Design for class “Order”

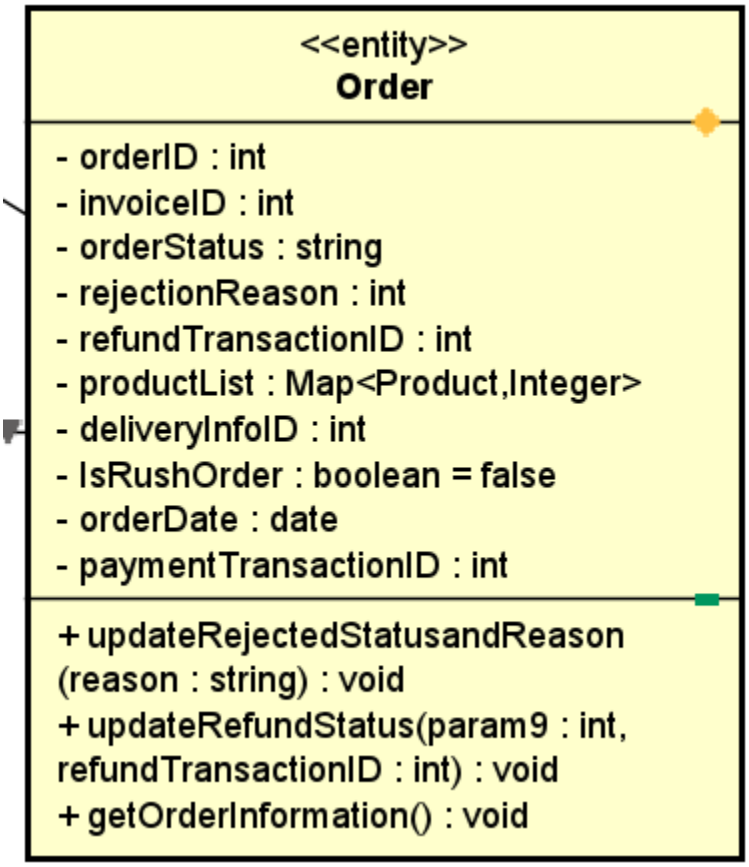


Figure 1: Design Class of Order

Table 1: Attribute design of Order

	Name	Data Type	Default value	Description
1	orderID	int	N/A	Unique identifier for the order
2	invoiceID	int	N/A	The corresponding invoice
3	productList	Map<Product, int>	N/A	A list of media items associated with the order
4	isRushOrder	boolean	N/A	A flag indicating whether the order is a rush order (true or false)
5	orderStatus	char	N/A	Status of the order

6	deliveryInfoID	intr	N/A	The corresponding delivery information ID
7	paymentTransactionID	int	N/A	The transaction ID used for refunds, if applicable
8	orderDate	date	N/A	The date when the order was placed
9	totalPrice	int		
10	totalItems	int		
11	curency	String		
12	refundTransactionID	Id		
13	rejectionReason	String		

Table 2: Operation design of Order

	Name	Return Type	Description
1	updateRejectedStatusandReason ()	void	Update the status of order to Rejected
2	updateRefundStatus ()	void	Update the status of order to Refund
3	getter()	void	Controllers get the information to create Invoice

1. updateRejectedStatusandReason ()

Parameter

Name	Default Value	Description
Reason		

Method

```
public void updateRejectedStatusAndReason(int reason) {
        this.status = "rejected";
        this.rejectionReason = reason;
    }
```

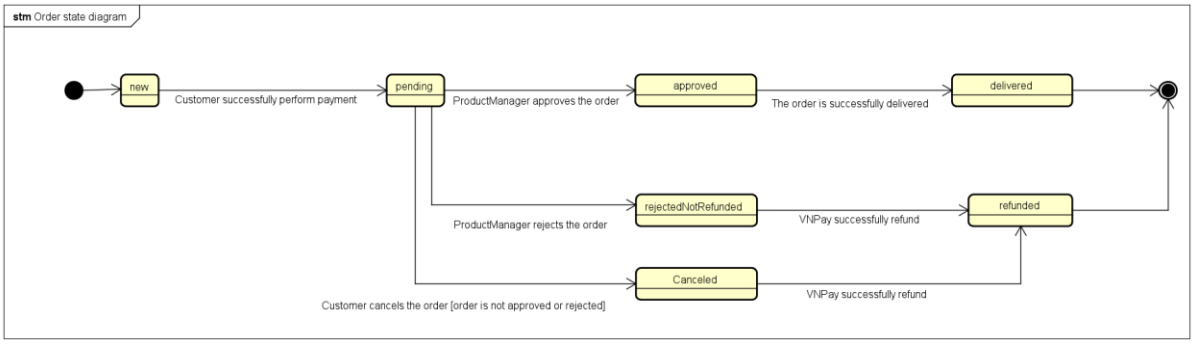
2. updateRefundStatus ()

