Link github nộp bài: https://github.com/SuPerEle14/TKXDPM\_TranTheLam

Đánh giá điểm thực hành:

* Chuyên cần, tham gia đầy đủ, đúng giờ, nghiêm túc hoàn thành bài theo mẫu
* Hoàn thành bài tập theo yêu cầu đúng hạn.
* Làm báo cáo đầy đủ theo mẫu, link bài tập nộp lên Git/GitHub nộp đúng hạn: trước trước 23h59, thứ 5, tuần học.

Yêu cầu:

* Báo cáo trên assigment: (30 %)  
  (Chụp ảnh kết quả các câu dán vào file theo mẫu sau:)
* Nộp code bài lên link Git/GitHub. (30%)
* Điểm trắc nghiệm nếu có bài trắc nghiệm cuối mỗi buổi (30%)
* Điểm danh (10%). – assignment.

Bài Thực Hành Số 03 – Tuần 8

[Phần 5. BÀI THỰC HÀNH SỐ 03 – THIẾT KẾ LỚP VÀ MÔ HÌNH HÓA DỮ LIỆU 2](#_Toc88249500)

[5.1. MỤC ĐÍCH VÀ NỘI DUNG 2](#_Toc88249501)

[5.2. CHUẨN BỊ 2](#_Toc88249502)

[5.3. NỘI DUNG CHI TIẾT 2](#_Toc88249503)

[5.3.1. Thiết kế lớp (Class Design) 2](#_Toc88249504)

[5.3.2. Mô hình hóa dữ liệu (Data Modeling) 9](#_Toc88249505)

[5.4. BÀI TẬP 14](#_Toc88249506)

[5.4.1. Mỗi quan hệ giữa các lớp thêm PlaceRushOrder 15](#_Toc88249507)

[5.4.2. Lớp Thiết kế 15](#_Toc88249508)

[5.4.3. Mô hình dữ liệu vật lý (Physical data model) 16](#_Toc88249509)

