GOOD DESIGN

DESIGN PATTERNS – DESIGN PRINCIPLES

1. Design principles:

Với phần programming chúng em đã xây dựng. Chúng em thấy rằng:

- "S": Mỗi class của chúng em chỉ thực hiện 1 task, không có sự đan xen lẫn lộn giữa task của thuê xe và trả xe cũng như xem thông tin xe.
- "O": Các class của chúng em đều có thể mở rộng bằng extension mà không cần phải sửa nội dung bên trong class cha.
- "L": Với phạm vi hệ thống không lớn, giao tiếp giữa các thành phần không phức tạp nên trong phần programming hiện tại chúng em chưa sử dụng nguyên lý này.
- "I": Trong chương trình, chúng em sử dụng 2 interface đó là IInterbank và IBarcodeConverter. Vì vậy mà 2 subsystem thực hiện các nhiệm vụ khác nhau sử dụng 2 interface khác nhau, không tồn tại sự chồng chéo giữa 2 subsystem này.
- "D": Các lớp controller sử dụng các chức năng trong subsystem thông qua interface mà không cần quan tâm tới chi tiết trong mỗi chức năng interface cung cấp.

2. Design pattern:

2.1 Vấn đề:

Ban đầu khi code, chúng em tạo ra class Card theo một cách thông thường và trong quá trình lập trình ta có thể có nhiều instance cho lớp đó.

Nhưng khi đọc yêu cầu về bài toán đặt ra và tính hợp lý trong cách lập trình chúng em nhận thấy rằng:

- Thông tin thẻ đã được cố định và gắn với người dùng.
- Việc tạo ra nhiều instance là thừa và không hiệu quả.
- Thông tin nếu cần thay đổi thì cũng sẽ phải cần thay đổi ở các instance khác.

Chính vì những lý do trên mà chúng em quyết định sử dụng design pattern Singleton cho class Card

2.2 Áp dụng Singleton cho class Card:

```java
package entities;

public class Card {
    private final String cardCode;
    private final String owner;
    private final String CVV;
    private final String expiredDate;
    private static final Card card = new Card( cardCode: "118131_group8_2020",  owner: "Group 8",  CVV: "427",  expiredDate: "1125");

    public Card(String cardCode, String owner, String CVV, String expiredDate) {
        this.cardCode = cardCode;
        this.owner = owner;
        this.CVV = CVV;
        this.expiredDate = expiredDate;
    }

    public static Card getInstance() { return card; }

    public String getCardCode() { return cardCode; }

    public String getOwner() { return owner; }

    public String getCVV() { return CVV; }

    public String getExpiredDate() { return expiredDate; }
}
```

Như vậy, thẻ bây giờ chỉ có duy nhất 1 instance và khi sử dụng thẻ, chúng ta chỉ cần gọi Card.getInstance() là có thể sử dụng tất cả những thông tin gắn sẵn mà thẻ cung cấp.