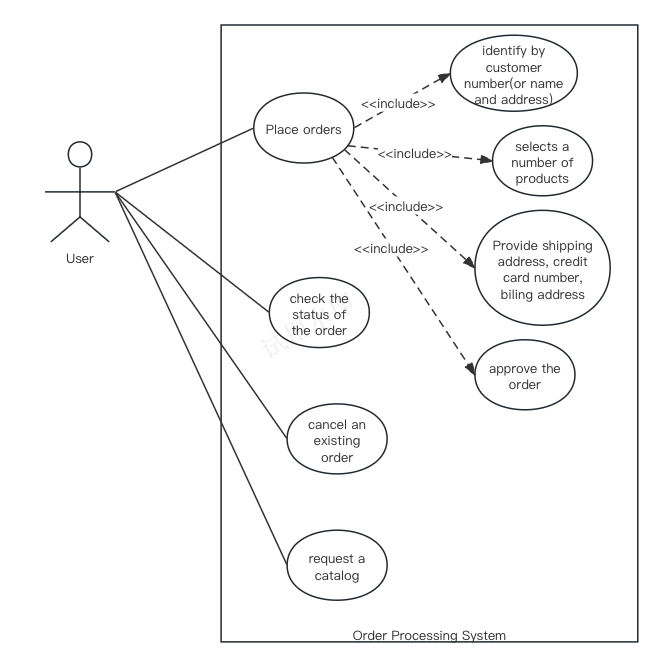
第二周实验

用例图：

自己完成部分：



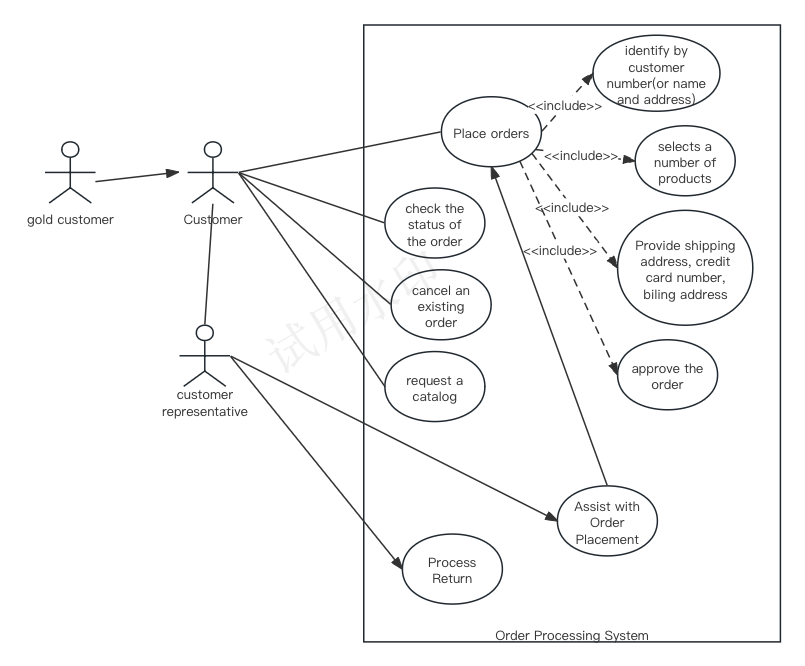
修改后内容：

金卡用户继承自客户，添加客户代表以及客户代表所对应的用例

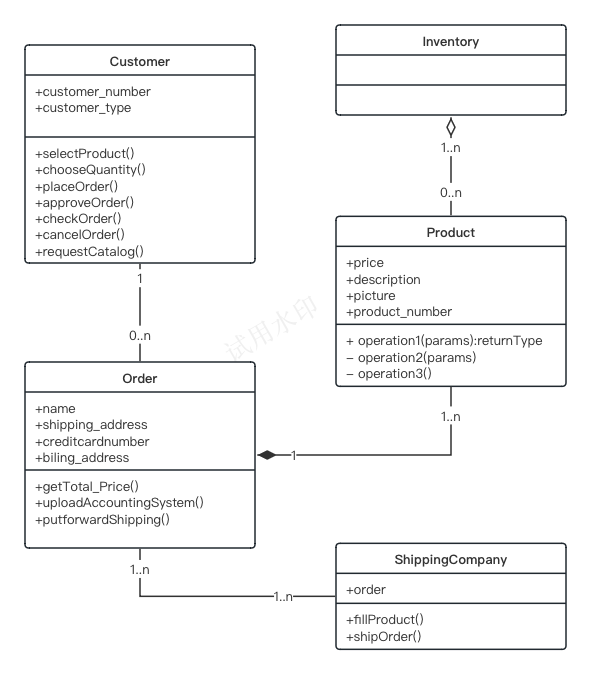
Chatgpt使用次数2

version1完成时间：39min

修改时间:12min



类图：

独立完成部分：  
修改内容：

添加了Accounting System类  
修改了部分类的部分属性的可见性

修改部分类的方法的参数以及返回值

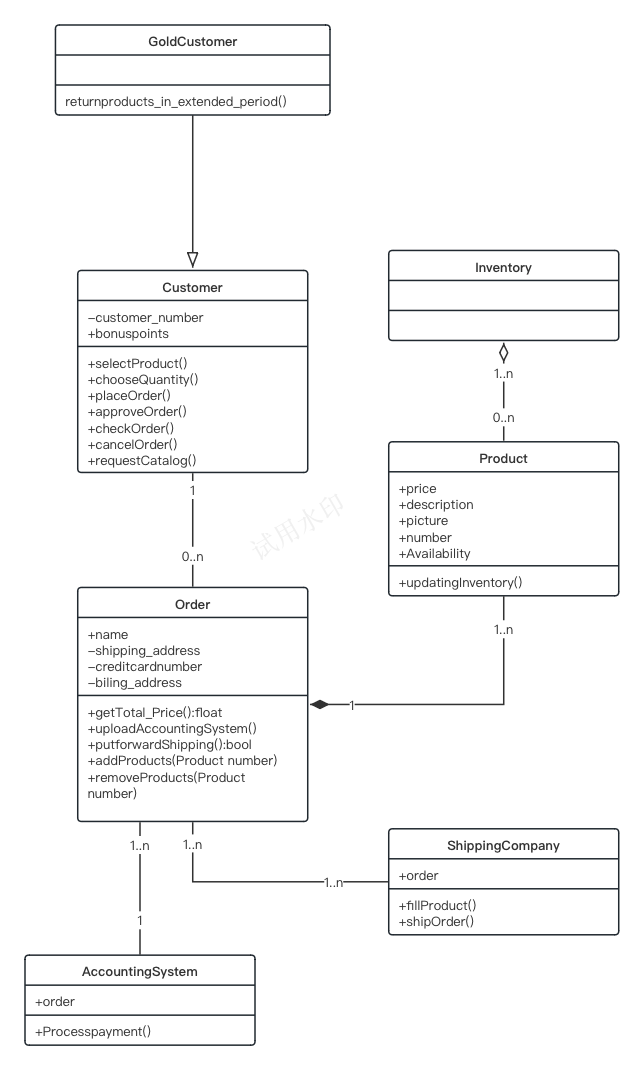
添加金卡用户继承自客户

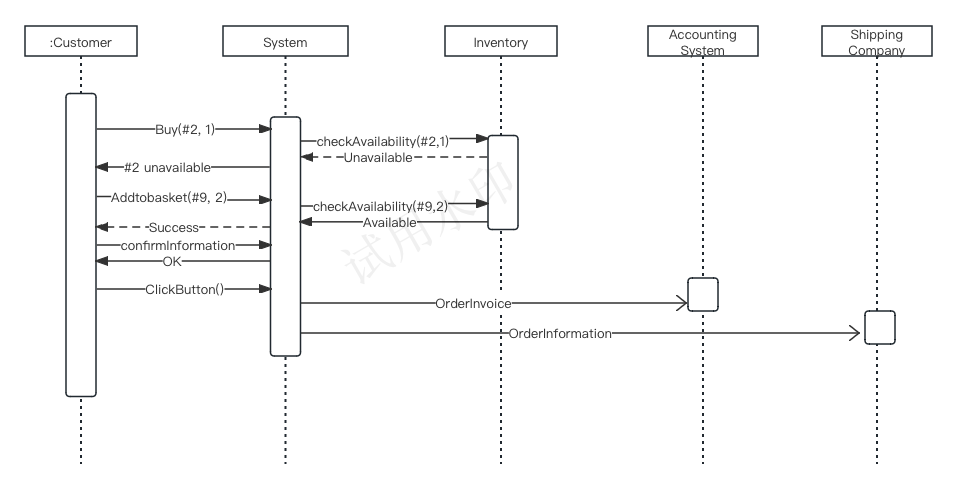
去掉了原来客户的customer\_type属性

chatgpt询问次数：2

v1独立完成时间：32min

修改时间：10min



顺序图：

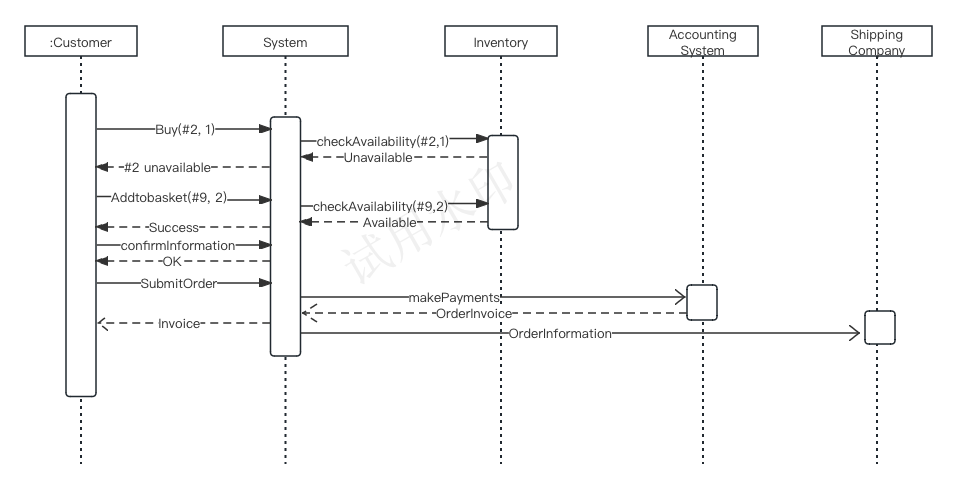
修改：

修改了顺序图的返回值为虚线

添加了AccountingSystem的返回值

完成时间：32min

chatgpt问询条数：1



以下附聊天记录：

You

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

这里是我的用例图、类图和顺序图，请你给予修改的提示

ChatGPT

