**Case study: Order Processing System**

Consider the following problem description: A mail-order company wants to automate its order processing. The initial version of the order processing system should be accessible to customers via the web. Customers can also call the company by phone and interact with the system via a customer representative. It is highly likely that the company will enhance this system in upcoming years with new features. The system allows customers to place orders, check the status of their orders, cancel an existing order and request a catalog. Customers may also return a product but this is only possible through the phone, not available on the web. When placing an order, the customer identifies himself by means of customer number (only for existing registered customers) or by means of his name and address. He then selects a number of products by giving the product number or by selecting products from the online catalogue. For each product, information such as price, a description and a picture (only on demand as they are usually high-resolution images of large size) are presented to the customer. Also, the availability of the product is obtained from the inventory. The customer indicates whether he wants to buy the product and in what quantity. When all desired products have been selected, the customer provides a shipping address and a credit card number and a billing address (if different from the shipping address). Then an overview of the ordered products and the total cost are presented. If the customer approves, the order is submitted. Credit card number, billing address and a specification of the cost of the order are used on the invoice, which is forwarded to the accounting system (an existing software module). Orders are forwarded to the shipping company, where they are filled and shipped. Customers who spent over a certain amount within the past year are promoted to be gold customers. Gold customers have additional rights such as being able to return products in an extended time period as well as earning more bonus points with each purchase. In addition, in cases where a product is on back order, gold customers have the option to sign up for an email notification for when the particular product becomes available.

(1). Identify actors and use cases for the system described above and show them on a UML Use Case Diagram.

(2) Perform a quick application domain analysis to come up with an object model for the above system. Express your findings with a UML Class Diagram, making sure to identify any critical operations of classes.

Consider the following use case scenario (for use case “place order”):

*Ali is an existing customer of the order processing company described earlier, registered with their web site. Also assume that having browsed the printed catalogue he has, he already identified the two items (including their prices) he likes to buy from the company’s website using their product numbers (i.e. #2 and #9). First, he tries to buy one of product #2, but it is listed as unavailable in the inventory. Then, he adds two quantities of product #9, which turns out to be available, to his basket. He is then asked to confirm his registered shipping and billing addresses and credit card information from the customer database. He completes the order by clicking the Submit button. You may ignore processing of customer authentication.*

(3) draw a UML Sequence Diagram for this particular scenario. You may use any software/solution domain objects if needed as well.

1. **Identify actors and use cases**

Actors:

1. Customer (Existing and potential new customers)
2. Customer Representative (who takes orders over phone)
3. Accounting System
4. Shipping Company
5. Inventory System (which provides product availability status)

Use Cases:

1. Place an Order
2. Check Order Status
3. Cancel an Order
4. Request a Catalog
5. Return a Product (via phone only)
6. Register as a New Customer
7. Browse Catalog
8. Select Product
9. Confirm Purchase
10. Process Order (Forward to Accounting and Shipping Company)
11. Promote to Gold Customer
12. Send Email Notification (For back-order products for gold customers)

**PlantUML code:**

@startuml

' Actors

actor Customer

actor "Customer\nRepresentative" as CR

actor "Accounting\nSystem" as AS

actor "Shipping\nCompany" as SC

actor "Inventory\nSystem" as IS

rectangle "Order Processing System" {

' Customer use cases

Customer --> (Register as a New Customer)

Customer --> (Request a Catalog)

Customer --> (Browse Catalog)

Customer --> (Place an Order)

Customer --> (Check Order Status)

Customer --> (Cancel an Order)

Customer --> (Return a Product) : over phone

' "Place an Order" extensions

(Place an Order) .> (Select Product) : <<include>>

(Select Product) ..> (Confirm Purchase) : <<include>>

(Confirm Purchase) ..> (Process Order) : <<include>>

' Other actors connecting to their use cases

CR --> (Place an Order) : over phone

CR --> (Return a Product) : over phone

AS --> (Process Order)

SC --> (Process Order)

IS --> (Select Product)

' Notes and extensions for Gold Customer

note right of Customer : Gold Customers can\nreturn products\nin extended period and\nsign up for email notification.

