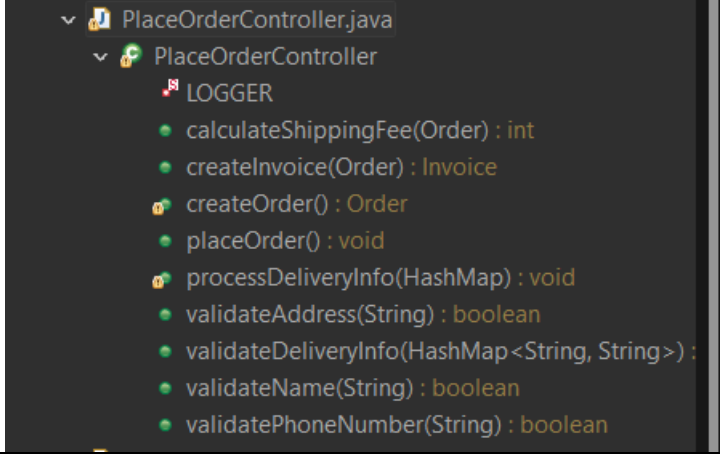


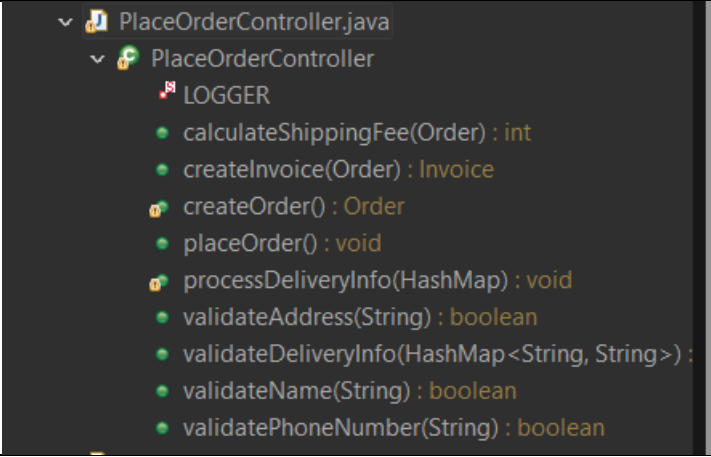
DESIGN PRINCIPLES

1. SINGLE RESPONSIBILITY PRINCIPLE

#	Related modules	Description	Improvement
1.1	Class PlaceOrderController	<p>Class PlaceOrderController phải thực hiện cả việc createOrder, validateInfo và calculateShippingFee, => vi phạm nguyên lý S</p> <pre>PlaceOrderController.java PlaceOrderController LOGGER calculateShippingFee(Order) : int createInvoice(Order) : Invoice createOrder() : Order placeOrder() : void processDeliveryInfo(HashMap) : void validateAddress(String) : boolean validateDeliveryInfo(HashMap<String, String>) : validateName(String) : boolean validatePhoneNumber(String) : boolean</pre>	Tách các task này thành các lớp khác, trong lớp PlaceOrderController chỉ cần gọi đến các lớp này.
1.2	Các class entity	<p>Có cả các phương thức để truy xuất vào database để lấy dữ liệu => vi phạm nguyên lý S</p> <pre>@Override public Media getMediaById(int id) throws SQLException { String sql = "SELECT * FROM " + "aims.Book " + "INNER JOIN aims.Media " + "ON Media.id = Book.id " + "where Media.id = " + id + ";"; Statement stm = AIMSDB.getConnection().createStatement(); ResultSet res = stm.executeQuery(sql); if(res.next()) { // from Media table String title = ""; String type = res.getString("type"); int price = res.getInt("price"); String category = res.getString("category"); int quantity = res.getInt("quantity"); // from Book table String author = res.getString("author"); String coverType = res.getString("coverType"); String publisher = res.getString("publisher"); Date publishDate = res.getDate("publishDate"); int numOfPages = res.getInt("numOfPages"); String language = res.getString("language"); String bookCategory = res.getString("bookCategory"); return new Book(id, title, category, price, quantity, type, author, coverType, publisher, publishDate, numOfPages, language,); } else { throw new SQLException(); } }</pre>	Tách phần truy xuất vào database thành một class riêng.

