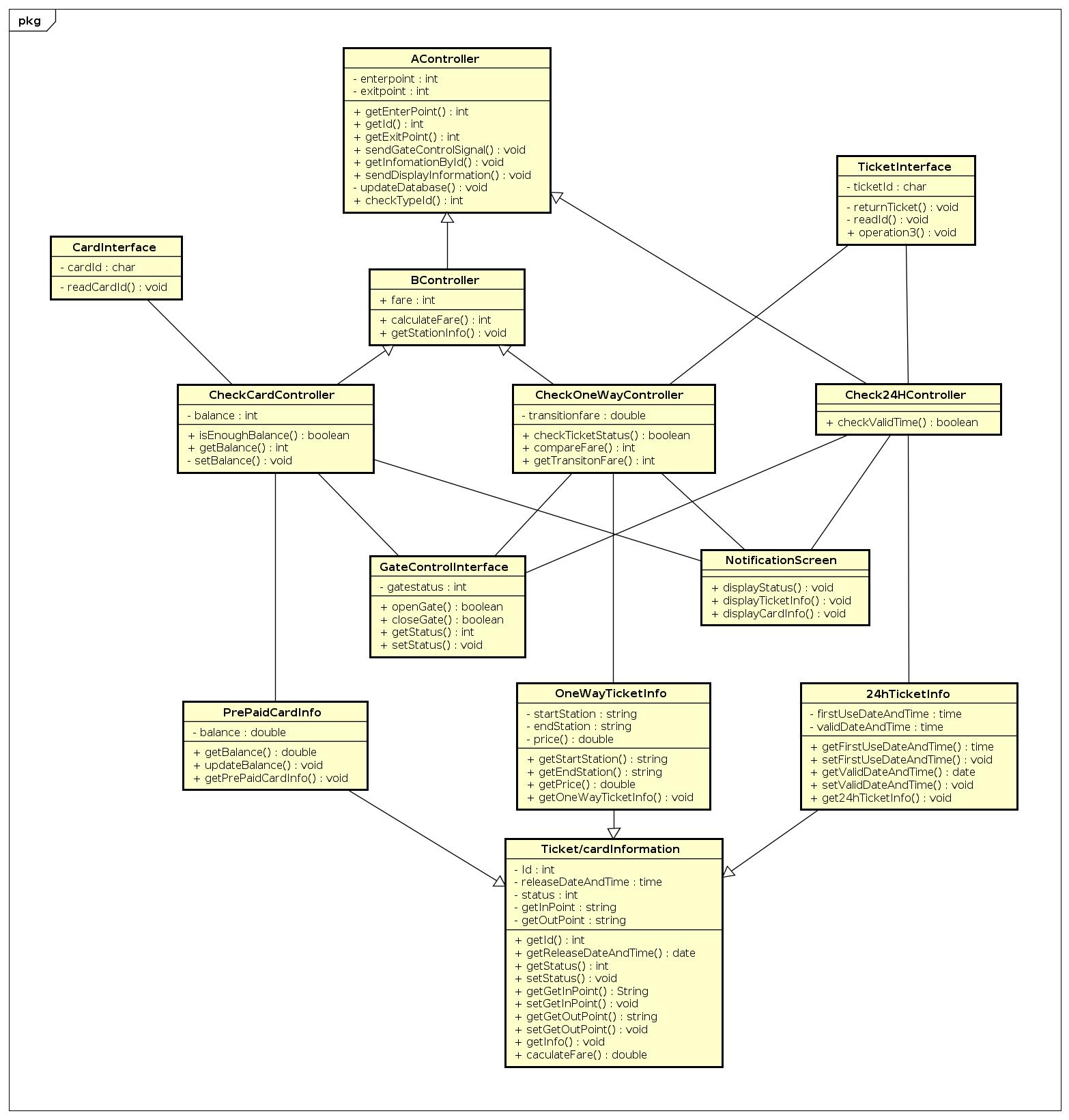
