHI TIẾT

### 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

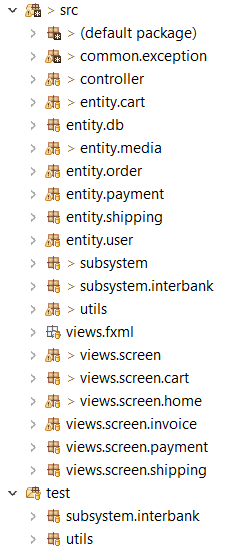
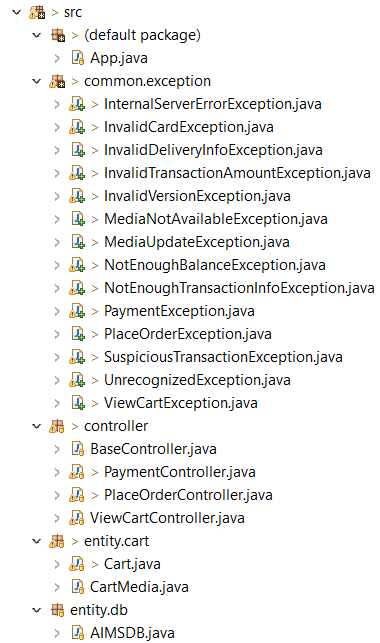
Với bản mô tả hiện tại và đối với 2 use case “Pay Order” và “Place Order”, đa số các lớp entity trong thiết kế kiến trúc đã đơn giản, và có thể ánh xạ 1-1 với lớp thiết kế.

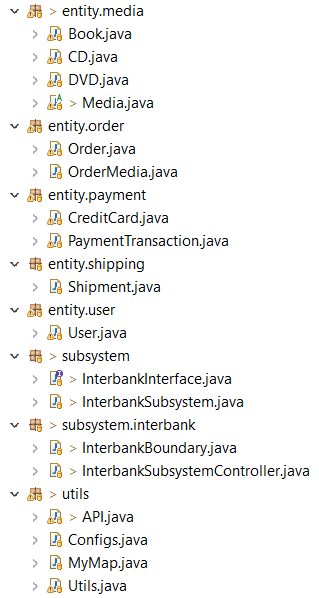
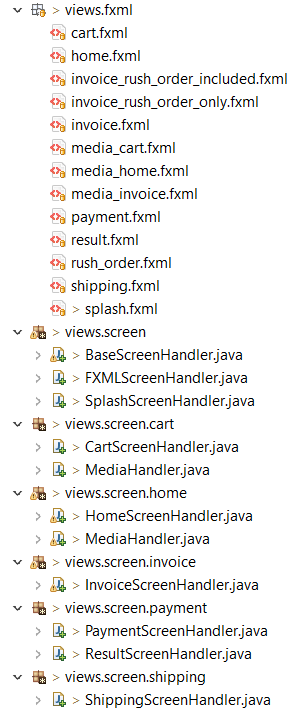
##### Thiết kế lớp control

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

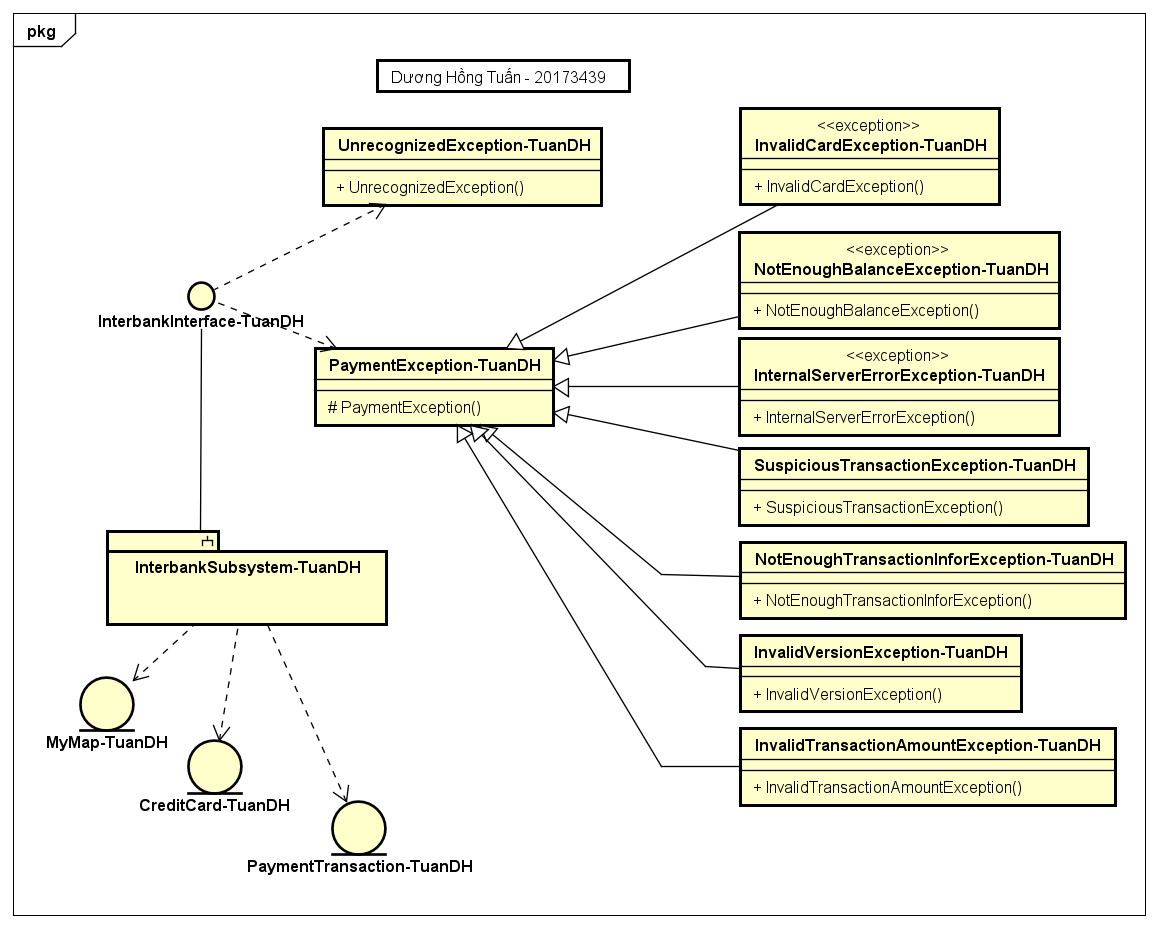
Sau đây là một cách để nhóm các lớp thiết kế vào các package.

