**进销存系统详细设计描述文档**

版本号 ：V1.0

WFI

**摘要**

南京大学软件学院软件工程

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 吴骁凯 | 2014-11-4 | 最初草稿 | V0.1 |
| 吴骁凯 | 2014-11-4 | 评审后的正式版 | V0.11 正式版 |
| 韩淳 | 2014-11-5 | 增加模块中的Controller | V0.2 正式版 |
| 韩淳 | 2014-12-25 | 增加状态图，修改一部分顺序图 | V1.0 正式版 |
| 韩淳 | 2014-12-30 | 增加数据层的设计 | V1.1正式版 |

目录

更新历史 2

1.引言 5

1.1编制目的 6

1.2词汇表 6

1.3参考资料 6

2.产品概述 6

3.体系结构设计概述 7

4.结构视角 7

4.1 逻辑层结构视角 7

4.1.1 CommodityBl模块 7

（1）模块概述 7

（2）整体结构 7

（3）模块内部类的接口规范 8

4.1.2 KindBl模块 17

（1）模块概述 17

（2）整体结构 17

（3）模块内部类的接口规范 18

4.1.3 userbl 23

（1）模块概述 23

（2）整体结构 23

（3）模块内部类的接口规范 24

（4）业务逻辑层动态模型 25

4.1.4 memberBL模块 25

（1）模块概述 25

（2）整体结构 25

（3）模块内部接口规范 26

（4）业务逻辑层的动态模型 30

（5）业务逻辑层的设计原理 33

4.1.5MarketBL模块 33

（1）模块概述 33

（2）整体结构 34

（3）模块内部的接口规范 35

（4） 业务逻辑层的动态模型 57

（5）业务逻辑层的设计原理 75

4.1.6Accountbl模块 75

（1）模块概述 75

（2）整体结构 76

（3）模块内接口规范 77

（4）业务逻辑层动态模型 79

（5）业务逻辑层的设计原理 80

4.1.7checkbl模块 80

（1）模块概述 80

（2）整体结构 80

（3）模块内接口规范 81

（4）业务逻辑层动态模型 85

（5）业务逻辑层的设计原理 86

4.1.8paymentbl模块 86

（1）模块概述 86

（2）整体结构 86

（3）模块内部类的接口规范 87

（4）业务逻辑层动态模型 96

（5）业务逻辑层的设计原理 98

4.1.9.billmanagementbl模块 98

（1）模块概述 98

（2）整体结构 99

（3）模块内部类的接口规范 100

（4）业务逻辑层的动态模型 103

（5）业务逻辑层的设计原理 104

4.1.10.promotionbl模块 105

4.2数据层层结构视角 137

4.2.1 KindData模块 137

（1）模块概述 137

(2)整体结构 137

(3)模块内部类接口规范 137

4.2.2 CommodityData模块 138

（1）模块概述 138

（2）整体结构 138

（3）模块内部接口规范 139

4.2.3 memberData模块 143

（1）模块概述 143

（2）整体结构 143

（3）模块内部接口规范 144

（4）业务逻辑层的动态模型 148

（5）业务逻辑层的设计原理 148

4.2.4MarketData模块 148

（1）模块概述 148

（2）整体结构 148

（3）模块内部的接口规范 149

（4） 业务逻辑层的动态模型 153

（5）业务逻辑层的设计原理 154

4.2.5.paymentdata模块 154

4.2.6.accountdata模块 154

4.2.7.promotiondata模块 154

（1）模块概述 154

（2）整体结构 154

（3）模块内部类的接口规范 156

（4） 业务逻辑层的动态模型 170

（5）业务逻辑层的设计原理 170

5.依赖视角 171

# 1.引言

## 1.1编制目的

本报告详细完成对进销存系统的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| IS | 进销存系统 | Invoicing System |
|  |  |  |
|  |  |  |

## 1.3参考资料

进销存系统用例文档

进销存系统软件需求规格说明文档

进销存系统体系结构描述文档

# 2.产品概述

参考进销存系统用例文档和进销存系统软件需求规格说明文档对产品的概括描述。

# 3.体系结构设计概述

参考进销存系统体系结构描述文档中对体结构设计的概述。

# 4.结构视角

# 4.1 逻辑层结构视角

## 4.1.1 CommodityBl模块

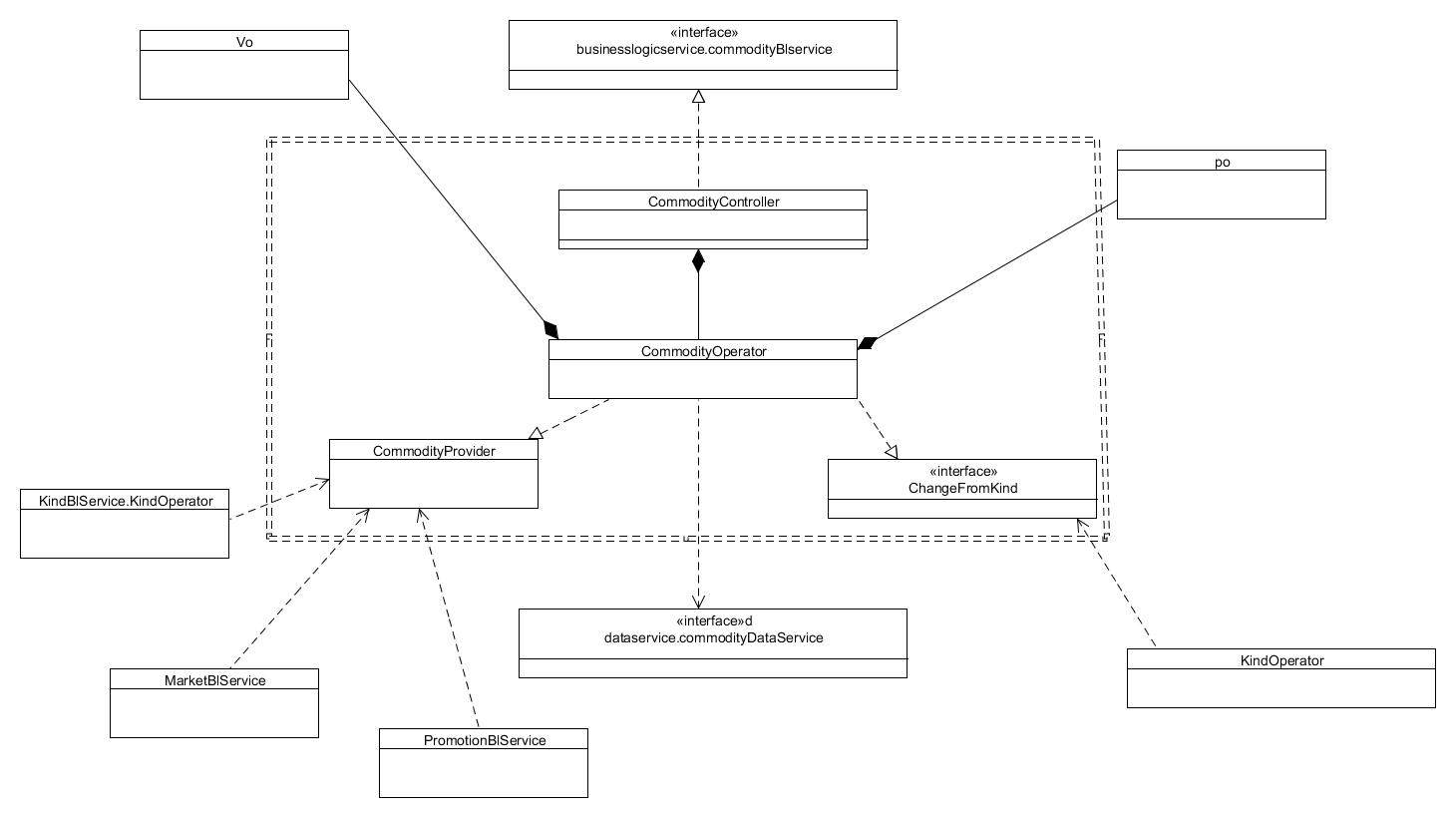
### （1）模块概述

CommodityBl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

CommodityBl模块的职责及接口参见软件体系设计文档。

### （2）整体结构

CommodityBl模块的设计如图



CommodityBl模块各个类职责如表所示

|  |  |
| --- | --- |
| 模块 | 职责 |
| CommodityController | 控制界面层对CommodityOperator的访问 |
| CommodityOperatior | 负责对库存商品的增加，删除，查找和修改。 |

### （3）模块内部类的接口规范

CommodityOperator类接口

|  |  |  |
| --- | --- | --- |
| CommodityOperator.addCommodity | 语法 | Public int addCommodity(CommodityVo comdVo); |
| 前置条件 | 用户正确选择了商品分类，并输入了商品名称、型号、库存数量、进价、零售价、最近进价、最近零售价等信息。 |
| 后置条件 | 返回是否添加成功 |
| CommodityOperator.delCommodity | 语法 | Public int delCommodity(String name); |
| 前置条件 | 用户正确输入了要删除产品的名称和编号。 |
| 后置条件 | 返回1，删除成功，返回0删除失败（不存在该商品） |
| CommodityOperator.changeCommodity | 语法 | Public int changeCommodityInfo(CommodityVo vo); |
| 前置条件 | 用户要改动的商品已经找到 |
| 后置条件 | 返回商品信息修改是否成功 |
| CommodityOperator.findCommodity | 语法 | Public CommodityVo findCommodiy(String name); |
| 前置条件 | 用户输入要查找的商品名称 |
| 后置条件 | 返回匹配的商品对象。 |
| CommodityOperator.findCommodity | 语法 | Public CommodityVo findCommodiy(String num); |
| 前置条件 | 用户输入要查找的编号 |
| 后置条件 | 返回匹配的商品对象。 |
| CommodityOperator.search | 语法 | Public ArrayList search(SearchType type,String info); |
| CommodityOperator.getCommodity() | 语法 | Public ArrayList<CommodityVo>  getCommoditys(String name) |
| 前置条件 | 需要根据名字获得商品 |
| 后置条件 | 系统返回匹配的商品项目 |
| CommodityOperator.getAllCommoditys | 语法 | Public ArrayList<CommodityVo>  getAllCommoditys() |
| 前置条件 | 需要获得所有库存商品信息 |
| **后置条件** | **系统返回所有的库存商品信息项目** |
| CommodityOperator**.getCommodity()** | **语法** | **Public CommodityVo getCommodity(String kindNum);** |
| **前置条件** | **系统需要根据商品类别编号获取商品项目** |
| **后置条件** | **系统返回匹配的商品项目** |
| CommodityOperator**.getCommoditys\_ByKindName** | **语法** | **Public ArrayList<CommodityVo> getCommoditys\_ByKindName(String kindName);** |
| **前置条件** | **系统需要获得某一分类的所有商品** |
| **后置条件** | **返回同属kindName的所有商品项目** |
| CommodityOperator**.getOneCommoditys** | **语法** | **Public CommodityVo getOneCommodity(int loc);** |
| **前置条件** | **系统需要获得在商品列表中的第loc个位置的商品信息** |
| **后置条件** | **返回处在第loc个位置的商品项目** |
| CommodityOperator**.getBills** | **语法** | **Public ArrayList<BillPo> getBills(infoVo vo)** |
| **前置条件** | **系统需要获得符合某些信息的单据** |
| **后置条件** | **返回符合标准的单据** |
| CommodityOperator**.addPkg** | **语法** | **Public int addPkg(PkgVO vo);** |
| **前置条件** | **系统需要添加一个新的特价包** |
| **后置条件** | **返回更新是否成功** |
| CommodityOperator**.delPkg** | **语法** | **Public int delPkg（PkgVO vo）;** |
| **前置条件** | **系统需要删除一个特价包** |
| **后置条件** | **返回是否删除成功** |
| CommodityOperator**.delPkg(String id);** | **语法** | **public int delPkg(String id)** |
| **前置条件** | **获得Pkg的Id** |
| **后置条件** |  |
| CommodityOperator**.changePkg** | **语法** | **Public int changePkg(PkgVO VO);** |
| **前置条件** | **系统需要更改一个特价包** |
| **后置条件** | **返回是否更改成功** |
| CommodityOperator**.getAllPkgs** | **语法** | **Public ArrayList<PkgVO>**  **getAllPkgs()** |
| **前置条件** | **系统需要所有特价包的信息** |
| **后置条件** | **返回所有特价包的信息** |
| CommodityOperator**.getPkg** | **语法** | **Public PkgVO getPkg(String id)** |
| **前置条件** | **已知pkg的id** |
| **后置条件** | **返回符合id的pkg** |
| CommodityOperator**.findPkg** | **语法** | **Public int findPkg(PkgVo vo);** |
| **前置条件** | **系统需要确定某一个特价包在特价包列表中的位置** |
| **后置条件** | **返回符合标准的特价包位置** |
| CommodityOperator**.changeStock** | **语法** | **public String changeStock(String kindNum, int number, double inPrice,**  **double outPrice)** |
| **前置条件** | **系统对库存数量，最近价格，进行了修改** |
| **后置条件** | **返回是否修改成功** |
| CommodityOperator**.MarketFind** | **语法** | **public** MarketCommodityVO MarketFind(String numberOfkind) |
| **前置条件** | **系统需要根据商品编号获得商品信息** |
| **后置条件** | **返回所需的商品信息** |
| CommodityOperator**. exportAsXls** | **语法** | **exportAsXls(JTable table)** |
| **前置条件** | **系统需要导出excel文档** |
| **后置条件** | **在客户端的exl文档中生成excel文档** |
| CommodityOperator**.updateCommodity\_Kind** | **语法** | **Public void updateCommodity\_Kind(String oldName,String newName)** |
| **前置条件** | **已知旧商品类别名和新的商品类别名** |
| **后置条件** | **所有属于旧商品类别的商品，其类别都更新为newName** |
| CommodityOperator**.**examineStock | **语法** | **public Ex\_StockVo examineStock(double beginTime, double endTime)** |
| **前置条件** | **已经获得了一个合理的时间区间** |
| **后置条件** | **返回这个时间段内的出入库数量和金额合计** |
| CommodityOperator**.** subStockBills | **语法** | **public int subStockBills(StockBillsVo vo)** |
| **前置条件** | **获得了完整的单据Vo** |
| **后置条件** | **提交到总经理处等待审批** |
| CommodityOperator**.**checkAlarm | **语法** | **public ArrayList<StockBillsVo> checkAlarm()** |
| **前置条件** | **无** |
| **后置条件** | **在系统启动时检查所有数量小于报警数量的商品项目，返回为ArrayList** |
| CommodityOperator**.**init | **语法** | **public** **void** init() |
| **前置条件** | **无** |
| **后置条件** | **清空库存数据** |
| CommodityOperator**.**initKind | **语法** | **public** **void** initKind() |
| **前置条件** | **无** |
| **后置条件** | **清空分类数据** |

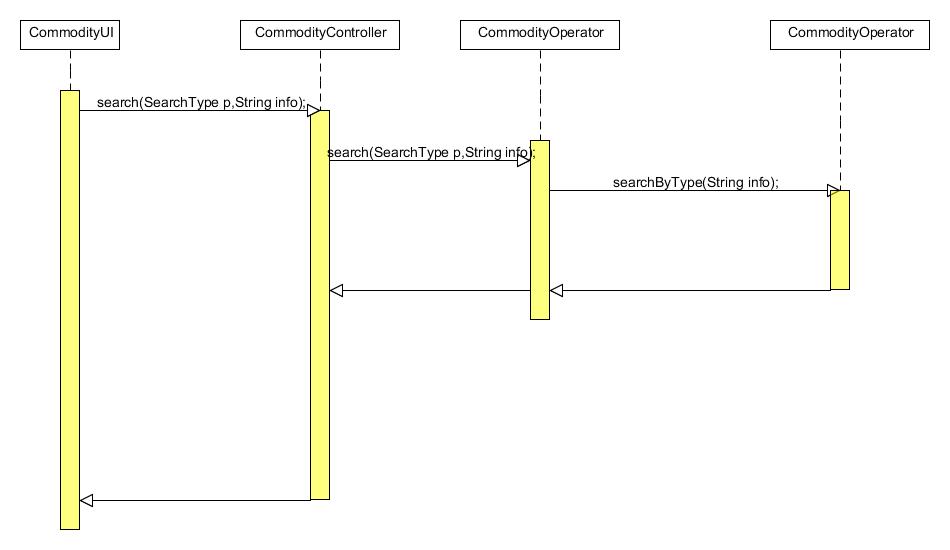
**需要的服务（需接口）**

|  |  |
| --- | --- |
| **Kindbl.get** | **提供分类树某一节点的所有子类** |
| **KindBl.init** | **清空商品分类树** |
| **KindBL**.addCommodityToKind | **向商品分类树中加入一个商品** |
| **KindBL.delCommodityToKind** | **在商品分类树中删除一个商品** |
| **KindBl.**changeCommodityName | **更改分类树中某一商品的名称** |
| **billManagementbl.sendBill** | **将单据发送给单据管理** |
| **CommodityDataService.addCommodity(CommodityPO po)** | **增加1个持久化对象** |
| **CommodityDataService.delCommodity(String name)** | **删除一个持久化对象** |
| **CommodityDataService.delCommodity(int loc);** | **删除一个持久化对象** |
| **CommodityDataService.getOneCommodity** | **获得一个商品PO** |
| **CommodityDataService.** updateCommodity\_Kind | **更新若干商品的分类** |
| **CommodityDataService.findCommodity** | **确定某个商品PO在商品列表中的位置** |
| **CommodityDataService.getCommoditys** | **获得某一商品名称的商品项目** |
| **CommodityDataService.getAllCommodity();** | **获取所有商品PO** |
| **CommodityDataService.getCommodity** | **获取某一商品编号的商品PO** |
| **CommodityDataService.** getCommoditys\_ByKindName | **获得属于某一种类的所有商品ＰＯ** |
| **CommodityDataService.** examineStock | **用于库存查看** |
| **CommodityDataService.** addBill | **向底层数据中添加一个单据** |
| CommodityDataService.ｇｅｔＡｌａｒｍ | **获得所有库存数量少于报警数量的商品PO** |
| **CommodityDataService．getBills** | **获得符合INfoVo标准的单据** |
| **CommodityDataService.addPkg** | **向底层数据中加入特价包** |
| **CommodityDataService.delPkg** | **在底层数据中删除某一个提价包** |
| **CommodityDataService.ChangePkg** |  |
| **CommodityDataService.getAllPkgs** | **获得所有的特价包数据** |
| **CommodityDataService.findPkg** | **确定某一个Pkg在特价包列表中的位置** |
| **CommodityDataService** .**changeStock** | **更改库存数据** |
| **CommodityDataService.** MarketFind | **获得所有属于某一编号的MarketFindVo** |
| **CommodityDataService.**delCommodityByKindName | **根据某一类别的名称删除商品** |
| **CommodityDataService** .delCommodityByName | **根据某一商品名称删除商品** |
| **CommodityDataService.Init** | **清空库存数据** |

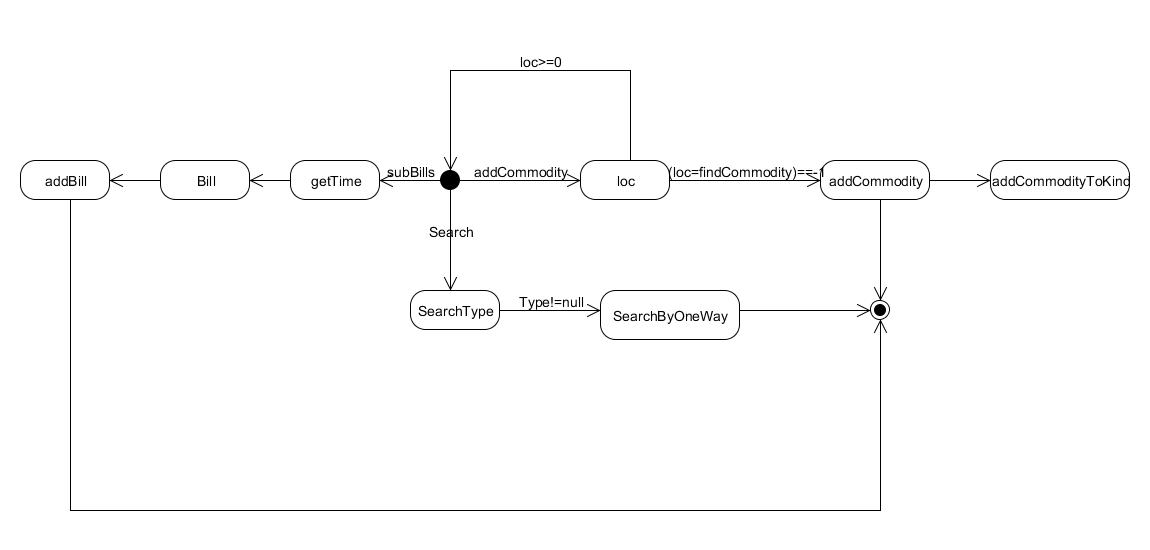
（4）业务逻辑层动态模型



添加商品的顺序图



查找商品顺序图



CommodityOperator典型操作顺序图

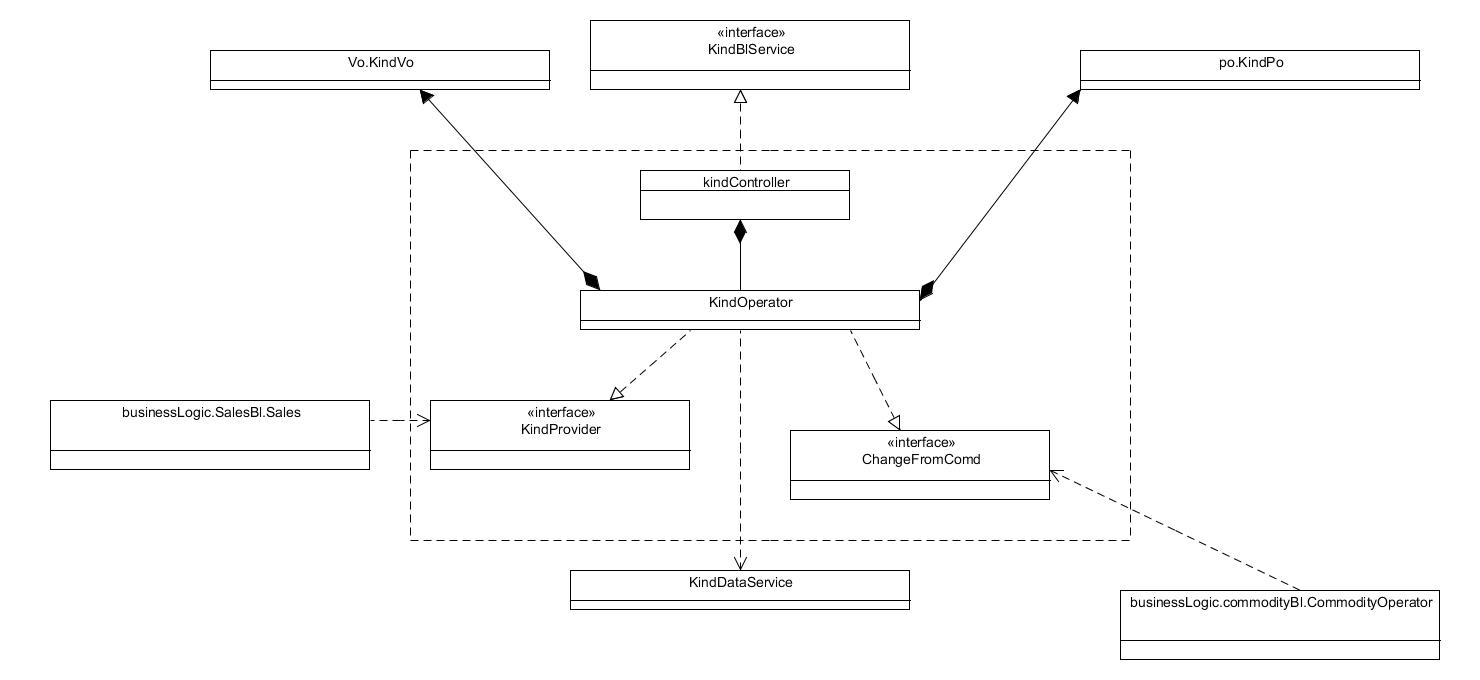
## 4.1.2 KindBl模块

### （1）模块概述

KindBl模块承担的需求参见需求规格说明文档

### （2）整体结构

KindBl模块的设计如图



KindBl模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| KindOperator | 负责对分类树的添加，删除，修改，获取子类等实际操作 |
| KindController | 控制界面层对逻辑层的访问 |

### （3）模块内部类的接口规范

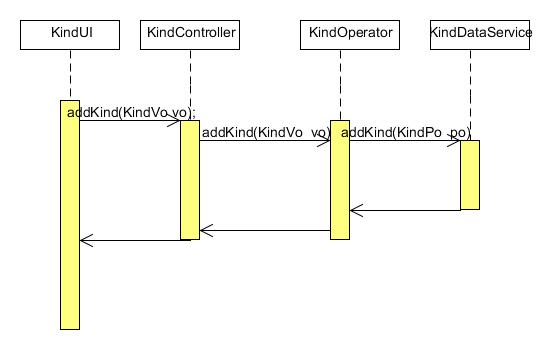
KindOperator的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| KindOperator.addkind | **语法** | **Public int addKind(KindVo vo)；** |
| **前置条件** | **用户选择了正确的分类树节点** |
| **后置条件** | **系统将新的类别加入到分类树中** |
| KindOperator.deleteKind | **语法** | **Public int deleteKind(KindVo vo);** |
| **前置条件** | **用户指定树状结构中某一分类或输入某一分类的名称** |
| **后置条件** | **系统将该分类从树状结构中删除并删除其下所有子类,并返回是否删除成功。** |
| KindOperator.changeKind | **语法** | **public int changeKind（KindVo vo);** |
| **前置条件** | **用户选择某一分类并输入了该分类的新名称** |
| **后置条件** | **系统对树状图中的元素进行更新** |
| **KindOperator.getSons** | **语法** | **Public ArrayList<KindVo> getKinds（KindVo vo);** |
| **前置条件** | **用户点击提供某一个分类的名称。** |
| **后置条件** | **返回与该名称对应的分类的所有子类** |
| **KindOperator.get\_root** | **语法** | **Public KindPo get\_root();** |
| **前置条件** | **界面层需要根节点** |
| **后置条件** | **返回根节点po** |
| **KindOperator.init;** | **语法** | **Public void init();** |
| **前置条件** | **发动清空数据** |
| **后置条件** | **底层数据被全部清空** |
| **KindOperator.delCommodityByKindName** | **语法** | **Public void delCommodityByKindName(String name)** |
| **前置条件** | **管理员要删除某一类别的所有商品** |
| **后置条件** | **库存信息中所有属于该商品的项目都被删除** |
| **KindOperator.delCommodityByName** | **语法** | **Public void delCommodityByName(String name);** |
| **前置条件** | **管理员要在商品分类管理的界面层删除某一项商品** |
| **后置条件** | **库存信息中该商品被删除。** |
| **KindOperator.changeComdKindName** | **语法** | **Public void changeComdKindName(String oldName,String newName);** |
| **前置条件** | **管理员在商品分类管理中修改某一分类的名称。** |
| **后置条件** | **库存信息中所有属于oldName类别的商品都将类别名称更新为newName** |
| **KindOperator.getWholeTree** | **语法** | **Public KindVo getWholeTree();** |
| **前置条件** | **界面层需要整棵分类树** |
| **后置条件** | **返回分类树的根节点。** |
| **KindOperator.getfather** | **语法** | **Public kindVo getFather()** |
| **前置条件** | **需要获得某一商品或分类的父类。** |
| **后置条件** | **返回父类。** |
| **KindOperator.getSons** | **语法** | **Public KindInfoVO getSons(String name);** |
| **前置条件** | **需要对某一分类的子类进行分类获得，子类中为商品的放在一个ArrayList中，子类中的商品类别放在另一ArrayList中** |
| **后置条件** | **返回含有两个ArrayList的KindInfoVo** |
| **KindOperator.initCommodty** | **语法** | **Public void initCommodity()** |
| **前置条件** | **界面层要求清空数据** |
| **后置条件** | **清空所有库存数据** |
| **KindOperator.addCommodityTokind；** | **语法** | **Public int addCommodityToKind(String fatherName,String kindnUm,String comdName)；** |
| **语法** | **Public void changeComdKindName(String oldName,String newName);** |
| **前置条件** | **管理员在商品分类管理中修改某一分类的名称。** |
| **kindOperator.addCommodityTokind；** | **语法** | **Public int addCommodityToKind(String fatherName,String kindnUm,String comdName)；** |
| **前置条件** | **库存管理界面增加了新商品** |
| **后置条件** | **在分类树中添加商品** |
| **kindOperator.delCommodityTokind** | **语法** | **Public void delCommoditytoKind(String name);** |
| **前置条件** | **库存管理界面删除了一项商品** |
| **后置条件** | **分类树中删除该项商品** |

**需要的服务（需接口）**

|  |  |
| --- | --- |
| **KindDataService.read\_root** | **读取根类的po** |
| **KindDataService.write\_root** | **写入根类的po** |
| **ChangeFromKind.init;** | **清空库存数据** |
| **ChangeFromKind.changeComdName** | **修改某一商品的名称** |
| **ChangeFromKind.delCommodityByKindName** | **删除某一分类的所有商品** |
| **ChangeFromKind.delCommodityByName** | **删除某一商品** |

