**SDD**

Nhóm: ISD.VN.20191-09

|  |  |
| --- | --- |
| 20164069 | Nguyễn Mạnh Tiến |
| 20144318 | Nguyễn Đình Thơ |
| 20153619 | Phạm Hữu Thọ |
| 20167386 | Trần Văn Thông |
| 20165645 | Phùng Thị Trang |

Mục Lục

[I. Static Structure 1](#_Toc26719666)

[1. Class diagram 1](#_Toc26719667)

[2. Data Modeling 22](#_Toc26719668)

[2.1. Conceptual Data Model 22](#_Toc26719669)

[2.2. Logical Data Model 22](#_Toc26719670)

[2.3. Database Design 23](#_Toc26719671)

[3. UI Design 24](#_Toc26719672)

[II. Dynamic Structure 26](#_Toc26719673)

[1. Sequence diagram 26](#_Toc26719674)

[1.1. [Nguyễn Mạnh Tiến] Enter with one-way ticket 26](#_Toc26719675)

[1.2. [Nguyễn Mạnh Tiến] Exit with one-way ticket 27](#_Toc26719676)

[1.3. [Phạm Hữu Thọ] Enter with prepaid card 28](#_Toc26719677)

[1.4. [Nguyễn Đình Thơ] Exit with prepaid card 29](#_Toc26719678)

[1.5. [Phùng Thị Trang] Enter with 24-hour ticket 31](#_Toc26719679)

[1.6. [Trần Văn Thông] Exit with 24-hour ticket 32](#_Toc26719680)

[2. Communication diagram 33](#_Toc26719681)

[2.1. [Nguyễn Mạnh Tiến] Enter with one-way ticket 33](#_Toc26719682)

[2.2. [Nguyễn Mạnh Tiến] Exit with one-way ticket 33](#_Toc26719683)

[2.3. [Phạm Hữu Thọ] Enter with prepaid card 34](#_Toc26719684)

[2.4. [Nguyễn Đình Thơ] Exit with prepaid card 34](#_Toc26719685)

[2.5. [Phùng Thị Trang] Enter with 24-hour ticket 35](#_Toc26719686)

[2.6. [Trần Văn Thông] Exit with 24-hour ticket 36](#_Toc26719687)

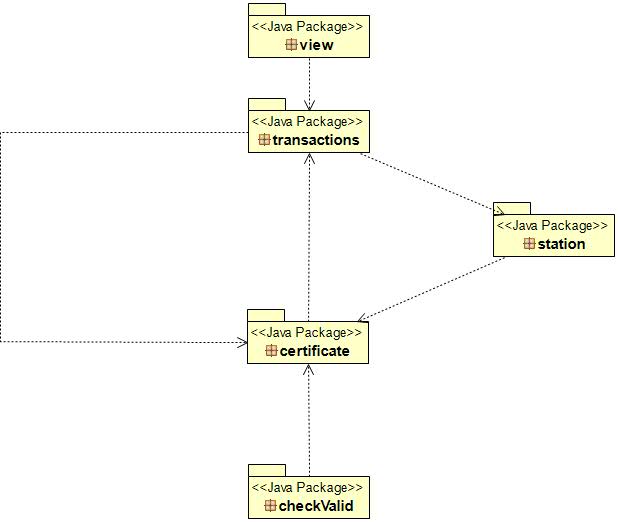
[III. Coupling and Cohesion 36](#_Toc26719688)

[IV. [Phạm Hữu Thọ] SOLID 45](#_Toc26719689)

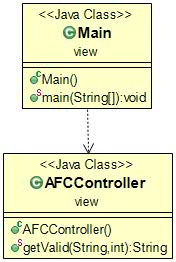
# Static Structure

# Class diagram

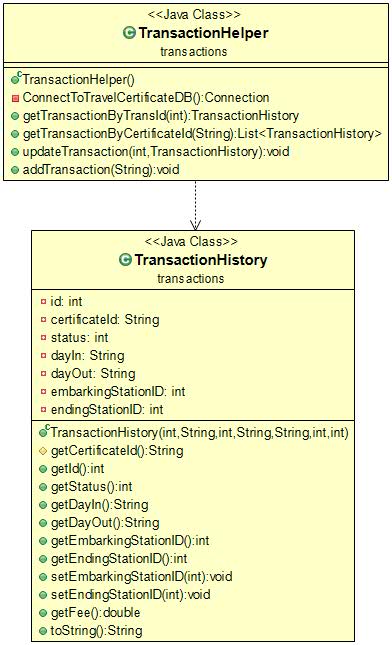
* 1. Package diagram



* 1. Class diagram for package view



* 1. Class diagram for package transactions



* + 1. Class “TransactionHelper”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | constructor() | int | khởi tạo transaction |
| 2 | ConnectToTravelCertificateDB() | String |  |
| 3 | getTransactionByTransId(int) | TransactionHistory | trả về lịch sử giao dịch từ id của giao dịch |
| 4 | getTransactionByCertificateDB(String) | List<TransactionHistory> | trả về lịch sử giao dịch từ DB |
| 5 | updateTransaction(int,TransactionHistory) | null | cập nhật giao dịch |
| 6 | addTransaction(Sting) | null | thêm giao dịch |

1.3.2 Class “TransactionHistory”

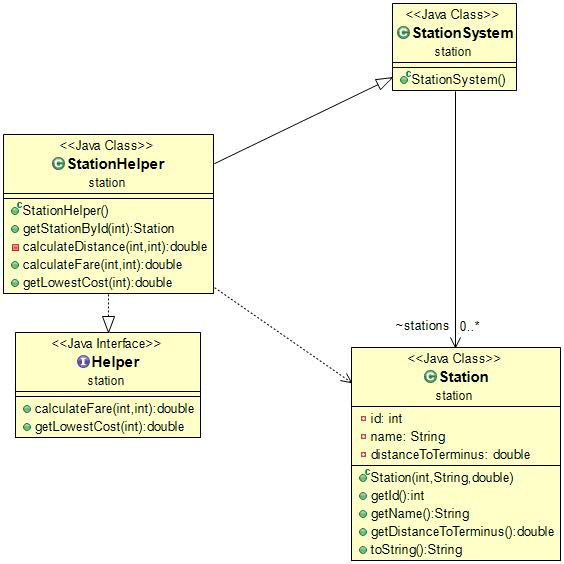
**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | int | 0 | id lịch sử của giao dịch |
| 2 | certificateId | string |  |  |
| 3 | status | int |  |  |
| 4 | dayIn | string |  |  |
| 5 | dayOut | string |  |  |
| 6 | embarkingStationId | int |  |  |
| 7 | endingStationId | int |  |  |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | constructor() | int | khởi tạo lịch sử giao dịch |
| 2 | getCertificateId | String |  |
| 3 | getId | int |  |
| 4 | getStatus | int |  |
| 5 | getDayIn | string |  |
| 6 | getDayOut | string |  |
| 7 | getEmbarkingStationID | int |  |
| 8 | getEndingStationID | int |  |
| 9 | setEmbarkingStationID(int) | null |  |
| 10 | setEndingStationID(int) | null |  |
| 11 | getFee() | double |  |
| 12 | toString() | String |  |

* 1. Class diagram for package station



* + 1. Class “Station”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | int | 0 | id của ga |
| 2 | name | string |  | tên ga |
| 3 | distanceToTerminus | double |  | khoảng cách từ Station tới Terminus |

