DESIGN PATTERN

Nhóm 1

STT	Họ tên	MSSV
1	Nguyễn Đức Anh	20172956
2	Lê Xuân An	20172931
3	Nguyễn Đình Quang Anh	20172936
4	Nguyễn Mạnh Cường	20172989

Clean Code và SOLID

Lớp vi phạm SRP, vì có method validate trong hàm, chuyển toàn bộ ra một Class có tên là Validate

```
public class Validate {
	private static Validate single_instance = null;
	public void validateDeliveryInfo(HashMap<String, String> info){}

	public boolean validatePhoneNumber(String phoneNumber) {};

	public boolean validateName(String name) {};
	public boolean validateAddress(String address) {};

}
```

Clean Code và SOLID

Lớp AuthenticationController có phương thức md5, vi phạm SRP

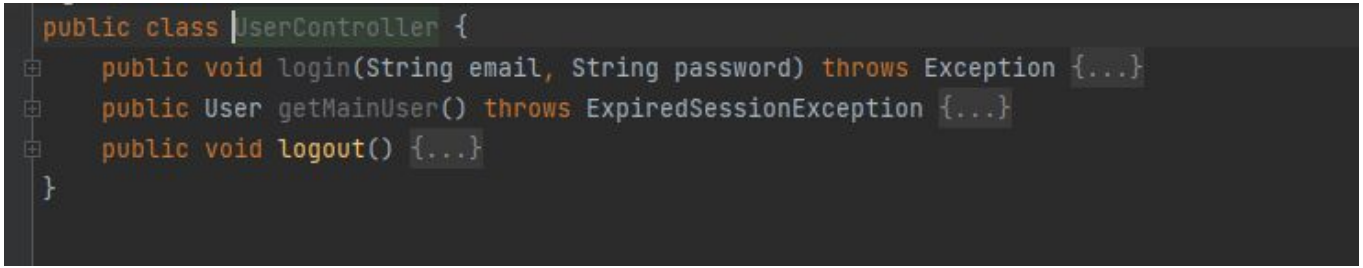
=> nên chuyển ra thành một lớp riêng biệt chuyên xử lý Security

```
8
9  public class Security {
0      private static Security single_instance = null;
1
2      public String md5Encryption(String message) {
3      }
4  }
```

Clean Code và SOLID

Vi phạm SRP

AuthenticationController thực hiện một lúc nhiều method nên tách riêng một class UserController chứa các phương thức login(), logout(), getMainUser(),...



Clean Code và SOLID

Vi phạm SRP

vì có method IsValidMonthAndYear, làm không rõ mục đích của class PaymentController

-> chuyển thành method trong class Validate để tái sử dụng

```
public class Validate {
    private static Validate single_instance = null;
    // variable of type String
    public String s;
    // private constructor restricted to this class itself
    private Validate() {}
    // static method to create instance of Singleton class
    public static Validate getInstance() {...}
    public void validateDeliveryInfo(HashMap<String, String> info)
            throws InterruptedException, IOException, InvalidDeliveryInfoException {...}
    public boolean validatePhoneNumber(String phoneNumber) {...}
    public boolean validateName(String name) {...}
    public boolean validateAddress(String address) {...}
    public boolean validateTimeOrder(int month, int year){
        if (month < 1 || month > 12 || year < Calendar.getInstance().get(Calendar.YEAR) % 100 || year > 100) {
            return false;
        }
        return true;
    }
}
```

```
private boolean isValidTimeOrder(int month, int year){
    Validate validate = Validate.getInstance();
    boolean isValid = validate.validateTimeOrder(month, year);
    return isValid;
}
```

Clean Code và SOLID

Vi phạm OCP

PaymentTransaction chỉ thực hiện duy nhất 1 cách thanh toán CreditCard, nếu sau này thêm phương thức thanh toán khác sẽ phải modify mã nguồn

Do đó tạo 1 class mới là GeneralPayment, và cho Card implement GeneralPayment

```
public interface GeneralPayment {
    public void PaymentByCard();
    public void PaymentByQR();
    public void PaymentByMomo();
    public void PaymentByZalopay();
    //etc...
}
```

Clean Code và SOLID

Vi phạm LSP

BaseController được kế thừa nhưng không sử dụng các phương thức ví dụ:

```
public class ViewCartController extends BaseController{

    /** This method checks the available products in Cart ...*/
    public void checkAvailabilityOfProduct() throws SQLException{...}

    /** This method calculates the cart subtotal ...*/
    public int getCartSubtotal(){
        return SessionInformation.cartInstance.calSubtotal();
    }

}
```

```
public class HomeController extends BaseController {
    /** this method gets all Media in DB and return back to home to display ...*/
    public static List getAllMedia() throws SQLException{...}
}
```

