TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Tài liệu mô tả thiết kế phần mềm

(Software Design Description – SDD)

Phiên bản 1.0

HỆ THỐNG SOÁT VÉ / THẺ TỰ ĐỘNG

Môn: Thiết kế và xây dựng phần mềm

Nhóm 04

* Trương Thành Duy
* Cao Văn Duy
* Phùng Việt Duy
* Nguyễn Ngọc Hải

*Hà Nội, 2019*

Mục lục

[1 Thiết kế cấu trúc tĩnh 3](#_Toc26696197)

[1.1 Biểu đồ lớp tổng quan 3](#_Toc26696198)

[1.2 Thiết kế cơ sở dữ liệu 4](#_Toc26696199)

[1.2.1 Biểu đồ quan hệ của hệ thống 4](#_Toc26696200)

[1.2.2 Thiết kế cơ sở dữ liệu 4](#_Toc26696201)

[1.3 Thiết kế UI 7](#_Toc26696202)

[1.3.1 Biểu đồ chuyển hướng UI 7](#_Toc26696203)

[1.3.2 Thiết kế giao diện 8](#_Toc26696204)

[2 Thiết kế hành vi động 14](#_Toc26696205)

[2.1 Biểu đồ tuần tự use case UC001 “Check in bằng vé một chiều” 14](#_Toc26696206)

[2.2 Biểu đồ tuần tự use case UC002 “Check out bằng vé một chiều” 15](#_Toc26696207)

[2.3 Biểu đồ tuần tự use case UC003 “Check in bằng vé 24h” 16](#_Toc26696208)

[2.4 Biểu đồ tuần tự use case UC004 “Check out bằng vé 24h” 17](#_Toc26696209)

[2.5 Biểu đồ tuần tự use case UC005 “Check in bằng thẻ trả trước” 18](#_Toc26696210)

[2.6 Biểu đồ tuần tự use case UC006 “Check out bằng thẻ trả trước” 19](#_Toc26696211)

[3 Phân tích Coupling và Cohesion 20](#_Toc26696212)

[3.1 Coupling 20](#_Toc26696213)

[3.1.1 Content coupling 20](#_Toc26696214)

[3.1.2 Common coupling 20](#_Toc26696215)

[3.1.3 Control coupling 20](#_Toc26696216)

[3.1.4 Stamp coupling 20](#_Toc26696217)

[3.1.5 Data coupling 20](#_Toc26696218)

[3.2 Cohesion 20](#_Toc26696219)

[3.2.1 Coincidental cohesion 20](#_Toc26696220)

[3.2.2 Logical cohesion 20](#_Toc26696221)

[3.2.3 Temporal cohesion 20](#_Toc26696222)

[3.2.4 Procedural cohesion 20](#_Toc26696223)

[3.2.5 Communicational cohesion 21](#_Toc26696224)

[3.2.6 Sequential cohesion 21](#_Toc26696225)

[3.2.7 Informational cohesion 21](#_Toc26696226)

[3.2.8 Functional cohesion 21](#_Toc26696227)

[4 Phân tích theo Design Principles và Design Pattern 22](#_Toc26696228)

[4.1 Design Principles 22](#_Toc26696229)

[4.1.1 Single responsibility principle 22](#_Toc26696230)

[4.1.2 Open-closed principle 22](#_Toc26696231)

[4.1.3 Liskov substitution principle 22](#_Toc26696232)

[4.1.4 Interface segregation principle 22](#_Toc26696233)

[4.1.5 Dependency Inversion principle 23](#_Toc26696234)

[4.2 Design Pattern 23](#_Toc26696235)

[4.3 Ưu điểm trong việc thay đổi yêu cầu 23](#_Toc26696236)

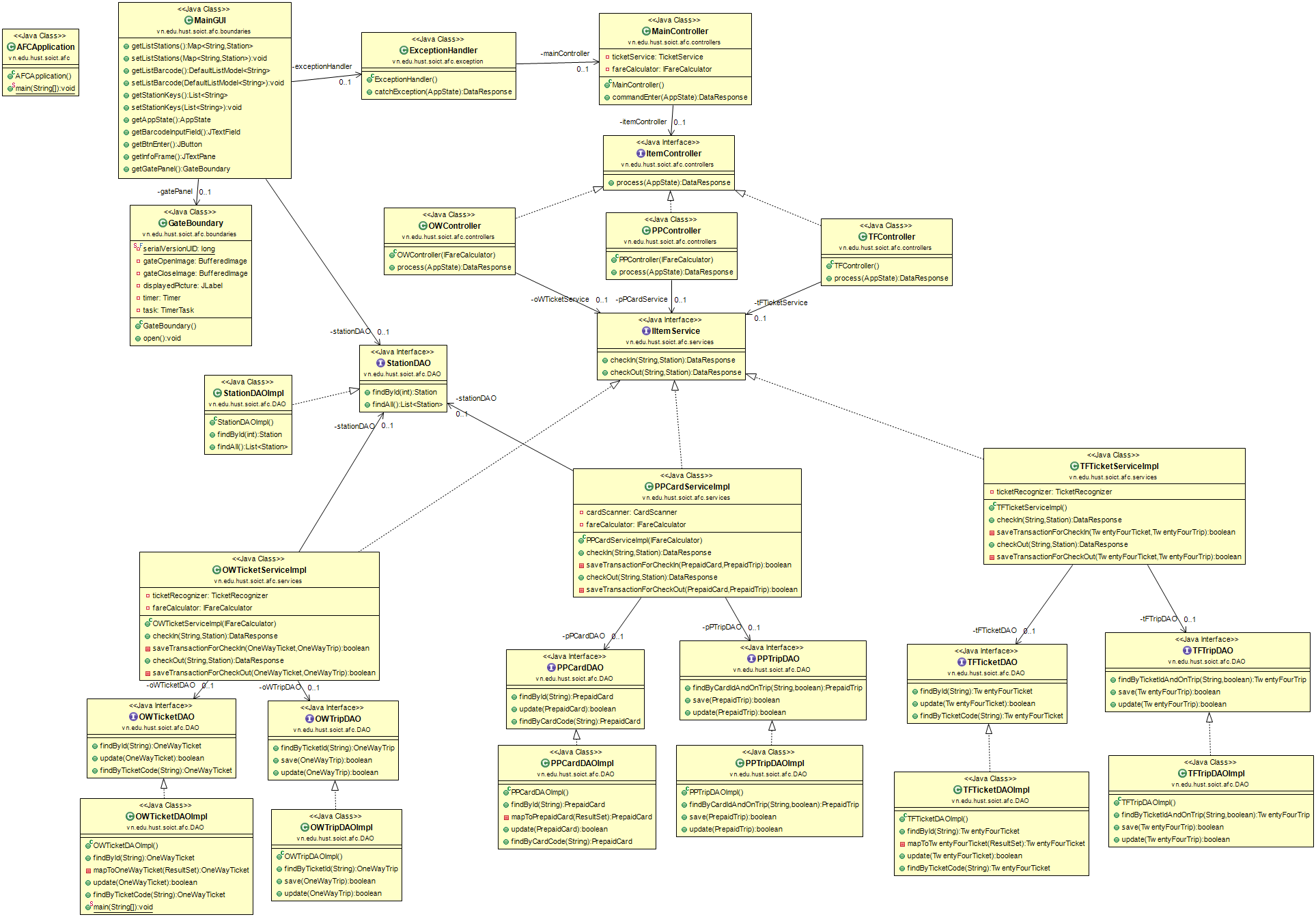
[4.3.1 Yêu cầu thay đổi cách tính giá tiền 23](#_Toc26696237)