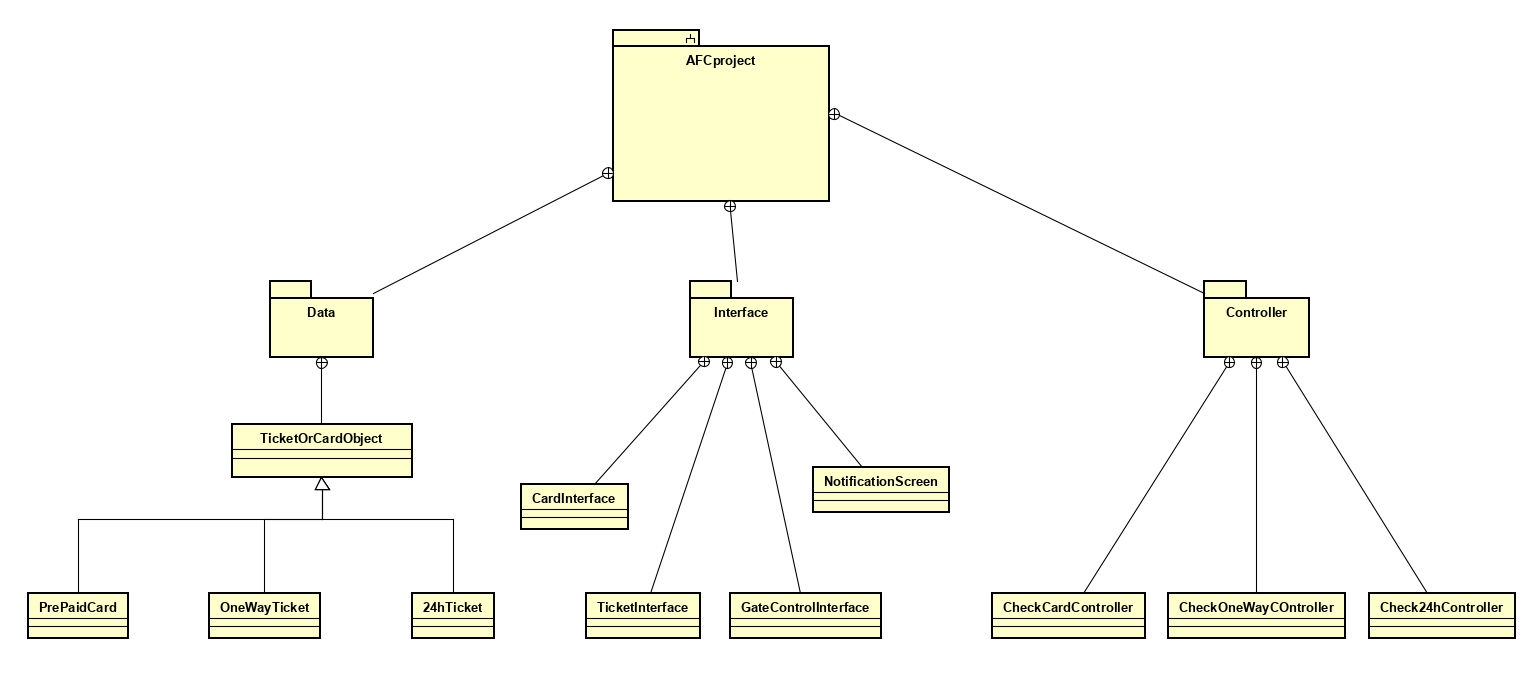
# Class diagram