**Operation**

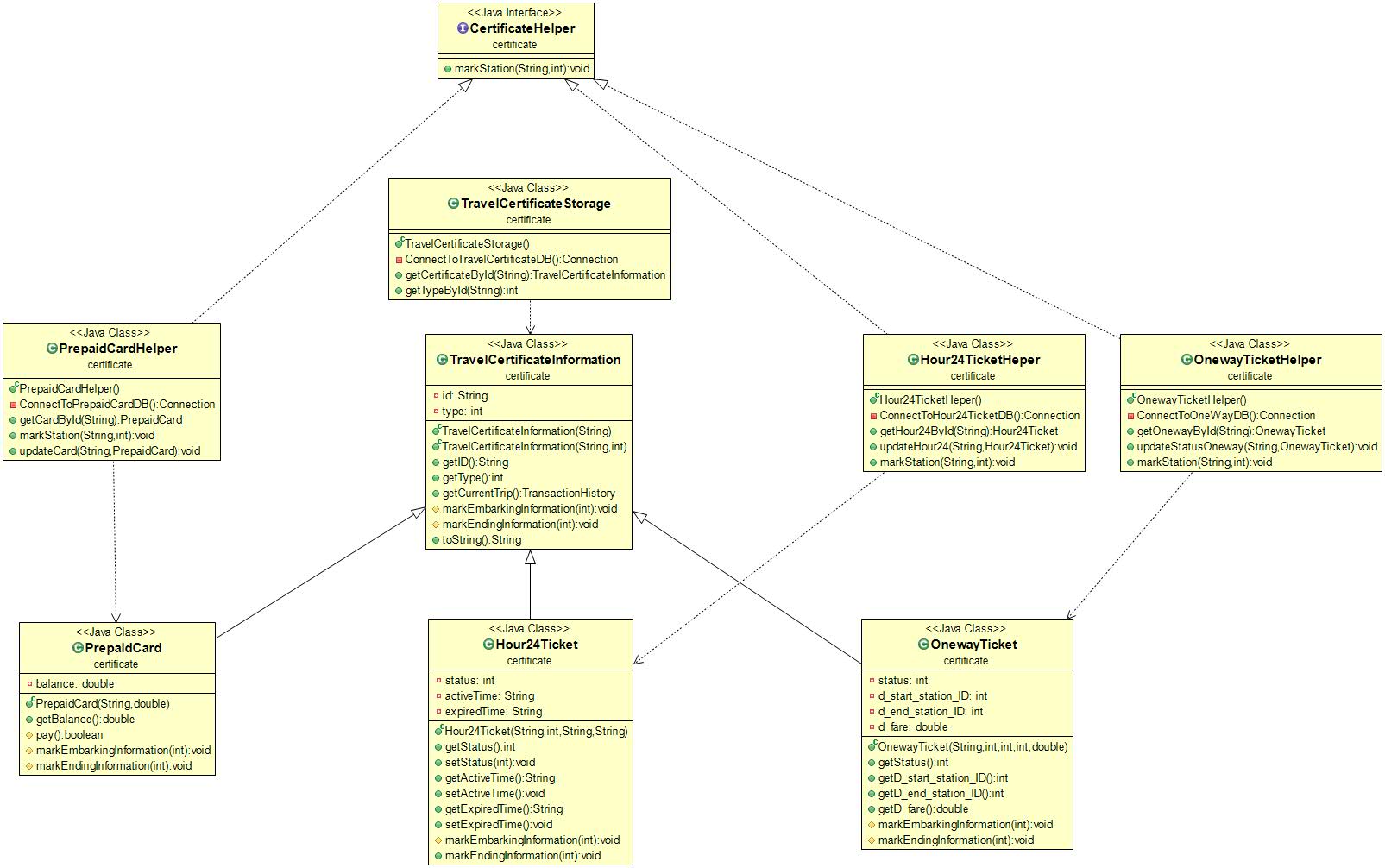
|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getId() | int | trả về ID của Station |
| 2 | getName() | String | trả về Name của Station |
| 3 | getDistanceToTerminus | double | trả về khoảng cách của từ Station tới bến cuối |

* + 1. Class “StationHelper”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getStationById(int) | Station | trả về thông tin ga từ id |
| 2 | caculateFare(int,int) | double | Tính toán phí di chuyển |
| 3 | getLowestCost(int) | double | trả về chi phí thấp nhất |
| 4 | toString() | string |  |

* 1. Class diagram for package certificate



* + 1. Class “TravelCertificateStorage”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getCertificateById(String) | TravelCertificateInformation | trả về thông tin của certificate |
| 2 | getTypeById(String ) | int | trả về type của certificate |

* + 1. Class “TravelCertificateInformation”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | string | 0 |  |
| 2 | type | int | null | Loại vé thẻ  1=OnewayTicket 2=24hourTicket 3=PrepaidCard |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | constructor (String, int) |  | Khởi tạo certificate |
| 2 | getId() | string | trả về id của certificate |
| 3 | getType() | int | trả về type của certificate |
| 4 | getCurrentTrip() | TransactionHistory | trả về thông tin của chuyến đi gần nhất của certificate |
| 5 | markEmbarkingStation(int) | void | lưu lại giá trị của ga vào vào chuyến đi gần nhất của certificate |
| 6 | markEndingStation(int) | void | lưu lại giá trị của ga ra vào chuyến đi gần nhất của certificate |
| 7 | toString() | String | trả về thông tin cơ bản của certificate dưới dạng văn bản |

* + 1. Class “OnewayTicket”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | status | int | 0 | Trạng thái của vé  0=new  1=in station  2=destroyed |
| 2 | d\_start\_station\_ID | int | null | Ga đầu mặc định |
| 3 | d\_end\_station\_ID | int | null | Ga cuối mặc định |
| 4 | d\_fare | double | null | Phí mặc định |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getStatus() | int | Lấy giá trị trạng thái của vé |
| 2 | getD\_start\_station\_ID() | int | Lấy giá trị ID ga đầu mặc định |
| 3 | getD\_end\_station\_ID() | int | Lấy giá trị ID ga cuối mặc định |
| 4 | getD\_fare() | double | Lấy giá trị phí mặc định |
| 5 | markEmbarkingInformation() | void | Lưu thông tin ga vào hợp lệ vào giao dịch |
| 6 | markEndingInformation() | void | Lưu thông tin ga ra hợp lệ vào giao dịch |

*Parameter*:

- stationID: int, ID của một ga

* + 1. Class “OnewayTicketHelper”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getOnewayById() | OnewayTicket | Lấy toàn bộ thông tin vé thông qua ID |
| 2 | updateStatusOneway() | void | Cập nhật status của vé |
| 3 | markStation() | void | Lưu giá trị ga (xác định dựa vào thông tin giao dịch mà họi method lưu ga vào ga ra của OnewayTIcket) |

*Parameter*:

- stationID: int, ID của một ga

- ID: string, ID của vé oneway

* + 1. Class “PrepaidCard”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | balance | double | 0 | số tiền dư trong thẻ |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | constructor (String, double) |  | Khởi tạo thẻ PrepaidCard với id và balance cho trước |
| 2 | getBalance() | double | trả về số tiền dư trong thẻ |
| 3 | pay() | boolean | thực hiện việc trả tiền cho trip tương ứng với thẻ, trả về false nếu việc trả tiền thất bại |
| 4 | markEmbarkingStation(int) | void | lưu lại giá trị của ga vào vào chuyến đi gần nhất của prepaidcard |
| 5 | markEndingStation(int) | void | lưu lại giá trị của ga ra vào chuyến đi gần nhất của prepaidcard nếu thanh toán thành công. |

* + 1. Class “PrepaidCardHelper”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | constructor() |  | Khởi tạo helper |
| 2 | ConnectToPrepaidCardDB() | Connection | trả về connections liên kết với database lưu trữ thông tin về prepaidcard |
| 3 | getCardById (String) | PrepaidCard | trả về prepaid card với Id tương ứng, trả về null nếu không tìm thấy |
| 4 | markStation (int) | void | lưu lại giá trị của ga vào chuyến đi gần nhất của prepaidcard |
| 5 | updateCard (int, PrepaidCard) | void | cập nhật thông tin của thẻ theo giá trị thẻ mới đã được truyền vào. |

* + 1. Class “Hour24Ticket”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | status | int | 0 | trạng thái của vé 24 hour ticket  0 = new  1= pending( chưa hết hạn sử dụng)  3= destroy(hết hạn sử dụng) |
| 2 | activeTime | String | null | Thời gian hoạt động hiện tại của vé |
|  | expiredTime | String | null | Thời gian hết hạn của vé |

**Operation**

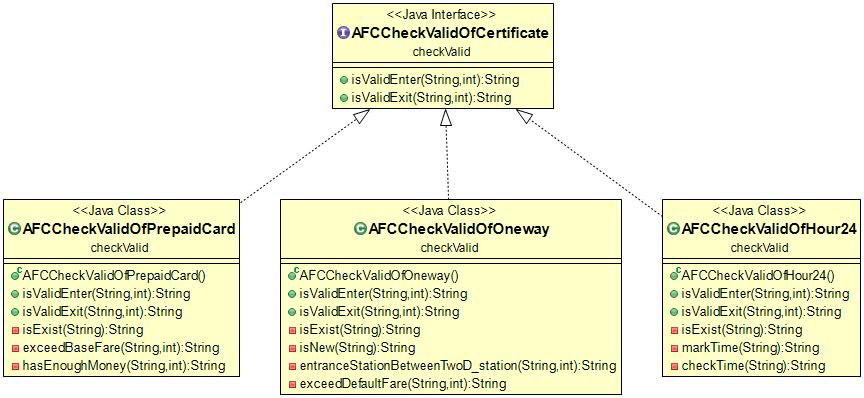
|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getStatus() | int | Lấy trạng thái của vé |
| 2 | setStatus() | void | Đặt lại trạng thái cho vé |
| 3 | getActiveTime() | String | Lấy thời gian hoạt động hiện tại của vé |
| 4 | setActiveTime() | void | Đặt thời gian hoạt động cho vé |
| 5 | getExpiredTime() | String | Lấy thời gian hết hạn của vé |
| 6 | setExpiredTime() | void | Đặt thời gian hết hạn sử dụng của vé |
| 7 | markEmbarkingInformation(int) | void | Lưu lại giá trị của ga vào chuyến đi gần nhất của 24 hour ticket nếu thời gian sử dụng vẫn còn hợp lệ. |
| 7 | markEndingStation(int) | void | Lưu lại giá trị của ga ra vào chuyến đi gần nhất của 24 hour ticket . |

