TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Tài liệu thiết kế phần mềm

(Software Design Document– SDD)

Phiên bản 1.0

HỆ THỐNG SOÁT VÉ / THẺ TỰ ĐỘNG

Môn: Thiết kế và xây dựng phần mềm

Giảng viên hướng dẫn: TS. Nguyễn Thị Thu Trang

Nhóm 04:

Trương Thành Duy - 20160790

Cao Văn Duy - 20160750

Phùng Việt Duy - 20160782

Nguyễn Ngọc Hải - 20141382

*Hà Nội, tháng 12 năm* *2019*

Mục lục

[Mục lục 1](#_Toc26691578)

[1 Giới thiệu 3](#_Toc26691579)

[1.1 Mục đích 3](#_Toc26691580)

[1.2 Phạm vi 3](#_Toc26691581)

[1.3 Từ điển thuật ngữ 3](#_Toc26691582)

[1.4 Tài liệu tham khảo 3](#_Toc26691583)

[2 Thiết kế cấu trúc tĩnh 4](#_Toc26691584)

[2.1 Biểu đồ gói 4](#_Toc26691585)

[2.2 Biểu đồ lớp 4](#_Toc26691586)

[2.2.1 Biểu đồ lớp tổng quan 4](#_Toc26691587)

[2.2.2 Biểu đồ lớp tổng quan “soát vé một chiều” 6](#_Toc26691588)

[2.2.3 Biểu đồ lớp tổng quan “soát vé 24h” 7](#_Toc26691589)

[2.2.4 Biểu đồ lớp tổng quan “Soát thẻ trả trước” 8](#_Toc26691590)

[2.2.5 Biểu đồ lớp chi tiết 9](#_Toc26691591)

[2.3 Thiết kế cơ sở dữ liệu 10](#_Toc26691592)

[2.3.1 Biểu đồ quan hệ 10](#_Toc26691593)

[2.3.2 Thiết kế cơ sở dữ liệu 10](#_Toc26691594)

[2.4 Thiết kế giao diện 12](#_Toc26691595)

[3 Thiết kế hành vi động 13](#_Toc26691596)

[3.1 Biểu đồ tuần tự use case UC001 “Check in bằng vé một chiều” 13](#_Toc26691597)

[3.2 Biểu đồ tuần tự use case UC002 “Check out bằng vé một chiều” 14](#_Toc26691598)

[3.3 Biểu đồ tuần tự use case UC003 “Check in bằng vé 24h” 15](#_Toc26691599)

[3.4 Biểu đồ tuần tự use case UC004 “Check out bằng vé 24h” 15](#_Toc26691600)

[3.5 Biểu đồ tuần tự use case UC005 “Check in bằng thẻ trả trước” 16](#_Toc26691601)

[3.6 Biểu đồ tuần tự use case UC006 “Check out bằng thẻ trả trước” 17](#_Toc26691602)

[4 Phân tích các mức Coupling và Cohesion 18](#_Toc26691603)

[5 Design principles trong SOLID và Design Pattern 19](#_Toc26691604)

[5.1 Design principles trong SOLID 19](#_Toc26691605)

[5.1.1 Single responsibility principle 19](#_Toc26691606)

[5.1.2 Open-closed principle 19](#_Toc26691607)

[5.1.3 Liskov substitution principle 19](#_Toc26691608)

[5.1.4 Interface segregation principle 20](#_Toc26691609)

[5.1.5 Dependency Inversion principle 20](#_Toc26691610)

[5.2 Design Pattern 20](#_Toc26691611)

[6 Ưu điểm của quá trình thay đổi 21](#_Toc26691612)

[6.1 Thêm các loại vé, thẻ mới 21](#_Toc26691613)

[6.2 Thay đổi cách tính giá vé 21](#_Toc26691614)

[6.2.1 Thay đổi ngưỡng, lượng tiền trong mỗi ngưỡng 21](#_Toc26691615)

[6.2.2 Thay đổi giá tiền theo khoảng cách, tính giá tiền theo vùng. 21](#_Toc26691616)

[6.3 Thay đổi tuyến đường 21](#_Toc26691617)

# Giới thiệu

## Mục đích

Tài liệu này đưa ra mô tả chi tiết cho Hệ thống soát vé tự động (Automated Fare Collection). Tài liệu mô tả mục đích và các tính năng của hệ thống, ràng buộc của hệ thống cần thực hiện để phản ứng tới các kích thích bên ngoài.

Tài liệu dành cho các bên liên quan (stakeholder) và các nhà phát triển phần mềm.

## Phạm vi

Trong phạm vi môn học, Hệ thống soát vé tự động được giả lập các hệ thống bên ngoài như Máy nhận diện vé, Máy quét thẻ và Cổng kiểm soát.

## Từ điển thuật ngữ

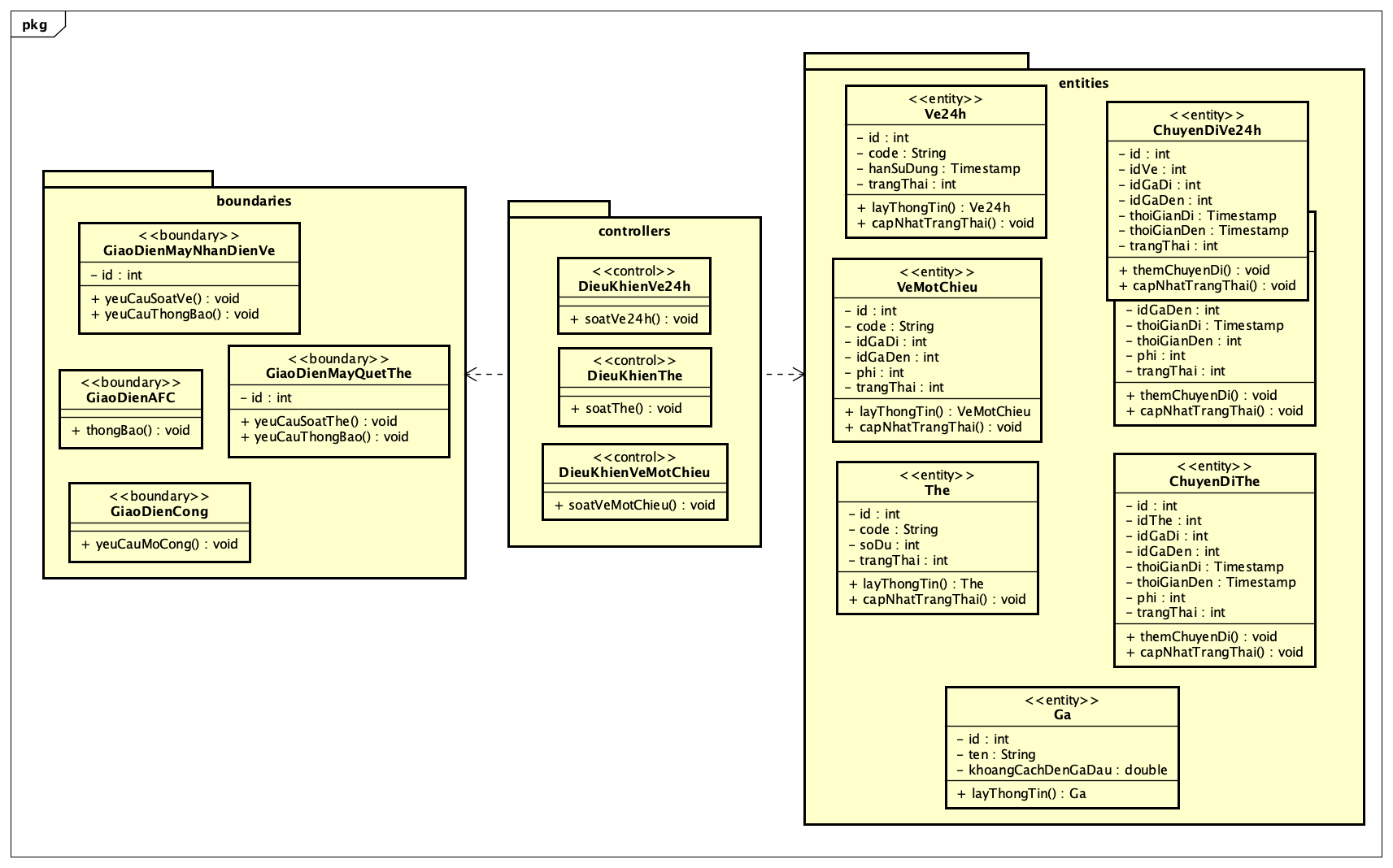
|  |  |  |
| --- | --- | --- |
| **STT** | **Từ** | **Nghĩa** |
|  | Automated fare collection (AFC) | Máy soát thẻ / vé tự động. |
|  | One-way ticket | Vé một chiều: Vé sử dụng để di chuyển một lần duy nhất. |
|  | Twenty-four-hour ticket | Vé 24 giờ (24h): Vé sử dụng để di chuyển không giới hạn trong vòng 24h. |
|  | Prepaid card | Thẻ (Thẻ trả trước): Thẻ sử dụng số dư để di chuyển. |
|  | Check in | Soát vé / thẻ vào. |
|  | Check out | Soát vé / thẻ ra. |
|  | One-way ticket Trip | Chuyến đi của vé 1 chiều. |
|  | Twenty-four-hour ticket Trip | Chuyến đi của vé 24 giờ. |
|  | Prepaid card Trip | Chuyến đi của thẻ trả trước. |
|  | Station | Ga. |
|  | Balance | Số dư tài khoản thẻ trả trước. |
|  | Fare | Giá cơ bản tính theo khoảng cách di chuyển. |
|  | BarCode | Mã vạch của vé/thẻ. |
|  | Distance | Khoảng cách giữa các ga. |