Hình ảnh biểu đồ lớp



Hình ảnh biểu đồ package



# Class design

## Class “AController”

Hình ảnh class "AController"

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | enterpoint | int |  | Chứa giá trị id của nhà ga mà hành khách đi vào. |
| 2 | exitpoint | int |  | Chứa giá trị id của nhà ga mà hành khách đi ra. |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getEnterPoint() | int | Lấy thông tin id nhà ga hành khách đã đi vào. |
| 2 | getId() | varchar | Lấy id của vé/thẻ |
| 3 | getExitPoint() | int | Lấy thông tin id nhà ga hành khách đã đi vào. |
| 4 | sendGateControlSignal() | void | Gửi tín hiệu điều khiển đến Cổng. |
| 5 | getInformationbyId(id) | void | Lấy thông tin vé theo id |
| 6 | sendDisplayInfo() | void | Gửi tín hiệu tới màn hình |
| 7 | setEnterPoint() | void | Nhận vào giá trị enterpoint |
| 8 | setExitPoint() | void | Nhận vào giá trị exitpoint |
| 9 | updateDatabase() | void | Cập nhật thay đổi vào database |

*Exception*:

* AException if …
* BException if …

**Method**

How to use parameters / attributes

Flowchart / Sequence diagram if the method has a complex/special algorithm

## Class “BController”

Hình ảnh class "BController"

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | fare | double |  | Giá tiền vào ga |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getStationInfo() | void | Lấy thông tin id nhà ga, thông tin tên, khoảng cách |
| 2 | calculateFare(st1,st2) | double | Tính giá vé đi từ st1 đến st2 |

*Parameter*:

- st1: thông tin id nhà ga hành khách đi vào.

- st2: thông tin id nhà ga hành khách đi ra.

## Class “CheckOnewayController”

Hình ảnh class "CheckOnewayController"

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | status | int |  | Trạng thái của vé |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getStatus() | int | Lấy thông tin trạng thái vé. |
| 2 | compareFare(tk) | boolean | So sánh giá mua vé với phí hành trình, trả về false nếu giá mua vé nhỏ hơn phí hành trình, trả về true nếu ngược lại. |

*Parameter*:

- tk: thông tin vé dùng một lần

## Class “Check24hController”

Hình ảnh class "Check24hController"

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
|  |  |  |  |  |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | checkValidTime() | boolean | Kiểm tra xem vé còn thời hạn sử dụng hay không. Trả về true nếu còn, trả về false nếu hết thời hạn. |

*Parameter*:

*Exception*:

## Class “CheckCardController”

Hình ảnh class "CheckCardController"

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | balance | double |  | Giá tiền vào ga |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | getBalance() | double | Lấy thông tin số dư tài khoản. |
| 2 | isEnoughBalance(balance) | boolean | Kiểm tra còn đủ tiền trong tài khoản hay không. |
| 3 | setBalance() | void | Lưu lại giá trị balance |

*Parameter*:

- balance: số dư tài khoản.

## Class “GateControllInterface”

Hình ảnh class "GateControllInterface"

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | gatestatus | int |  | Chứa giá trị biểu thị trạng thái của cửa: 1 là mở, 0 là đóng |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | openGate() | boolean | Mở cổng |
| 2 | closeGate() | boolean | Đóng cổng |
| 3 | getStatus() | int | Lấy thông tin về trạng thái hiện thời của cửa |
| 4 | setStatus() | void | Đặt giá trị trạng thái của cửa |

## Class “ScreenInterface”

Hình ảnh class "ScreenInterface"

**Attribute**

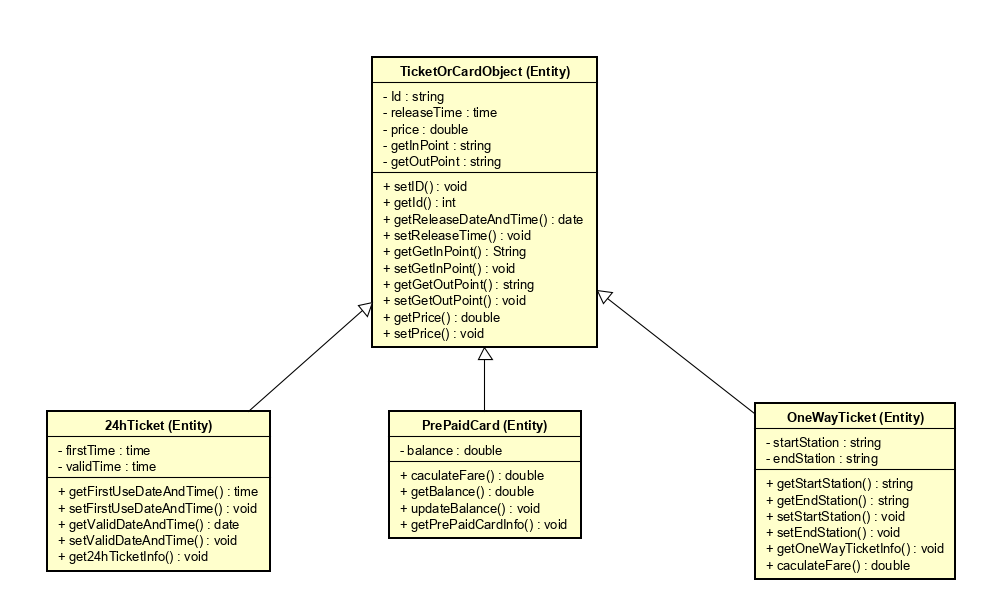
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
|  |  |  |  |  |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | displayStatus() | void | Hiển thị trạng thái |
| 2 | displayTicketInfor() | void | Hiển thị thông tin của vé |
| 3 | displayCardInfor() | void | Hiển thị thông tin của thẻ |