* + 1. Class “Hour24TicketHeper”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getHour24ById(String) | Hour24Ticket | Lấy toàn bộ thông tin vé thông qua ID |
| 2 | updateHour24(String, Hour24Ticket) | void | Cập nhật status, activeTime, expiredTime của vé |
| 3 | markStation(String, int) | void | Lưu giá trị ga (xác định dựa vào thông tin giao dịch mà họi method lưu ga vào ga ra của Hour24Ticket) |

* 1. Class diagram for package checkValid



* + 1. Class “AFCCheckValidOfOneway”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | isValidEnter (String,  int) | String | trả về null nếu oneway đủ điều kiện qua cổng ga vào tương ứng, không thì trả về thông tin lỗi. |
| 2 | isValidExit (String, int) | String | trả về null nếu oneway đủ điều kiện qua cổng ga ra tương ứng, không thì trả về thông tin lỗi. |
| 3 | isExist(String) | String | Kiểm tra xem oneway với id tương ứng có tồn tại không, không thì trả về thông tin lỗi. |
| 4 | isNew(String) | String | Kiểm tra xem oneway với id tương ứng đã sử dụng chưa, nếu có thì trả về thông tin thông báo. |
| 5 | entranceStationBetweenTwoD\_station(String, int) | String | Kiểm tra xem ga vào có hợp lệ (nằm giữa hai ga được ấn định trong vé), nếu không thì thông báo lỗi. |
| 6 | String exceedDefaultFare(String, int) | String | Kiểm tra xem với ga ra đó, phí di chuyển có vượt quá phí mặc định trong vé hay không, nếu có thì thông báo lỗi. |

*Parameter*:

- stationID: int, ID của một ga

- ID: String, ID của vé oneway

* + 1. Class “AFCCheckValidOfHour24”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | constructor() |  | Khởi tạo helper |
| 2 | isValidEnter (String,  int) | String | trả về null nếu 24 hour ticket đủ điều kiện qua cổng ga vào tương ứng, không thì trả về thông tin lỗi. |
| 3 | isValidExit (String, int) | String | trả về null nếu 24 hour ticket đủ điều kiện qua cổng ga ra tương ứng, không thì trả về thông tin lỗi. |
| 4 | isExist(String) | String | Kiểm tra xem 24 hour ticket với id tương ứng có tồn tại không, không thì trả về thông tin lỗi. |
| 5 | markTime (String) | String | Kiểm tra trạng thái của vé và đặt activeTime tương ứng, trường hợp là vé đã hết hạn thì trả về thông tin lỗi. |
| 6 | checkTime(String) | String | kiểm tra xem 24 hour ticket tương ứng có còn thời hạn sử dụng không, không thì trả về thông tin lỗi. |

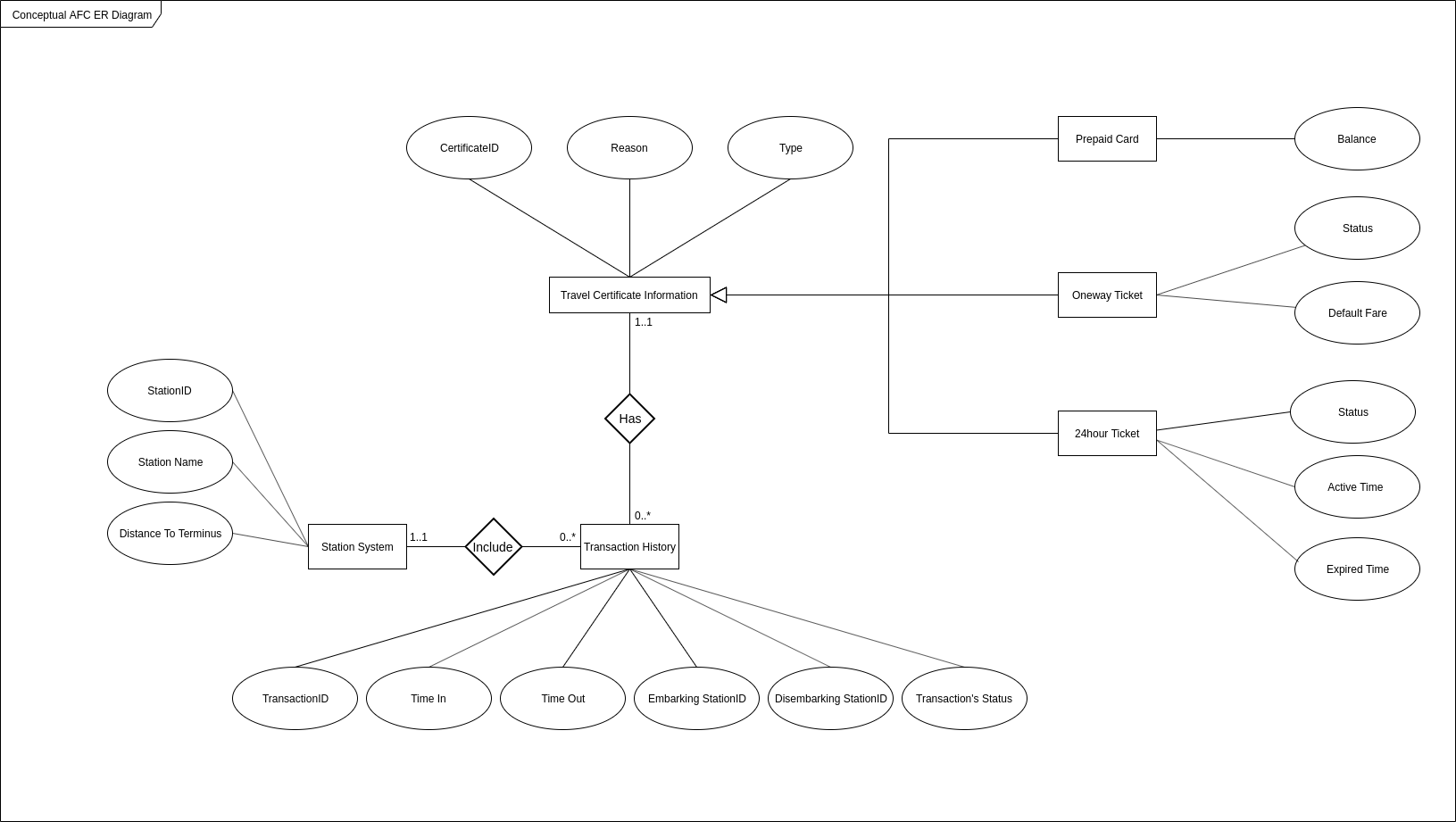
* + 1. Class “AFCCheckValidOfPrepaidCard”

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | constructor() |  | Khởi tạo helper |
| 2 | isValidEnter (String,  int) | String | trả về null nếu prepaid card đủ điều kiện qua cổng ga vào tương ứng, không thì trả về thông tin lỗi. |
| 3 | isValidExit (String, int) | String | trả về null nếu prepaid card đủ điều kiện qua cổng ga ra tương ứng, không thì trả về thông tin lỗi. |
| 4 | isExist(String) | String | Kiểm tra xem prepaid card với id tương ứng có tồn tại không, không thì trả về thông tin lỗi. |
| 5 | exceedBaseFare (String,  int) | String | kiểm tra xem tiền dư của prepaid card tương ứng quá hơn mức thấp nhất cho phép không và có đủ điều kiện để trả cho một chuyến đi ngắn nhất không, không thi trả về thông tin lỗi. |
| 6 | hasEnoughMoney(String, int) | String | kiểm tra xem prepaidcard tương ứng có đủ tiền để trả cho chuyến đi không, không thì trả về thông tin lỗi. |

# Data Modeling

# Conceptual Data Model



* Description of entities and relationships:

|  |  |  |
| --- | --- | --- |
| No. | Entity | Relationships |
| 1 | Travel Certificate Information |  |
| 2 | Oneway Ticket | Kế thừa Travel Certificate Information |
| 3 | Prepaid Card | Kế thừa Travel Certificate Information |
| 4 | 24hour Ticket | Kế thừa Travel Certificate Information |
| 5 | Transaction History | Một Travel Certificate có thể có từ một cho đến nhiều giao dịch hoặc chưa phát sinh giao dịch nào. |
| 6 | Station System | Một ga có thể được bao gồm trong một đến nhiều giao dịch hoặc không có trong giao dịch nào |