## Tài liệu tham khảo

* TS. Nguyễn Thị Thu Trang, SRS-UGMS-Sample-VN.
* TS. Nguyễn Thị Thu Trang, IT4490 Software Design and Construction slides.
* TS. Nguyễn Thị Thu Trang, Mini-project Topic v1.5.1.pdf

# Thiết kế cấu trúc tĩnh

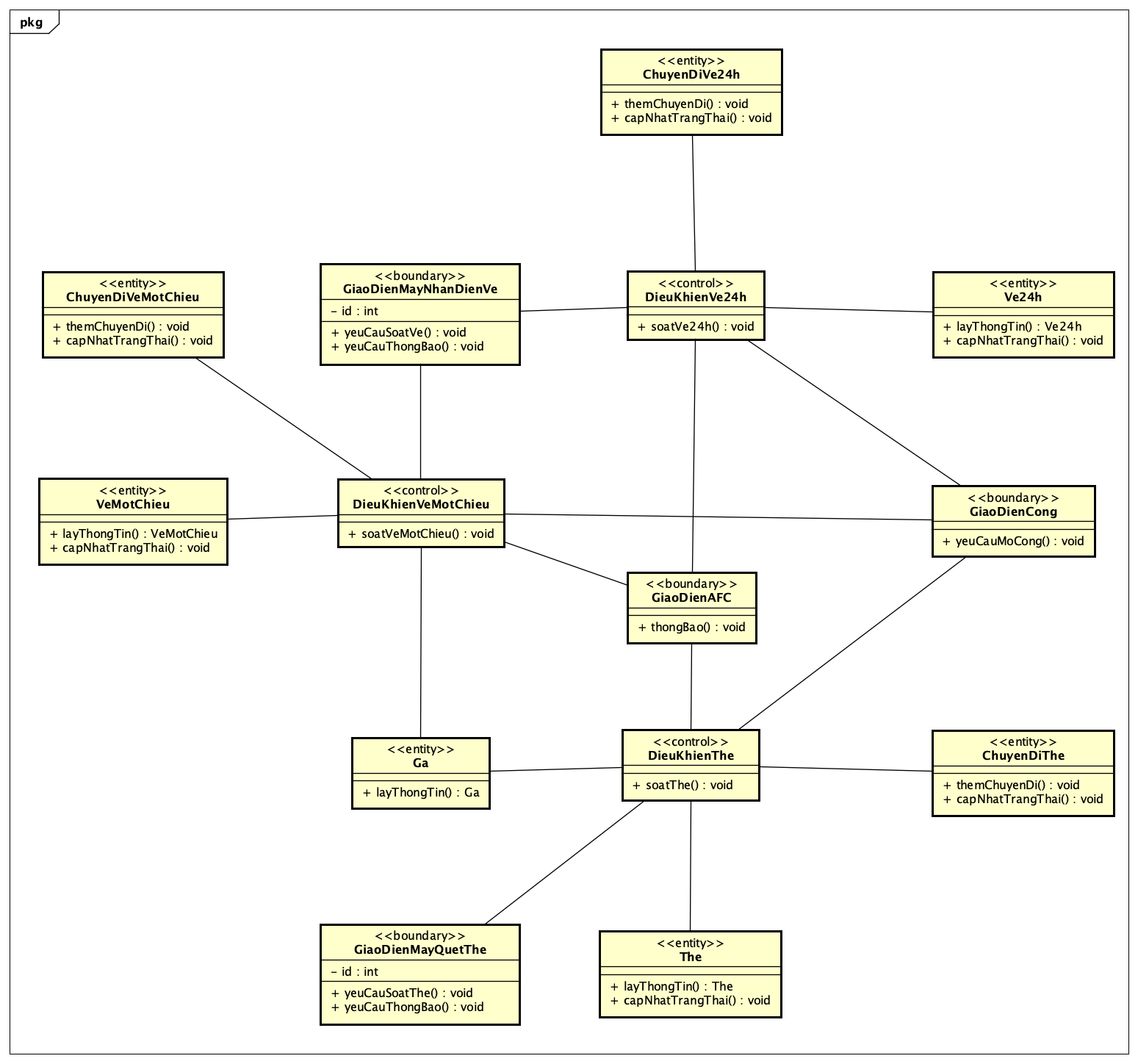
## Biểu đồ gói



Hình 2.1: Biểu đồ gói

