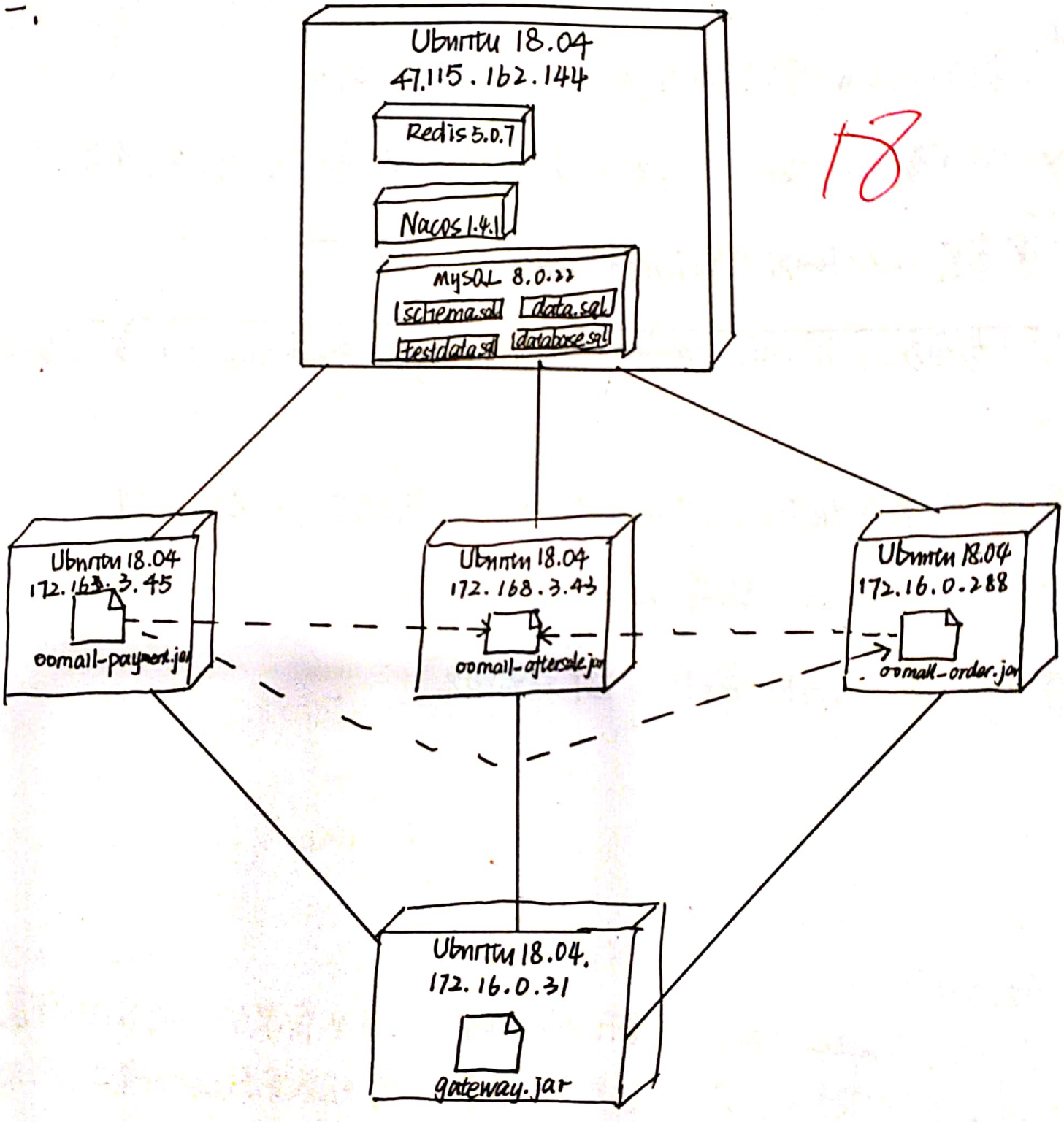
一、


Ubuntu 18.04
47.115.162.144
Redis 5.0.7
Nacos 1.4.1
MySQL 8.0.22
schema.sql
data.sql
Testdata.sql
database.sql
Ubuntu 18.04
172.168.3.45
oomall-payment.jar
Ubuntu 18.04
172.168.3.43
oomall-aftersale.jar
Ubuntu 18.04
172.16.0.288
oomall-order.jar
Ubuntu 18.04.
172.16.0.31
gateway.jar

18

二.