## Class “TicketOrCardObject”

Hình ảnh các lớp Entity



**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | id | string |  | Mã ID của vé hoặc thẻ được tạo khi phát hành vé hoặc thẻ đó |
| 2 | releaseTime | time |  | Thời gian phát hành vé hoặc thẻ |
| 3 | getInPoint | string |  | Tên nhà ga mà hành khách vừa mới đi vào |
| 4 | getOutPoint | string |  | Tên nhà ga mà hành khách muốn đi ra |
| 5 | price | double |  | Giá của thẻ hoặc vé |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 |  |  |  |
| 2 |  |  |  |

## Class “PrePaidCard”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | balance | double |  | Số dư có trong thẻ |
|  |  |  |  |  |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | caculateFare | double | Tính phí mà hành khách phải trả khi đi ra khỏi một nhà ga |
| 2 | updateBalance | void | Cập nhật số dư trong tài khoản khi hành khách ra khỏi nhà ga |
| 3 | getPrePaidCardInfo | void | In ra màn hình thông tin của thẻ bao gồm: mã ID, số dư, loại |

## Class “24hTicket”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | firstTime | time |  | Lưu thời điểm mà hành khách sử dụng vé lần đầu tiên (ngày và giờ) |
| 2 | validTime | time |  | Lưu thời điểm mà vé hết hạn sử dụng (ngày và giờ) |

**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | get24hTicketInfo | void | In ra màn hình thông tin của thẻ bao gồm: mã ID, giá vé,thời hạn sử dụng của vé, loại vé |

## Class “OneWayTicket”

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 | startStation | string |  | Tên nhà ga xuất phát mặc định của vé |
| 2 | endStation | string |  | Tên nhà ga kết thúc mặc định của vé |

