Improvement Report

Sau khi tiếp thu ý kiến từ cô góp ý cho nhóm em cũng như các nhóm khác thì chúng em đã thực hiện một số những thay đổi để hoàn thiện bài tập lớn của nhóm hơn.

Cụ thể là:

1 Thay đổi về nghiệp vụ

- Chúng em tiến hành cho người dùng nhập thông tin thẻ trước khi xử lý yêu cầu thuê xe. Điều này xảy ra vì hệ thống không xét đến đăng nhập do đó mỗi người dùng cần phải nhập thông tin thẻ trước khi thuê xe.
- Đảm bảo với mỗi thẻ chỉ được thuê một xe duy nhất: Hệ thống sẽ kiểm tra thông tin thẻ người dùng nhập xem có đang được sử dụng để thuê xe hay không, nếu đang thuê xe thì hệ thống sẽ báo người dùng cần phải sử dụng thẻ khác.
- Khi trả xe cần phải nhập thông tin mã giao dịch thuê xe là "rental code". Điều này giúp cho hệ thống của chúng em đảm bảo khi người dùng tắt app đi thì khi bật lại sẽ không xảy ra hiện tượng: "không đăng nhập mà hệ thống vẫn biết đó là ai và đang thuê xe nào để trả".
- Người dùng sau khi nhập mã giao dịch thuê xe thì hệ thống sẽ tiến hành kiểm tra và lấy thông tin xe được thuê tương ứng với giao dịch đó. Sau đó, hệ thống sẽ tiến hành xử lý yêu cầu trả xe của khách hàng.

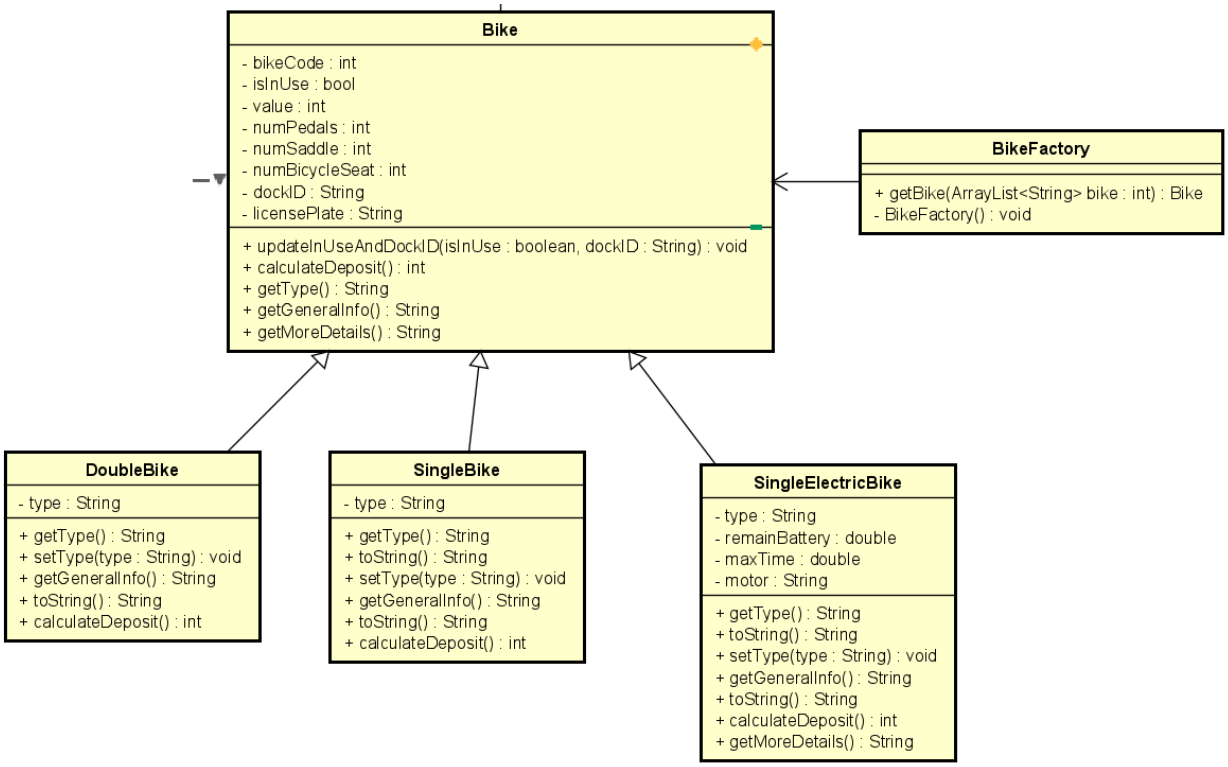
2 Thay đổi về thiết kế

2.1 Sử dụng design pattern factory

Trước vấn đề thêm loại xe mới có cách tính tiền cọc mới vào hệ thống, chúng em đã sử dụng kế thừa và pattern Factory để xử lý vấn đề này. Cụ thể:

