

# 面向对象分析与设计

## Object Oriented Analysis and Design

### ——产品模块设计

#### Design of Product Catalog Module

邱明 博士

厦门大学信息学院

mingqiu@xmu.edu.cn

2024年秋季学期

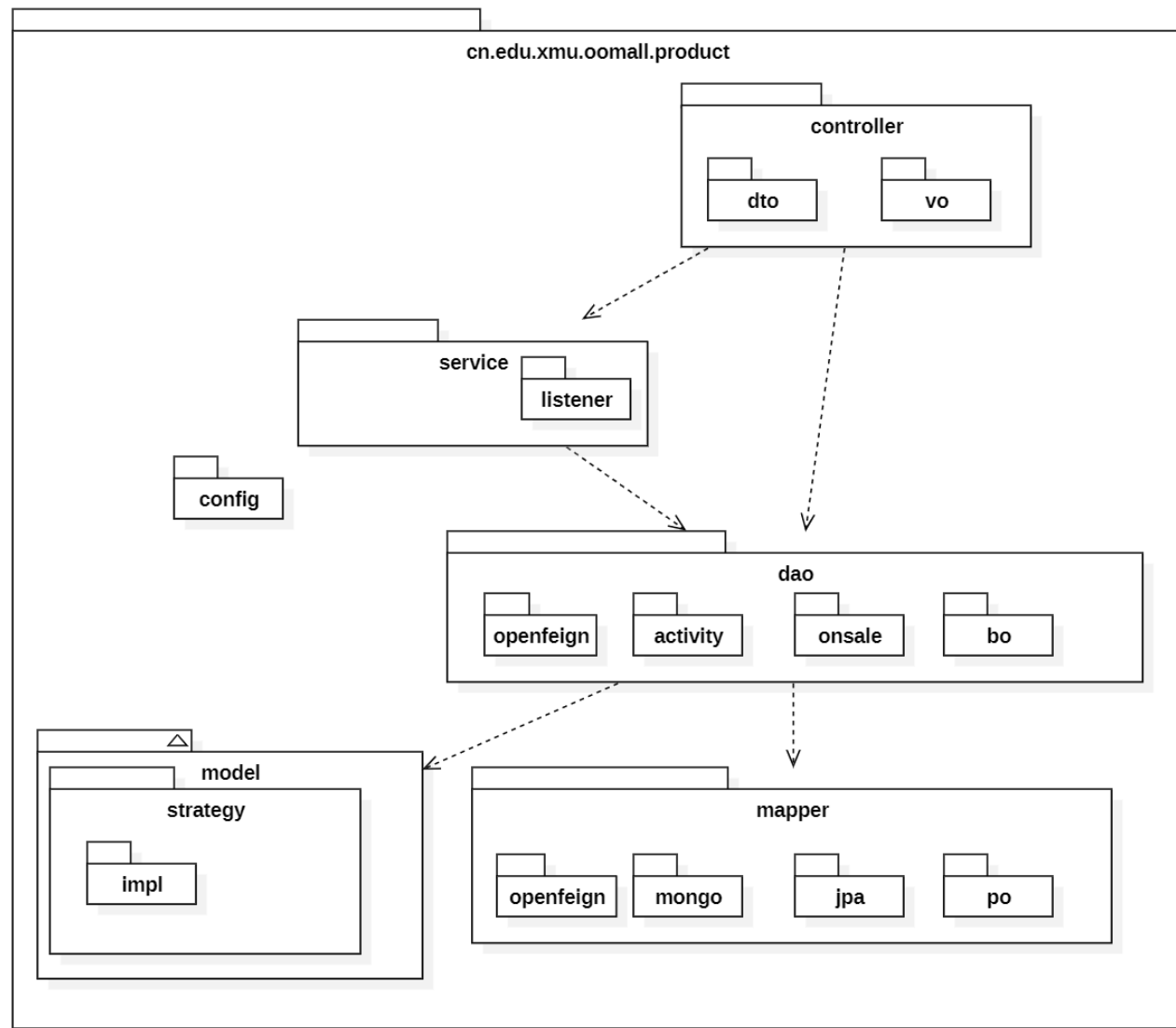
# 1. 静态模型设计

## Static Model Design



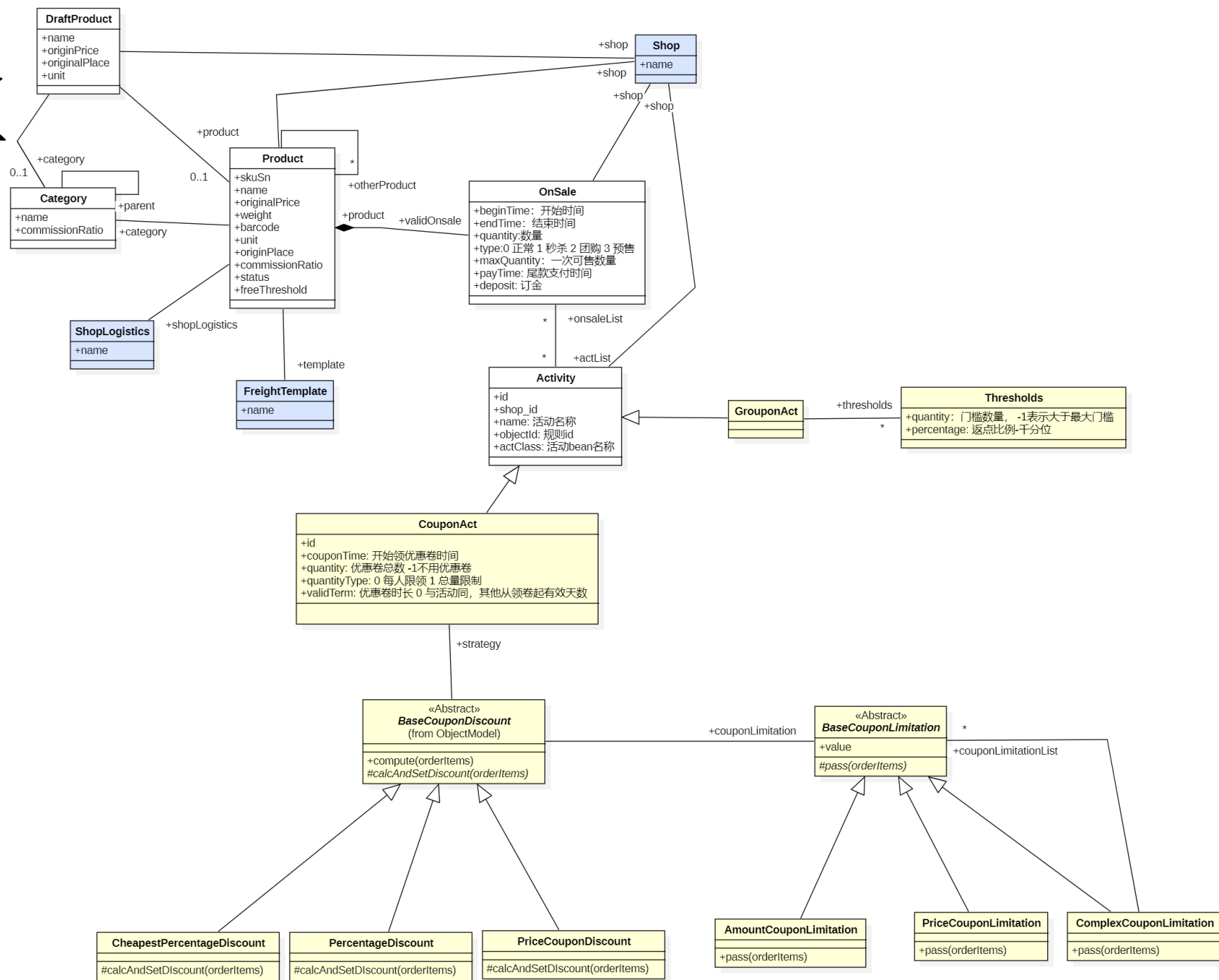
# 1.1 包结构

## Package



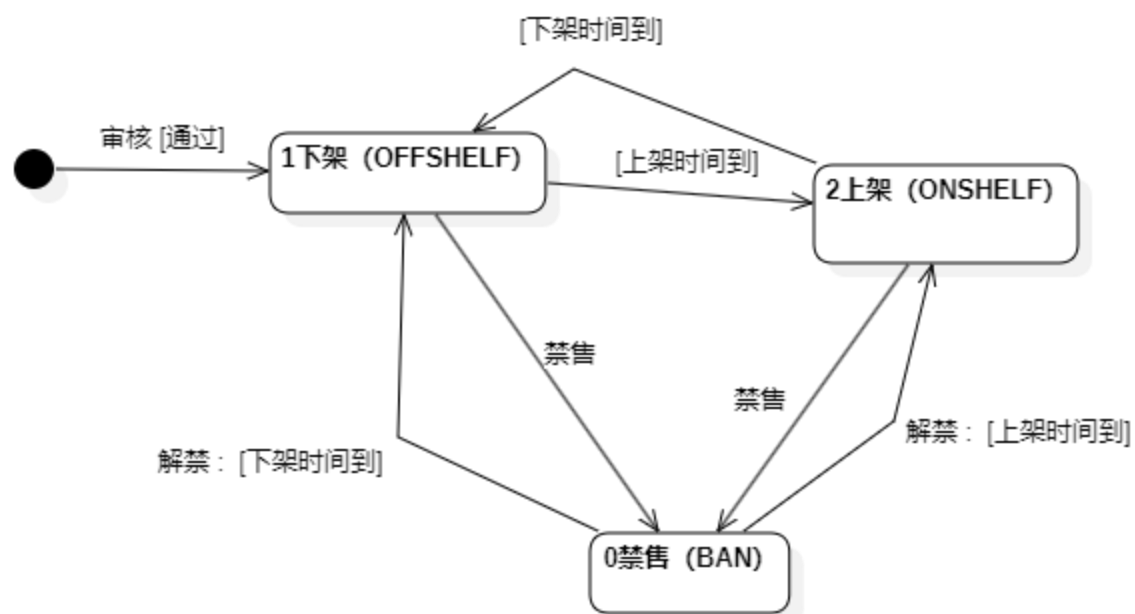
# 1.2 对象模型

## Object Model



# 1.2 对象模型

## Object Model



产品对象状态



# 1.2 对象模型

## Object Model

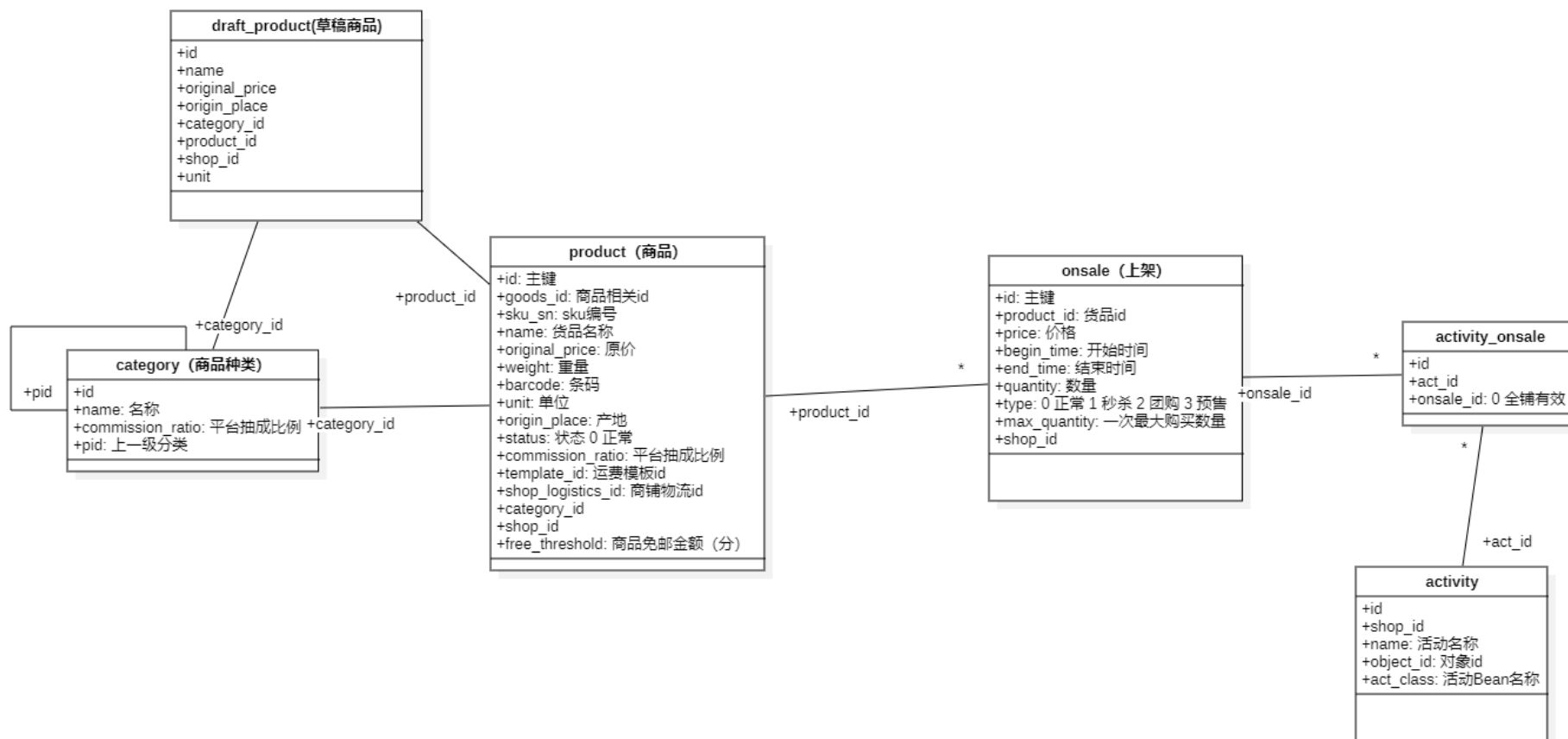