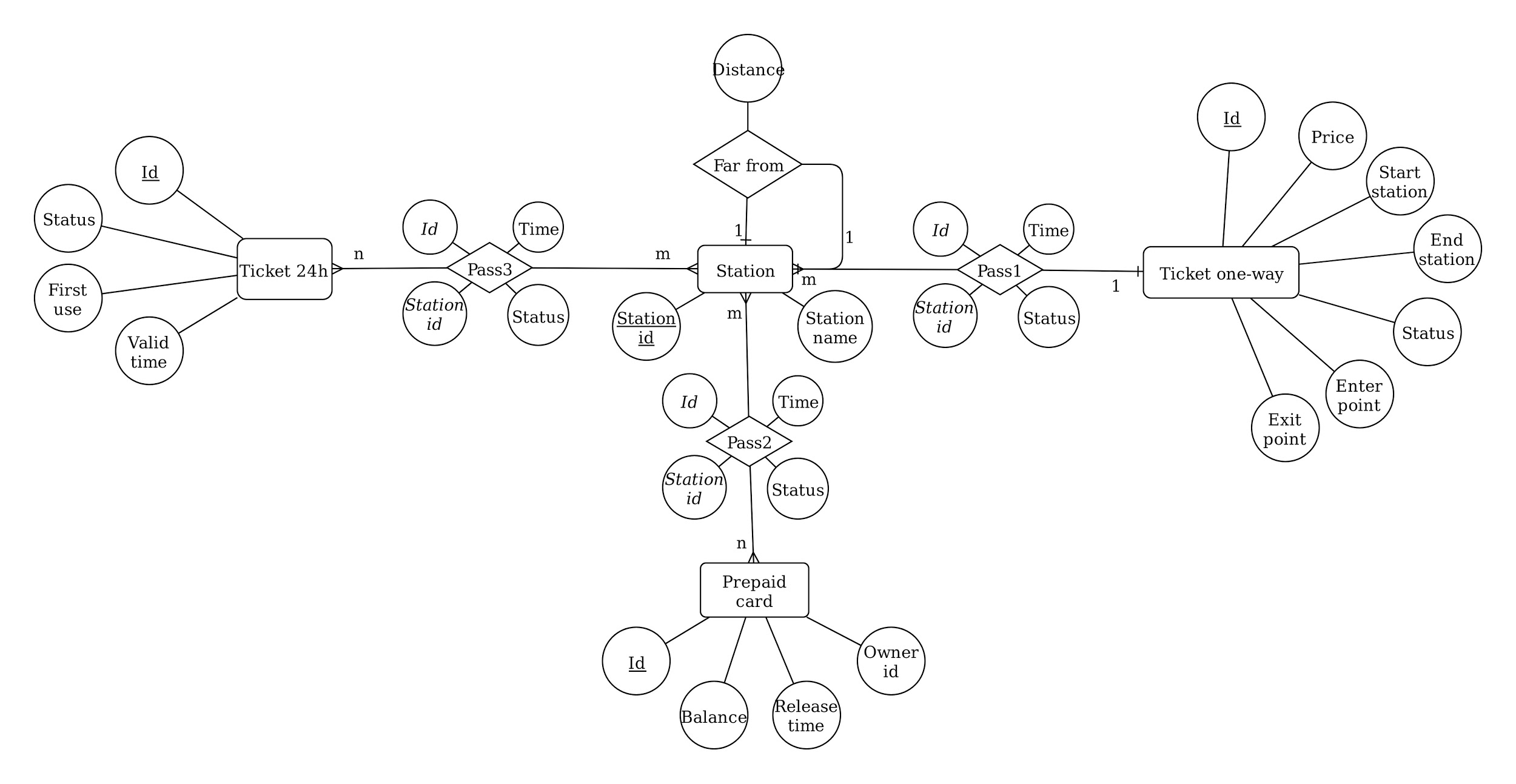
**Operation**

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 | CaculateFare | double | Tính phí mà hành khách phải trả khi đi ra khỏi một nhà ga |
| 2 | getOneWayTicketInfo | void | In ra màn hình thông tin của vé 1 chiều gồm: mã ID, giá vé, loại vé |

# Data modeling

# Conceptual Data Model

Biểu đồ thực thể liên kết:



**Thực thể:**

Tiket 24h:

* Id: Là khóa ghi thông tin định danh của vé.
* First use: Thời điểm lần đầu tiên sử dụng vé.
* Valid time: Thời điểm vé hết hạn.
* Status: Trạng thái vé.

Ticket one-way:

* Id: Là khóa ghi định danh của vé.
* Price: Giá tiền mua vé.
* Start station: Ga bắt đầu ( thông tin này được nhập khi mua vé)
* End station: Ga kết thúc ( thông tin này được nhập khi mua vé)
* Status: Trạng thái vé.
* Enter point: Điểm vào ga.
* Exit: point: Điểm ra ga.

Prepaid card:

* Id: Là khóa ghi định danh thẻ.
* Balance: Số dư trong thẻ.
* Release time: Ngày phát hành.
* Owner id: Khóa ngoài, định danh của chủ sở hữu.

Station:

* Station id: Là khóa ghi định danh của nhà ga.
* Name: Tên nhà ga.

Pass:

* *Id*: Khóa ngoài, thông tin định danh vé được sử dụng để đi vào/ra nhà ga.
* *Station id*: Khóa ngoài, thông tin nhà ga được đi qua.
* Time: Thời gian đi qua.
* Status: Trạng thái đi vào hay đi ra.

