vi phạm Liskov Substitution Principle (LSP):

public class AuthenticationController extends BaseController {

public boolean isAnonymousSession() {

Lí do vi phạm:

Lớp AuthenticationController mở rộng từ BaseController, nhưng không sử dụng bất kỳ hàm kế thừa nào từ BaseController.

Việc kế thừa từ BaseController mà không sử dụng bất kỳ hàm nào từ lớp cơ sở không phản ánh mối quan hệ "is-a" chính xác, mà là một dấu hiệu của vi phạm LSP.

Điều này có thể gây nhầm lẫn và làm mất đi tính linh hoạt trong việc mở rộng và tái sử dụng mã nguồn.

Để khắc phục vi phạm này, bạn có thể xem xét việc cắt bỏ kế thừa từ BaseController nếu không cần thiết, hoặc tái cấu trúc lớp để sử dụng các phương thức được kế thừa một cách hợp lý.

vi phạm của Interface Segregation Principle (ISP) trong lớp PaymentController:

public class PaymentController extends BaseController {

Lí do vi phạm:

Lớp PaymentController kế thừa từ BaseController, nhưng không sử dụng bất kỳ hàm kế thừa nào từ BaseController.

Điều này làm cho lớp PaymentController phụ thuộc vào các phương thức không cần thiết và không liên quan từ BaseController.

Việc này làm cho việc sử dụng PaymentController trở nên rối rắm và có thể dẫn đến việc thừa thãi hoặc phụ thuộc vào các phương thức không cần thiết.

Để sửa vi phạm này, lớp PaymentController nên được thiết kế sao cho chỉ kế thừa từ lớp cơ sở nếu có mối quan hệ "is-a" chặt chẽ và cần thiết. Nếu không,

nên tránh kế thừa và thay vào đó sử dụng composition hoặc chia nhỏ thành các lớp nhỏ hơn và có thể tái sử dụng được.

Vi phạm DIP:

Lớp PaymentController trực tiếp tạo một thể hiện của CreditCard, một lớp cụ thể, thay vì phụ thuộc vào một lớp trừu tượng hoặc một giao diện. Việc này làm cho PaymentController ràng buộc với một lớp cụ thể, làm giảm tính linh hoạt và khả năng mở rộng của mã nguồn. Thay vào đó, nó nên phụ thuộc vào một giao diện hoặc một lớp trừu tượng, để dễ dàng thay đổi hoặc mở rộng hơn trong tương lai.

vi phạm Liskov Substitution Principle (LSP):

public class BookDAO extends MediaDAO {

Lí do vi phạm:

Vi phạm LSP xảy ra khi lớp con BookDAO kế thừa từ lớp cha MediaDAO nhưng không thể sử dụng lại các phương thức của lớp cha một cách đúng đắn.

Nếu có sự khác biệt quan trọng giữa cách mà các phương thức của MediaDAO được sử dụng và cách mà chúng cần phải được sử dụng trong BookDAO, điều này dẫn đến vi phạm của LSP.

Vi phạm Liskov Substitution Principle (LSP):

public class CDDAO extends MediaDAO {

Lí do vi phạm:

Vi phạm LSP xảy ra khi lớp con CDDAO kế thừa từ lớp cha MediaDAO nhưng không thể sử dụng lại các phương thức của lớp cha một cách đúng đắn.

Nếu có sự khác biệt quan trọng giữa cách mà các phương thức của MediaDAO được sử dụng và cách mà chúng cần phải được sử dụng trong CDDAO, điều này dẫn đến vi phạm của LSP.

src/main/java/dao/media/MediaDAO.java
Vi phạm Liskov Substitution Principle (LSP) giống vơi CDDAO

src/main/java/dao/media/DVDDAO.java

Vi phạm Liskov Substitution Principle (LSP) giống vơi CDDAO

Vi phạm Liskov Substitution Principle (LSP):

```
public void checkAvailabilityOfProduct() throws SQLException{
    boolean allAvailable = true;
    for (Object object : lstCartItem) {
        CartItem cartItem = (CartItem) object;
        int requiredQuantity = cartItem.getQuantity();
        int availQuantity = cartItem.getMedia().getQuantity();
        if (requiredQuantity > availQuantity) allAvailable = false;
    }
    if (!allAvailable) throw new MediaNotAvailableException("Some media not available");
}
```