[4.3.2 Yêu cầu thay đổi thành nhiều tuyến đường 24](#_Toc26696238)

[4.3.3 Thêm các loại vé/thẻ mới 24](#_Toc26696239)

# Thiết kế cấu trúc tĩnh

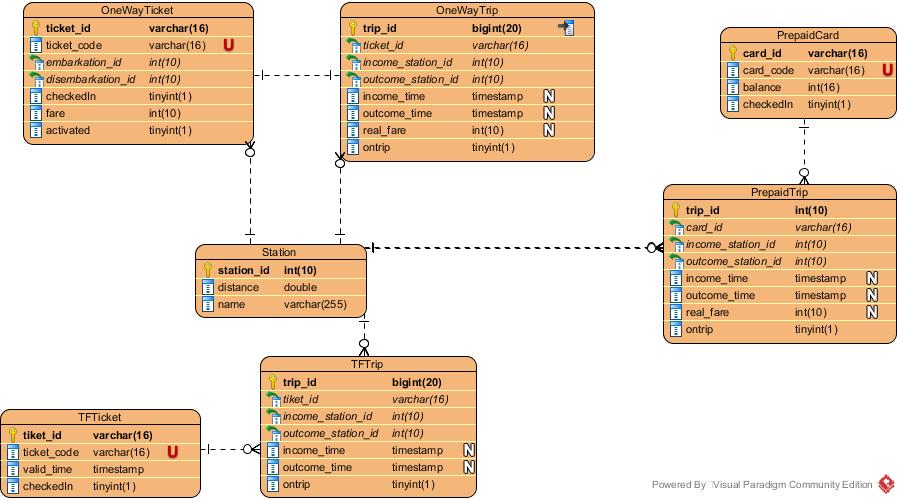
## Biểu đồ lớp tổng quan



Ảnh 1‑1: Lớp tổng quan hệ thống.

## Thiết kế cơ sở dữ liệu

### Biểu đồ quan hệ của hệ thống



Ảnh 1‑2: Biểu đồ quan hệ của hệ thống

### Thiết kế cơ sở dữ liệu

#### OneWayTicket (Vé một chiều)

|  |  |  |
| --- | --- | --- |
| 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 (Vé 24h)

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

#### PrepaidCard (Thẻ trả trước)

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

#### Station (Ga)

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

#### OneWayTrip (chuyến đi vé một chiều)

|  |  |  |
| --- | --- | --- |
| 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 (Chuyến đi vé 24h)

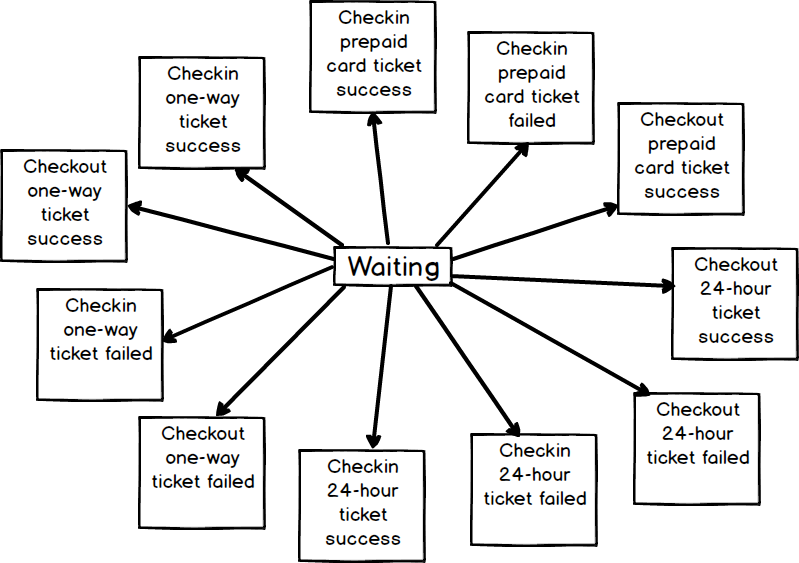
|  |  |  |
| --- | --- | --- |
| Column name | Datatype | Note |
| trip\_id | BIGINT(20) | PK |
| ticket\_id | VARCHAR(16) | FK |
| income\_station\_id | INT(10) | FK |
| outcome\_station\_id | INT(10) | FK |
| income\_time | TIMESTAMP |  |
| outcome\_time | TIMESTAMP |  |
| ontrip | TINYINT(1) |  |

#### PrepaidTrip (Chuyến đi thẻ trả trước)

|  |  |  |
| --- | --- | --- |
| 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 UI