Clean Code và SOLID

Vi phạm ISP

BaseScreenHandle được HomeScreenHandler kế thừa nhưng không sử dụng phương thức setPreviousScreen, getPreviousScreen

Tách ra mộ class riêng PreviousScreenHandler có 2 phương thức setPreviousScreen, getPreviousScreen

```
public class PreviousScreenHandler {
    private BaseScreenHandler prev;
    public void setPreviousScreen(BaseScreenHandler prev) { this.prev = prev; }
    public BaseScreenHandler getPreviousScreen() { return this.prev; }
}
```

Clean Code và SOLID

Vi phạm DIP

PaymentTransaction Class này phụ thuộc vào một class mà đã được extend từ một class khác

Thay thế lớp Card cho lớp CreditCard

```
public class PaymentTransaction {
    private String errorCode;
    private Card card;
    private String transactionId;
    private String transactionContent;
    private int amount;
    private String createdAt;

    public PaymentTransaction(String errorCode, Card card, String transactionId, String transactionContent,
                              int amount, String createdAt) {...}

    public String getErrorCode() { return errorCode; }
}
```

Clean Code và SOLID

Clear name

Class	Mô tả	Chỉnh sửa
InterbankSubsystem	Biến ctrl viết tắt dễ gây hiểu nhầm	ctrl -> interbankSubsystemController
PaymentController	Biến đặt tên không mô tả được nội dung	isValidMonthAndYear -> isValidTimeOrder

Singleton

Vấn đề: Cart trong nghiệp vụ chỉ cần 1 instance

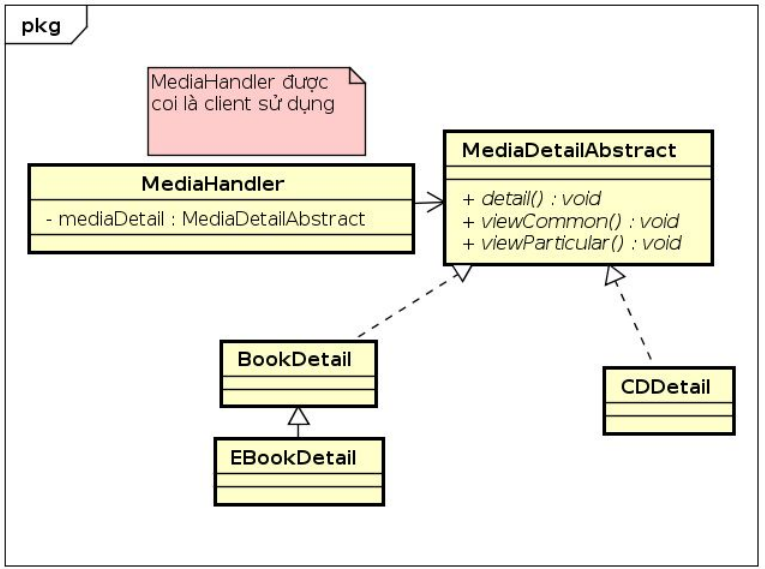
Validate chỉ cần 1 instance

```
public class Cart {
    private static Cart instance;
    private List<CartItem> lstCartItem = new ArrayList<>();

    private Cart() {
    }

    public synchronized static Cart getInstanse() {
        if (instance == null) {
            instance = new Cart();
        }
        return instance;
    }
```

