Refactoring a poor quality design model

Order-Customer-OrderLine Example from Fowler

We will use the following example from Fowler to illustrate some point relating to coupling and cohesion in Object-oriented design. USE code and sample objects are provided.

Figure 3.1. A simple class diagram

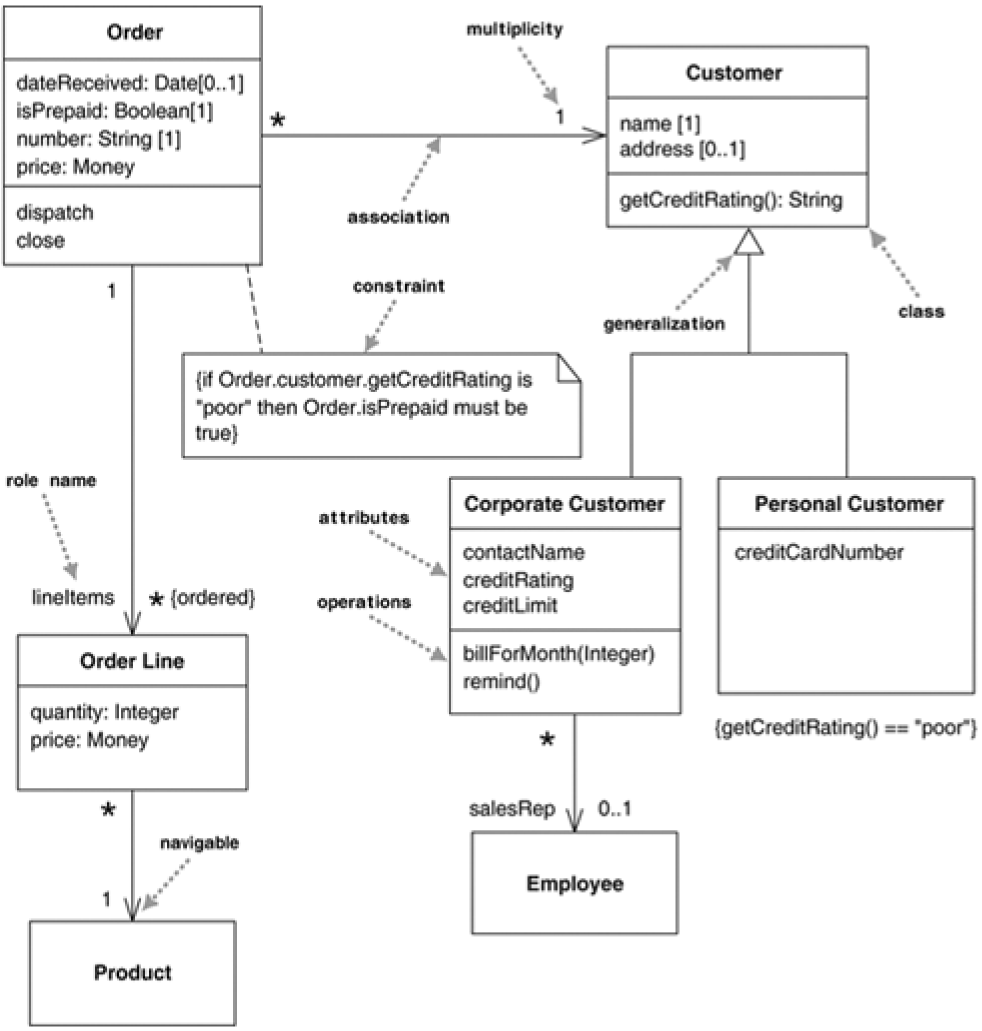
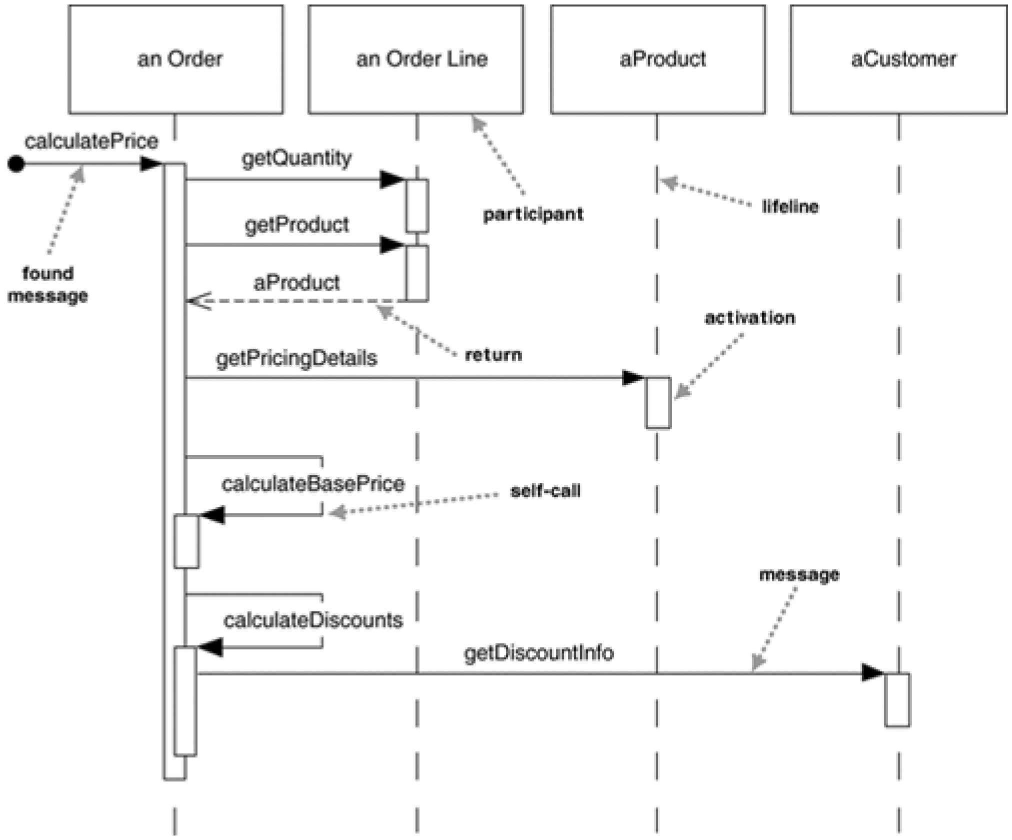
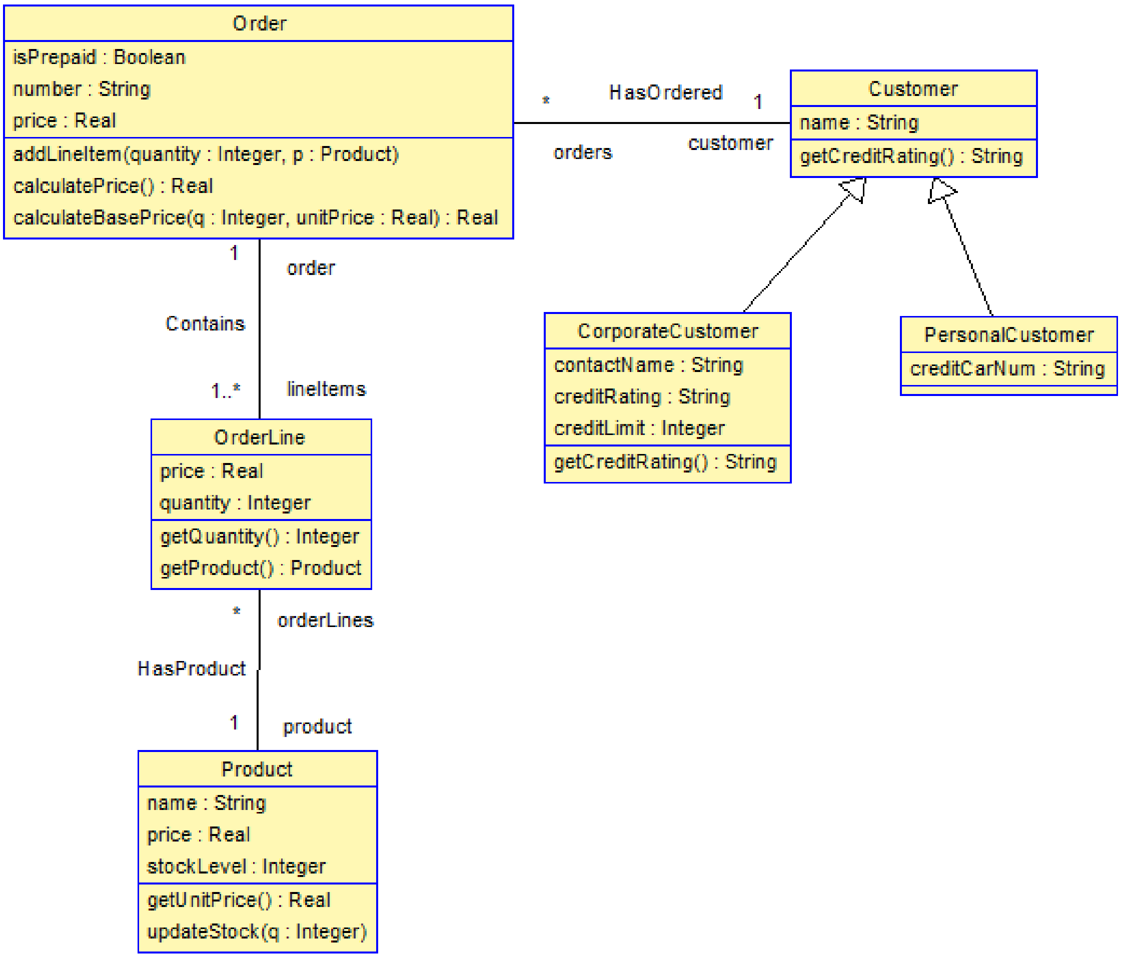