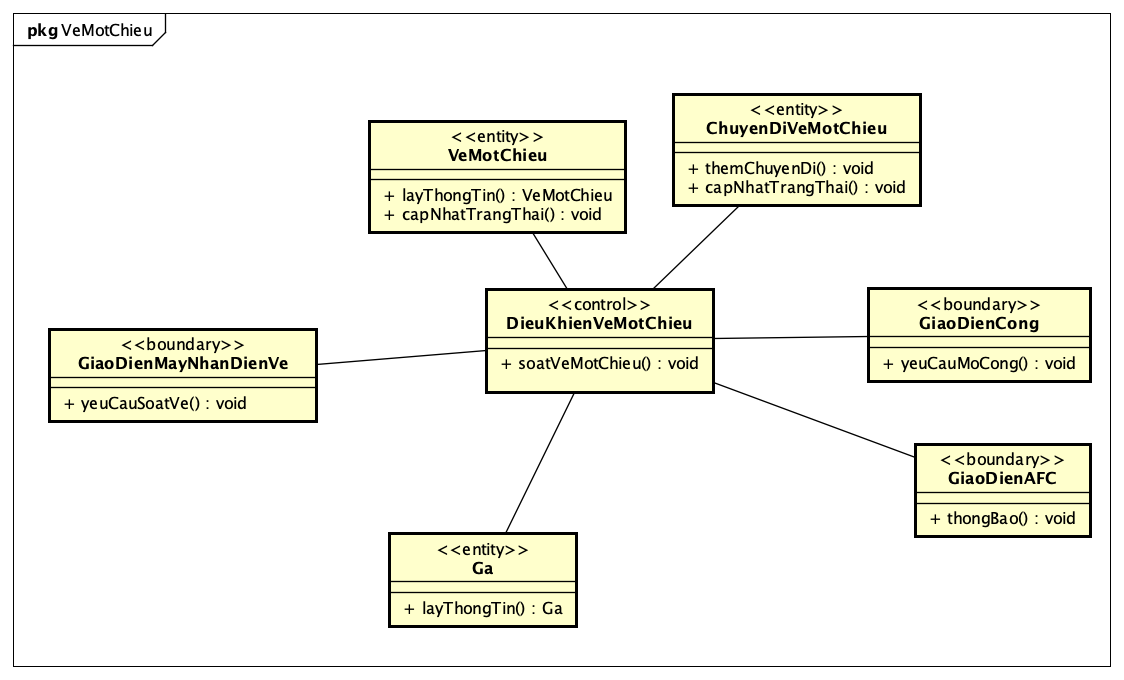
## Biểu đồ lớp

### Biểu đồ lớp tổng quan



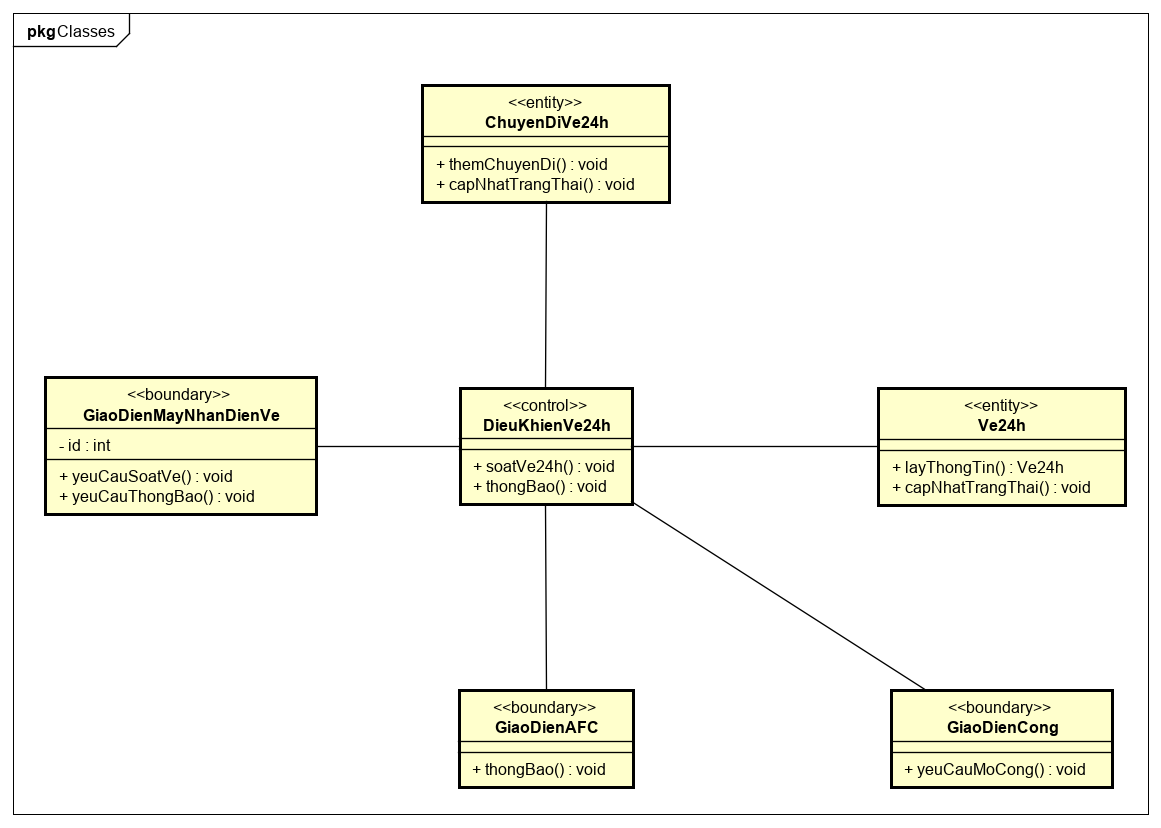
Hình 2.2.1: Biểu đồ lớp tổng quan

### Biểu đồ lớp tổng quan “soát vé một chiều”



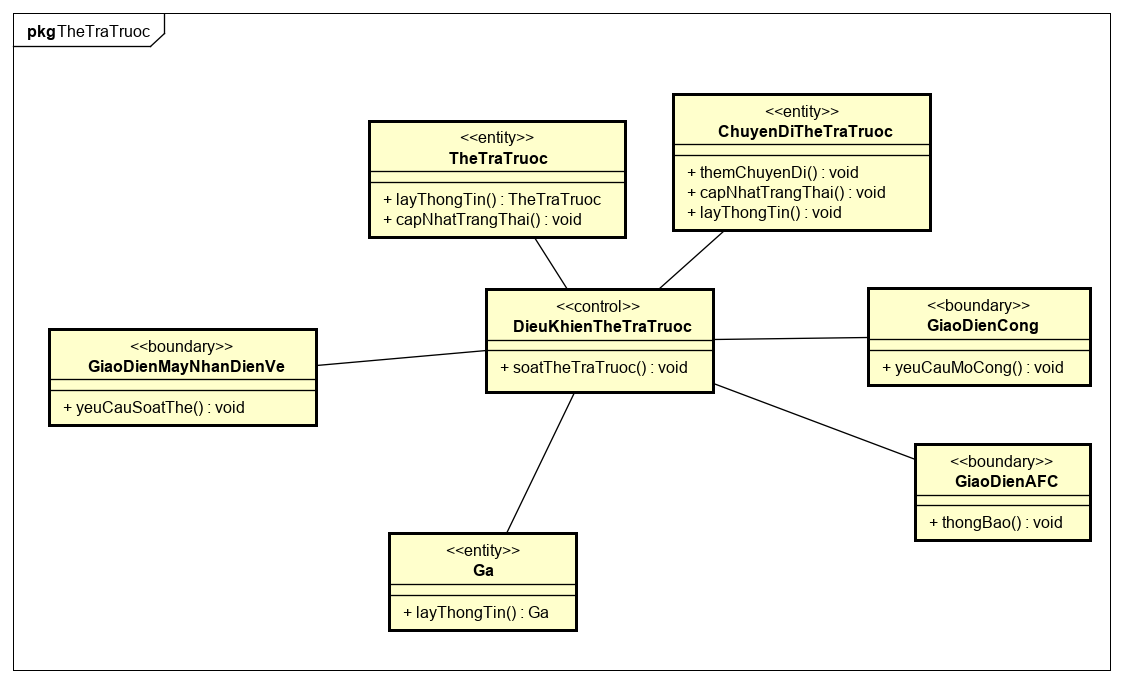
Hình 2.2.2: Biểu đồ lớp tổng quan “soát vé một chiều”