Name	Default Value	Description
refundTransactionID		The payment transaction ID

Method

```
public void updateRefundStatus(String refundTransactionID) {
        this.status = "refunded";
        this.refundTransactionID = refundTransactionID;
    }
```

State Diagram of order



Design for class “RejectOrderController”

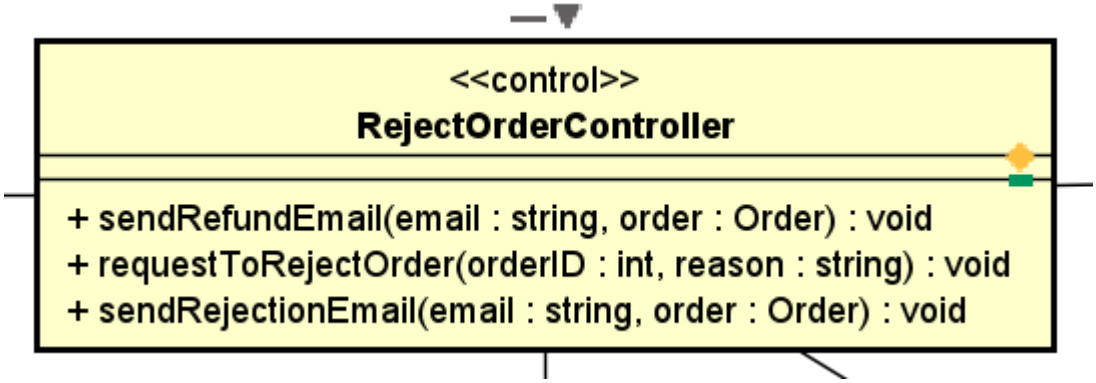


Figure 2: Design Class of RejectOrderController

Table 3: Operation design of RejectOrderController

	Name	Return Type	Description
1	sendRefundEmail ()	void	User submit the information
2	requestToRejectOrder()	void	Display the delivery information form
3	sendRejectionEmail ()	void	

1. sendRefundEmail ()

Parameter

Name	Default Value	Description
email		
order		

2. sendRejectionEmail

Parameter

Name	Default Value	Description
email		
order		

3. requestToRejectOrder()

Parameter

Name	Default Value	Description
orderID		
reason		

Exception

Name	Description
InvalidRejectionReasonException	
OrderNotFoundException	If orderID is not valid

Method

```
public void requestToRejectOrder(String reason, int orderID) throws
InvalidRejectionReasonException, OrderNotFoundException {
        if (reason == null || reason.isEmpty()) {
            throw new InvalidRejectionReasonException("Rejection reason cannot be empty.");
        } else {
            Order order = Order.get(orderID);
            if (order == null) {
                throw new OrderNotFoundException("Order with ID " + orderID + " not found.");
            }
            order.updateRejectedStatusandReason(reason);
        }
    }
```

Design for class “RefundTransaction”