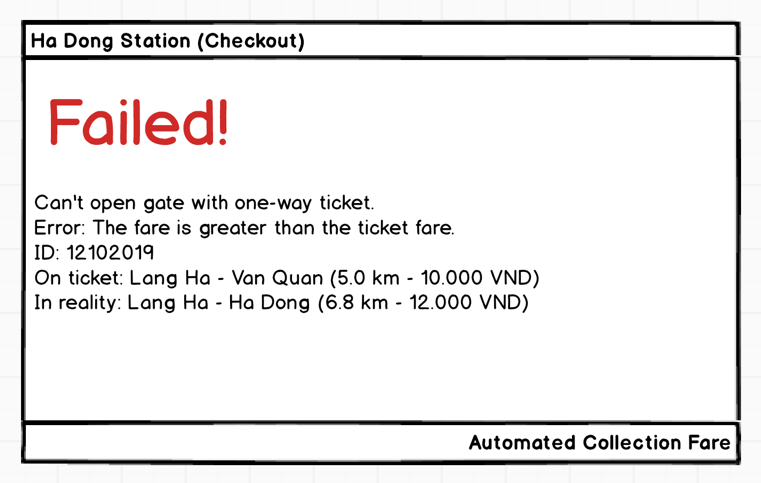
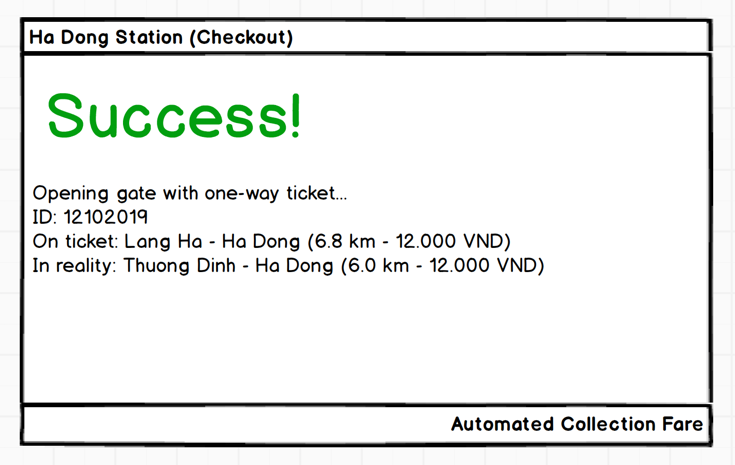
### Biểu đồ chuyển hướng UI

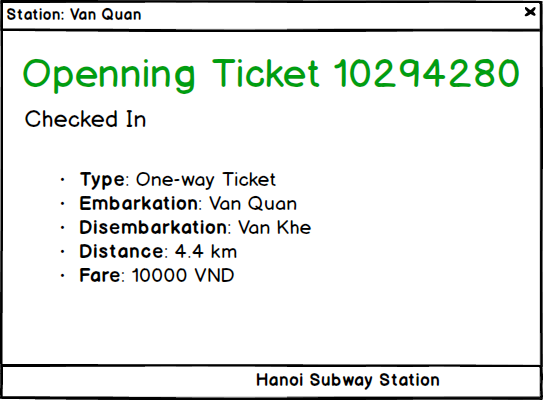
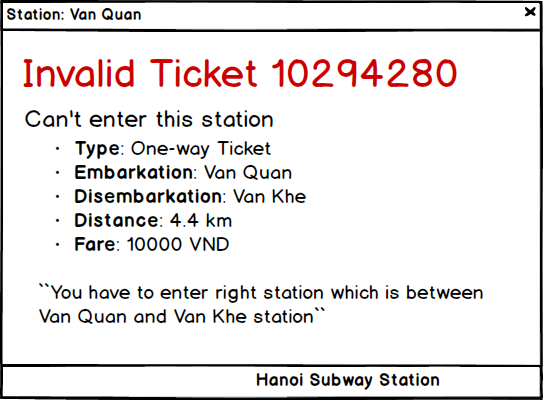