① 应改为
redisTemplate1: RedisTemplate ，少写了名称

② 第2步，不需要写 skuDao.modifyInventory，
直接写 2. ret = modifyInventory(shopId, skuId, quantity) 即可
包括后面的 redisTemp1、this、redisTemp2

③第1个opt的框画得偏上

④ 在 skuDao 里的 ret == true 的情况没有画出，ret==true 也应扣库存.

⑤ RedisTemplate 的生命盒中间断了，应连续

⑥ 第2个 opt 框的判断条件应写为 ret==true

⑦ 第6步 对 redisTemplate2 的方法进行调用，没有画出 ⟶

⑧ new的方法应改为调用 sku 的 create方法

⑨ 第2个 opt 框没有包住，应画得更长一些

⑩ ret、inventory、stock、message 变量没有来源.

⑪ ~~Rocketmq 应写回的是 扣减过库存，而非修改后的库存量 stock~~

三.

Dao层和Service层都对Redis进行了访问，查了两次Redis
浪费了Redis，增加Redis负载，并且Dao层还做了扣库存的处理

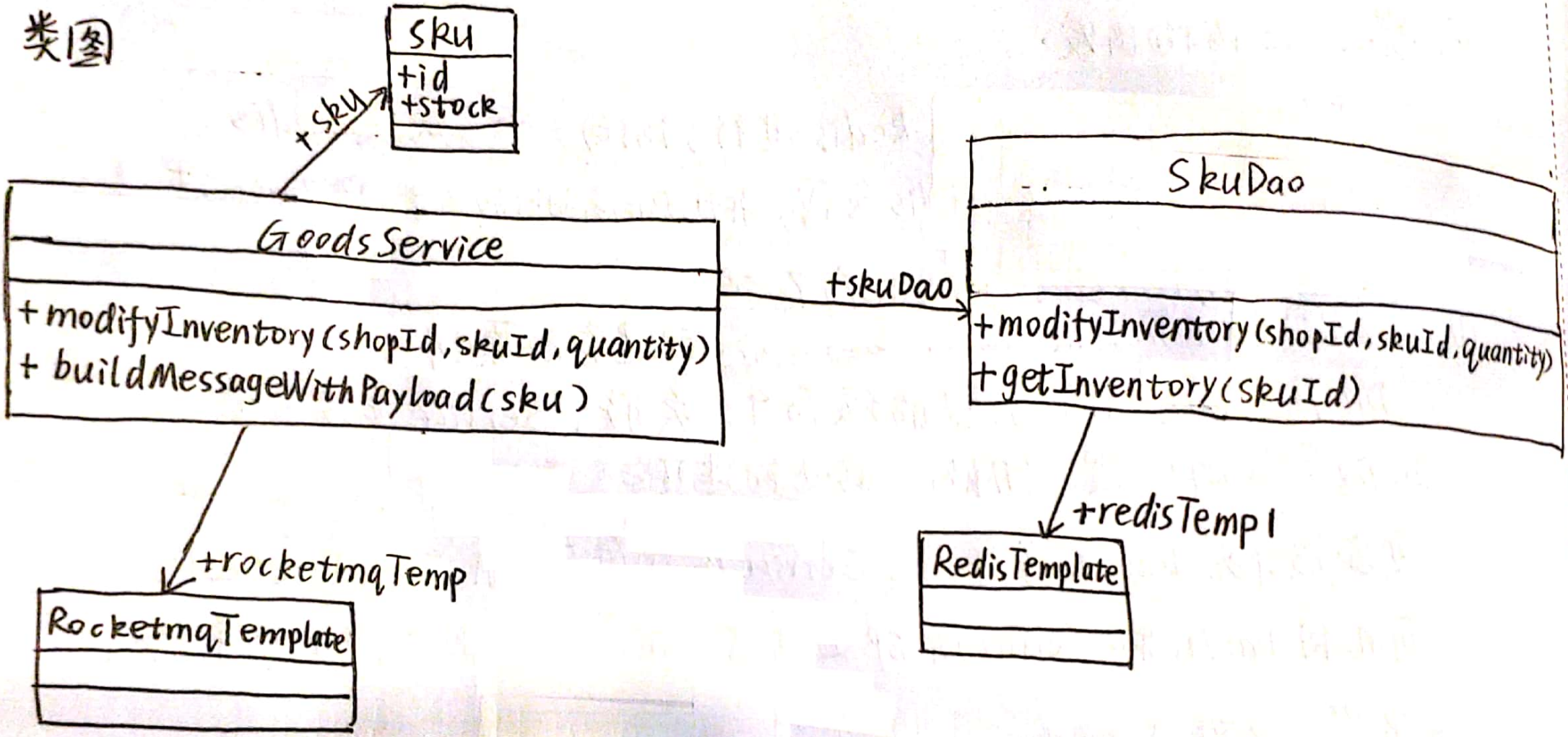
不符合高内聚、低耦合、创建者的原则
Dao层应负责对象模型的创建和关联，Service主要负责
业务逻辑的处理、协调。此处创建其实是查询。
更应设计为Dao层拿库存，Service层扣库存，并写回数据库。
可以使Dao层和Service层都更内聚，同时隔离两层，降低耦合。