活动对象状态



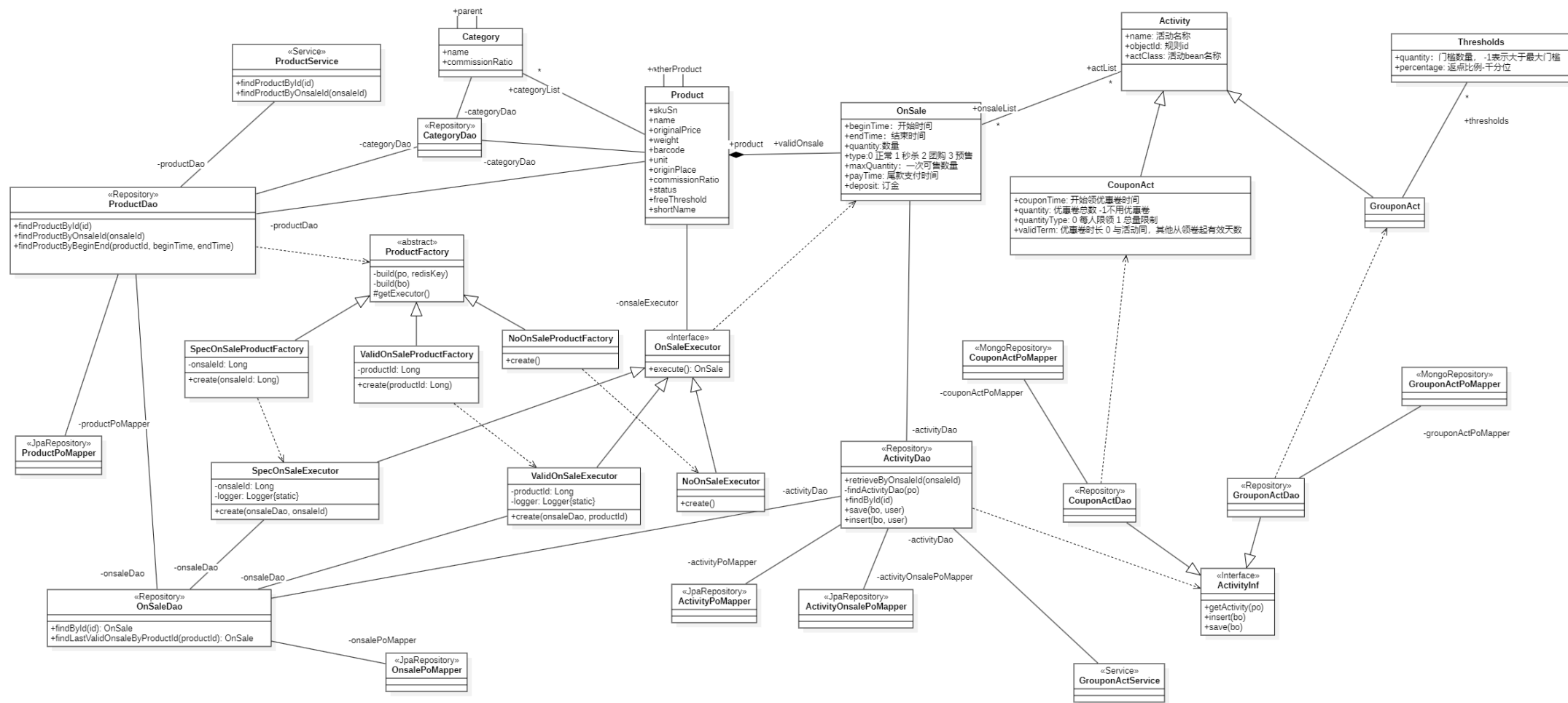
# 1.3 数据库设计

## Database Design



# 1.4 静态模型

## Static Model



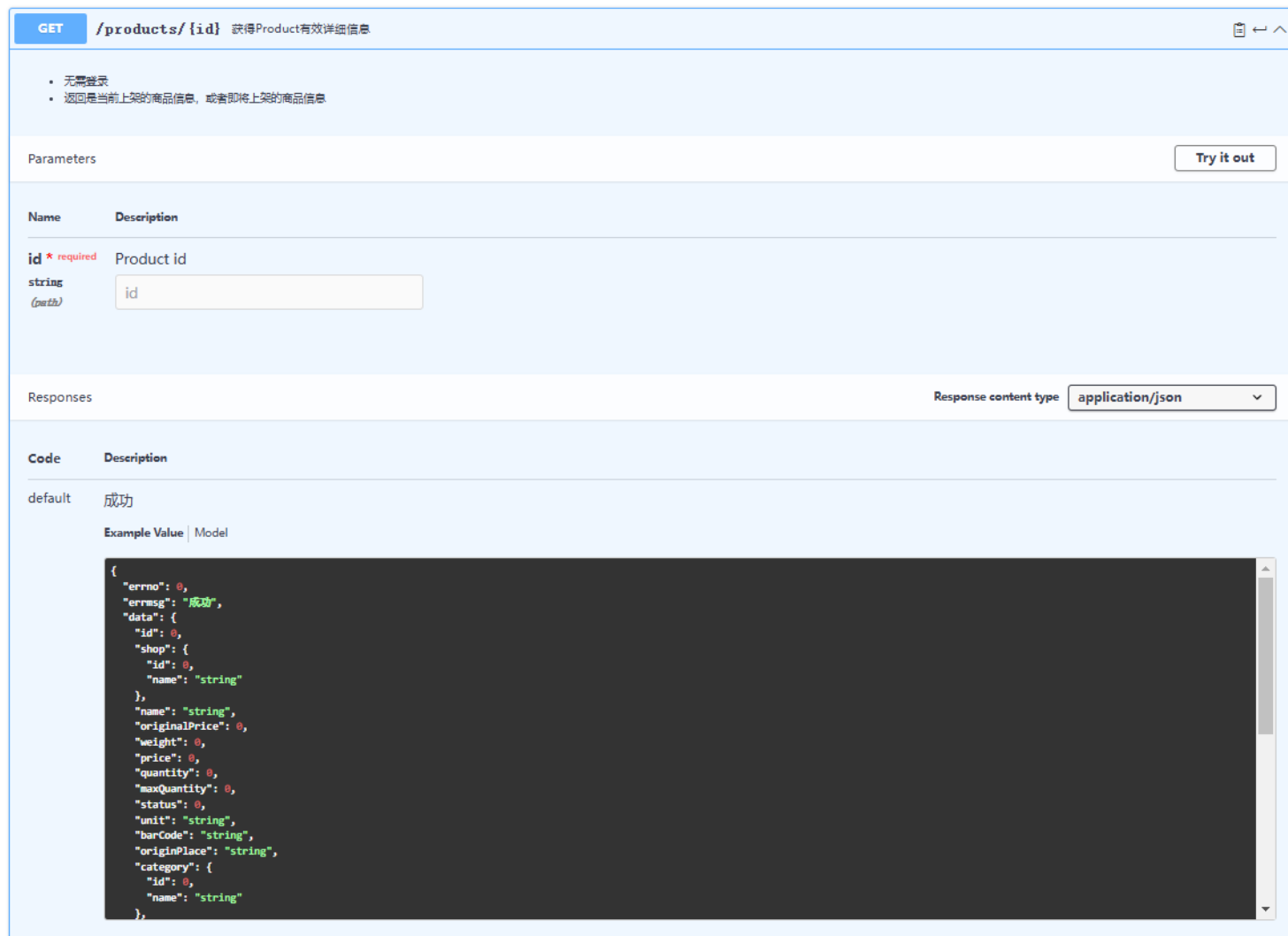


# 2. 动态模型设计

**Dynamic Model Design**



## 2.1 顾客查看商品详细信息



GET /products/{id} 获得Product有效详细信息

- 无需登录
- 返回当前上架的商品信息，或者即将上架的商品信息

Parameters Try it out

Name	Description
<b>id</b> <small>* required</small> string <small>(path)</small>	Product id <input type="text" value="id"/>