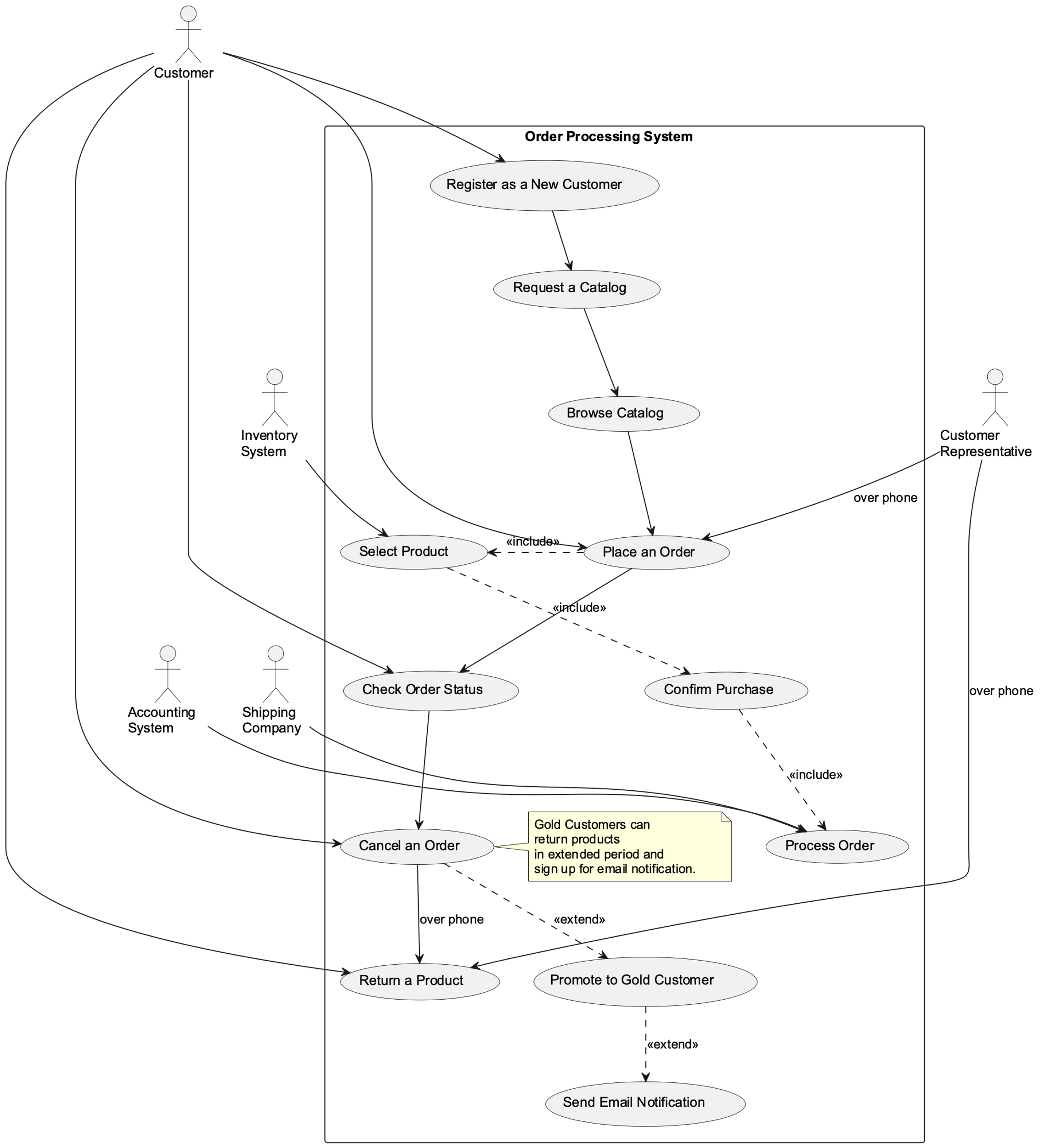
(Customer) ..> (Promote to Gold Customer) : <<extend>>

(Customer) ..> (Send Email Notification) : <<extend>>

}

@enduml

**Use Diagram (auto generated) :**



完成第一问的对话记录：

Model: GPT-4

浏览器访问：

<https://chat.openai.com/share/da0ab612-187f-4054-9e76-e6cc374528c1>

1. Perform a quick application domain analysis to come up with an object model for the above system. Express your findings with a UML Class Diagram, making sure to identify any critical operations of classes.

