Design concepts

(ver 2)

1. Coupling

1.1. Content coupling

Related modules	Description	Improvement
Order, RushOrder	Attribute deliveryInfo is a HashMap, which can be modified with its function but can be accessed from outside using getDeliveryInfo()	Only returning the data of the Hashmap needed, not the whole deliveryInfo object

1.2. Common coupling

Related modules	Description	Improvement
Cart	Cart is used in multiple modules at the same time (PlaceOrderController, CartScreenHandler, ..), so simultaneous access may occur, which harm the integration of the data	- Apply singleton pattern - Use locking mechanism such as mutex, semaphore,..

1.3. Control coupling

Related modules	Description	Improvement
CartScreenHandler	Function requestToPlaceOrder(bool isRush) needs the boolean flag to determine which route to execute	Separate it into 2 functions, requestToPlaceOrder() and requestToPlaceRushOrder() with each responds to a different button

1.4. Stamp coupling

Related modules	Description	Improvement
cartMedia	Passing the whole Cart object to cartMedia is excessive	Only use the necessary params while passing into cartMedia: media, quantity, price

1.5. Data coupling

Related modules	Description	Improvement
PlaceOrderController, PlaceRushOrderController	The two controllers	

2. Cohesion

2.1. Coincidental cohesion

Related modules	Description	Improvement
.utils packages	Utilities of all kinds are currently held in the same package	If the util is module-specific, we should place it in that specific module, not in the common utils

2.2. Logical cohesion

Related modules	Description	Improvement
PlaceRushOrderController	The controller takes in the info and then validates, and then creates an invoice and a rush order controller based on the information. All the code for these functions is in the same component. Operations are related, but the functions are significantly	

	different.	

2.3. Temporal cohesion

Related modules	Description	Improvement

2.4. Procedural cohesion

Related modules	Description	Improvement

2.5. Communication cohesion

Related modules	Description	Improvement
InterbankSubsytem	InterbankSubsystemController has pay and refund method, which both works on the same input and returns the same type PaymentTransaction of output data	

2.6. Sequential cohesion

Related modules	Description	Improvement
PlaceRushOrderController	In the validateDeliveryInfo method, we validate the data fields one by one with different methods	

2.7. Informational cohesion

Related modules	Description	Improvement

2.8. Functional cohesion

Related modules	Description	Improvement
API	The output of one method is used as the input of another	

Design principles

1. Coupling Single Responsibility Principle

Related modules	Description	Improvement
InterbankSubsytem	Currently handle 2 tasks at once: - Input validation - Data processing	Divide into 2 classes which handle the 2 corresponding tasks
PlaceOrderController	Current have 3 tasks: - Create order - Validate delivery information - Calculate shipping fees	Use 3 interfaces to implement the 3 tasks
PlaceRushOrderController	Same as above	Same as above

2. Open/ Closed Principle

Related modules	Description	Improvement

PlaceRushOrderController	Currently use the same code of normal order with some small modification	Use an additional interface to separate the two logic
PlaceRushOrderController	Same problem as above with processDeliveryInfo()	Use the same way as above

3. Liskov Substitution Principle

Related modules	Description	Improvement
Media	Function getAllMedia() currently returns List<> but children classes override and return null instead	Remove redundant methods in children classes

4. Interface Segregation Principle

Related modules	Description	Improvement
InterbankSubsystem	We should have payOrder() and refund() for PaymentSubsystem	Put the 2 methods into the same interface and let both modules implement it

5. Dependency Inversion Principle

Related modules	Description	Improvement
PaymentTransaction, CreditCard	Currently, we need to modify PaymentTransaction if we need to add a new type of card	Make an abstract class so that all type of cards can extend it
PlaceOrderController, PlaceRushOrderController, ShippingFeeCalculator	Must modify code in order to change new formula	Make an abstract class as the parent of all kind of shipping fee