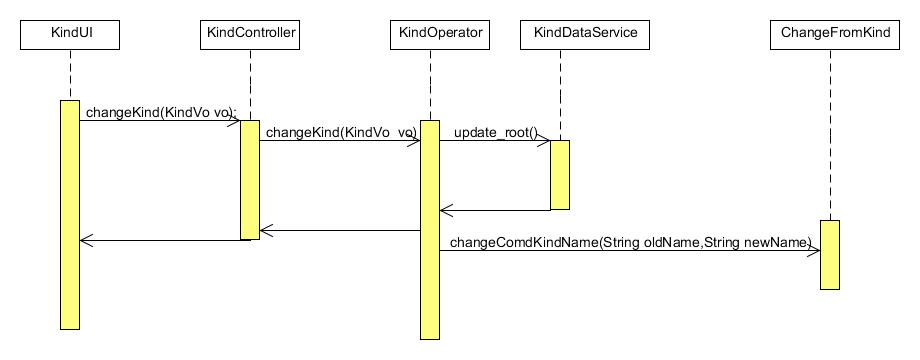
（4）业务逻辑层的动态模型



添加分类的顺序图

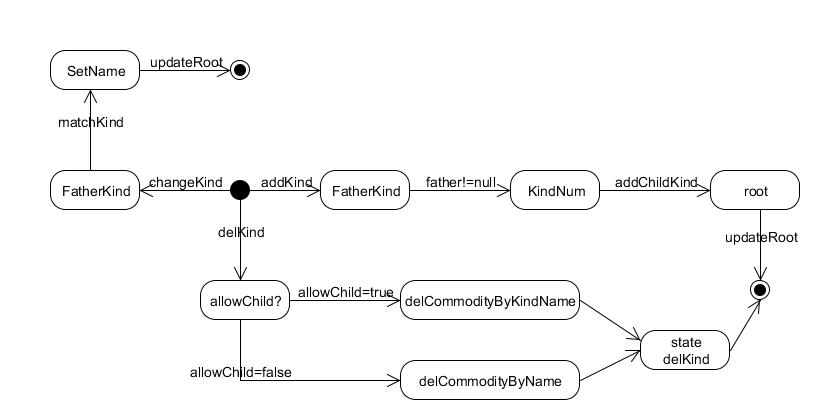


删除分类的顺序图



修改分类的顺序图

kind



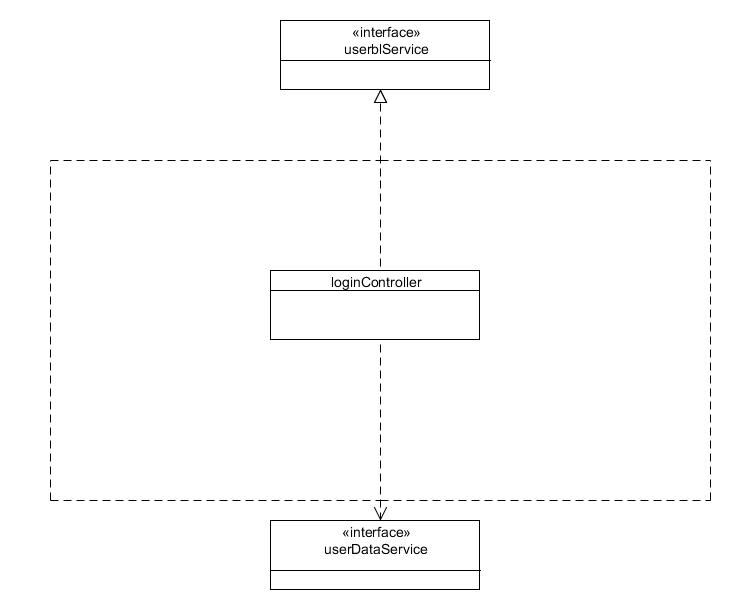
kindoperator状态图

## 4.1.3 userbl

### （1）模块概述

Userbl承担的需求详见需求规格说明文档。

### （2）整体结构



Userbl模块各个类的职责

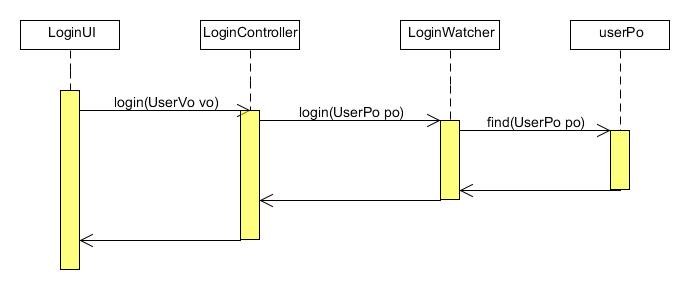
|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginContoller | 控制所有人员的登录行为 |

### （3）模块内部类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| UserBl.loginController.login | 语法 | Public boolean login(UserVo vo); |
| 前置条件 | 用户输入了用户名和密码 |
| 后置条件 | 返回是否登录成功 |

|  |  |
| --- | --- |
| 需要的服务 | |
| UserDataService.loginWatcher.login | 验证登录信息和低层数据信息是否匹配 |

### （4）业务逻辑层动态模型



## 4.1.4 memberBL模块

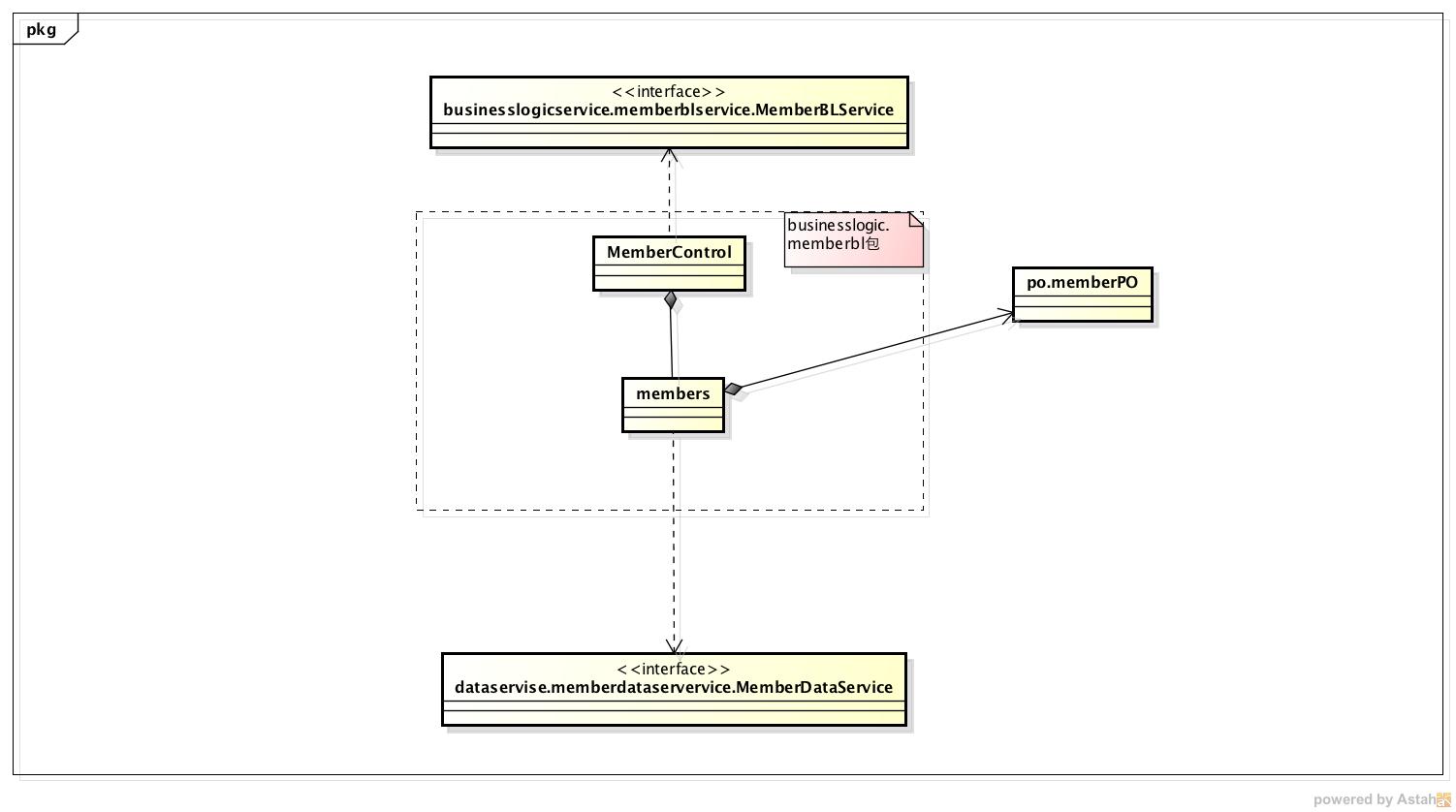
### （1）模块概述

memberBL模块承担了需求参见需求规格说明文档的功能需求以及非功能需求。

### （2）整体结构

根据体系结构的设计，我们将系统分为展示层，逻辑层，数据层。每层之间有接口。为了隔离业务逻辑职责和业务控制职责，我们添加了MemberControl，这样就将客户管理的业务逻辑委托给Member对象。MemberPO是作为客户的持久化对象被储存的。

memberBL模块设计：



MemberBL模块各个类的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| MemberControl | 负责实现客户管理界面所需要的服务 |
| Member | 客户管理的领域模型对象，拥有一次客户管理所持有的客户信息，可以帮助完成客户管理界面所需服务 |

### （3）模块内部接口规范

MemberControl的接口规范：

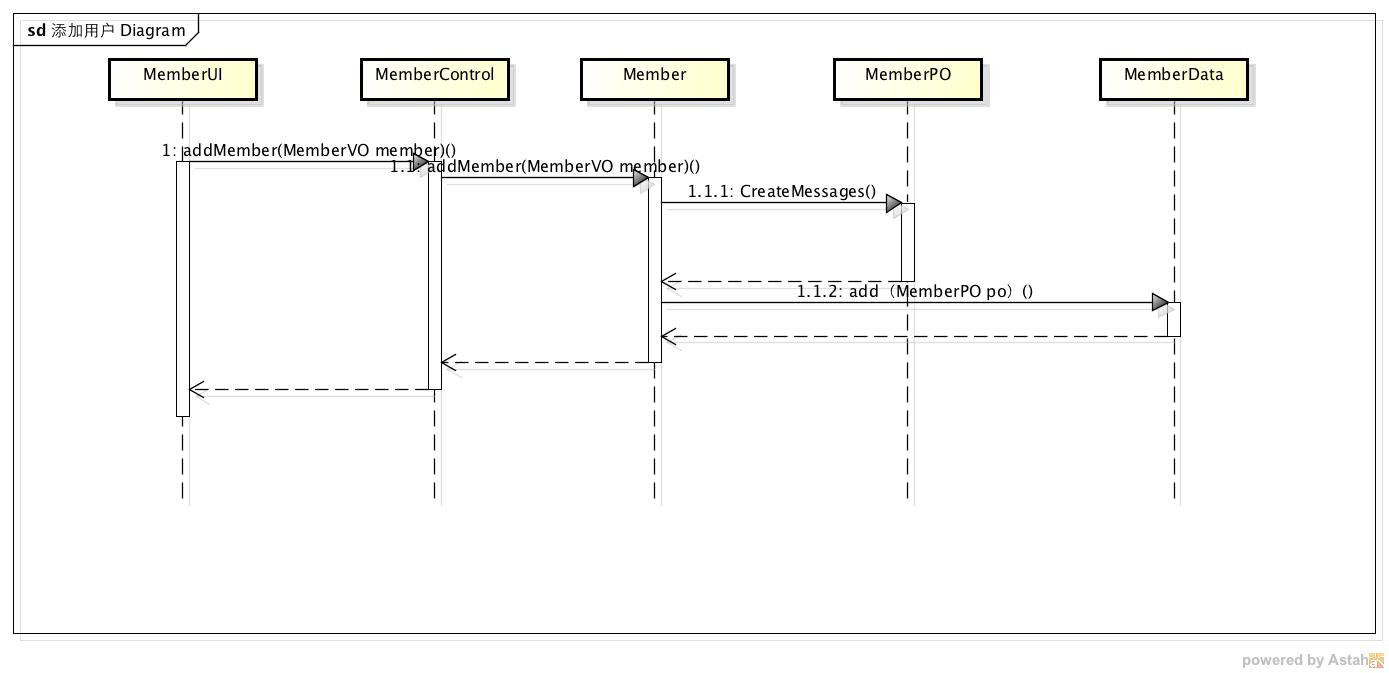
|  |  |  |
| --- | --- | --- |
| **提供的服务（供借口）** | | |
| **memberControl.addmember** | **语法** | **public String addmember(MemberVO member)** |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberControl.delmember** | **语法** | **public String delmember(MemberPO member)** |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberControl.updmember** | **语法** | **public String updmember(MemberPO member)** |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberControl.findmember** | **语法** | **public String findmember(MemberPO member)** |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **需要的服务（需接口）** | | |
| **Member.addmember(memberVO)** | **得到符合客户信息** | |
| **Member.del(memberPO po)** | **删除一个客户** | |
| **Member.add(memberPO po)** | **新增一个客户** | |
| **Member.upd(memberPO po)** | **更新一个客户** | |

Member的借口规范

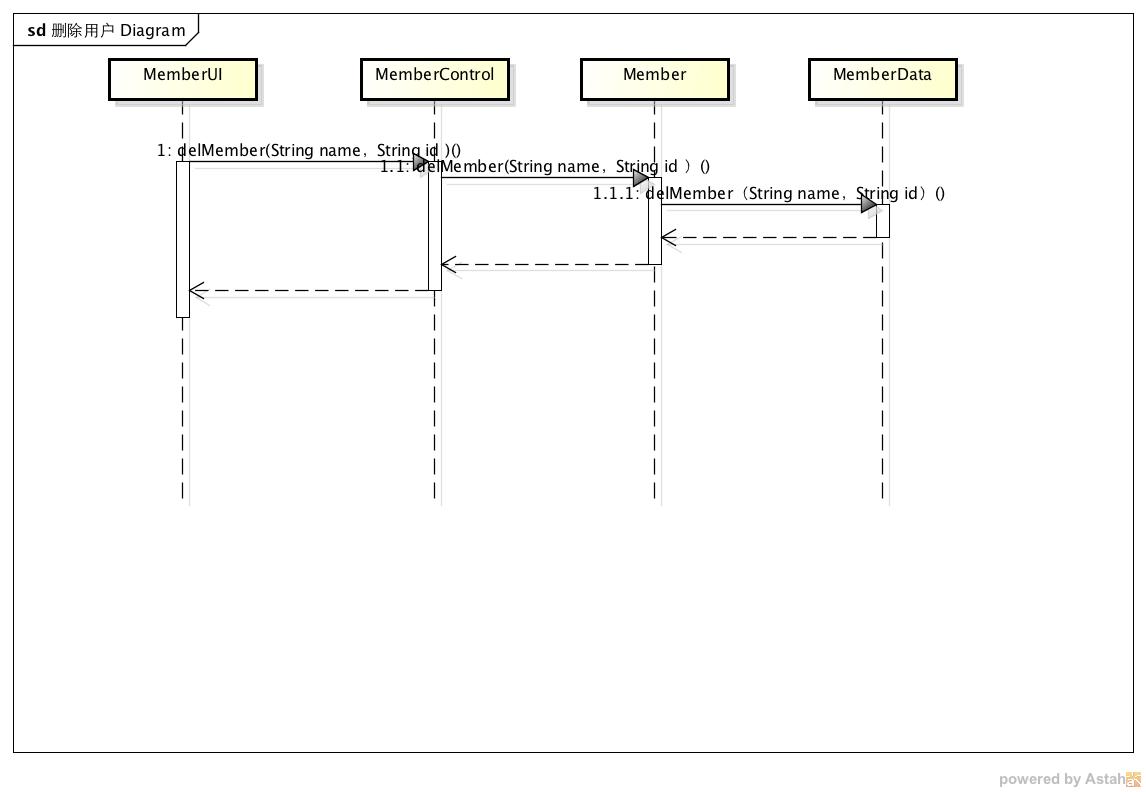
|  |  |  |
| --- | --- | --- |
| **提供的服务（供借口）** | | |
| **memberbl.addmember** | **语法** | **public String addmember(MemberPO member)** |
| **前置条件** | **PO包含所有信息** |
| **后置条件** | **根据PO查找是否存在该客户，没有则新增用户** |
| **memberbl.delmember** | **语法** | **public String delmember(MemberPO member)** |
| **前置条件** | **PO包含id与姓名** |
| **后置条件** | **根据PO查找是否存在该客户，有则删除用户** |
| **memberbl.updmember** | **语法** | **public String updmember(MemberPO member)** |
| **前置条件** | **PO包含所有信息** |
| **后置条件** | **根据PO查找是否存在该客户，有则更新用户** |
| **memberbl.getMoney(String id,double b)** | **语法** | **public String getMoney(String id,double b)** |
| **前置条件** | **该用户存在，从用户处得到钱** |
| **后置条件** | **减少客户应该付金额** |
| **memberbl.payMoney(String id,double b)** | **语法** | **public String payMoney(String id,double b)** |
| **前置条件** | **该用户存在，给用户钱** |
| **后置条件** | **减少客户的应收金额** |
| **memberbl.findmember** | **语法** | **public String findmember(MemberPO member)** |
| **前置条件** | **无要求** |
| **后置条件** | **根据PO查找是否存在该客户，有则更新用户** |
| **需要的服务（需接口）** | | |
| **memberdataservice.get(int id,string name)** | **得到符合客户信息** | |
| **memberdataservice.del(memberPO po)** | **删除一个单一持久化对象** | |
| **memberdataservice.add(memberPO po)** | **新增一个单一持久化对象** | |
| **memberdataservice.getMoney(String id,double money)** | **从用户处得到钱，减少用户的应付** | |
| **memberdataservice.payMoney(String id,double money)** | **给用户钱，减少用户的应收** | |
| **memberdataservice.upd(memberPO po)** | **更新一个单一持久化对象** | |

### （4）业务逻辑层的动态模型

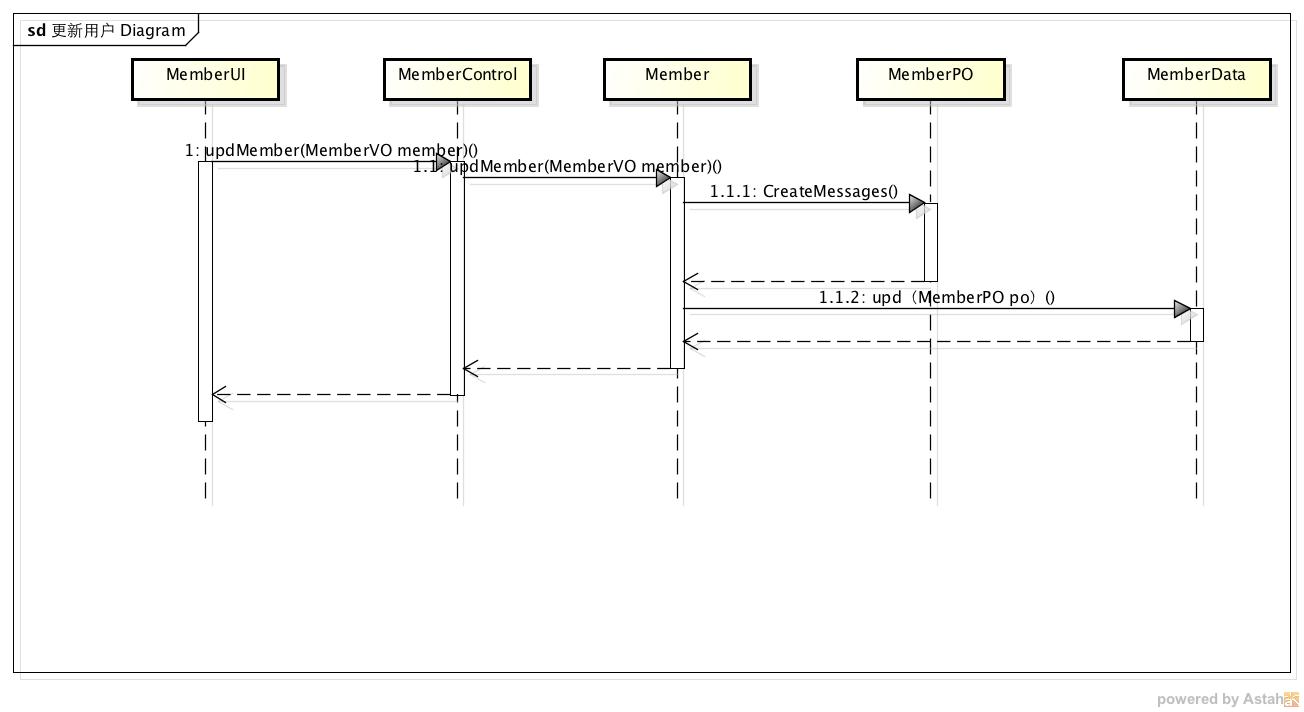
添加客户的顺序图：



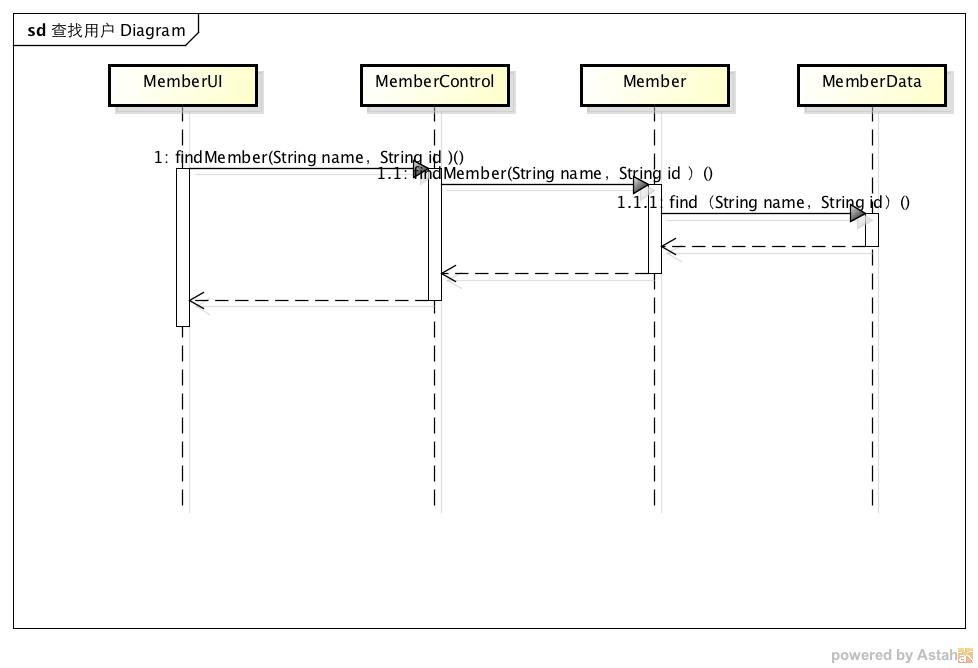
删除客户的顺序图；



更新客户的顺序图：

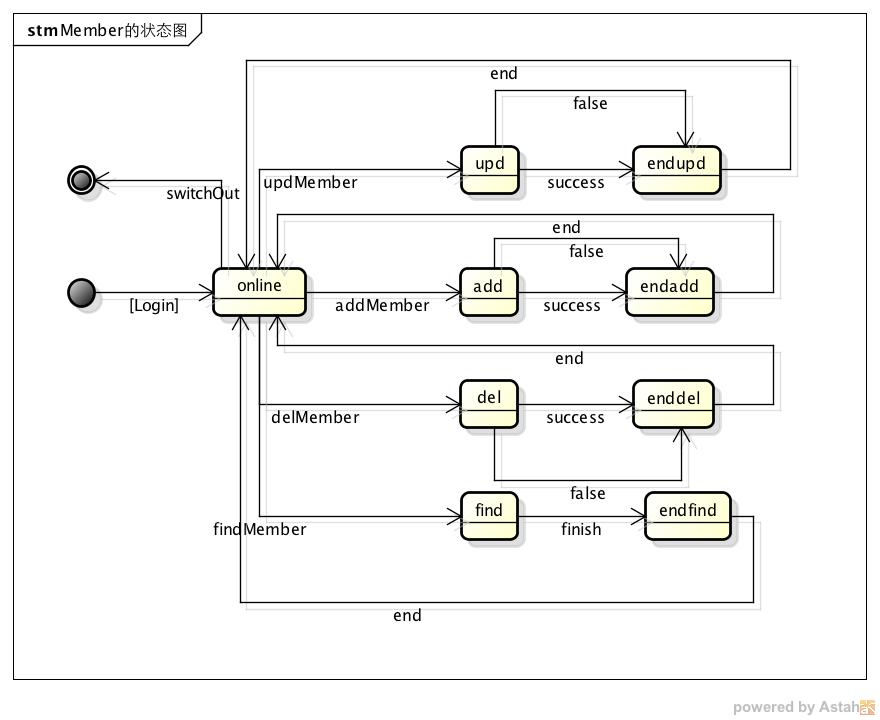


查找客户的顺序图：



状态图：

Member对象的状态图



### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

## 4.1.5MarketBL模块

### （1）模块概述

MarketBL模块承担了需求参见需求规格说明文档的功能需求以及非功能需求。

### （2）整体结构

根据体系结构的设计，我们将系统分为展示层，逻辑层，数据层。每层之间有接口。为了隔离业务逻辑职责和业务控制职责，我们添加了MarketControl，这样就将客户管理的业务逻辑委托给Market对象。SalePO，UnsalePO，StackPO，UnstackPO，FreePO，GoodsListPO是作为销售单据，赠送单和商品清单的持久化对象被储存的。其中GoodsList和GoodsLine,FreeList,FreeLine是添加Commodity的容器类。GoodsLine和FreeLine保有商品的数量以及商品的信息。GoodsList和FreeList是各自封装了关于Goodsline和FreeLine的数据集合的数据结构，具有计算总价的职责。

Marketbl的模块设计：



MarketBL模块各个类的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| MarketControl | 负责对应销售界面所需的服务 |
| Market | 销售任务的领域模型对象， 拥有销售任务的类型，商品的清单总价格这些信息，可以帮助完成销售页面所需服务 |
| Stack | 进货的领域模型对象， 拥有进货所持有的客户，编号，仓库，备注等信息，可以帮助完成销售页面进货操作所需服务 |
| Unstack | 退货的领域模型对象， 拥有退货所持有的客户，编号，仓库，备注等信息，可以帮助完成销售页面退货操作所需服务 |
| Sale | 销售的领域模型对象， 拥有销售所持有的客户，赠品的清单，编号，仓库，备注，折扣信息，赠送信息等信息，可以帮助完成销售页面销售操作所需服务 |
| Unsale | 销售退货的领域模型对象， 拥有销售退货所持有的客户，编号，仓库，备注，折扣信息，赠送信息等信息，可以帮助完成销售页面销售退货操作所需服务 |

### （3）模块内部的接口规范

MarketControl的接口规范：

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **MarketControl.enter** | **语法** | **Public BuyingsVO enter（）** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的enter方法** |
| **MarketControl.inputinformation** | **语法** | **public String inputinf(InformationVO info)** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的inputinf方法** |
| **MarketControl.listenter** | **语法** | **public BuyingsVO listenter(String commodity)** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的listenter方法** |
| **MarketConrol.listadd** | **语法** | **public String listadd(String commodity ,double value,int number)** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的listadd方法** |
| **MarketControl.listdel** | **语法** | **public String listdel(String commodity ,double value,int number)** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的listdel方法** |
| **MarketControl.listupd** | **语法** | **public String listupd(String commodity ,double value,int number)** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的listupd方法** |
| **MarketControl.listshow** | **语法** | **public listVO listshow（）** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的listshow方法** |
| **MarketControl.reduce** | **语法** | **Public String reduce(double price)** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的reduce方法** |
| **MarketControl.discount** | **语法** | **Public String discount() //可以放入Market中** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的discount方法** |
| **MarketControl.** **sure** | **语法** | **Public String sure ()** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的sure方法** |
| **MarketControl.reReduce** | **语法** | **public String reReduce（double price）** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的reReduce方法** |
| **MarketControl.showFree（）** | **语法** | **public ListVO showFree（）** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的showFree方法** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **Market.enter** | **提供进入购物车所需的目录或商品** | |
| **Market.inputinformation** | **得到销售任务的基本信息** | |
| **Market.listenter** | **提供商品目录子目录的信息** | |
| **Market.listadd** | **添加商品进入购物车** | |
| **Market.listdel** | **删除商品进入购物车** | |
| **Market.listupd** | **更新商品进入购物车** | |
| **Market.listshow** | **在销售界面查看商品列表** | |
| **Market.discount** | **查看促销策略** | |
| **Market.** **Sure** | **确认单据** | |
| **Market.showFree** | **查看赠品单** | |
| **Market.reduce** | **销售员优惠** | |
| **Market.reReduce** | **销售员更新优惠** | |