**Mục lục**

# Phần 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

##### 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

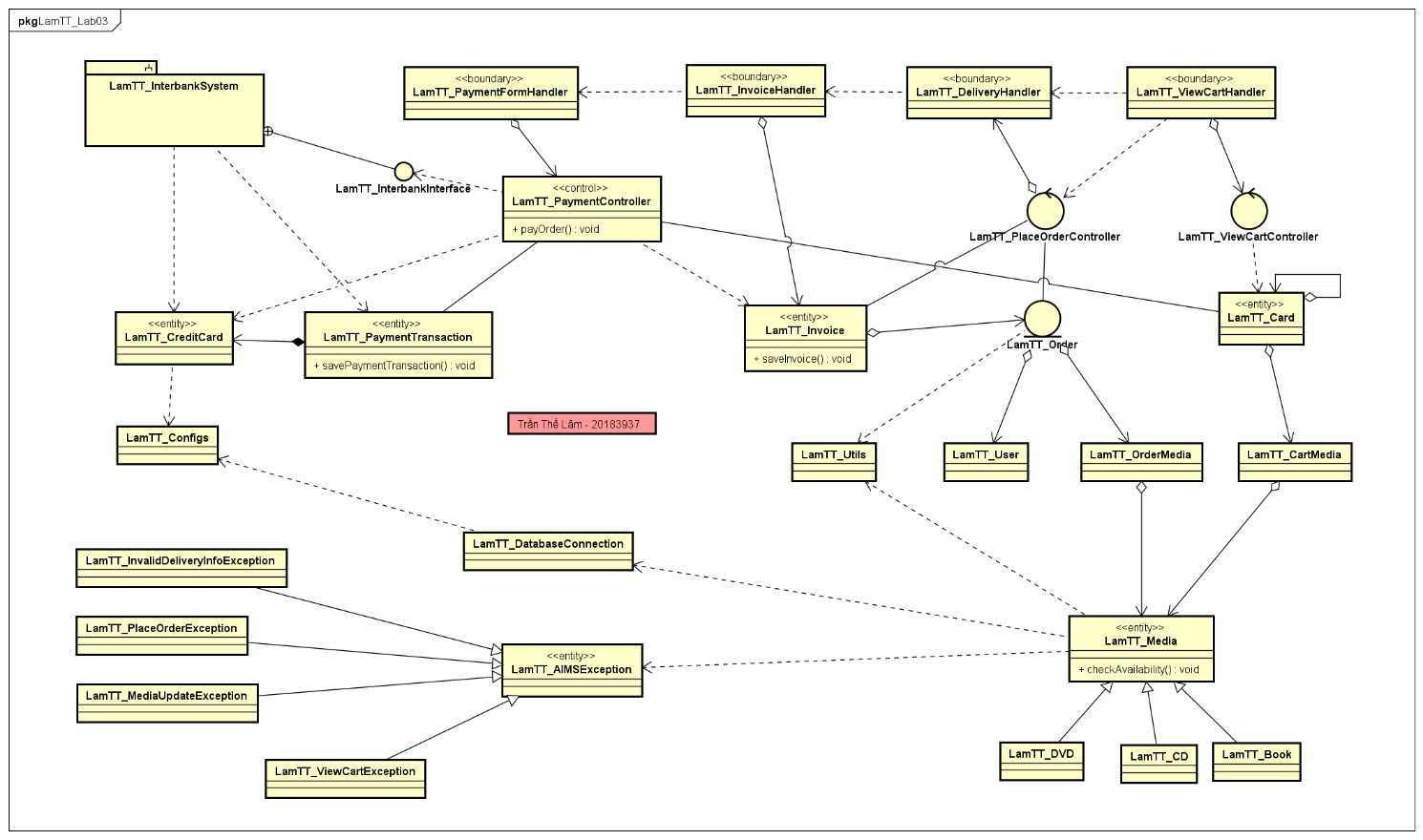
