# 1.用例图

1）询问Ai的过程：（有些长，AI总生成时间为84.24s，部分AI调整code的过程未计时）

AI生成中主要的问题是它的plantUML代码生成的图比较混乱，难以辨别哪个Actor和哪个用例连接起来，经过不断调整，最后终于生成了我可以看懂的用例图（虽然也是混乱的，而且不符合标准）

我：

First, remember 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.

AI:

Problem Description

The mail-order company wants to automate its order processing system accessible via the web and phone. Customers can place orders, check order status, cancel orders, and request a catalog. The system requires customers to identify themselves through a customer number or their name and address. Customers can select products from an online catalog, view product information such as price, description, and picture, and indicate whether they want to buy the product and in what quantity. Customers provide shipping and billing addresses, credit card information, and an overview of the order before submitting it. The accounting system receives credit card number, billing address, and order cost information to create an invoice. Orders are forwarded to the shipping company for fulfillment. Gold customers have additional benefits such as extended product return period and more bonus points per purchase. They can also sign up for email notification when a back-ordered product becomes available.

我：

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

AI：

@startuml

left to right direction

actor Customer

actor Customer Representative

actor Accounting System

actor Shipping Company

rectangle System {

Customer -> (Place Order)

Customer -> (Check Order Status)

Customer -> (Cancel Order)

Customer -> (Request Catalog)

Customer -> (Return Product by Phone)

Customer -> (Sign up for Email Notification)

Customer -> (Become Gold Customer)

Customer Representative -> (Place Order)

Customer Representative -> (Check Order Status)

Customer Representative -> (Cancel Order)

Customer Representative -> (Request Catalog)

Customer Representative -> (Return Product by Phone)

(Create Invoice) -> Accounting System

(Fill and Ship Order) -> Shipping Company

}

@enduml

我：

Please adjust the above code to make the actors in the diagram distributed on the left and right sides of the diagram.

AI：

@startuml

left to right direction

actor Customer

actor Customer Representative

actor Accounting System

actor Shipping Company

rectangle System {

Customer -> (Place Order)

Customer -> (Check Order Status)

Customer -> (Cancel Order)

Customer -> (Request Catalog)

Customer -> (Return Product by Phone)

Customer -> (Sign up for Email Notification)

Customer -> (Become Gold Customer)

Customer Representative -> (Place Order)

Customer Representative -> (Check Order Status)

Customer Representative -> (Cancel Order)

Customer Representative -> (Request Catalog)

Customer Representative -> (Return Product by Phone)

(Create Invoice) -> Accounting System

(Fill and Ship Order) -> Shipping Company

}

Customer |o-right-> (Place Order)

Customer |o-right-> (Check Order Status)

Customer |o-right-> (Cancel Order)

Customer |o-right-> (Request Catalog)

Customer |o-right-> (Return Product by Phone)

Customer |o-right-> (Sign up for Email Notification)

Customer |o-right-> (Become Gold Customer)

Customer Representative |o-left-> (Place Order)

Customer Representative |o-left-> (Check Order Status)

Customer Representative |o-left-> (Cancel Order)

Customer Representative |o-left-> (Request Catalog)

Customer Representative |o-left-> (Return Product by Phone)

@enduml

我：

There is a syntax error in line 26:"Customer |o-right-> (Place Order)"

AI：

Apologies for the mistake in the previous code. Here is the corrected code:

@startuml

left to right direction

actor Customer

actor Customer Representative

actor Accounting System

actor Shipping Company

rectangle System {

Customer -> (Place Order)

Customer -> (Check Order Status)

Customer -> (Cancel Order)

Customer -> (Request Catalog)

Customer -> (Return Product by Phone)

Customer -> (Sign up for Email Notification)

Customer -> (Become Gold Customer)

Customer Representative -> (Place Order)

Customer Representative -> (Check Order Status)

Customer Representative -> (Cancel Order)