Market的接口规范：

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **Market.enter** | **语法** | **Public BuyingsVO enter（）** |
| **前置条件** | **进入商品选择界面** |
| **后置条件** | **返回商品和商品文件夹** |
| **Market.inputinformation** | **语法** | **public String inputinf(InformationVO info)** |
| **前置条件** | **进入销售任务，且输入信息正确** |
| **后置条件** | **根据信息建立sale，stack，unstack，unsale中的一个** |
| **Market.listenter** | **语法** | **public BuyingsVO listenter(String commodity)** |
| **前置条件** | **已经建立sale，stack，unstack，unsale中的一个** |
| **后置条件** | **返回文件夹内商品或者商品信息** |
| **Market.listadd** | **语法** | **public String listadd(String commodity ,double value,int number)** |
| **前置条件** | **已经建立sale，stack，unstack，unsale中的一个** |
| **后置条件** | **添加一个商品进入list** |
| **Market.listdel** | **语法** | **public String listdel(String commodity ,double value,int number)** |
| **前置条件** | **该商品已被添加** |
| **后置条件** | **删除商品列表中的商品** |
| **Market.listupd** | **语法** | **public String listupd(String commodity ,double value,int number)** |
| **前置条件** | **该商品已被添加** |
| **后置条件** | **更新商品列表中的商品** |
| **Market.listshow** | **语法** | **public listVO listshow（）** |
| **前置条件** | **已经完成了对商品列表的操作** |
| **后置条件** | **显示商品清单** |
| **Market.reduce** | **语法** | **Public String reduce(double price)** |
| **前置条件** | **已经建立sale的领域模型** |
| **后置条件** | **销售员折扣完成** |
| **Market.discount** | **语法** | **Public String discount()** |
| **前置条件** | **已经建立sale的领域模型** |
| **后置条件** | **促销策略添加** |
| **Market.** **Sure（）** | **语法** | **Public String sure（）** |
| **前置条件** | **完成了对商品清单的构建** |
| **后置条件** | **将单据传输给billmanagement** |
| **Market.reReduce** | **语法** | **public String reReduce（double price）** |
| **前置条件** | **已经完成对商品清单的构建** |
| **后置条件** | **更新销售人员优惠** |
| **Market.** **replyUnstack** | **语法** | **public String replyUnstack（）** |
| **前置条件** | **已经建立unstack领域模型** |
| **后置条件** | **储存单据进入数据层** |
| **Market.** **replyUnsale** | **语法** | **public String replyUnsale（）** |
| **前置条件** | **已经建立unsale领域模型** |
| **后置条件** | **储存单据进入数据层** |
| **Market.** **replyFree** | **语法** | **public String replyFree（）** |
| **前置条件** | **已经建立sale领域模型** |
| **后置条件** | **储存单据进入数据层** |
| **Market.** **replyStack** | **语法** | **public String replyStack（）** |
| **前置条件** | **已经建立stack领域模型** |
| **后置条件** | **储存单据进入数据层** |
| **Market.** **replySale** | **语法** | **public String replySale（）** |
| **前置条件** | **已经建立sale领域模型** |
| **后置条件** | **储存单据进入数据层** |
| **Market.** **findStack** | **语法** | **public StackVO findStack（）** |
| **前置条件** | **已经建立stack领域模型** |
| **后置条件** | **得到符合要求的单据** |
| **Market.** **findUnstack** | **语法** | **public UnstackVO findUnstack（）** |
| **前置条件** | **已经建立Unstack领域模型** |
| **后置条件** | **得到符合要求的单据** |
| **Market.** **findSale** | **语法** | **public SaleVO findSale（）** |
| **前置条件** | **已经建立sale领域模型** |
| **后置条件** | **得到符合要求的单据** |
| **Market.** **findUnsale** | **语法** | **public UnsaleVOfindUnsale（）** |
| **前置条件** | **已经建立Unsale域模型** |
| **后置条件** | **得到符合要求的单据** |
| **Market.** **findFree** | **语法** | **public FreeVO findFree（）** |
| **前置条件** | **已经建立sale领域模型** |
| **后置条件** | **得到符合要求的单据** |
| **Market.** **cal** | **语法** | **public String cal（）** |
| **前置条件** | **商品清单中有商品变化** |
| **后置条件** | **计算出商品清单中的总价及各项商品价格** |
| **Market.showFree（）** | **语法** | **public ListVO showFree（）** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的showFree方法** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **GoodsList.getPrice()** | **返回商品列表的价格** | |
| **Sale.discount(int level,double allPrice)** | **得到新的总价格，折扣价格** | |
| **GoodsList.del()** | **更新商品信息从而减少库存中某个商品的数量** | |
| **GoodsList.add()** | **更新商品信息从而增加库存中某个商品的数量** | |
| **GoodsList.replyFree(FreeVO)** | **处理赠品单所造成的库存变化** | |
| **GoodsList.** **listdel(String commodity ,double value,int number)** | **删除商品从商品列表中** | |
| **GoodsList.** **listadd(String commodity ,double value,int number)** | **添加商品从商品列表中** | |
| **GoodsList.** **listupd(String commodity ,double value,int number)** | **更新商品从商品列表中** | |
| **Commodity.find(String id)** | **从Commodity获取商品** | |
| **Kind.find(String id)** | **从kind获取文件下的子目录** | |
| **Sale.reReduce（double price）** | **更新销售员优惠** | |
| **Sale.reduce（double price）** | **输入销售员优惠** | |
| **GoodsList.listshow()** | **显示商品清单** | |
| **Sale.showFree()** | **显示赠品清单** | |
| **Stack.save(GoodsListPO)** | **将单据传给总经理** | |
| **Stack.sure(GoodsListPO)** | **储存已审批的单据** | |
| **Unstack.save(GoodsListPO)** | **将单据传给总经理** | |
| **Unstack.sure(GoodsListPO)** | **储存已审批的单据** | |
| **Sale.save(GoodsListPO)** | **将单据传给总经理** | |
| **Sale.sure(GoodsListPO)** | **储存已审批的单据** | |
| **Unsale.save(GoodsListPO)** | **将单据传给总经理** | |
| **Unsale.sure(GoodsListPO)** | **储存已审批的单据** | |
| **Stack.find(inforVO)** | **查询进货单据** | |
| **Unsale.find(inforVO)** | **查询销售退货单据** | |
| **Unstack.find(inforVO)** | **查询退货单据** | |
| **Sale.findsale(inforVO)** | **查询销售单据** | |
| **Sale.findFree(inforVO)** | **查询赠品单据** | |
| **GoodsList.sure()** | **审批通过后更新清单中商品的库存** | |
| **Sale.saveFree(FreePO po)** | **审批通过后储存赠品单** | |
| **GoodsList.cal（）** | **计算商品列表中总金额** | |

Unity接口规范

|  |  |  |
| --- | --- | --- |
| **Unity.sure** | **语法** | **Public String sure(GoodsListPO po)** |
| **前置条件** | **已经建立符合要求的类** |
| **后置条件** | **建立新的可持久对象** |
| **Unity.find** | **语法** | **Public String find（inforVO vo）** |
| **前置条件** | **输入信息格式符合要求** |
| **后置条件** | **查询符合信息的单据** |
| **Unity.save** | **语法** | **public String save（GoodsListPO po）** |
| **前置条件** | **已建立符合要求的类** |
| **后置条件** | **储存单据进入数据层** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **StackDataService.save(StackPO po)** | **储存单据进入数据层** | |
| **StackDataservice.find(infor VO)** | **在数据层查询符合信息的单据** | |
| **Po.StackPO** | **对可持久对象的建立** | |
| **Billmanegement.send(StackPO po)** | **将单据发送给总经理** | |
| **StackDataService.save(StackPO po)** | **储存单据进入数据层** | |
| **StackDataservice.find(infor VO)** | **在数据层查询符合信息的单据** | |
| **Po.StackPO** | **对可持久对象的建立** | |
| **UnsalekDataService.save(StackPO po)** | **储存单据进入数据层** | |
| **UnsaleDataservice.find(infor VO)** | **在数据层查询符合信息的单据** | |
| **Po.UnsalePO** | **对可持久对象的建立** | |
| **SaleDataService.save(StackPO po)** | **储存单据进入数据层** | |
| **SaleDataservice.find(infor VO)** | **在数据层查询符合信息的单据** | |
| **Po.SalePO** | **对可持久对象的建立** | |

GoodsList接口规范

|  |  |  |
| --- | --- | --- |
| **GoodsList.getPrice（）** | **语法** | **Public String getPrice()** |
| **前置条件** | **无** |
| **后置条件** | **返回返回商品列表的价格** |
| **GoodsList.del()** | **语法** | **public String del（）** |
| **前置条件** | **有审批单据通过** |
| **后置条件** | **减少商品列表中的商品的库存** |
| **GoodsList.add()** | **语法** | **public String add（）** |
| **前置条件** | **有审批单据通过** |
| **后置条件** | **增加商品列表中的商品的库存** |
| **GoodsList.replyFree(FreeVO)** | **语法** | **public String replyFree(FreeVO)** |
| **前置条件** | **有审批赠品单据通过** |
| **后置条件** | **更新赠品列表中的商品的库存** |
| **GoodsList.** **listdel(String commodity ,double value,int number)**  **)** | **语法** | **public String listdel(String commodity ,double value,int number)** |
| **前置条件** | **该商品已被添加** |
| **后置条件** | **删除商品列表中的商品** |
| **GoodsList.** **listadd(String commodity ,double value,int number)** | **语法** | **public String listadd(String commodity ,double value,int number)** |
| **前置条件** | **无** |
| **后置条件** | **添加商品列表中的商品** |
| **GoodsList.** **listupd (String commodity ,double value,int number)** | **语法** | **public String listupd (String commodity ,double value,int number)** |
| **前置条件** | **该商品已被添加** |
| **后置条件** | **更新商品列表中的商品** |
| **GoodsList.listshow()** | **语法** | **Public ListVO listshow()** |
| **前置条件** | **无** |
| **后置条件** | **返回列表中的商品信息** |
| **GoodsList.sure()** | **语法** | **Public GoodsListPO sure()** |
| **前置条件** | **任务已经完成** |
| **后置条件** | **返回商品列表的可持续对象** |
| **GoodsList.cal（）** | **语法** | **Public double cal（）** |
| **前置条件** | **商品列表有变化** |
| **后置条件** | **计算列表中的价格，返回总价格** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **GoodsLine.** **add（String name,int number）** | **将商品项的商品在库存中增加** | |
| **GoodsLine.** **del（String name,int number）** | **将商品项的商品在库存中减少** | |
| **GoodLine.** **updGoods（int number，double price）** | **将商品列表中的该商品项更改** | |
| **GoodLine.show()** | **返回商品列表该商品项的信息** | |
| **GoodLine.addGoods（String commodityid，int number，double price）** | **新建一个商品项以所给信息** | |
| **GoodsLine.cal()** | **给出该商品项的总额** | |
| **Po.GoodsListPO** | **构造一个包含该信息的可持续文件** | |

GoodsLine的接口规范

|  |  |  |
| --- | --- | --- |
| **GoodsLine.** **add（String name,int number）** | **语法** | **Public String add（String name,int number）** |
| **前置条件** | **输入信息格式符合要求** |
| **后置条件** | **增加该商品的库存** |
| **GoodsLine.** **del（String name,int number）** | **语法** | **Public String del（String name,int number）** |
| **前置条件** | **输入信息格式符合要求** |
| **后置条件** | **减少该商品的库存** |
| **GoodLine.** **updGoods（int number，double price）** | **语法** | **public String updGoods（int number，double price）** |
| **前置条件** | **输入信息符合格式** |
| **后置条件** | **更新对象本身** |
| **GoodLine.show()** | **语法** | **public BuyingVO show（）** |
| **前置条件** | **无** |
| **后置条件** | **返回该商品的值对象** |
| **GoodLine.addGoods（String commodityid，int number，double price）** | **语法** | **public String addGoods（String commodityid，int number，double price）** |
| **前置条件** | **输入符合格式** |
| **后置条件** | **将对象赋值为所给值** |
| **GoodsLine.cal()** | **语法** | **public double cal（）** |
| **前置条件** | **无** |
| **后置条件** | **返回该商品的总价** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **Commodity.** **add（String name,int number）** | **增加该商品的库存** | |
| **Commodity.** **del（String name,int number）** | **减少该商品的库存** | |
| **Commodity.find(Stringid)** | **得到该商品** | |

Stack的接口规范

|  |  |  |
| --- | --- | --- |
| **Stack.sure** | **语法** | **Public String sure(GoodsListPO po)** |
| **前置条件** | **已经建立符合要求的类** |
| **后置条件** | **建立新的可持久对象** |
| **Stack.find** | **语法** | **Public String find（inforVO vo）** |
| **前置条件** | **输入信息格式符合要求** |
| **后置条件** | **查询符合信息的单据** |
| **Stack.save** | **语法** | **public String save（GoodsListPO po）** |
| **前置条件** | **已建立符合要求的类** |
| **后置条件** | **储存单据进入数据层** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **StackDataService.save(StackPO po)** | **储存单据进入数据层** | |
| **StackDataservice.find(infor VO)** | **在数据层查询符合信息的单据** | |
| **Po.StackPO** | **对可持久对象的建立** | |
| **Billmanegement.send(StackPO po)** | **将单据发送给总经理** | |

Unstack的接口规范

|  |  |  |
| --- | --- | --- |
| **Unstack.sure** | **语法** | **Public String sure(GoodsListPO po)** |
| **前置条件** | **已经建立符合要求的类** |
| **后置条件** | **建立新的可持久对象** |
| **Unstack.find** | **语法** | **Public String find（inforVO vo）** |
| **前置条件** | **输入信息格式符合要求** |
| **后置条件** | **查询符合信息的单据** |
| **Unstack.save** | **语法** | **public String save（GoodsListPO po）** |
| **前置条件** | **已建立符合要求的类** |
| **后置条件** | **储存单据进入数据层** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **UnstackDataService.save(StackPO po)** | **储存单据进入数据层** | |
| **UnstackDataservice.find(infor VO)** | **在数据层查询符合信息的单据** | |
| **Po.UnstackPO** | **对可持久对象的建立** | |
| **Billmanegement.send(UnstackPO po)** | **将单据发送给总经理** | |

Unsale的接口规范

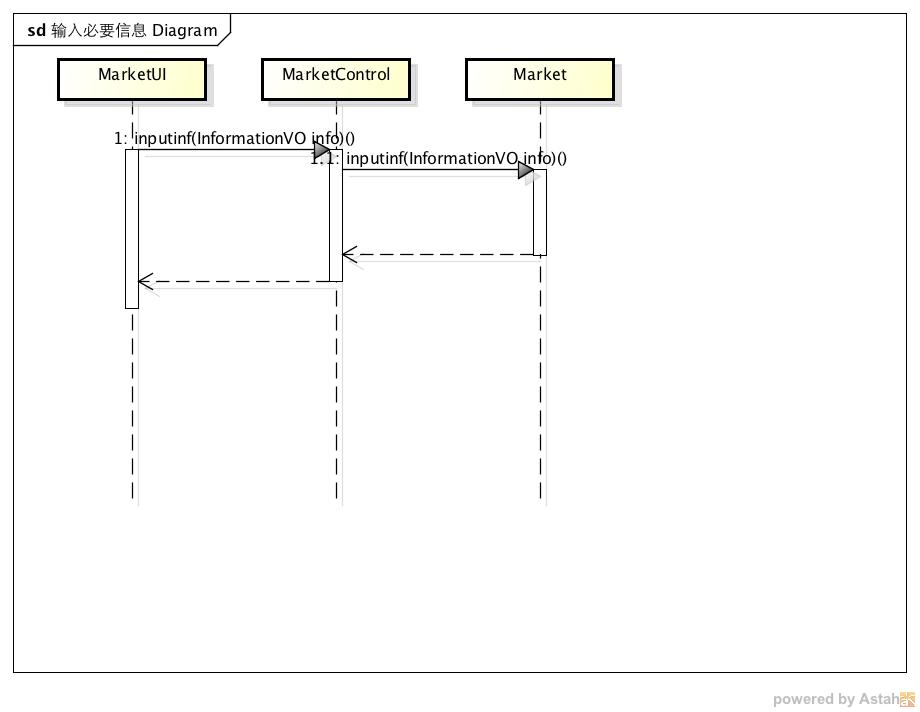
|  |  |  |
| --- | --- | --- |
| **Unsale.sure** | **语法** | **Public String sure(GoodsListPO po)** |
| **前置条件** | **已经建立符合要求的类** |
| **后置条件** | **建立新的可持久对象** |
| **Unsale.find** | **语法** | **Public String find（inforVO vo）** |
| **前置条件** | **输入信息格式符合要求** |
| **后置条件** | **查询符合信息的单据** |
| **Unsale.save** | **语法** | **public String save（GoodsListPO po）** |
| **前置条件** | **已建立符合要求的类** |
| **后置条件** | **储存单据进入数据层** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **UnsalekDataService.save(StackPO po)** | **储存单据进入数据层** | |
| **UnsaleDataservice.find(infor VO)** | **在数据层查询符合信息的单据** | |
| **Po.UnsalePO** | **对可持久对象的建立** | |
| **Promotion.discount（int Memberlevel，double price）** | **获取促销信息所得到的优惠** | |
| **Billmanegement.send(Unsale po)** | **将单据发送给总经理** | |

Sale的接口规范：

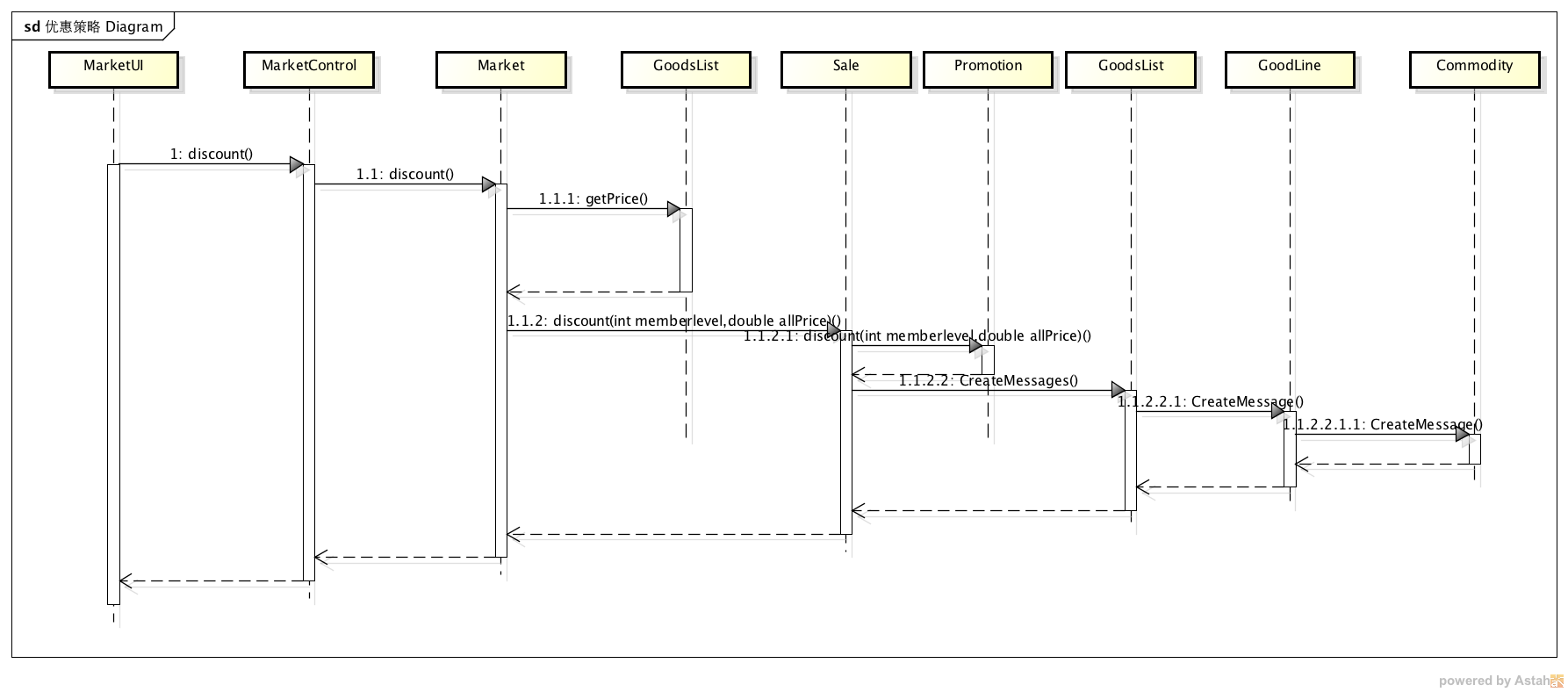
|  |  |  |
| --- | --- | --- |
| **Sale.saveFree** | **语法** | **Public String saveFree（FreeVO）** |
| **前置条件** | **已经建立符合要求的类** |
| **后置条件** | **储存已审批的赠品单** |
| **Sale.findFree** | **语法** | **Public FreeVO fineFree（inforVO vo）** |
| **前置条件** | **已经建立符合要求的类** |
| **后置条件** | **查找符合信息的赠品单** |
| **Sale.find** | **语法** | **Public String find（inforVO vo）** |
| **前置条件** | **输入信息格式符合要求** |
| **后置条件** | **查询符合信息的单据** |
| **Sale.save** | **语法** | **public String save（GoodsListPO po）** |
| **前置条件** | **已建立符合要求的类** |
| **后置条件** | **储存单据进入数据层** |
| **Sale.sure** | **语法** | **public String sure（GoodsListPO po）** |
| **前置条件** | **已建立符合要求的类** |
| **后置条件** | **储存单据进入数据层** |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| **SaleDataService.save(StackPO po)** | **储存单据进入数据层** | |
| **SaleDataservice.find(infor VO)** | **在数据层查询符合信息的单据** | |
| **Po.SalePO** | **对可持久对象的建立** | |
| **SaleDataService.saveFree(FreePO po)** | **存储完成审批的赠品单** | |
| **SaleDataService.findFree(inforVO vo)** | **查询数据层的赠品单** | |
| **Billmanegement.send(FreePO po)** | **将单据发送给总经理** | |
| **Promotion.** **discountA(int memberlevel,double allPrice)** | **获取赠送策略的赠品** | |
| **Promotion.** **discountP(int memberlevel,double allPrice)** | **获取赠送策略的折扣** | |
| **Billmanegement.send(SalePO po)** | **将单据发送给总经理** | |