### Biểu đồ lớp tổng quan “soát vé 24h”



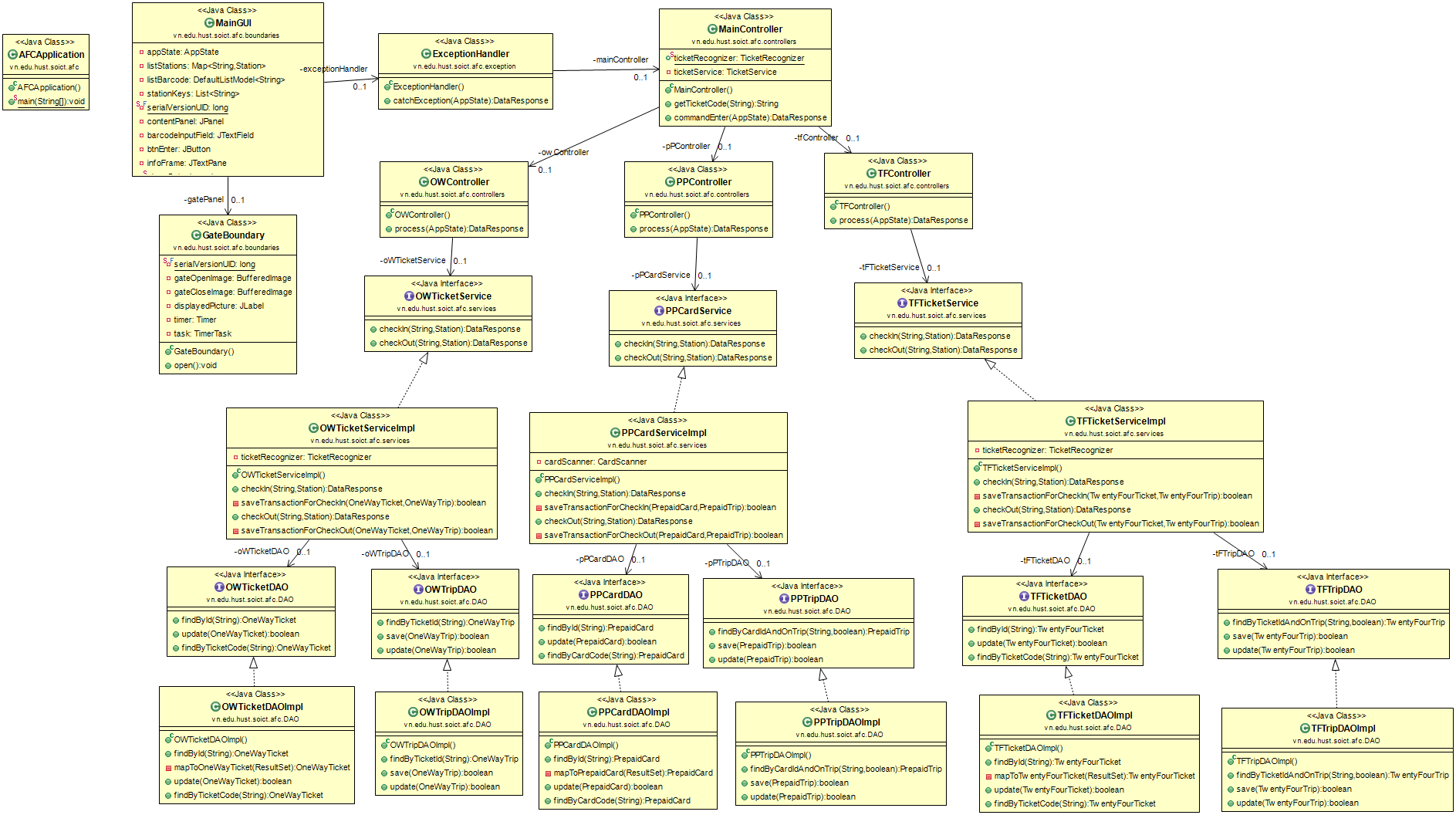
Hình 2.2.3: Biểu đồ lớp tổng quan “soát vé 24h”

### Biểu đồ lớp tổng quan “Soát thẻ trả trước”



Hình 2.2.4: Biểu đồ lớp tổng quan “soát thẻ trả trước”

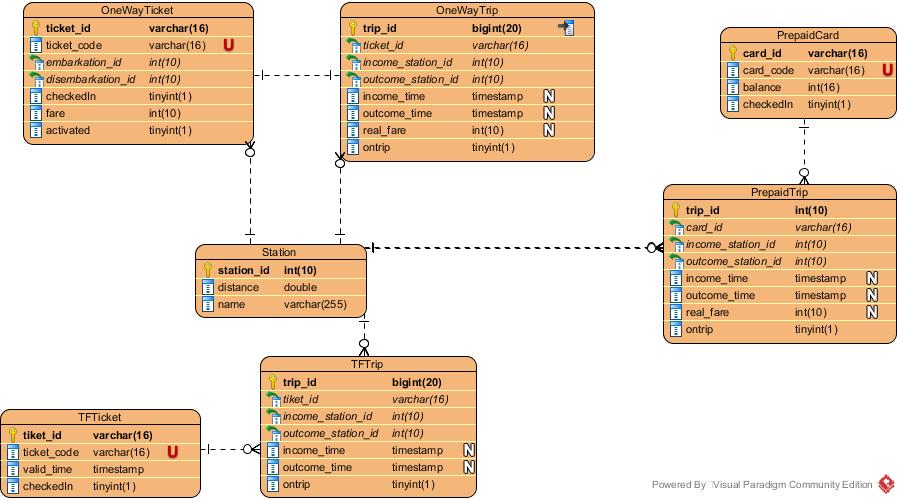
### Biểu đồ lớp chi tiết



Hình 2.2.5: Biểu đồ lớp chi tiết

## Thiết kế cơ sở dữ liệu

### Biểu đồ quan hệ



Hình 2.3.1: Biểu đồ quan hệ

### Thiết kế cơ sở dữ liệu

#### OneWayTicket

|  |  |  |
| --- | --- | --- |
| Column name | Datatype | Note |
| ticket\_id | VARCHAR(16) | PK |
| ticket\_code | VARCHAR(16) |  |
| embarkation\_id | INT(10) | FK |
| disembarkation\_id | INT(10) | FK |
| checkedin | TINYINT(1) |  |
| fare | INT(10) |  |
| activated | TINYINT(1) |  |

#### TFTicket (Twenty-Four Hours Ticket)

|  |  |  |
| --- | --- | --- |
| Column name | Datatype | Note |
| card \_id | VARCHAR(16) | PK |
| ticket\_code | VARCHAR(16) |  |
| valid\_time | TIMESTAMP |  |
| checkedin | TINYINT(1) |  |

#### PrepaidCard

|  |  |  |
| --- | --- | --- |
| Column name | Datatype | Note |
| ticket\_id | VARCHAR(16) | PK |
| card\_code | VARCHAR(16) |  |
| balance | INT(16) |  |
| checkedin | TINYINT(1) |  |