低耦合不符，导致也
不符合PV原则，未来如果有新的修改库存方式的需求，
既需要改动Dao层，又需改动Service层，

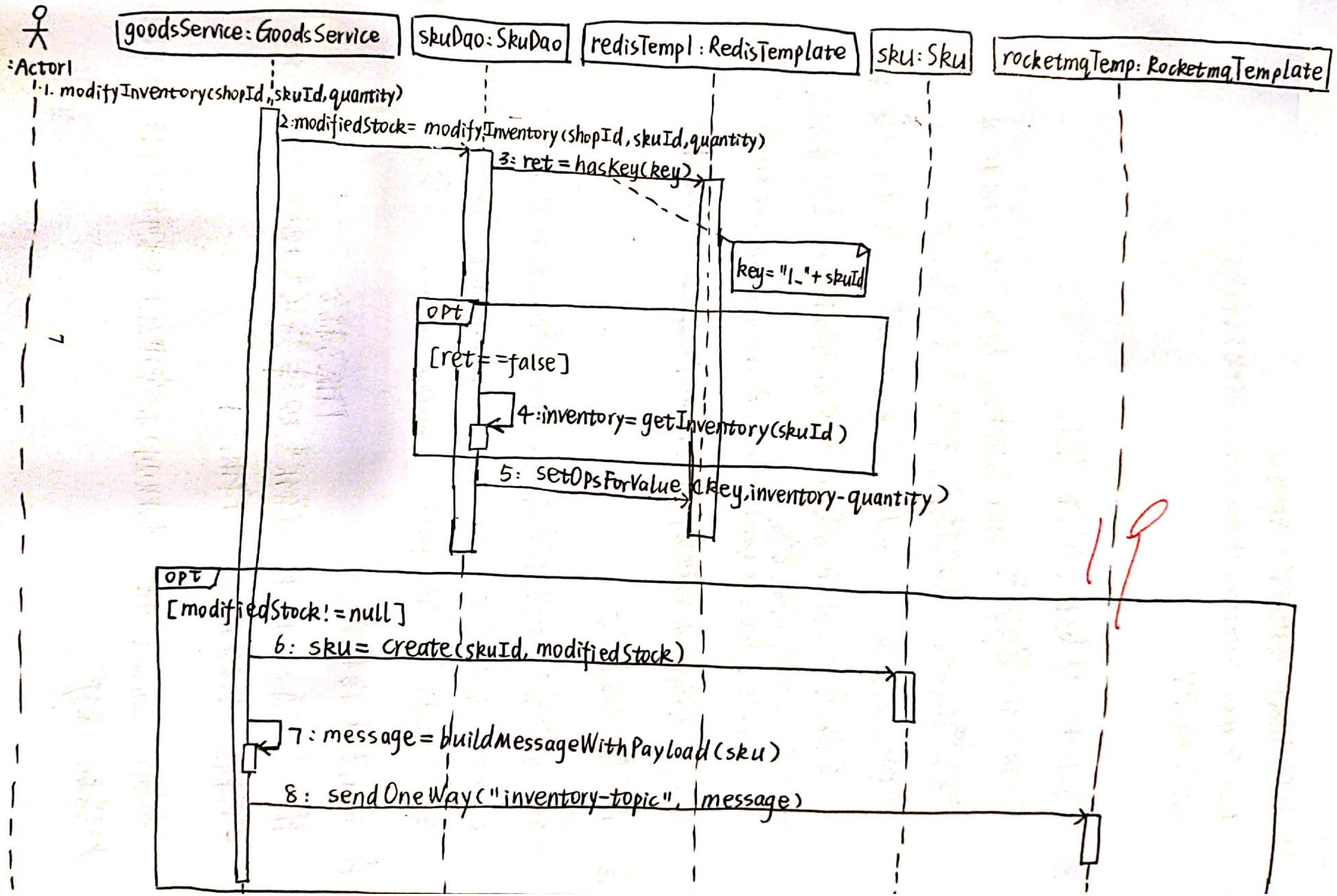
10

四.

类图


Sku
+id
+stock
+sku
GoodsService
+modifyInventory(shopId,skuId,quantity)
+buildMessageWithPayload(sku)
+skuDao
SkuDao
+modifyInventory(shopId,skuId,quantity)
+getInventory(skuId)
+redisTempl
RedisTemplate
+rocketmqTemp
RocketmqTemplate

modifiedStock 表示修改后的库存.



王：①该设计使用了桥接器的设计模式，将计算优惠限制的过程抽离出来，作为计算~~计算~~优惠折扣类的一个属性，~~[illegible]~~通过discount类引用limitation类，体现了间接的设计原则；

②. 对于不同的折扣计算方式实现了不同的子类，对于不同的优惠限制方式也实现了不同的子类，每个子类根据模板方法的设计模式，在方法中达到不同的计算功能，这体现了高内聚、低耦合、多态的设计原则；

③. 当增加一种折扣计算方式或限制计算方式时，不需要对整个优惠规则重新排列组合，只需要增加一个子类，不会影响其他的任何代码，这体现了保护变化的设计原则。

它不会对动态模型~~[illegible]~~产生影响

10

大. 类图: ①合理之处 类图中 service层调dao并且将有关goods操作调用dubbo接口，符合信息专家原则，由goods完成有关商品的操作

②不合理之处：PieceItem，WeightItem最好分开来设计不好，应当用一个基类然后进行继承，这两样就符合Grasp的多态，并且保护变化考虑到以后可能还有更多的计算运费的方法，不止重量，单品.

③ service层和dao层都有bo对象的成员变量，如都有PieceItem.