### （4） 业务逻辑层的动态模型

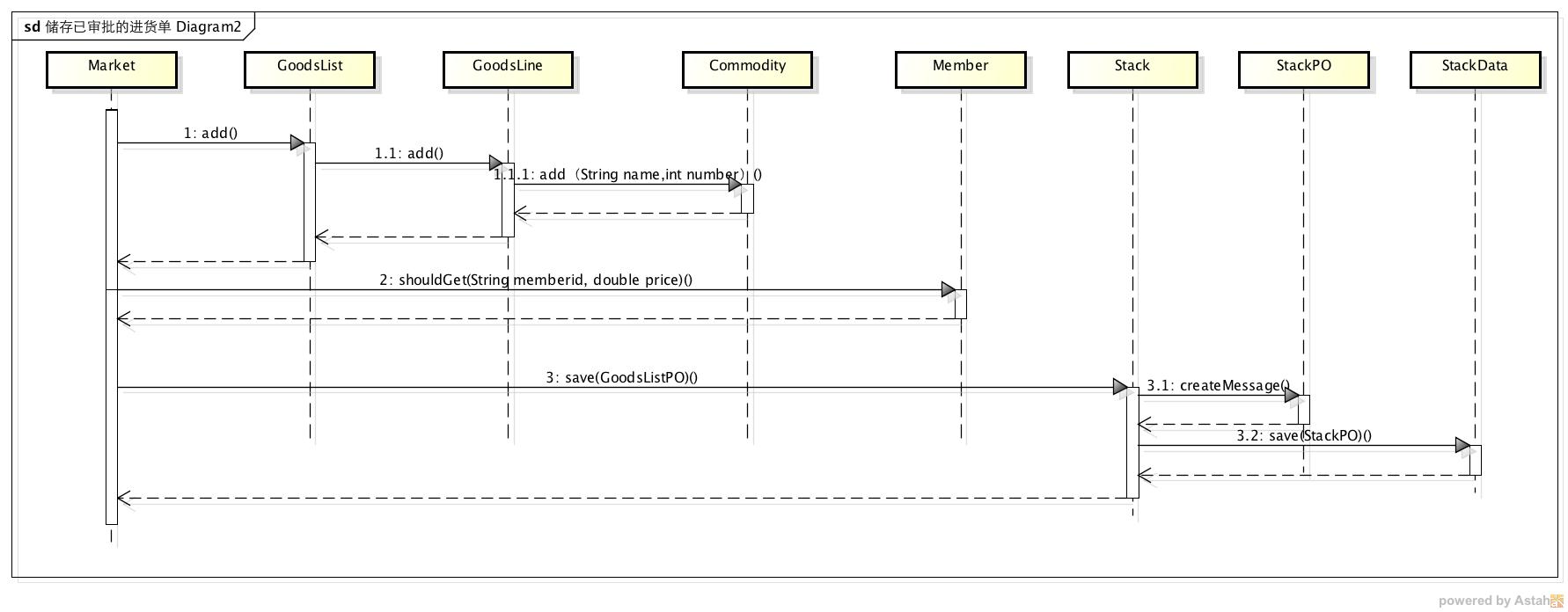
展示层输入销售任务主要信息的顺序图



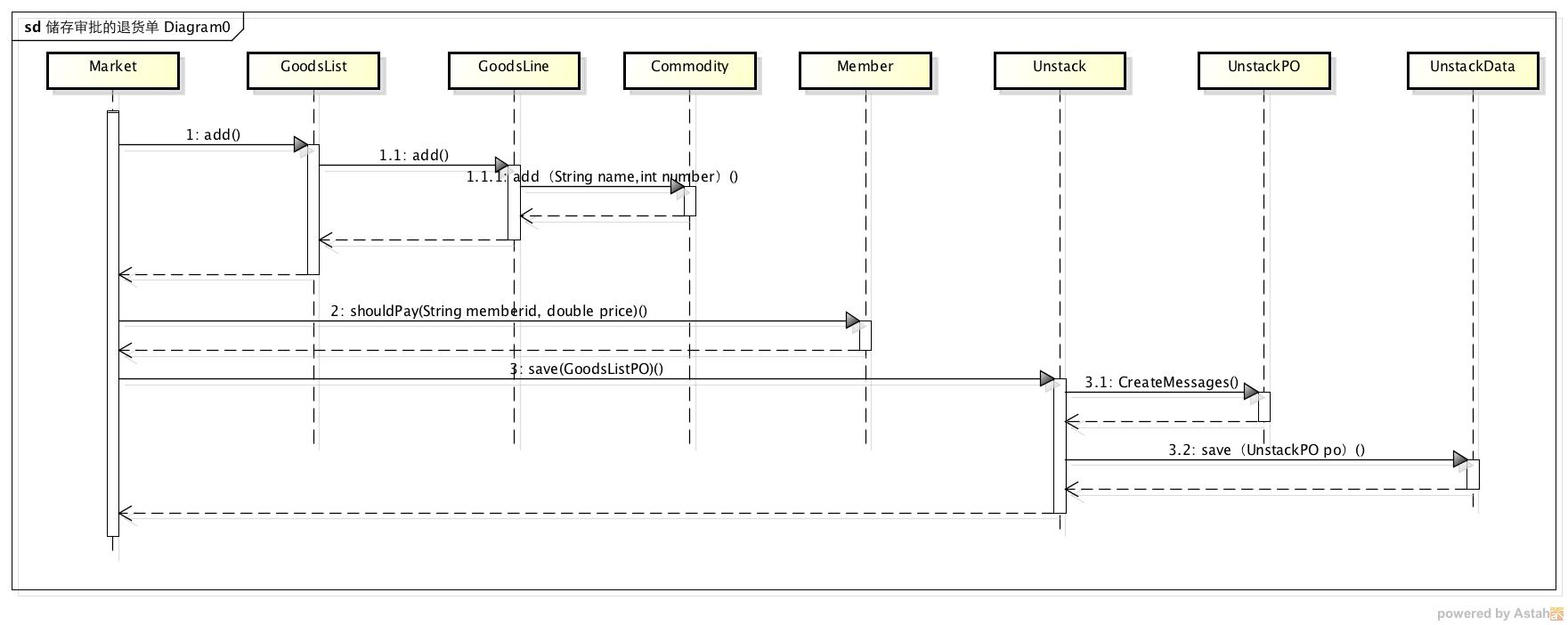
获得策略优惠的顺序图



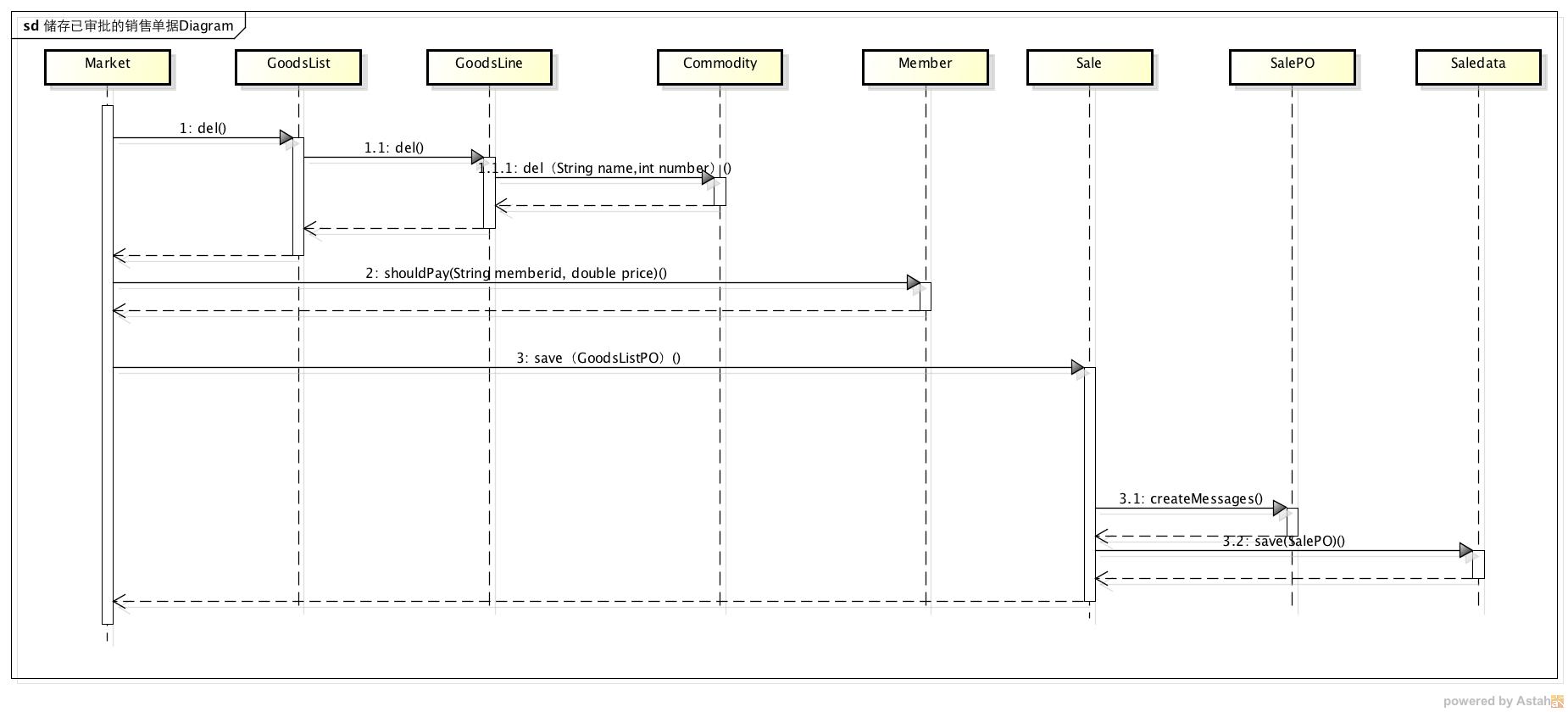
储存已审批的进货单的顺序图



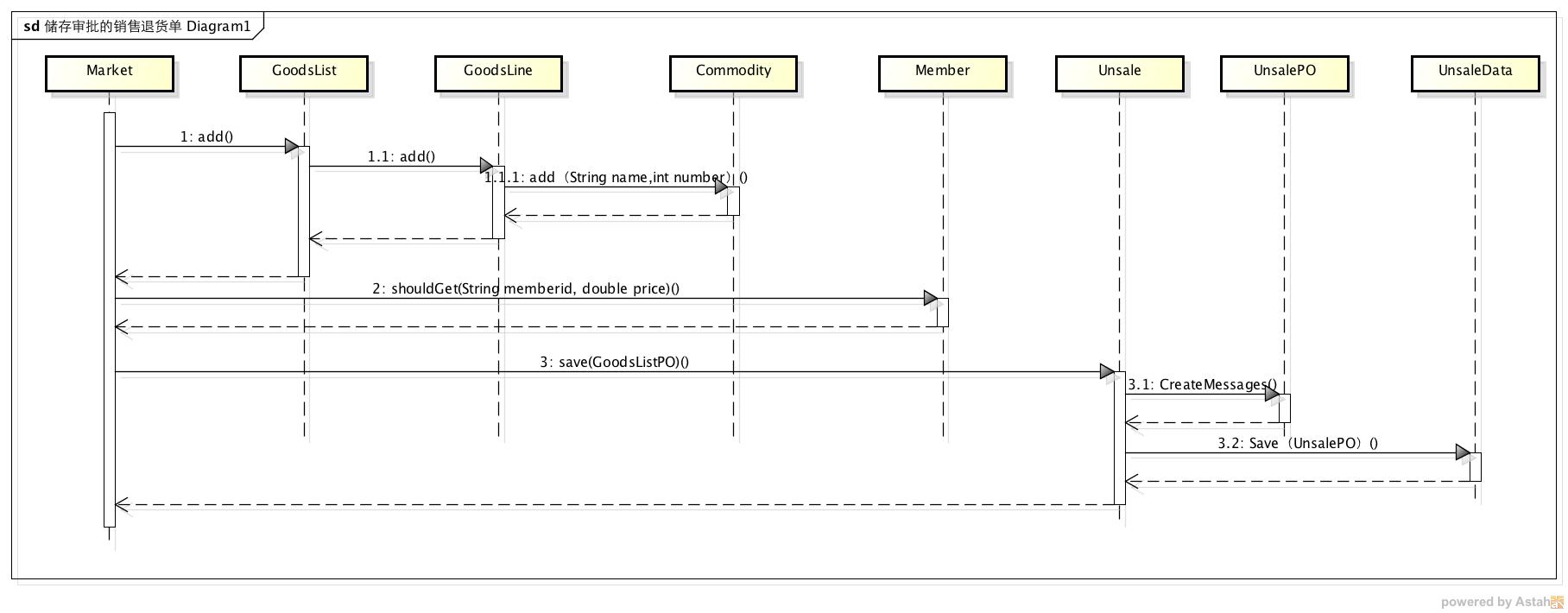
储存已审批的退货单的顺序图



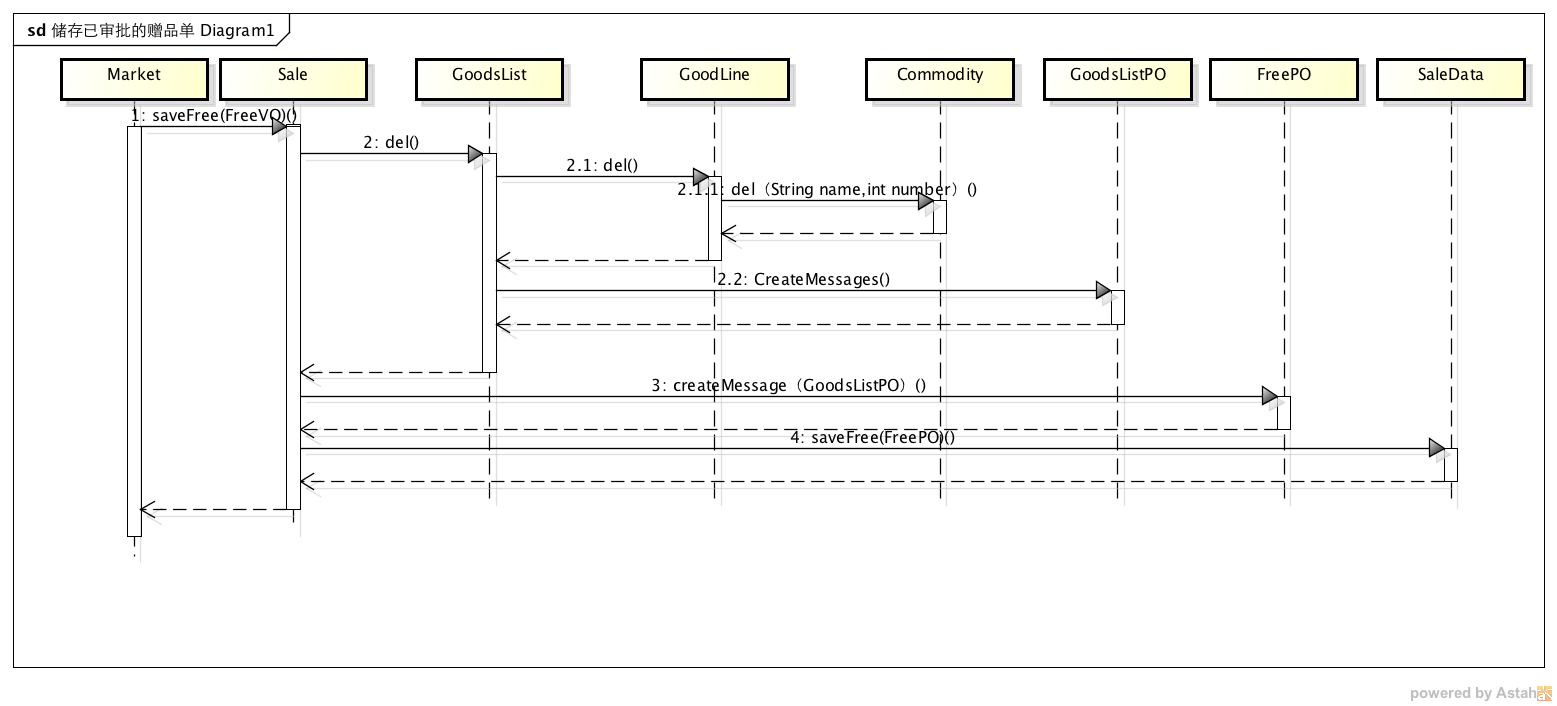
储存已审批的销售单的顺序图



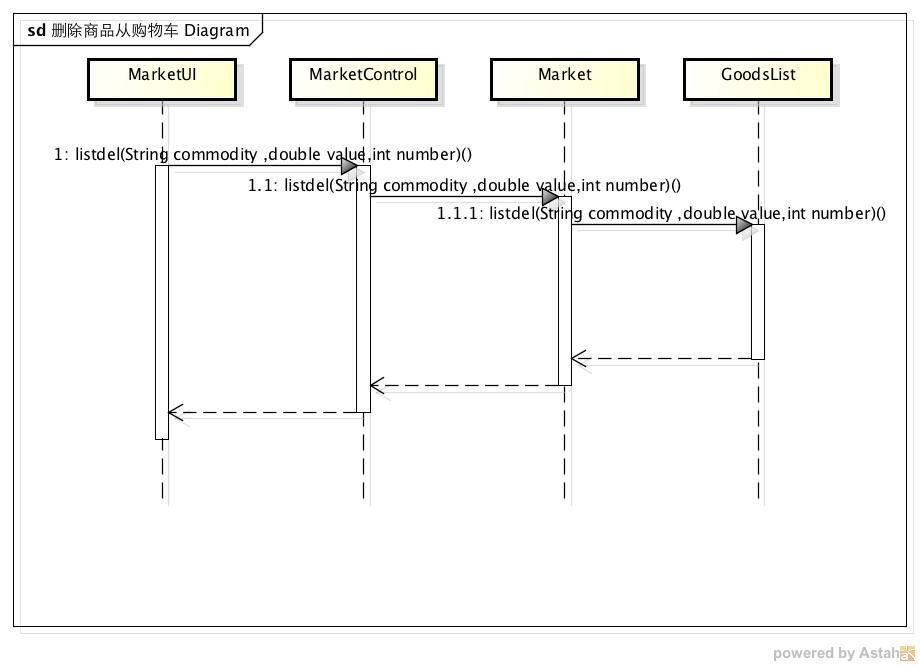
储存已审批的销售退货单的顺序图



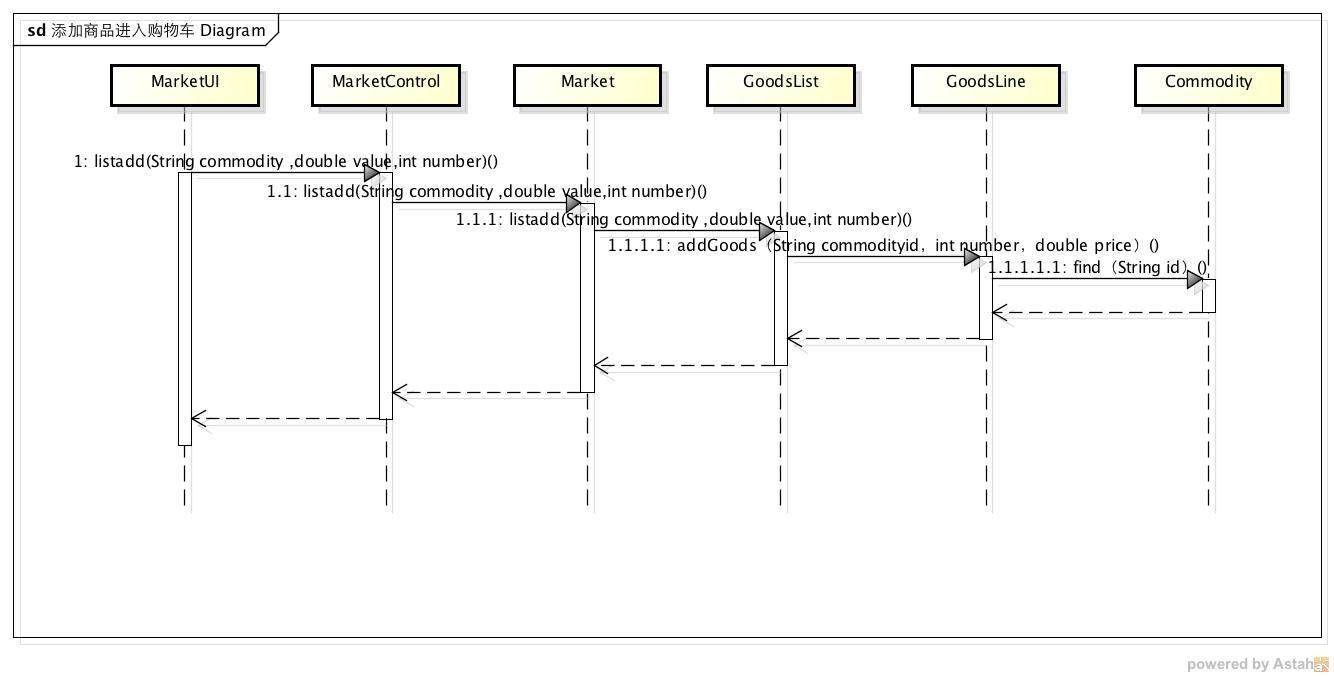
储存已审批的赠品单的顺序图



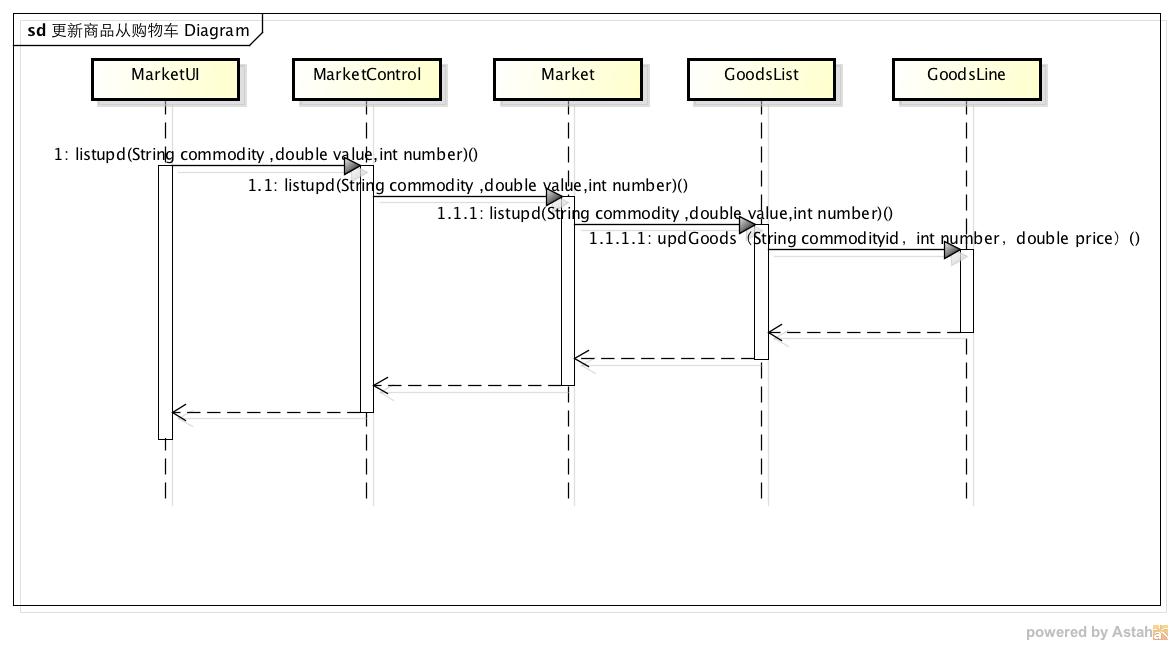
从购物车中删除商品的顺序图



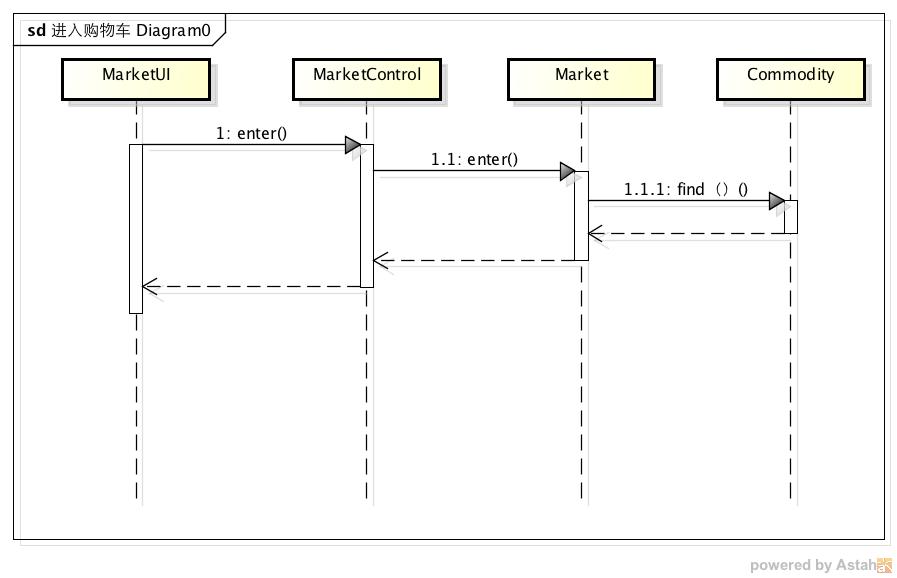
从购物车中添加商品的顺序图



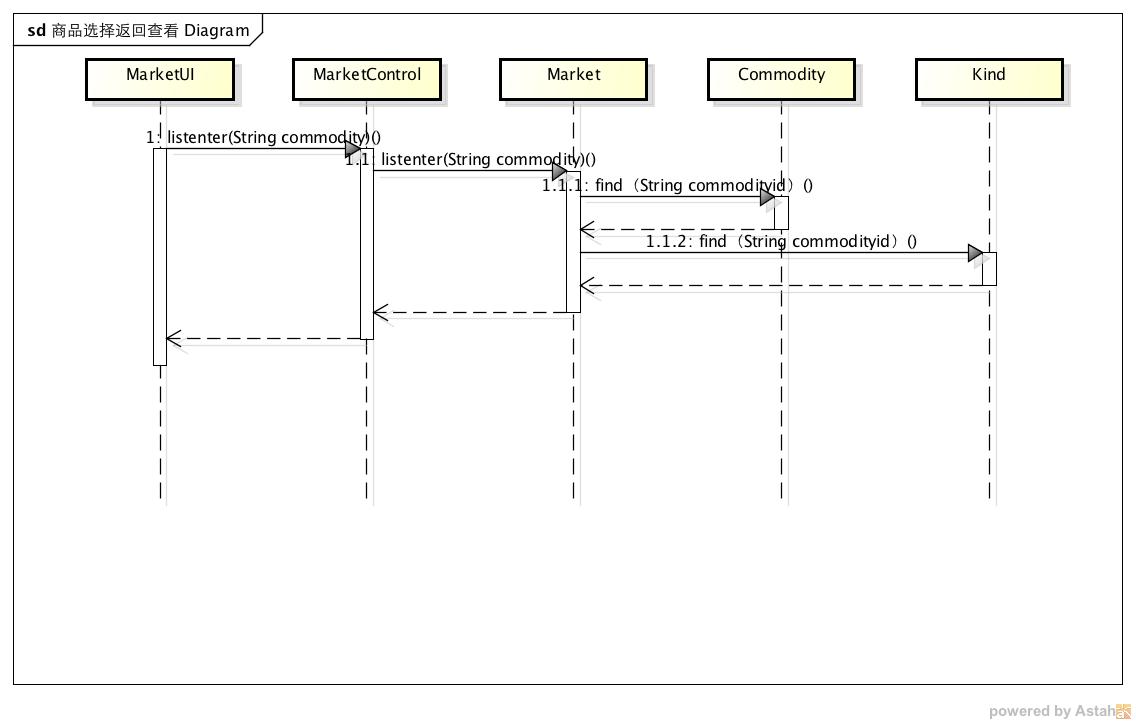
从购物车中更新商品的顺序图



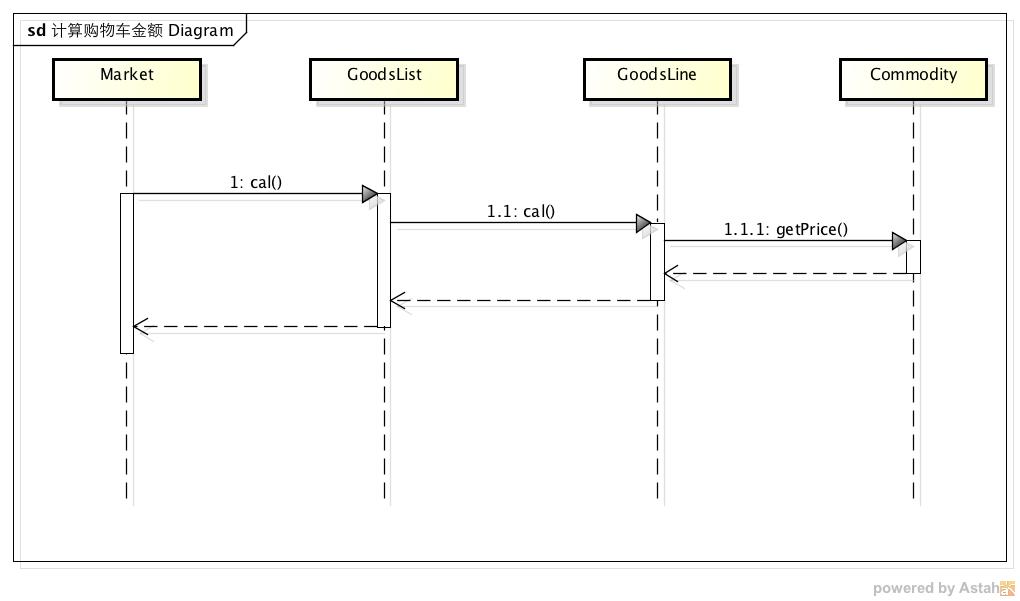
选择进入商品选择界面的顺序图



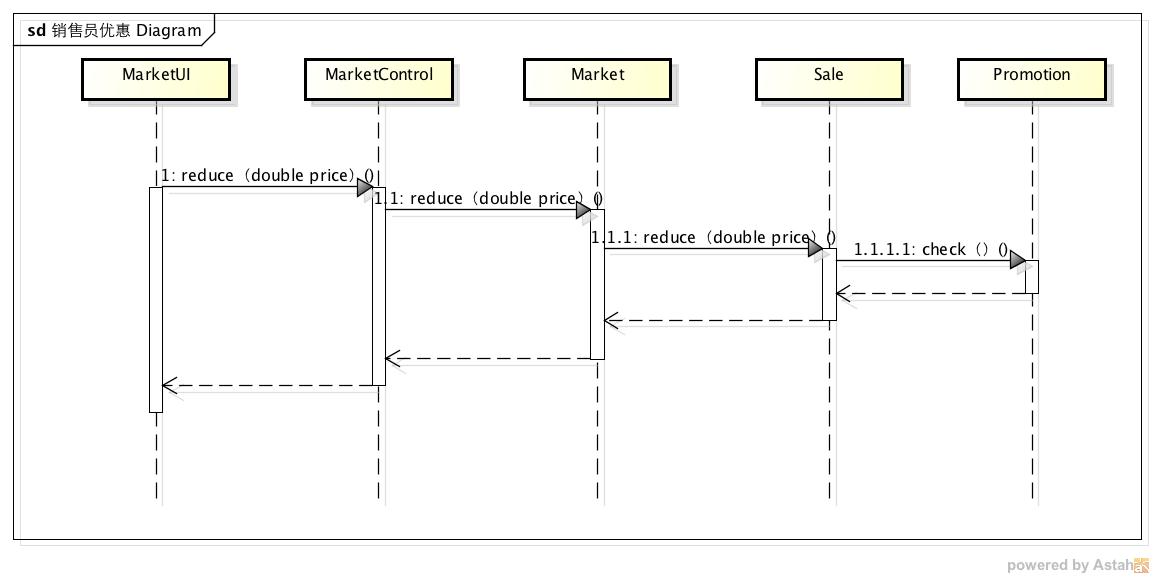
在商品选择界面选择商品或者商品夹的顺序图



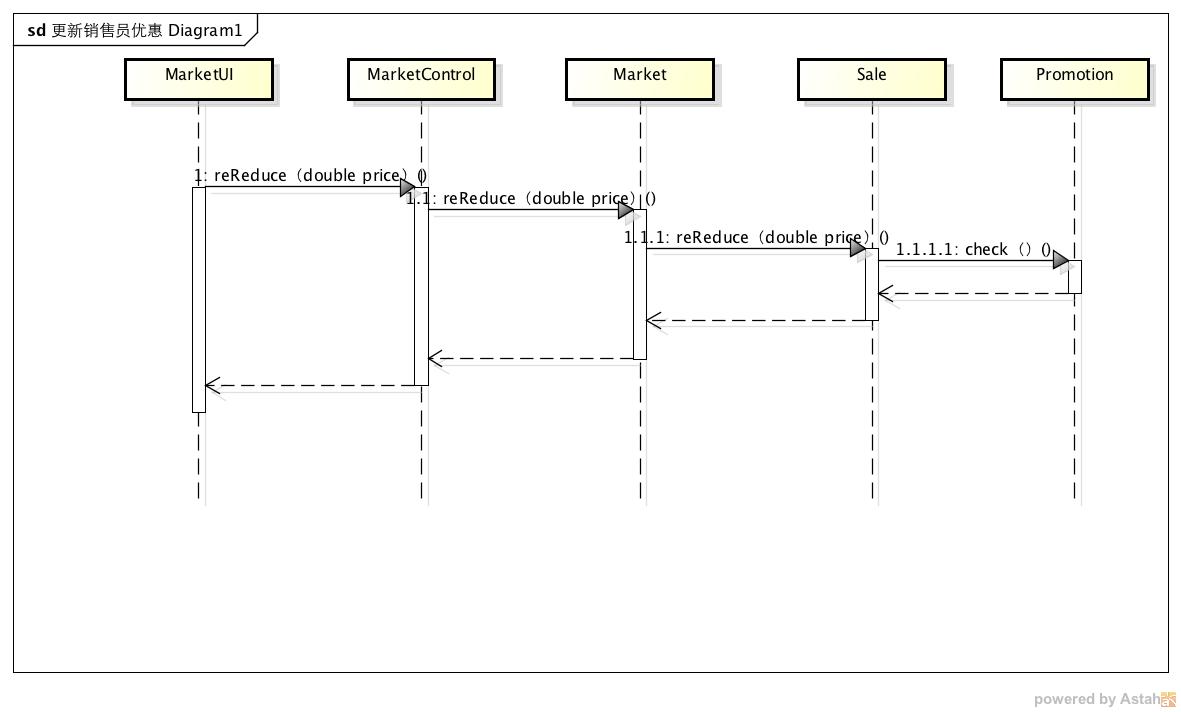
计算购物车金额的顺序图



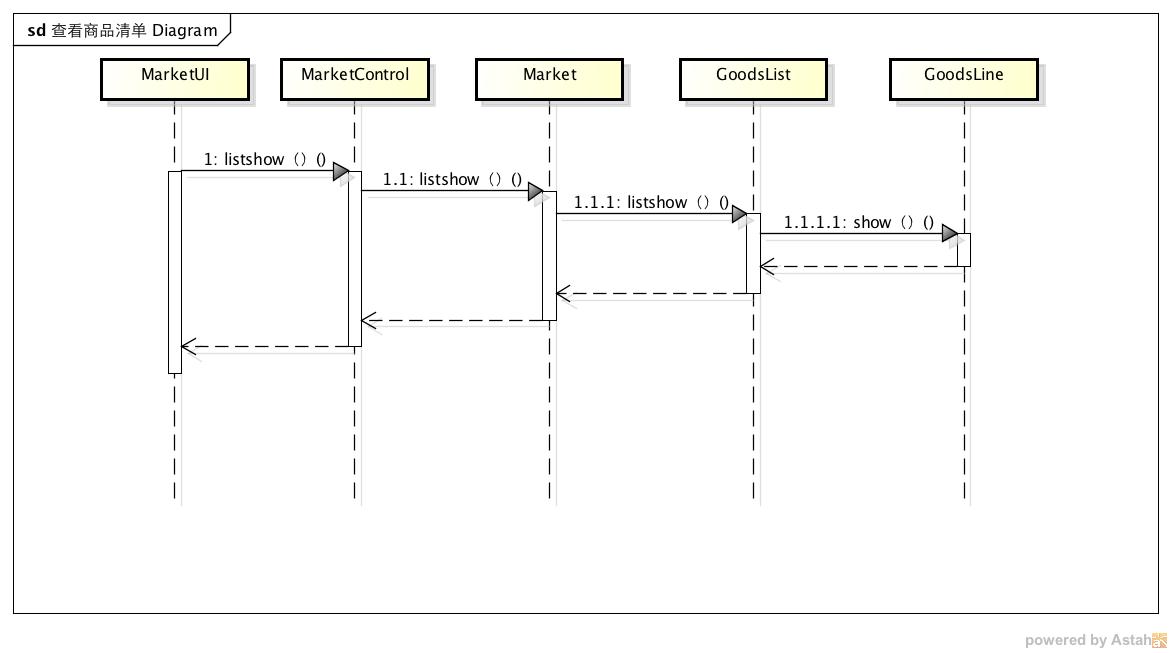
输入销售员折扣与折扣券的顺序图



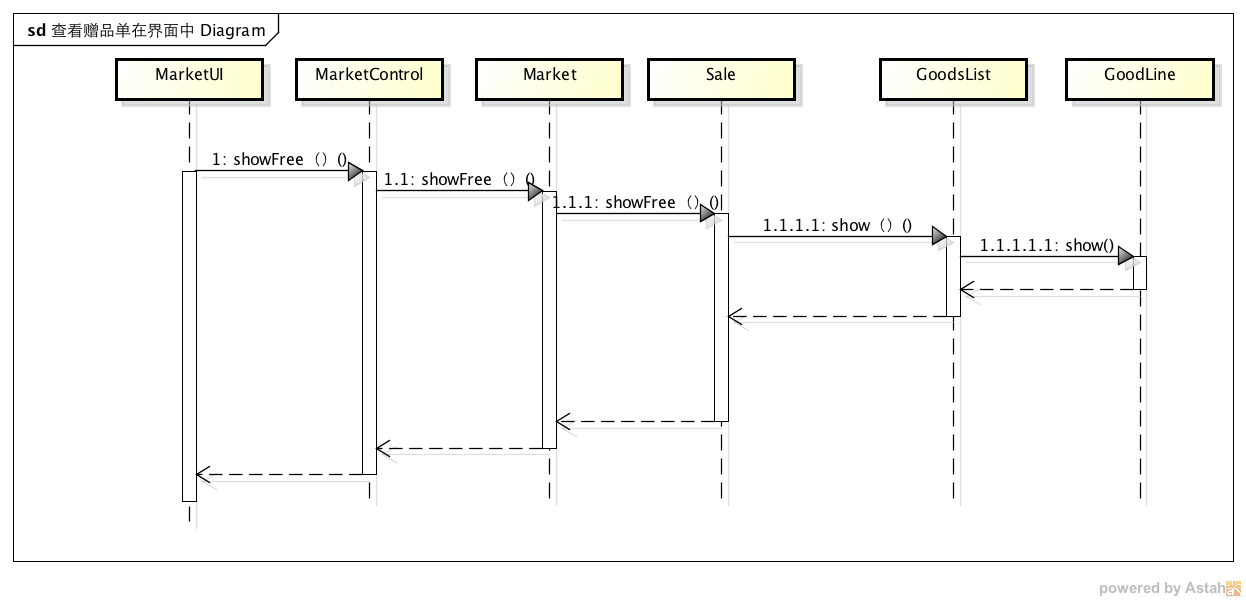
更新销售人员优惠和折扣券使用的顺序图



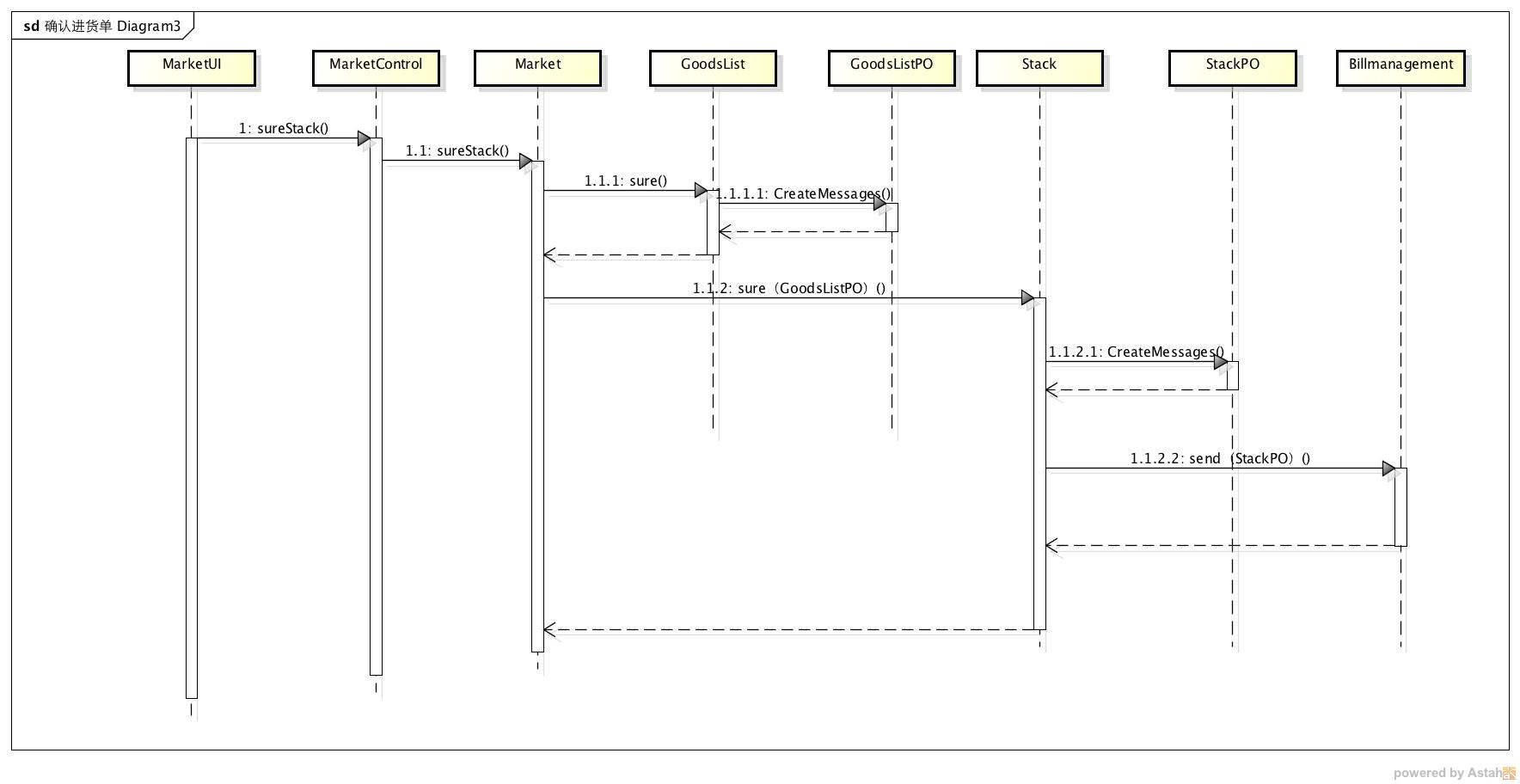
在销售界面查看商品清单的顺序图



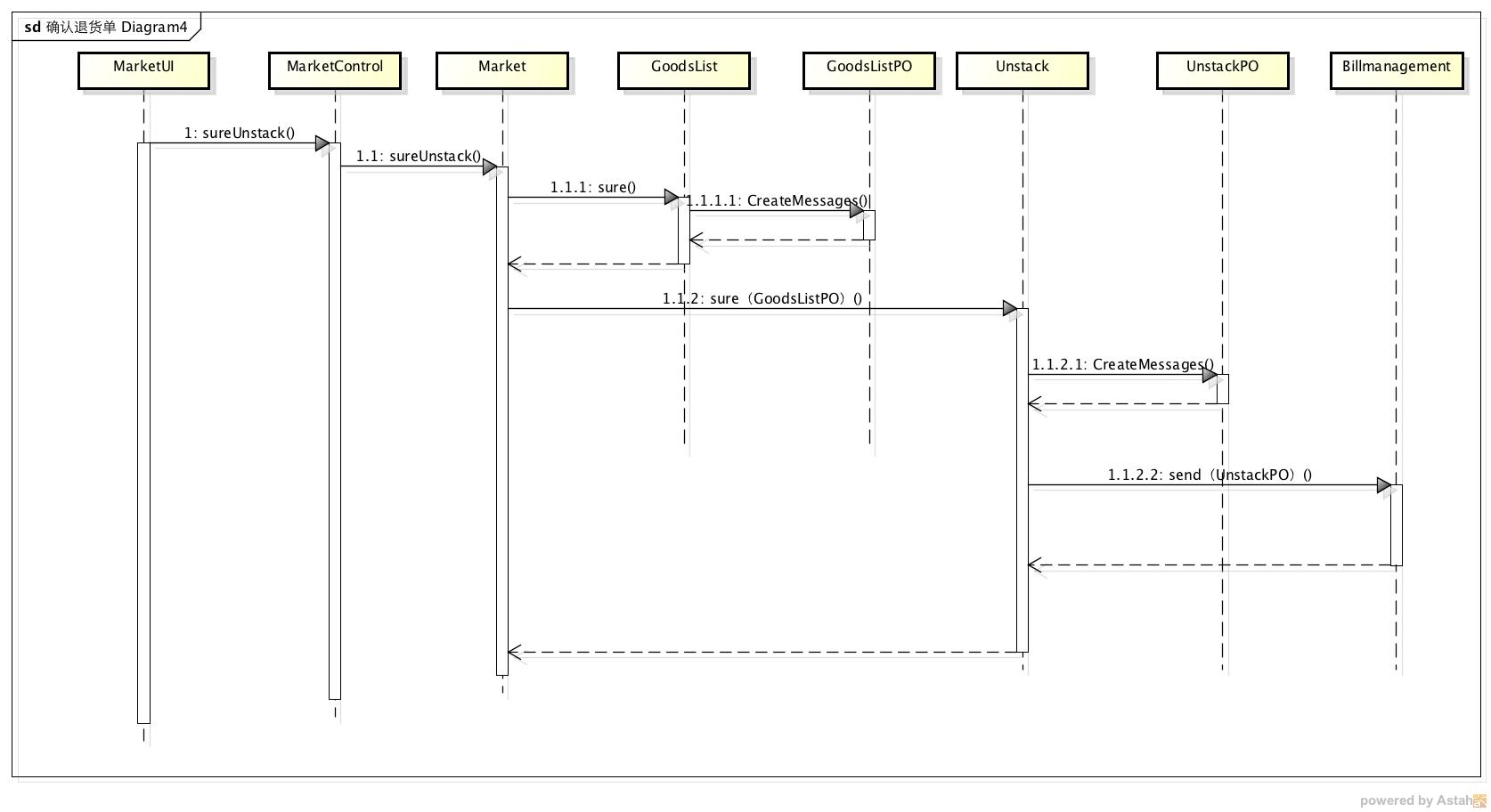
在销售界面查看赠品清单的顺序图



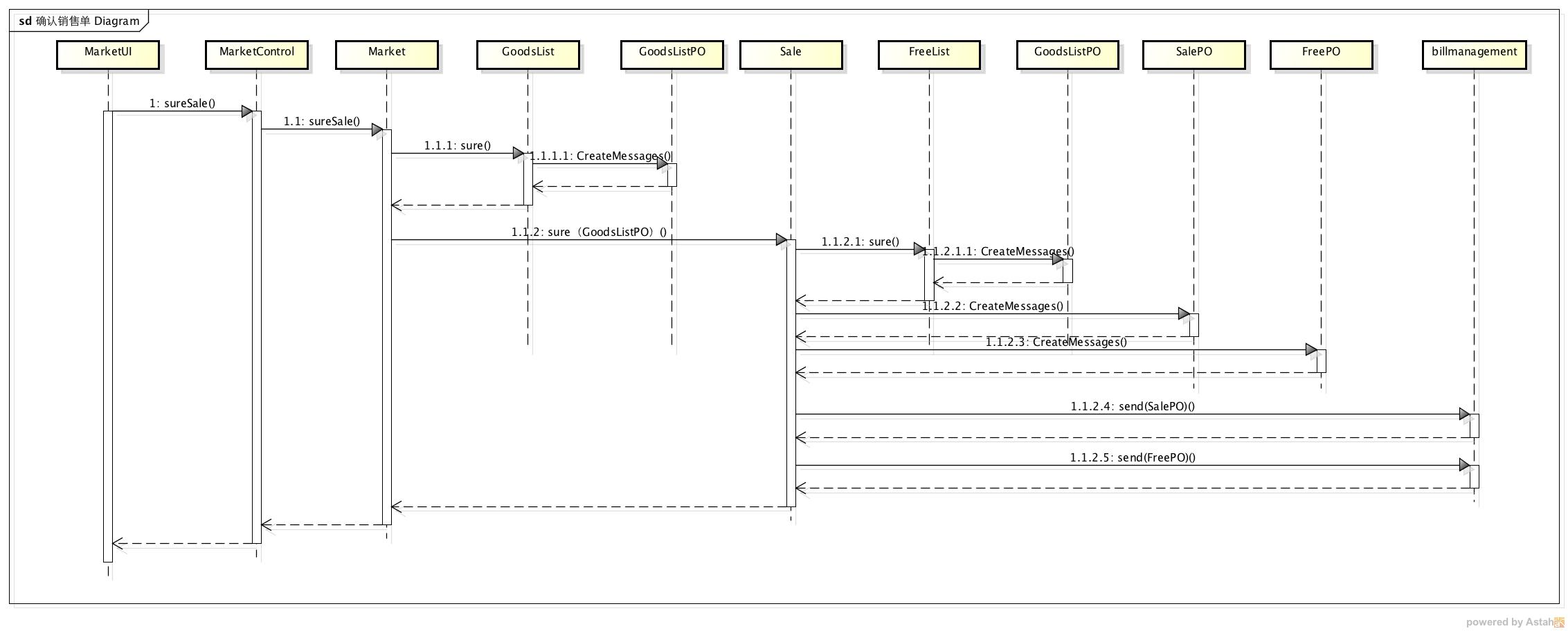
确认完成进货单的顺序图



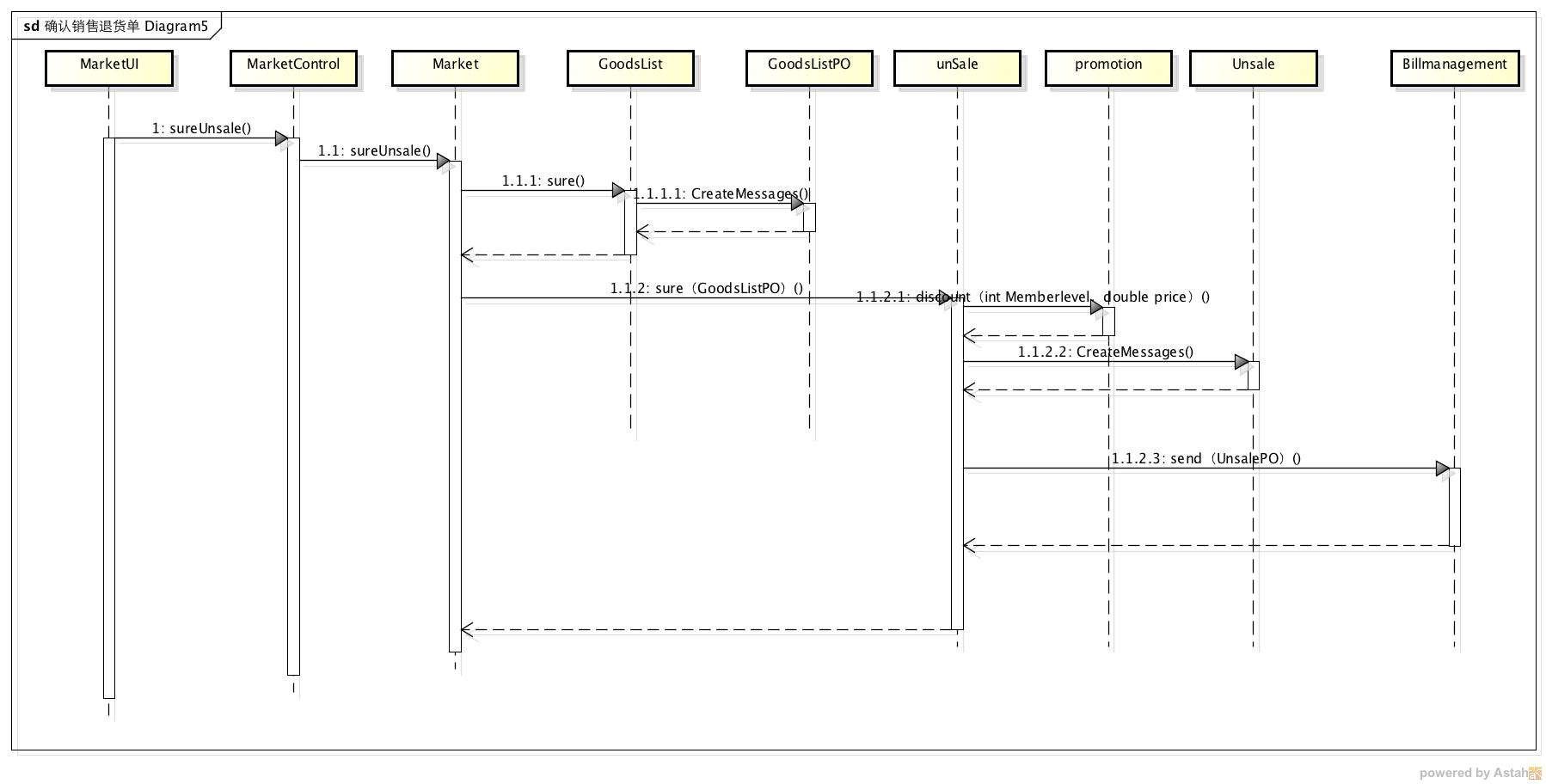
确认完成退货单的顺序图



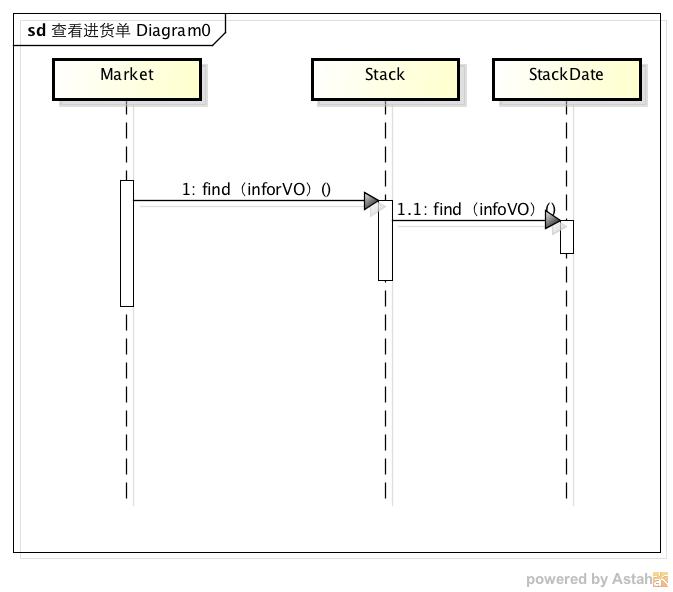
确认完成销售单的顺序图



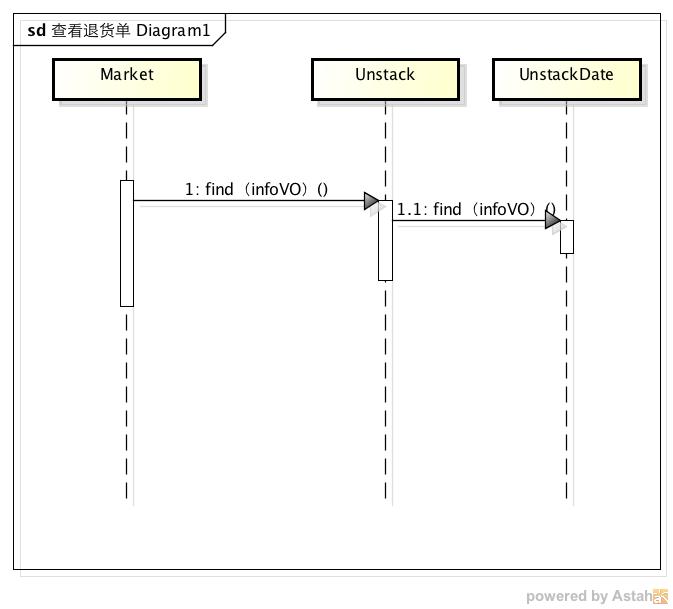
确认完成销售退货单的顺序图



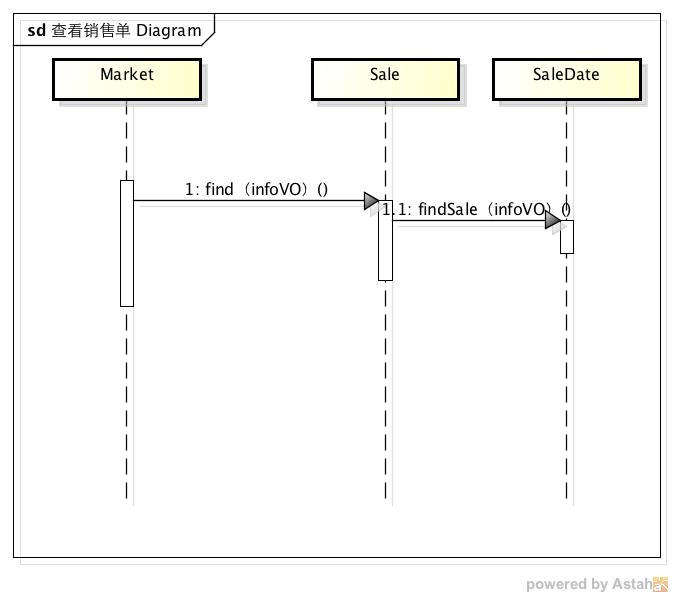
提供财务人员查看进货单的顺序图



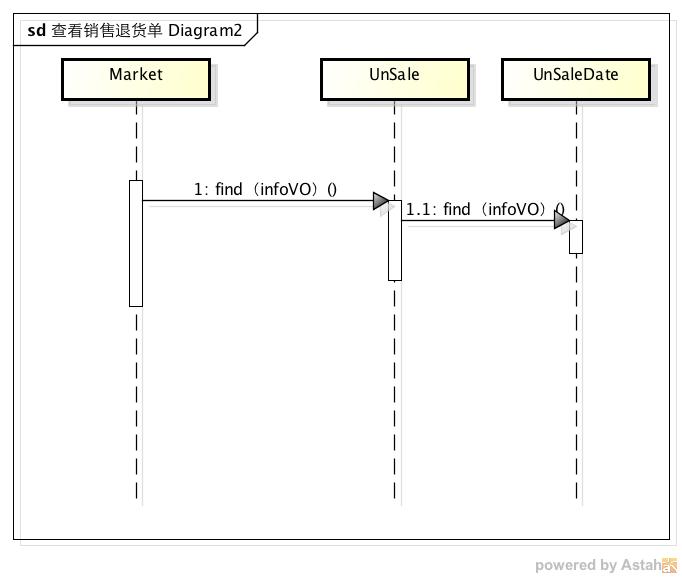
提供财务人员查看退货单的顺序图



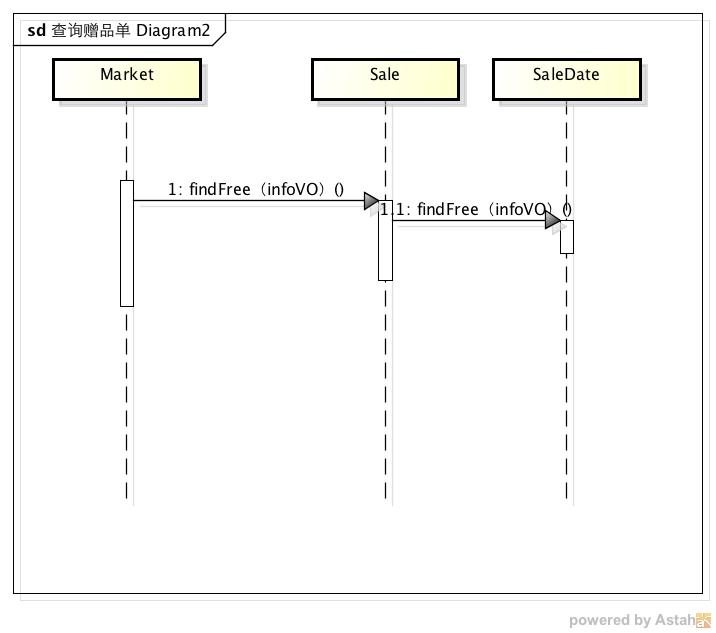
提供财务人员查看销售单的顺序图



提供财务人员查看销售退货单的顺序图

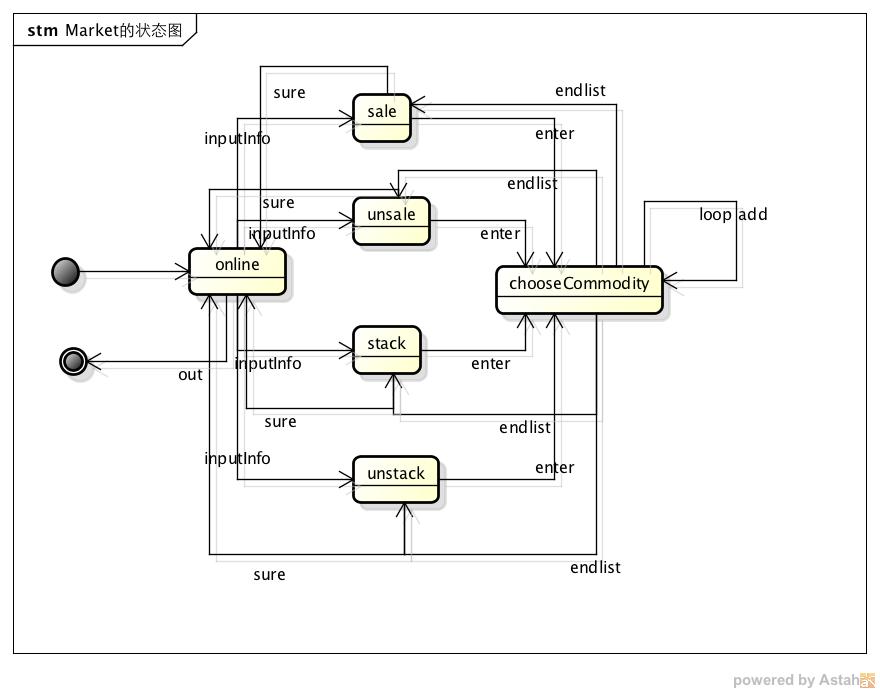


提供财务人员查看赠品单的顺序图

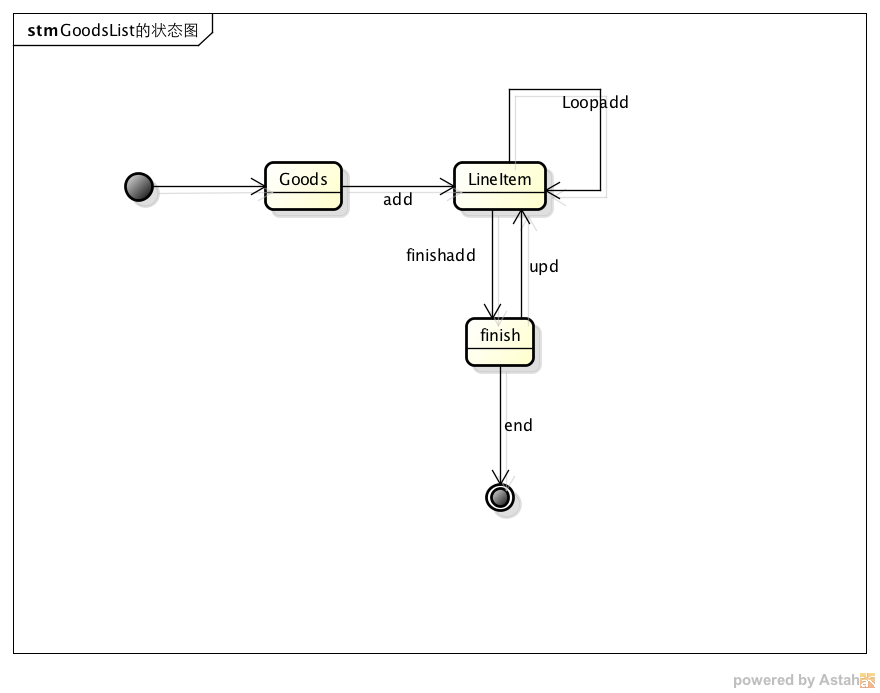


状态图：

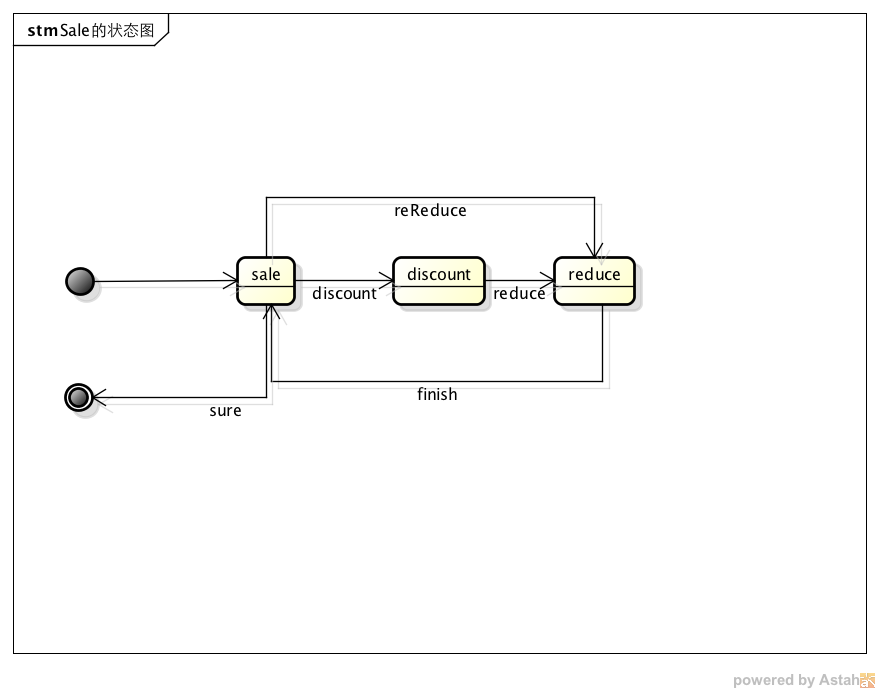
Market对象的状态图



GoodsList对象的状态图



Sale对象的状态图



### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

## 4.1.6Accountbl模块

### （1）模块概述

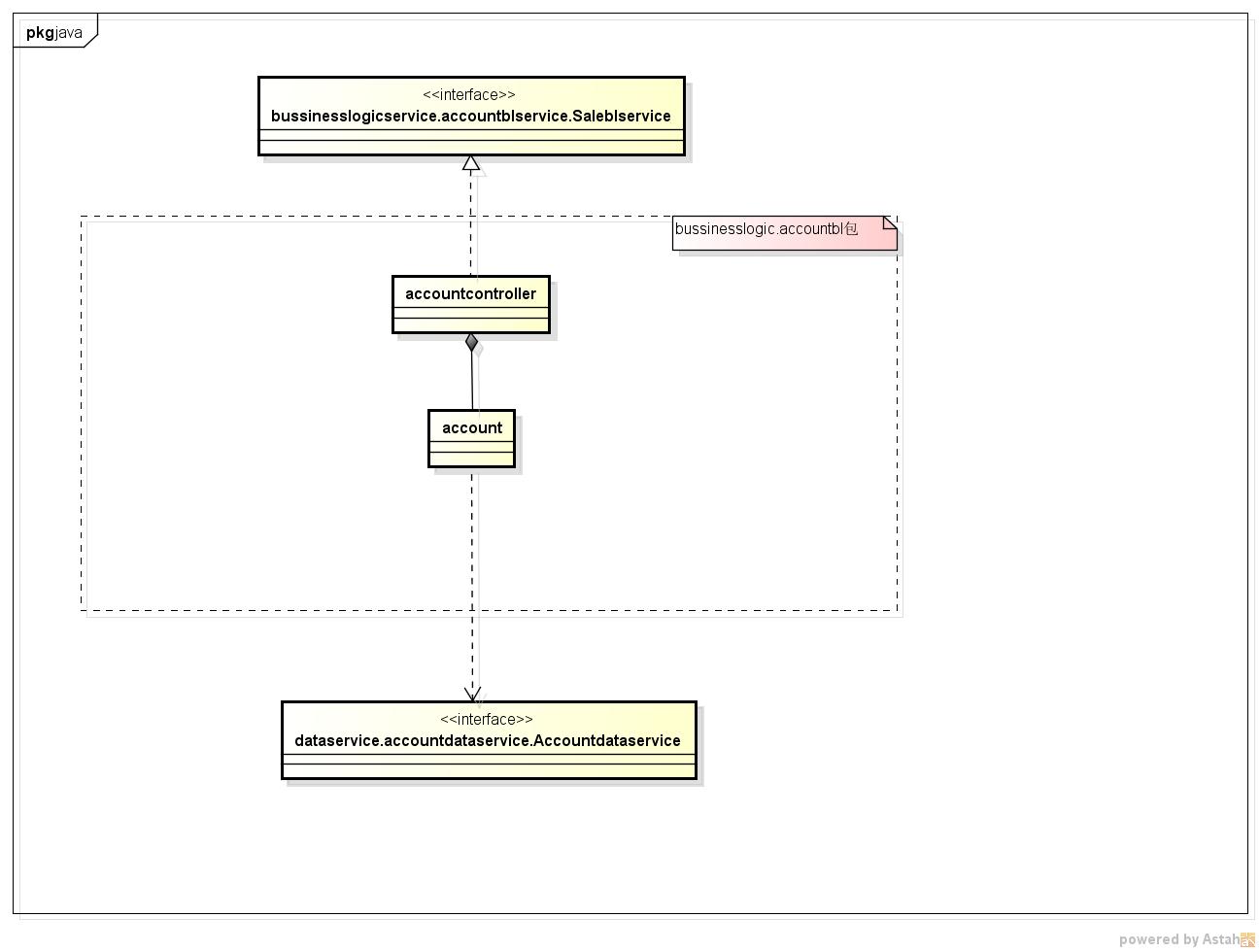
Accountbl模块承担的需求参见需求规格说明文档的功能需求级相应非功能需求

Accountbl模块的职责及接口参见软件体系结构文档

### （2）整体结构

根据体系结构的设计，我们将系统分为展示层，逻辑层与数据层，每一层之间添加接口，在展示层与逻辑层间我们添加bussinesslogic.accountblservice.accountblservice接口，逻辑层与数据层之间我们添加dataservice.accountdataservice.accountdataservice接口为了隔离业务逻辑职责与逻辑控制职责，我们增加了accountcontroller，这样accountcontroller会将账户管理的业务逻辑委托给account对象处理。

Accountbl模块的设计如图所示



Accountbl各个类的职责如图

|  |  |
| --- | --- |
| 模块 | 职责 |
| Accountcontroller | 负责实现账户管理界面所需要的服务 |
| Account | 账户管理的领域模型对象，负责账户的增删改查，完成账户管理界面所需服务 |

### （3）模块内接口规范

Accountcontroller与account的接口规范如下

Accountcontroller接口规范

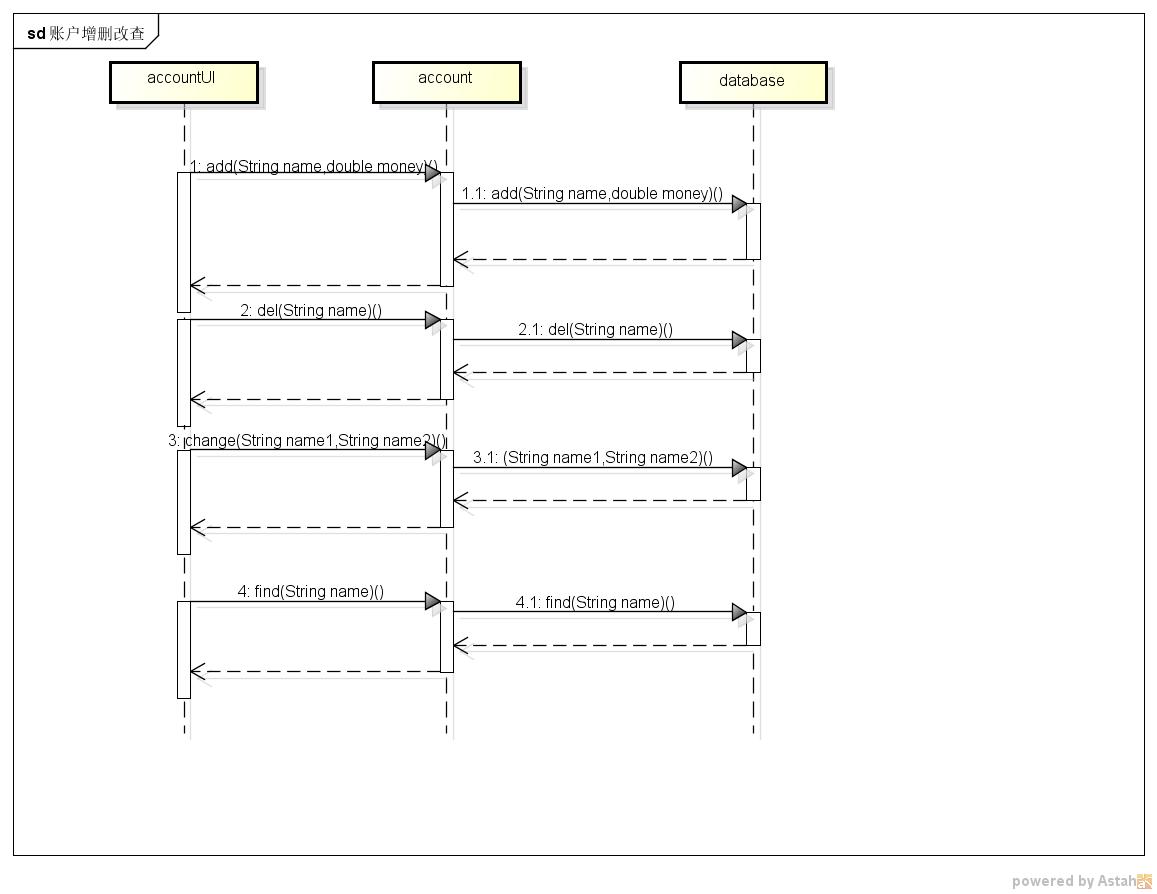
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Accountcontroller.add | 语法 | Public int add(String name,double money) |
| 前置条件 | 已创建account对象且输入符合规则 |
| 后置条件 | 调用account对象的add方法 |
| Accountcontroller.del | 语法 | Public int del(String name) |
| 前置条件 | 已创建account对象且账户存在 |
| 后置条件 | 调用account对象的del方法 |
| Accountcontroller.change | 语法 | Public int change(String name1, String name2) |
| 前置条件 | 已创建account对象且存在名为name1的账户 |
| 后置条件 | 调用account对象的change方法 |
| Accountcontroller.find | 语法 | Public String find( String name) |
| 前置条件 | 已创建account对象且账户存在 |
| 后置条件 | 调用account的find方法 |
| Accountcontroller.change2 | 语法 | Public int change2(String name, double money) |
| 前置条件 | 已创建account对象且存在名为name的账户 |
| 后置条件 | 调用account对象的change2方法 |
| 需要的服务（需接口） | | |