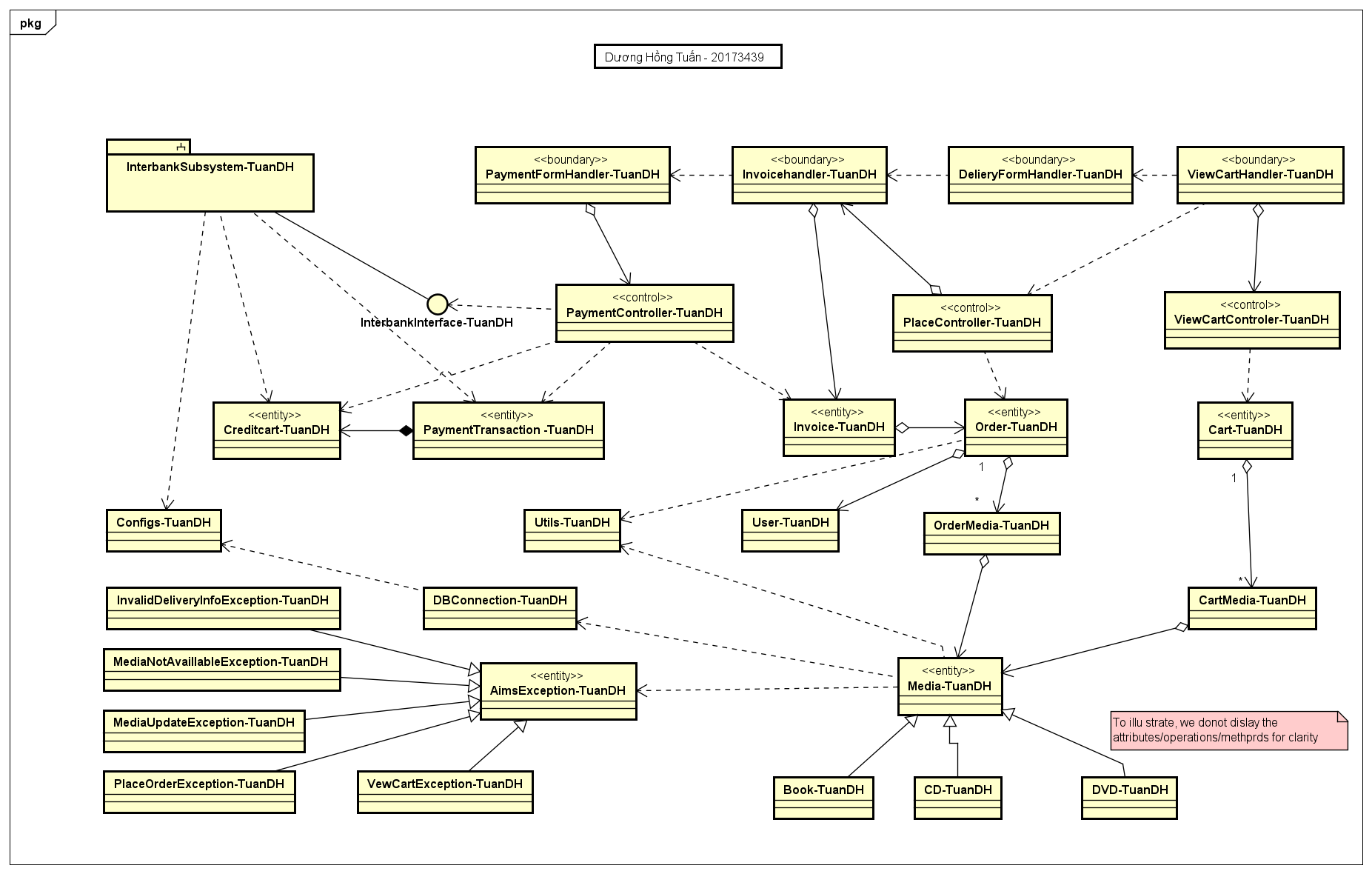


#### 5.3.1.2. Xác định mối quan hệ giữa các lớp

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.