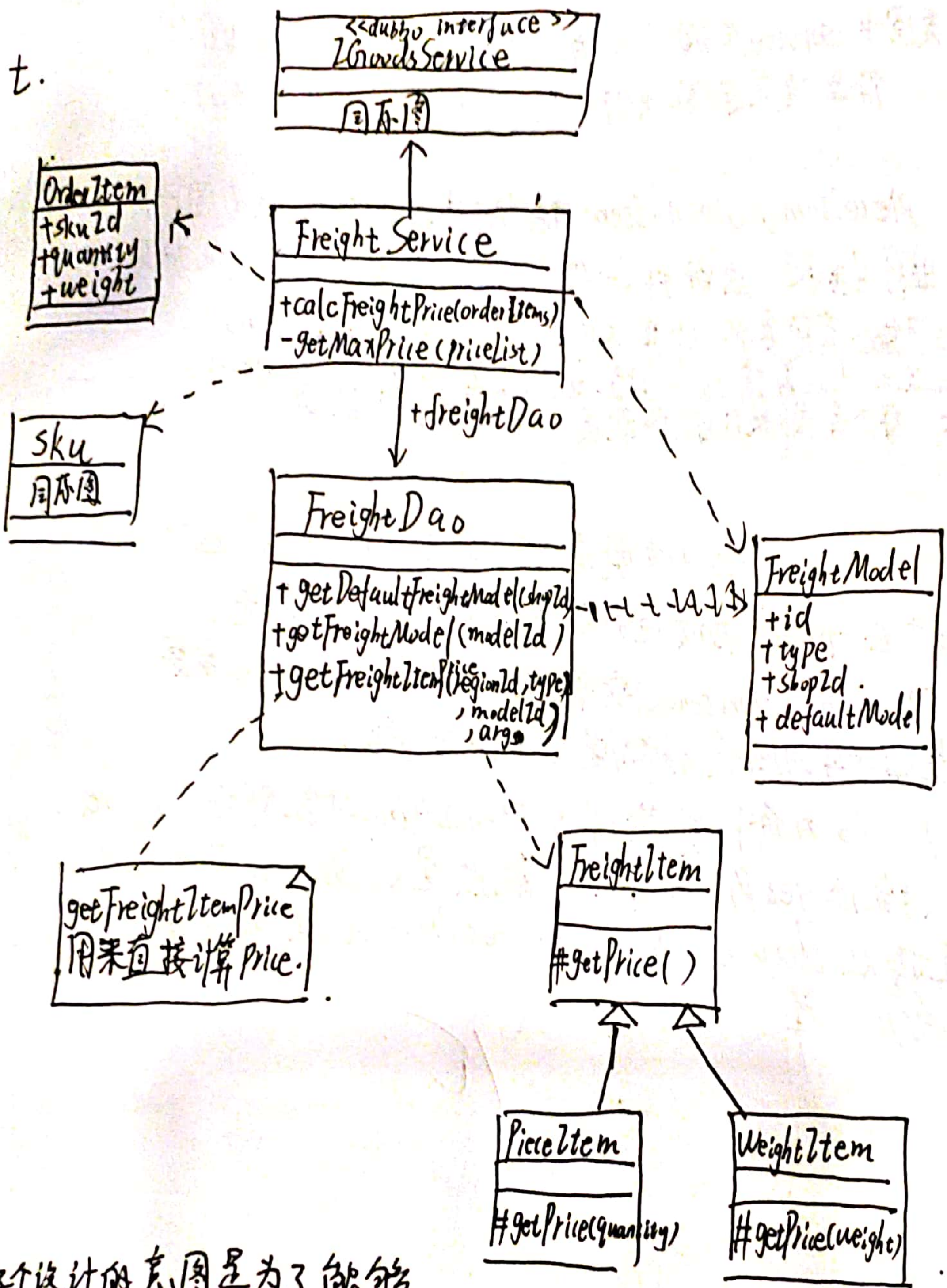
时序图：①合理之处，每个层都执行自己的职责，符合信息专家的原则.

②不合理之处：如果models的类型有很多，则在loop models这个循环中，对于每个model都要进行很多的判断，并且新增新的模板.loop orderItems需要修改，loop models也需要修改，整体计算时间会非常慢，不符合.PV.

③ 这里以9，10为例 9获得了Item的bo对象却在service调bo对象的getPrice方法，既然是从dao层获得bo应该直接从dao层调bo的getPrice方法来计算运费不需要多此一举.

