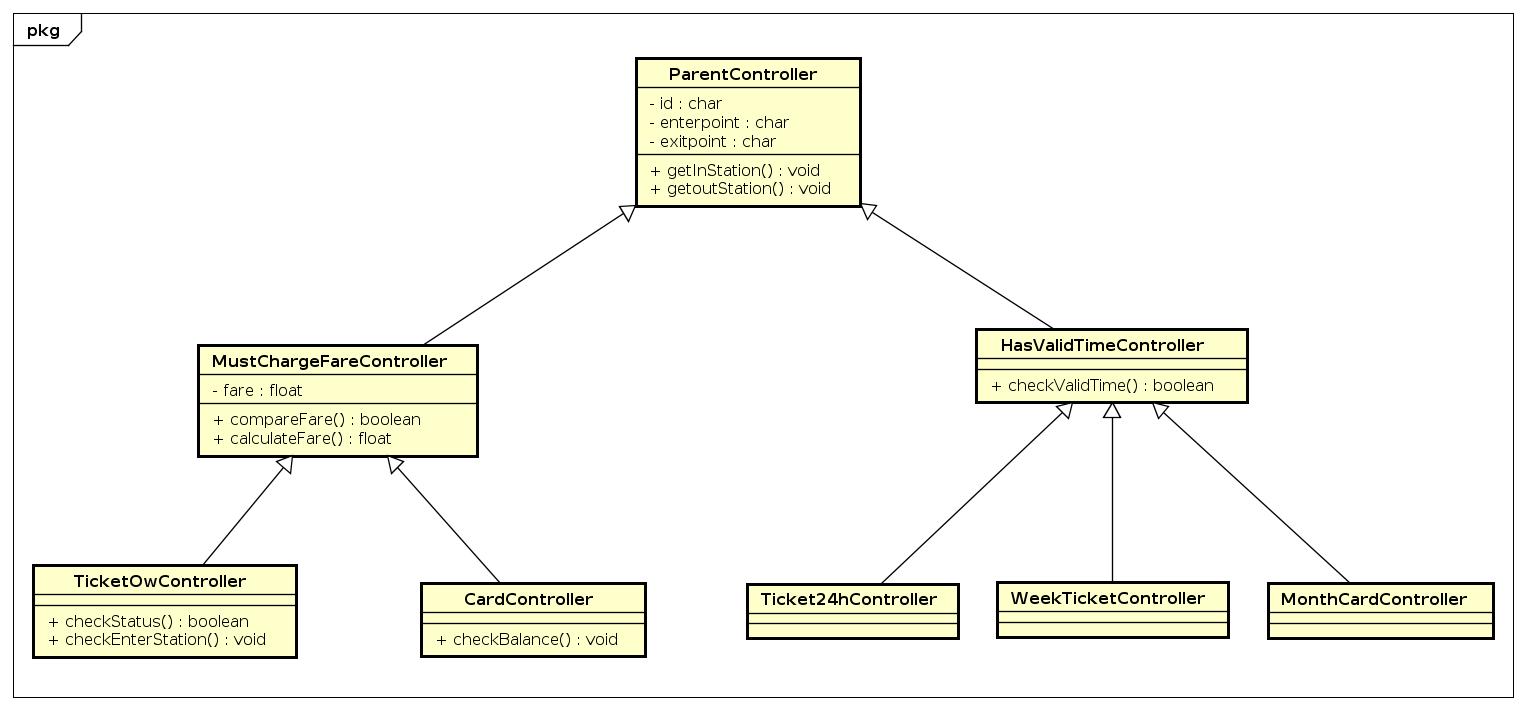
|  |  |
| --- | --- |
| Bài tập tuần 11 - Group 01 | |
| Phạm Ngọc Bá | Yêu cầu: Thêm các loại thẻ, vé mới |
| Nguyễn Quang Anh | Yêu cầu: Thay đổi cách tính phí |
| Hoàng Gia Bảo | Yêu cầu: Xử lý khi có nhiều tuyến và các tuyến có thể cắt nhau |
| Source code sẽ được bổ sung và chỉnh sửa trong folder MiniProjectItss | |

1. Yêu cầu thêm thẻ tháng và vé tuần

Đối với yêu cầu "Thêm thẻ tháng hay vé tuần" ta cần thêm các class mới, tổ chức lại cấu trúc như sau:

* Ở Package **dto**, **dao**: ta thêm các class tương ứng để tương tác, lấy dữ liệu từ database.
* Còn tại Package **controller**: Ta tổ chức lại cấu trúc class như sau: 
  + Class cha ***ParentController*** bao gồm các thuộc tính và phương thức chung nhất của các loại thẻ và vé
  + Class ***MustChargeFareController*** và ***HasValidTimeController*** kế thừa từ class ***ParentController***
  + Các class ***TicketOwController***, ***CardController*** kế thừa từ class ***MustChargeFareController***
  + Các class ***Ticket24hController***, ***WeekTicketController*** và ***MonthCardController*** kế thừa từ class ***HasValidTimeController***

