Design for class “OrderUI”

Table 1. Example of attribute design

#	Name	Data type	Default value	Description
1	order	Order	null	Đối tượng đơn hàng được yêu cầu đặt hàng nhanh
2	totalCost	double	0.0	Tổng chi phí của đơn hàng, bao gồm phí giao hàng nhanh

Table 2. Example of operation design

#	Name	Return type	Description (purpose)
1	askToPlaceRushOrder(order)	void	Yêu cầu đặt đơn hàng nhanh
2	displayTotalCostOnInterface()	void	Hiển thị tổng chi phí đơn hàng trên giao diện
3	userConfirmsPayment()	void	Người dùng xác nhận thanh toán đơn hàng

Parameter

- `order`: Đơn hàng mà người dùng muốn đặt với dịch vụ giao hàng nhanh.
- `totalCost`: Tổng chi phí, bao gồm phí vận chuyển nhanh.

Exception

- `OrderNotFoundException`: Nếu đơn hàng không tồn tại hoặc không hợp lệ.
- `PaymentFailedException`: Nếu thanh toán không thành công.

Method

- `askToPlaceRushOrder(order)`: Kích hoạt quy trình đặt hàng nhanh.
- `displayTotalCostOnInterface()`: Hiển thị tổng chi phí của đơn hàng cho người dùng.
- `userConfirmsPayment()`: Xác nhận thanh toán đơn hàng.

How to use parameters / attributes

- `order` được truyền vào khi người dùng yêu cầu đặt hàng nhanh.
- `totalCost` được tính toán và hiển thị trên giao diện trước khi xác nhận thanh toán.

Flowchart / Activity diagram / Sequence diagram

- Có thể vẽ **sequence diagram** để thể hiện quy trình đặt hàng nhanh và xác nhận thanh toán.

State

- **Order Pending** → Khi người dùng chưa xác nhận thanh toán.
- **Order Confirmed** → Khi người dùng xác nhận thanh toán thành công.

State diagram (if any)

- Có thể vẽ sơ đồ trạng thái để mô tả quá trình chuyển đổi giữa các trạng thái của đơn hàng trong giao diện.

Design for class "PlaceRushOrderController"

Table 1. Example of attribute design

#	Name	Data type	Default value	Description
1	order	Order	null	Đối tượng đơn hàng được yêu cầu đặt hàng nhanh
2	totalCost	int	0	Tổng chi phí đơn hàng, bao gồm phí vận chuyển nhanh

Table 2. Example of operation design

#	Name	Return type	Description (purpose)
1	askToPlaceRushOrder(order)	void	Yêu cầu đặt đơn hàng nhanh
2	returnsTotalCostIncludingExpeditedShipping(order)	int	Trả về tổng chi phí đơn hàng, bao gồm phí vận chuyển nhanh
3	submitRequestToContinuePayment()	void	Gửi yêu cầu tiếp tục thanh toán

Parameter

- `order`: Đơn hàng mà người dùng muốn đặt với dịch vụ giao hàng nhanh.
- `totalCost`: Tổng chi phí, bao gồm phí vận chuyển nhanh.

Exception

- `OrderNotFoundException`: Nếu đơn hàng không tồn tại hoặc không hợp lệ.
- `PaymentProcessingException`: Nếu có lỗi xảy ra trong quá trình xử lý thanh toán.

Method

- `askToPlaceRushOrder(order)`: Xử lý yêu cầu đặt đơn hàng nhanh.
- `returnsTotalCostIncludingExpeditedShipping(order)`: Tính toán tổng chi phí đơn hàng.
- `submitRequestToContinuePayment()`: Gửi yêu cầu xử lý thanh toán.

How to use parameters / attributes

- `order` được truyền vào khi người dùng yêu cầu đặt hàng nhanh.
- `totalCost` được tính toán dựa trên đơn hàng và hiển thị cho người dùng trước khi thanh toán.

Flowchart / Activity diagram / Sequence diagram

- Có thể vẽ **sequence diagram** để thể hiện quá trình đặt hàng nhanh và xác nhận thanh toán.

State

- **Order Pending** → Khi đơn hàng đang chờ xác nhận.
- **Order Confirmed** → Khi đơn hàng được xác nhận.
- **Payment Processing** → Khi thanh toán đang được xử lý.

State diagram (if any)

- Có thể vẽ sơ đồ trạng thái để mô tả quá trình xử lý đơn hàng nhanh từ lúc yêu cầu đến khi hoàn tất thanh toán.

Design for class “AIMS”

Table 1. Example of attribute design

#	Name	Data type	Default value	Description
1	payment	Payment	null	Đối tượng thanh toán liên quan đến đơn hàng

Table 2. Example of operation design

#	Name	Return type	Description (purpose)
1	callPayOrderUseCase(payment)	void	Gọi Use Case “Pay Order” để xử lý thanh toán
2	returnsPaymentResult(payment)	String	Trả về kết quả thanh toán (thành công hoặc thất bại)

Parameter

- `payment`: Đối tượng chứa thông tin thanh toán của đơn hàng.

Exception

- `PaymentFailedException`: Nếu quá trình thanh toán không thành công.
- `InvalidPaymentException`: Nếu thông tin thanh toán không hợp lệ.

Method

- `callPayOrderUseCase(payment)`: Kích hoạt quá trình thanh toán đơn hàng.
- `returnsPaymentResult(payment)`: Kiểm tra trạng thái thanh toán và trả về kết quả.

How to use parameters / attributes

- `payment` được truyền vào khi người dùng muốn thanh toán đơn hàng.
- Kết quả được trả về dưới dạng chuỗi `"success"` hoặc `"failure"`.

Flowchart / Activity diagram / Sequence diagram

- Có thể vẽ **sequence diagram** mô tả quá trình thực hiện thanh toán.

State

- **Payment Pending** → Khi thanh toán chưa được xử lý.
- **Payment Processing** → Khi thanh toán đang được thực hiện.
- **Payment Successful** → Khi thanh toán thành công.
- **Payment Failed** → Khi thanh toán thất bại.

State diagram (if any)

- Có thể vẽ sơ đồ trạng thái để thể hiện quá trình xử lý thanh toán từ lúc bắt đầu đến khi hoàn tất

Design for class "Order"

Table 1. Example of attribute design

#	Name	Data type	Default value	Description
1	payment	Payment	null	Đối tượng thanh toán liên quan đến đơn hàng

Table 2. Example of operation design

#	Name	Return type	Description (purpose)
1	Get order list from system	void	Lấy DS order
2	Request to calculate total cost of order	String	Yêu cầu tính tổng

Design for class “Payment”

Table 1. Example of attribute design

#	Name	Data type	Default value	Description
1	payment	Payment	null	Đối tượng thanh toán liên quan đến đơn hàng

Table 2. Example of operation design

#	Name	Return type	Description (purpose)
1	Perform payment processing	Void	Hiển thị hóa đơn