Great, you’ve provided UML diagrams for analysis and design, which is an excellent way to represent system components and their interactions visually. Let's review each one and provide feedback.

Use Case Diagram (用例图)

Ensure all relevant actors are present. Based on the problem description, we should have at least the following actors: Customer, Registered Customer (Gold Customer), Customer Representative, and Accounting System.

The use cases should cover all the functionalities mentioned, like "Place Order", "Check Order Status", "Cancel Order", "Request Catalog", and "Return Product" (which should be accessible only to the Customer Representative, not directly by the Customer).

Class Diagram (类图)

Check if all necessary classes are present. You should have classes for Customer, Order, Product, Inventory, ShippingCompany, and others as needed.

Each class should have the appropriate attributes; for example, Customer might have attributes like customer\_number, customer\_type, and a status to check if they are gold customers.

Operations/methods should be defined for each class; for example, the Order class might have methods like addProduct(), removeProduct(), and calculateTotal().

Relationships between classes should be accurately represented with correct multiplicity. For example, a Customer can have multiple Orders, and each Order can contain multiple Products.

Check for any inheritance relationships, such as GoldCustomer inheriting from Customer.

Sequence Diagram (顺序图)

Ensure the sequence diagram reflects the sequence of interactions for the use case "Place Order" accurately.