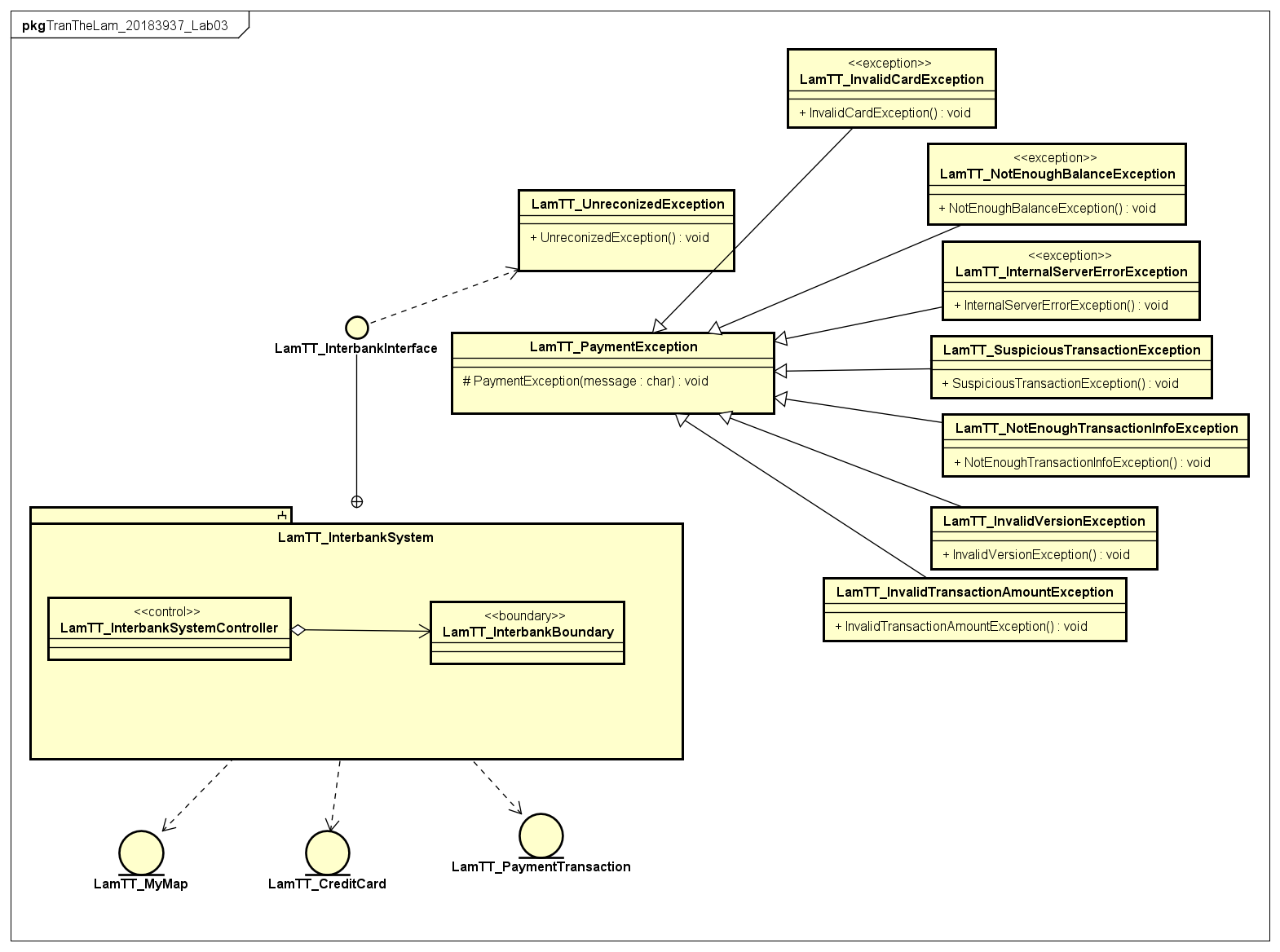
Với bản mô tả hiện tại và đối với 2 use case “Pay Order” và “Place Order”, đa số các lớp entity trong thiết kế kiến trúc đã đơn giản, và có thể ánh xạ 1-1 với lớp thiết kế.