Responses Response content type: application/json

Code	Description
default	成功

Example Value | Model

```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 0,
    "shop": {
      "id": 0,
      "name": "string"
    },
    "name": "string",
    "originalPrice": 0,
    "weight": 0,
    "price": 0,
    "quantity": 0,
    "maxQuantity": 0,
    "status": 0,
    "unit": "string",
    "barCode": "string",
    "originPlace": "string",
    "category": {
      "id": 0,
      "name": "string"
    }
  }
}
```



## 2.1 顾客查看商品详细信息

GET /onsales/{id} 获得Product的指定onsale历史信息

- 无需登录
- 显示历史的商品价格信息

Parameters Try it out

Name	Description
<b>id</b> * required string (path)	onsale id <input type="text" value="id"/>

Responses Response content type: application/json

Code	Description
default	成功

Example Value | Model

```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 0,
    "shop": {
      "id": 0,
      "name": "string"
    },
    "name": "string",
    "originalPrice": 0,
    "weight": 0,
    "price": 0,
    "quantity": 0,
    "maxQuantity": 0,
    "status": 0,
    "unit": "string",
    "barCode": "string",
    "originPlace": "string",
    "category": {
      "id": 0,
      "name": "string"
    },
  },
}
```



## 2.1 顾客查看商品详细信息

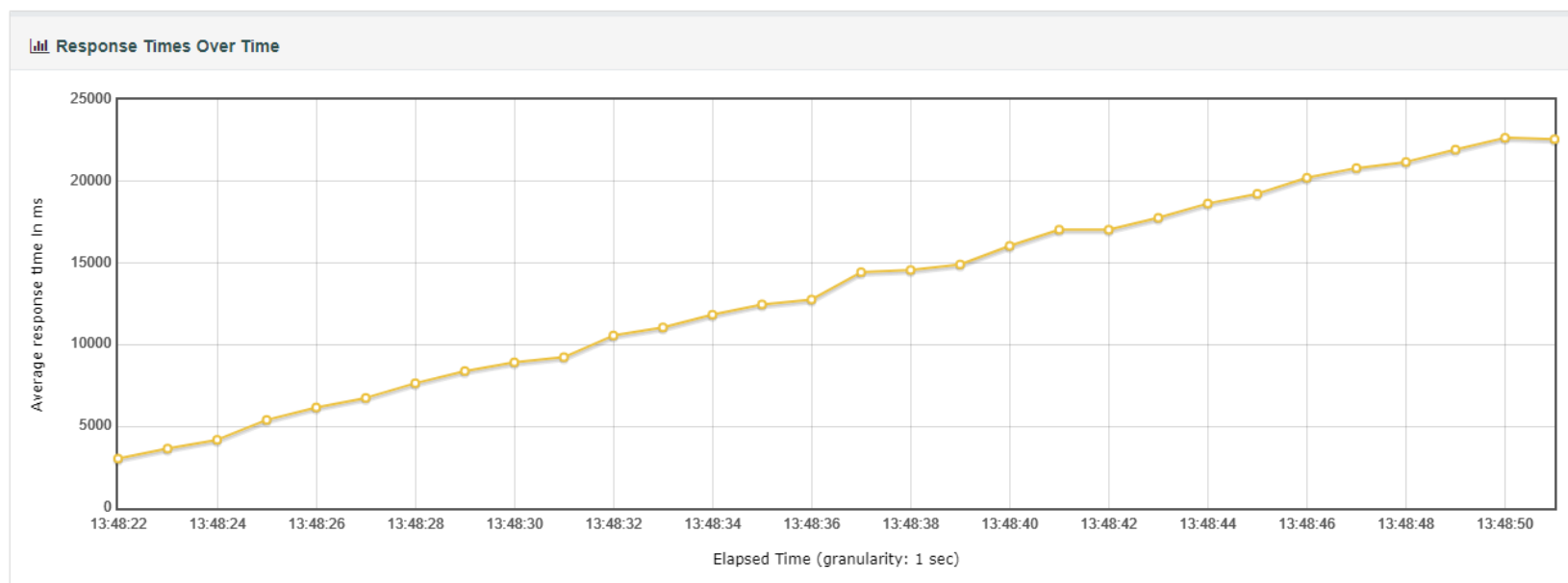
- 两个API有相似性，需要尽量重用代码
- 信息来自于对象模型中多个对象
  - 需要多次查表
  - 而且在大负载的情况下，性能衰减严重

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	400	4	1.00%	15050.78	2931	23888	15480.50	22527.80	22887.70	23690.16	12.54	11.47	1.63
redis	400	4	1.00%	15050.78	2931	23888	15480.50	22527.80	22887.70	23690.16	12.54	11.47	1.63

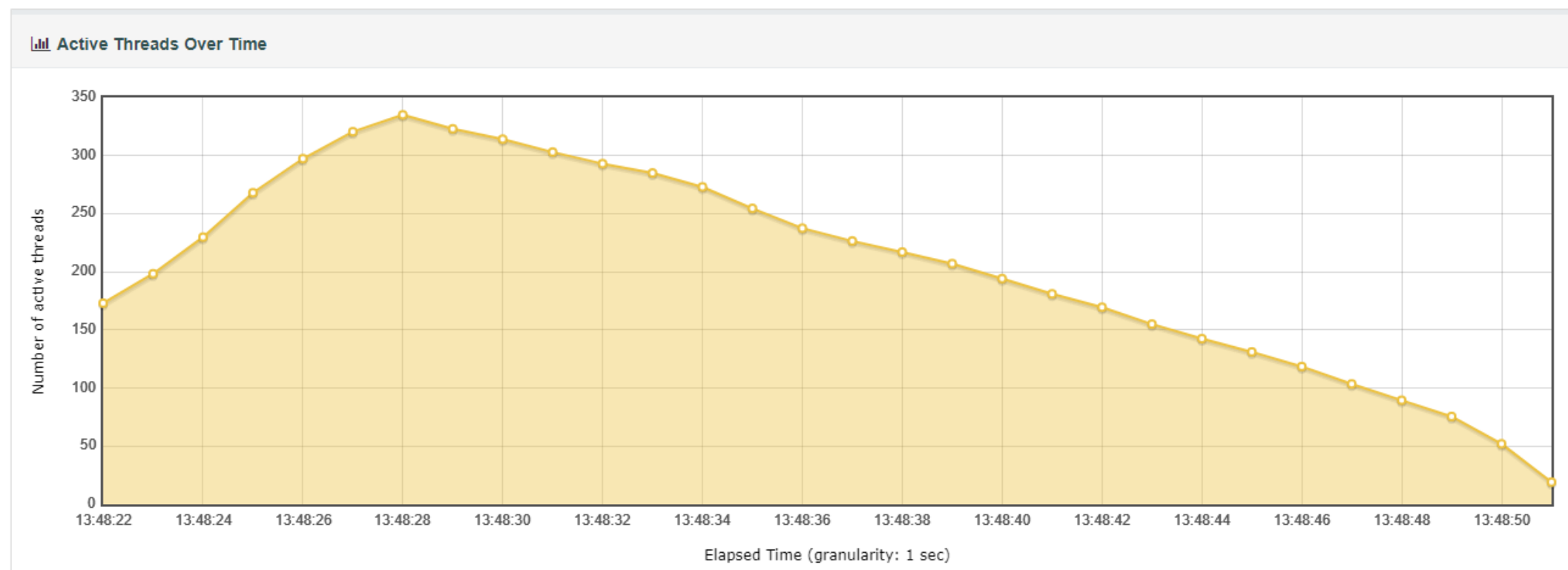


## 2.1 顾客查看商品详细信息

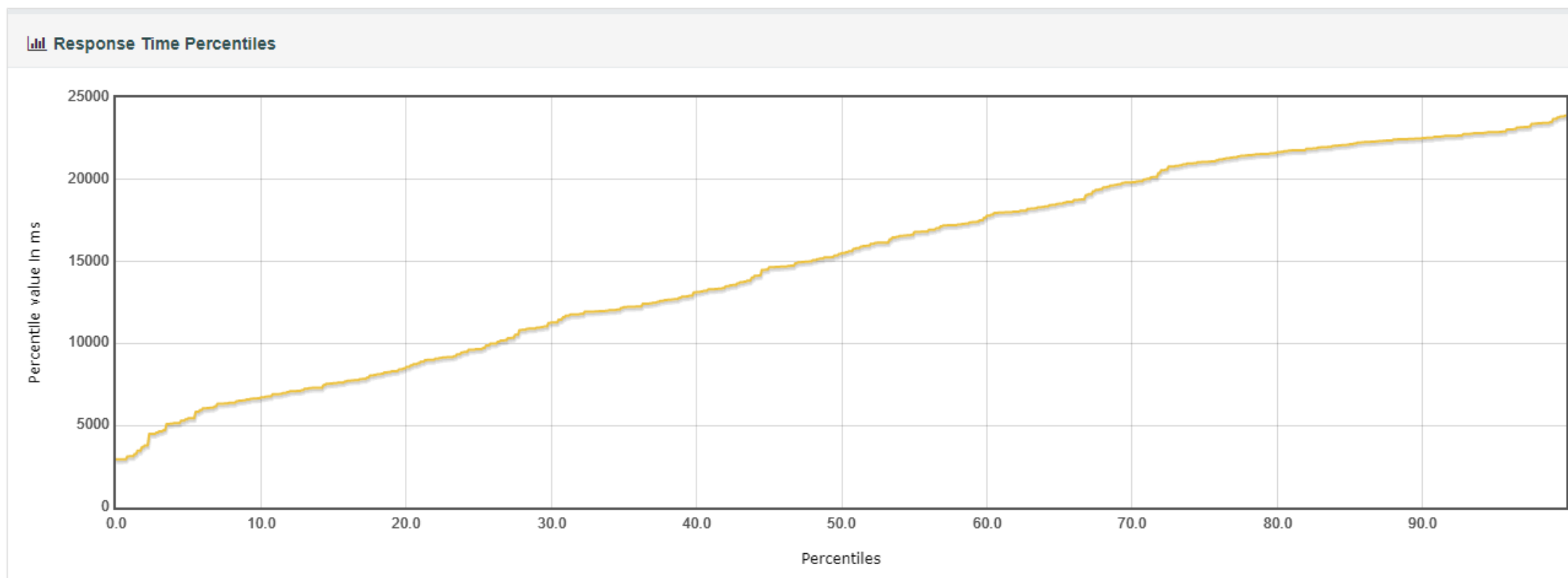
Requests		Executions			Response Times (ms)						Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	400	4	1.00%	15050.78	2931	23888	15480.50	22527.80	22887.70	23690.16	12.54	11.47	1.63
redis	400	4	1.00%	15050.78	2931	23888	15480.50	22527.80	22887.70	23690.16	12.54	11.47	1.63



