Class Diagram Case study

"Our system is designed to inventory and ship uniquely identified products. These products may be purchased directly from vendors and resold as is, or we can package vendor products together to make our own custom product. Customers place orders for one or more items, but we acknowledge interested customers in the system whether they have purchased yet or not. Each item corresponds to a product. We identify each product using a unique serial number. The Customer may inquire on the status of his Orders using the order number."

"Shipments of products from vendors are received and placed into inventory. Each product is assigned to a location so that we can easily find it later when filling orders. Each location has a unique location identifier. Customer orders are shipped as the products become available, so there may be more than one shipment to satisfy a single customer order. But a single shipment may contain products from multiple orders. Any items that have not been shipped are placed on a backorder with a reference to the original order."

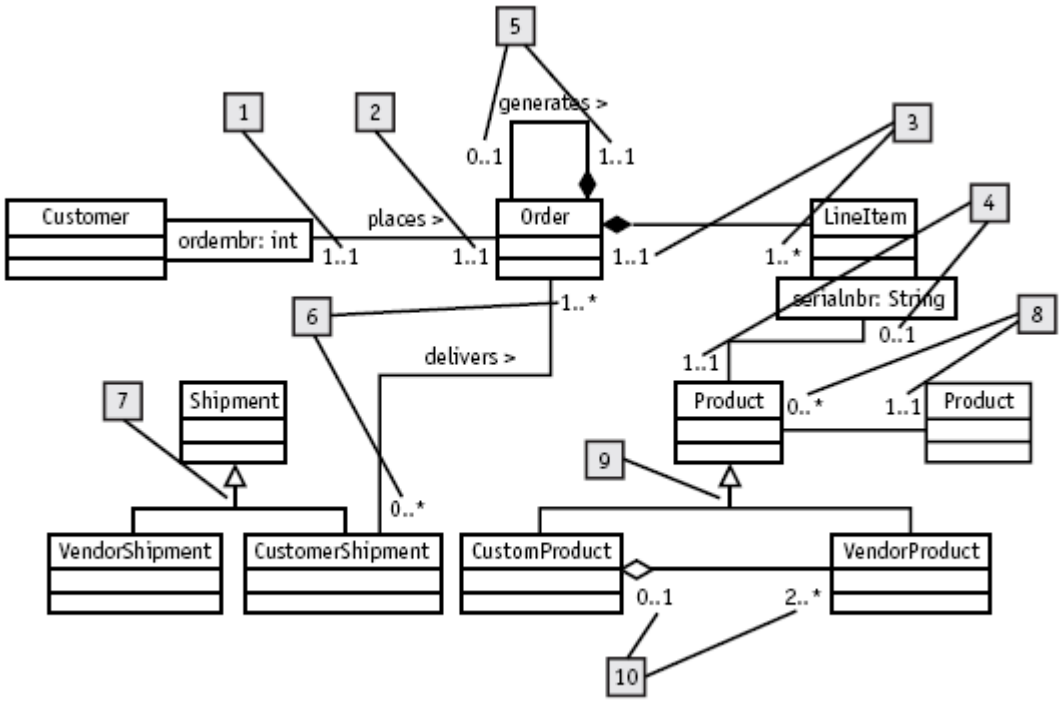

Figure 12-1 *Case study Class diagram*

1. *"Customers place orders for one or more items, but we acknowledge interested customers in the system whether they have purchased yet or not."*
On the "places" association between Customer and Order, the multiplicity of 1..1 means that every Order must be placed by a Customer. An Order cannot exist on its own.

2. *"Customers place orders for one or more items, but we acknowledge interested customers in the system whether they have purchased yet or not. The Customer may inquire on the status of his Orders using the order number."*
On the "places" association between Customer and Order, some customers may not yet have placed any orders while others may have been doing business with us for a long time. The Order multiplicity should be 0..*. But a Customer can use the order number as a qualifier to look up a specific Order (qualified association), so the multiplicity with the qualifier is 1..1.