| Account.add(String name,double money) | 增加一个账户 | |
| Account.del(String name) | 删除一个账户 | |
| Account.change(String name1, String name2) | 修改一个账户的名字 | |
| Account.find( String name) | 返回一个账户信息 | |
| Account.change2(String name,double money) | 修改一个账户余额 | |

Account接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| account.add | 语法 | Public int add(String name,double money) |
| 前置条件 | 无同名账户 |
| 后置条件 | 增加一个账户 |
| account.del | 语法 | Public int del(String name) |
| 前置条件 | 有此账户 |
| 后置条件 | 删除此账户 |
| account.change | 语法 | Public int change(String name1, String name2) |
| 前置条件 | 存在名为name1的账户 |
| 后置条件 | 修改此账户名字 |
| account.find | 语法 | Public String find( String name) |
| 前置条件 | 无 |
| 后置条件 | 返回账户信息 |
| Account.change2 | 语法 | Public int change2(String name,double money) |
| 前置条件 | 存在name的账户 |
| 后置条件 | 修改账户余额 |
| 需要的服务（需接口） | | |
| Accountdataservice.add(String name,double money) | 增加一个账户 | |
| Accountdataservice.del(String name) | 删除一个账户 | |
| Accountdataservice.change(String name1, String name2) | 修改一个账户的名字 | |
| Accountdataservice.find( String name) | 返回一个账户信息 | |
| Accountdataservice.change2(String name1, double money) | 修改一个账户的余额 | |

### （4）业务逻辑层动态模型

下图表示了账户增删改查时业务逻辑层相关对象的协作



### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑有各自的控制器委托给不同的领域对象。

## 4.1.7checkbl模块

### （1）模块概述

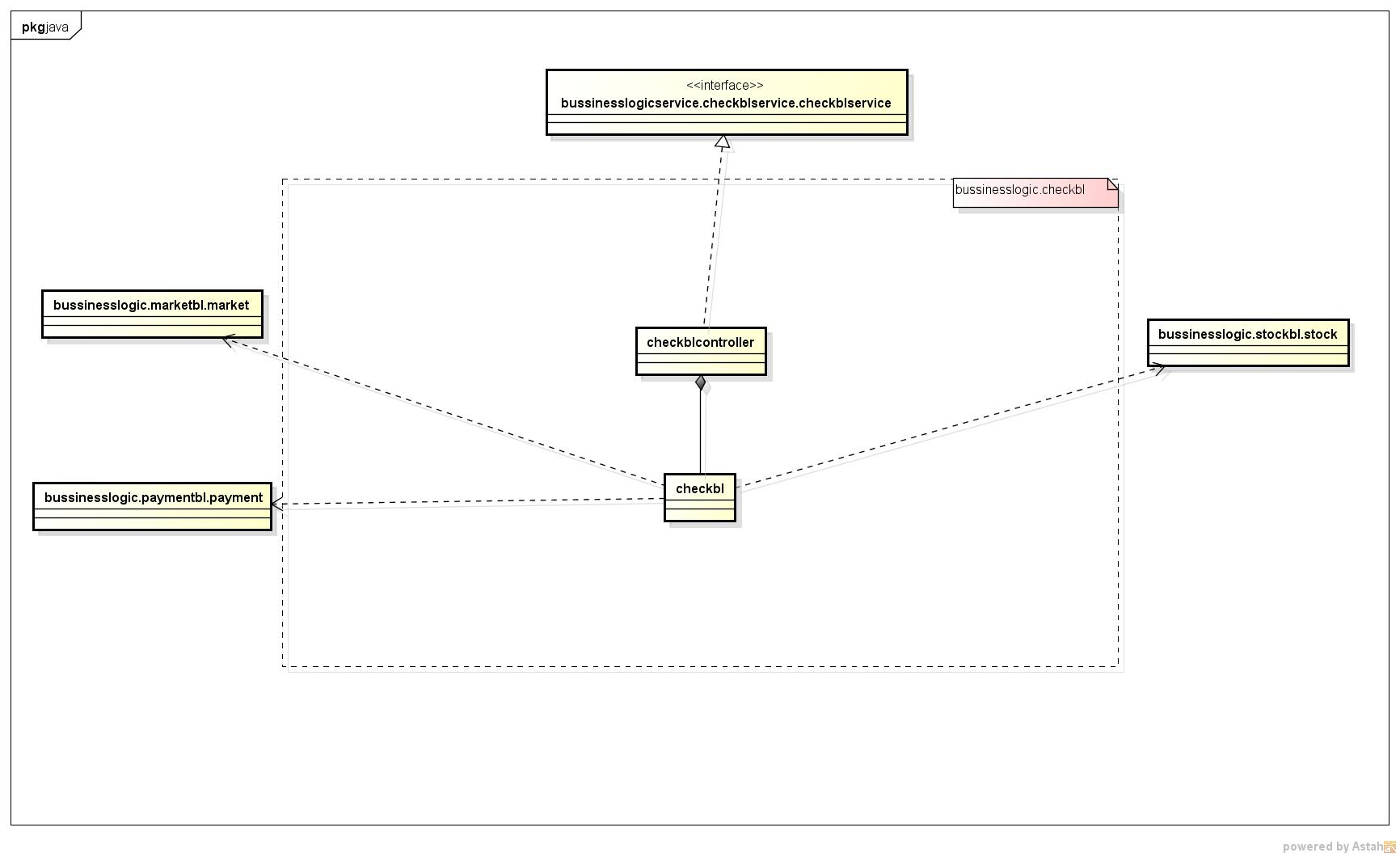
Checkbl模块承担的需求参见需求规格说明文档的功能需求级相应非功能需求

Checkbl模块的职责及接口参见软件体系结构文档

### （2）整体结构

根据体系结构的设计，我们将系统分为展示层，逻辑层与数据层，每一层之间添加接口，在展示层与逻辑层间我们添加bussinesslogic.checkblservice.checkblservice接口，为了隔离业务逻辑职责与逻辑控制职责，我们增加了checkcontroller，这样checkcontroller会将账户管理的业务逻辑委托给check对象处理。Check类需要调用其他三个类的方法

Checkbl模块的设计如图所示



Checkbl模块各个类职责如下

|  |  |
| --- | --- |
| 模块 | 职责 |
| Checkcontroller | 负责实现查表界面所需要的服务 |
| Check | 查表的领域模型对象，完成查表界面所需服务 |

### （3）模块内接口规范

Checkcontroller接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Checkcontroller.checksales | 语法 | public ArrayList<SaleListPO> checksales(infoVO infoVO) |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| Checkcontroller.checkunsales | 语法 | public ArrayList<UnsaleListPO> checkunsales(infoVO infoVO); |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| Checkcontroller.checkstack | 语法 | public ArrayList<StackListPO> checkstack(infoVO infoVO); |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| Checkcontroller.checkunstack | 语法 | public ArrayList<UnstackListPO> checkunstack(infoVO infoVO); |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| Checkcontroller.checkstock | 语法 | Public ArrayList<BillPO> checkstock(infoVO infoVO); |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| Checkcontroller.checkgift | 语法 | public ArrayList<GiftListPO> checkgift(infoVO infoVO); |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| Checkcontroller.checkpayment | 语法 | public ArrayList<BillPO> checkpayment(infoVO infoVO); |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| Checkcontroller.checksituation | 语法 | public ArrayList<String> checksituation(infoVO infoVO); |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| Checkcontroller.checkgoods | 语法 | public ArrayList<String> checkgoods(infoVO infoVO); |
| 前置条件 | 已创建check对象 |