## 2.1 顾客查看商品详细信息



## 2.1 顾客查看商品详细信息



## 2.1 顾客查看商品详细信息

- 解决的办法
  - 利用Redis缓存，将重复查找的数据改由缓存读取
  - 缓存po，bo还是vo？
  - 关系如何缓存？

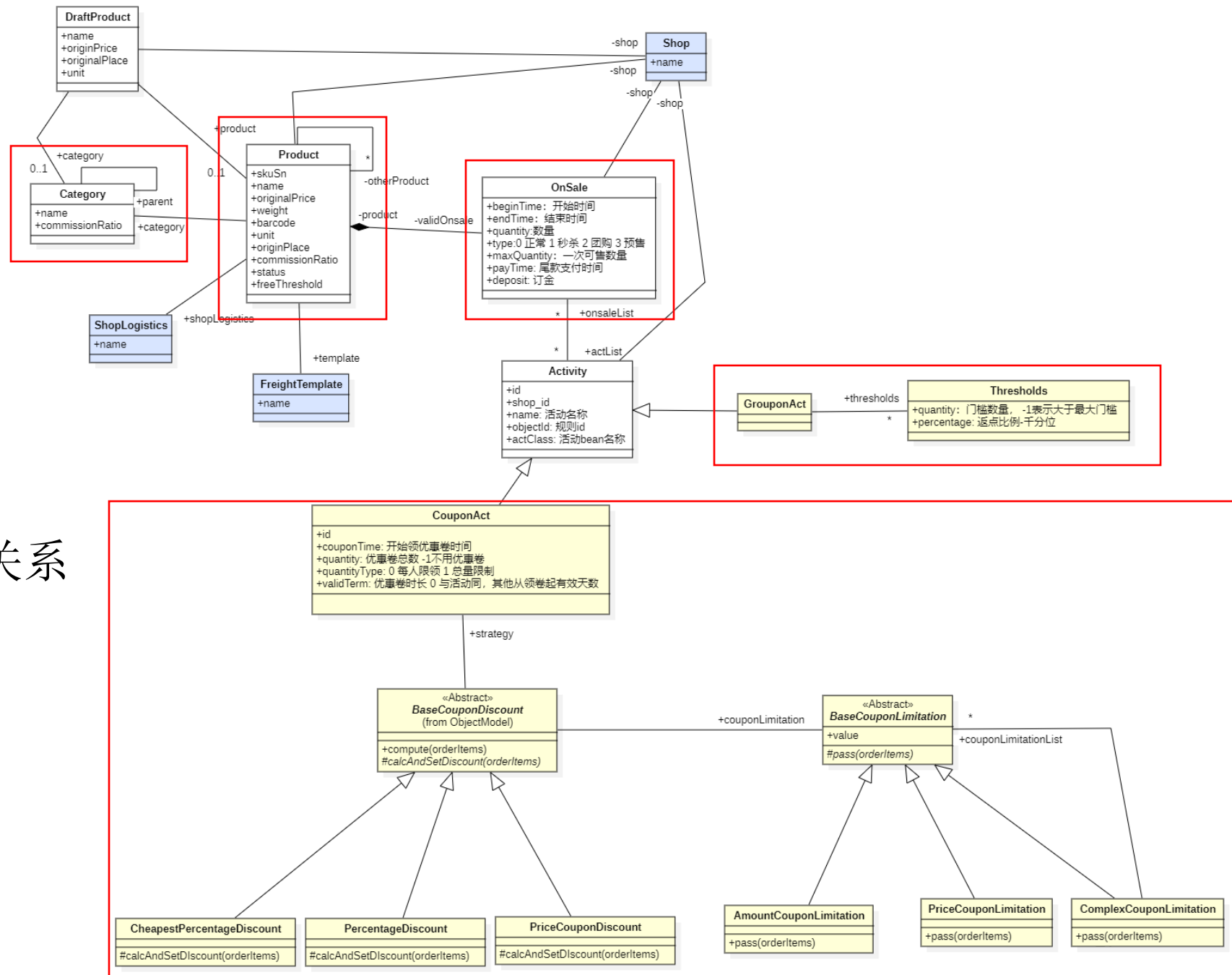




## 2.1 顾客查看商品详细信息

- 需要缓存

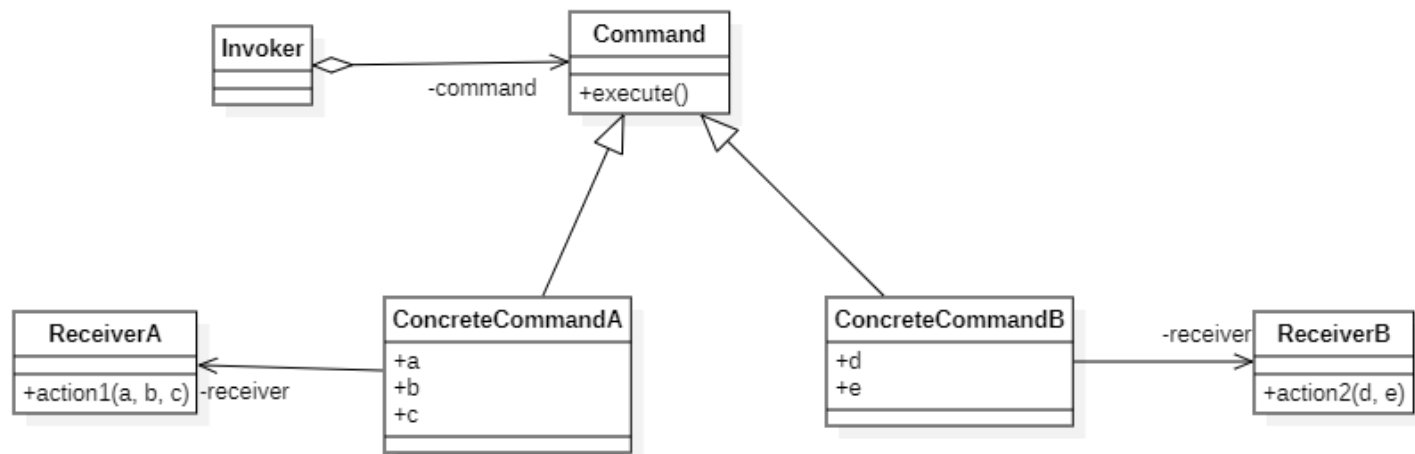
- Category
- Product
- Onsale
- 各类Activity
- 以及它们之间的关系