Figure 4.1. A sequence diagram for centralized control

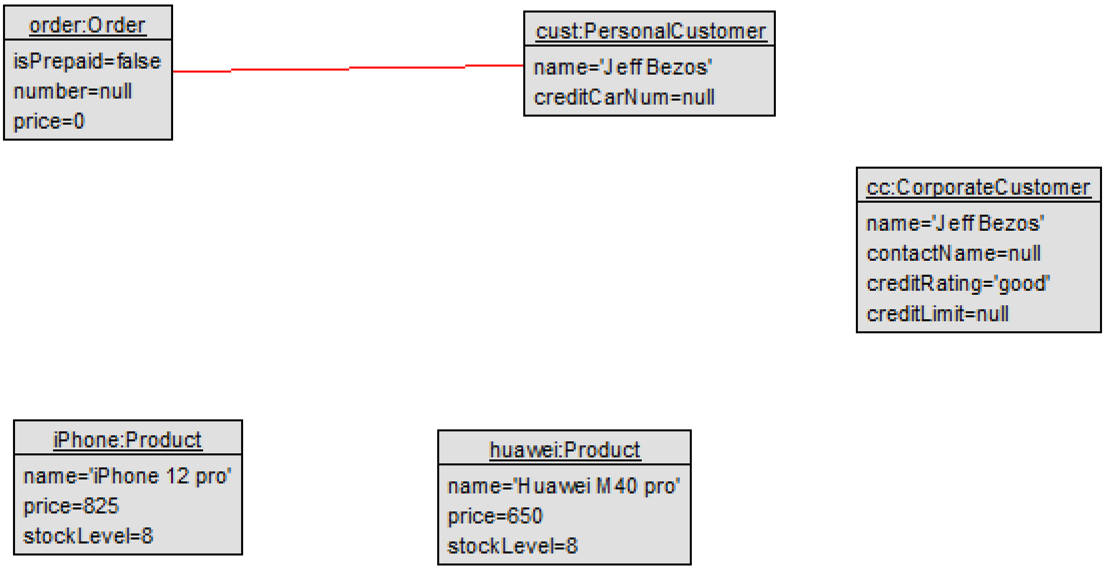


Centralised processing

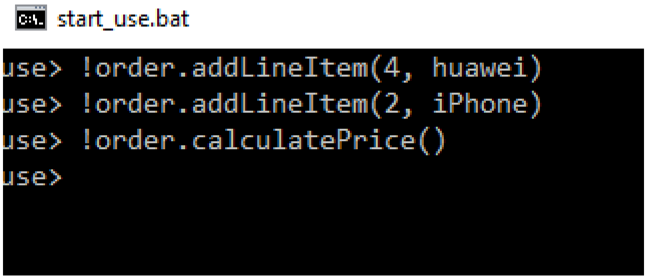
1. Load this model into USE from the USE file provided and arrange layout.