Template

Thêm xem chi tiết sản phẩm

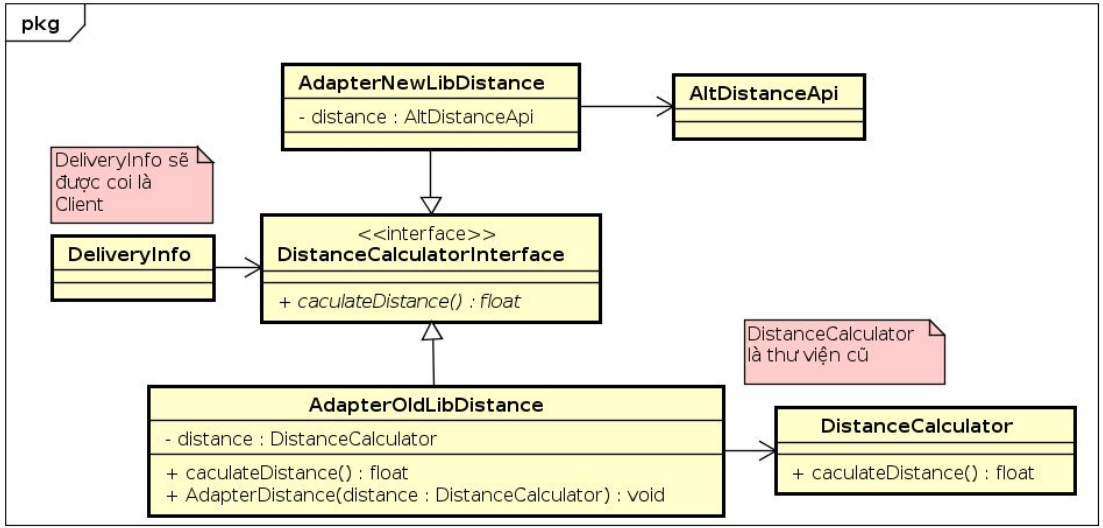
```
1  package views.screen.detail;
2
3  public abstract class MediaDetailAbstract {
4      public void detail() {
5          viewCommon();
6          viewParticular();
7      }
8
9      protected abstract void viewCommon();
10
11     protected abstract void viewParticular();
12 }
13
```