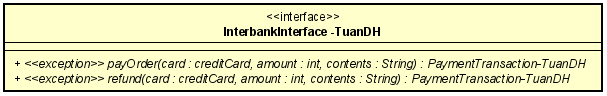




#### 5.3.1.3. Lớp thiết kế

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

##### 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

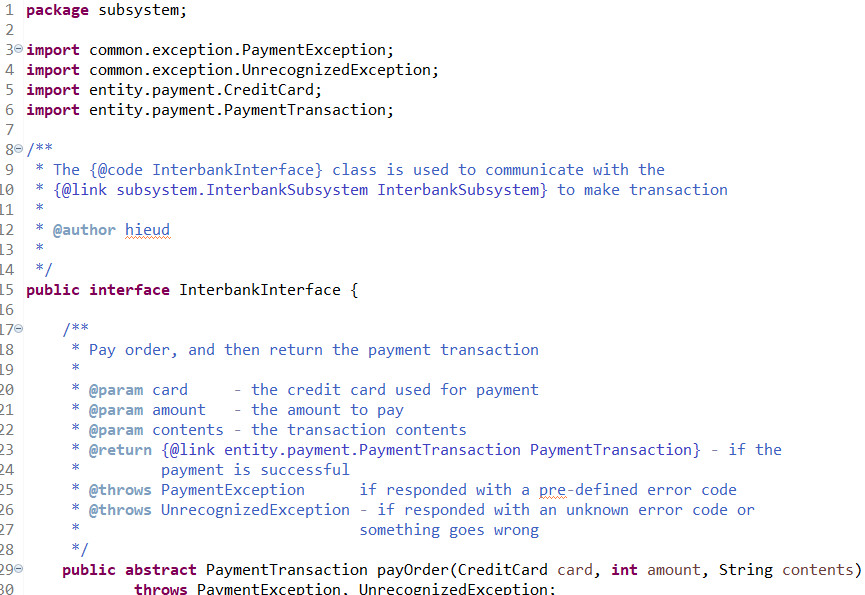
***Method***

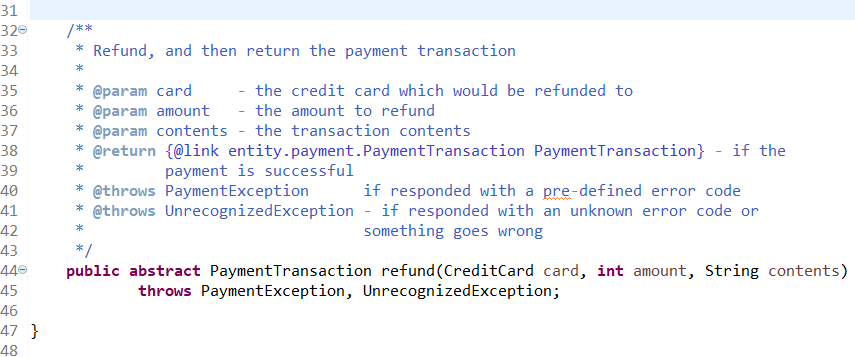
Không

***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.



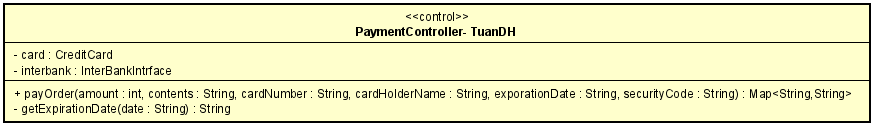


Một số đường dẫn tham khảo:

<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”



***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:*

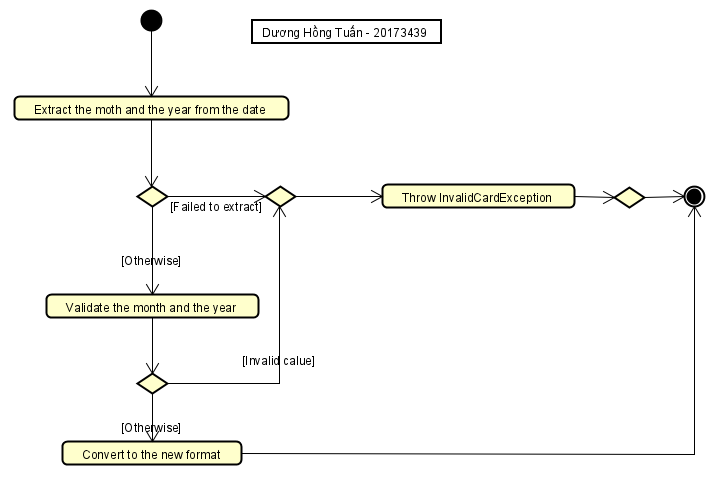
* amount – số tiền giao dịch
* contents – nội dung giao dịch
* cardNumber – số thẻ
* cardHolderName – tên chủ sở hữu
* expirationDate – ngày hết hạn theo định dạng "mm/yy"
* securityCode - mã bảo mật cvv/cvc