Initial Object Diagram in order.soil



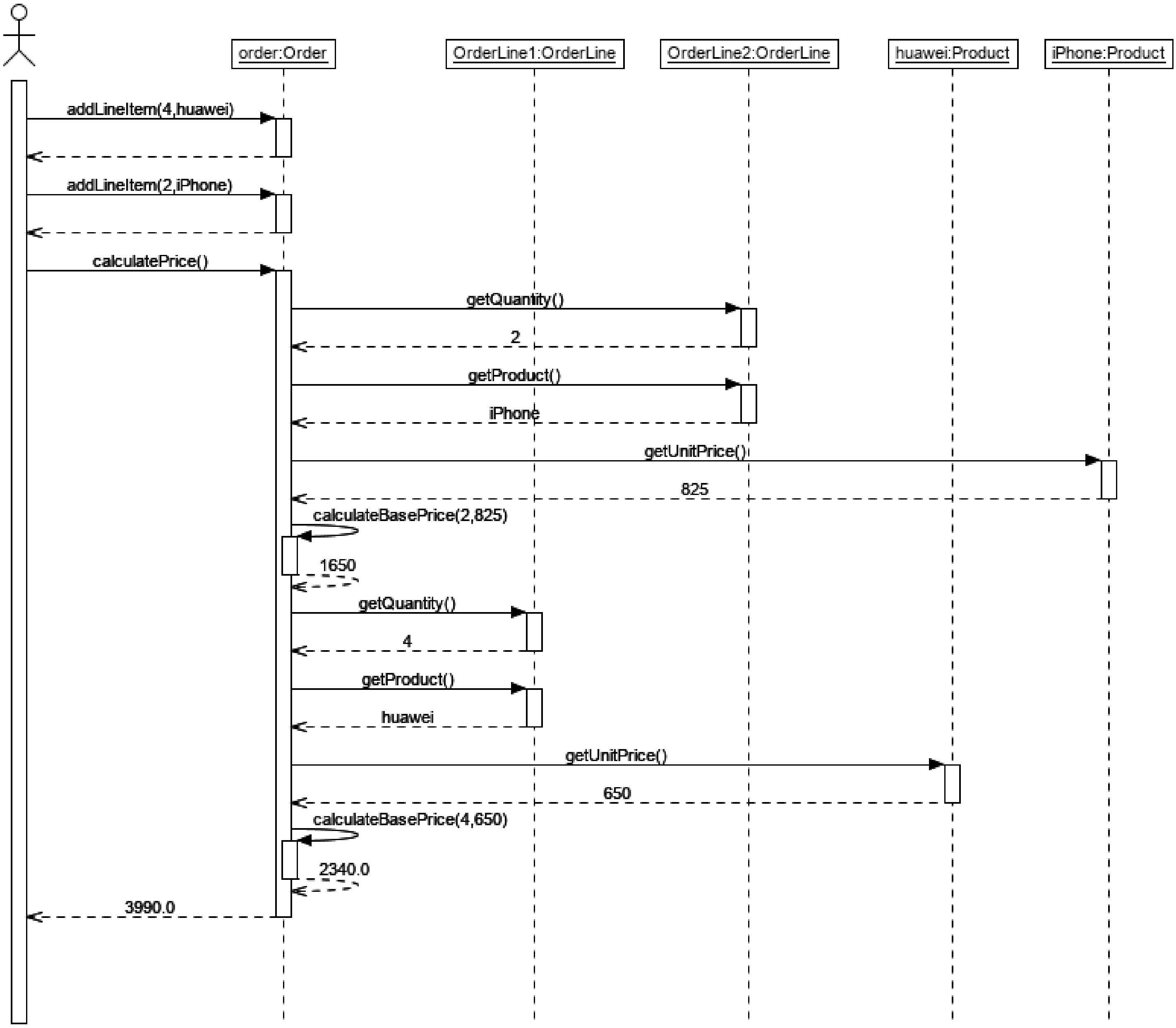
2. Load the object from the soil file: **open order.soil** and execute the following operations:



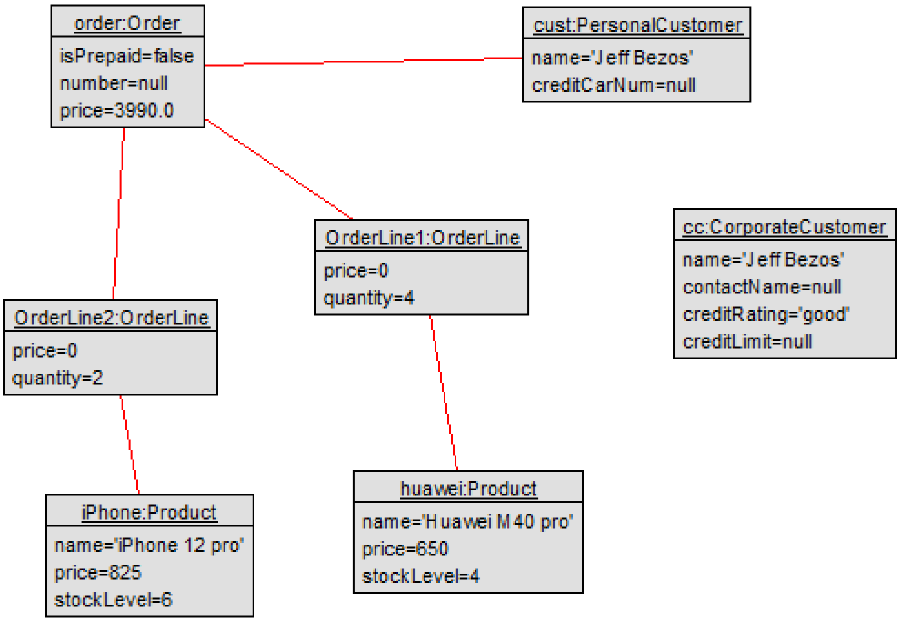
start_use.bat

```
use> !order.addLineItem(4, huawei)
use> !order.addLineItem(2, iPhone)
use> !order.calculatePrice()
use>
```

3. Create a sequence diagram view to get something like below after rearranging. See how **order** object does most of the processing and it is tightly coupled to the other objects.