##### 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)

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

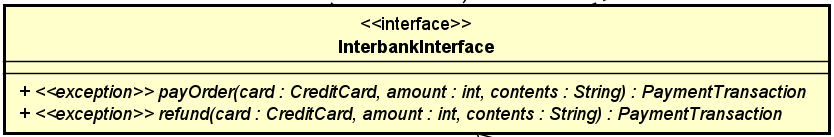
Để phục vụ mục đích minh họa mối quan hệ giữa các lớp được dễ nhìn và rõ ràng, các lớp ở các hình sau không bao gồm các attribute cũng như operation/method.



#### 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

##### Lớp “PaymentController”

Ảnh có chứa văn bản

Mô tả được tạo tự động

***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:*

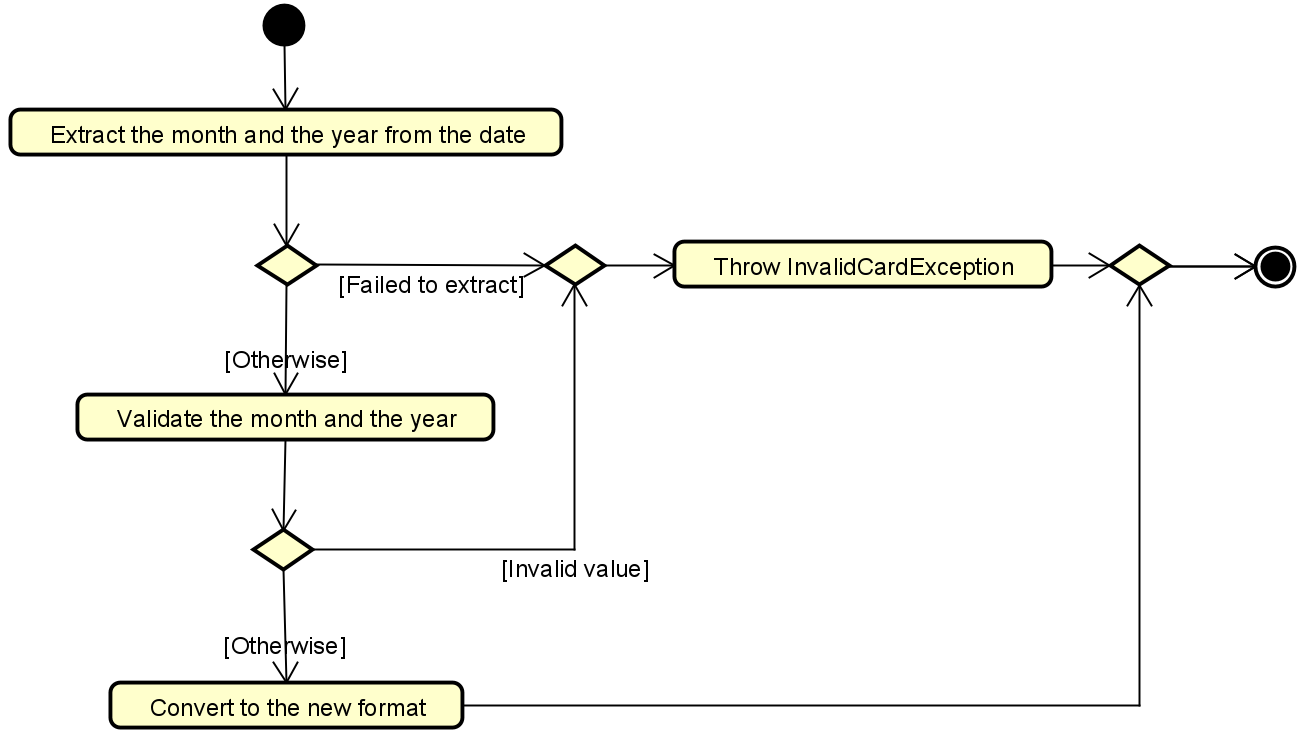
* amount – số tiền giao dịch
* contents – nội dung giao dịch
* cardNumber – số thẻ
* cardHolderName – tên chủ sở hữu
* expirationDate – ngày hết hạn theo định dạng "mm/yy"
* securityCode - mã bảo mật cvv/cvc