* **Customer**: Has attributes such as customerID (for registered customers), name, address, customerType (regular or gold), email (optional), and a list of Orders. Operations could include placeOrder(), checkOrderStatus(), cancelOrder(), requestCatalog(), and returnProduct().
* **Order**: Has attributes such as orderID, status, list of OrderLines (product and quantity), shippingAddress, billingAddress, and totalCost. Operations could include addProduct(), removeProduct(), changeQuantity(), calculateTotalCost(), and submitOrder().
* **Product**: Has attributes such as productID, price, description, picture, and availability. Operations could include getProductDetails(), getAvailability(), and updateAvailability().
* **Catalog**: Has a list of Products. Operations could include searchProduct(), listProducts(), and getProductDetails().
* **Inventory**: Has a list of Products. Operations could include checkAvailability() and updateAvailability().
* **ShippingCompany**: Has methods like processOrder() and shipOrder().
* **AccountingSystem**: This is an existing software module that will interact with the system. It has methods like generateInvoice().

PlantUML code:

@startuml

class Customer {

-customerID

-name

-address

-customerType

-email

+placeOrder()

+checkOrderStatus()

+cancelOrder()

+requestCatalog()

+returnProduct()

}

class Order {

-orderID

-status

-OrderLines[]

-shippingAddress

-billingAddress

-totalCost

+addProduct()

+removeProduct()

+changeQuantity()

+calculateTotalCost()

+submitOrder()

}

class Product {

-productID

-price

-description

-picture

-availability

+getProductDetails()

+getAvailability()

+updateAvailability()

}

class Catalog {

-Products[]

+searchProduct()

+listProducts()

+getProductDetails()

}

class Inventory {

-Products[]

+checkAvailability()

+updateAvailability()

}

class ShippingCompany {

+processOrder()

+shipOrder()

}

class AccountingSystem {

+generateInvoice()