Customer Representative -> (Request Catalog)

Customer Representative -> (Return Product by Phone)

(Create Invoice) -> Accounting System

(Fill and Ship Order) -> Shipping Company

}

Customer -right-> (Place Order)

Customer -right-> (Check Order Status)

Customer -right-> (Cancel Order)

Customer -right-> (Request Catalog)

Customer -right-> (Return Product by Phone)

Customer -right-> (Sign up for Email Notification)

Customer -right-> (Become Gold Customer)

Customer Representative -left-> (Place Order)

Customer Representative -left-> (Check Order Status)

Customer Representative -left-> (Cancel Order)

Customer Representative -left-> (Request Catalog)

Customer Representative -left-> (Return Product by Phone)

@enduml

我：

Still, the diagram is very tall in height and narrow in width, would you please adjust the code to make the diagram wide in width and short in height?

AI：

Sure, here is the updated code to adjust the dimensions of the diagram:

@startuml

left to right direction

actor Customer

actor Customer Representative

actor Accounting System

actor Shipping Company

scale 800 width

rectangle System {

Customer -> (Place Order)

Customer -> (Check Order Status)

Customer -> (Cancel Order)

Customer -> (Request Catalog)

Customer -> (Return Product by Phone)

Customer -> (Sign up for Email Notification)

Customer -> (Become Gold Customer)

Customer Representative -> (Place Order)

Customer Representative -> (Check Order Status)

Customer Representative -> (Cancel Order)

Customer Representative -> (Request Catalog)

Customer Representative -> (Return Product by Phone)

(Create Invoice) -> Accounting System

(Fill and Ship Order) -> Shipping Company

}

Customer -right-> (Place Order)

Customer -right-> (Check Order Status)

Customer -right-> (Cancel Order)

Customer -right-> (Request Catalog)

Customer -right-> (Return Product by Phone)

Customer -right-> (Sign up for Email Notification)

Customer -right-> (Become Gold Customer)

Customer Representative -left-> (Place Order)

Customer Representative -left-> (Check Order Status)

Customer Representative -left-> (Cancel Order)

Customer Representative -left-> (Request Catalog)

Customer Representative -left-> (Return Product by Phone)

@enduml

This should produce a wider and shorter diagram.

我：

Are you sure you put the actor on the left and right side?

AI：

Apologies for the confusion. Here is the updated code with the actors placed on the left and right sides of the diagram:

@startuml

left to right direction

actor Customer

actor Customer Representative

actor Accounting System

actor Shipping Company

scale 800 width

Customer -left-> (Place Order)

Customer -left-> (Check Order Status)

Customer -left-> (Cancel Order)

Customer -left-> (Request Catalog)

Customer -left-> (Return Product by Phone)

Customer -left-> (Sign up for Email Notification)

Customer -left-> (Become Gold Customer)

Customer Representative -left-> (Place Order)

Customer Representative -left-> (Check Order Status)

Customer Representative -left-> (Cancel Order)

Customer Representative -left-> (Request Catalog)

Customer Representative -left-> (Return Product by Phone)

rectangle System {

(Create Invoice) -> Accounting System

(Fill and Ship Order) -> Shipping Company

Customer --> (Place Order)

Customer --> (Check Order Status)

Customer --> (Cancel Order)

Customer --> (Request Catalog)

Customer --> (Return Product by Phone)

Customer --> (Sign up for Email Notification)

Customer --> (Become Gold Customer)

Customer Representative --> (Place Order)

Customer Representative --> (Check Order Status)

Customer Representative --> (Cancel Order)

Customer Representative --> (Request Catalog)

Customer Representative --> (Return Product by Phone)