Lí do vi phạm:

Phương thức checkAvailabilityOfProduct chịu trách nhiệm kiểm tra sự khả dụng của sản phẩm trong giỏ hàng. Tuy nhiên, nó không nên trực tiếp xử lý các ngoại lệ SQLException. Việc này làm cho Cart phụ thuộc vào cơ sở dữ liệu và các chi tiết triển khai của nó, làm giảm tính linh hoạt và khả năng kiểm thử.

Vi phạm Interface Segregation Principle (ISP):

```
public List getListMedia(){
    return lstCartItem;
}
```

Lí do vi phạm:

Phương thức getListMedia trả về một List, không cung cấp giao diện tinh gọn cho khách hàng. Trong trường hợp này, nếu khách hàng chỉ muốn biết số lượng

phương tiện trong giỏ hàng, việc trả về một danh sách đầy đủ các CartItem là không cần thiết và có thể làm rối.

Vi phạm Interface Segregation Principle (ISP):

```
public List getListOrderMedia() {
    return this.orderMediaList;
}
```

Lí do vi phạm:

Phương thức getListOrderMedia trả về một List, không cung cấp giao diện tinh gọn cho khách hàng. Thay vào đó, nó có thể trả về một List<OrderItem> để giảm bớt sự phụ thuộc của khách hàng vào chi tiết triển khai của Order và làm cho giao diện của Order dễ hiểu hơn.

Vi phạm Dependency Inversion Principle (DIP):

```
for (Object object : SessionInformation.cartInstance.getListMedia()) {
    CartItem cartItem = (CartItem) object;
    OrderItem orderItem = new OrderItem(cartItem.getMedia(),
            cartItem.getQuantity(),
            cartItem.getPrice());
    orderItems.add(orderItem);
}
```

Lí do vi phạm:

Lớp Order hiện tại phụ thuộc trực tiếp vào SessionInformation để truy cập Cart và CartItem. Điều này làm giảm tính linh hoạt và gây ra mối quan hệ chặt chẽ giữa Order và SessionInformation, làm cho việc kiểm thử và bảo trì trở nên khó khăn hơn. Thay vào đó, Order nên nhận Cart và CartItem thông qua các giao diện hoặc tham số truyền vào để giảm bớt sự phụ thuộc cụ thể.

Vi phạm Dependency Inversion Principle (DIP):

import entity.payment.CreditCard;

import entity.payment.PaymentTransaction;

Lí do vi phạm:

Lớp InterbankPayloadConverter phụ thuộc trực tiếp vào các lớp cụ thể CreditCard và PaymentTransaction. Điều này làm cho lớp InterbankPayloadConverter rất cụ thể và không linh hoạt, vì nó không thể sử dụng bất kỳ loại thẻ thanh toán hoặc giao dịch thanh toán nào khác ngoài CreditCard và PaymentTransaction. Thay vào đó, nên sử dụng các giao diện hoặc lớp trừu tượng để giảm sự phụ thuộc vào chi tiết triển khai cụ thể và tăng tính linh hoạt của mã nguồn.

Vi phạm Interface Segregation Principle (ISP):

import common.exception.*;

import entity.payment.CreditCard;

import entity.payment.PaymentTransaction;

import utils.MyMap;

Lí do vi phạm:

ISP khuyến khích việc phân tách các giao diện thành các giao diện nhỏ hơn và tập trung vào nhu cầu cụ thể của người sử dụng. Trong trường hợp này, nếu InterbankPayloadConverter cung cấp các phương thức không cần thiết hoặc không liên quan đến các giao diện, có thể gây ra vi phạm của ISP. Tuy nhiên, không có vi phạm cụ thể nào trong mã nguồn InterbankPayloadConverter.