}

Customer "1" -- "0..\*" Order : places

Order "0..\*" -- "1..\*" Product : contains

Customer -- Catalog : requests

Product -- Catalog : listed in

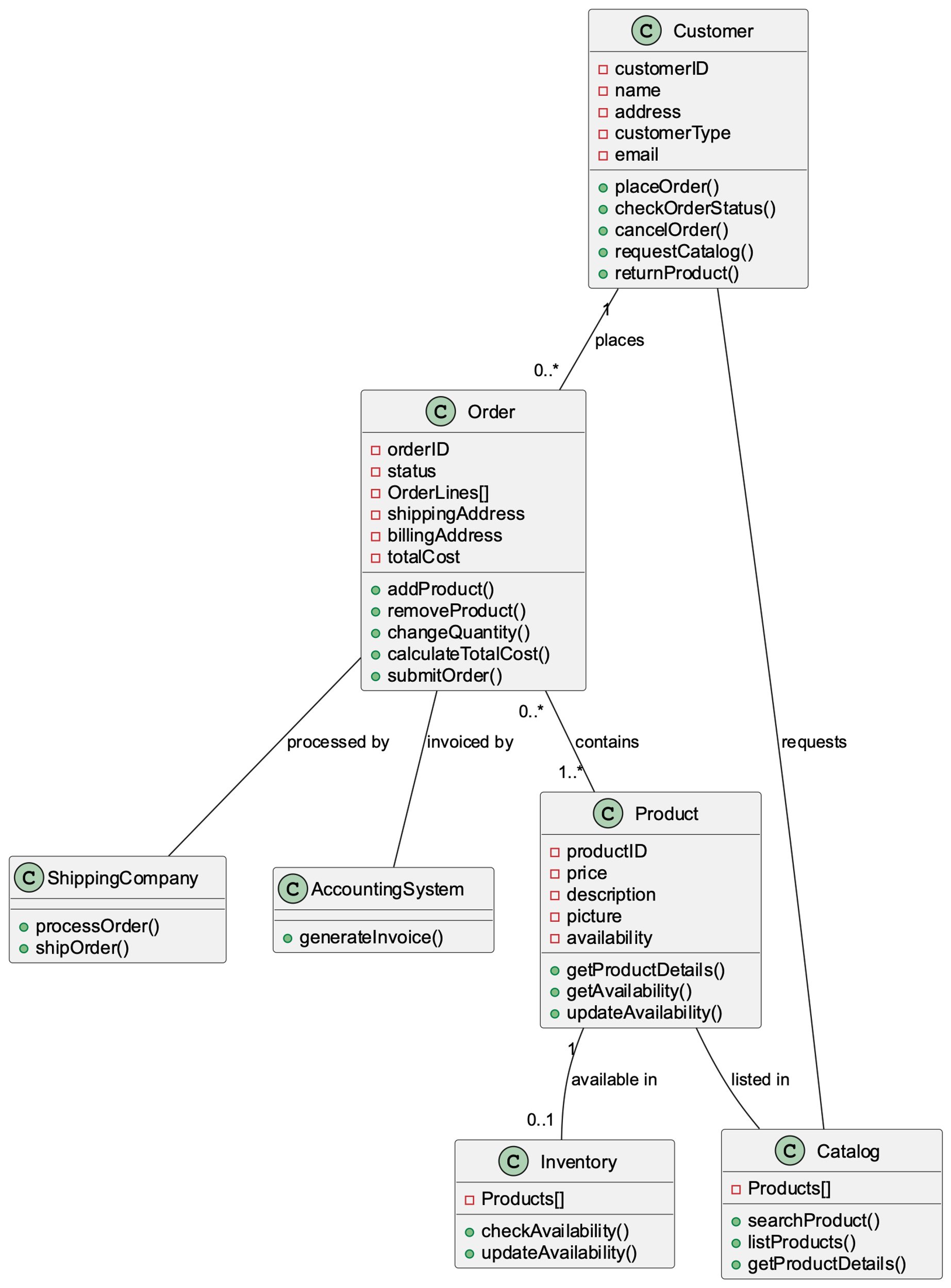
Product "1" -- "0..1" Inventory : available in