#### Station

|  |  |  |
| --- | --- | --- |
| Column name | Datatype | Note |
| station\_id | INT(10) | PK |
| distance | DOUBLE |  |
| name | VARCHAR(255) |  |

#### OneWayTrip

|  |  |  |
| --- | --- | --- |
| Column name | Datatype | Note |
| trip\_id | BIGINT(20) | PK |
| ticket\_id | VARCHAR(16) | FK |
| income\_station\_id | INT(10) | FK |
| outcome\_station\_id | INT(10) | FK |
| income\_time | TIMESTAMP |  |
| outcome\_time | TIMESTAMP |  |
| real\_fare | INT(10) |  |
| ontrip | TINYINT(1) |  |

#### TFTrip (Twenty\_Four Hour Ticket Trip)

|  |  |  |
| --- | --- | --- |
| Column name | Datatype | Note |
| trip\_id | BIGINT(20) | PK |
| ticket\_id | VARCHAR(16) | FK |
| income\_station\_id | INT(10) | FK |
| outcome\_station\_id | INT(10) | FK |
| income\_time | TIMESTAMP |  |
| outcome\_time | TIMESTAMP |  |
| ontrip | TINYINT(1) |  |

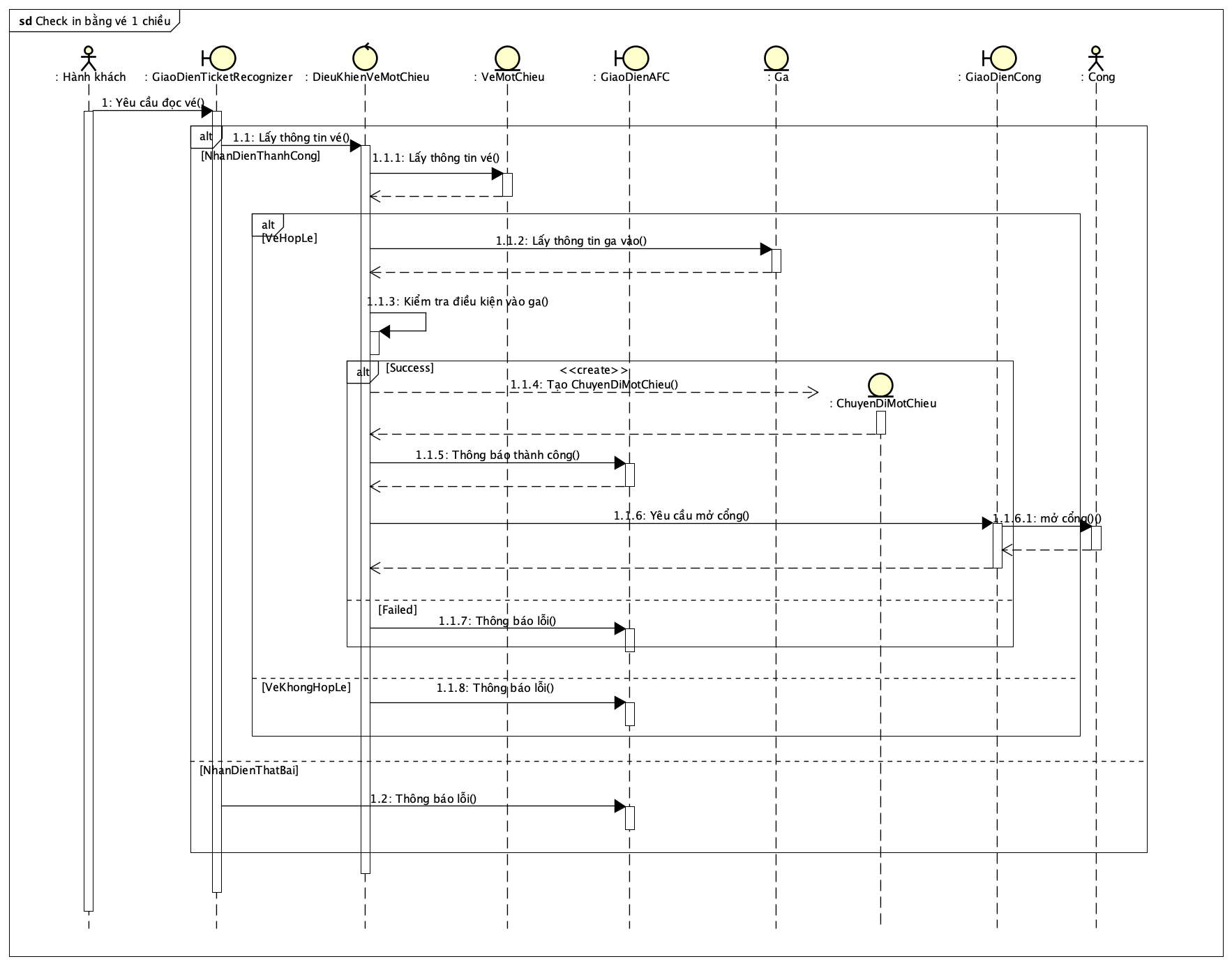
#### PrepaidTrip

|  |  |  |
| --- | --- | --- |
| Column name | Datatype | Note |
| trip\_id | BIGINT(20) | PK |
| card\_id | VARCHAR(16) | FK |
| income\_station\_id | INT(10) | FK |
| outcome\_station\_id | INT(10) | FK |
| income\_time | TIMESTAMP |  |
| outcome\_time | TIMESTAMP |  |
| real\_fare | INT(10) |  |
| ontrip | TINYINT(1) |  |