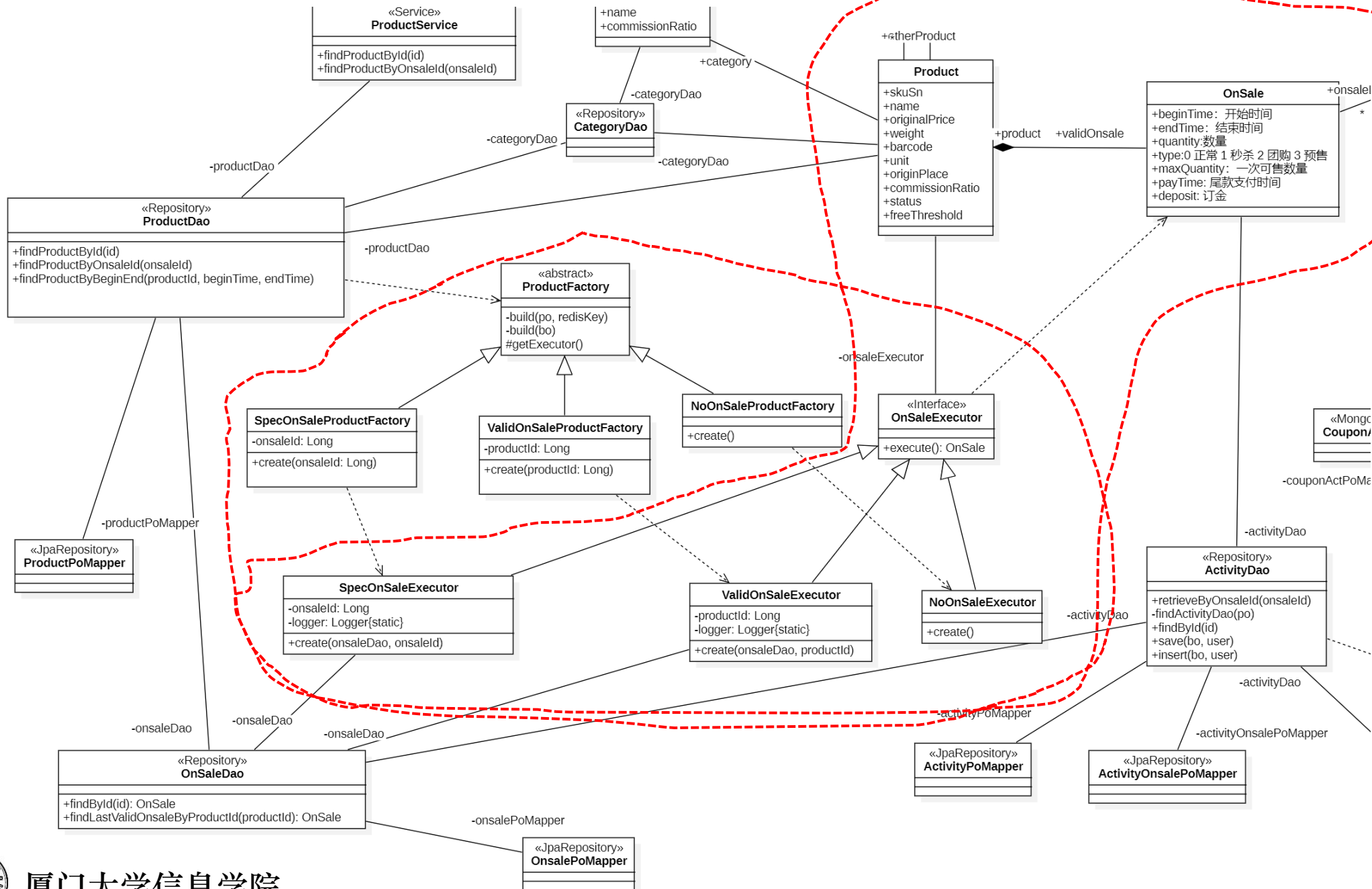


## 2.1 顾客查看商品详细信息

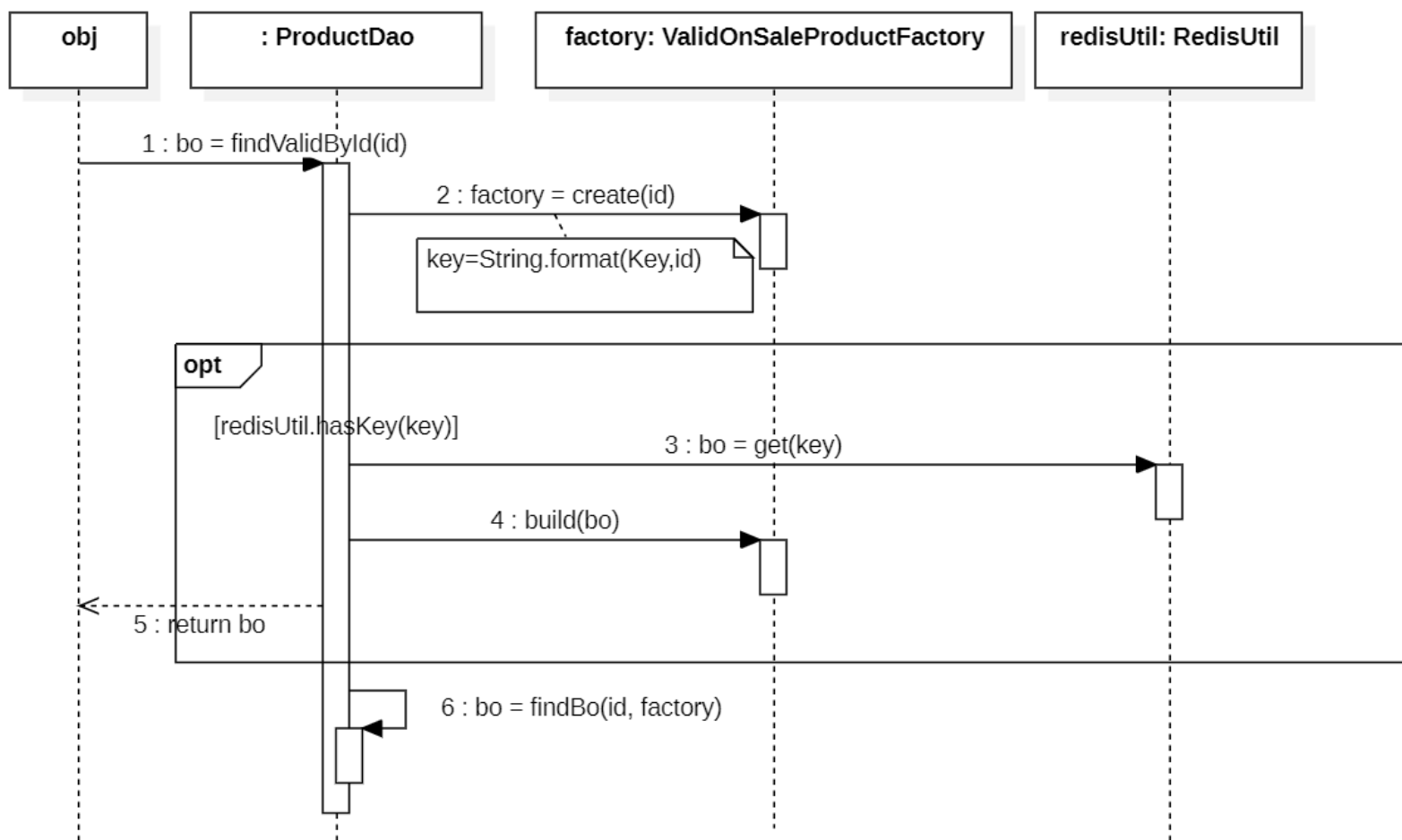
- 命令设计模式 (Command)

- 将一个请求封装为一个对象，从而使你可用不同的请求对客户进行参数化；
  - 实现发送者和接收者之间的低耦合，发送请求的对象只需要知道如何发送请求，而不必知道如何完成请求。
  - 将请求本身变成一个对象向未制定的应用对象提出请求。