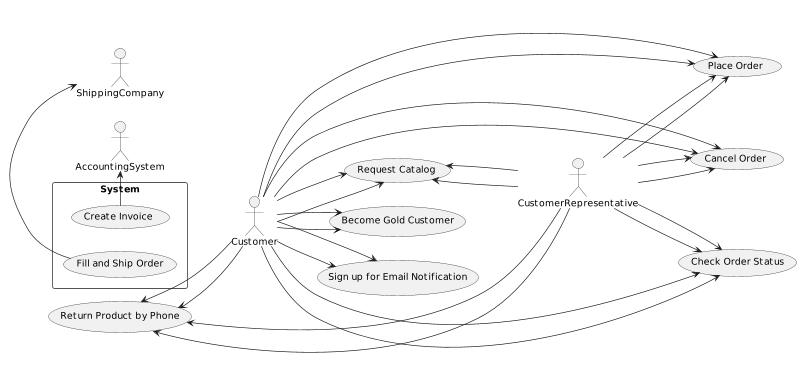
}

@enduml

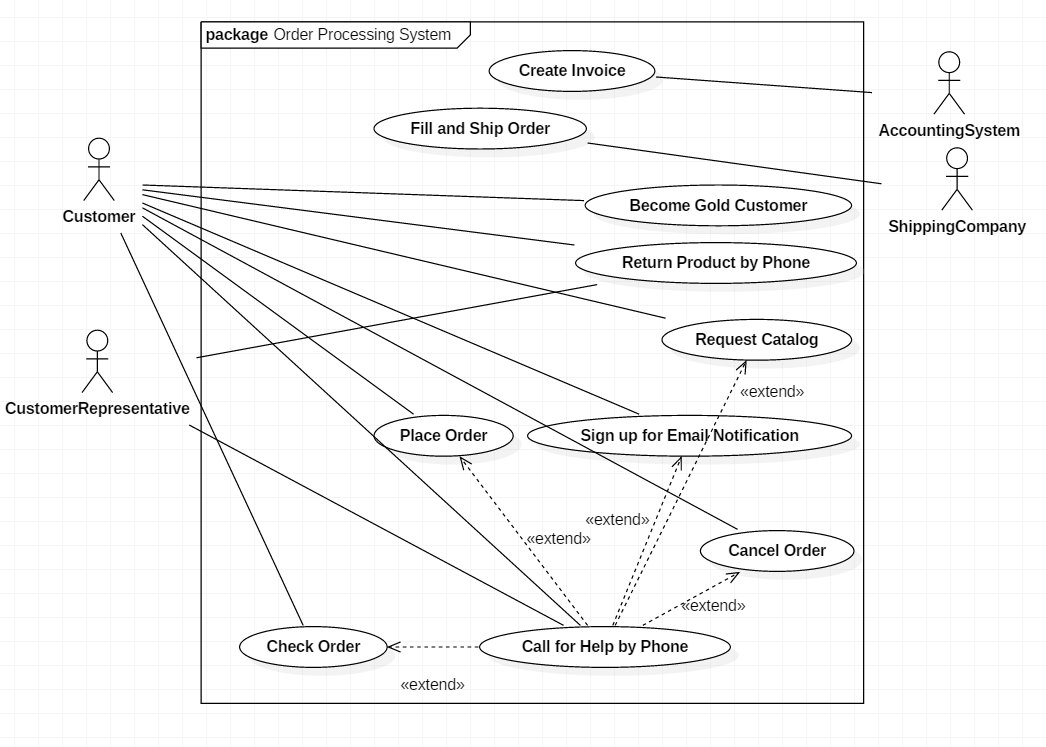
This should display the actors on the left side and the system use cases on the right side of the diagram.