## Logical Data Model

* Database Management System (DBMS): Relational DBMS (RDBMS)
  + Cơ sở dữ liệu quan hệ (relational database) là một cơ sở dữ liệu số dựa trên mô hình quan hệ dữ liệu. Một hệ thống phần mềm sử dụng để duy trì cơ sở dữ liệu quan hệ là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Nhiều hệ thống cơ sở dữ liệu quan hệ có tùy chọn sử dụng SQL (Ngôn ngữ truy vấn có cấu trúc) tiêu chuẩn để truy vấn và duy trì cơ sở dữ liệu.
* Show the process to design database from E-R diagram
  + Đặc tả đầy đủ các thực thể và mối quan hệ giữa chúng
  + Liệt kê hết các thuộc tính của từng thực thể
  + Khóa chính, khóa ngoại (khóa xác định quan hệ với các thực thể khác) của từng thực thể được đặc tả
* Show the diagram of DB design result

A screenshot of a map

Description automatically generated

## Database Design

* Nguyễn Mạnh Tiến

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TravelCertificateInformation | | | |  |  |  |  |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | certificateID | VARCHAR | NULL | x |  |
| 2 |  |  | reason | VARCHAR | NULL |  |  |
| 3 |  |  | type | INT | NULL | x |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| OnewayTicket | | | |  |  |  |  |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x | x | onewayID | VARCHAR | NULL | x |  |
| 2 |  |  | status | INT | 0 | x | 0=new  1=in station  2=destroyed |
| 3 |  |  | defaultFare | FLOAT | NULL | x |  |

* Phạm Hữu Thọ

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PrepaidCard | | | |  |  |  |  |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x | x | cardID | VARCHAR | NULL | x |  |
| 2 |  |  | balanceID | FLOAT | 5.0 | x | min=2.5 |

* Phùng Thị Trang

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 24hourTicket | | | |  |  |  |  |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x | x | 24hourID | VARCHAR | NULL | x |  |
| 2 |  |  | status | INT | 0 | x | 0=new  1=pending  3=destroyed |
| 3 |  |  | activeTime | String | NULL | x |  |
| 4 |  |  | expiredTime | String | NULL | x |  |

* Trần Văn Thông

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TransactionHistory | | | |  |  |  |  |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | transactionID | INT | NULL | x |  |
| 2 |  |  | timeIn | TIME | NULL | x |  |
| 3 |  |  | timeOut | TIME | NULL | x |  |
| 4 |  | x | certificateID | VARCHAR | NULL | x |  |
| 5 |  | x | embarkingStationID | INT | NULL | x |  |
| 6 |  | x | endingStationID | INT | NULL | x |  |
| 7 |  |  | status | INT | NULL | x |  |

* Nguyễn Đình Thơ

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| StationSystem | | | |  |  |  |  |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | StationID | INT | NULL | x |  |
| 2 |  |  | name | VARCHAR | NULL | x |  |
| 3 |  |  | distanceToTerminus | FLOAT | NULL | x |  |

# UI Design

* 1. Main screen

These are stations in the line M14 of Paris:

1. Saint-Lazare

2. Madeleine

3. Pyramides

4. Chatelet

5. Gare de Lyon

6. Bercy

7. Cour Saint-Emilion

8. Bibliotheque Francois Mitterrand

9. Olympiades

Station ID:

* 1. Chọn Card hay Ticket

What are you using?:

1. Card

2. Ticket

* 1. Nhập Barcode

Please enter your barcode:

* 1. Thông báo lỗi

ERRRRRRR!

[Nội dung lỗi]

* Ví dụ:

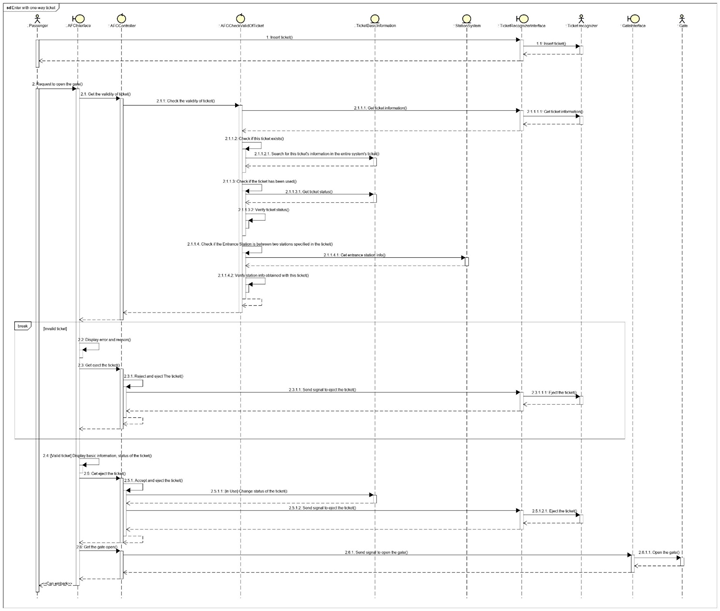
ERRRRRRR!

Ticket has been used. Please buy a new one.

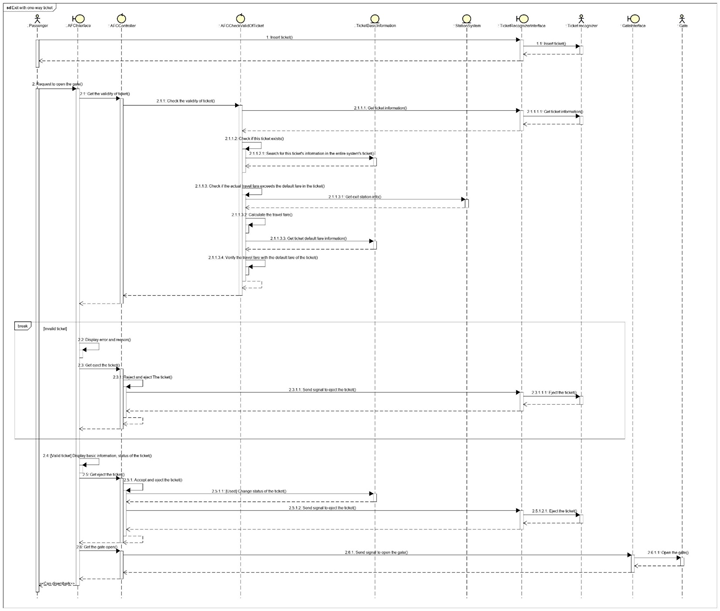
# Dynamic Structure

# Sequence diagram

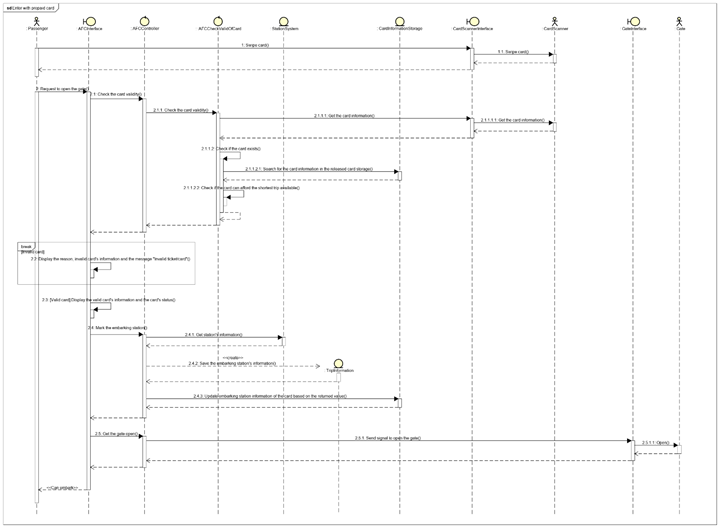
# [Nguyễn Mạnh Tiến] Enter with one-way ticket