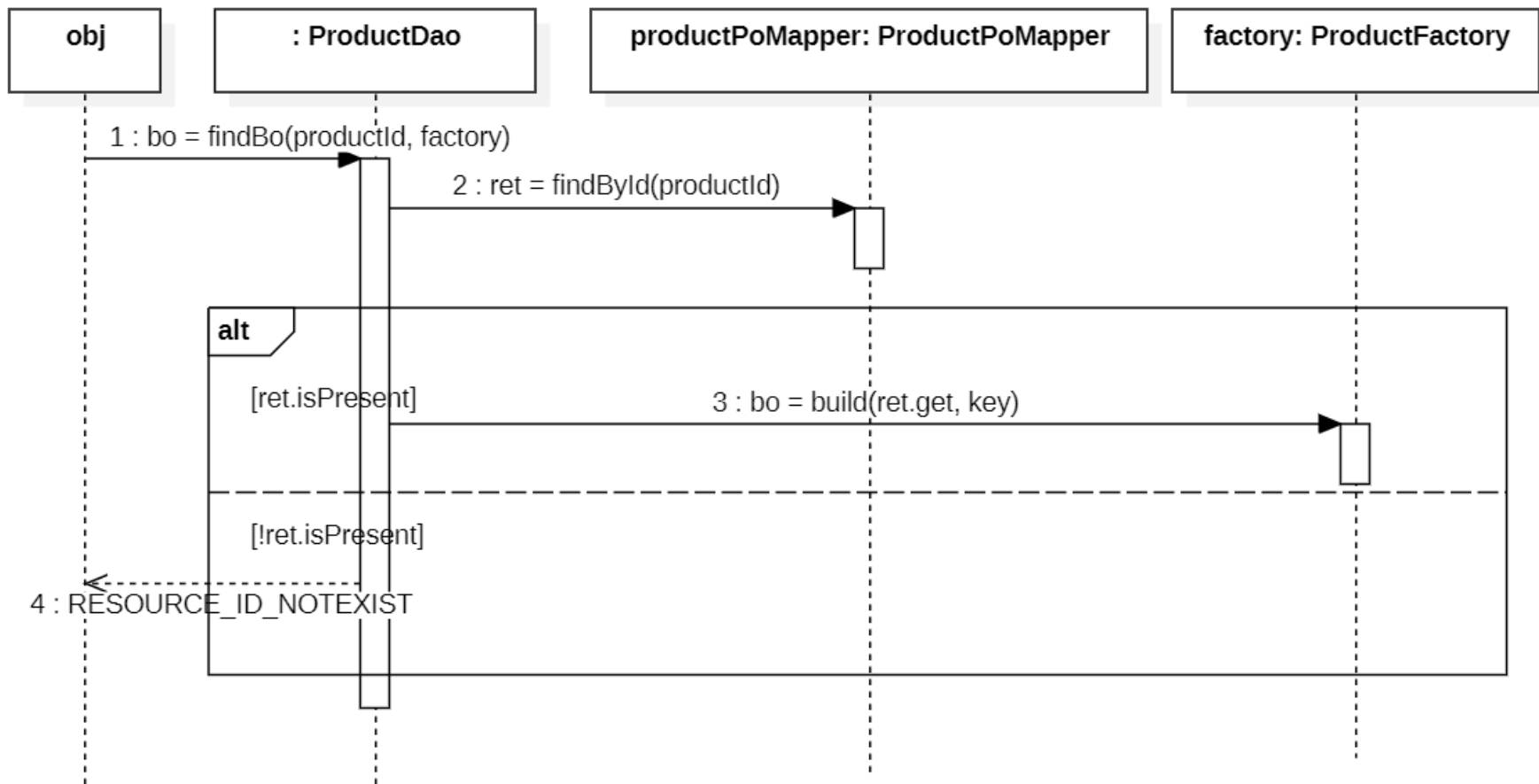




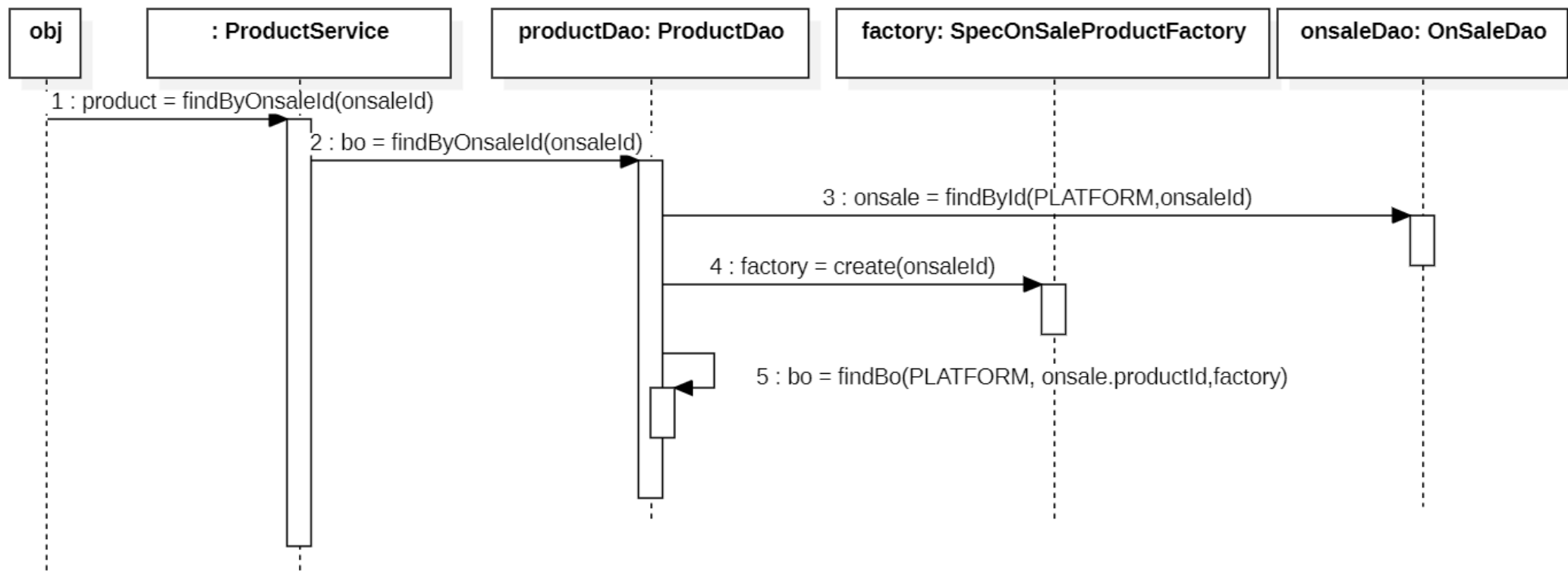
## 2.1 顾客查看商品详细信息



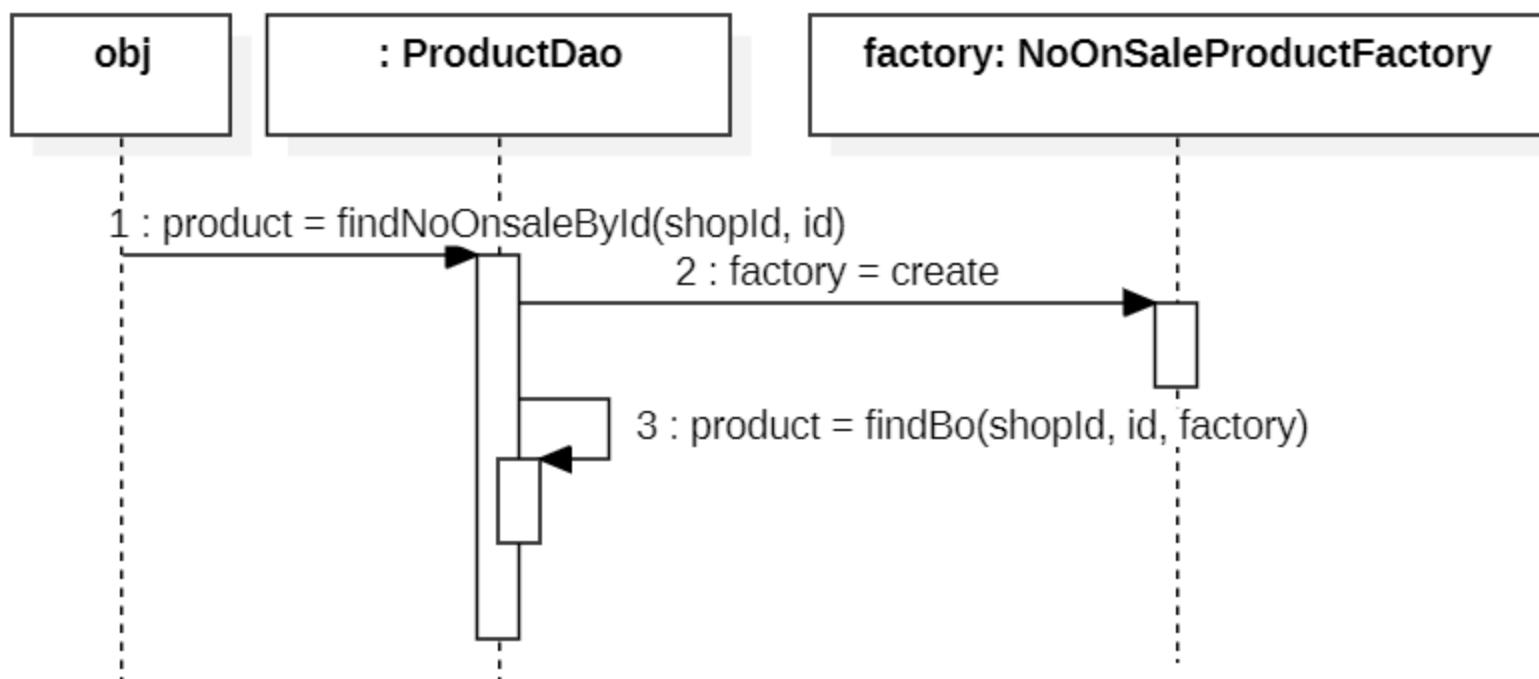
## 2.1 顾客查看商品详细信息



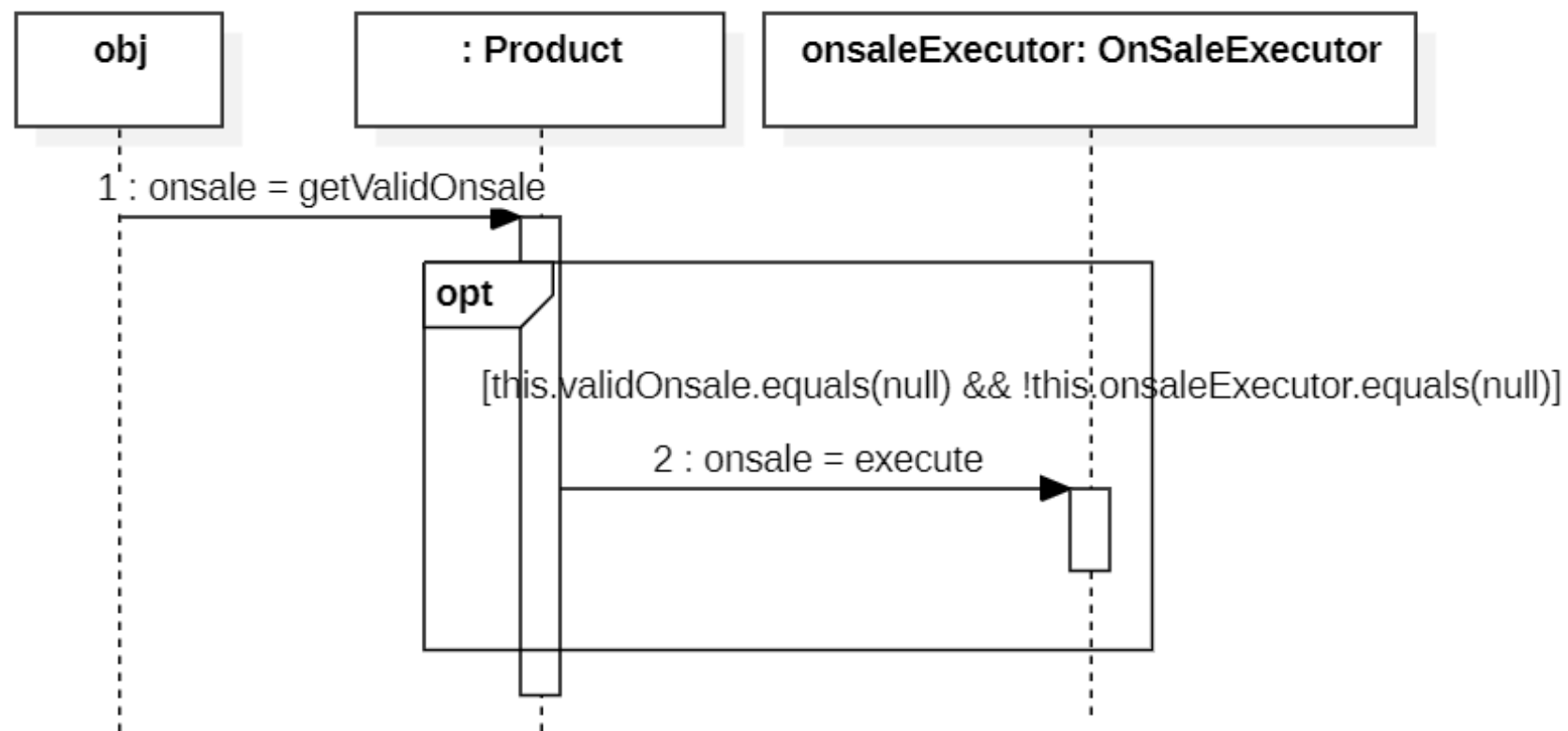
## 2.1 顾客查看商品详细信息



## 2.1 顾客查看商品详细信息

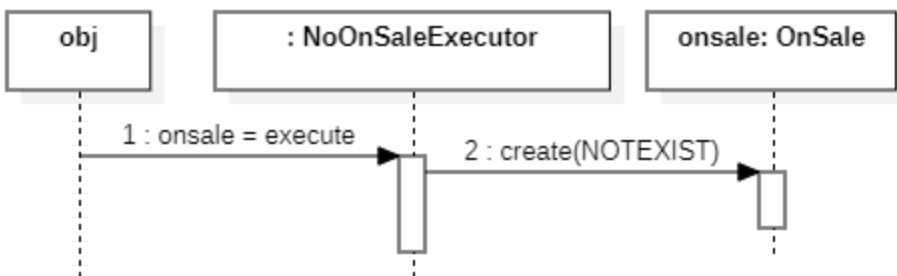
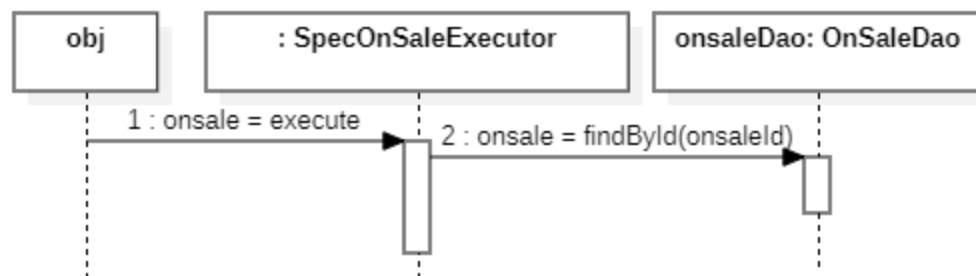
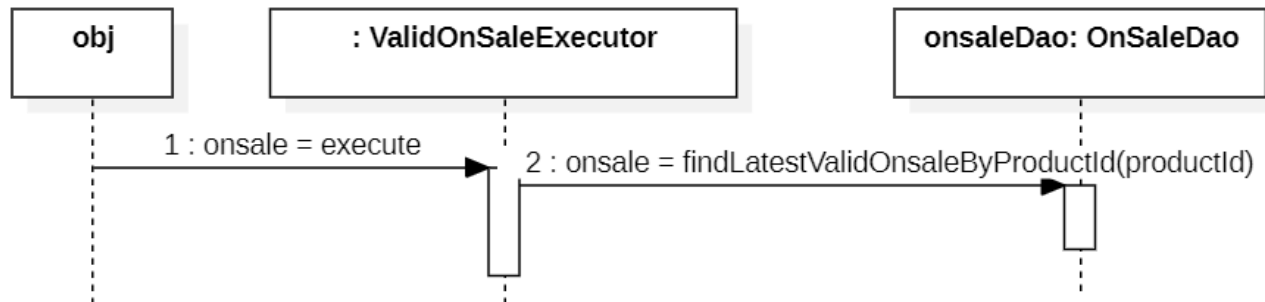