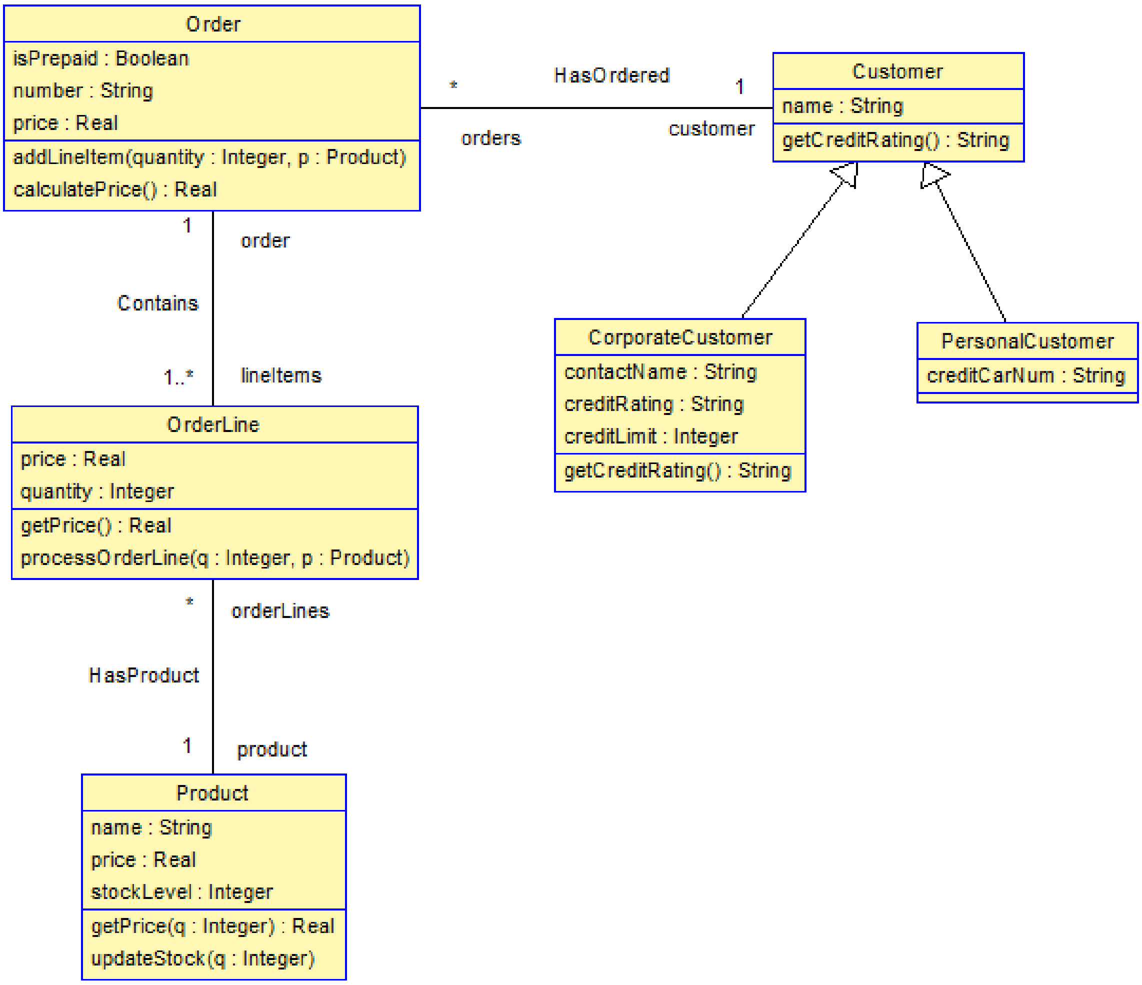


Object diagram should now look like:

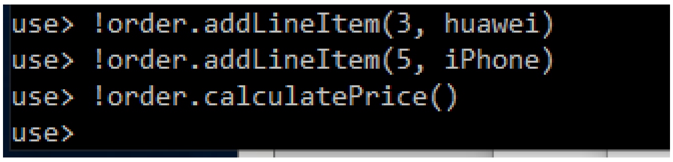


Distributed Processing

4. Refactor the USE code provided so that each object does its own processing. Use operations shown in the following refactored class diagram. There will be quite a bit of work in this step but it is worth completing.

