

DESIGN CONCEPTS AIMS

1. Coupling

1.1. Content Coupling

Related modules	Description	Improvement
Order.java	Trong tương lai, có thể xuất hiện một module mới mà nó gọi đến phương thức <code>getDeliveryInfo()</code> để quản lý <code>deliveryInfo</code> thì module ấy có thể thay đổi trực tiếp vào đối tượng đó bằng phương thức <code>put</code> của <code>HashMap</code>	Xóa bỏ phương thức <code>getDeliveryInfo()</code> <pre>public HashMap getDeliveryInfo() { return (HashMap) deliveryInfo.clone(); }</pre> và thay thế bằng phương thức <code>setDeliveryInfoField()</code> <pre>public String setDeliveryInfoField(String key, String value) { return this.deliveryInfo.put(key, value); }</pre>

1.2. Common Coupling

Related modules	Description	Improvement
Utils.java	<pre>public static boolean isNullOrEmpty(String str) { return str == null str.isEmpty(); }</pre> Những biến <code>public static</code> được sử dụng ở nhiều module khác nhau, các module có thể thay đổi dữ liệu	Nên tạo một lớp riêng chỉ có 1 instance, hoặc đặt cho biến là <code>final</code>

1.3. Control Coupling

Project không tồn tại loại coupling này.

1.4. Stamp Coupling

Related modules	Description	Improvement
Cart.java	<pre>public void addMedia(Media media) { for (HashMap.Entry<String, Media> entry : deliveryInfo.entrySet()) if (entry.getKey().equals("media")) entry.setValue(media); }</pre> Tham số truyền vào cả object <code>media</code> , nhưng trong method này chỉ sử dụng <code>id</code> của <code>media</code>	Chỉ cần truyền vào <code>id</code>
PlaceOrderController.java	<pre>public void placeOrder(String address, String name, String phone) { PlaceOrderRequest request = new PlaceOrderRequest(); request.setAddress(address); request.setName(name); request.setPhone(phone); placeOrder(request); }</pre> Tham số truyền vào là cả object <code>info</code> nhưng chỉ dùng 3 trường là <code>address</code> , <code>name</code> , <code>phone</code>	Chỉ cần truyền vào 3 trường

1.5. Data Coupling

Related modules	Description	Improvement
PaymentController.java	<pre>public void pay(String cardNo, String cardExp, String cardHolderName, String cardType) { PaymentRequest request = new PaymentRequest(); request.setCardNo(cardNo); request.setCardExp(cardExp); request.setCardHolderName(cardHolderName); request.setCardType(cardType); pay(request); }</pre>	


2. Cohesion

2.1. Coincidental Cohesion

Related modules	Description	Improvement
API.java	<pre>public void delete(String id) { delete(id); }</pre> trường	Có thể xóa đi trường

	DATE_FOMATER được khai báo, nhưng không được sử dụng ở bất kỳ đâu	
--	--	--

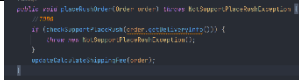

2.2. Logical Cohesion

Related modules	Description	Improvement
PlaceOrderController.java	 <p>Trong việc xử lý xác thực các thông tin về phone, name, address tuy đều là xác thực, nhưng chúng khác nhau về cách xử lý.</p>	Tách chúng ra các class riêng và cùng implement phương thức validate từ Interface ValidateInterface

2.3. Temporal Cohesion

Related modules	Description	Improvement
HomeScreenHandler.java	 <p>Phương thức initialize chỉ sử dụng trong thời điểm khởi tạo đối tượng</p>	Không có phương pháp tối ưu nhất


2.4. Procedural Cohesion

Related modules	Description	Improvement
PlaceRushOrderController.java	 <p>phương thức checkSupportPlaceRush() và updateCalculateShippingFee() chỉ gắn kết với nhau về mặt thứ tự</p>	
PlaceOrderController.java	 <p>phương thức validate chỉ gắn kết với nhau về mặt thứ tự</p>	


2.5. Communicational Cohesion

Related modules	Description	Improvement
InterbankSubsystemController.java	Cả module cùng trả về kiểu dữ liệu là PaymentTransaction	

2.6. Sequential Cohesion

Related modules	Description	Improvement
API.java	 <p>Đầu ra của phương thức này là đầu ra của các phương thức khác trong module</p>	

2.7. Functional Cohesion

Related modules	Description	Improvement
API.java	 <p>Đầu ra của phương thức này chính là đầu vào của các phương thức khác, và nó là cần thiết cho chức năng của module</p>	