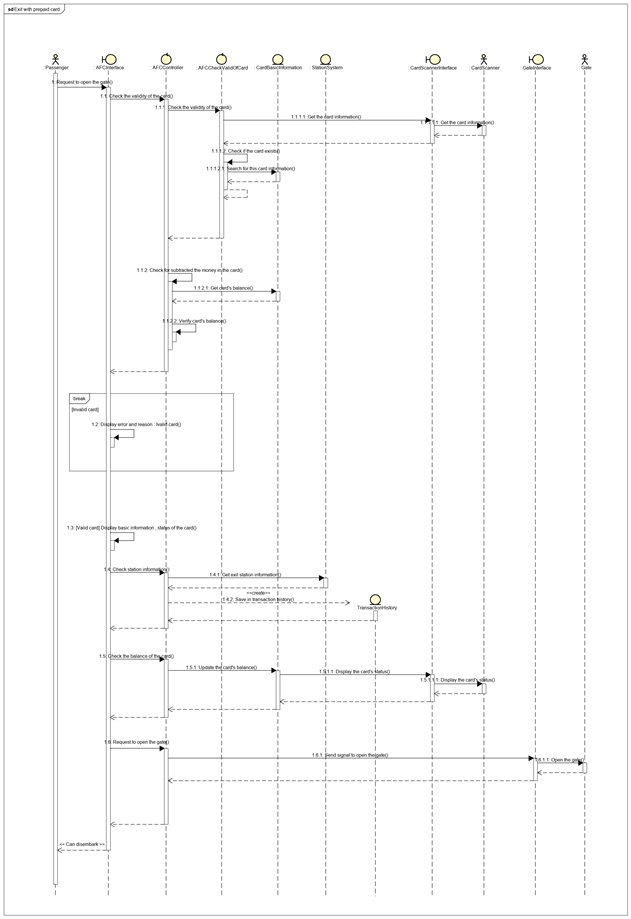
# [Nguyễn Mạnh Tiến] Exit with one-way ticket



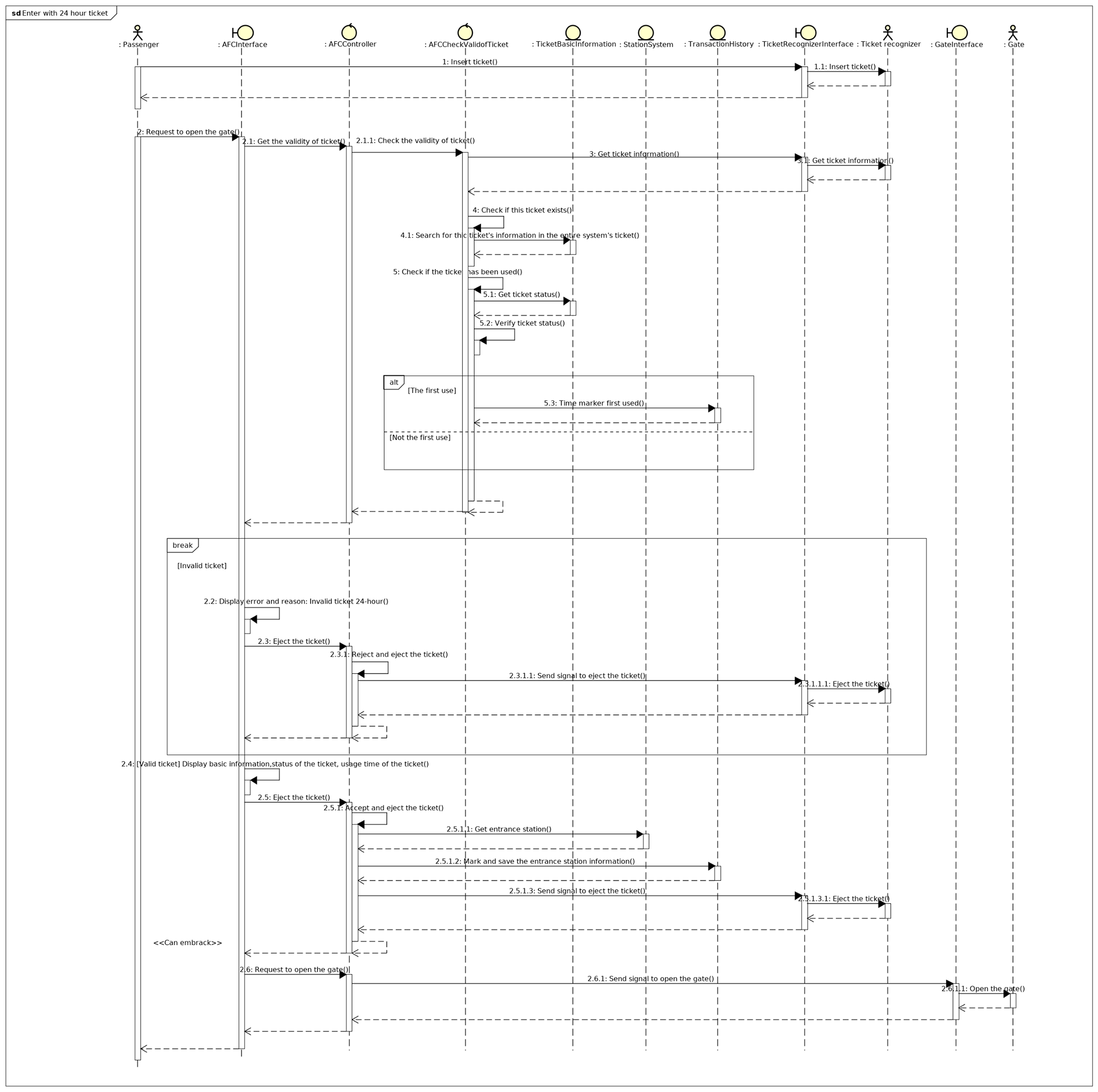
# [Phạm Hữu Thọ] Enter with prepaid card



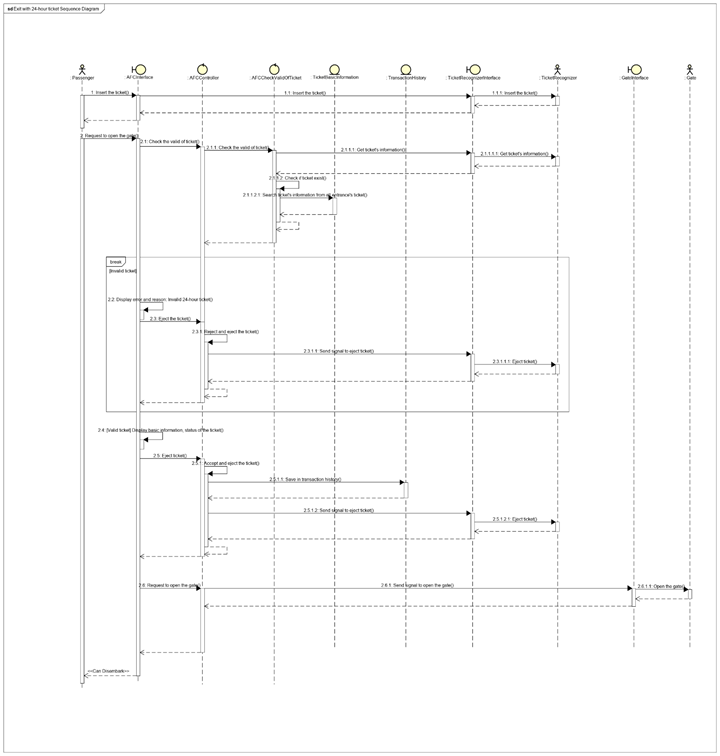
# [Nguyễn Đình Thơ] Exit with prepaid card



# [Phùng Thị Trang] Enter with 24-hour ticket

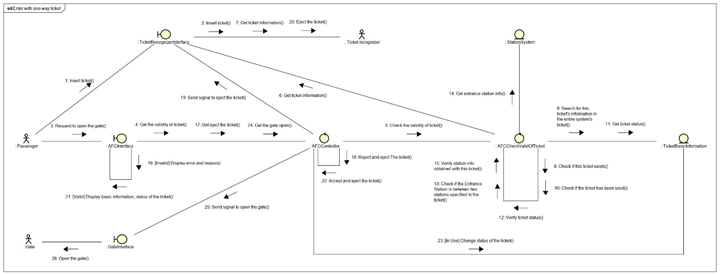


# [Trần Văn Thông] Exit with 24-hour ticket

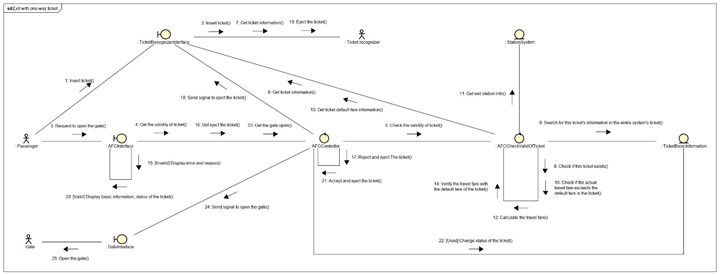


# Communication diagram

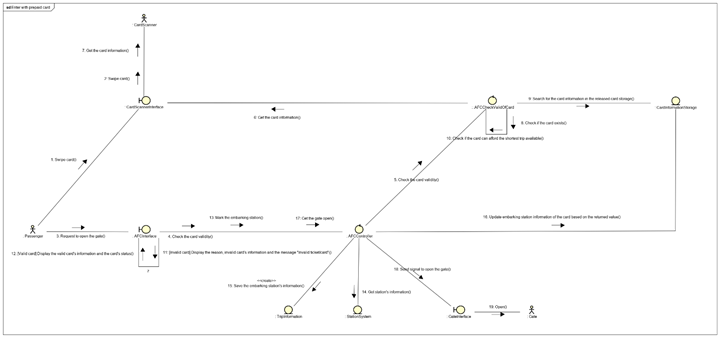
# [Nguyễn Mạnh Tiến] Enter with one-way ticket