*Exception:*

* Không

***Method***

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



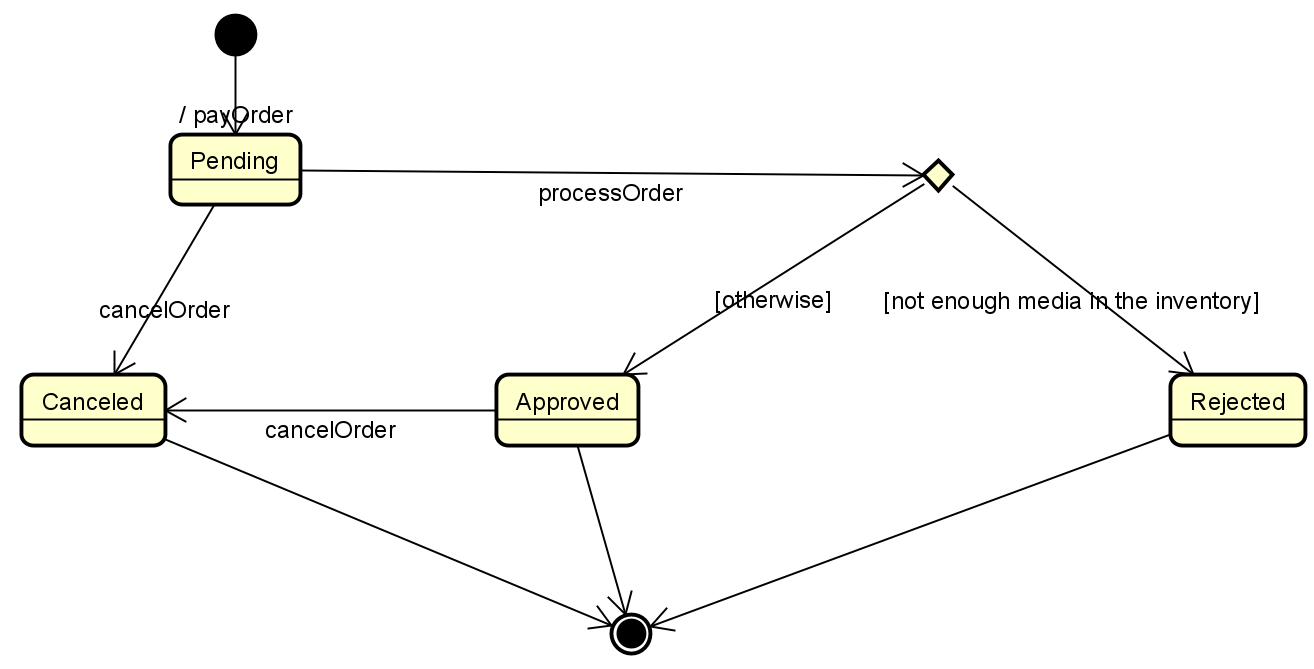
***State***

Không

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

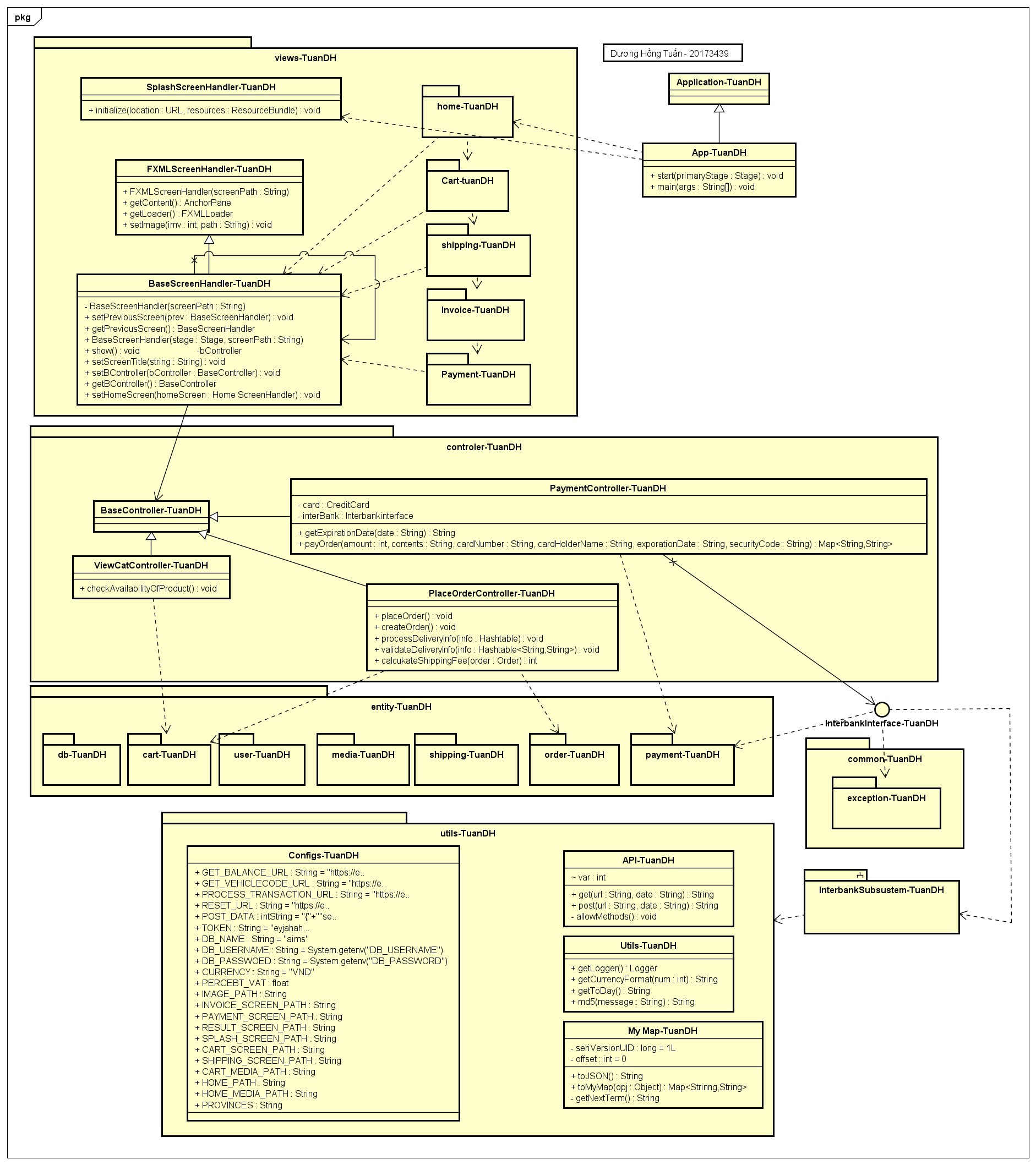
Chúng ta chỉ cần xác định các trạng thái được đặt tên (named states) của đối tượng, từ đó nắm được các trạng thái có thể có, và sự chuyển đổi giữa các trạng thái đó khi có các sự kiện tác động. Trước tiên, chúng ta tìm những lớp có các trạng thái được đặt tên. Sau đó, tìm các trạng thái có thể cho một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