Ảnh 1‑3: Biểu đồ chuyển hướng giao diện soát vé/thẻ

### Thiết kế giao diện

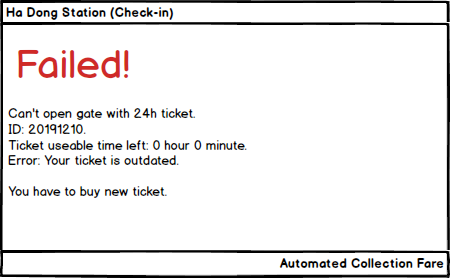
#### Soát vé một chiều

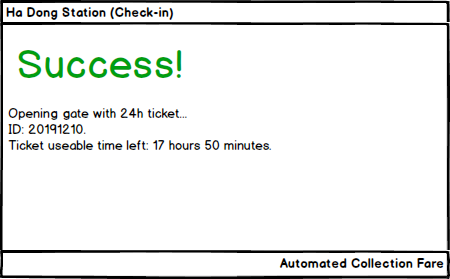


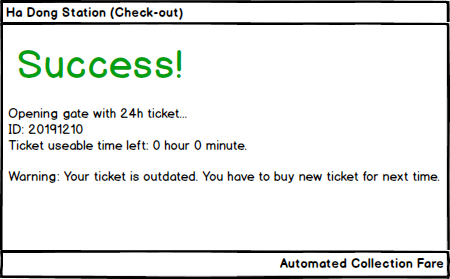


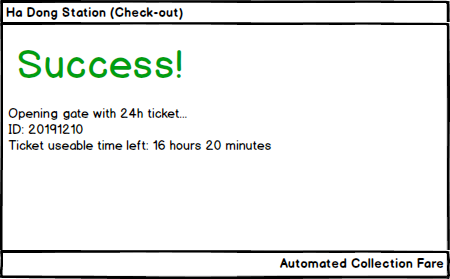
Ảnh 1‑4,5,6,7: Soát vé một chiều

#### Soát vé 24h



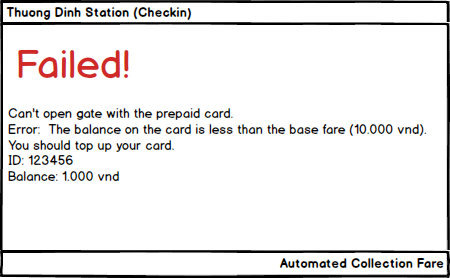


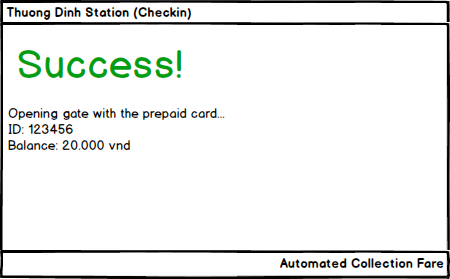


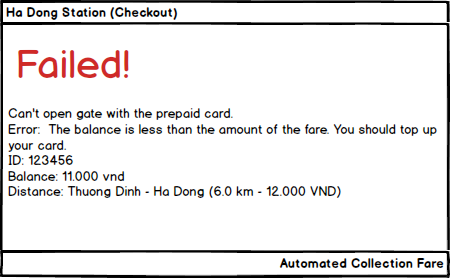


Ảnh 1-8,9,10,11: Soát vé 24h

#### Soát thẻ trả trước







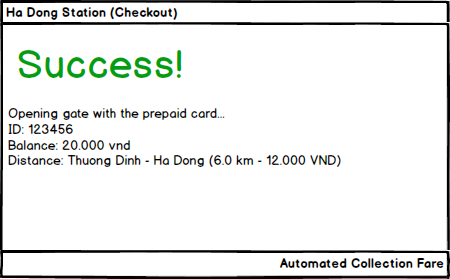
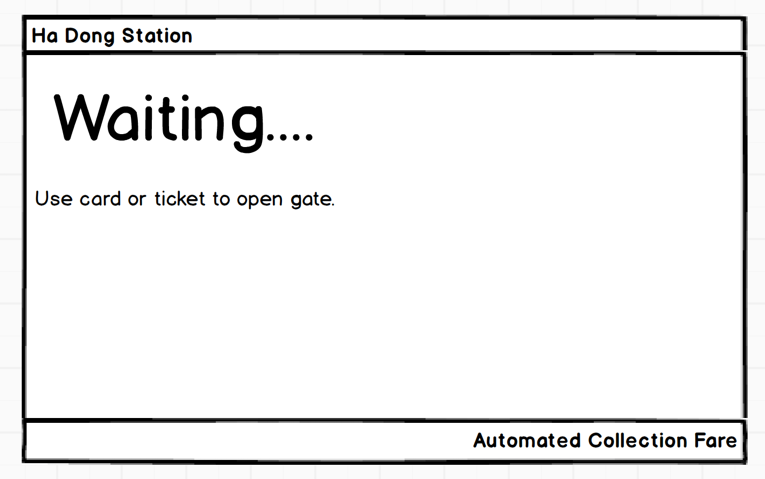