| 后置条件 | 调用check的对应方法 |
| 需要的服务（需接口） | | |
| Check.checksales | | 找到销售单 |
| Check.checkunsales | | 找到销售退货单 |
| Check.checkstack | | 找到进货单 |
| Check.checkunstack | | 找到退货单 |
| Check.checkstock | | 找到库存类单据 |
| Check.checkgift | | 找到赠品单 |
| Check.checkpayment | | 找到财务类单据 |
| Check.checksituation | | 得到收入支出利润折让 |
| Check.checkgoods | | 得到销售记录 |

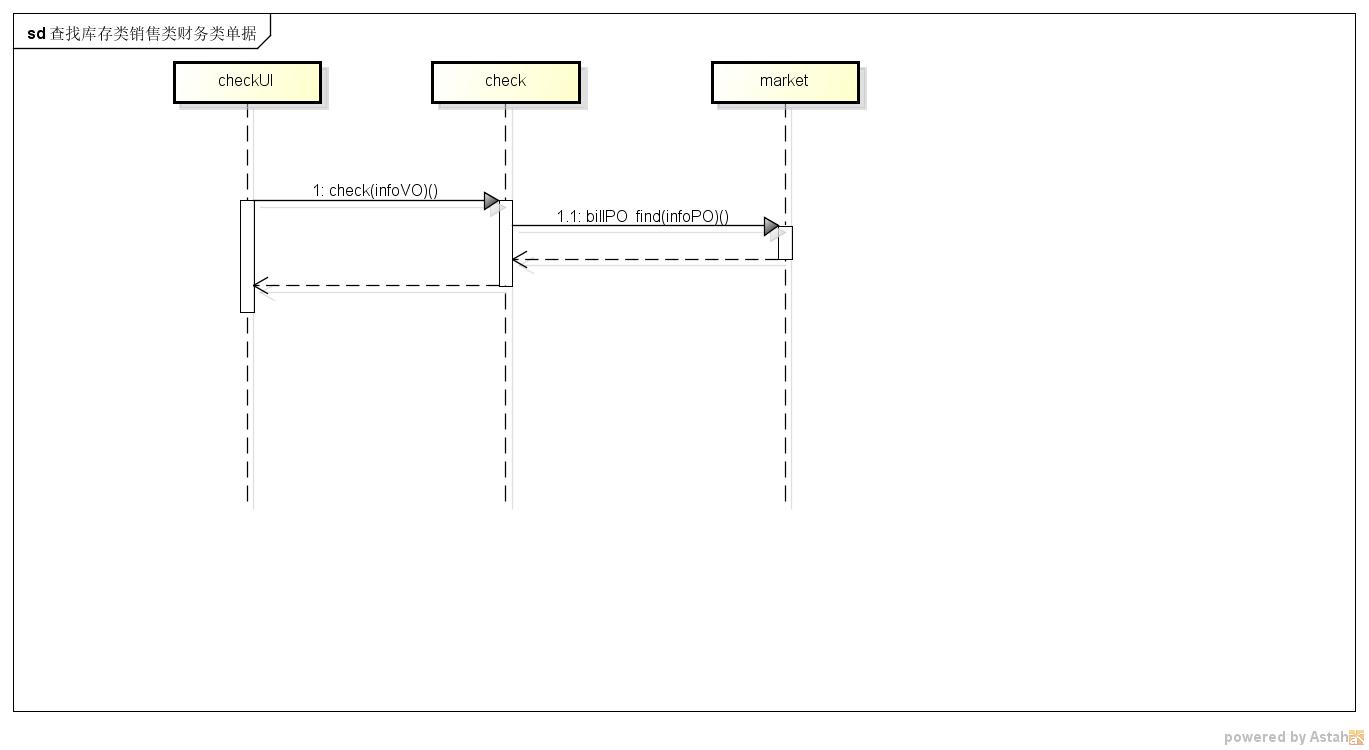
Check接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| check.checksales | 语法 | Public ArrayList<String> checksales(infoVo infoVo) |
| 前置条件 | 无 |
| 后置条件 | 返回销售类单据 |
| check.checkprogress | 语法 | Public ArrayList<String> checkprogress(infoVo infoVo) |
| 前置条件 | 无 |
| 后置条件 | 返回销售类，财务类及库存类单据 |
| check.checksituition | 语法 | Public ArrayList<String> checksituition(infoVo infoVo) |
| 前置条件 | 无 |
| 后置条件 | 返回收入支持及利润 |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| check.checksales | 语法 | public ArrayList<SaleListPO> checksales(infoVO infoVO) |
| 前置条件 | 无 |
| 后置条件 | 返回所查销售单据信息 |
| Check.checkunsales | 语法 | public ArrayList<UnsaleListPO> checkunsales(infoVO infoVO); |
| 前置条件 | 无 |
| 后置条件 | 返回满足条件的销售退货单据 |
| Check.checkstack | 语法 | public ArrayList<StackListPO> checkstack(infoVO infoVO); |
| 前置条件 | 无 |
| 后置条件 | 返回进货单 |
| Check.checkunstack | 语法 | public ArrayList<UnstackListPO> checkunstack(infoVO infoVO); |
| 前置条件 | 无 |
| 后置条件 | 返回退货单 |
| Check.checkstock | 语法 | Public ArrayList<BillPO> checkstock(infoVO infoVO); |
| 前置条件 | 无 |
| 后置条件 | 返回库存类单据 |
| Check.checkgift | 语法 | public ArrayList<GiftListPO> checkgift(infoVO infoVO); |
| 前置条件 | 无 |
| 后置条件 | 返回赠品单据 |
| Check.checkpayment | 语法 | public ArrayList<BillPO> checkpayment(infoVO infoVO); |
| 前置条件 | 无 |
| 后置条件 | 返回财务类单据单据 |
| Check.checkgoods | 语法 | public ArrayList<String> checksituation(infoVO infoVO); |
| 前置条件 | 无 |
| 后置条件 | 返回商品销售记录 |
| Check.checksituation | 语法 | public ArrayList<String> checkgoods(infoVO infoVO); |
| 前置条件 | 无 |
| 后置条件 | 返回收入，支出，利润，折扣 |
| 需要的服务（需接口） | | |
| Market.findsales | | 找到销售单 |
| Market.findunsales | | 找到销售退货单 |
| Market.findstack | | 找到进货单 |
| Market.findunstack | | 找到退货单 |
| Stock.getbills | | 找到库存类单据 |
| Market.findgift | | 找到赠品单 |
| Payment.find | | 找到财务类单据 |

### （4）业务逻辑层动态模型

下图显示了查询销售类单据时业务逻辑层各个相关对象的协作



查找库存类单据与财务类单据时的情况与上图相似

### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑有各自的控制器委托给不同的领域对象。

## 4.1.8paymentbl模块

### （1）模块概述

Paymentbl模块承担的需求参见需求规格说明文档的功能需求级相应非功能需求

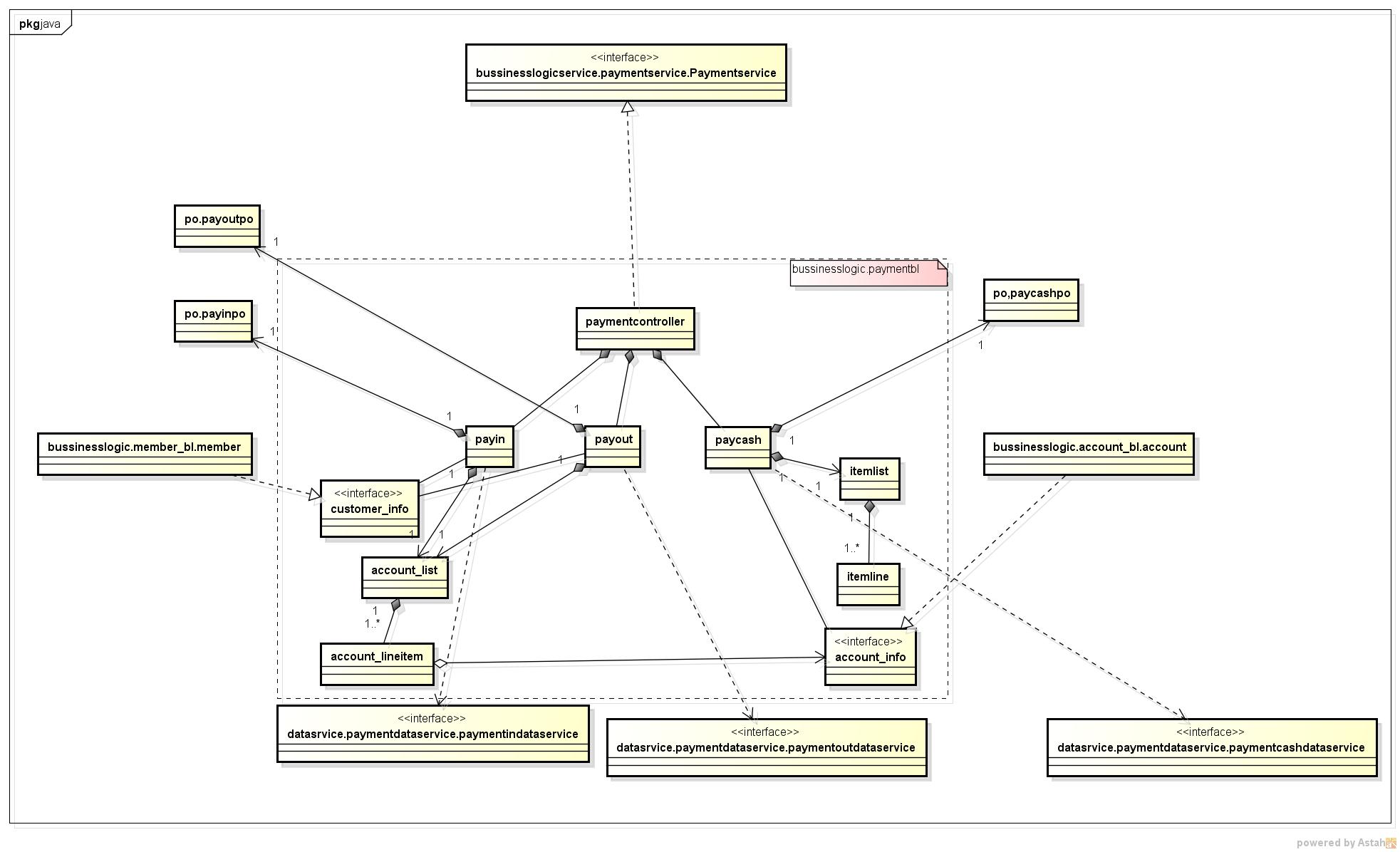
Paymentbl模块的职责及接口参见软件体系结构文档

### （2）整体结构

根据体系结构的设计，我们将系统分为展示层，逻辑层与数据层，每一层之间添加接口，在展示层与逻辑层间我们添加bussinesslogic.paymentblservice.paymentblservice接口，逻辑层与数据层之间我们添加dataservice.paymentdataservice.paymentdataservice接口为了隔离业务逻辑职责与逻辑控制职责，我们增加了paymentcontroller，这样paymentcontroller会将账户管理的业务逻辑委托给payin,payout,和paycash对象处理。Itemlist及account\_list是保存条目列表与转账列表的数据的容器，收款单和付款单均要客户信息，所以添加了memberinfo接口，

收款付款与现金支付均要账户信息所以加入了accountinfo接口，payinpo，payoutpo与paycashpo是作为收款付款现金支付单据的持久化对象保存的。

Paymentbl模块的设计如图所示



Paymentbl模块各个类的职责如图

|  |  |
| --- | --- |
| 模块 | 职责 |
| Paymentcontroller | 负责实现收款付款现金支付界面所需服务 |
| Payin | 负责完成收款任务，持有一次收款所需数据，并有查找单据功能 |
| Payout | 负责完成付款任务，持有一次付款所需数据，并有查找单据功能 |
| Paycash | 负责完成现金支付任务，持有一次现金支付所需数据，并有查找单据功能 |