*Exception:*

* Không

***Method***

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

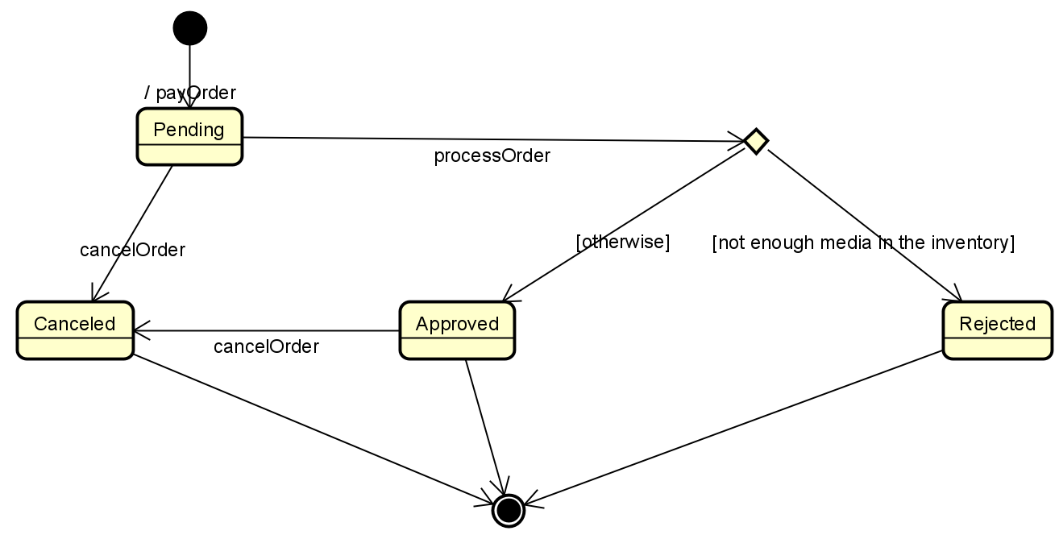


***State***

Không

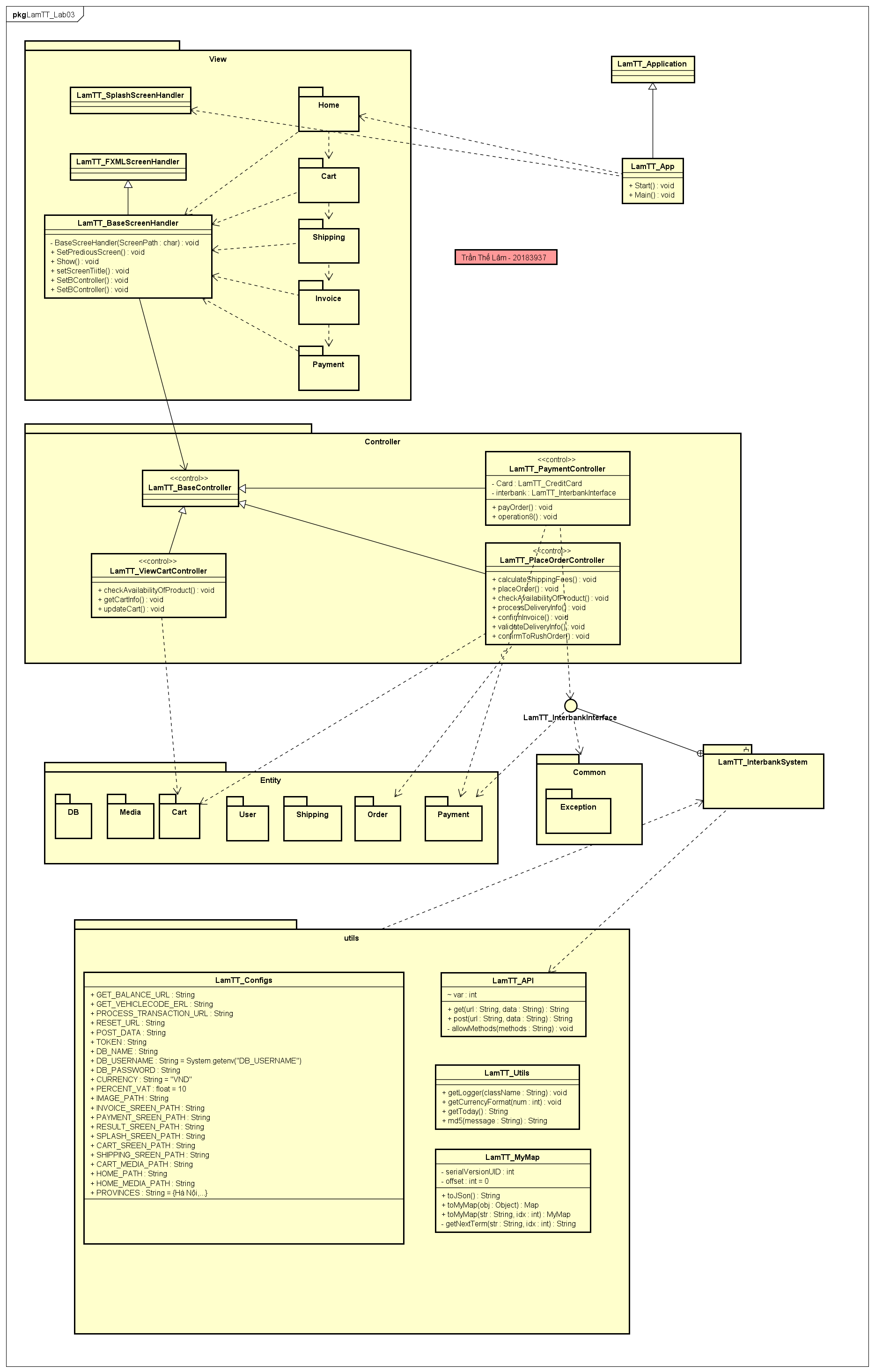
##### Biểu đồ trạng thái (state machine) cho đối tượng “Order”

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.