Adapter Pattern

Vấn đề: Thay đổi phương thức tính khoảng cách

Ta sử dụng Adapter để các interface có thể giao tiếp với nhau

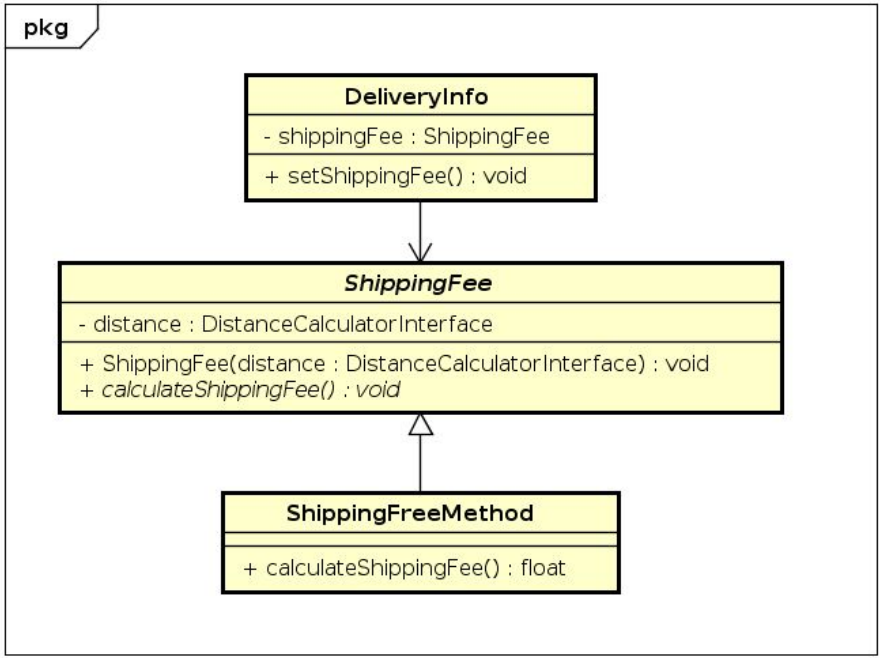


Strategy

Thay đổi phương thức tính phí ship

- Áp dụng Adapter vừa sử dụng

- ShippingFee là Strategy

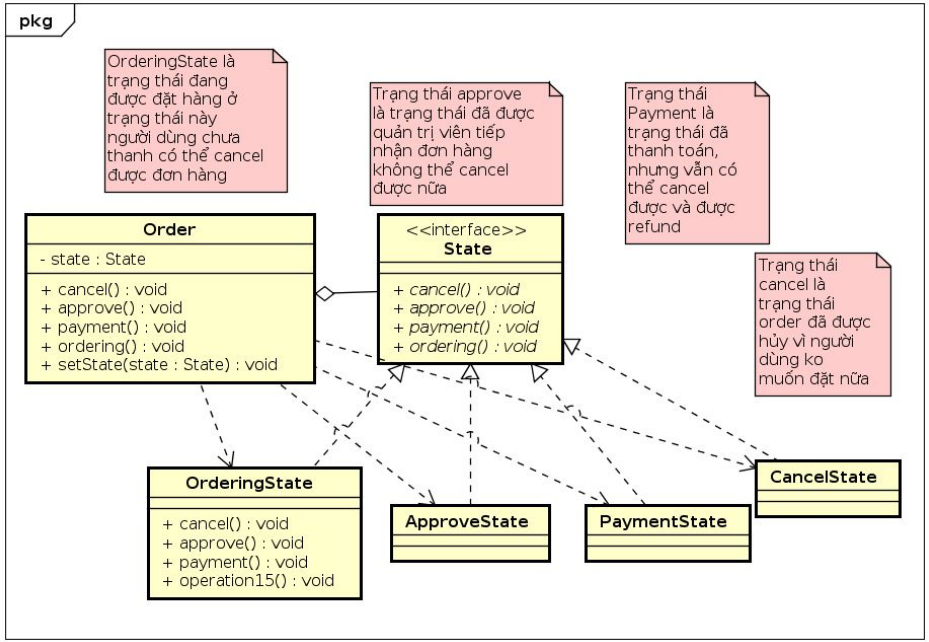


State

Đơn hàng có thể thay đổi trạng thái

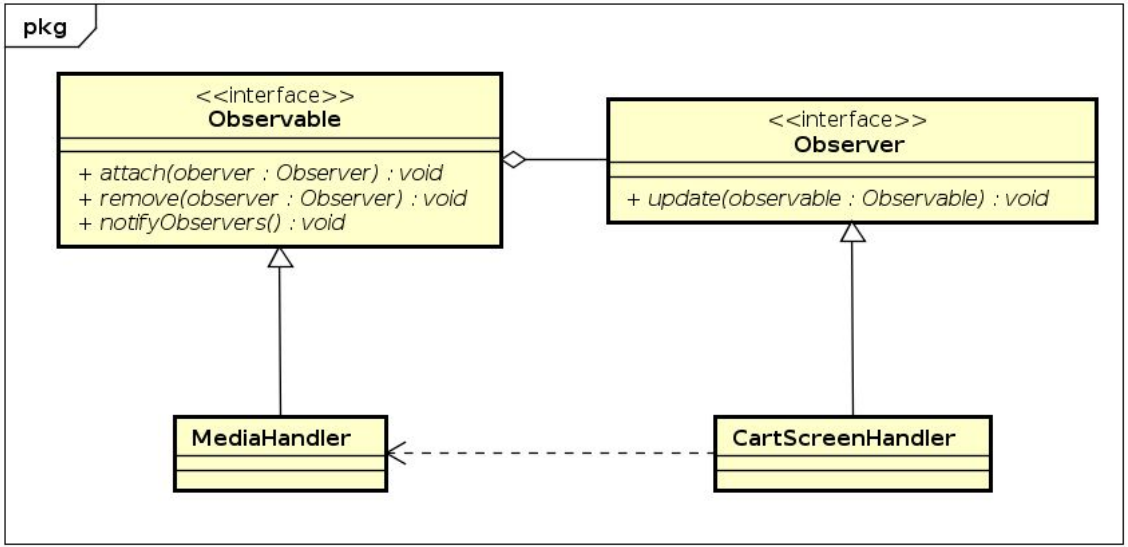
Các trạng thái ví dụ của đơn hàng

- Ordering
- Approve
- Payment
- Cancel



Observer

Vấn đề: Xóa item trong màn hình Cart



Coupling, Cohesion

Có khá nhiều vi phạm liên quan đến Coupling Cohesion. Đa phần các vi phạm không ảnh hưởng gì nhiều.

Cohesion

Coincidental Cohesion.

Hàm getToday trong class InterbankPayloadConverter

Không liên quan tới các hàm khác trong class. => Khiến class cồng kềnh

Giải pháp: đưa vào 1 class khác chứa các hàm dùng chung. Khi nào cần mới gọi đến

```
private String getToday() {
    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Date date = new Date();
    return dateFormat.format(date);
}
```

Cohesion

Coincidental Cohesion

Các method trong class InvoiceScreenHandler không liên quan gì tới nhau.

=> Tách thành các class độc lập nhau. Mỗi class thực hiện 1 nhiệm vụ duy nhất.

```
protected void setupData(Object dto) throws Exception {...}

protected void setupFunctionality() throws Exception {...}

@FXML
void confirmInvoice(MouseEvent event) throws IOException {...}
```

Coupling

Stamp Coupling

Hàm calculateShippingFee này không sử dụng đến đối tượng order nhưng vẫn nhận order là tham số truyền vào.

