

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> @throws SQLException */ public void placeOrder() throws SQLException {...} /** * This method creates the new Order based on the Cart * * @return Order * @throws SQLException */ public Order createOrder() throws SQLException {...} /** * This method creates the new Invoice based on order * * @param order * @return Invoice */ public Invoice createInvoice(Order order) {...} You, 2 months ago • Release lab 08 /** * This method takes responsibility for processing the shipping info from user * * @param info * @throws InterruptedException * @throws IOException */ public void processDeliveryInfo(HashMap<String, String> info) throws InterruptedException, IOException {...} /** * The method validates the info * * @param info * @throws InterruptedException * @throws IOException */ public void validateDeliveryInfo(HashMap<String, String> info) throws InterruptedException, IOException {...} /** * This method calculates the shipping fees of order * * @param order * @return shippingFee */ public int calculateShippingFee(Order order) {...} /** * This method get product available place rush order media * * @param order * @return media * @throws SQLException */ public Media getProductAvailablePlaceRush(Order order) throws SQLException {...} public boolean validateAddressPlaceRushOrder(String province, String address) {...} public boolean validateMediaPlaceRushOrder() {...} </pre>	Tách các task này thành các lớp khác, trong lớp PlaceOrderController chỉ cần gọi đến các lớp này.
1. 2	Các class entity	Có cả các phương thức để truy xuất vào database để lấy dữ liệu => vi phạm nguyên lý S	Tách phần truy xuất vào database thành một class riêng.

		<pre> } @Override public String toString() { return "(" + super.toString() + " artist=" + artist + " ", recordLabel=" + recordLabel + " " + " " + ", musicType=" + musicType + " " + ", releasedDate=" + releasedDate + " " + ")"; } @Override public Media getMediaById(int id) throws SQLException { String sql = "SELECT * FROM " + "aims.CD " + "INNER JOIN aims.Media " + "ON Media.id = CD.id " + "where Media.id = " + id + ";"; ResultSet res = stmt.executeQuery(sql); if(res.next()) { // from media table String title = ""; String type = res.getString(columnLabel: "type"); int price = res.getInt(columnLabel: "price"); String category = res.getString(columnLabel: "category"); int quantity = res.getInt(columnLabel: "quantity"); // from CD table String artist = res.getString(columnLabel: "artist"); String recordLabel = res.getString(columnLabel: "recordLabel"); String musicType = res.getString(columnLabel: "musicType"); Date releasedDate = res.getDate(columnLabel: "releasedDate"); return new CD(id, title, category, price, quantity, type, artist, recordLabel, musicType, releasedDate); } else { throw new SQLException(); } } @Override public List getAllMedia() { return null; } } </pre>	
--	--	---	--

2. OPEN/CLOSED PRINCIPLE

#	Related modules	Description	Improvement
2. 1	Class PlaceOrderContro ller	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	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

		<pre> */ public void validateDeliveryInfo(HashMap<String, String> info) throws InterruptedException, IOException { ... } /** * This method calculates the shipping fees of order * * @param order * @return shippingFee */ public int calculateShippingFee(Order order) { ... } /** * This method get product available place rush order media * * @param order * @return media * @throws SQLException */ public Media getProductAvailablePlaceRush(Order order) throws SQLException { ... } public boolean validateAddressPlaceRushOrder(String province, String address) { ... } public boolean validateMediaPlaceRushOrder() { ... } </pre>	cần sửa vào những lớp khác.
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> <pre> 84 */ 85 > public int calculateShippingFee(Order order) { 86 ... 87 } 88 89 /** 90 * This method get product available place rush order media </pre>	Xây dựng một interface calculate và một lớp thực thi việc calculate shipping fee.
2.3	Class CartScreenHandler	<p>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.</p>	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	Improve ment
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

```

public void requestToPlaceOrder() throws SQLException, IOException {
    try {
        // create placeOrderController and process the order
        PlaceOrderController placeOrderController = new PlaceOrderController();
        if (placeOrderController.getListCartMedia().size() == 0){
            PopupScreen.error(message: "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, Confi
        ShippingScreenHandler.setPreviousScreen(this);
        ShippingScreenHandler.setHomeScreenHandler(homeScreenHandler);
        ShippingScreenHandler.setScreenTitle(string: "Shipping Screen");
        ShippingScreenHandler.setBController(placeOrderController);
        ShippingScreenHandler.show();

    } catch (MediaNotAvailableException e) {
        // if some media are not available then display cart and break usecase Place Order
        displayCartWithMediaAvailability();
    }
}

```

5. 2	Class PaymentCont roller	Phụ thuộc trực vào class CreditCard, vì vậy khi thay đổi phương thức thanh toán cần phải sửa trực tiếp vào trong lớp này	
---------	--------------------------------	--	--

		<pre> 21 * @author hieud 22 * 23 */ 24 public class PaymentController extends BaseController { 25 26 /** 27 * Represent the card used for payment 28 */ 29 private CreditCard card; 30 31 /** 32 * Represent the Interbank subsystem 33 */ 34 private InterbankInterface interbank; </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> 9 public class AIMSDB { 10 11 private static Logger LOGGER = Utils.getLogger(Connection.class.getName()); 12 private static Connection connect; 13 14 public static Connection getConnection() { 15 if (connect != null) return connect; 16 try { 17 Class.forName(className: "org.sqlite.JDBC"); 18 String url = "jdbc:sqlite:assets/db/aims.db"; 19 connect = DriverManager.getConnection(url); 20 LOGGER.info(msg: "Connect database successfully"); 21 } catch (Exception e) { 22 LOGGER.info(e.getMessage()); 23 } 24 return connect; 25 } 26 27 </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</p>

			interface này.
--	--	--	----------------