## 2）AI生成的图，与我修改后的图：

AI：



我修改后的图：



我将AI生成图的格式调整后，增加了Call for Help by Phone的用例，然后以此用例扩展出了Request Catalog、Sign up for Email Notification、Place Order、Cancel Order、Check Order等用例。不同于AI生成图，我认为Return Product by Phone用例是必须要Customer和CustomerRepresentative共同完成的，而其他的一些用例是否由两者共同完成是可选的。

# 2.类图

1）询问AI的过程（忘了计时，大概40s?，比用例图顺利多了）

我：

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.(just produce the plantUML code)

AI：

Here is the PlantUML code for the UML Class Diagram based on the application domain analysis:

@startuml

class Customer {

+name: string

+email: string

+phone: string

+address: string

+isGoldCustomer: boolean

+placeOrder(): void

+checkOrderStatus(orderId: int): string

+cancelOrder(orderId: int): void

+requestCatalog(): void

+returnProduct(orderId: int): void

+signUpForEmailNotification(): void

+becomeGoldCustomer(): void

}

class Order {

+orderId: int

+customer: Customer

+orderDate: date

+orderItems: List<OrderItem>

+orderStatus: string

+calculateTotalOrderAmount(): double

+addItem(product: Product, quantity: int): void

+removeItem(orderItemId: int): void

+updateOrderStatus(status: string): void

}

class OrderItem {

+orderItemId: int

+product: Product

+quantity: int

+calculateTotalItemAmount(): double

}

class Product {

+productId: int

+productName: string

+productDescription: string

+productPrice: double

+productQuantity: int

}

class CustomerRepresentative {

+name: string

+email: string

+phone: string

+address: string

+placeOrder(order: Order): void

+checkOrderStatus(orderId: int): string

+cancelOrder(orderId: int): void

+requestCatalog(): void

+returnProduct(orderId: int): void

}

class AccountingSystem {

+generateInvoice(order: Order): void

}

class ShippingCompany {

+shipOrder(order: Order): void

}

Customer -right-\* Order

Order o-left- OrderItem

OrderItem -right-\* Product

CustomerRepresentative -right-\* Order

Order --> AccountingSystem

Order --> ShippingCompany

@enduml

The Customer class has attributes for customer details and methods for placing an order, checking order status, canceling an order, requesting a catalog, returning a product, signing up for email notifications, and becoming a gold customer.

The Order class has attributes for order details, a list of OrderItems, and methods for calculating total order amount, adding/removing OrderItems, and updating the order status.

The OrderItem class has attributes for Product details and quantity, and a method for calculating the total item amount.

The Product class has attributes for product details and pricing information.

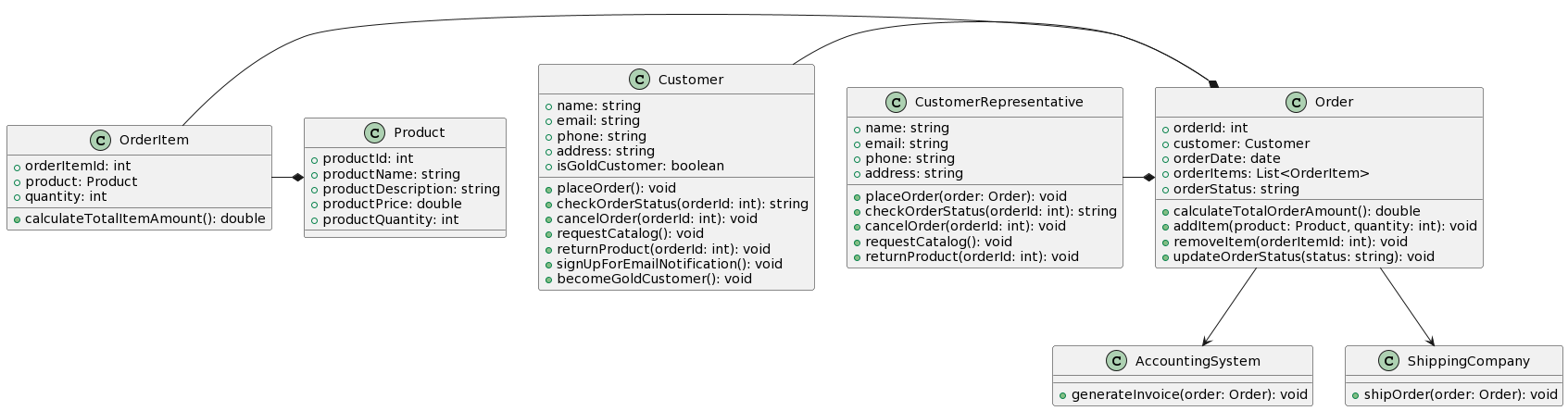
The CustomerRepresentative class has attributes for customer representative details and methods for placing an order, checking order status, canceling an order, and requesting a catalog.