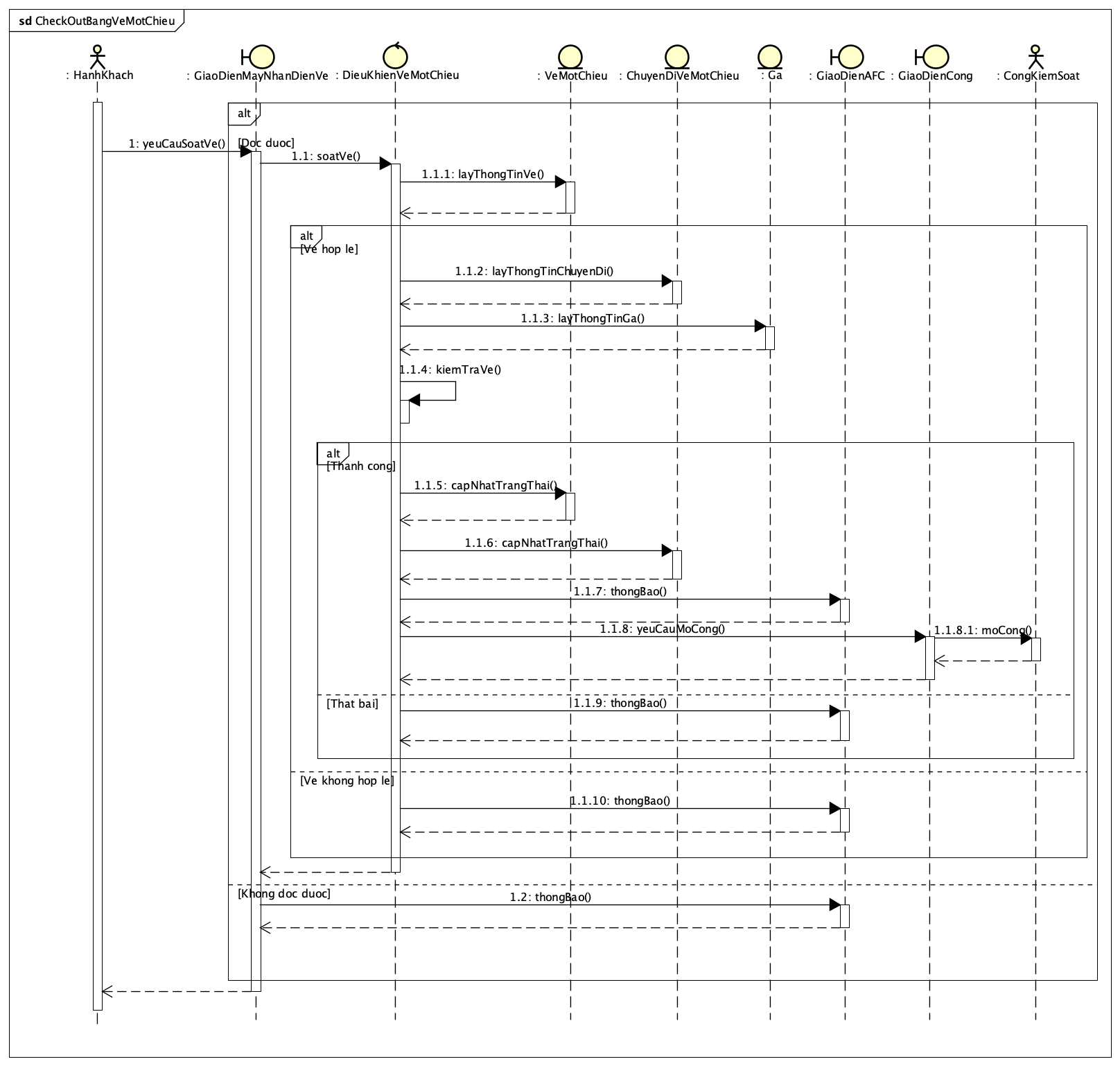
## Thiết kế giao diện

# Thiết kế hành vi động

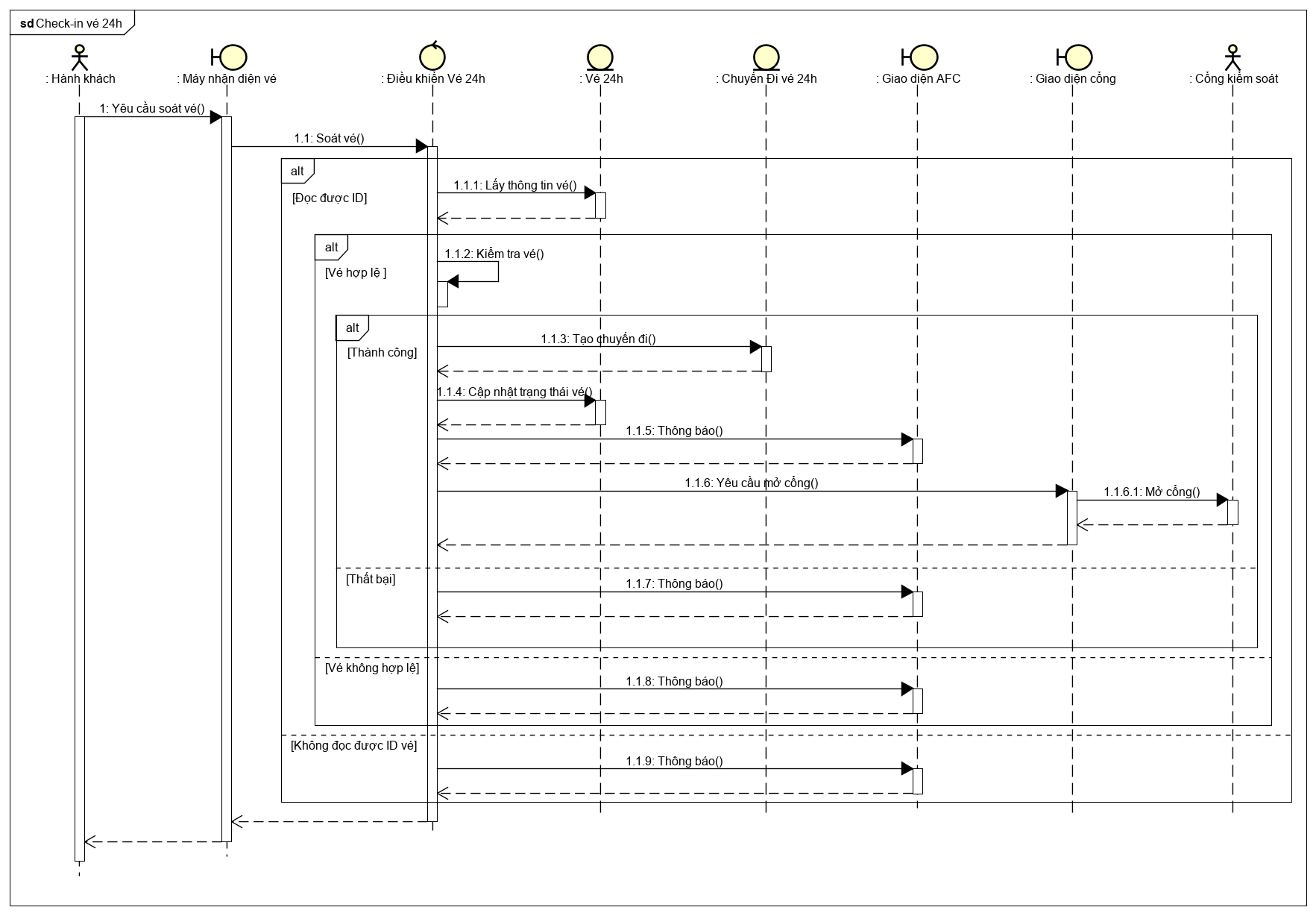
## Biểu đồ tuần tự use case UC001 “Check in bằng vé một chiều”