Order -- ShippingCompany : processed by

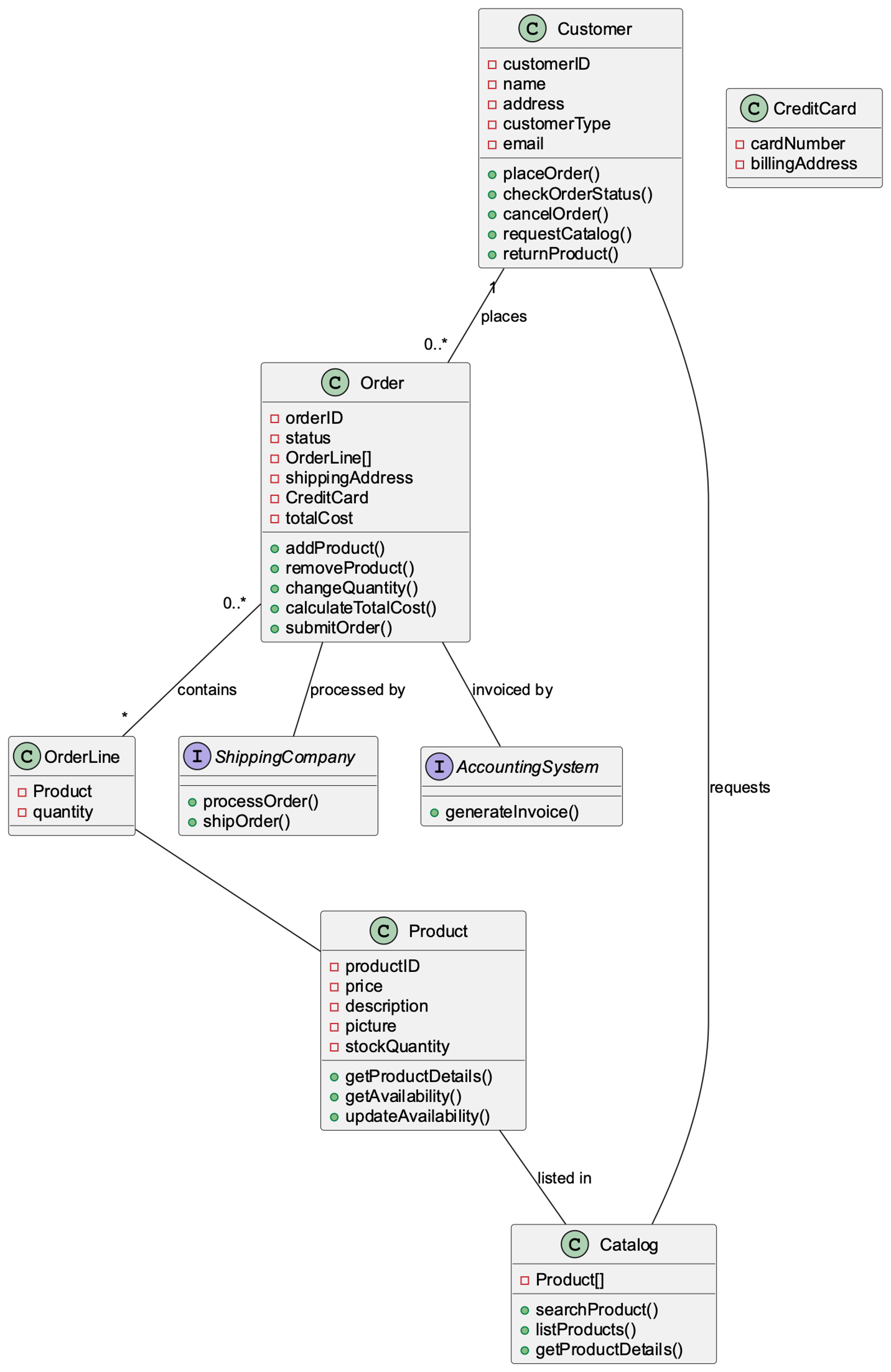
Order -- AccountingSystem : invoiced by

@enduml

UML Class Diagram:



优化后的UML Class Diagram:



优化项目：

1. 合并 Product 和 Inventory 类：将库存信息与每个 Product 一起保存，效率会更高。
2. 将 AccountingSystem 和 ShippingCompany 作为接口：由于它们是我们正在设计的系统的外部实体，因此将它们视为接口是合适的，并且我们的系统可以具有实现这些接口的具体类。
3. OrderLine类：引入该类，分别处理订单中每个产品的数量，简化设计。
4. CreditCard 类：在自己的类中分离出与信用卡相关的详细信息。