The AccountingSystem class has a method for generating an invoice for an order.

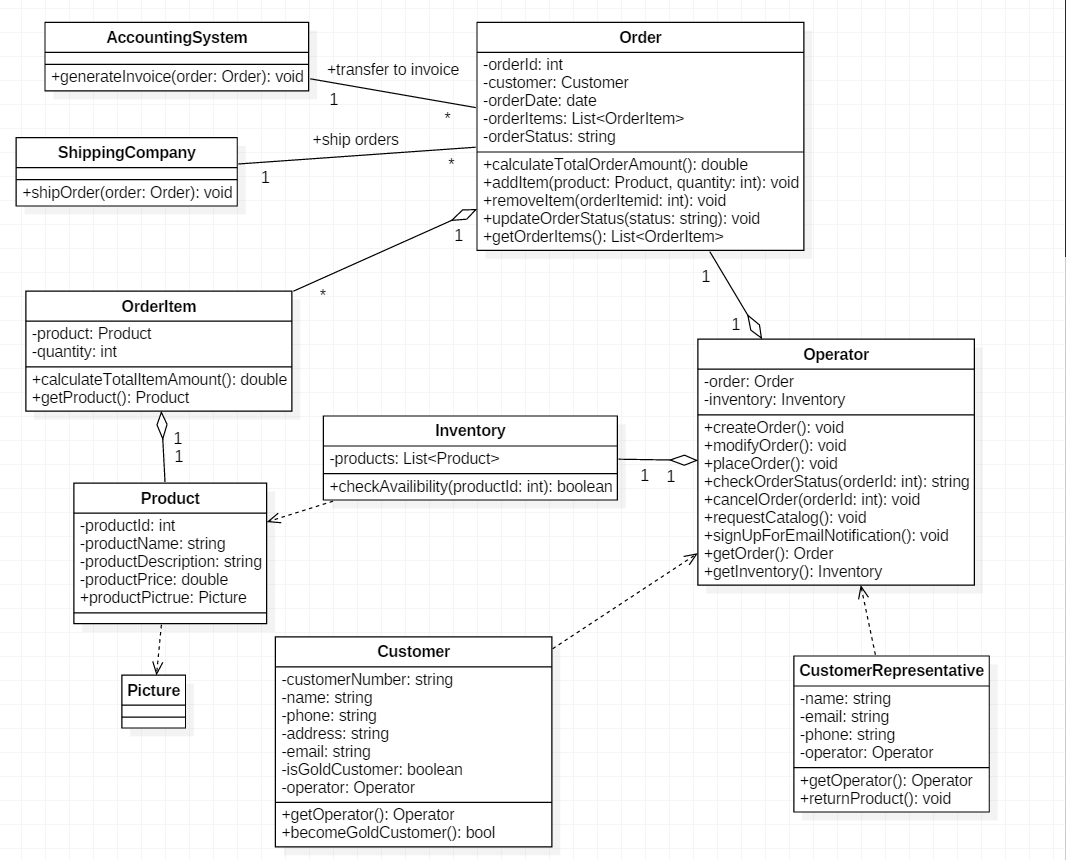
The ShippingCompany class has a method for shipping an order.