#### 5.3.1.4. Biều đồ lớp thiết kế

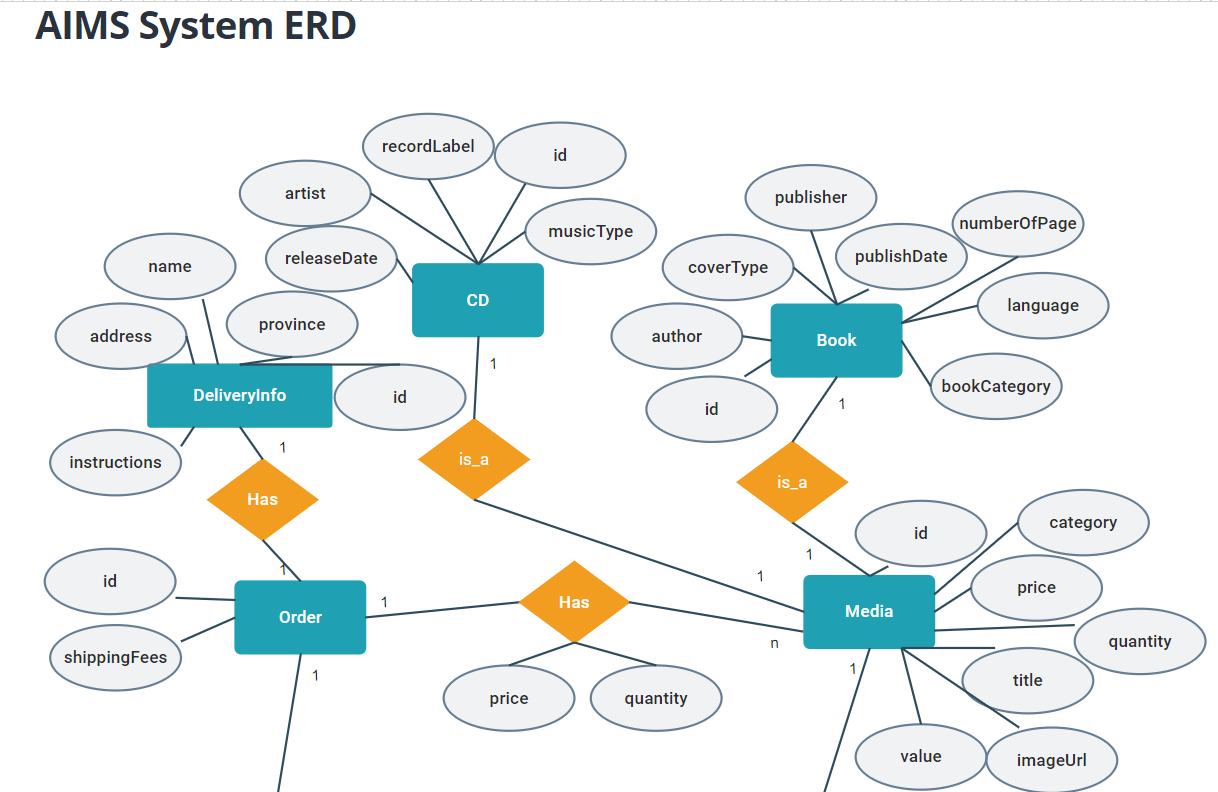
Cuối cùng, đặt tất cả các lớp thiết kế vào một biểu đồ lớp thiết kế tổng quan. Lưu ý không thể hiện chi tiết của subsystem trong biểu đồ này. Ngoài ra, nếu có quá nhiều chi tiết trong biểu đồ, có thể làm tương tự với các package và tạo biểu đồ lớp thiết kế cho từng package.

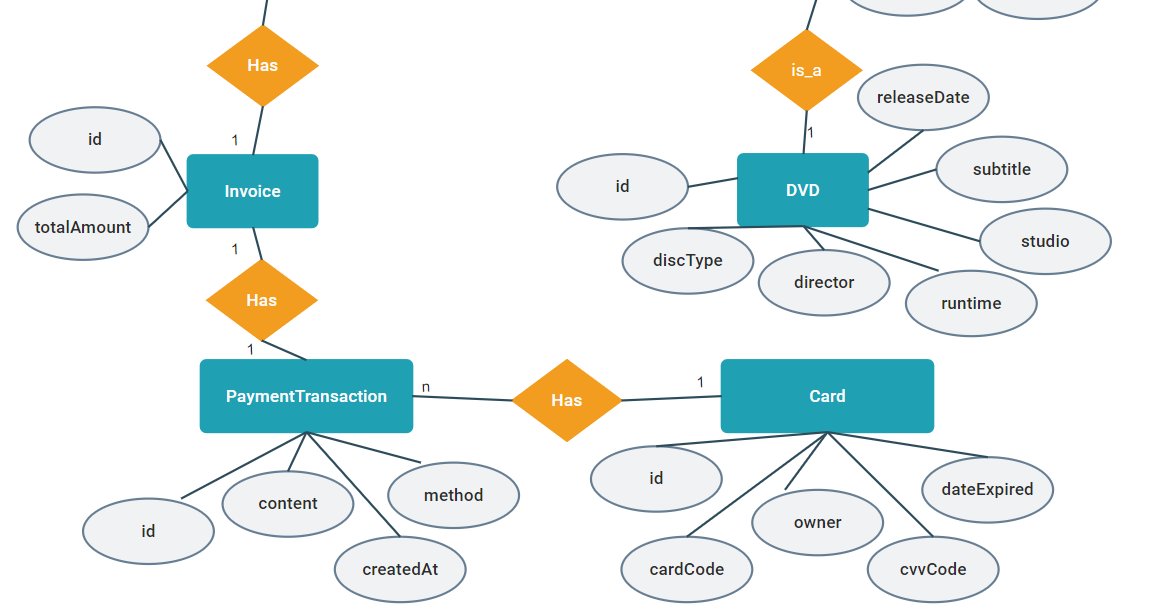


### 5.3.2. Mô hình hóa dữ liệu (Data Modeling)

#### 5.3.2.1. Mô hình khái niệm (Conceptual Data Model)

Mô hình khái niệm là một mô hình dữ liệu có tính trừu tượng cao, trừu tượng hóa các biểu hiện tự nhiên (natural expression) mà không phụ thuộc vào bất kỳ hệ thống quản trị dữ liệu (database management system – DBMS) nào như PostgreSQL, SQLite, Microsoft Access, hoặc MongoDB. Một mô hình khái niệm có thể được thể hiện bởi biểu đồ thực thể-liên kết (Entity-Relationship diagram – ER diagram). Sau đây là một ví dụ cho AIMS Software.