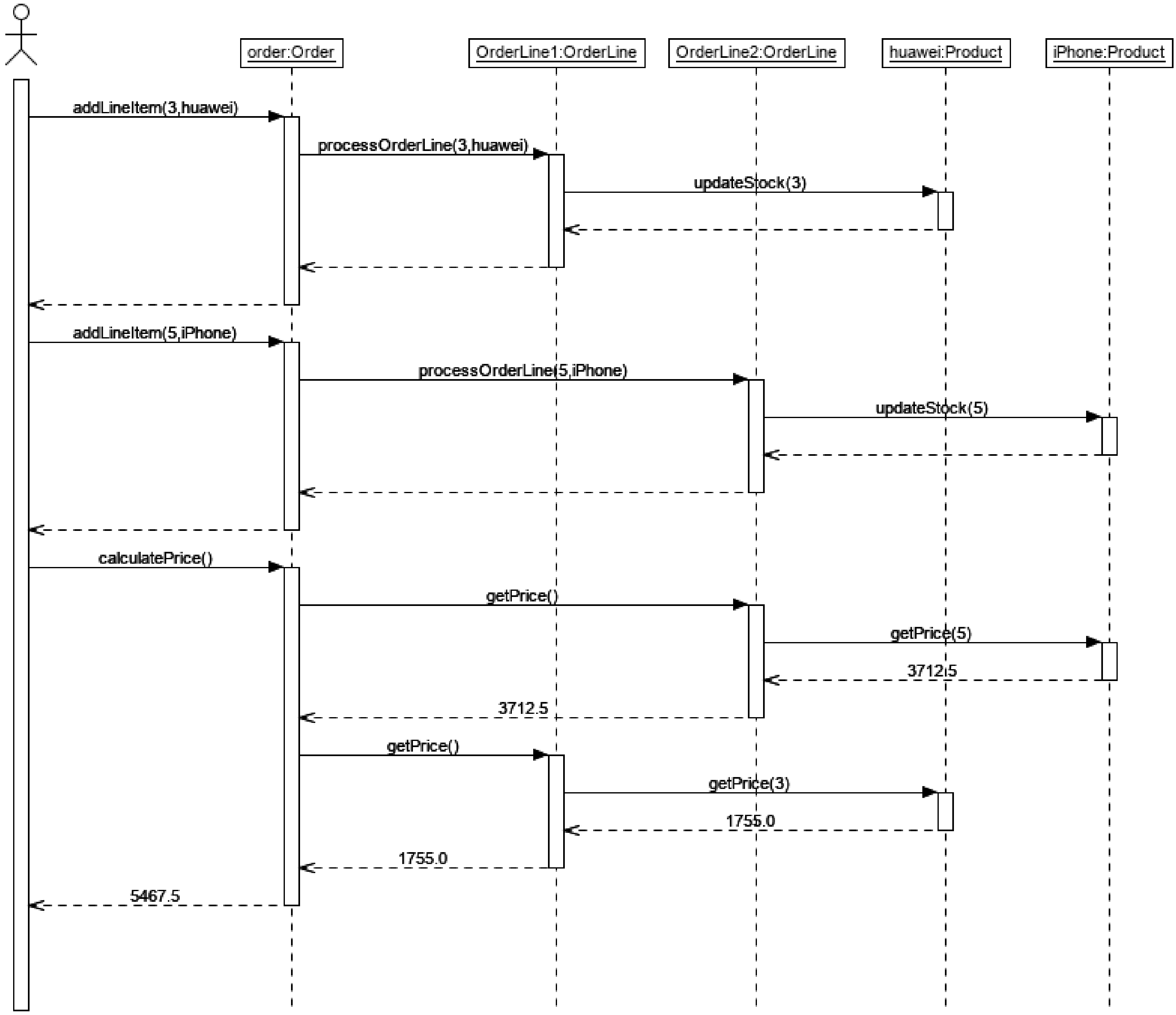


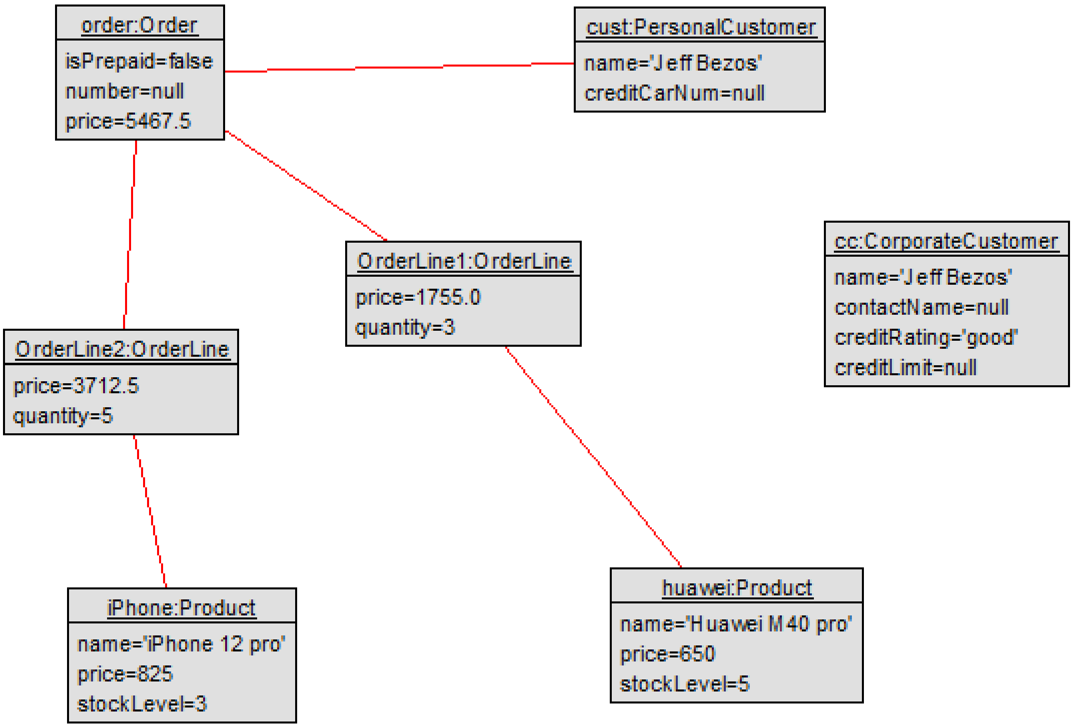
5. Then reload the objects and run the commands:



```
use> !order.addLineItem(3, huawei)
use> !order.addLineItem(5, iPhone)
use> !order.calculatePrice()
use>
```

6. Create a sequence diagram view to get something like below after rearranging. Note that the sequence diagram is much less cluttered, **order** does less processing. This represents a better design with more cohesion in **Order** and less coupling with other classes.



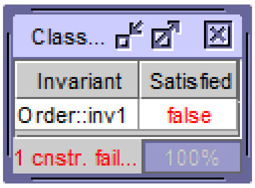


OCL Constrains

Finally write some constraints and test them.

7. Write a class constraint or invariant in OCL for class ***Order*** which says: if an order's customer has a poor credit rating, then the order's ***isPrepaid*** attribute must be ***True***.

   Is it true for the above object diagram? When written you can test this invariant by clicking on the ⚡ button. A small window like this should appear.



Invariant	Satisfied
Order::inv1	false

1 cnstr. fail... 100%

8. Write an appropriate simple precondition and postcondition for ***updateStock(q : Integer)*** in class ***Product*** and test them.