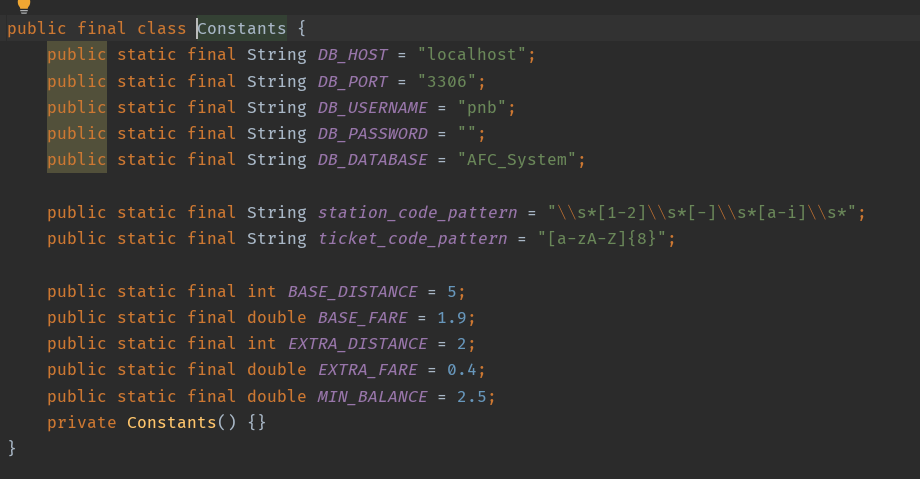
Nhận thấy các loại vé/card được sử dụng để đi qua nhà ga có 2 tính chất chính đó là: 1 loại khi nào dùng trả tiền lần đó, cần thiết phải tính phí và 1 loại không giới hạn số lần sử dụng nhưng giới hạn trong một khoảng thời gian nào đó. Em thiết kế ra 2 class ***MustChargeFareController*** và ***HasValidTimeController*** .

Đối với các loại vé thẻ mới muốn thêm vào hệ thống thì tùy theo tính chất nó sẽ kế thừa từ 1 trong 2 lớp trên. Các phương thức trong lớp mới cũng sẽ được viết lại cho phù hợp.

2. Thay đổi cách tính phí

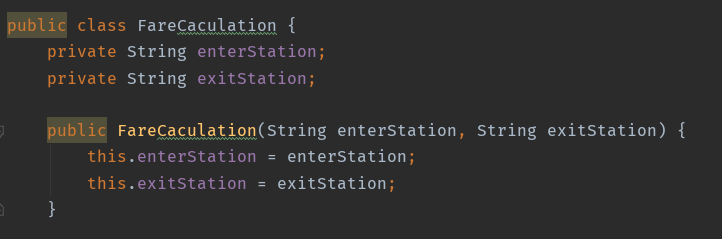
a) Thay đổi các mốc, hạn mức giá trị:

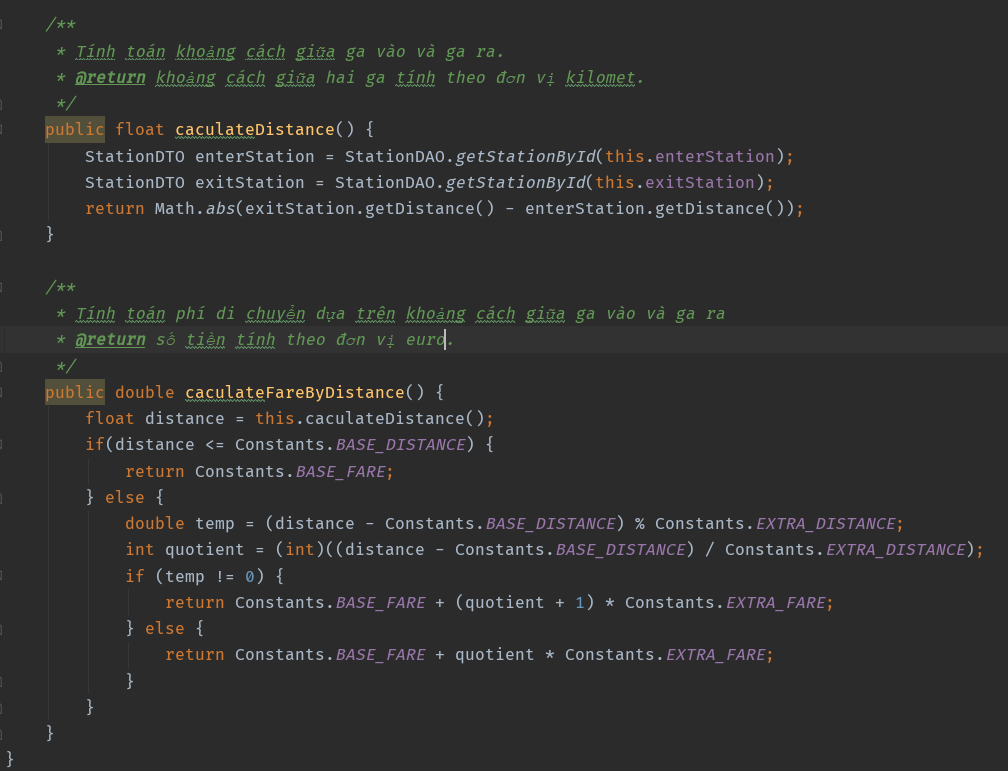
+) Các giá trị như: phí cơ bản và giới hạn về khoảng cách để áp dụng phí cơ bản (ví dụ: 7 euro/5km đầu) , phí phát sinh (ví dụ: 1,5 euro/2 km), các mốc khoảng cách để tính tiền (ví dụ: 5 euro/3 km đầu, 8 euro/(3 km - 5 km). 10 euro/(5 km – 7 km)), …đều được lưu thành các hằng số trong lớp Constants và khi cần sử dụng đến những giá trị này ta sẽ gọi đến lớp Constants