#### 5.3.2.2. Thiết kế cơ sở dữ liệu

Trong phần này, ta cần phải xác định sự lựa chọn hệ thống quản trị dữ liệu (DBMS) và mô tả DBMS.

Ví dụ, ta sẽ chọn SQLite 3.7.2 là DBMS cho Case Study vì:

- SQLite là một DBMS mã nguồn mở.

- SQLite không chỉ nhẹ, nhanh, phổ biến, không cần cài đặt, có độ tin cậy cao, đầy đủ tính năng, và là một engine cơ sở dữ liệu SQL; ngoài ra còn ổn định, không phụ thuộc vào nền tảng, tương thích ngược và hỗ trợ lâu dài.

- Có thể kết nối với Java ( <https://github.com/xerial/sqlite-jdbc> )

Tuy nhiên, SQLite không thể đảm đương trách nhiệm của một cơ sở dữ liệu ở máy chủ trong thực tế (hệ thống thương mại điện tử) như MySQL hay PostgreSQL vì SQLite lưu dữ liệu của người dùng tại chính thiết bị.

Chúng ta vẫn sử dụng SQLite cho Case Study bởi project mẫu sẽ chỉ chạy local mà vẫn có thể minh họa đầy đủ cho người học.

##### Mô hình dữ liệu logic (Logical data model)

Từ mô hình khái niệm (được thể hiện bởi biểu đồ ER) trong phần trước, ta có thể thiết kế mô hình dữ liệu logic tương thích với DBMS đã chọn (SQLite). Sau đây là mô hình dữ liệu logic ví dụ.

Diagram

Description automatically generated

##### Mô hình dữ liệu vật lý (Physical data model)

Trong phần này, chúng ta sẽ thiết kế chi tiết cho từng phần tử trong biểu đồ trên. Ví dụ, trong biểu đồ cơ sở dữ liệu quan hệ (RDBMS), ta thiết kế chi tiết cho dừng bảng và ràng buộc.

Chú thích:

PK: Primary Key

FK: Foreign Key

* **Media**

| # | PK | FK | Column Name | Data type | Mandatory | Description |
| --- | --- | --- | --- | --- | --- | --- |
|  | x |  | id | Integer | Yes | ID, auto increment |
|  |  |  | category | VARCHAR(45) | Yes | Media type, e.g., CD, DVD |
|  |  |  | price | Integer | Yes | Current price |
|  |  |  | quantity | Integer | Yes | Number of products |
|  |  |  | title | VARCHAR(45) | Yes | Product name |
|  |  |  | value | Integer | Yes | Value of the product |
|  |  |  | imageUrl | VARCHAR(45) | Yes | Product image path |

* **CD**

| # | PK | FK | Column Name | Data type | Mandatory | Description |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | x | id | Integer | Yes | ID, same as ID of Media of which type is CD |
|  |  |  | artist | VARCHAR(45) | Yes | Artist's name |
|  |  |  | recordLabel | VARCHAR(45) | Yes | Record label |
|  |  |  | musicType | VARCHAR(45) | Yes | Music genres |
|  |  |  | releasedDate | DATE | No | Release date |

* **Book**

| # | PK | FK | Column Name | Data type | Mandatory | Description |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | x | id | Integer | Yes | ID, same as ID of Media of which type is Book |
|  |  |  | author | VARCHAR(45) | Yes | Author |
|  |  |  | coverType | VARCHAR(45) | Yes | Cover type |
|  |  |  | Publisher | VARCHAR(45) | Yes | Publishing house |
|  |  |  | publishDate | DATETIME | Yes | Date of publishing |
|  |  |  | numOfPages | Integer | Yes | Page number |
|  |  |  | language | VARCHAR(45) | Yes | Language |
|  |  |  | bookCategory | VARCHAR(45) | Yes | Book category |

* **DVD**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data type | Mandatory | Description |
|  |  | x | id | Integer | Yes | ID, same as ID of Media of which type is DVD |
|  |  |  | discType | VARCHAR(45) | Yes | Disc type |
|  |  |  | director | VARCHAR(45) | Yes | Director |
|  |  |  | runtime | Integer | Yes | Duration |
|  |  |  | studio | VARCHAR(45) | Yes | Manufacturer |
|  |  |  | subtitle | VARCHAR(45) | Yes | Subtitles |
|  |  |  | releasedDate | DATETIME | Yes | Release date |
|  |  |  | filmType | VARCHAR(45) | Yes | Genres |

* **Card**

| # | PK | FK | Column Name | Data type | Mandatory | Description |
| --- | --- | --- | --- | --- | --- | --- |
|  | x |  | id | Integer | Yes | ID, auto increment |
|  |  |  | cardCode | VARCHAR(45) | Yes | Card code |
|  |  |  | owner | VARCHAR(45) | Yes | Cardholders |
|  |  |  | cvvCode | VARCHAR(3) | Yes | CVV code |
|  |  |  | dateExpired | VARCHAR(4) | Yes | Expiration date |

* **DeliveryInfo**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data type | Mandatory | Description |
|  | X |  | id | Integer | Yes | ID, auto increment |
|  |  |  | name | VARCHAR(45) | Yes | Receiver name |
|  |  |  | province | VARCHAR(45) | Yes | Provinces |
|  |  |  | instructions | VARCHAR(200) | No | Delivery instructions |
|  |  |  | address | VARCHAR(100) | Yes | Delivery address |