Figure 1‑12,13,14,15: Soát thẻ trả trước

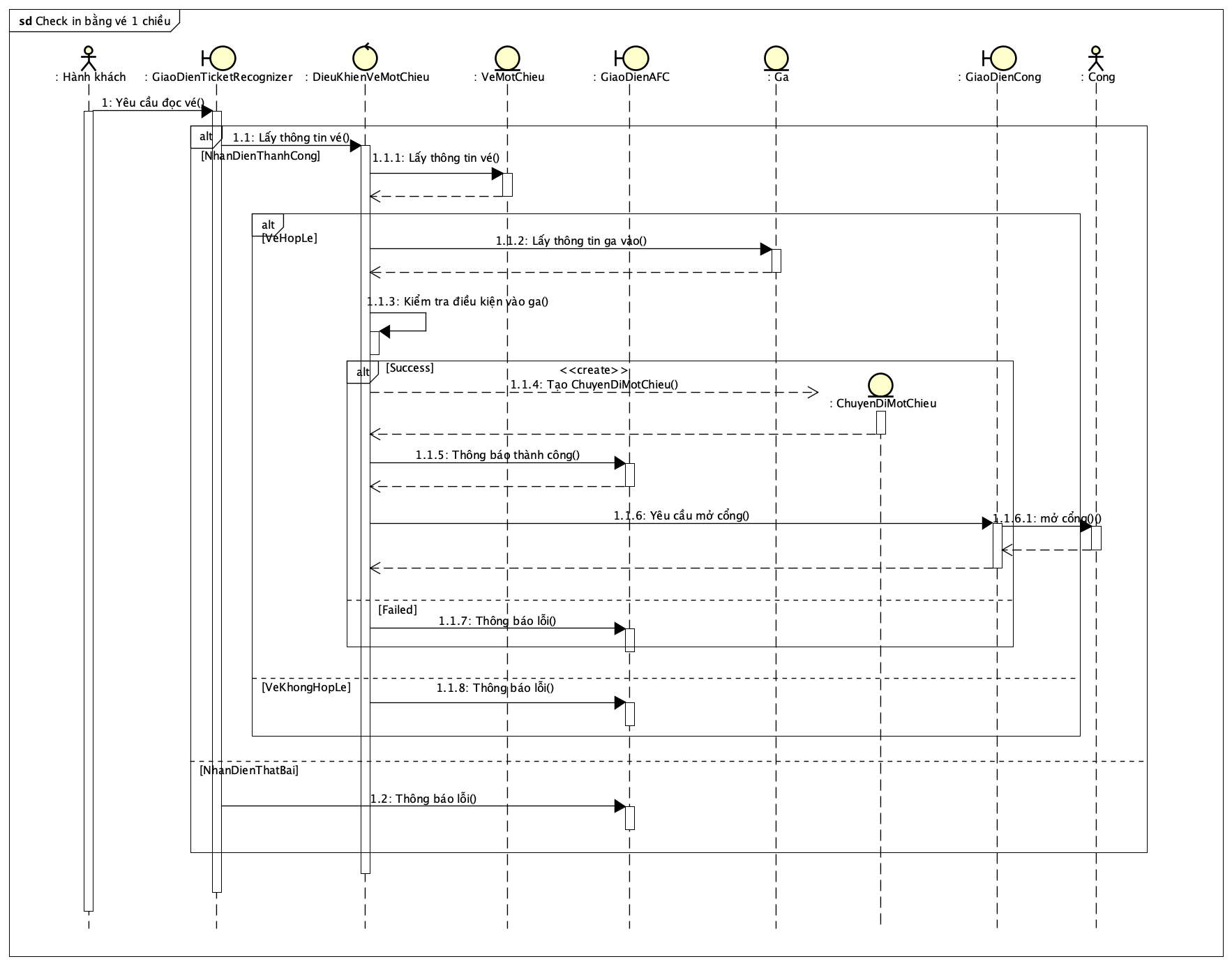
#### Màn hình chờ



Ảnh 1‑16: Màn hình chờ

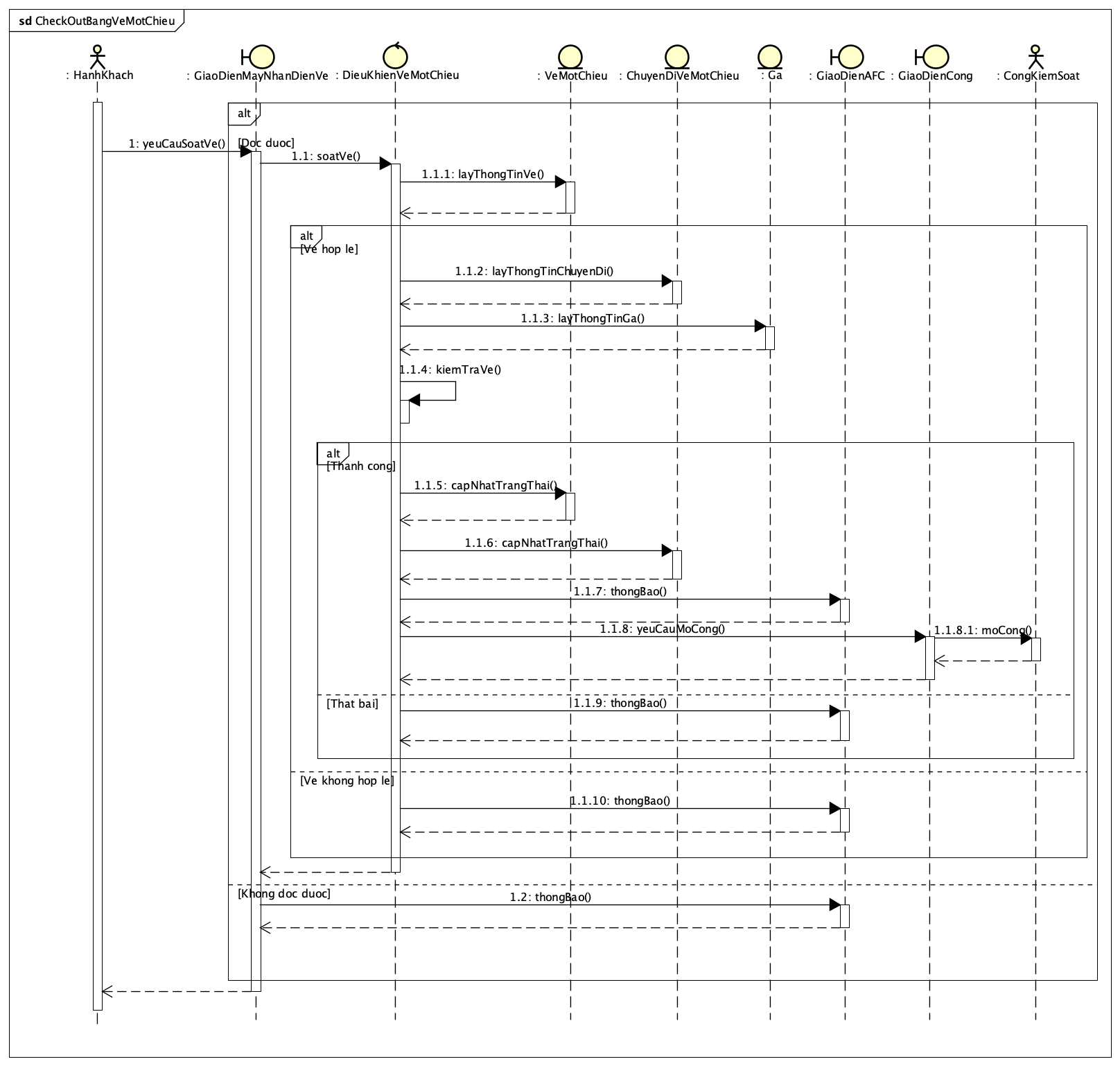
# Thiết kế hành vi động

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