## 2.1 顾客查看商品详细信息





## 2.1 顾客查看商品详细信息



## 2.2 新增团购活动

**POST** /shops/{shopId}/products/{pid}/groupons 管理员新增团购活动

- 团购与优惠活动不并存 出216错误

Parameters Try it out

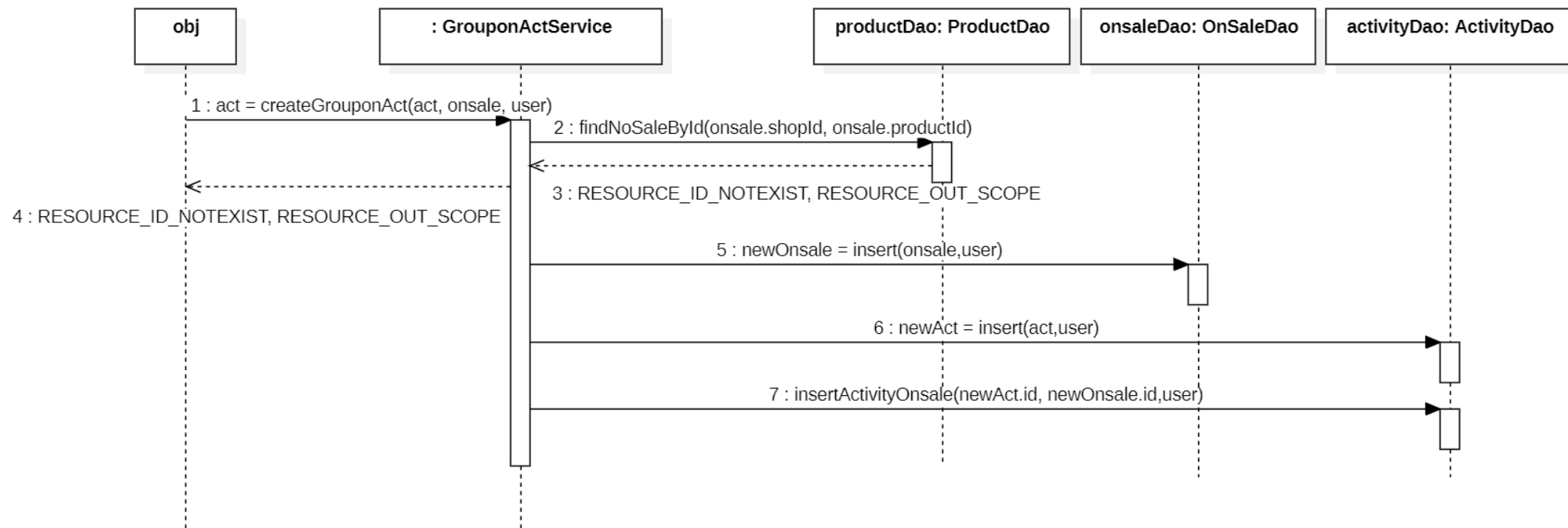
Name	Description
<b>authorization</b> * required string (header)	用户token <input type="text" value="authorization"/>
<b>shopId</b> * required integer (path)	店铺id <input type="text" value="shopId"/>
<b>pid</b> * required integer (path)	产品id <input type="text" value="pid"/>
<b>body</b> * required (body)	可修改的信息 Example Value   Model <pre>{   "name": "string",   "thresholds": [     {       "quantity": 0,       "percentage": 0     }   ],   "price": 0,   "beginTime": "string",   "endTime": "string",   "quantity": 0,   "maxQuantity": 0 }</pre> <div>Parameter content type <input type="text" value="application/json"/></div>

Responses Response content type application/json

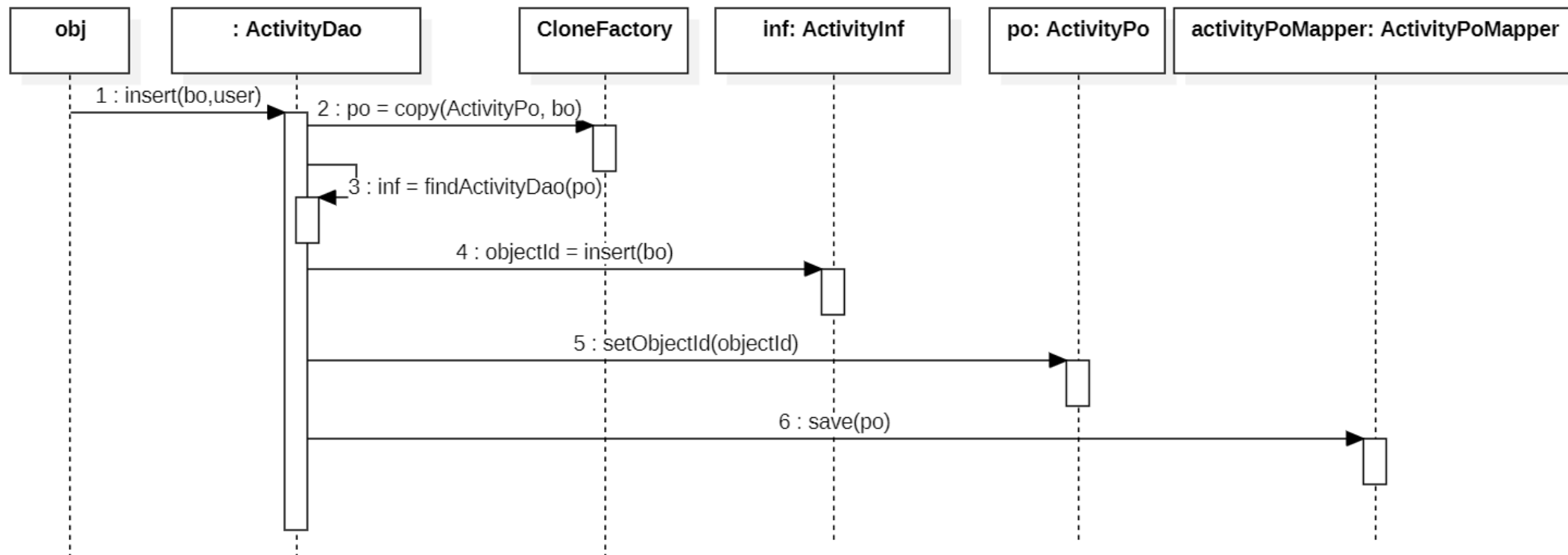
Code	Description
default	成功 Example Value   Model <pre>{   "errno": 0,   "errmsg": "成功",   "data": {     "id": 0,     "name": "string"   } }</pre>



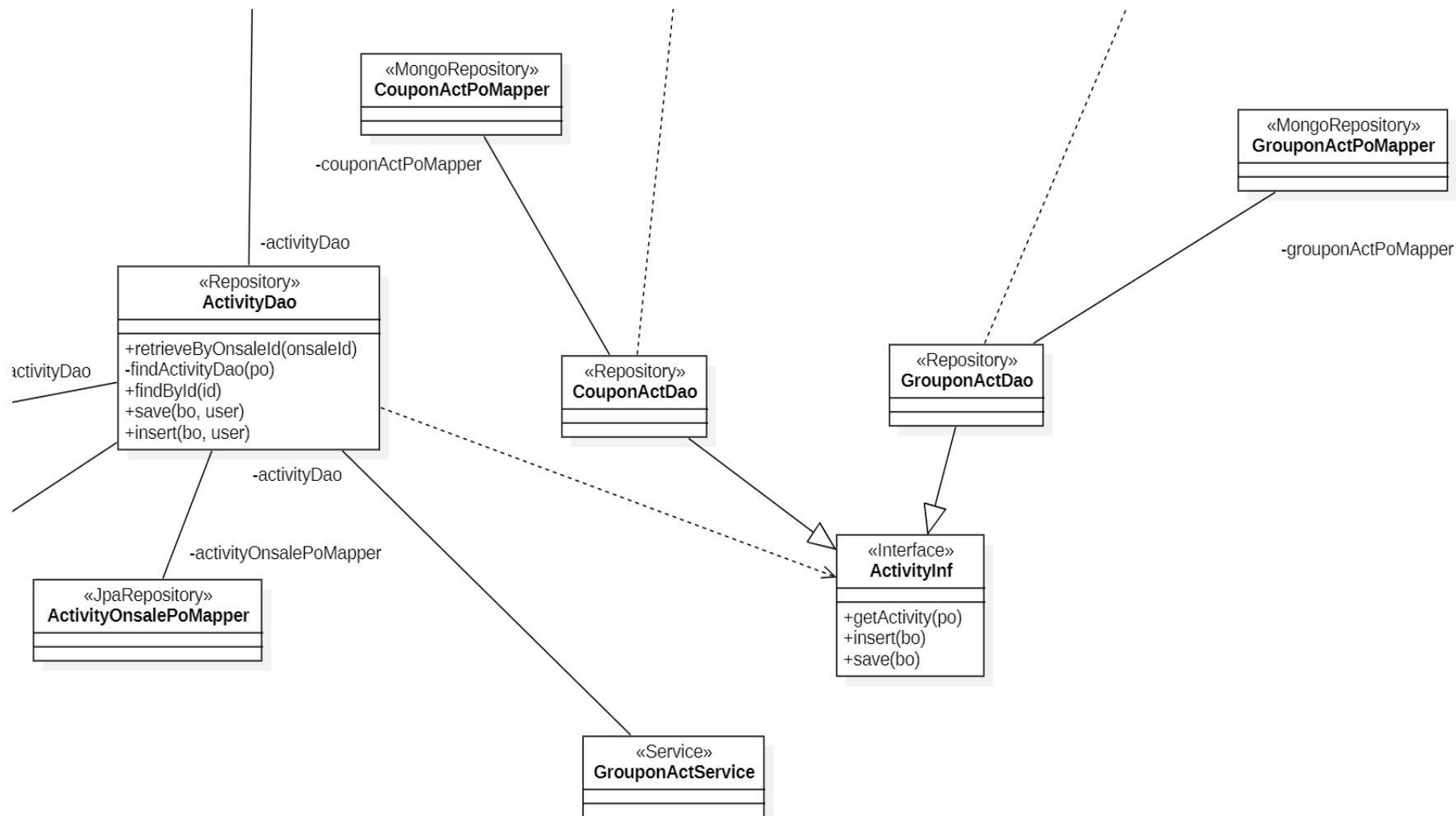
## 2.2 新增团购活动



## 2.2 新增团购活动



## 2.2 新增团购活动



## 2.3 取消团购活动

**DELETE** /shops/{shopId}/products/{pid}/groupons/{id} 将产品上的团购活动取消

Parameters

**authorization** \* required

string  
(header)