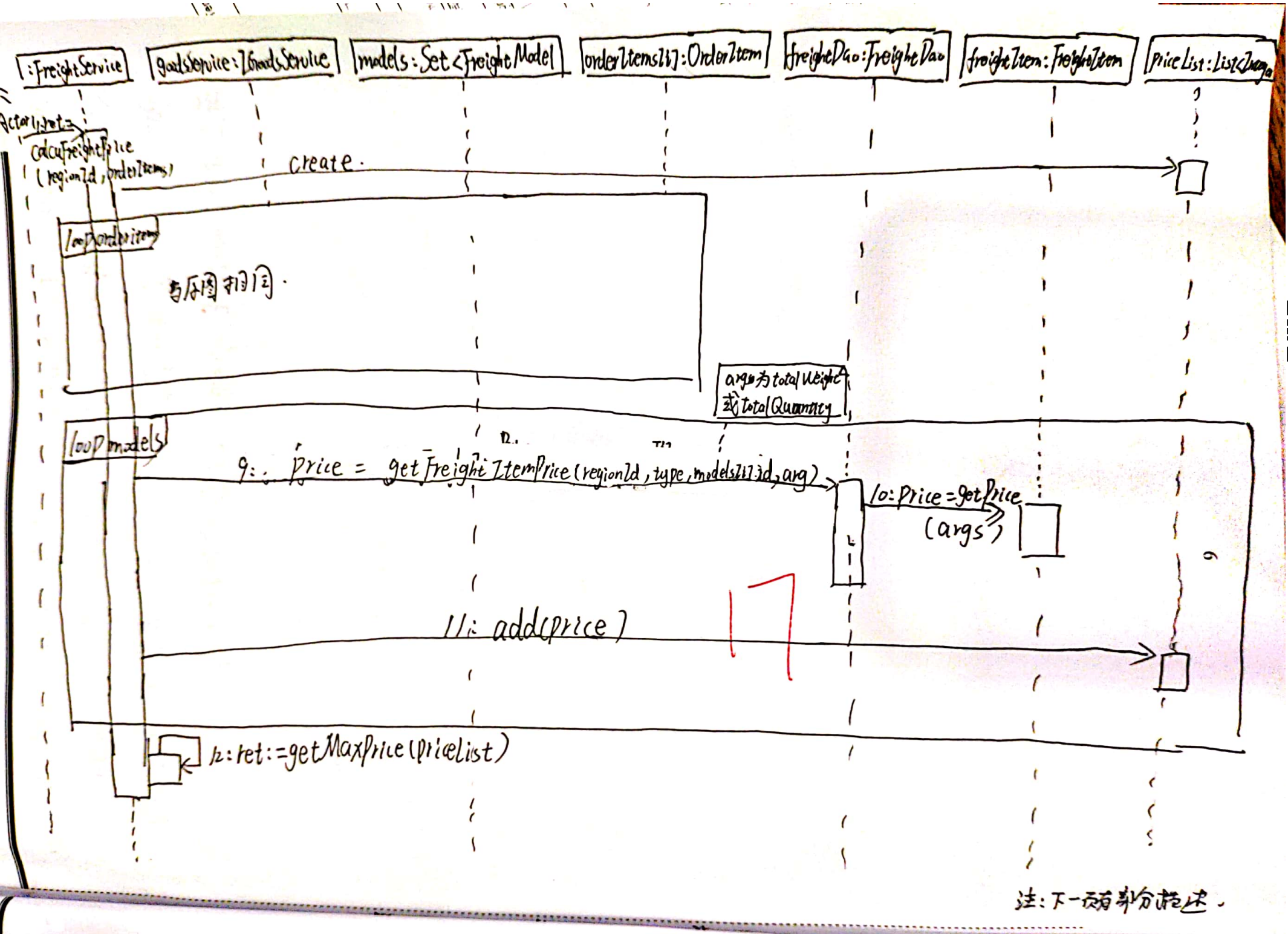
10.

t.



这个设计的意图是为了能够
让之后增加更多的freightItem，~~为了并且~~降低Dao与多个Item的耦合，
并将Item的代码内聚在freightItem的实现中，这里还用到模板方法，
每个子类只改自己部分的变化，~~不需要~~父类接口并没有改变，保护了变化，符合PV.

getFreightItemPrice


:FreightService
goodsService:GoodsService
models:Set<FreightModel
orderItems[i]:OrderItem
freightDao:FreightDao
freightItem:FreightItem
priceList:List<Integer>
actor
calcuFreightPrice
(regionId, orderItems)
create.
loop orderItems
当前项目.
args为totalWeight
或totalQuantity
loop models
9: price = getFreightItemPrice(regionId, type, modelsId.id, arg)
10: price=getPrice
(args)
11: add(price)
12: ret:=getMaxPrice(priceList)
注:下一项为部分描述.