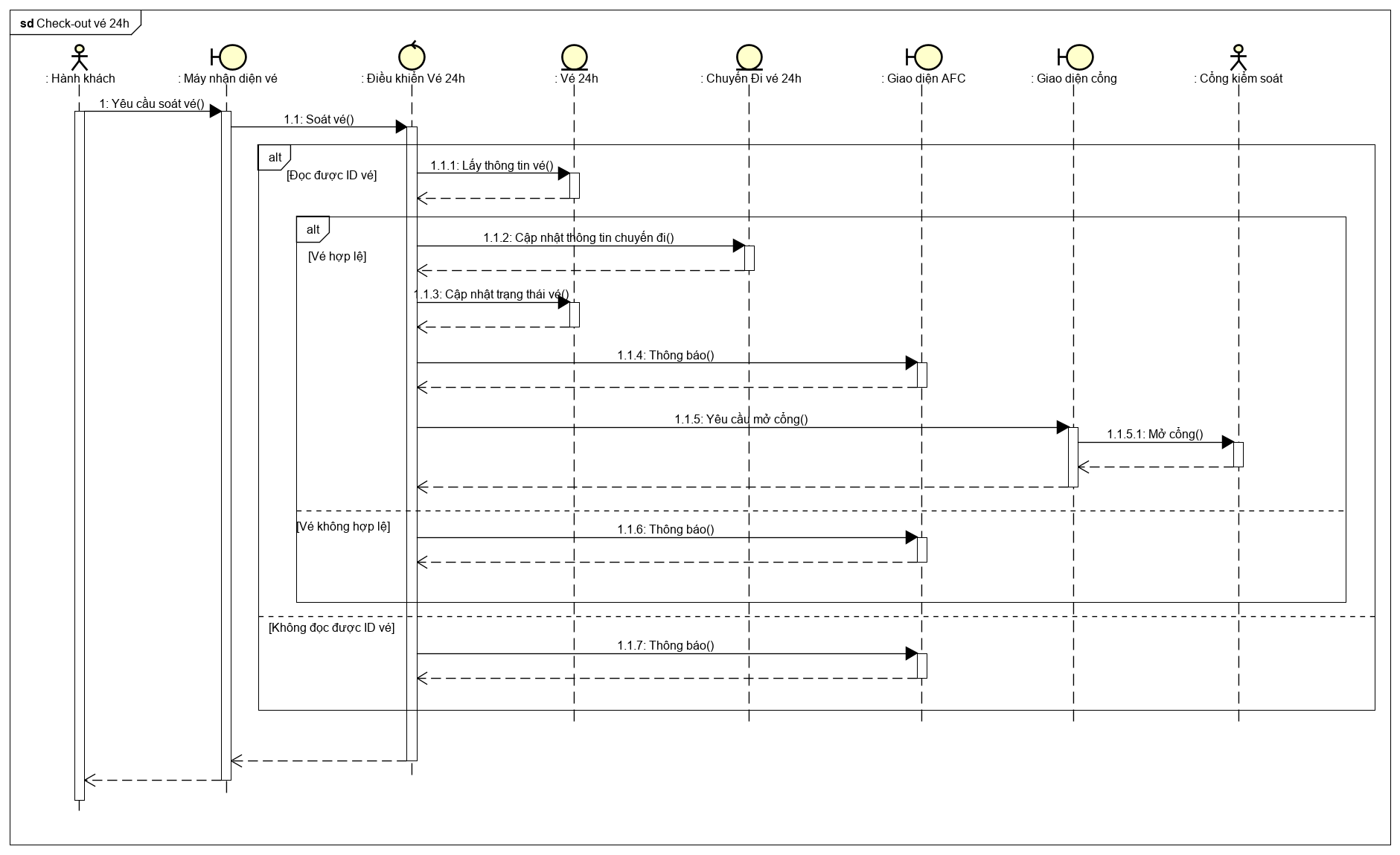
## Biểu đồ tuần tự use case UC002 “Check out bằng vé một chiều”



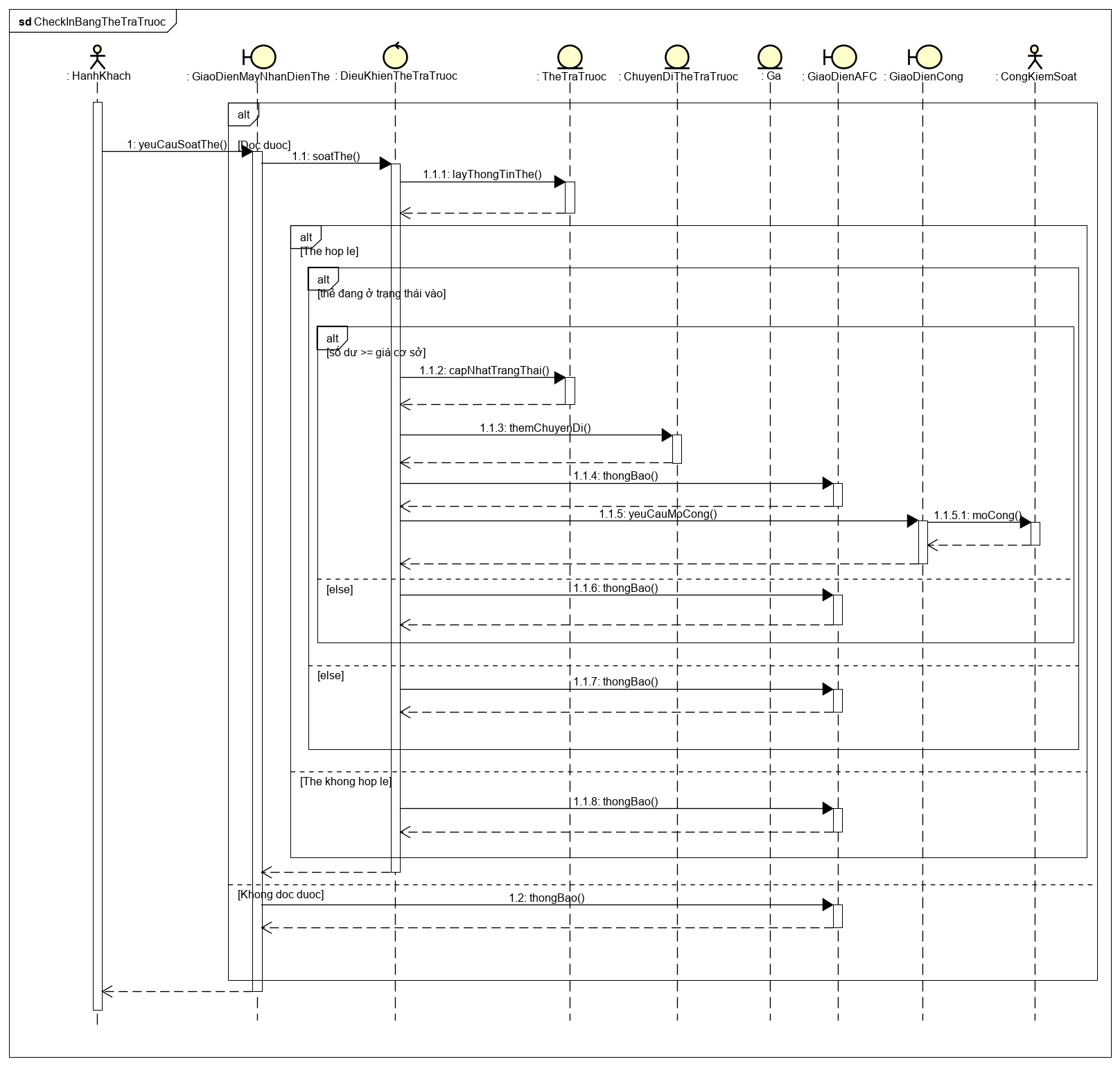
## Biểu đồ tuần tự use case UC003 “Check in bằng vé 24h”