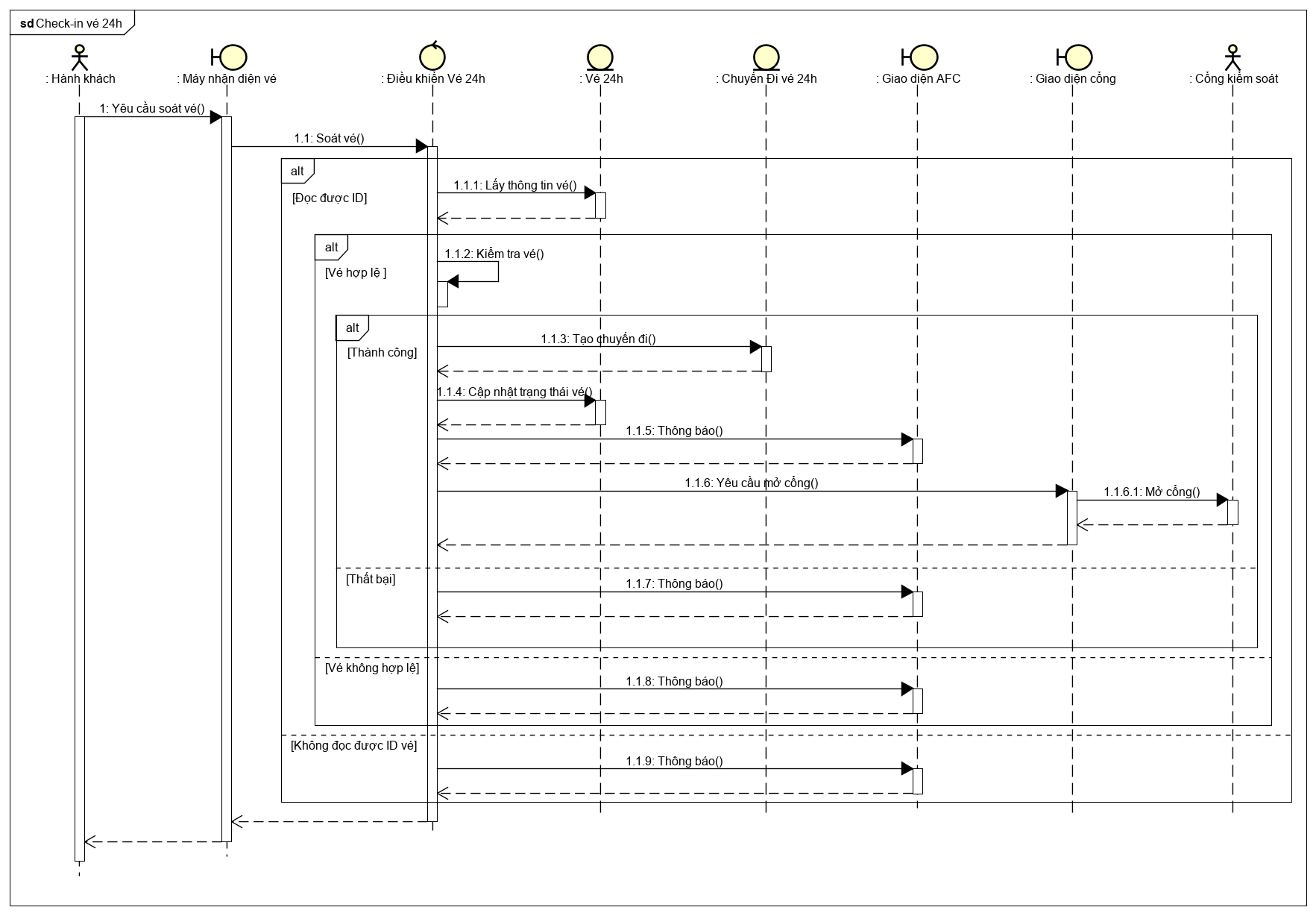
Ảnh 2‑1: Biểu đồ tuần tự use case UC001 “Check in bằng vé một chiều”

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



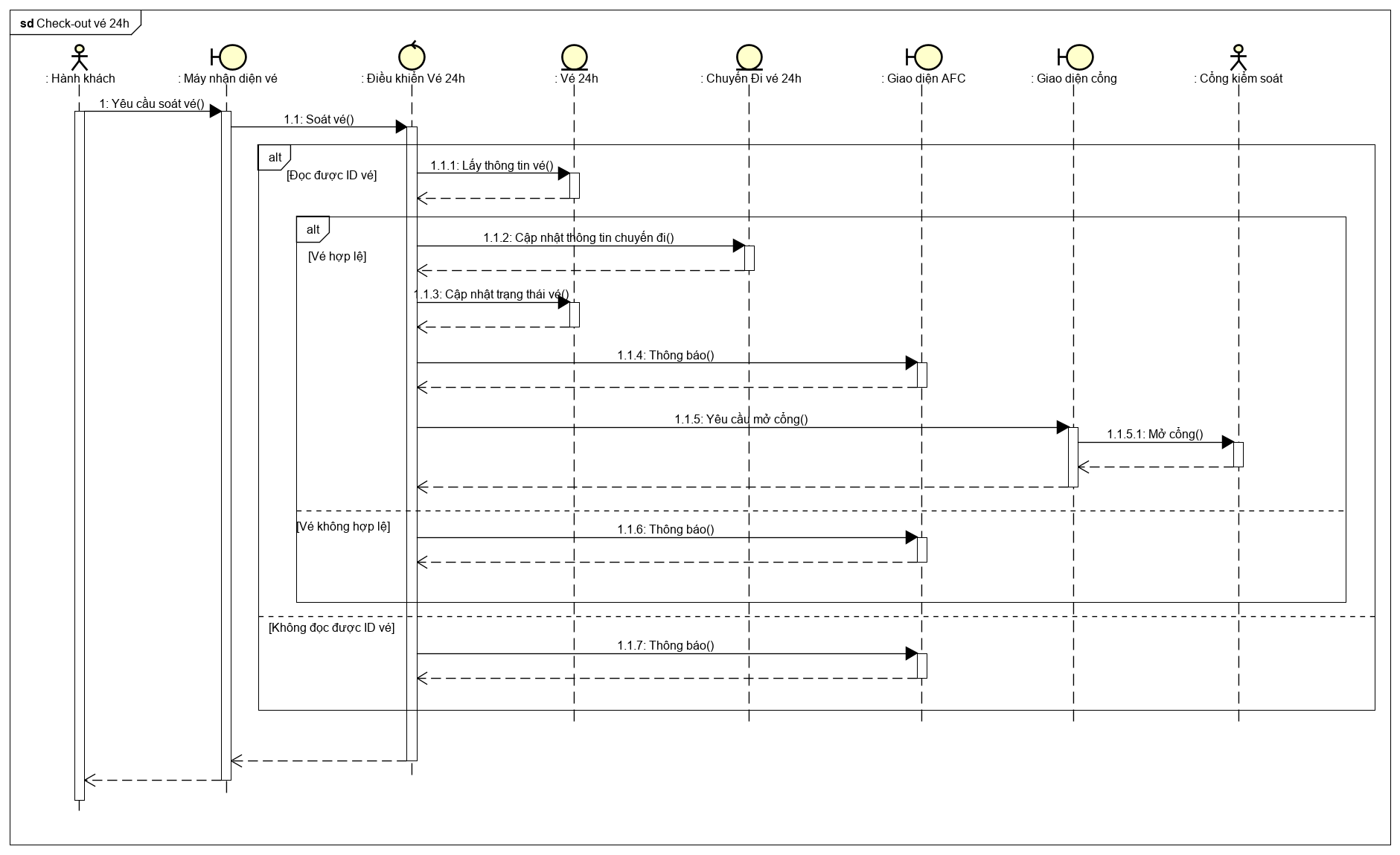
Ảnh 2‑2: Biểu đồ tuần tự use case UC002 “Check out bằng vé một chiều”