### （3）模块内部类的接口规范

Paymentcontroller接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Paymentcontroller.addaccountlist | 语法 | public int addaccountlist(accountlineitemVO v); |
| 前置条件 | 已创建payin或payout对象 |
| 后置条件 | 调用对象的对应方法 |
| Paymentcontroller.addcashlist | 语法 | public int addcashlist(cashlineitemVO v) |
| 前置条件 | 已创建paycash对象 |
| 后置条件 | 调用对象的对应方法 |
| Paymentcontroller.addaccountr | 语法 | public int addaccount(String s); |
| 前置条件 | 已创建paycash对象 |
| 后置条件 | 调用对象的对应方法 |
| Paymentcontroller.addcustomer | 语法 | public int addcustomer(String s); |
| 前置条件 | 已创建payin或payout对象 |
| 后置条件 | 调用对象的对应方法 |
| Paymentcontroller.delaccountlist | 语法 | public int delaccountlist(accountlineitemVO v); |
| 前置条件 | 已创建payin或payout对象 |
| 后置条件 | 调用对象的对应方法 |
| Paymentcontroller.delcsahlist | 语法 | public int delcashlist(cashlineitemVO v); |
| 前置条件 | 已创建paycash对象 |
| 后置条件 | 调用对象的对应方法 |
| Paymentcontroller.submit | 语法 | public int submit(String oper); |
| 前置条件 | 已创建payin或payout或paycash对象 |
| 后置条件 | 调用对象的对应方法 |
| Paymentcontroller.refresh | 语法 | public ArrayList<ArrayList<BillPO>> refresh(); |
| 前置条件 | 已创建payin，payout，paycash对象 |
| 后置条件 | 调用三个对象的对应方法 |
| Paymentcontroller.store | 语法 | public void store(BillPO bp,int i) |
| 前置条件 | 已创建payin或payout或paycash对象 |
| 后置条件 | 调用对象的对应方法 |
| Paymentcontroller.find | 语法 | public ArrayList<BillPO> find(infoVO v) |
| 前置条件 | 已创建payin，payout，paycash对象 |
| 后置条件 | 调用三个对象的对应方法 |
| 需要的服务（需接口） | | |
| Payin.find | 找到对应收款单 | |
| Payout.find | 找到对应付款单 | |
| Paycash.find | 找到对应现金支付单 | |
| Payin.store | 存储收款单 | |
| Payout.store | 存储付款单 | |
| Paycash.store | 存储现金支付单 | |
| Payin.refresh | 返回已经过审批收款单据 | |
| Payout.refresh | 返回已经过审批付款单据 | |
| Paycash.refresh | 返回已经过审批现金付款单据 | |
| Payin.submit | 提交收款单 | |
| Payout.submit | 提交付款单 | |
| Paycash.submit | 提交现金支付单 | |
| Payin.addaccountlist | 在收款单添加转账列表 | |
| Payout.addaccountlist | 在付款单添加转账列表 | |
| Paycash.addcashlist | 在现金付款单添加条目列表 | |
| Payin.delaccountlist | 在收款单删除转账列表 | |
| Payout.delaccountlist | 在付款单删除转账列表 | |
| Paycash.delcashlist | 在现金付款单删除条目列表 | |
| Payin.addcustomer | 在收款单添加客户 | |
| Payout.addcustomer | 在付款单添加客户 | |
| Paycash.addaccount | 在现金付款单添加账户 | |

Payin接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Payinaddaccountlist | 语法 | public int addaccountlist(accountlineitemVO v); |
| 前置条件 | 无 |
| 后置条件 | 添加一个转账列表项 |
| Payin.addcustomer | 语法 | public int addcustomer(String s); |
| 前置条件 | 无 |
| 后置条件 | 添加一个客户 |
| Payin.delaccountlist | 语法 | public int delaccountlist(accountlineitemVO v); |
| 前置条件 | 无 |
| 后置条件 | 删除一个转账列表项 |
| Payin.submit | 语法 | public int submit(String oper); |
| 前置条件 | 完成单据 |
| 后置条件 | 提交总经理 |
| Payin.refresh | 语法 | public ArrayList<ArrayList<PayinPO>> refresh(); |
| 前置条件 | 总经理完成审批 |
| 后置条件 | 查看送审收款单据情况 |
| Payin.store | 语法 | public void store(PayinPO bp,int i) |
| 前置条件 | 总经理完成审批 |
| 后置条件 | 存储单据，更改对应账户客户信息 |
| Payin.find | 语法 | public ArrayList<PayinPO> find(infoVO v) |
| 前置条件 | 无 |
| 后置条件 | 返回收款单据 |
| 需要的服务（需接口） | | |
| PaymentInDataservice.find | 找到对应收款单 | |
| Account.change2 | 修改账户余额信息 | |
| PaymentInDataservice.store | 存储收款单 | |
| Member.paymoney | 更改客户信息 | |
| PaymentInDataservice.refresh | 返回已经过审批收款单据 | |

Payout接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Payout.addaccountlist | 语法 | public int addaccountlist(accountlineitemVO v); |
| 前置条件 | 无 |
| 后置条件 | 添加一个转账列表项 |
| Payout.addcustomer | 语法 | public int addcustomer(String s); |
| 前置条件 | 无 |
| 后置条件 | 添加一个客户 |
| Payout.delaccountlist | 语法 | public int delaccountlist(accountlineitemVO v); |
| 前置条件 | 无 |
| 后置条件 | 删除一个转账列表项 |
| Payout.submit | 语法 | public int submit(String oper); |
| 前置条件 | 完成单据 |
| 后置条件 | 提交总经理 |
| Payout.refresh | 语法 | public ArrayList<ArrayList<PayoutPO>> refresh(); |
| 前置条件 | 总经理完成审批 |
| 后置条件 | 查看送审付款单据情况 |
| Payout.store | 语法 | public void store(PayoutPO bp,int i) |
| 前置条件 | 总经理完成审批 |
| 后置条件 | 存储单据，更改对应账户客户信息 |
| Payout.find | 语法 | public ArrayList<PayoutPO> find(infoVO v) |
| 前置条件 | 无 |
| 后置条件 | 返回付款单据 |
| 需要的服务（需接口） | | |
| PaymentOutDataservice.find | 找到对应付款单 | |
| Account.change2 | 修改账户余额信息 | |
| PaymentOutDataservice.store | 存储付款单 | |
| Member.getmoney | 更改客户信息 | |
| PaymentOutDataservice.refresh | 返回已经过审批付款单据 | |

Paycash接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Paycash.addcashlist | 语法 | public int addcashlist(cashlineitemVO v) |
| 前置条件 | 无 |
| 后置条件 | 添加一个条目列表项 |
| Paycash.addaccount | 语法 | public int addaccount(String s); |
| 前置条件 | 无 |
| 后置条件 | 添加账户 |
| Paycash.delcashlist | 语法 | public int delcashlist(cashlineitemVO v); |
| 前置条件 | 无 |
| 后置条件 | 删除一个条目列表项 |
| Paycash.submit | 语法 | public int submit(String oper); |
| 前置条件 | 完成单据 |
| 后置条件 | 提交总经理 |
| Paycash.refresh | 语法 | public ArrayList<ArrayList<PaycashPO>> refresh(); |
| 前置条件 | 总经理完成审批 |
| 后置条件 | 查看送审单据情况 |
| Paycash.store | 语法 | public void store(PaycashPO bp,int i) |
| 前置条件 | 总经理完成审批 |
| 后置条件 | 存储单据，更改对应账户客户信息 |
| Paycash.find | 语法 | public ArrayList<PaycashPO> find(infoVO v) |
| 前置条件 | 无 |
| 后置条件 | 返回现金付款单据 |
| 需要的服务（需接口） | | |
| PaymentCashDataservice.find | 找到对应现金支付单 | |
| Account.change2 | 修改账户余额信息 | |
| PaymentCashDataservice.store | 存储现金支付单 | |
| PaymentCashDataservice.refresh | 返回已经过审批现金付款单据 | |

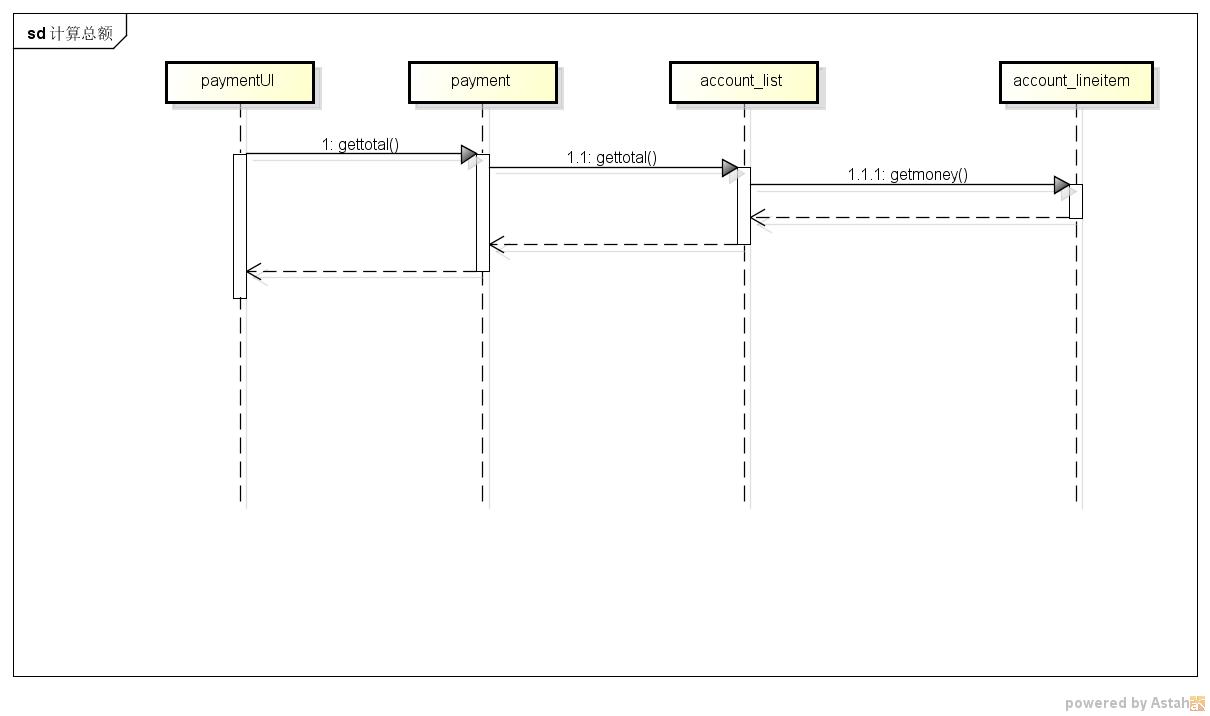
### （4）业务逻辑层动态模型

添加转账条目时的业务逻辑层相关对象协作

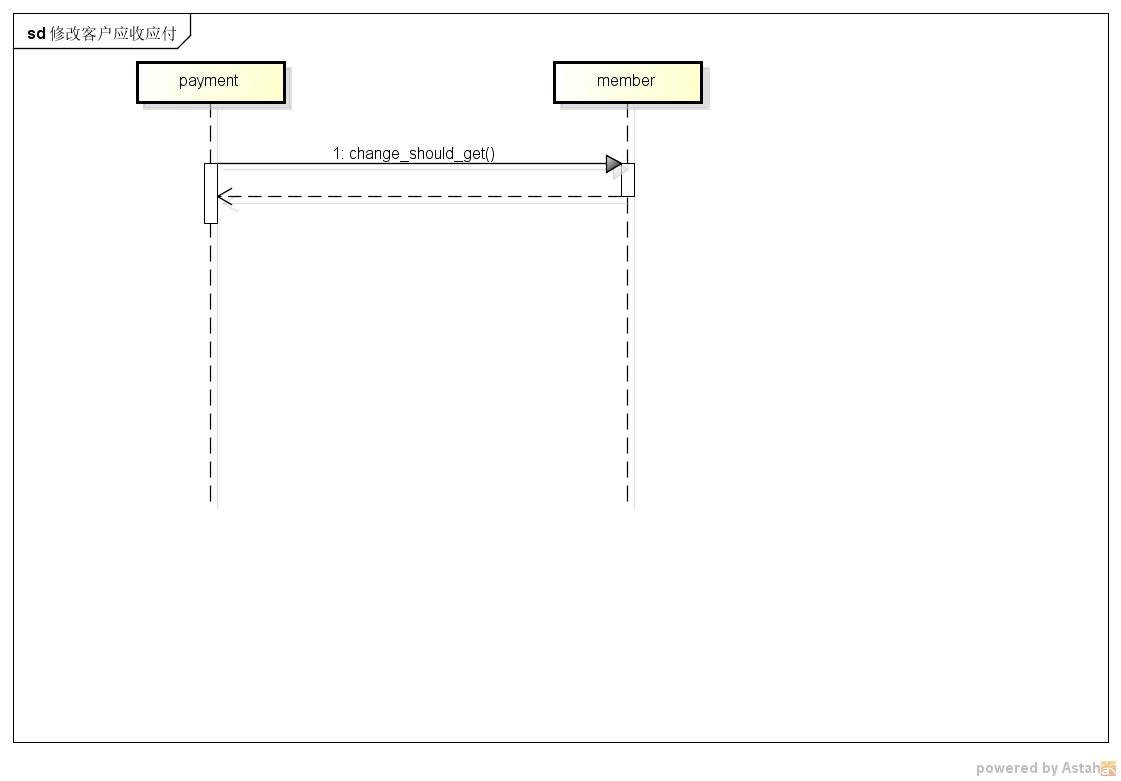


添加条目项时与此相同

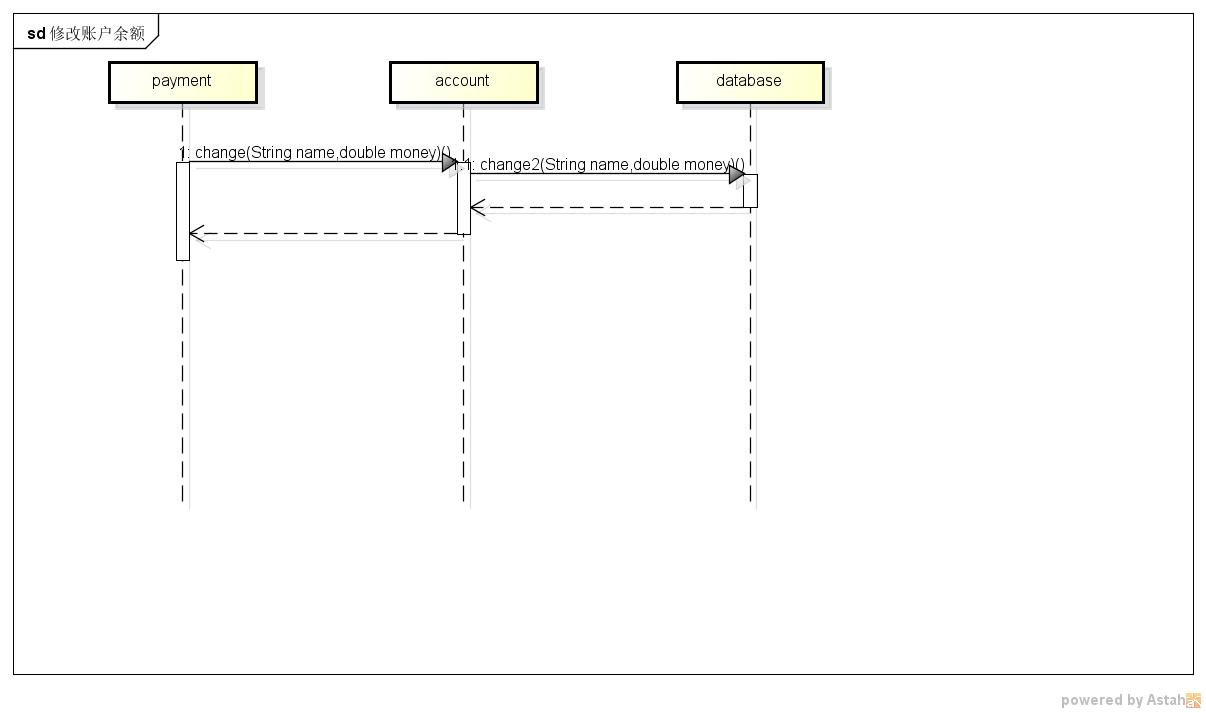
计算总额时的业务逻辑层各相关对象协作



修改客户应收应付信息时业务逻辑层相关对象协作



修改账户余额时业务逻辑层相关对象协作



### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑有各自的控制器委托给不同的领域对象。

## 4.1.9.billmanagementbl模块

### （1）模块概述

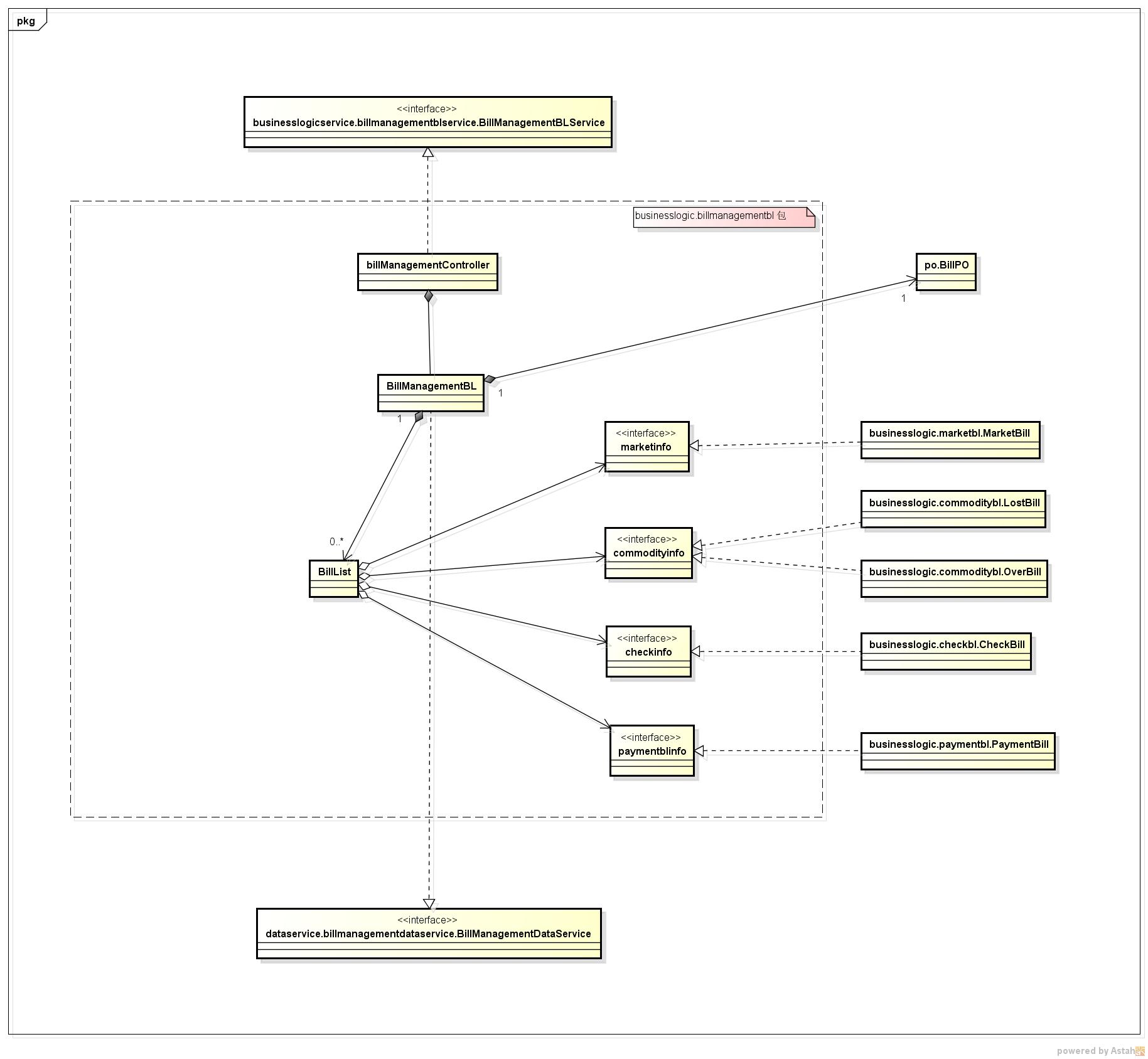
billmanagementbl模块承担的需求参见进销存系统需求规格说明文档功能需求及相关非功能需求。

billmanagementbl模块的职责及接口需求参见进销存系统软件系统结构描述文档表\*。

## （2）整体结构

依据体系结构的设计，我们将系统分为展示层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们添加了接口。例如展示层和业务逻辑层之间，我们添加了billmanagementlogicservice.billmanagementblservice.BillManagementBLService接口。为了业务逻辑职责和逻辑控制指责，我们增加了billManagementController,这样billManagementController会将对制定促销策略的业务逻辑交给BillManagementBL对象。BillPO是作为单据的持久化对象而加入设计模型中去的。BillList是marketinfo,commodityinfo,checkinfo,paymentblinfo的容器类，BillList封装了单据的数据结构和单据分类的职责。而marketinfo,commodityinfo,checkinfo,paymentblinfo是根据依赖倒置原则，为了消除循环依赖而产生的接口。

billmanagementbl模块的设计图如下图所示。



billmanagementbl模块各个类的职责如下表所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| billManagementController | 负责实现审批单据所需要的服务 |
| BillManagementBL | 审批单据的领域模型对象，拥有审批所持有的单据信息，可以帮助完成审批单据所需要的服务。 |

### （3）模块内部类的接口规范

**billmanagementController模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| billmanagementController.getBillList | 语法 | | public BillVO getBillList() |
| 前置条件 | | 已创建一个BillManagementBL领域对象 |
| 后置条件 | | 调用BillManagementBL领域对象的getBillList方法 |
| billmanagementController.approveBill | 语法 | | public String approveBill(BillVO vo) |
| 前置条件 | | 已创建一个BillManagementBL领域对象 |
| 后置条件 | | 调用BillManagementBL领域对象的approveBill方法 |
| billmanagementController.editBill | 语法 | | public String editBill(BillVO vo) |
| 前置条件 | | 已创建一个BillManagementBL领域对象 |
| 后置条件 | | 调用BillManagementBL领域对象的editBill方法 |
| billmanagementController.mutiApproveBill | 语法 | | public String mutiApproveBill(ArrayList<BillVO> vo) |
| 前置条件 | | 已创建一个BillManagementBL领域对象 |
| 后置条件 | | 调用BillManagementBL领域对象的mutiApproveBill方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| BillmanagementBL.getBillList | | 得到需审批的单据对象列表 | |
| BillmanagementBL.approveBill(BillVO vo) | | 审批单个单据 | |
| BillmanagementBL.editBill(BillVO vo) | | 编辑单据 | |
| BillmanagementBL.mutiApproveBill(ArrayList<BillVO> vo) | | 批量审批多个单据 | |

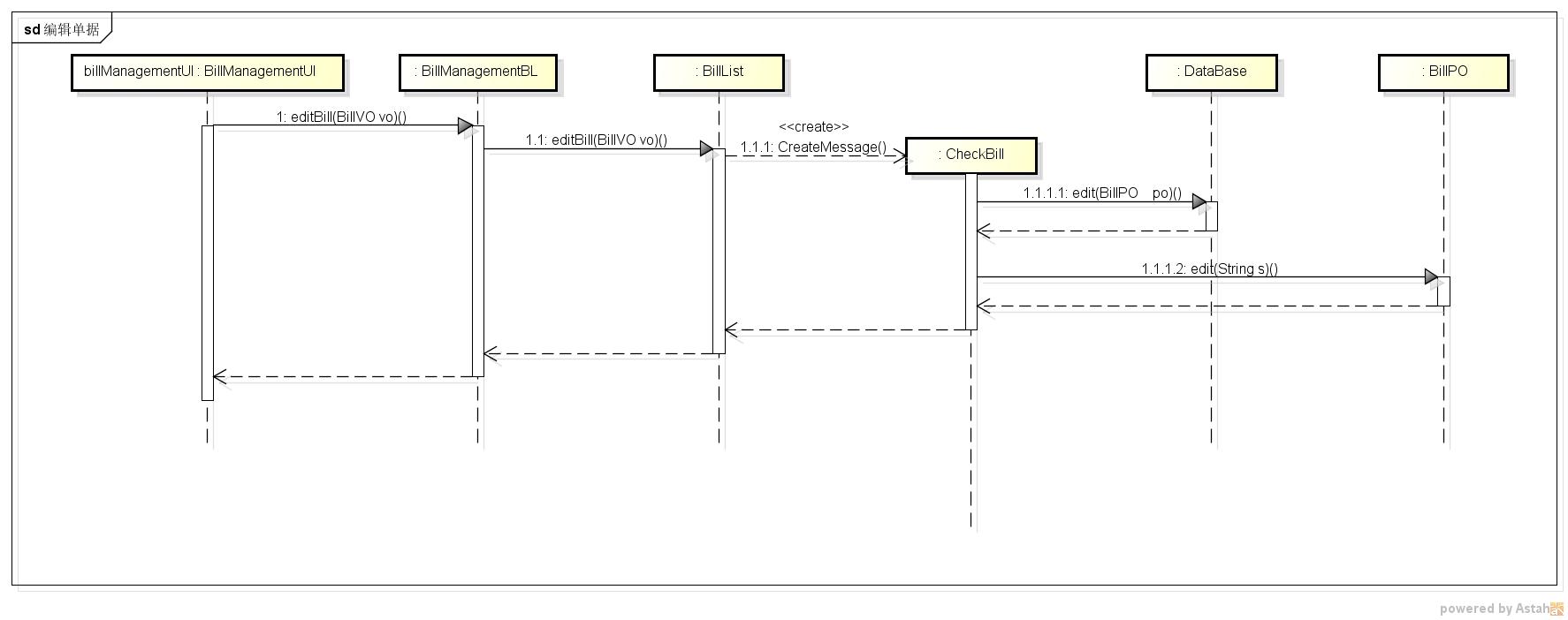
**BillmanagementBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| BillmanagementBL.getBillList | 语法 | | public BillVO getBillList() |
| 前置条件 | | 启动审批单据 |
| 后置条件 | | 开始对单据进行审批 |
| BillmanagementBL.approveBill | 语法 | | public String approveBill(BillVO vo) |
| 前置条件 | | 审阅一个单据 |
| 后置条件 | | 发送回执给单据发起方 |
| BillmanagementBL.editBill | 语法 | | public String editBill(BillVO vo) |
| 前置条件 | | 审阅一个单据 |
| 后置条件 | | 保存修改后的单据 |
| BillmanagementBL.mutiApproveBill | 语法 | | public String mutiApproveBill(ArrayList<BillVO> vo) |
| 前置条件 | | 审阅多个单据 |
| 后置条件 | | 发送回执给单据发起方 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| billManagementDataService.getBill(BillPO po) | | 得到需审批的单据对象 | |
| billManagementDataService.updateBill(BillPO po) | | 更新需审批的单据对象 | |