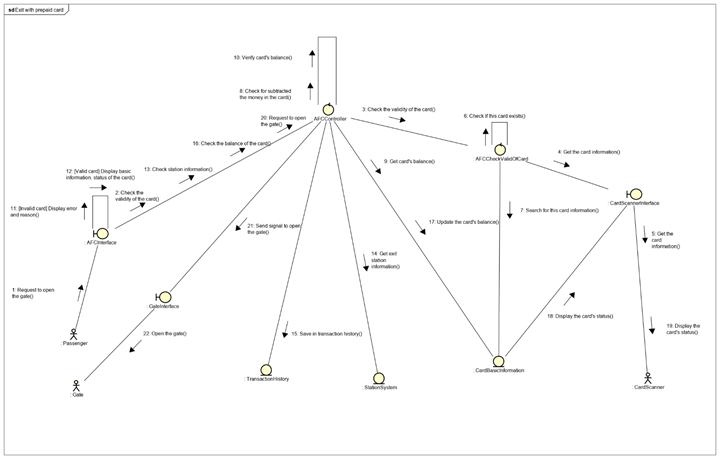
# [Nguyễn Mạnh Tiến] Exit with one-way ticket



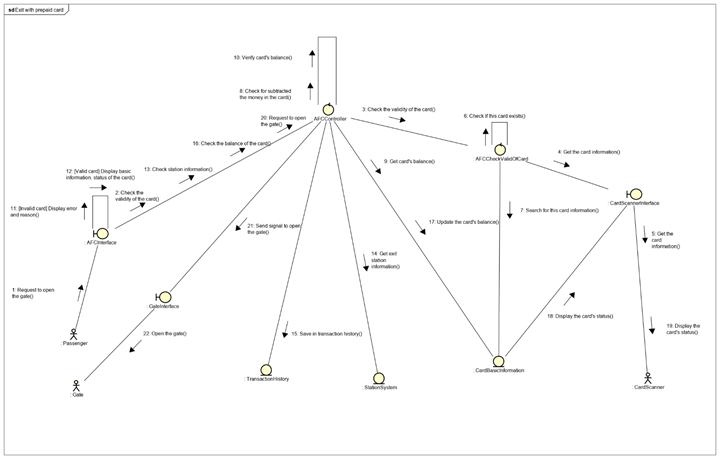
# [Phạm Hữu Thọ] Enter with prepaid card



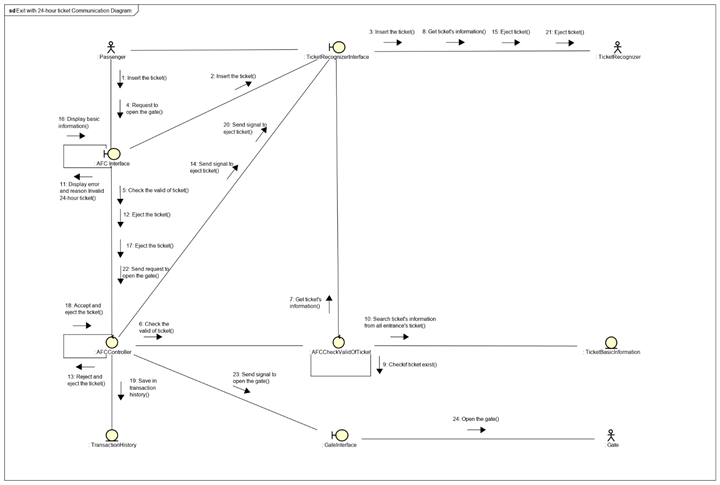
# [Nguyễn Đình Thơ] Exit with prepaid card



# [Phùng Thị Trang] Enter with 24-hour ticket



# [Trần Văn Thông] Exit with 24-hour ticket



# Coupling and Cohesion

* **Mức hiện tại:**
  + Coupling: data coupling
  + Cohesion: sequential cohesion

1. [Phạm Hữu Thọ] Base Project and PrepaidCard
   1. Phạm vi sửa:

· PrepaidCard

· PrepaidCardHelper

· TravelCertificateStorage

· AFCCheckValidOfCeritificate

· AFCCheckValidOfPrepaidCard

· (package)view

· (package)transactions

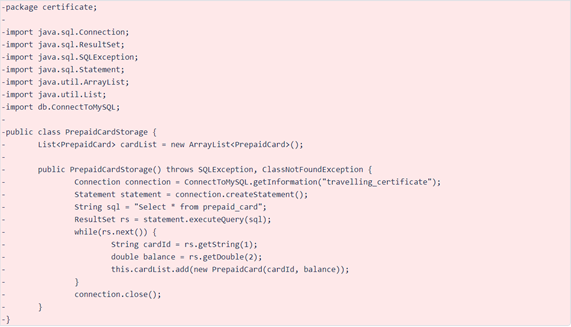
· (package)stations

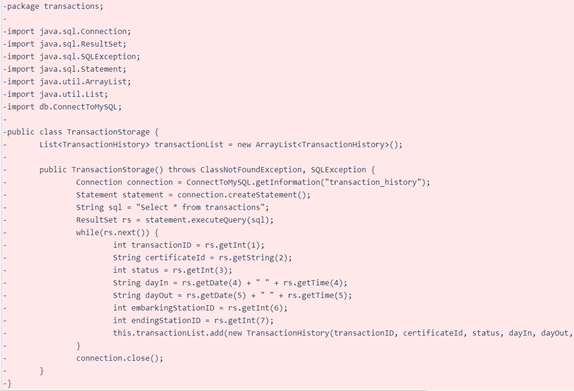
⇒ Nhận xét sơ qua: Chủ yếu là sửa lỗi liên quan tới Coupling. Lỗi liên quan tới Cohesion gần như không hề có. Sau khi sửa thì khả năng mở rộng phần mềm (thêm chức năng, loại thẻ) và khả năng phát hiện lỗi đã trở nên dễ dàng hơn nhiều.

* 1. Các thay đổi cụ thể

**Link pull:**<https://bitbucket.org/tientt_glose/isd.vn.20191-09/pull-requests/102/week10-programming/diff>

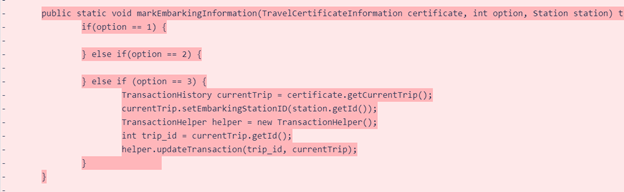
* + 1. Xóa toàn bộ PrepaidCardStorage và TransactionStorage

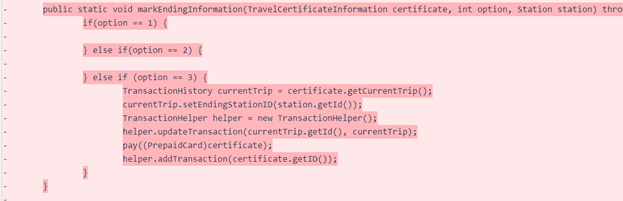


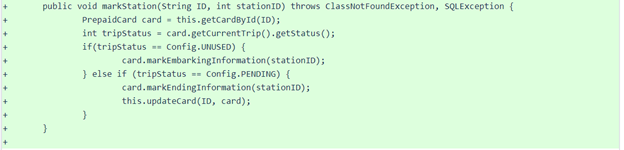


**Mục đích**: Khắc phục lỗi “i” trong SOLID, cung cấp quá nhiều thông tin không cần thiết.

* + 1. Xóa hàm markEmbarkingStation() và markEndingStation() trong AFCController và thêm hàm markStation() trong class PrepaidCardHelper

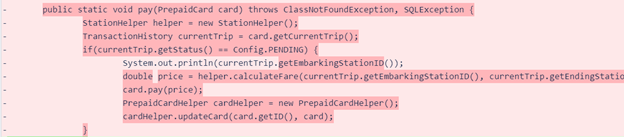


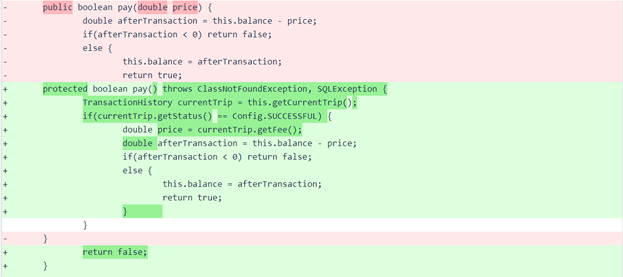




**Mục đích:** Sửa lỗi Common Coupling và Temporal Cohesion. Do hai hàm đó của AFCController dễ gây hiểu lầm với markStation () trong PrepaidCard, và hai hàm đó không có giá trị kế thừa.