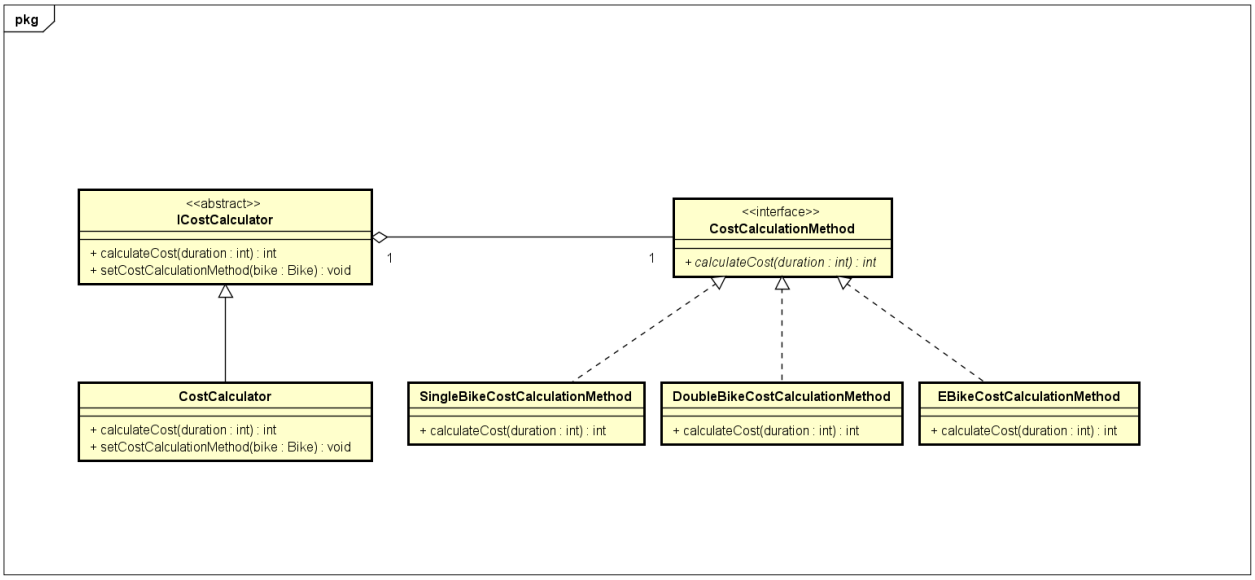
- Chúng em xây dựng lớp cha Bike chứa các trường thông tin cần thiết nhất, bắt buộc xe nào cũng phải sở hữu và cần khai báo khi tạo ra đối tượng xe mới.
- Xây dựng thêm các method calculateDeposit(..), getMoreDetailInfo(..) để có thể xử lý các trường thông tin thêm riêng đối với từng loại xe.

- Mỗi loại xe sẽ được kế thừa từ lớp cha "Bike", hiện tại chúng em có 3 lớp con là "SingleBike", "DoubleBike" và "SingleElectricBike" ứng với 3 loại xe theo yêu cầu. Mỗi loại xe sẽ có các method được implement lại của method lớp cha sao cho phù hợp với từng loại xe.
- Chúng em sử dụng pattern Factory cho mục đích: "cần tạo ra các instance xe tương ứng với lớp cần tạo". Ví dụ: Khi chúng em lấy được thông tin xe từ database là 1 ArrayList<String>, từ đây chúng em muốn tạo ra instance xe tương ứng với loại xe mà mình muốn. Từ đó chúng em chọn Factory là một pattern dùng để tạo ra các instance tương ứng với lớp con mà mình mong muốn.



Lớp BikeFactory sẽ có nhiệm vụ là tạo ra các instance mà mình mong muốn, và khi thêm xe mới với loại mới, ta chỉ cần tạo class tương ứng với loại xe đó, cho lớp đó kế thừa lớp cha "Bike". Sau đó, thêm phần khởi tạo tương ứng với loại xe đó vào lớp BikeFactory. Việc thay đổi nay được thực hiện rất dễ dàng và nó không ảnh hưởng gì về phía code của các lớp khác.

2.2 Sử dụng stategy



- Thay vì tình toán chi phí thuê xe trực tiếp trong lớp ReturnBikeController chúng em sử dụng statrgy pattern để tách biệt việc tính toán chi phí ra, cụ thể với mỗi loại xe sẽ có 1 phương pháp tính toán chi phí khác nhau ứng với 3 lớp SingleBikeCostCalculationMethod, DoubleBikeCostCalculationMethod, EbikeCostCalculationMethod. 3 lớp này implement lớp interface, từ đó cho phép hệ thống có thể thêm các cách tính tiền mới mà không làm ảnh hưởng tới code trước đó của hệ thống.
- Trong quá trình thực thi chương trình, tùy theo mỗi loại xe ứng với một kiểu tính tiền khác nhau. Để thực hiện điều này lớp ICostCalculator có 1 thuộc tính là CostCalculationMethod và sẽ có hàm set thuộc tính đó setCostCalculationMethod (bike: Bike). Hàm này sẽ cho class chọn phương pháp tính tiền tùy theo loại xe tương ứng. Việc sử dụng lớp abstact cũng cho phép hệ thống mở rộng dễ dàng hơn và làm giảm độ phụ thuộc giữa lớp ReturnBikeController và lớp CostCalculator.