The diagram should include the following objects: Customer, System, Inventory, Accounting System, and Shipping Company.

Messages between objects should represent the sequence of steps mentioned in the scenario, like checking the availability of a product, adding a product to the basket, and confirming the order.

Since I don’t have the visual content of the files, I would suggest that you compare your diagrams with the points above. If there are mismatches or missing elements, you might want to update the diagrams accordingly. If you would like me to review the diagrams directly, you would need to display the images within this chat so I can provide specific feedback.

You

用例图中customer和customer representative的关系是什么

2 / 2

ChatGPT

ChatGPT

在用例图中，Customer 和 Customer Representative 都是参与者（actors），代表与系统交互的外部实体。他们的关系与系统如何被使用有关，但它们在用例图中不直接相互关联。这是因为用例图主要描述参与者与用例之间的关系，而不是参与者之间的关系。

Customer 参与者通常直接与系统中的用例进行交互，如下：

Place Order

Check Order Status

Cancel Order

Request a Catalog

Customer Representative 是作为一个桥梁或代理来代表顾客与系统进行交互的参与者。在电话订单或特殊请求（例如退货处理，这在系统的网页版中不可用）的情况下，代表会代替顾客执行操作。Customer Representative 参与的用例可能包括：

Process Return (因为退货只能通过电话完成)

Assist with Order Placement (当客户通过电话下单时)

如果你的系统设计允许 Customer Representative 执行所有 Customer 可以执行的用例（比如为不熟悉网络操作的客户下单或者帮助客户查询订单状态等），那么 Customer Representative 参与者将会连接到所有 Customer 参与者可以访问的用例。这不是它们之间的直接关系，而是表示 Customer Representative 也可以代表客户执行这些操作。在用例图中，你可能会看到 Customer Representative 与多个用例有关联，以展示其代理角色。

You

这是我的类图，哪些地方可以改进？

ChatGPT

Based on the class diagram image you’ve provided, let’s consider some areas for potential improvement:

Attributes and Methods: Ensure that each class has all necessary attributes and methods to fulfill its responsibilities. For example, the Order class might have methods to add or remove products from the order, which are not currently visible. Also, consider if Product needs more operations to handle business logic, like updating inventory levels.

Associations and Multiplicity: Check the associations between classes. For example, an Order should likely be associated with multiple Product instances (as it seems to be indicated with the 1..n multiplicity), but also each Product could be associated with multiple Orders (because many customers can order the same product). So, the multiplicity should be adjusted to reflect that.

Visibility: In UML, + denotes public visibility, which is typical for methods that need to be accessible from outside the class. However, ensure that attributes (like creditcardnumber in Order) have the correct visibility. Sensitive information often has private visibility (denoted by -), and accessed through public methods to ensure data encapsulation and security.

Data Types and Parameters: For operations, it is good practice to specify the data type for parameters and return types. For example, the getTotal\_Price() method in Order should indicate what type of value it returns (float, double, etc.).

Aggregation vs. Composition: You have an aggregation relationship from Inventory to Product, which is typically used to denote a whole-part relationship where the part can exist independently of the whole. If a Product can only exist as part of an Inventory, then this should be a composition relationship instead.

Inheritance: If there are specialized types of any class (such as GoldCustomer extending Customer), they should be represented with inheritance (a solid line with a closed, unfilled arrowhead).

Interfaces or Abstract Classes: If there are operations that can be generalized, consider using interfaces or abstract classes. For example, if multiple entities have a shipOrder() method, there could be a Shippable interface that they all implement.

Order and ShippingCompany Relationship: The Order class has a method putforwardShipping(), which suggests a relationship with ShippingCompany. Make sure there's a clear association or dependency drawn between these two classes to indicate their interaction.

Missing Classes or Modules: Based on the problem description, you might need additional classes, like AccountingSystem to process payments and CustomerAccount to keep track of customer information and status (regular or gold).

Remember that class diagrams should evolve as the understanding of the system grows, so it is perfectly fine to make multiple iterations of these diagrams as the system's requirements become clearer.