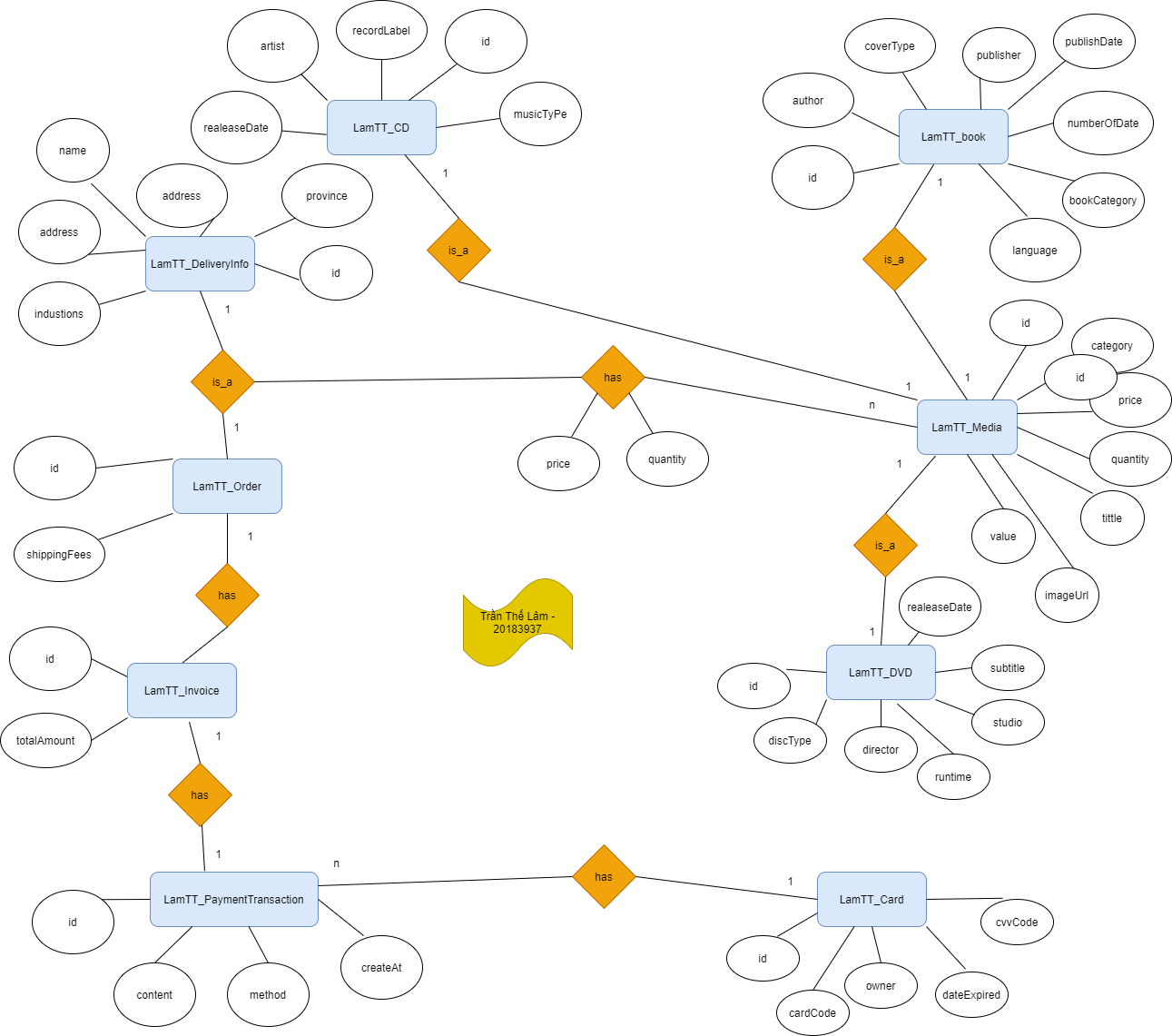
#### 5.3.1.4. Biều đồ lớp thiết kế

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)



#### 5.3.2.2. Thiết kế cơ sở dữ liệu

Em chọn MySQL là DBMS cho Case Study vì:

- MySQL là một DBMS mã nguồn mở.