* **Order**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data type | Mandatory | Description |
|  | X |  | id | Integer | Yes | ID |
|  |  |  | shippingFees | VARCHAR(45) | Yes | Shipping fee |
|  |  | X | deliveryInfoId | Integer | Yes | Delivery Info ID |

* **OrderMedia**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data type | Mandatory | Description |
|  |  | X | mediaID | Integer | Yes | Media ID |
|  |  | X | orderID | Integer | Yes | Order ID |
|  |  |  | price | Integer | Yes | Selling price |
|  |  |  | quantity | Integer | Yes | Number |

* **Invoice**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data type | Mandatory | Description |
|  | x |  | id | Integer | Yes | ID |
|  |  |  | totalAmount | Integer | Yes | Total |
|  |  | x | orderId | Integer | Yes | Order ID |

* **PaymentTransaction**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data type | Mandatory | Description |
|  | x |  | id | Integer | Yes | ID |
|  |  |  | createAt | DATETIME | Yes | Date of creation |
|  |  |  | content | VARCHAR(45) | Yes | Transaction contents |
|  |  |  | method | VARCHAR(45) | Yes | Payment methods |
|  |  | x | cardId | Integer | Yes | ID of used card |
|  |  | x | invoiceId | Integer | Yes | Invoice ID |

Cuối cùng, ta cần có database script. Với những công cụ thiết kế database chuyên nghiệp và các plugins, ta có thể tự động tạo database script trực tiếp từ mô hình dữ liệu logic.

BEGIN;

CREATE TABLE "aims"."Media"(

"id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

"category" VARCHAR(45) NOT NULL,

"price" INTEGER NOT NULL,

"quantity" INTEGER NOT NULL,

"title" VARCHAR(45) NOT NULL,

"value" INTEGER NOT NULL,

"imageUrl" VARCHAR(45) NOT NULL

);

CREATE TABLE "aims"."CD"(

"id" INTEGER PRIMARY KEY NOT NULL,

"artist" VARCHAR(45) NOT NULL,

"recordLabel" VARCHAR(45) NOT NULL,

"musicType" VARCHAR(45) NOT NULL,

"releasedDate" DATE,

CONSTRAINT "fk\_CD\_Media1"

FOREIGN KEY("id")

REFERENCES "Media"("id")

);

CREATE TABLE "aims"."Book"(

"id" INTEGER PRIMARY KEY NOT NULL,

"author" VARCHAR(45) NOT NULL,

"coverType" VARCHAR(45) NOT NULL,

"publisher" VARCHAR(45) NOT NULL,

"publishDate" DATETIME NOT NULL,

"numOfPages" INTEGER NOT NULL,

"language" VARCHAR(45) NOT NULL,

"bookCategory" VARCHAR(45) NOT NULL,

CONSTRAINT "fk\_Book\_Media1"

FOREIGN KEY("id")

REFERENCES "Media"("id")

);

CREATE TABLE "aims"."DeleveryInfo"(

"id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

"name" VARCHAR(45),

"province" VARCHAR(45),

"instructions" VARCHAR(200),

"address" VARCHAR(100)

);

CREATE TABLE "aims"."Card"(

"id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

"cardCode" VARCHAR(15) NOT NULL,

"owner" VARCHAR(45) NOT NULL,

"cvvCode" VARCHAR(3) NOT NULL,

"dateExpired" VARCHAR(4) NOT NULL

);

CREATE TABLE "aims"."DVD"(

"id" INTEGER PRIMARY KEY NOT NULL,

"discType" VARCHAR(45) NOT NULL,

"director" VARCHAR(45) NOT NULL,

"runtime" INTEGER NOT NULL,

"studio" VARCHAR(45) NOT NULL,

"subtitle" VARCHAR(45) NOT NULL,

"releasedDate" DATETIME,

CONSTRAINT "fk\_DVD\_Media1"

FOREIGN KEY("id")

REFERENCES "Media"("id")

);

CREATE TABLE "aims"."Order"(

"id" INTEGER NOT NULL,

"shippingFees" VARCHAR(45),

"deleveryInfoId" INTEGER NOT NULL,

PRIMARY KEY("id","deleveryInfoId"),

CONSTRAINT "fk\_Order\_DeleveryInfo1"

FOREIGN KEY("deleveryInfoId")

REFERENCES "DeleveryInfo"("id")

);

CREATE INDEX "aims"."Order.fk\_Order\_DeleveryInfo1\_idx" ON "Order" ("deleveryInfoId");

CREATE TABLE "aims"."OrderMedia"(

"orderID" INTEGER NOT NULL,

"price" INTEGER NOT NULL,

"quantity" INTEGER NOT NULL,

"mediaId" INTEGER NOT NULL,

PRIMARY KEY("orderID","mediaId"),

CONSTRAINT "fk\_ordermedia\_order"

FOREIGN KEY("orderID")

REFERENCES "Order"("id"),

CONSTRAINT "fk\_OrderMedia\_Media1"

FOREIGN KEY("mediaId")

REFERENCES "Media"("id")

);

CREATE INDEX "aims"."OrderMedia.fk\_ordermedia\_order\_idx" ON "OrderMedia" ("orderID");

CREATE INDEX "aims"."OrderMedia.fk\_OrderMedia\_Media1\_idx" ON "OrderMedia" ("mediaId");

CREATE TABLE "aims"."Invoice"(

"id" INTEGER PRIMARY KEY NOT NULL,

"totalAmount" INTEGER NOT NULL,

"orderId" INTEGER NOT NULL,

CONSTRAINT "fk\_Invoice\_Order1"

FOREIGN KEY("orderId")