3. "*Customers place orders for one or more items. . . .*"
An Order is constructed using one or more Line Items. Each Line Item includes information like a price and any applicable discount. But every Line Item exists only as part of an Order represented by composition and a multiplicity of 1..1 on the Order. There must be at least one item on the Order so the LineItem multiplicity is 1..*.

4. "*Each item corresponds to a product. We identify each product using a unique serial number*."
Each Line Item is associated with a specific Product (1..1). The Line Item refers to the Product using a serial number as a qualifier (qualified association).
A Product might not ever be ordered, so the multiplicity on the Line Item end is zero to one (0..1). In other words, a Product might not yet be associated with a Line Item.

5. "*Any items that have not been shipped are placed on a backorder with a reference to the original order*."
An Order that is not filled completely will generate another Order that it refers to as a backorder (role name) and that backorder is associated with the Order that generated it (reflexive composition). Each backorder refers to exactly one other Order, its source (1..1). But each Order may or may not generate backorders (0..*).

6. "*Customer orders are shipped as the products become available, so there may be more than one shipment to satisfy a single customer order. But a single shipment may contain products from multiple orders.*"
The Order is shipped to the Customer via a Customer Shipment. When the
Order has not yet been shipped, the multiplicity on the Customer Shipment is zero (that is, there is no Shipment associated with the Order). When more than one Shipment is needed to fill the Order (for example, the items are being shipped from multiple locations or are restricted by shipping requirements), the multiplicity is "many." Hence the complete multiplicity range is 0..*. A shipment may contain products from many orders, resulting in an Order multiplicity of 1..*.

7. "*Shipments of products from vendors are received and placed into stock. . . . Customer orders are shipped as the products become available*."

Customer Shipment is just one *type* of Shipment (generalization). Another *type* of Shipment is the incoming Vendor Shipment referred to in the receiving process. CustomerShipment and VendorShipment are specializations of Shipment and so inherit all the properties of Shipment.

8. "*Each product is assigned to a location so that we can easily find it later when filling orders. Each location has a unique location identifier*."
Many Products or no Products may be in a given Location (0..*). But in order for you to record a Product into inventory, you have to assign it to a Location.
So there will never be a Product that is not associated with a Location. This requires a multiplicity of 1..1 on the Location end of the association.

9. "*These products may be purchased directly from vendors and resold as is, or we can package vendor products together to make our own custom product*."
VendorProduct and CustomProduct are both *types* of Product (generalization), specializations of the class Product. Both can be ordered and shipped. But CustomProducts are configurations of VendorProducts and VendorProducts are standalone items that are ordered and shipped independently, not in a configuration of other Products.

10. ". . . *we can package vendor products together to make our own custom product*."
We can create custom products using VendorProducts; for example, a home entertainment system might consist of a receiver, CD player, speakers, TV, and so on (aggregation). Why is it aggregation and not composition? Because the VendorProducts, like the CD player, may exist and be sold separately from the entertainment system. The multiplicity on VendorProduct is 2..* because a CustomProduct is only a logic entity made up of a combination of at least two VendorProducts. A VendorProduct may be sold individually and does not have to be part of any CustomProduct configuration (0..1).
Remember to pay close attention to the vocabulary of the problem description. The people who work every day with this information have already created and refined their own verbal abstractions to describe their environment. When you write software, you're only copying their abstractions into another form (software) by applying the rigor and precision of a disciplined analytical approach.

Draw Class Diagram:

Each **customer** has unique id and is linked to exactly one **account**. Account owns shopping cart and orders. Customer could register as a **web user** to be able to buy items online. Customer is not required to be a web user because purchases could also be made by phone or by ordering from catalogues. Web user has login name which also serves as unique id. Web user could be in several states - new, active, temporary blocked, or banned, and be linked to a **shopping cart**. Shopping cart belongs to account.

Account owns customer **orders**. Customer may have no orders. Customer orders are sorted and unique. Each order could refer to several **payments**, possibly none. Every payment has unique id and is related to exactly one account.

Each order has current order status. Both order and shopping cart have **line items** linked to a specific product. Each line item is related to exactly one product. A **product** could be associated to many line items or no item at all.