* + 1. Xóa hàm pay() trong AFCController và update hàm pay() bên trong PrepaidCard.



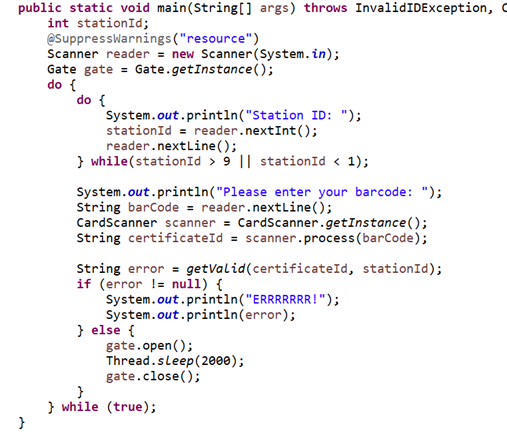


**Mục đích:** Sửa lỗi Common Coupling và Temporal Cohesion. Do hàm pay() của AFCController dễ gây hiểu lầm với pay() trong PrepaidCard, và nó không có giá trị kế thừa.

* + 1. Thay đổi tất cả nhưng giá trị parameter là object thành ID của tất cả các class trong phạm vi sửa (bên trên cùng) và áp dụng lại cách implement. Ví dụ trong interface AFCCheckValidOfCertificate:



Hàm main từ việc dùng nhiều if else sang thành:



**Mục đích:** Tránh lỗi Common Coupling. Gọi nguyên cả object (dù không cần thiết) và tránh được lỗi Control Coupling.

1. [Nguyễn Mạnh Tiến] OnewayTicket
   1. Phạm vi sửa:

* OnewayTicket
* OnewayTicketHelper
* OnewayTicketStorage 🡺 Xóa bỏ
* AFCCheckValidOf OnewayTicket
* Nhận xét chung: Vì qua bàn luận trước khi code, nên toàn bộ chương trình được code theo một khung chung, các lỗi có thể tương tự và cũng sẽ được khắc phục dựa trên kiến thức học được qua từng bài trên lớp. Sửa đổi chủ yếu là sửa lỗi liên quan tới Coupling, lỗi liên quan tới Cohesion gần như không hề có.
  1. Các thay đổi cụ thể:

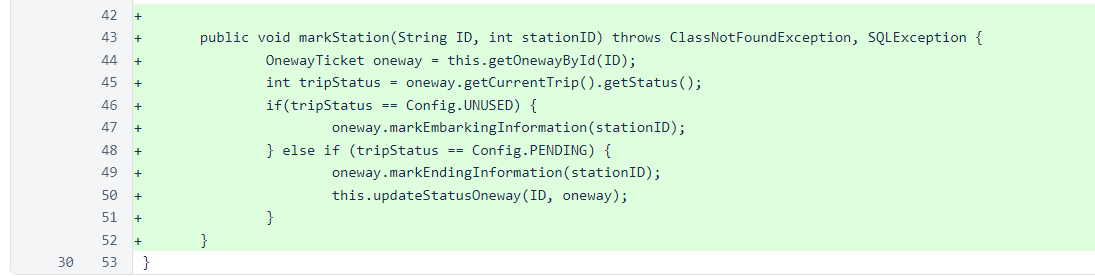
**Link pull:** <https://bitbucket.org/tientt_glose/isd.vn.20191-09/pull-requests/103/week10_handle-coupling-cohesion/diff>

* + 1. Xóa bỏ: OnewayTicketStorage



**Mục đích:** Xử lý Common Coupling, cung cấp quá nhiều thông tin không cần thiết, nhiều module chia sẻ chung 1 dữ liệu, cùng trực tiếp sử dụng dư liệu; gây bất đồn bộ khi cập nhật cả <List> lấy ra và trong dữ liệu data base.

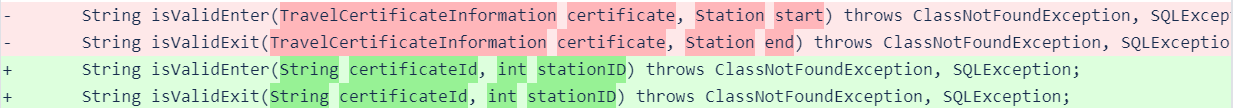
* + 1. Thêm hàm markStation() trong class OnewayTicketHelper (dựa trên kết quả xóa hàm markEmbarkingStation() và markEndingStation() trong AFCController của thành viên Phạm Hữu Thọ)

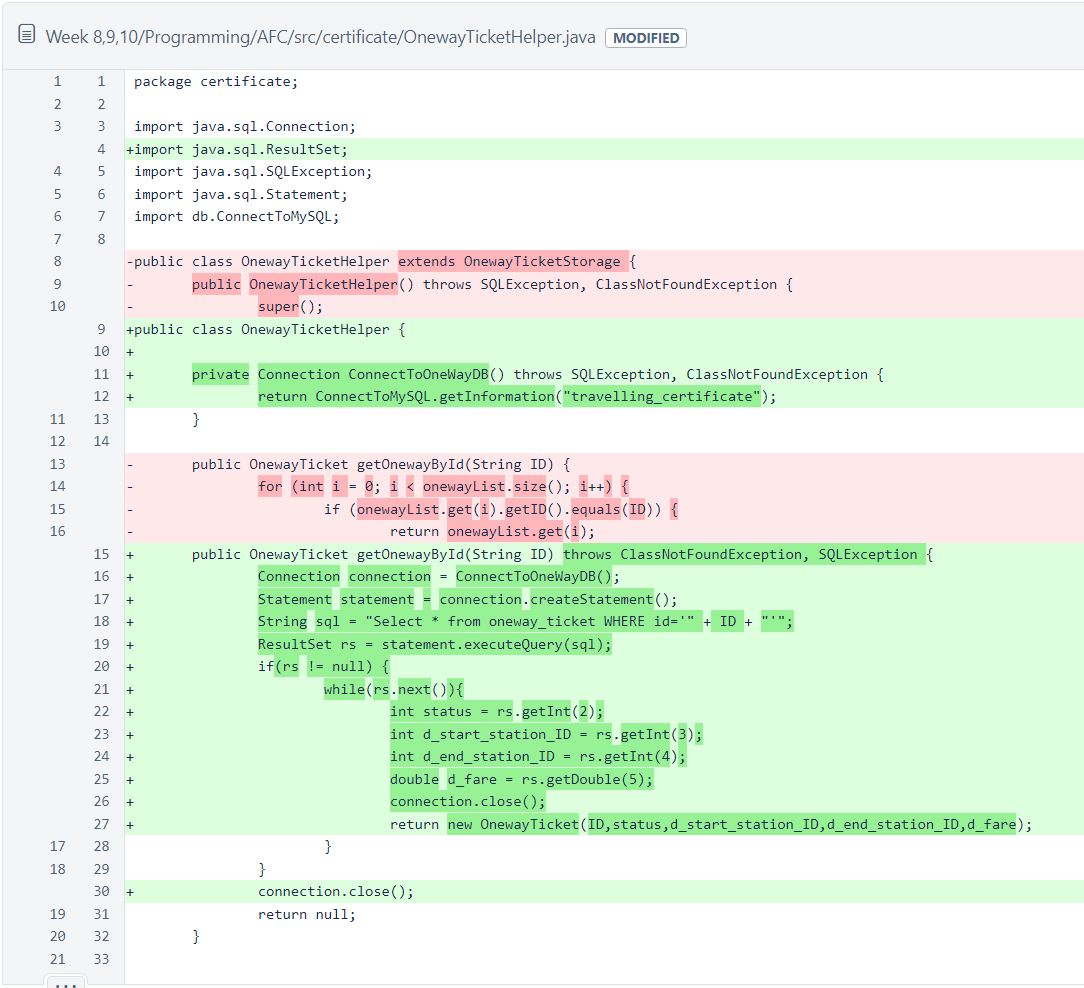


**Mục đích:** Sửa lỗi Common Coupling và Temporal Cohesion. Do hai hàm đó của AFCController dễ gây hiểu lầm với markStation () trong OnewayTicket, và hai hàm đó không có giá trị kế thừa.

* + 1. Thay đổi tất cả nhưng giá trị parameter là object thành ID của tất cả các class trong phạm vi sửa (bên trên cùng) và áp dụng lại cách implement

Ví dụ:





**Mục đích:** Không truyền quá nhiều thông tin vào một method. Tránh việc Common Coupling khi gọi nguyên cả object dù không cần thiết.