.

Trong trường hợp muốn thay đổi các giá trị này ta chỉ cần vào trong lớp Constants để chỉnh sửa mà không cần phải sửa trực tiếp trong các lớp khác.

+)Tạo một lớp có tên là FareCaculation trong packet Controller.

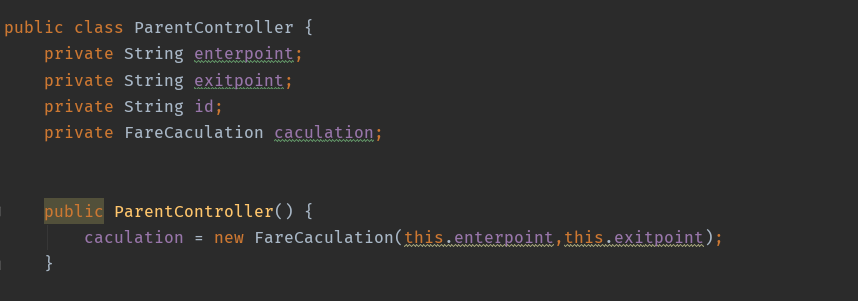
 Lớp này có 2 thuộc tính là enterStation và exitStation lưu lại id của nhà ga mà hành khách đã đi vào và đi ra; và chứa các hàm để tính toán chi phí đi lại dựa trên id của ga vào/ra được lưu ở trên.



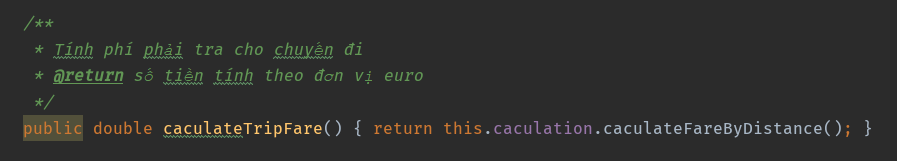
Mỗi một cách tính sẽ được viết thành một hàm riêng. Chẳng hạn hàm caculateFareByDistance dùng để tính toán chi phí đi lại dựa trên khoảng cách giữa hai ga.

+) Cách sử dụng lớp FareCaculation:

- Khai báo thêm một đối tượng caculation và khởi tạo cùng với Controller

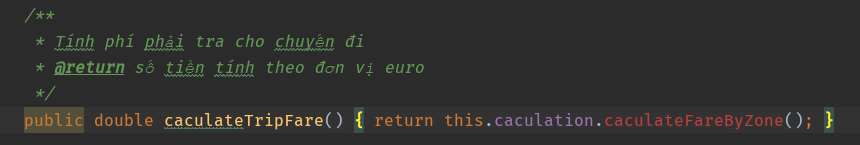


- Gọi hàm bên trong đối tượng “calculation” để tính toán:



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

Ta sẽ bổ sung một hàm caculationFareByZone dung để tính phí dựa trên zone của hai nhà ga bên trong lớp FareCaculation và gọi đến hàm này trong lớp Controller:



3. Xử lý khi có nhiều tuyến, các tuyến có thể cắt nhau.

* Thêm Class LineDTO, LineDAO
* Thay đổi database: thêm bảng line, thêm trường lineId vào bảng station.