=> Không truyền cả đối tượng order vào. Nếu cần đến giá trị cụ thể nào thì truyền giá trị đó vào.

```
public int calculateShippingFee(Order order) { // Vi pham nguyen tac Stamp Coupling
    // boi vi truyen doi tuong order vao nhung khong su dung cac thuoc tinh cua doi
    // tuong nay
    int distance = distanceCalculatorInterface.calculateDistance(address, province);
    return (int) (distance * MULTIPLIER);
}
```

Coupling

Commong Coupling

Method này cho phép bên sử dụng kết nối trực tiếp đến database nhưng lại để public, class nào cũng có thể sử dụng được

=> Khó kiểm soát dữ liệu

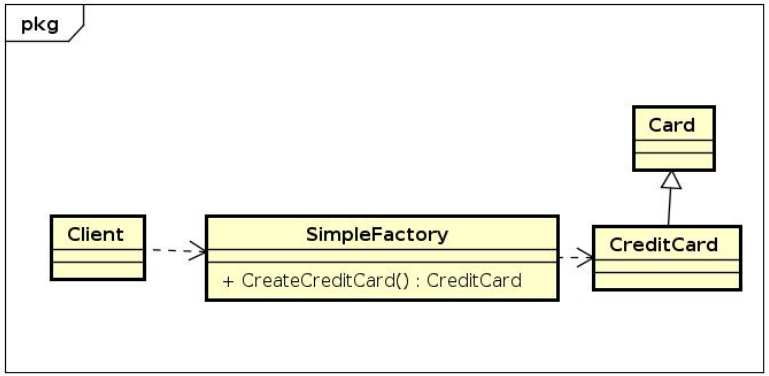
Giải pháp: tạo một hàm thực hiện câu truy vấn được gửi tới.

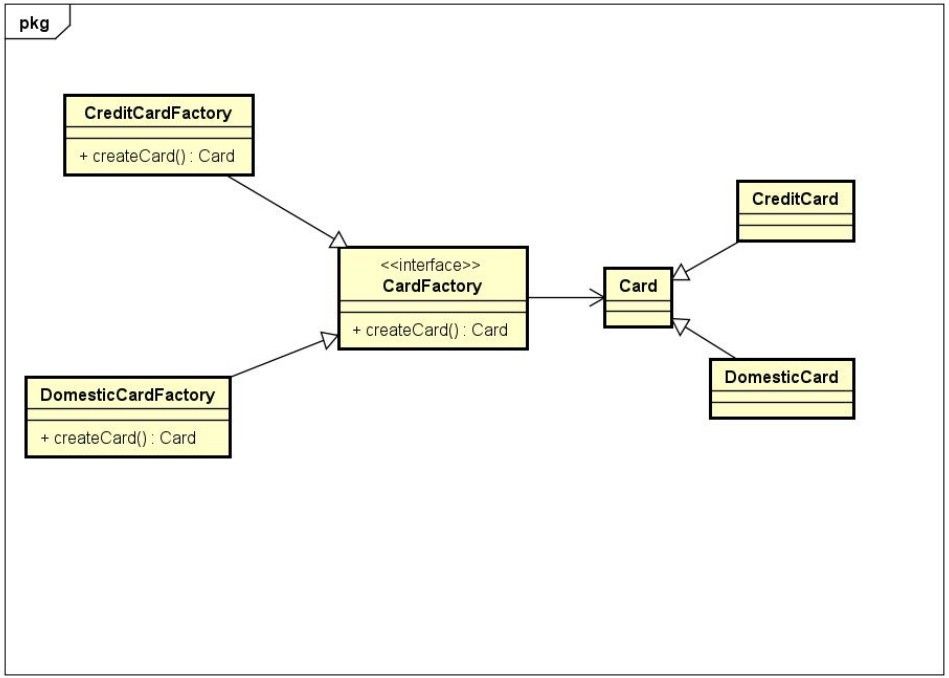
```
public static Connection getConnection() {
    if (connect != null) return connect;
    try {
        Class.forName(NAME_JDBC);
        String url = DATABASE_URL;
        connect = DriverManager.getConnection(url);
        LOGGER.info("Connect database successfully");
    } catch (Exception e) {
        LOGGER.info(e.getMessage());
    }
    return connect;
}
```

Factory Method

Thêm phương thức thanh toán mới là thẻ nội địa Domestic Card.

Áp dụng Factory Method





Factory Method