## 2）AI生成的图，与我修改后的图：

AI：



我修改后的图：



我增加了Inventory类，认为AI漏识别了该类，另外对于Product应该有图片介绍，为此引进了Picture类。将Product的Quantity成员移入到了OrderItem类中，因为Product应该只含有Product产品本身的信息，而且并没有增加不必要的耦合。对于OrderItem，我认为不需要Id来特别标识，因为Order本身会有Id标识，只需要标识产品本身和订单就可以了，不需要标识中间量。对于Customer和CustomerRepresentative公用的方法，我抽离出来建立了Operator类，以减少代码重复。另外，我也适当增加了一些get方法。

# 3.顺序图

1）询问AI的过程（总体用时大概53.91s，AI一开始并不能区分系统顺序图和顺序图的区别，直到我给了它简单的讲解）

我：

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.

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

AI：（17.70s）

Here is the PlantUML code for the UML Sequence Diagram for the "place order" scenario:

@startuml

actor Customer

Customer -> System: Log in to Website

Customer -> System: Add Product #2 to Basket

System -> Customer: Product #2 Unavailable

Customer -> System: Add 2 Quantity of Product #9 to Basket

System -> Customer: Confirm Shipping and Billing Addresses

System -> Customer: Confirm Credit Card Information

Customer -> System: Click Submit

System -> Customer: Order Confirmation

@enduml

In this scenario, the Customer logs in to the website of the System and tries to add Product #2 to the basket. The System returns a message that the product is unavailable. The Customer then adds 2 quantities of Product #9 to the basket, and the System prompts them to confirm their shipping and billing addresses and credit card information. The Customer confirms the information and clicks the Submit button. The System confirms the order and sends an order confirmation to the Customer.

我：

I don't want the System Sequence Diagram, I want the Sequence Diagram, which means more objects come to interact with each other.

AI：(36.21s)

I apologize for the confusion. Here is the updated PlantUML code for the UML Sequence Diagram for the "place order" scenario, with additional objects included:

@startuml

actor Customer

participant Website

participant Catalog

participant ProductInventory

participant Order

participant OrderItem

participant CustomerDatabase

participant CreditCardProcessor

participant ShippingCompany

Customer -> Website: Log in

Customer -> Website: Browse Catalog

Customer -> Website: Add Product #2 to Basket

Website -> Catalog: Check Product Availability

Catalog --> Website: Product #2 Unavailable

Website -> Catalog: Add Product #9 Quantity 2 to Basket

Catalog -> ProductInventory: Check Product Availability

ProductInventory --> Catalog: Product #9 Available

Website -> Customer: Confirm Shipping and Billing Addresses

Website -> CustomerDatabase: Retrieve Customer Information

CustomerDatabase --> Website: Customer Information

Website -> Customer: Confirm Credit Card Information

Website -> CreditCardProcessor: Process Payment

CreditCardProcessor --> Website: Payment Confirmation

Website -> ShippingCompany: Ship Order

ShippingCompany --> Website: Shipping Confirmation

Website -> Order: Create Order

Order -> OrderItem: Add Item #9 Quantity 2

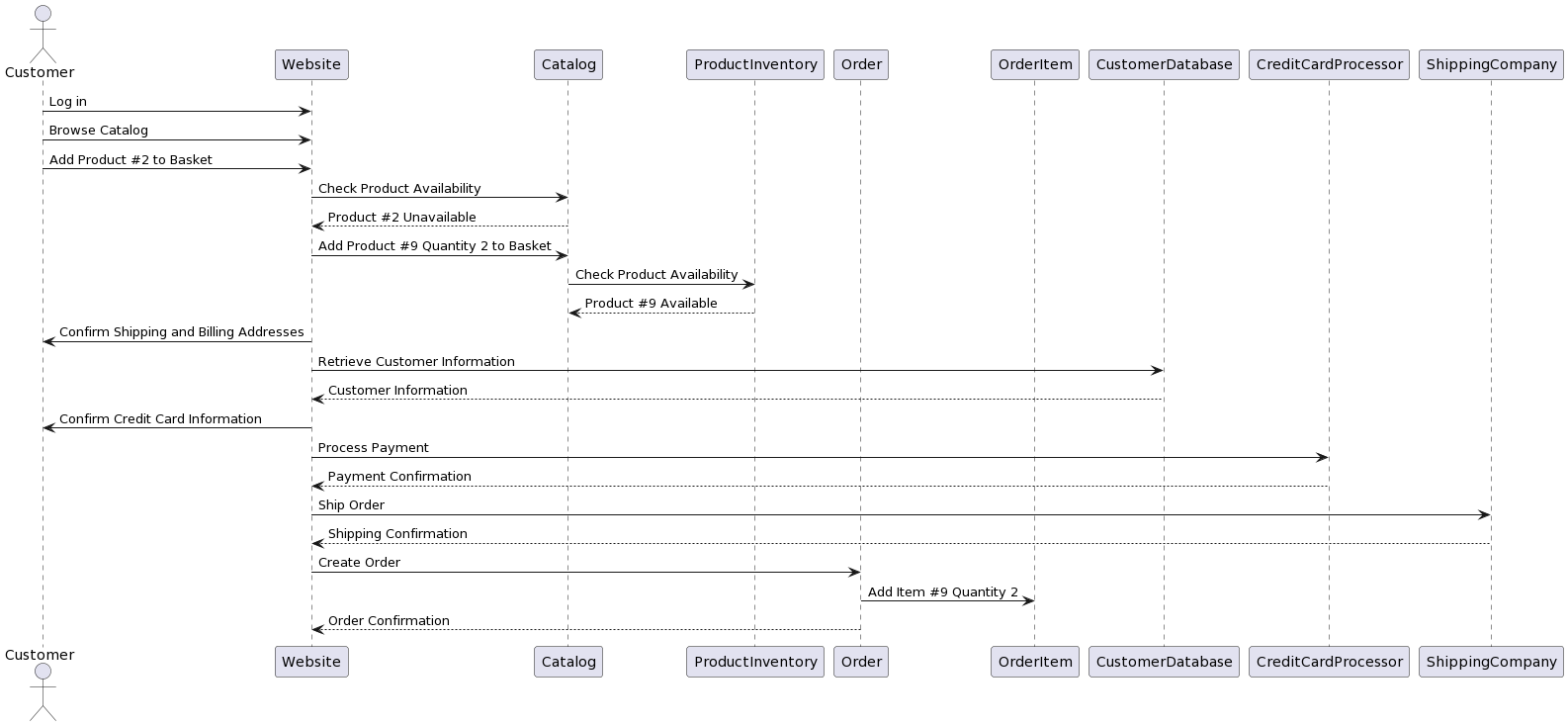
Order --> Website: Order Confirmation

@enduml

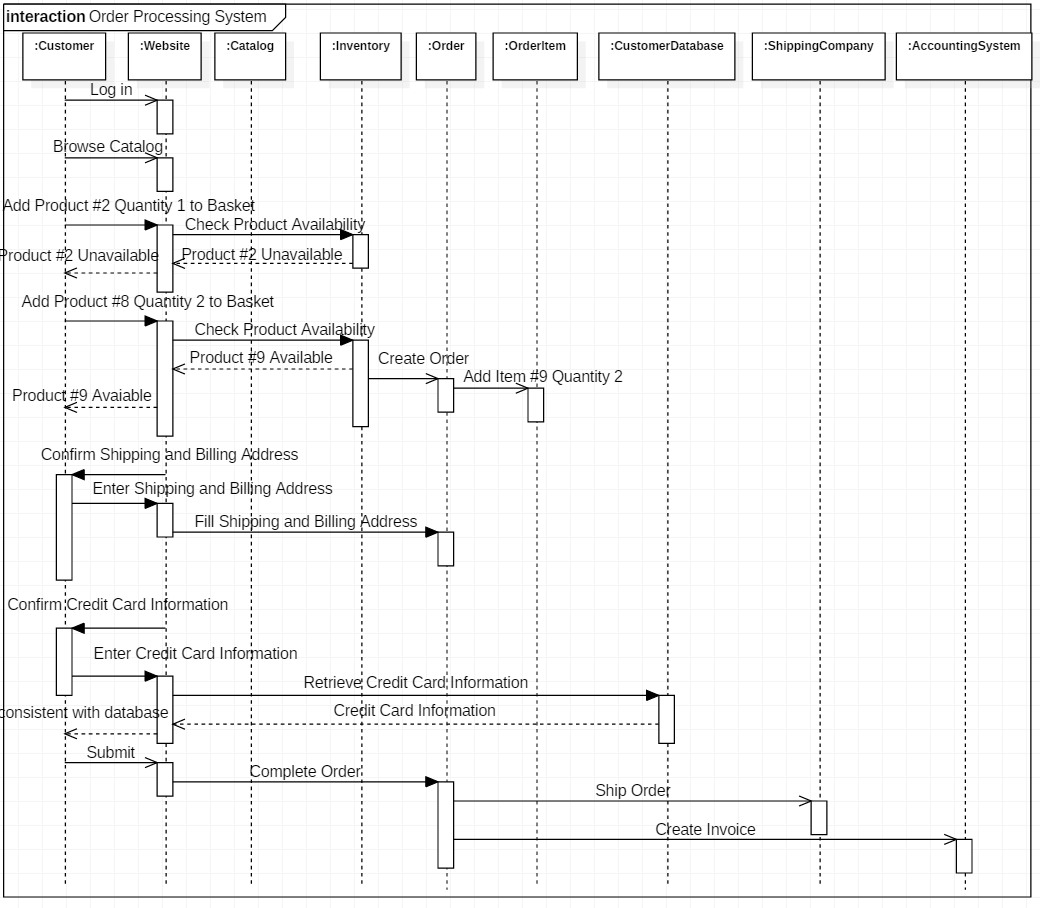