1. [Phùng Thị Trang] Hour 24 Ticket
   1. Phạm vi sửa**:**

* Hour24Ticket
* Hour24TicketHelper
* Hour24Storage
* AFCCheckValidOfHour24Ticket
* Nhận xét chung: Vì qua bàn luận trước khi code, nên toàn bộ chương trình được code theo một khung chung, các lỗi có thể tương tự và cũng sẽ được khắc phục dựa trên kiến thức học được qua từng bài trên lớp. Sửa đổi chủ yếu là sửa lỗi liên quan tới Coupling, lỗi liên quan tới Cohesion gần như không hề có.
  1. Các thay đổi cụ thể:

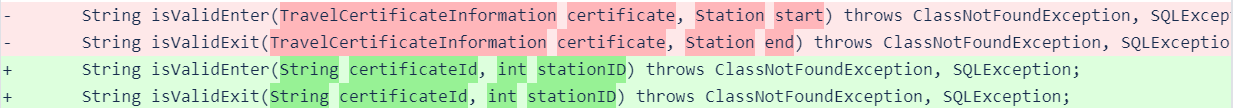
**Mục đích:** Xử lý Common Coupling, cung cấp quá nhiều thông tin không cần thiết, nhiều module chia sẻ chung 1 dữ liệu, cùng trực tiếp sử dụng dư liệu; gây bất đồng bộ khi cập nhật cả <List> lấy ra và trong dữ liệu data base.

* + 1. Thêm hàm markStation() trong class Hour24TicketHelper (dựa trên kết quả xóa hàm markEmbarkingStation() và markEndingStation() trong AFCController của thành viên Phạm Hữu Thọ)

**Mục đích:** Sửa lỗi Common Coupling và Temporal Cohesion. Do hai hàm đó của AFCController dễ gây hiểu lầm với markStation () trong OnewayTicket, và hai hàm đó không có giá trị kế thừa.

* + 1. Thay đổi tất cả nhưng giá trị parameter là object thành ID của tất cả các class trong phạm vi sửa (bên trên cùng) và áp dụng lại cách implement

Ví dụ:



**Mục đích:** Không truyền quá nhiều thông tin vào một method. Tránh việc Common Coupling khi gọi nguyên cả object dù không cần thiết.

# [Phạm Hữu Thọ] SOLID

5 tính chất của SOLID trong project:

**· S (Single responsibility principle)**

Nhận xét: Về cơ bản thì nhóm đã thiết kế theo nguyên lý S này ngay từ đầu, phân chia class và nhiệm vụ của từng class đã rất rõ ràng. Mỗi function của từng class cũng không có hơn 1 nhiệm vụ.

Ưu điểm: Vì đã theo nguyên tắc S ngay từ đầu, mỗi khi thành viên trong nhóm gặp khó khăn trong việc code, người khác có thể dễ dàng hỗ trợ cùng vì có thể xác định được ngay phần lỗi ở đâu, chỗ nào cần sửa. Và khi một phần của hệ thống bị lỗi (giả sử như vé 24h bị lỗi thì cũng không ảnh hưởng việc chương trình chạy cho những vé khác).

**· O (Open/closed principle)**

Nhận xét: Chương trình không có element hay function nào public thừa. Khi cần phát triển thêm tính năng mới cho class đang có sẵn thì không cần sửa code của class cũ mà tạo class mới.

**· L (Liskov Substitution Principle)**

Nhận xét: Nguyên lý L được thể hiện rõ qua việc phát triển 3 loại vé. 3 loại vé cùng implement 1 interface nhưng chúng không kế thừa nhau mà kế thừa qua một abstract class cha là TravelCertificateInformation (chứa tất cả những yếu tố cần thiết mà một vé thẻ luôn có).

**· I (Interface Segregation)**

Nhật xét: vì độ lớn của chương trình không lớn nên không có class nào phải implement nhiều hơn 1 interface nên ưu điểm của nó chưa được thể hiện rõ rệt qua chương trình này.

Ưu điểm: Vì làm theo nguyên lý này, việc implement và quản lý code rất dễ dàng. Interface nhỏ tương ứng với việc không dư thừa chức năng và đỡ tốn công code.

**· D (Dependency Inversion)**

Nhận xét: Tương tự với nguyên lý I, vì độ lớn của chương trình không lớn nên nguyên lý này chưa được thể hiện rõ rệt. Có 1 điểm rõ nhất là trong file AFCController, để cho chương trình chạy, nó không quan tâm là loại vé/thẻ nào đã được nhập vào, nó chỉ nhìn vào interface, interface cho nó hàm gì thì nó cần gọi hàm getValid là chạy được.

Ưu điểm: Việc sửa đổi và test code rất dễ dàng, blackbox test chỉ cần nhìn vào danh sách function trong interface.

1. Phân tích các ưu điểm của những việc này đối với những yêu cầu mới cô đã cung cấp

Nhận xét sơ qua: Nhóm đã bám theo sơ đồ thiết kế chặt chẽ và đã cố gắng code theo SOLID ngay từ đầu nên căn bản code gần như không phải sửa mấy.

1. Thêm các loại thẻ vé
   1. Thẻ tháng:

Bước thêm:

· Trong package certificate

a. Thêm class MonthlyCard, extends class Hour24Ticket, override hàm getStatus() để phù hợp với cách kiểm tra thẻ còn hay hết hạn.

b. Thêm helper MonthlyCardHelper implements interface Helper.

· Trong package checkValid, thêm class AFCCheckValidOfMonthlyCard, và implement interface AFCCheckValidOfCertificate.

Giải thích: Bản chất của thẻ tháng không khác gì với vé 24h về mặt properties nhưng lại khá khác nhau về cách kiểm tra tính hợp lệ, nên ta sẽ kế thừa cái có sẵn là class Hour24Ticket và tạo mới các class còn lại cần thiết.

b. Vé cuối tuần:

Bước thêm:

· Trong package certificate

a. Thêm class WeeklyTicket, extends class Hour24Ticket, override hàm getStatus() để phù hợp với cách kiểm tra vé còn hay hết hạn.

b. Thêm helper WeeklyTicket Helper implements interface Helper.

· Trong package checkValid, thêm class AFCCheckValidOfWeeklyTicket, và implement interface AFCCheckValidOfCertificate.

Giải thích: Một lần nữa, bản chất của Vé cuối tuần không khác gì với vé 24h mặt properties nhưng lại khá khác nhau về cách kiểm tra tính hợp lệ, nên ta sẽ kế thừa cái có sẵn là class Hour24Ticket và tạo mới các class còn lại cần thiết.

1. Thay đổi cách tính phí:

a. Thay đổi các mốc, hạn mức giá trị:

Cách làm: Vào file Config trong class certificate và thay đổi thông số tương ứng:

***PREPAID\_MINIMUM\_BALANCE***

***BASED\_FARE***

***BASED\_DISTANCE***

***DEFAULT\_ADDITION\_DISTANCE***

***ADDITIONAL\_FARE***

b. Không tính phí đi lại trên khoảng cách, mà tính dựa trên zone của hai nhà ga:

Bước thêm:

· Trong package station

a. Thêm class StationHelper2, extends class StationSystem và implements interface SystemHelper. (Khác nhau duy nhất ở hàm này và hàm calculateDistance vì cơ chế tính tiền khác nhau)

· Trong package transactions

a. Người mua chương trình muốn dùng cách tính tiền nào thì thay bằng class tương tự, thay thành helper tương ứng trong class TransactionHistory của package transactions

StationHelper helper = **new** StationHelper();

1. Có nhiều tuyến tàu (line). Các tuyến này có thể cắt nhau.

a. Tạo tuyến tàu (StationSystem1, StationSystem2, …) trong package stations.

· Vì StationSystem hiện tại giống như là một tuyến tàu vip (đi qua mọi trạm), các tuyến khác thì cũng tương tự, chẳng qua là ít số lượng ga đi, nên là hàm constructor của các hàm system phải trên phải sửa sao cho đúng với loại ga đi qua.

· Các hàm trong class sẽ là: kiểm tra xem tuyến có đi qua ga X không.

· Sau đó tự tạo thêm class Helper implements interface Helper cho mỗi tuyến đó. (Mỗi tuyến có cách tính tiền khác nhau)

b. Các tuyến có thể cắt nhau.

· Các tuyến có chung điểm dừng thì có thể cắt nhau.

· Trong các helper giờ phải có thêm function kiểm tra có qua trạm X không.

· AFCController giờ sẽ là fat data aggregation, tụ hội lại chỗ helper trên và cho người dùng chọn một trong các helper đó để đi. (như là chọn tuyến để mà đi).