**Quan hệ**

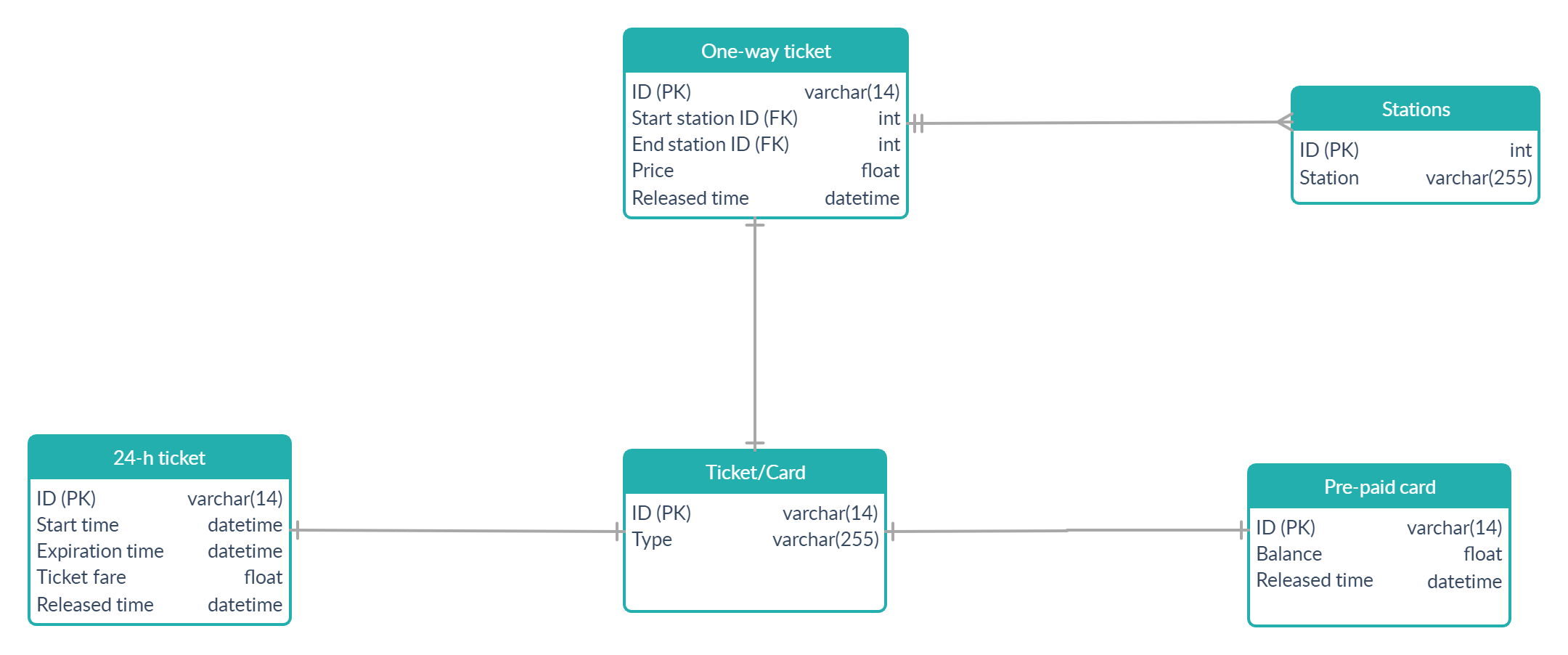
Ticket one-way --1------------------------<Pass1>---------------------------n--Station

Ticket 24h --n------------------------------<Pass2>---------------------------n--Station

Prepaid card --n--------------------------<Pass3>----------------------------n--Station

Station --1-------------------------------<Far from>----------------------------1--Station

# Logical Data Model

Hệ thống quản trị cơ sở dữ liệu sẽ sử dung: MySQL

Mô tả:

* MySQL là một Relational Database Management System hoạt động theo mô hình client-server
* MySQL được sử dụng để tạo và quản lý các cơ sở dữ liệu (Database) theo hình thức quản lý các mối liên hệ giữa chúng.
* Máy tính cài đặt và chạy phần mềm MySQL được gọi là client (máy khách). Mỗi khi chúng cần truy cập dữ liệu, chúng kết nối tới máy chủ (server) MySQL
* Cách vận hành chính trong môi trường MySQL:
* MySQL tạo ra bảng để lưu trữ dữ liệu, định nghĩa sự liên quan giữa các bảng đó
* Client sẽ gửi yêu cầu SQL bằng một lệnh đặc biệt trên MySQL
* Ứng dụng trên server sẽ phản hồi thông tin và trả về kết quả trên máy client
* Sử dụng MySQL do tính linh hoạt, hiệu năng cao và an toàn của nó.