用户token

authorization

**shopId** \* required

integer(\$int64)  
(path)

商店ID

shopId

**id** \* required

integer(\$int64)  
(path)

活动ID

id

**pid** \* required

integer(\$int64)  
(path)

产品ID

pid

Responses

Response content type application/json

**Code**

**Description**

default

成功

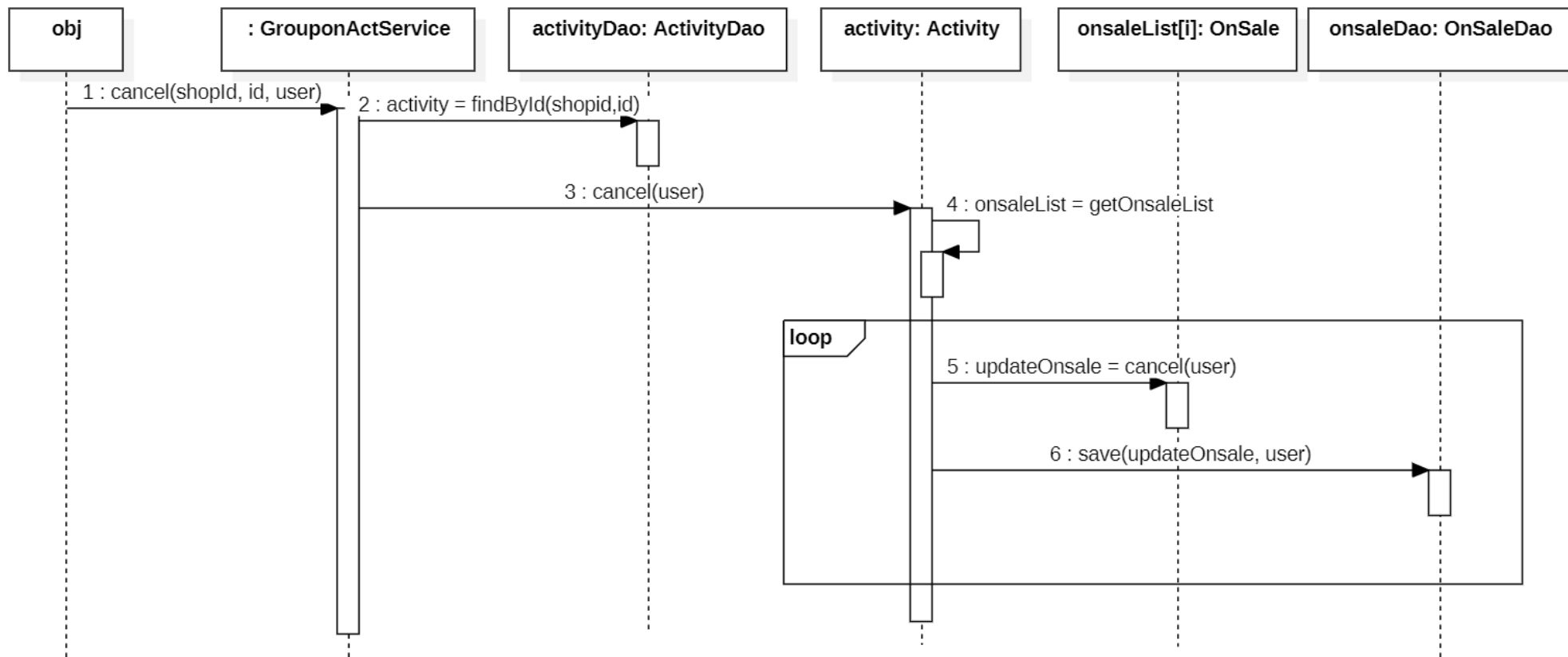
Example Value

Model

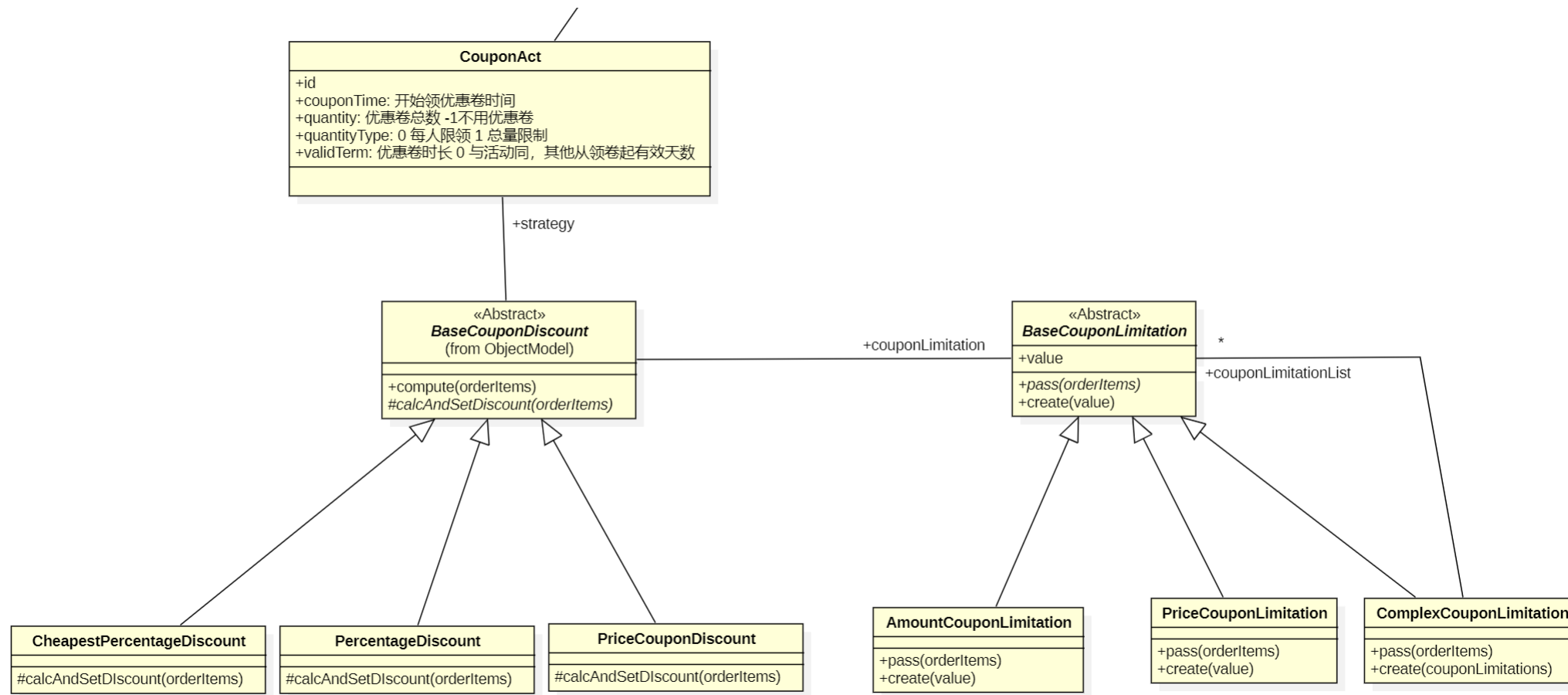
```
{  "errno": 0,  "errmsg": "成功"}
```



## 2.3 取消团购活动



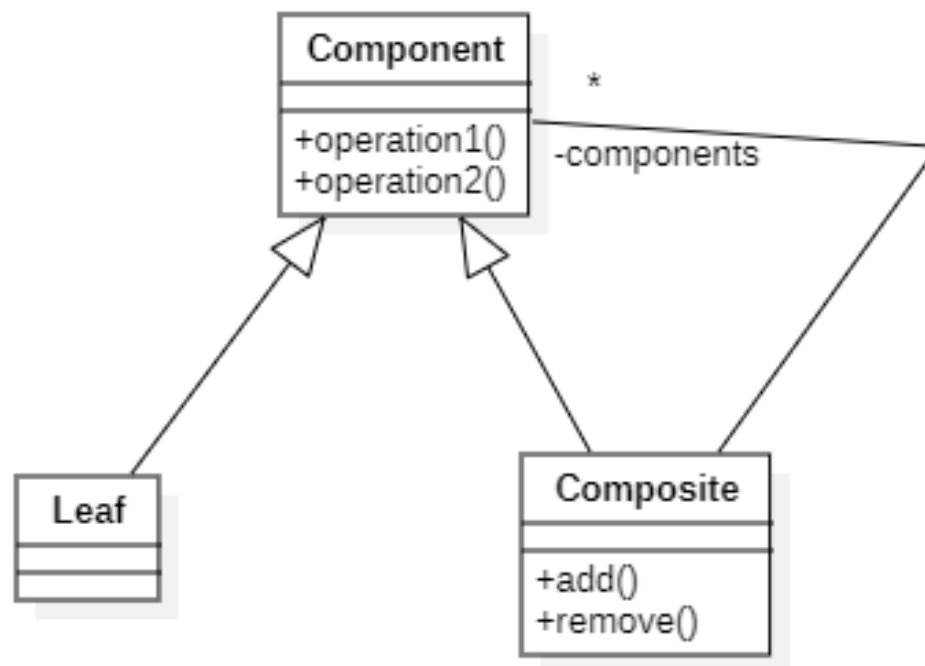
## 2.4 计算优惠





## 2.4 计算优惠

- 组合设计模式(Composite)
  - 将对象组合成树形结构，使得调用者对单个对象和组合对象的使用具有一致性。



## 2.4 计算优惠