2. OPEN/CLOSED PRINCIPLE

#	Related modules	Description	Improvement
2.1	Class PlaceOrderController	<p>Phương thức validateInfo có nhiều phương thức cấp thấp hơn là validateAddress, validateName, validatePhone, như vậy khi cần bổ sung thêm phương thức validate mới thì phải sửa vào trong lớp cũ => vi phạm nguyên lý O</p>  <pre> PlaceOrderController.java PlaceOrderController LOGGER calculateShippingFee(Order) : int createInvoice(Order) : Invoice createOrder() : Order placeOrder() : void processDeliveryInfo(HashMap) : void validateAddress(String) : boolean validateDeliveryInfo(HashMap<String, String>) : validateName(String) : boolean validatePhoneNumber(String) : boolean </pre>	<p>Xây dựng một interface validate và các lớp thực thi validate cho từng thông tin, như vậy sau này muốn bổ sung validate cho một thông tin khác, chúng ta chỉ cần xây dựng thêm một lớp khác thực thi interface validate mà ko cần sửa vào những lớp khác.</p>
2.2	Class PlaceOrderController	<p>Phương thức calculateShippingFee được viết trực tiếp trong lớp PlaceOrderController, như vậy sau này khi thay đổi cách tính phí ship, sẽ phải sửa trực tiếp vào lớp cũ => vi phạm nguyên lý O</p>	<p>Xây dựng một interface calculate và một lớp thực thi việc calculate shipping fee.</p>

			
2.3	Class CartScreenHandler	Trong phương thức requestToPlaceOrder, sử dụng trực tiếp đến PlaceOrderController dẫn đến việc thêm các phương thức để thực thi cho chức năng PlaceRushOrder gặp khó khăn, phải thay đổi trong lớp cũ.	Tách thành một lớp khác thực hiện việc requestToPlaceOrder, khi có thêm yêu cầu RushOrder thì tạo lớp khác là extend lớp cũ.

3. LISKOV SUBSTITUTION PRINCIPLE

#	Related modules	Description	Improvement
3,1			

4. INTERFACE SEGREGATION PRINCIPLE

#	Related modules	Description	Improvement
4.1			

5. DEPENDENCY INVERSION PRINCIPLE

#	Related modules	Description	Improvement
5.1	Class CartScreenHandler	Tương tự như 2.3, trong phương thức requestToPlaceOrder đang fix cứng là gọi đến PlaceOrderController, nên khi muốn PlaceRushOrder khó khăn trong việc thay đổi	Giải pháp tương tự 2.3

		<pre> • public void requestToPlaceOrder() throws SQLException, IOException { try { // create placeOrderController and process the order PlaceOrderController placeOrderController = new PlaceOrderController(); if (placeOrderController.getListCartMedia().size() == 0){ PopupScreen.error("You don't have anything to place"); return; } placeOrderController.placeOrder(); // display available media displayCartWithMediaAvailability(); // create order Order order = placeOrderController.createOrder(); // display shipping form ShippingScreenHandler shippingScreenHandler = new ShippingScreenHandler(this.stage, Config); shippingScreenHandler.setPreviousScreen(this); shippingScreenHandler.setHomeScreenHandler(homeScreenHandler); shippingScreenHandler.setScreenTitle("Shipping Screen"); shippingScreenHandler.setController(placeOrderController); shippingScreenHandler.show(); } catch (MediaNotAvailableException e) { // if some media are not available then display cart and break <u>usecase</u> Place Order displayCartWithMediaAvailability(); } } </pre>	
5.2	Class PaymentController	<p>Phụ thuộc trực vào class CreditCard, vì vậy khi thay đổi phương thức thanh toán cần phải sửa trực tiếp vào trong lớp này</p> <pre> public class PaymentController extends BaseController { /** * Represent the card used for payment */ private CreditCard card; /** * Represent the <u>Interbank</u> subsystem */ private InterbankInterface interbank; /** * Validate the input date which should be in the format "<u>mm/yy</u>", * return a {@link java.lang.String String} representing the date * @param inputDate input date */ } </pre>	
5.3	Class AIMSDB	<p>Trong phương thức getConnection đang fix cứng là sử dụng sqlite, vì vậy đang bị phụ thuộc chi tiết vào cơ sở dữ liệu, sau này nếu thay đổi DBMS thì sẽ gặp khó khăn</p> <pre> public static Connection getConnection() { if (connect != null) { return connect; } try { Class.forName("org.sqlite.JDBC"); String url = "jdbc:sqlite:assets/db/aims.db"; connect = DriverManager.getConnection(url); LOGGER.info("Connect database successfully"); } catch (Exception e) { LOGGER.info(e.getMessage()); } return connect; } </pre>	<p>Xây dựng interface có các phương thức chung để truy xuất vào cơ sở dữ liệu, sau đó tùy vào việc sử dụng loại DBMS nào thì sẽ xây dựng các lớp để thực thi interface này.</p>