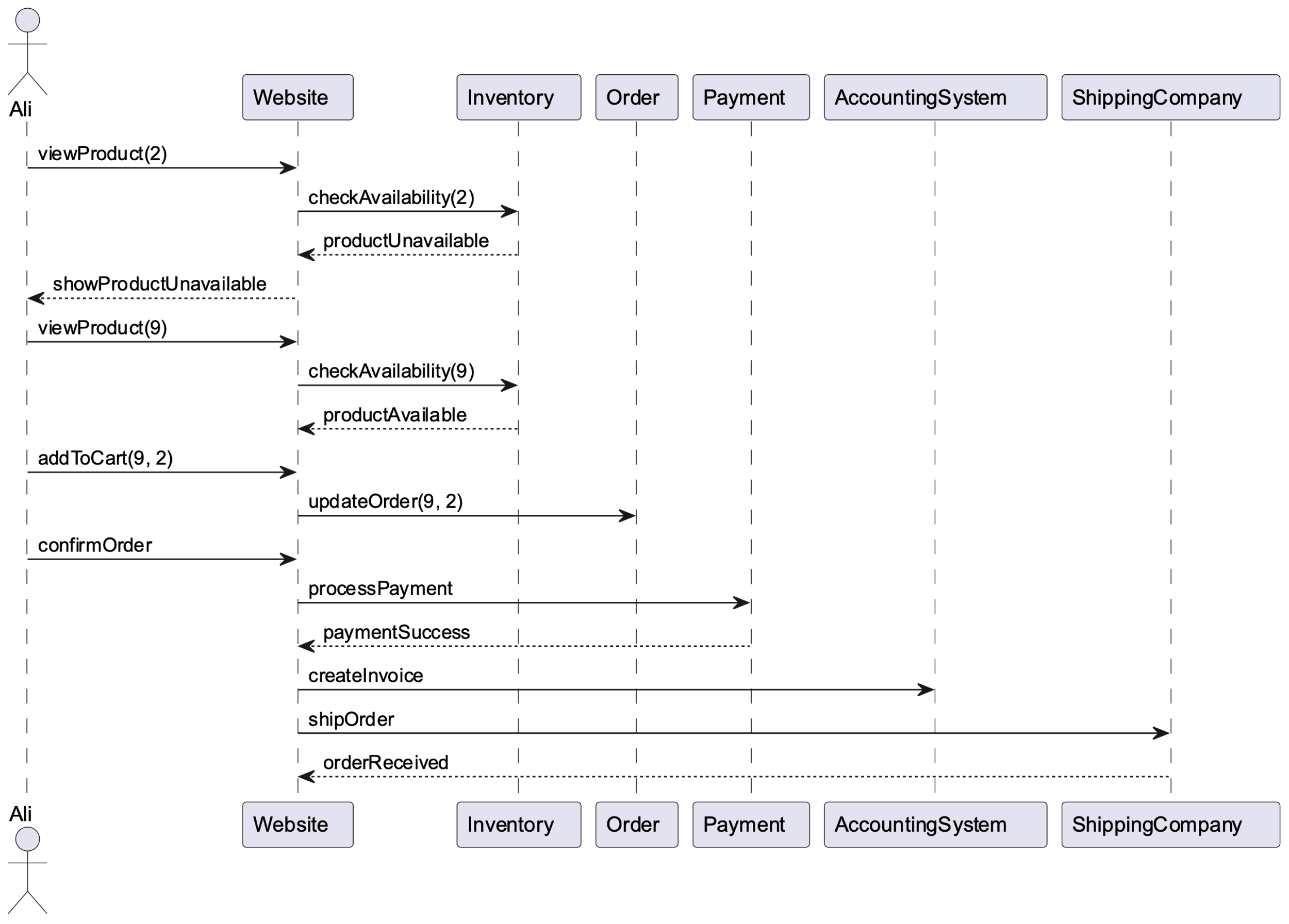
完成第二问的对话记录：

Model: GPT-4

浏览器访问：

<https://chat.openai.com/share/8fe519f9-245e-4433-91bb-13d48d986d14>

1. draw a UML Sequence Diagram for this particular scenario. You may use any software/solution domain objects if needed as well.



Generated PlantUML code:

@startuml

actor Ali

participant Website

participant Inventory

participant Order

participant Payment

participant AccountingSystem

participant ShippingCompany

Ali -> Website: viewProduct(2)

Website -> Inventory: checkAvailability(2)

Inventory --> Website: productUnavailable

Website --> Ali: showProductUnavailable

Ali -> Website: viewProduct(9)

Website -> Inventory: checkAvailability(9)

Inventory --> Website: productAvailable

Ali -> Website: addToCart(9, 2)

Website -> Order: updateOrder(9, 2)