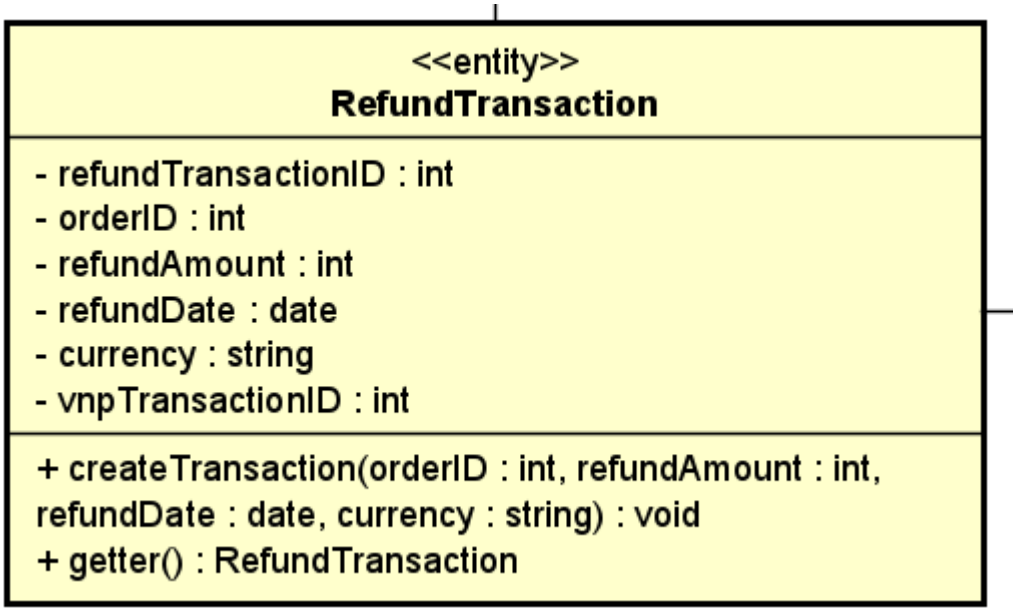


Figure 3: Design Class of RefundTransaction

Table 4: Attribute design of RefundTransaction

	Name	Data Type	Default value	Description
1	refundTransactionID	int	N/A	Unique identifier
2	orderID	int	N/A	The corresponding order
3	refundAmount	int	N/A	
4	refundDate	date	N/A	
5	currency	string	N/A	
6	vnpTransactionID	int	N/A	The transaction information from VNPay

Table 5: Operation design of RefundTransaction

	Name	Return Type	Description
1	createTransaction ()	RefundTransaction	Update the status of order to Rejected
2	getter ()	void	Update the status of order to Refund

1. createTransaction ()

Parameter

Name	Default Value	Description
orderID		

refundAmount		
refundDate	date	N/A
currency	string	N/A
vnpTransactionID	int	

Exception

Name	Description
OrderAlreadyRefundedException	If Order is already refunded
OrderNotFoundException	If orderID is not valid

Method

```
public void createTransaction(int orderID, int refundAmount, Date refundDate, String
currency)
        throws OrderAlreadyRefundedException, OrderNotFoundException {

    if (order.getOrderID() != orderID) {
        throw new OrderNotFoundException("Order with ID " + orderID + " not found.");
    }

    if (order.isRefunded()) {
        throw new OrderAlreadyRefundedException("Order with ID " + orderID + " has
already been refunded.");
    }

    RefundTransaction transaction = new RefundTransaction(orderID, refundAmount,
refundDate, currency);

}
```

Design for class “PendingOrderScreen”

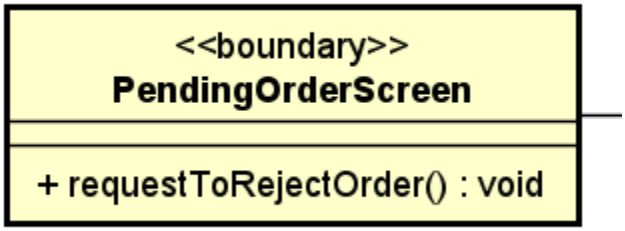


Figure 4: Design Class of PendingOrderScreen

Table 6: Operation design of PendingOrderScreen

	Name	Return Type	Description
1	requestToRejectOrder ()	void	

Design for class “PaymentGateway”

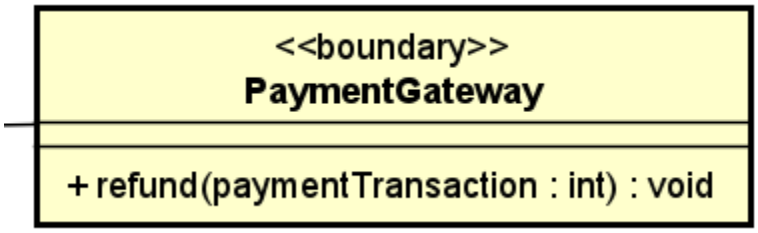


Figure 5: Design Class of InvoiceScreen

Table 7: Operation design of InvoiceScreen

	Name	Return Type	Description
1	refund ()	void	Request VNPay to refund the corresponding order

Parameter

Name	Default Value	Description
paymentTransaction		