Các bước thiết kế cơ sở dữ liệu từ sơ đồ được thể liên kết:

* Bước 1: Xác định các mối quan hệ giữa các thực thể:

- One to one

- One to Many

- Many to Many

* Bước 2: Chuẩn hóa cơ sở dữ liệu (1NF, 2NF, 3NF)

- 1NF: + Các thuộc tính của bảng phải là nguyên tố

+ Giá trị của các thuộc tính trên các hàng phải là đơn trị, không chứa nhóm lặp

+ Không có một thuộc tính nào có giá trị có thể tính toán được từ một thuộc tính khác

- 2NF:

+ Loại bỏ các thuộc tính không khóa phụ thuộc vào một bộ phận khóa chính và tách ra thành một bảng riêng, khóa chính của bảng là bộ phận của khóa mà chúng phụ thuộc vào.

+ Các thuộc tính còn lại lập thành một quan hệ, khóa chính của nó là khóa chính ban đầu.

- 3NF:

+ Mọi thuộc tính không khóa phụ thuộc bắc cầu vào thuộc tính khóa (nghĩa là tất cả các thuộc tính không khóa phải được suy ra trực tiếp từ thuộc tính khóa)

# Database design

Bảng lưu thông tin của thẻ trả trước

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *prepaid\_cards* | | | | | | | |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | id | varchar |  |  | Mã ID của thẻ được tạo ngay khi phát hành thẻ |
| 2 |  |  | released\_time | datetime |  | Not null | Thời điểm phát hành thẻ (ngày và giờ) |
| 3 |  |  | get\_in\_point | varchar |  |  | Tên của nhà ga mà hành khách vừa mới dùng thẻ để đi vào. Ban đầu trường này có giá trị null |
| 4 |  |  | balance | double |  | Not null | Số dư có trong thẻ (lớn hơn hoặc bằng 0) |

Bảng lưu thông tin của vé 24 giờ

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Twenty\_four\_hour\_tickets* | | | | | | | |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | id | varchar |  |  | Mã ID của vé được tạo ngay khi in vé |
| 2 |  |  | released\_time | datetime |  | Not null | Thời điểm in vé (ngày và giờ) |
| 3 |  |  | price | double |  | Not null | Giá vé |
| 4 |  |  | first\_use\_time | datetime |  |  | Thời điểm mà vé được sử dụng lần đầu tiên (ngày và giờ). Ban đầu trường này sẽ có giá trị null |
| 5 |  |  | valid\_time | datetime |  |  | Thời điểm mà vé sẽ không còn giá trị sử dụng(ngày và giờ). Ban đầu trường này sẽ có giá trị null và sẽ được update mỗi khi sử dụng thẻ để đi vào ga |
| 6 |  |  | status | int | 1 |  | Trạng thái sử dụng của vé |

Bảng lưu thông tin của vé một chiều

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *oneway\_tickets* | | | | | | | |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | id | varchar |  |  | Mã ID của vé được tạo ngay khi in vé |
| 2 |  |  | released\_time | datetime |  | Not null | Thời điểm in vé (ngày và giờ) |
| 3 |  |  | price | double |  | Not null | Giá vé |
| 4 |  |  | start\_station | varchar |  | Not null | Tên nhà ga khởi hành được mặc định trên vé |
| 5 |  |  | end\_station | varchar |  | Not null | Tên nhà ga kết thúc được mặc định trên vé |
| 6 |  |  | get\_in\_point | char |  |  | Tên của nhà ga mà hành khách vừa mới sử dụng dùng vé để đi vào. Ban đầu trường này có giá trị null |
| 7 |  |  | status | int | 1 |  | Trạng thái của vé |

Bảng lưu thông tin của các nhà ga

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *stations* | | | | | | | |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | id | varchar |  |  | Mã ID của gắn với mỗi nhà ga |
| 2 |  |  | name | varchar |  |  | Tên của nhà ga |

Bảng lưu khoảng cách các nhà ga

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *station\_distance* | | | | | | | |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x | x | station\_id1 | varchar |  | Not null | Mã ID của ga thứ 1 |
| 2 | x | x | station \_id2 | varchar |  | Not null | Mã ID của ga thứ 2 |
| 3 |  |  | distance | double |  | Not null | Khoảng cách giữa 2 nhà ga |