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



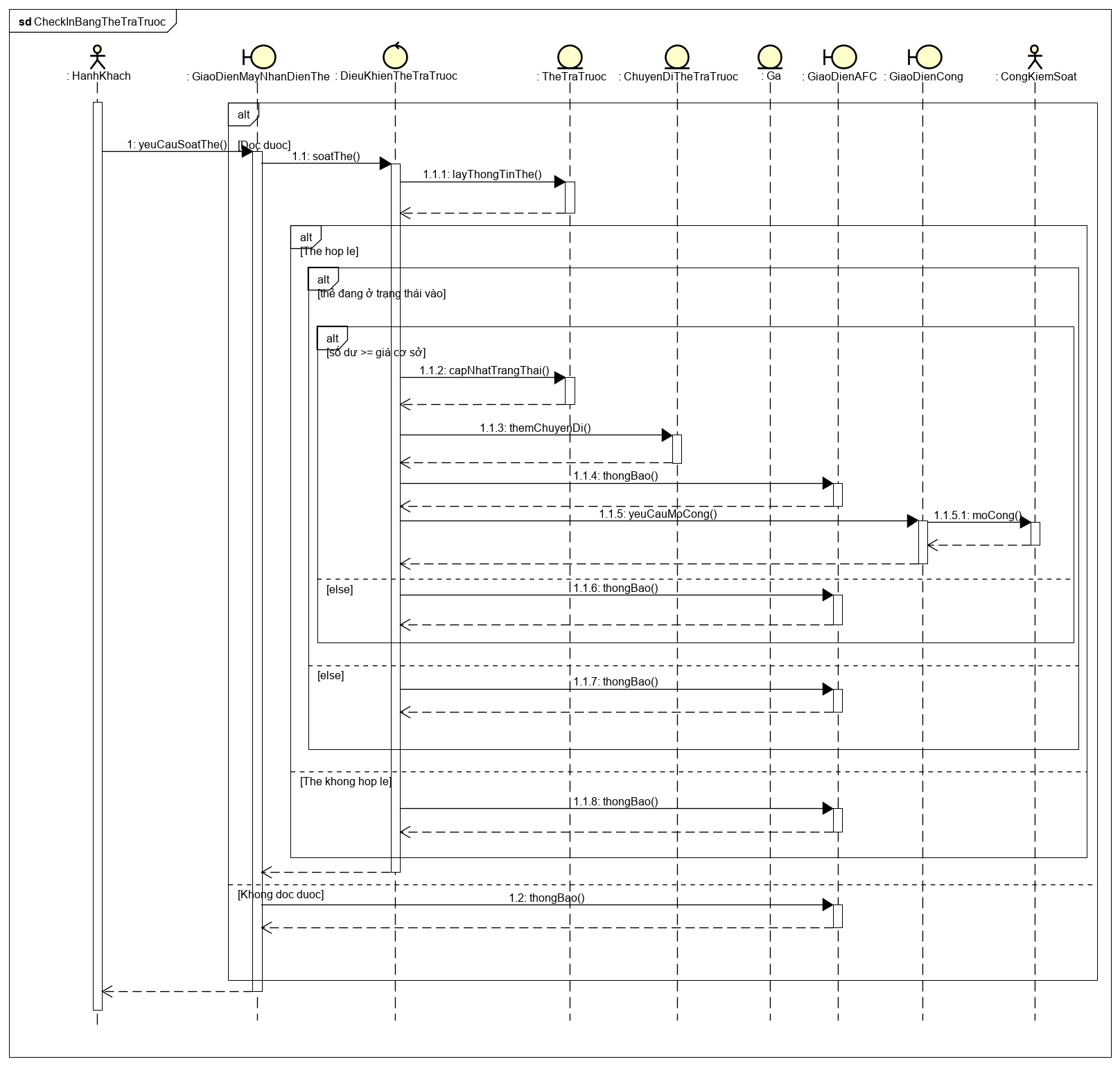
Ảnh 2‑3: Biểu đồ tuần tự use case UC003 “Check in bằng vé 24h”

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



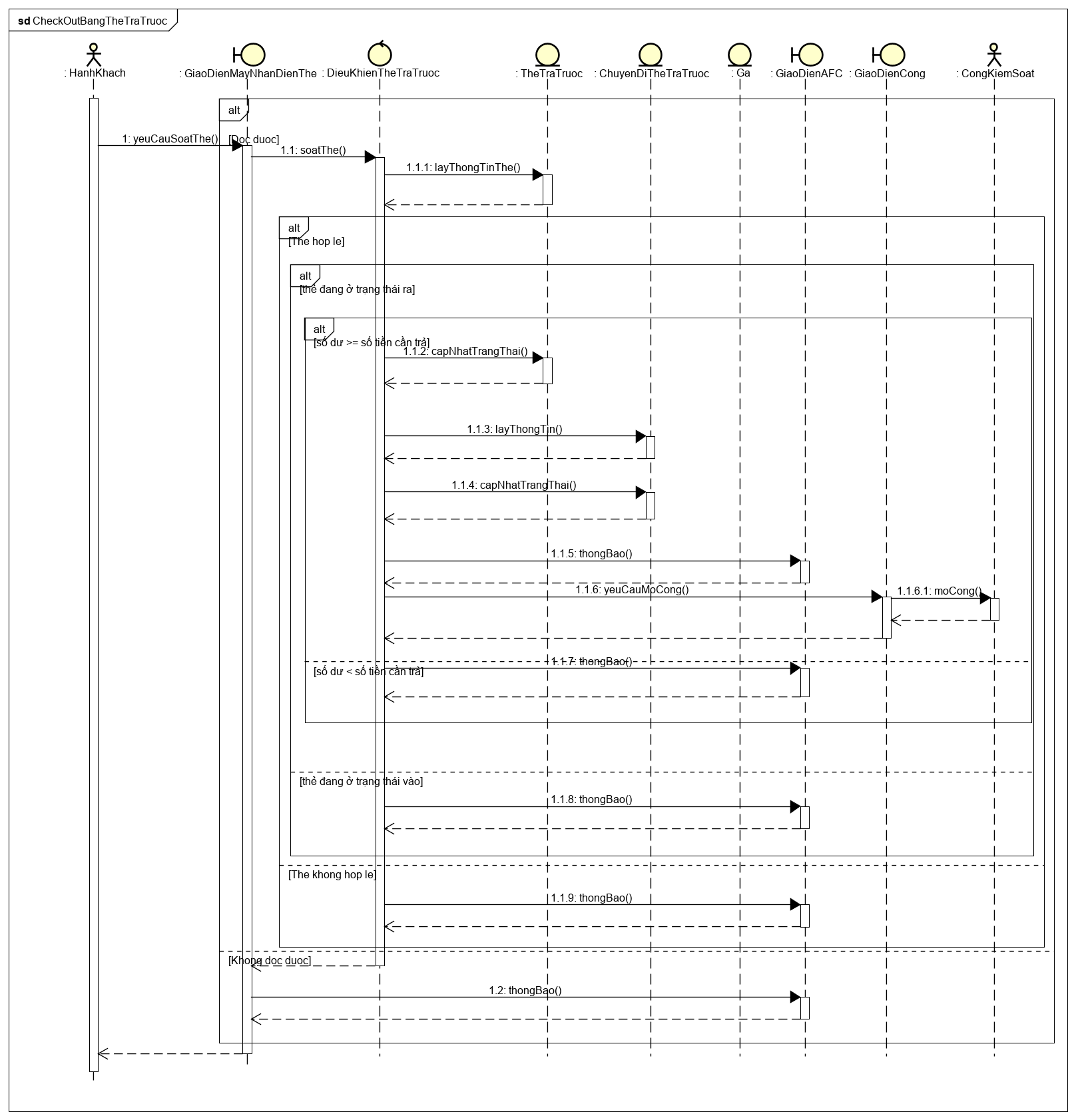
Ảnh 2‑4: Biểu đồ tuần tự use case UC004 “Check out bằng vé 24h”