REFERENCES "Order"("id")

);

CREATE INDEX "aims"."Invoice.fk\_Invoice\_Order1\_idx" ON "Invoice" ("orderId");

CREATE TABLE "aims"."PaymentTransaction"(

"id" INTEGER NOT NULL,

"createAt" DATETIME NOT NULL,

"content" VARCHAR(45) NOT NULL,

"method" VARCHAR(45),

"cardId" INTEGER NOT NULL,

"invoiceId" INTEGER NOT NULL,

PRIMARY KEY("id","cardId","invoiceId"),

CONSTRAINT "fk\_PaymentTransaction\_Card1"

FOREIGN KEY("cardId")

REFERENCES "Card"("id"),

CONSTRAINT "fk\_PaymentTransaction\_Invoice1"

FOREIGN KEY("invoiceId")

REFERENCES "Invoice"("id")

);

CREATE INDEX "aims"."PaymentTransaction.fk\_PaymentTransaction\_Card1\_idx" ON "PaymentTransaction" ("cardId");

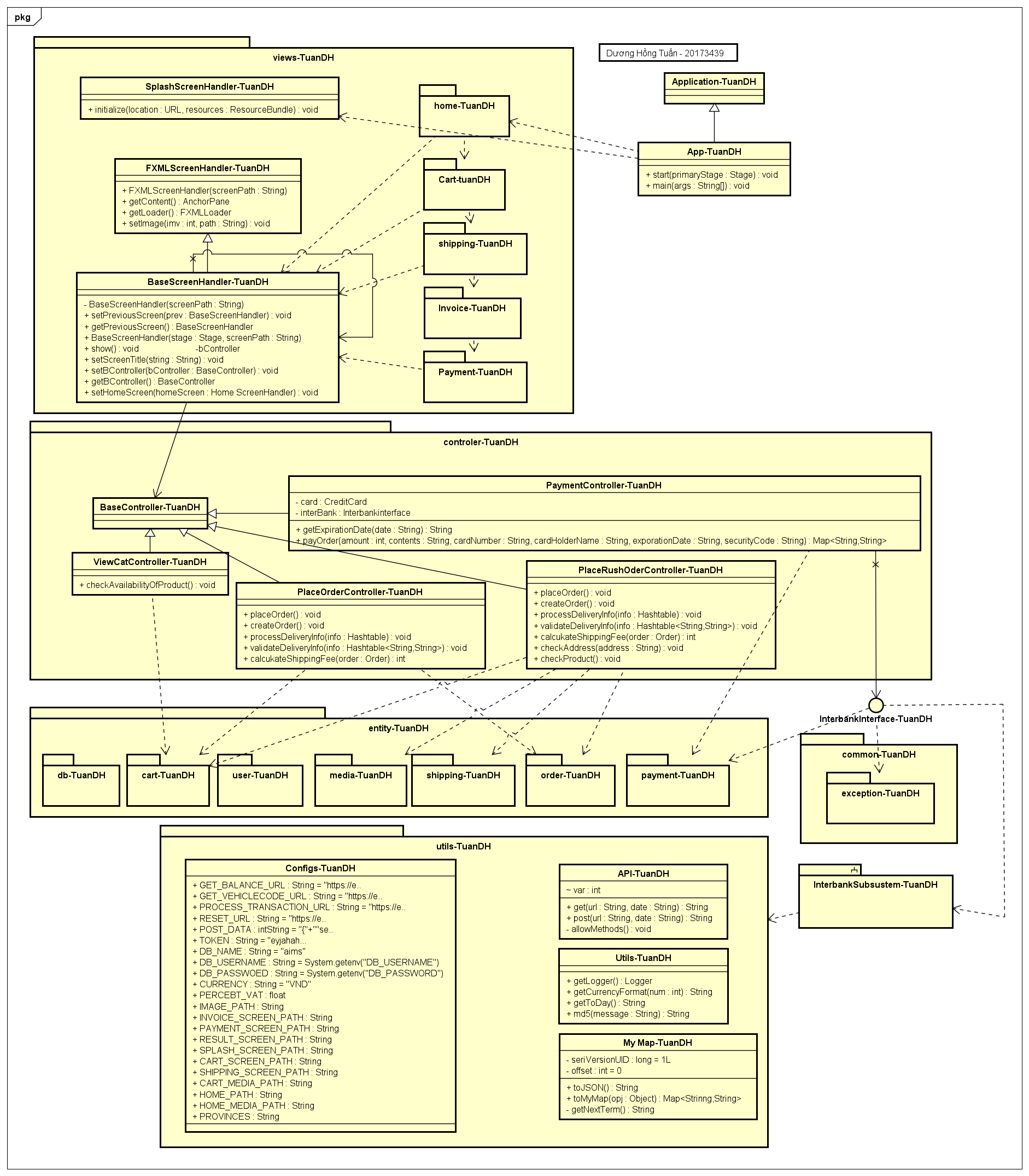
CREATE INDEX "aims"."PaymentTransaction.fk\_PaymentTransaction\_Invoice1\_idx" ON "PaymentTransaction" ("invoiceId");

COMMIT;

## BÀI TẬP

**Hãy thiết kế lớp chi tiết và mô hình hóa dữ liệu cho Use case “Place Rush Order”.**

**Lớp chi tiết:**

****

**Mô hình hóa dữ liệu**

* **Order**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | PK | FK | Column Name | Data type | Mandatory | Description |
| 1 | X |  | id | Integer | Yes | ID |
| 2 |  |  | shippingFees | VARCHAR(45) | Yes | Shipping fee |
| 3 |  | X | deliveryInfoId | Integer | Yes | Delivery Info ID |
| 4 |  | x | productId | Integer | Yes | Product id support rush order |
| 5 |  |  | adress | VARCHAR(200) | Yes |  |