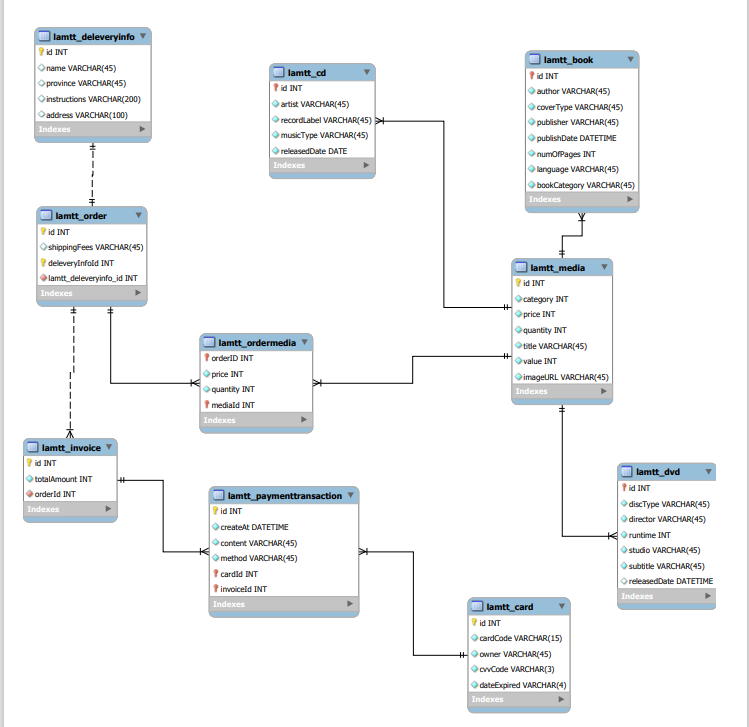
- MySQL không chỉ nhẹ, nhanh, phổ biến, có độ tin cậy cao, đầy đủ tính năng, và là một engine cơ sở dữ liệu SQL; ngoài ra còn ổn định, tương thích ngược và hỗ trợ lâu dài

- Có thể kết nối với Java

- MySQL có thể đảm đương trách nhiệm của một cơ sở dữ liệu ở máy chủ trong thực tế (hệ thống thương mại điện tử).