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



Ảnh 2‑5: 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”



Ảnh 2‑6: Biểu đồ tuần tự use case UC006 “Check out bằng thẻ trả trước”

# Phân tích Coupling và Cohesion

## Coupling

### Content coupling

(Không có)

### Common coupling

(Không có)

### Control coupling

* Lớp **MainController**: Sử dụng thuộc tính **isByTicket** của tham số **appState** trong phương thức **commandEnter** để xác định điều hướng đến controller bằng vé hay thẻ.
* Các lớp **OWController**, **PPController**, **TFController**: Sử dụng thuộc tính **isActCheckIn** của tham số **appState** trong phương thức **process** để xác định hành động soát vé ra hay vào.

### Stamp coupling

* Lớp **AFareCalculator**: Truyền trực tiếp cả đối tượng **incomeStation** và **outcomeStation** vào trong phương thức **calculate**, phục vụ việc các lớp con kế thừa để ghi đè hàm này có thể sử dụng các thuộc tính khác nhau của Station cho việc tính toán.

### Data coupling

(Không có)

## Cohesion

### Coincidental cohesion

* Các hàm tiện ích đặt chung với nhau: **calculate** (tính giá vé), **roundOneDigitAfterAComma** (làm tròn số) trong một lớp.

Sửa: tách sang các lớp khác nhau.

### Logical cohesion

(Không có)

### Temporal cohesion

(Không có)

### Procedural cohesion

* Các lớp thao tác với Cơ sở dữ liệu và thao tác nghiệp vụ chung trong cùng 1 file **Service**.

Sửa: thêm package **dao** và các lớp **DAO** phục vụ riêng cho việc thao tác với Cơ sở dữ liệu.

### Communicational cohesion

(Không có)

### Sequential cohesion

(Không có)

### Informational cohesion

(Không có)

### Functional cohesion

(Không có)

# Phân tích theo Design Principles và Design Pattern

## Design Principles

### 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 thực 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 áp dụng trong project).

### 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

Mẫu thiết kế: **DAO (Data access object) Pattern:** Là một trong những pattern thuộc nhóm cấu trúc (Structural Pattern). Mẫu thiết kế DAO được sử dụng để phân tách logic lưu trữ dữ liệu trong một lớp riêng biệt. Theo cách này, các service được che dấu về cách các hoạt động cấp thấp để truy cập cơ sở dữ liệu được thực hiện.

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 trong việc thay đổi yêu cầu

### Yêu cầu thay đổi cách tính giá tiền

#### Thay đổi ngưỡng và lượng tiền mỗi ngưỡng

Để thay đổi ngưỡng và lượng tiền mỗi ngưỡng, ta sửa lại các hằng số trong interface **IFareCalculator** mà không phải sửa đổi bất kì phương thức hay lớp nào.

#### Đổi cách tính tiền theo khoảng cách sang tính tiền theo vùng

Để đổi cách tính tiền theo khoảng cách sang tính tiền theo vùng, trước hết ta sẽ thay đổi về cách lưu của **Station** trong Cơ sở dữ liệu và lớp entity: xóa thuộc tính **distance** (khoảng cách từ ga đầu) và thêm thuộc tính **zone** (số vùng). Thêm bảng trong Cơ sở dữ liệu lưu khoảng cách các vùng. Sau đó thêm một lớp **FareCalculatorByZone** implements **IfareCalculator** và ghi đè phương thức calculate (giống như lớp **FareCalculatorByDistance**), tiếp đến thay đổi khai báo thuộc tính **fareCalculator** trong lớp **MainController** bằng constructor của **FareCalculatorByZone**. Lớp **FareCalculatorByDistance** cũ không phải thay đổi gì cả.

### Yêu cầu thay đổi thành nhiều tuyến đường

Để thay đổi thành nhiều tuyến đường, trước hết ta sẽ thay đổi về cách lưu của **Station** trong Cơ sở dữ liệu và lớp entity: xóa thuộc tính **distance** (khoảng cách từ ga đầu). Thêm bảng trong Cơ sở dữ liệu lưu khoảng cách các giữa 2 ga gần nhất có đường đi. Sau đó thêm một lớp chứa thuật toán tính quãng đường ngắn nhất giữa 2 ga bất kì, và dựa vào con số đó để tính giá vé.

### Thêm các loại vé/thẻ mới

Bổ sung thêm các lớp Controller, DAO, Entity, Service và các exception mới. Phần điều hướng trong MainController thêm một tùy chọn là loại vé/thẻ mới.