You

order和shippingcompany的关系是什么

ChatGPT

在你的类图中，Order 和 ShippingCompany 类之间的关系代表了它们如何交互来完成订单的发货过程。这种关系通常在 UML 中用以下方式之一来表示：

关联 (Association): Order 和 ShippingCompany 可能会有一个直接的关联，因为订单（Order）需要被发货公司（ShippingCompany）发出。这种关系通常是双向的，并且可以通过在两个类之间画一条线来表示。这条线可能会有箭头来指明关系的方向（通常从订单指向发货公司），以及在端点附近的数字来表示多重性（multiplicity），说明一个订单可以被多个发货公司处理（尽管在实际应用中这种情况较少），或者一个发货公司可以处理多个订单。

依赖 (Dependency): 如果Order类的某个方法如putforwardShipping()仅仅是调用ShippingCompany的服务，而不保持ShippingCompany的一个长期引用，那么这可以被建模为一个依赖关系。在UML中，依赖关系通过带有箭头的虚线表示，箭头从Order指向ShippingCompany。

聚合 (Aggregation) 或组合 (Composition): 如果你的业务逻辑中，发货公司是独立于订单存在的（比如，即使没有订单，发货公司也会存在），那么可以用聚合来表示这种关系，聚合是通过带空心菱形的线来表示的。如果订单创建了发货公司的一个实例，那么这就是一个组合关系，通过带实心菱形的线来表示。

具体到你的系统中，Order 类可能会有一个指向 ShippingCompany 的关联，表示每个订单都需要通过一个发货公司来发货。这个关系通常会有一个方法，如 shipOrder()，在 Order 类中触发，在 ShippingCompany 类中执行具体的发货操作。这个方法可能会在 Order 类中调用，但是它的执行是在 ShippingCompany 类中完成的。所以，在UML类图中，这个关系最可能是一个关联或者依赖关系。

You

请根据最上面的描述来为我的顺序图提出建议

ChatGPT

Reviewing the sequence diagram you provided, here are some suggestions based on the problem description you’ve given:

Actor Interaction:

Make sure that the :Customer lifeline initiates the interaction, which seems to be the case.

There should be an explicit return message from the System to the :Customer after each significant interaction, like after adding to basket and confirming information.

Handling Unavailability:

After #2 unavailable, the system should indicate to the :Customer that the product is not available before proceeding to Addtobasket(#9, 2).

System Components:

The System lifeline likely represents your web system where the customer performs actions. Ensure that it handles the logic between :Customer actions and backend processes.

Inventory Check:

The message checkAvailability(#2,1) is correctly shown. Make sure that the inventory system somehow confirms the unavailability back to the system and that the diagram reflects this message.

Confirmation of Actions:

When Addtobasket is successful, there should be a confirmation from the System to the :Customer, which is correctly indicated as Success.

Completing the Order:

The confirmInformation step should correctly reflect the retrieval of customer information from the database.

The ClickButton() action by the :Customer could be more descriptively named, like SubmitOrder() to clearly reflect the action being taken.

Order Processing:

There should be a message from the System to the Accounting System to process the payment. The return message might be a confirmation of payment authorization.

The OrderInvoice message to the Accounting System is correctly shown. There should also be an acknowledgment returned to the System.

Shipping:

The sequence of messages to the Shipping Company looks correct, but make sure to include a return message to confirm the order has been shipped or processed.

Return Messages:

All operations should have a return message to indicate completion or success status, especially if the operation impacts the flow (e.g., an unavailable product or successful payment).

Error Handling:

Consider what happens if there's a failure at any point, such as a payment authorization failure. These paths should also be represented in the sequence diagram.

Gold Customer Considerations:

Since the description mentions gold customers, consider if there should be a different path or additional steps for gold customers in your diagram.

Product Returns:

Although not shown in the "place order" sequence, if you expand the diagram to include more use cases, remember that product returns are handled via phone and should involve a Customer Representative.

Remember, sequence diagrams are a tool to communicate the sequence of events and interactions for a particular scenario. It should be detailed enough that someone could understand the flow of events and use it as a guide for implementation.