**I – Coupling & Cohesion**

1. Coupling

1.1. Content Coupling : Không

1.2. Common Coupling : Không

1.3. Control Coupling : Không

1.4. Stamp Coupling :

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| PlaceRushOrderController | Phương thức CalculateShippingFee() trong controller PlaceOrderController  Text  Description automatically generated  Phương thức này truyền vào đối tượng order tuy nhiên chỉ sử dụng 1 phương thức của order là getAmount(), phương thức này return giá trị giỏ hàng hiện tại. | Thay vì truyền vào 1 đối tượng của lớp Order thì truyền vào int amount, Sau đó chỉ cần tìm đến phương thức calculate và truyền vào phương thức getAmount() của Order. |

* 1. Data Coupling:

Tính ra cái coupling này ok hơn mà tôi không tìm ra ông ạ =))

1. Cohesion

2.1. Coincidental Cohesion :

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| Utils | Các phương thức trong utils không liên quan đến module nào cả và không liên quan đến nhau, mà là các phương thức được sử dụng trong các modules khác | Không |

2.2. Logical Cohesion : Không có

2.3. Temporal Cohesion : Không

2.4. Procedual Cohesion : Không

2.5. Communicational Cohesion: Không

2.6. Sequential Cohesion :

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| API | Output của setupConnection() là input của get và post và readResponse() | Không |

2.7. Functional Cohesion:

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| PlaceRushOrderController | Các phương thức trong controller này là cần thiết cho mục đích của controller | Không |

**II – Nguyên lý SOLID**

1. Single Responsibility Principle

Mỗi lớp chịu 1 trách nhiệm, thiết kế hiện tại đã tốt

1. Open/Closed Principle

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| PlaceRushOrderController | Phương thức calculateShippingFee() nếu ta muốn tính phí ship theo kiểu khác, hoặc cần lưu nhiều kiểu tính phí ship khác nhau, thì chúng ta sẽ phải sửa đoạn code trên bằng một đoạn code với cách tính hoàn toàn khác. Điều này vi phạm nguyên tắc open/closed | tạo ra một interface ShippingFeeCalculator với phương thức trừu tượng là calculateShippingFee. Khi chúng ta muốn thêm một hay nhiều cách tính phí ship thì chúng ta chỉ cần viết thêm một lớp mới implement interface trên. Và tại PlaceRushOrderController chúng ta khởi tạo một đối tượng là interface ShippingFeeCalculator với instance là loại chiến lược tính phí ship mà chúng ta muốn. |

1. Liskov substitution principle

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| Media | phương thức Media.getAllMedia() được kì vọng trả về một List, tất cả các class con override lại phương thức này nhưng lại trả về null | xoá đoạn code Override getAllMedia() trên Các Class con của Media :DVD,CD,Book |

1. Interface segregation principle

Không

1. Dependency inversion principle

|  |  |  |
| --- | --- | --- |
| Related modules | Description | Improvement |
| CreditCard | PaymentTransaction đang phụ thuộc chặt chẽ vào lớp CreditCard, sau này giả sử không sử dụng CreditCard để thanh toán mà sử dụng một loại phương thức thanh toán khác, ví dụ như domestic debit card | Tạo một lớp abstract là PaymentCard và lớp PaymentTransaction chỉ quan tâm đến lớp PaymentMethod này |