In this scenario, the Customer logs in to the Website and browses the Catalog. The Customer tries to add Product #2 to the basket, which prompts the Website to check its availability with the Catalog. The Catalog responds that the product is unavailable. The Customer then adds 2 quantities of Product #9 to the basket, which prompts the Website to check its availability with the ProductInventory. The ProductInventory confirms the availability of Product #9, and the Website prompts the Customer to confirm their shipping and billing addresses. The Website retrieves the Customer's information from the CustomerDatabase and prompts the Customer to confirm their credit card information. The Website processes the payment with the CreditCardProcessor and receives a payment confirmation. The Website then ships the order with the ShippingCompany. The Website creates an Order and adds an OrderItem for Product #9 quantity 2. Finally, the Website sends an order confirmation to the Customer.

## 2）AI生成的图，与我修改后的图：

AI：



我修改后的图：



AI生成的顺序图质量有些低。

将Product加入Basket，检查Product是否有存货，再到返回结果的过程应该是类似的，但在AI顺序图中Product#2和Product#9的加入和检验过程却不同，另外我又加入了Website到Customer的返回结果，以告诉用户将Product加入Basket是否成功。

创建订单（Create Order）和对OrderItem的修改(Add Item #9 Quantity 2)应该在检查完库存有改商品时就进行，而不是等到最后才进行。

验证邮递和付款地址（Confirm Shipping and Billing Address）应该不需要进入到CustomerDataBase中取数据进行验证，邮递和付款地址应该可以灵活改变。并且，在验证邮递和付款地址后，可以将相关信息填入到Order中，AI的顺序图中漏了这点。

增加了提示用户信用卡信息正确的消息（consistent with database）。

AI顺序图结尾部分不完整，我增加了用户点击“提交”按钮（Submit），浏览器告知Order要进行收尾工作（Complete Order），最后通知ShippingCompany和AccountingSystem分别开始进行各自的任务（Ship Order、Create Invoice）。