Ali -> Website: confirmOrder

Website -> Payment: processPayment

Payment --> Website: paymentSuccess

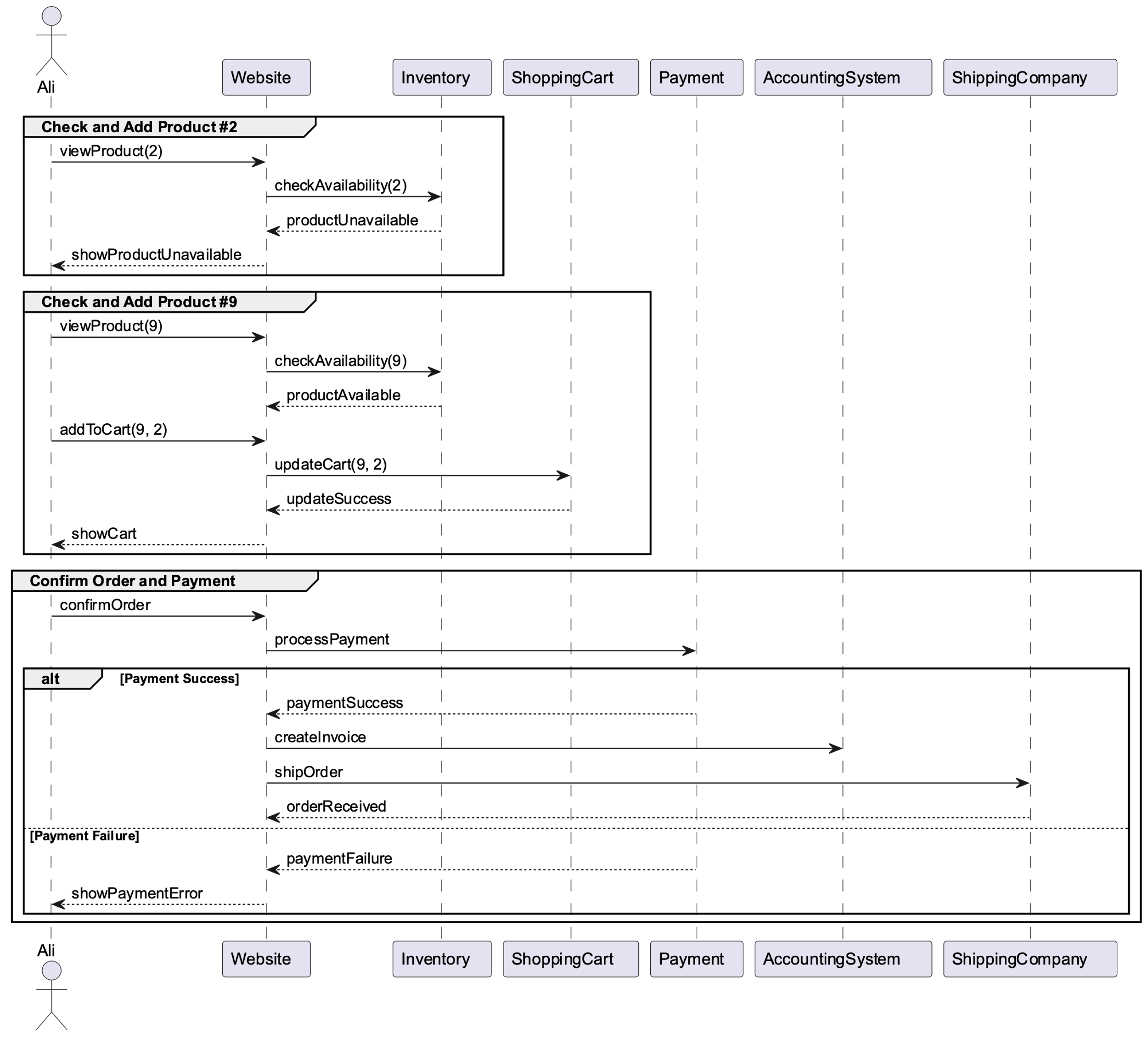
Website -> AccountingSystem: createInvoice

Website -> ShippingCompany: shipOrder

ShippingCompany --> Website: orderReceived

@enduml

优化后：



优化内容：

将流程分为三个主要组：检查和添加产品 #2、检查和添加产品 #9，以及确认订单和处理付款。 这些组有助于组织图中的交互。

在支付处理组中添加了一个“alt”块，以显示序列如何根据支付是否成功而不同。 如果付款成功，订单将被处理并发货； 否则，向 Ali 显示一条错误消息。

添加了一个 ShoppingCart 参与者，以更好地表示将商品添加到购物车并对其进行更新的过程。

完成第三问的对话记录：

Model: GPT-4

浏览器访问：

<https://chat.openai.com/share/0945e3c0-2a0a-418b-a359-0bada2fb8f53>