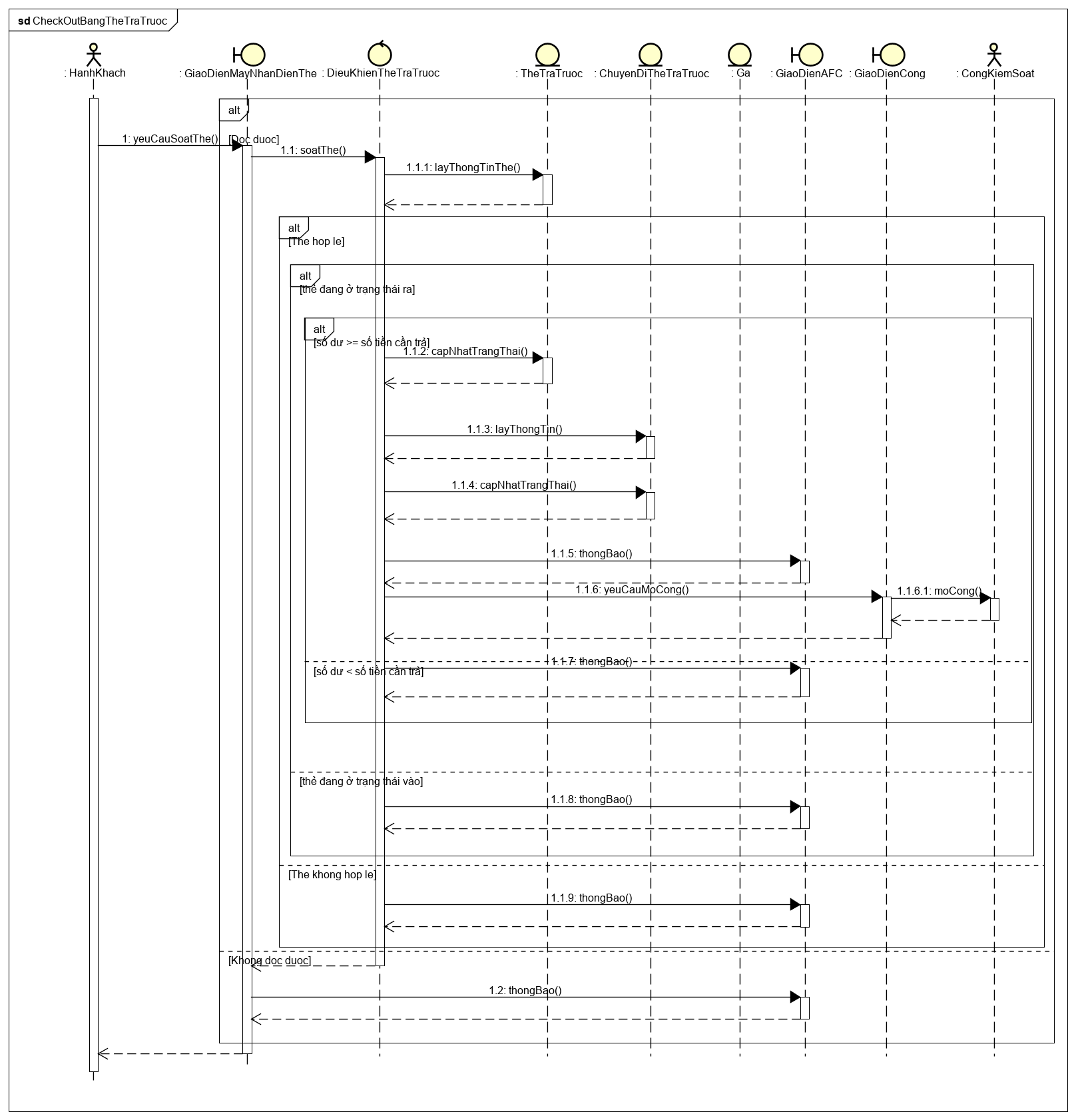
## Biểu đồ tuần tự use case UC004 “Check out bằng vé 24h”



## Biểu đồ tuần tự use case UC005 “Check in bằng thẻ trả trước”



## Biểu đồ tuần tự use case UC006 “Check out bằng thẻ trả trước”



# Phân tích các mức Coupling và Cohesion

# Design principles trong SOLID và Design Pattern

## Design principles trong SOLID

### Single responsibility principle

Một class chỉ nên giữ 1 trách nhiệm duy nhất (chỉ có thể sửa đổi class với 1 lý do duy nhất.

* Lớp **OneWayTicket** chỉ sử dụng cho lưu trữ thông tin vé 1 chiều.
* Lớp **TwentyFourTicket** chỉ sử dụng cho lưu trữ thông tin vé 24h.
* Lớp **PrepaidCard** chỉ sử dụng cho lưu trữ thông tin thẻ trả trước.
* Lớp **OWController** chỉ sử dụng để kiểm tra vé 1 chiều.
* Lớp **TFController** chỉ sử dụng để kiểm tra vé 24h.
* Lớp **PPController** chỉ sử dụng để kiểm tra thẻ trả trước.

Kết luận: Thực hiện tốt nguyên tắc **Single responsibility principle**, đảm bảo hầu hết các lớp và phương thức chỉ thự hiện một trách nhiệm duy nhất.

### Open-closed principle

Có thể thoải mái mở rộng 1 class, nhưng không được sửa đổi bên trong class đó.

Nguyên lý trên yêu cầu đáp ứng 2 điều kiện:

* Dễ mở rộng: Có thể dễ dàng nâng cấp, mở rộng, thêm tính năng mới cho một module khi có yêu cầu.
* Khó sửa đổi: Hạn chế hoặc cấm việc sửa đổi source code của module sẵn có.

Sử dụng:

* OWController, TFController và PPController chỉ định nghĩa các phương thức cần thiết cho việc check-in và check-out. Khi sử dụng thì chỉ cần kế thừa mà không cần sửa đổi mã nguồn.
* Khó sửa đổi: Do phân tích còn hạn chế và trong quá trình xây dựng còn nhiều lúc mắc lỗi nên mã nguồn luôn bị thay đổi mỗi lần chỉnh sửa. Khó áp dụng trong quá trình làm.

### Liskov substitution principle

Trong một chương trình, các đối tượng của lớp con có thể thay thế lớp cha mà không làm thay đổi tính đúng đắn của chương trình.

Chưa thấy áp dụng trong project này.

### Interface segregation principle

Thay vì dùng 1 giao diện lớn với mục định chung, ta nên tách thành nhiều giao diện nhỏ, với nhiều mục đích cụ thể.

Do đã áp dụng tốt nguyên tắc đầu tiên Single responsibility principle nên các giao diện được xây dựng theo các mục đích cụ thể, phục vụ một công việc cụ thể, không có sự xung đột trong các giao diện.

Kết luận: Nguyên tắc đã được áp dụng tốt trong project

### Dependency Inversion principle

Lớp chỉ phụ thuộc vào giao diện, ko phải cách cài đặt:

* Các module cấp cao không nên phụ thuộc vào các modules cấp thấp. Cả 2 nên phụ thuộc vào abstraction.
* Các lớp giao tiếp với nhau thông qua giao diện, không phải thông qua implementation.

Việc áp dụng DAO nên các lớp được giao tiếp với nhau đều qua giao diện.

## Design Pattern

Design Pattern: DAO Pattern

Controller có các giao diện tương ứng với từng nhiệm vụ riêng biệt. Việc giao tiếp giữa các lớp thông quan giao diện cụ thể.

# Ưu điểm của quá trình thay đổi

## Thêm các loại vé, thẻ mới

Các lớp liên quan đến từng loại vé vé và thẻ đều có giao diện cụ thể:

* Controller có giao diện của controller cho từng loại vé, thẻ.
* Entity có giao diện của entity cho từng loại vé, thẻ
* Boundary có giao diện của boundary cho từng loại vé, thẻ.
* Việc mở rộng thêm các loại vé, thẻ mới là hết sức đơn giản.

## Thay đổi cách tính giá vé

### Thay đổi ngưỡng, lượng tiền trong mỗi ngưỡng

### Thay đổi giá tiền theo khoảng cách, tính giá tiền theo vùng.

## Thay đổi tuyến đường