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

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



##### 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 | 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 |

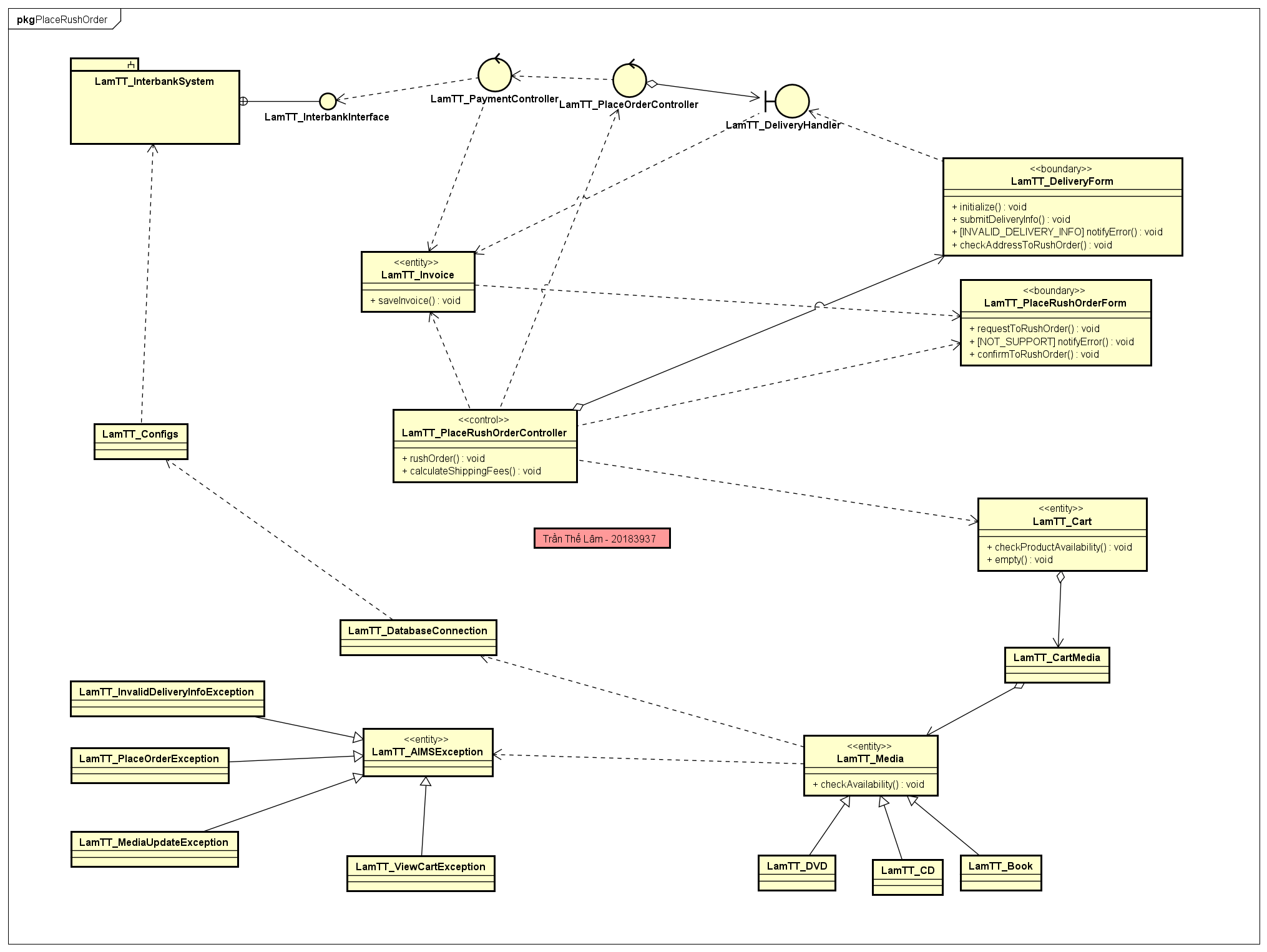
* **PaymentTransaction**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 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 |

## BÀI TẬP

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

### 5.4.1. Mỗi quan hệ giữa các lớp thêm PlaceRushOrder



### 5.4.2. Lớp Thiết kế

Ảnh có chứa văn bản

Mô tả được tạo tự động

***Attribute***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu trả về* | *Giá trị mặc định* | *Mô tả* |
| 1 | timeDelivery | DateTime | Null | Thời gian giao hàng Đã chọn |

***Operation***

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu trả về* | *Mô tả (mục đích)* |
| 1 | rushOrder | initilize | Tạo biểu mẫu Rush Order |
| 2 | calculateShippingFees | int | Tính phí vận chuyển |
| 3 | validateDeliveryInfo | boolean | Xác nhận Hình thức giao hàng nhanh |

*Exception:*

* Không

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

Chú thích:

PK: Primary Key

FK: Foreign Key

* **LamTT\_PlaceRushOrderController**

| # | PK | FK | Column Name | Data type | Mandatory | Description |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | x |  | id | Integer | Yes | ID, auto increment |
| 2 |  | x | customerID | Integer | Yes | ID |
| 3 |  |  | price | Integer | Yes | Current price |
| 4 |  |  | quantity | Integer | Yes | Number of products |
| 5 |  |  | title | VARCHAR(45) | Yes | Product name |
| 6 |  |  | shippingFees | Integer | Yes | Ex: 20.000 VND |
| 7 |  |  | value | Integer | Yes | Value of the product |
| 8 |  |  | imageUrl | VARCHAR(45) | Yes | Product image path |
| 9 |  |  | validateRushOrder | BOOLEAN | Yes | Validate Rush Order: Yes - No |
| 10 |  |  | notes | VARCHAR(200) |  | Note to Store |