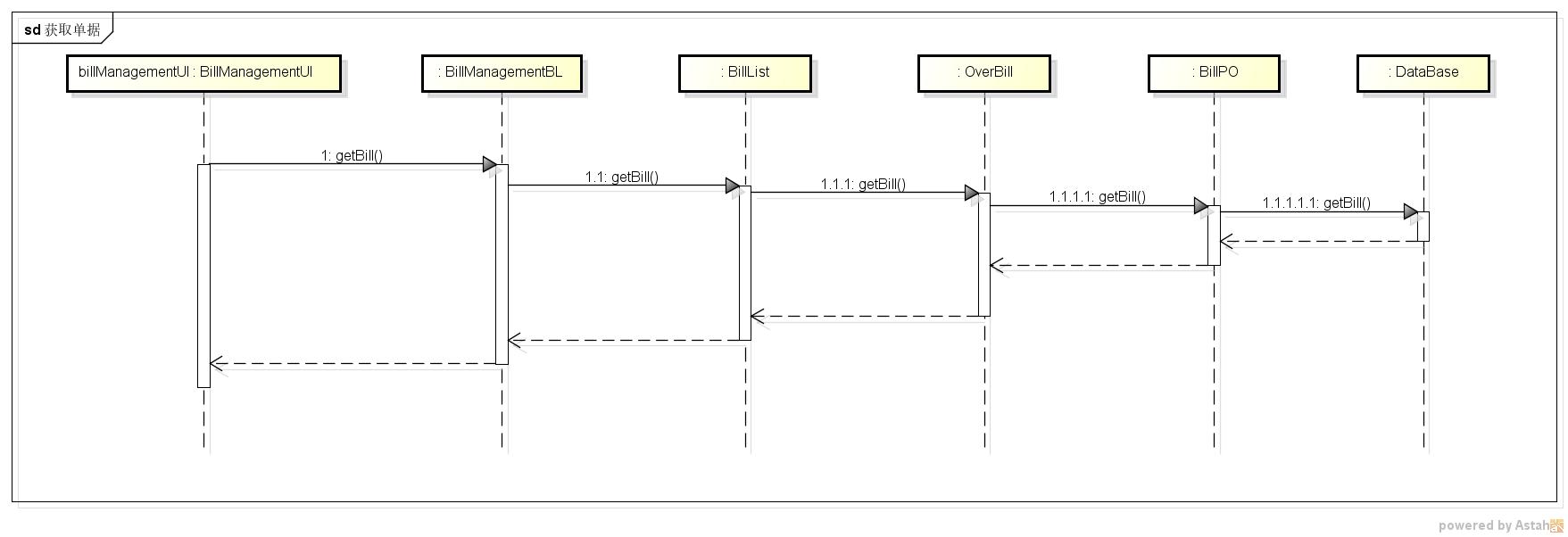
### （4）业务逻辑层的动态模型

下列图表明了在审批单据中涉及的相关对象间的协作

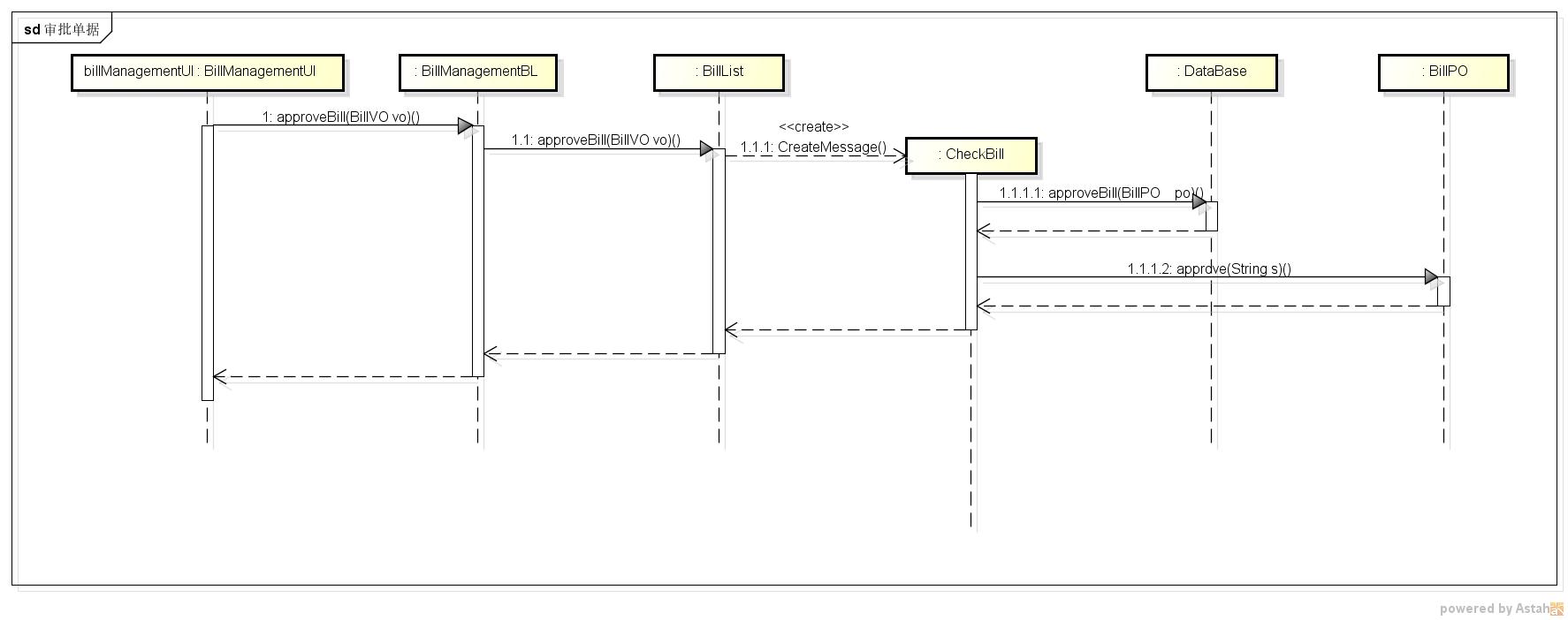
**编辑单据的顺序图**



**获取单据的顺序图**



**审批单据的顺序图**



### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自控制器委托给不同的领域对象。

## 4.1.10.promotionbl模块

**（1）**模块概述

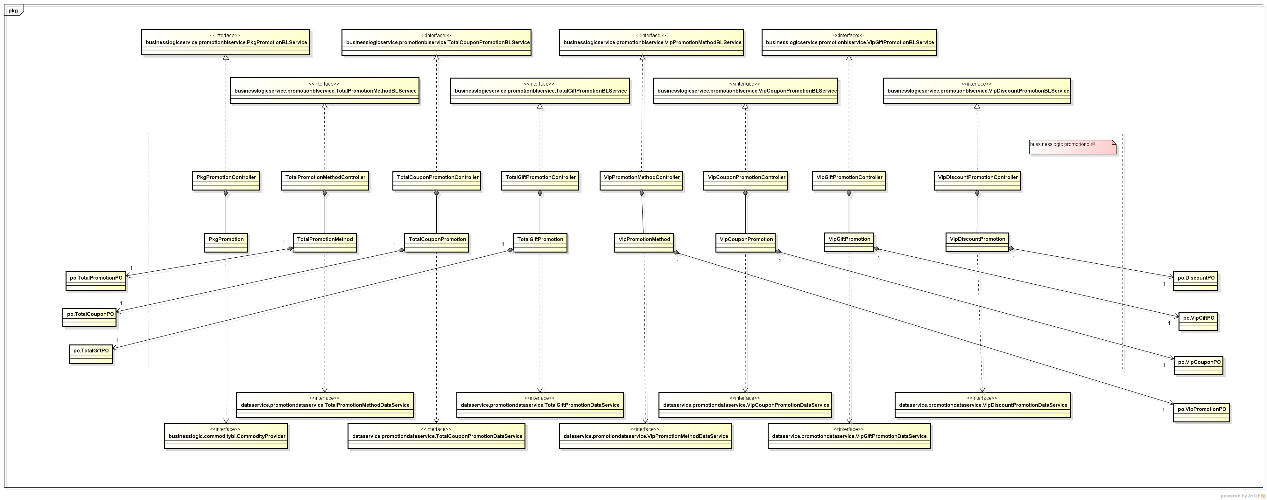
promotionbl模块承担的需求参见进销存系统需求规格说明文档功能需求及相关非功能需求。

promotionbl模块的职责及接口需求参见进销存系统软件系统结构描述文档表\*。

**（2）**整体结构

依据体系结构的设计，我们将系统分为展示层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们添加了接口。例如展示层和业务逻辑层之间，我们添加了promotionlogicservice.promotionblservice.PkgPromotionBLService和promotionlogicservice.promotionblservice.TotalPromotionMethodBLService和promotionlogicservice.promotionblservice.VipPromotionMethodBLService等一系列接口。为了业务逻辑职责和逻辑控制指责，我们增加了PkgPromotionController等一系列controller,这样controller会将对制定促销策略的业务逻辑交给PkgPromotion等对象。VipPromotionPO等po对象都是最为持久化对象而加入设计模型中去的。而businessslogic.commoditybl.CommodityProvider是同层调用的接口。

promotionbl模块的设计图如下图所示。



promotionbl模块各个类的职责如下表所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| PkgPromotionController | 负责实现制定特价包促销策略所需要的服务 |
| TotalPromotionMethodController | 负责实现制定总价额满优惠的对应具体总价额促销策略所需要的服务 |
| TotalCouponPromotionController | 负责实现制定总价额满优惠的代金券促销策略所需要的服务 |
| TotalGiftPromotionController | 负责实现制定总价额满优惠的赠品促销策略所需要的服务 |
| VipPromotionMethodController | 负责实现制定vip优惠的具体等级对应促销策略所需要的服务 |
| VipCouponPromotionController | 负责实现制定vip优惠的代金券促销策略所需要的服务 |
| VipGiftPromotionController | 负责实现制定vip优惠的赠品促销策略所需要的服务 |
| VipDiscountPromotionController | 负责实现制定vip优惠的折扣促销策略所需要的服务 |
| PkgPromotion | 特价包优惠的领域模型对象，拥有各个特价包及对应商品的引用，可以帮住完成制定特价包优惠所需要的服务。 |
| TotalPromotionMethod | 总价额满优惠的关于各个总价额的领域模型对象，拥有各个总价额项对应的促销策略，可以帮助完成制定总价额满优惠所需要的服务。 |
| TotalCouponPromotion | 总价额满优惠的关于代金券的领域模型对象，拥有各个总价额项对应的代金券促销策略，可以帮助完成制定总价额满优惠的代金券所需要的服务。 |
| TotalGiftPromotion | 总价额满优惠的关于赠品的领域模型对象，拥有各个总价额项对应的赠品促销策略，可以帮助完成制定总价额满优惠的赠品所需要的服务。 |
| VipPromotionMethod | vip优惠的关于各个vip等级的领域模型对象，拥有各个vip等级对应促销策略，可以帮助完成制定vip优惠所需要的服务。 |
| VipCouponPromotion | vip优惠的关于代金券的领域模型对象，拥有各个vip等级对应的代金券促销策略，可以帮助完成制定vip优惠的代金券所需要的服务。 |
| VipGiftPromotion | vip优惠的关于赠品的领域模型对象，拥有各个vip等级对应的赠品促销策略，可以帮助完成制定vip优惠的赠品所需要的服务。 |
| VipDiscountPromotion | vip优惠的关于折扣的领域模型对象，拥有各个vip等级对应的折扣促销策略，可以帮助完成制定vip优惠的折扣所需要的服务。 |

**（3）**模块内部类的接口规范

**PkgPromotionController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| PkgPromotionController.getPkgList | 语法 | | public ArrayList<PkgVO> getPkgList(); |
| 前置条件 | | 已创建一个PkgPromotion领域对象 |
| 后置条件 | | 调用PkgPromotion的getPkgList方法 |
| PkgPromotionController.getPkgGoods | 语法 | | public PkgVO getPkgGoods(String id); |
| 前置条件 | | 已创建一个PkgPromotion领域对象 |
| 后置条件 | | 调用PkgPromotion的getPkgGoods方法 |
| PkgPromotionController.addPkg | 语法 | | public String addPkg(PkgVo vo) |
| 前置条件 | | 已创建一个PkgPromotion领域对象 |
| 后置条件 | | 调用PkgPromotion的addPkg方法 |
| PkgPromotionController.delPkg | 语法 | | public String delPkg(String id) |
| 前置条件 | | 已创建一个PkgPromotion领域对象 |
| 后置条件 | | 调用PkgPromotion的delPkg方法 |
| PkgPromotionController.editPkg | 语法 | | public String editPkg(PkgVO vo) |
| 前置条件 | | 已创建一个PkgPromotion领域对象 |
| 后置条件 | | 调用PkgPromotion的editPkg方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| PkgPromotion.getPkgList() | | 取得特价包列表 | |
| PkgPromotion.getPkgGoods(String id) | | 取得特价包内商品 | |
| PkgPromotion.addPkg(PkgVo vo) | | 增加特价包 | |
| PkgPromotion.editPkg(PkgVo vo) | | 编辑特价包 | |
| PkgPromotion.delPkg(PkgVo vo) | | 删除特价包 | |

**TotalPromotionMethodController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| TotalPromotionMethodController.updateTotalMethod | 语法 | | public String updateTotalMethod(double totalMoney, TotalPromotionVO vo) |
| 前置条件 | | 已创建一个TotalPromotionMethod领域对象 |
| 后置条件 | | 调TotalPromotionMethod领域对象的updateTotalMethod方法 |
| TotalPromotionMethodController.addTotalMethod | 语法 | | public String addTotalMethod(double totalMoney, TotalPromotionVO vo) |
| 前置条件 | | 已创建一个TotalPromotionMethod领域对象 |
| 后置条件 | | 调用TotalPromotionMethod领域对象的addTotalMethod方法 |
| TotalPromotionMethodController.delTotalMethod | 语法 | | public String delTotalMethod(double totalMoaney) |
| 前置条件 | | 已创建一个TotalPromotionMethod领域对象 |
| 后置条件 | | 调用TotalPromotionMethod领域对象的delTotalMethod方法 |
| TotalPromotionMethodController.getTotalMethod | 语法 | | public TotalPromotionVO getTotalMethod(double totalMoney) |
| 前置条件 | | 已创建一个TotalPromotionMethod领域对象 |
| 后置条件 | | 调用TotalPromotionMethod领域对象的getTotalMethod方法 |
| TotalPromotionMethodController.getTotalMethod | 语法 | | public ArrayList<TotalPromotionVO> getTotalMethod() |
| 前置条件 | | 已创建一个TotalPromotionMethod领域对象 |
| 后置条件 | | 调用TotalPromotionMethod领域对象的getTotalCouppon方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| TotalPromotionMethod.getTotalMethod(double totalMoney) | | 根据总价额项获得当前优惠策略 | |
| TotalPromotionMethod.getTotalMethod() | | 取得总价额优惠列表 | |
| TotalPromotionMethod.updateTotalMethod(double totalMoney, TotalPromotionVO vo) | | 更新对应总价额项的优惠策略 | |
| TotalPromotionMethod.addTotalMethod(double totalMoney, TotalPromotionVO vo) | | 增加对应总价额项的优惠策略 | |
| TotalPromotionMethod.delTotalMethod(double totalMoney) | | 删除对应总价额项的优惠策略 | |

**TotalCouponPromotionController的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| TotalCouponPromotionController.getTotalCoupon | 语法 | | | Public TotalCouponVO getTotalCoupon(int level); |
| 前置条件 | | | 已创建一个TotalCouponPromotion领域对象 |
| 后置条件 | | | 调用TotalCouponPromotion领域对象的getTotalCoupon方法 |
| TotalCouponPromotionController.getTotalCouponList | 语法 | | public ArrayList<TotalCouponVO>getTotalCouponList(); | |
| 前置条件 | | 已创建一个TotalCouponPromotion领域对象 | |
| 后置条件 | | 调用TotalCouponPromotion领域对象的getTotalCouponList方法 | |
| TotalCouponPromotionController.addTotalCoupon | 语法 | | public String addVipCoupon(TotalCouponVO vo) | |
| 前置条件 | | 已创建一个TotalCouponPromotion领域对象 | |
| 后置条件 | | 调用TotalCouponPromotion领域对象的addTotalCoupon方法 | |
| TotalCouponPromotionController.delTotalCoupon | 语法 | | public String delTotalCoupon(double totalMoney) | |
| 前置条件 | | 已创建一个TotalCouponPromotion领域对象 | |
| 后置条件 | | 调用TotalCouponPromotion领域对象的delTotalCoupon方法 | |
| TotalCouponPromotionController.editTotalCoupon | 语法 | | public String editTotalCoupon(TotalCouponVO vo) | |
| 前置条件 | | 已创建一个TotalCouponPromotion领域对象 | |
| 后置条件 | | 调用TotalCouponPromotion领域对象的editTotalCoupoon方法 | |
| **需要的服务（需接口）** | | | | |
| 服务名 | | 服务 | | |
| TotalConponPromotion.getTotalCoupon(int totalMoney) | | 根据当前总价额获得代金券 | | |
| TotalConponPromotion.getTotalCouponList() | | 获得代金券列表 | | |
| TotalConponPromotion.addTotalCoupon(TotalCouponVO vo) | | 增加对应总价额的代金券 | | |
| TotalConponPromotion.delTotalCoupon(TotalCouponVO vo); | | 删除对应总价额的代金券 | | |
| TotalConponPromotion.editTotalCoupon(TotalCouponVO vo); | | 编辑对应总价额的代金券 | | |

**TotalGiftPromotionController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| TotalGiftPromotionController.getTotalGift | 语法 | | public ArrayList<TotalGiftVO> getTotalGift(double totalMoney) |
| 前置条件 | | 已创建一个TotalGiftPromotion领域对象 |
| 后置条件 | | 调用TotalGiftPromotion领域对象的getTotalGift方法 |
| TotalGiftPromotionController.addTotalGift | 语法 | | public String addTotalGift(TotalGiftVO vo); |
| 前置条件 | | 已创建一个TotalGiftPromotion领域对象 |
| 后置条件 | | 调用TotalGiftPromotion领域对象的addTotalGift方法 |
| TotalGiftPromotionController.editTotalGift | 语法 | | public String editTotalGift(TotalGiftVO vo); |
| 前置条件 | | 已创建一个TotalPromotionBL领域对象 |
| 后置条件 | | 调用TotalGiftPromotion领域对象的editTotalGift方法 |
| TotalGiftPromotionController.delTotalGift | 语法 | | public String delTotalGift(TotalGiftVO vo); |
| 前置条件 | | 已创建一个TotalGiftPromotion领域对象 |
| 后置条件 | | 调用TotalGiftPromotion领域对象的delTotalGift方法 |
| TotalGiftPromotionController.delTotalGiftByMoney | 语法 | | public String delTotalGiftByMoney (double money); |
| 前置条件 | | 已创建一个TotalGiftPromotion领域对象 |
| 后置条件 | | 调用TotalGiftPromotion领域对象的delTotalGiftByMoney方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| TotalGiftPromotion.getTotalGift(double totalMoney) | | 根据当前总价额获得赠品 | |
| TotalGiftPromotion.updateTotalGift(TotalGiftVO vo) | | 更新对应总价额的赠品 | |
| TotalGiftPromotion.addTotalGift(TotalGiftVO vo) | | 增加对应总价额的赠品 | |
| TotalGiftPromotion.delTotalGift(TotalGiftVO vo) | | 删除对应的赠品 | |
| TotalGiftPromotion.delTotalGiftByMoney (double money) | | 删除对应总价额的赠品 | |

**VipPromotionMethodController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipPromotionMethodController.updateVipMethod | 语法 | | public String updateTotalMethod(int level,VipPromotionVO vo) |
| 前置条件 | | 已创建一个VipPromotionMethod领域对象 |
| 后置条件 | | 调VipPromotionMethod领域对象的updateVipMethod方法 |
| VipPromotionMethodController.getVipMethod | 语法 | | public VipPromotionVO getVipMethod(int vipLevel) |
| 前置条件 | | 已创建一个VipPromotionMethod领域对象 |
| 后置条件 | | 调用VipPromotionMethod领域对象的getVipMethod方法 |
| VipPromotionMethodController.getVipMethod | 语法 | | public ArrayList<VipPromotionVO> getVipMethod() |
| 前置条件 | | 已创建一个VipPromotionMethod领域对象 |
| 后置条件 | | 调用VipPromotionMethod领域对象的getVipCouppon方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipPromotionMethod.getVipMethod(int level) | | 根据VIP等级获得当前优惠策略 | |
| VipPromotionMethod.getVipMethod() | | 根据VIP等级获得当前优惠策略列表 | |
| VipPromotionMethod.updateVipMethod(int level,VipPromotionVO vo) | | 更新对应VIP等级的优惠策略 | |

**VipCouponPromotionController的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| VipCouponPromotionController.getVipCoupon | 语法 | | | public ArrayList<VipCouponVO>getVipCoupon(int level); |
| 前置条件 | | | 已创建一个VipCouponPromotion领域对象 |
| 后置条件 | | | 调用VipCouponPromotion领域对象的getVipCoupon方法 |
| VipCouponPromotionController.getVipCouponList | 语法 | | public ArrayList<VipCouponVO>getVipCouponList(); | |
| 前置条件 | | 已创建一个VipCouponPromotion领域对象 | |
| 后置条件 | | 调用VipCouponPromotion领域对象的getVipCouponList方法 | |
| VipCouponPromotionController.addVipCoupon | 语法 | | public String addVipCoupon(VipCouponVO vo) | |
| 前置条件 | | 已创建一个VipCouponPromotion领域对象 | |
| 后置条件 | | 调用VipCouponPromotion领域对象的addVipCoupon方法 | |
| VipCouponPromotionController.delVipCoupon | 语法 | | public String delVipCoupon(int level) | |
| 前置条件 | | 已创建一个VipCouponPromotion领域对象 | |
| 后置条件 | | 调用VipCouponPromotion领域对象的delVipCoupon方法 | |
| VipCouponPromotionController.editVipCoupon | 语法 | | public String editVipCoupon(VipCouponVO vo) | |
| 前置条件 | | 已创建一个VipCouponPromotion领域对象 | |
| 后置条件 | | 调用VipCouponPromotion领域对象的editVipCoupoon方法 | |
| **需要的服务（需接口）** | | | | |
| 服务名 | | 服务 | | |
| VipCouponPromotion.getVipCoupon(int level) | | 根据当前VIP等级获得代金券 | | |
| VipCouponPromotion.getVipCoupon() | | 获得所有等级的代金券列表 | | |
| VipCouponPromotion.editVipCoupon(VipCouponVO vo) | | 更新对应VIP等级的代金券 | | |
| VipCouponPromotion.addVipCoupon(VipCouponVO vo) | | 增加对应VIP等级的代金券 | | |
| VipCouponPromotion.delVipCoupon(int level) | | 删除对应VIP等级的代金券 | | |

**VipGiftPromotionController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipGiftPromotionController.getVipGift | 语法 | | public ArrayList<VipGiftVO> getVipGift(int level) |
| 前置条件 | | 已创建一个VipGiftPromotion领域对象 |
| 后置条件 | | 调用VipGiftPromotion领域对象的getVipGift方法 |
| VipGiftPromotionController.addVipGift | 语法 | | public String addVipGift(VipGiftVO vo); |
| 前置条件 | | 已创建一个VipGiftPromotion领域对象 |
| 后置条件 | | 调用VipGiftPromotion领域对象的addVipGift方法 |
| VipGiftPromotionController.editVipGift | 语法 | | public String editVipGift(VipGiftVO vo); |
| 前置条件 | | 已创建一个VipGiftPromotion领域对象 |
| 后置条件 | | 调用VipGiftPromotion领域对象的editVipGift方法 |
| VipGiftPromotionController.delVipGift | 语法 | | public String delVipGift(VipGiftVO vo); |
| 前置条件 | | 已创建一个VipGiftPromotion领域对象 |
| 后置条件 | | 调用VipGiftPromotion领域对象的delVipGift方法 |
| VipGiftPromotionController.delVipGiftByLevel | 语法 | | public String delVipGiftByLevel (int level); |
| 前置条件 | | 已创建一个VipGiftPromotion领域对象 |
| 后置条件 | | 调用VipGiftPromotion领域对象的delVipGiftByLevel方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipGiftPromotion.getVipGift(int level); | | 根据VIP等级获得当前赠品 | |
| VipGiftPromotion.editVipGift(VipGiftVO vo) | | 更新对应VIP等级的赠品 | |
| VipGiftPromotion.addVIPGift(VipGiftVO vo) | | 增加对应VIP等级的赠品 | |
| VipGiftPromotion.delVIPGift(VipGiftVO vo) | | 删除对应的赠品 | |
| VipGiftPromotion.delVIPGiftByLevel(int level) | | 删除对应VIP等级的赠品 | |

**VipDiscountPromotionController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipDiscountPromotionController.getDiscount | 语法 | | public ArrayList<DiscountVO> getDiscount(int level) |
| 前置条件 | | 已创建一个VipDiscountPromotion领域对象 |
| 后置条件 | | 调用VipDiscountPromotion领域对象的getDiscount方法 |
| VipDiscountPromotionController.getDiscount | 语法 | | public DiscountVO getDiscount(int vipLevel) |
| 前置条件 | | 已创建一个VipDiscountPromotion领域对象 |
| 后置条件 | | 调用VipDiscountPromotion领域对象的getDiscount方法 |
| VipDiscountPromotionController.addDiscount | 语法 | | public String addDiscount(DiscountVO vo) |
| 前置条件 | | 已创建一个VipDiscountPromotion领域对象 |
| 后置条件 | | 调用VipDiscountPromotion领域对象的addDiscount方法 |
| VipDiscountPromotionController.delDiscount | 语法 | | public String delDiscount(int level) |
| 前置条件 | | 已创建一个VipDiscountPromotion领域对象 |
| 后置条件 | | 调用VipDiscountPromotion领域对象的delDiscount方法 |
| VipDiscountPromotionController.editDiscount | 语法 | | public String editDiscount(DiscountVO vo) |
| 前置条件 | | 已创建一个VipDiscountPromotion领域对象 |
| 后置条件 | | 调用VipDiscountPromotion领域对象的editDiscount方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipDiscountPromotion.getDiscount(int level) | | 取得对应VIP等级的折扣 | |
| VipDiscountPromotion.getDiscount() | | 取得各个VIP等级的折扣列表 | |
| VipDiscountPromotion.addDiscount(DiscountVO vo) | | 增加折扣 | |
| VipDiscountPromotion.delDiscount(int level) | | 删除折扣 | |
| VipDiscountPromotion.editDiscount(DiscountVO vo) | | 编辑折扣 | |

**PkgPromotion的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| PkgPromotion.getPkgList | 语法 | | public ArrayList<PkgVO> getPkgList(); |
| 前置条件 | | 已创建一个CommodityProvider领域对象 |
| 后置条件 | | 调用CommodityProvider的getAllPkgs方法 |
| PkgPromotion.getPkgGoods | 语法 | | public PkgVO getPkgGoods(String id); |
| 前置条件 | | 已创建一个CommodityProvider领域对象 |
| 后置条件 | | 调用CommodityProvider的get\_Pkg方法 |
| PkgPromotion.addPkg | 语法 | | public String addPkg(PkgVo vo) |
| 前置条件 | | 已创建一个CommodityProvider领域对象 |
| 后置条件 | | 调用CommodityProvider的addPkg方法 |
| PkgPromotion.delPkg | 语法 | | public String delPkg(String id) |
| 前置条件 | | 已创建一个CommodityProvider领域对象 |
| 后置条件 | | 调用CommodityProvider的delPkg方法 |
| PkgPromotion.editPkg | 语法 | | public String editPkg(PkgVO vo) |
| 前置条件 | | 已创建一个CommodityProvider领域对象 |
| 后置条件 | | 调用CommodityProvider的changePkg方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| CommodityProvider.getAllPkgs() | | 取得特价包列表 | |
| CommodityProvider.get\_Pkg(String id) | | 取得特价包内商品 | |
| CommodityProvider.addPkg(PkgVo vo) | | 增加特价包 | |
| CommodityProvider.editPkg(PkgVo vo) | | 编辑特价包 | |
| CommodityProvider.changePkg(PkgVo vo) | | 删除特价包 | |

**TotalPromotionMethod的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| TotalPromotionMethod.updateTotalMethod | 语法 | | public String updateTotalMethod(double totalMoney, TotalPromotionVO vo) |
| 前置条件 | | 已创建一个TotalPromotionMethodDataService领域对象 |
| 后置条件 | | 调TotalPromotionMethodDataService领域对象的updateTotalMethod方法 |
| TotalPromotionMethod.addTotalMethod | 语法 | | public String addTotalMethod(double totalMoney, TotalPromotionVO vo) |
| 前置条件 | | 已创建一个TotalPromotionMethodDataService领域对象 |
| 后置条件 | | 调用TotalPromotionMethodDataService领域对象的addTotalMethod方法 |
| TotalPromotionMethodController.delTotalMethod | 语法 | | public String delTotalMethod(double totalMoaney) |
| 前置条件 | | 已创建一个TotalPromotionMethodDataService领域对象 |
| 后置条件 | | 调用TotalPromotionMethodDataService领域对象的delTotalMethod方法 |
| TotalPromotionMethod.getTotalMethod | 语法 | | public TotalPromotionVO getTotalMethod(double totalMoney) |
| 前置条件 | | 已创建一个TotalPromotionMethodDataService领域对象 |
| 后置条件 | | 调用TotalPromotionMethodDataService领域对象的getTotalMethod方法 |
| TotalPromotionMethod.getTotalMethod | 语法 | | public ArrayList<TotalPromotionVO> getTotalMethod() |
| 前置条件 | | 已创建一个TotalPromotionMethodDataService领域对象 |
| 后置条件 | | 调用TotalPromotionMethodDataService领域对象的getTotalCouppon方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| TotalPromotionMethodDataService.getTotalMethod(double totalMoney) | | 根据总价额项获得当前优惠策略 | |
| TotalPromotionMethodDataService.getTotalMethod() | | 取得总价额优惠列表 | |
| TotalPromotionMethodDataService.updateTotalMethod(double totalMoney, TotalPromotionPO po) | | 更新对应总价额项的优惠策略 | |
| TotalPromotionMethodDataService.addTotalMethod(double totalMoney, TotalPromotionPO po) | | 增加对应总价额项的优惠策略 | |
| TotalPromotionMethodDataService.delTotalMethod(double totalMoney) | | 删除对应总价额项的优惠策略 | |

**TotalCouponPromotion的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| TotalCouponPromotion.getTotalCoupon | 语法 | | | Public TotalCouponVO getTotalCoupon(int level); |
| 前置条件 | | | 已创建一个TotalCouponPromotionDataService领域对象 |
| 后置条件 | | | 调用TotalCouponPromotionDataService领域对象的getTotalCoupon方法 |
| TotalCouponPromotion.getTotalCouponList | 语法 | | public ArrayList<TotalCouponVO>getTotalCouponList(); | |
| 前置条件 | | 已创建一个TotalCouponPromotionDataService领域对象 | |
| 后置条件 | | 调用TotalCouponPromotionDataService领域对象的getTotalCouponList方法 | |
| TotalCouponPromotion.addTotalCoupon | 语法 | | public String addVipCoupon(TotalCouponVO vo) | |
| 前置条件 | | 已创建一个TotalCouponPromotionDataService领域对象 | |
| 后置条件 | | 调用TotalCouponPromotionDataService领域对象的addTotalCoupon方法 | |
| TotalCouponPromotion.delTotalCoupon | 语法 | | public String delTotalCoupon(double totalMoney) | |
| 前置条件 | | 已创建一个TotalCouponPromotionDataService领域对象 | |
| 后置条件 | | 调用TotalCouponPromotionDataService领域对象的delTotalCoupon方法 | |
| TotalCouponPromotion.editTotalCoupon | 语法 | | public String editTotalCoupon(TotalCouponVO vo) | |
| 前置条件 | | 已创建一个TotalCouponPromotionDataService领域对象 | |
| 后置条件 | | 调用TotalCouponPromotionDataService领域对象的updateTotalCoupoon方法 | |
| **需要的服务（需接口）** | | | | |
| 服务名 | | 服务 | | |
| TotalCouponPromotionDataService.getTotalCoupon(double totalMoney) | | 根据当前总价额获得代金券 | | |
| TotalCouponPromotionDataService.getTotalCouponList() | | 获得代金券列表 | | |
| TotalCouponPromotionDataService.updateTotalCoupon(double totalMoney,TotalCouponPO po) | | 更新对应总价额的代金券 | | |
| TotalCouponPromotionDataService.addTotalCoupon(double totalMoney ,TotalCouponPO po) | | 增加对应总价额的代金券 | | |
| TotalCouponPromotionDataService.delTotalCoupon(TotalCouponPO po); | | 删除对应总价额的代金券 | | |

**TotalGiftPromotion的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| TotalGiftPromotion.getTotalGift | 语法 | | public ArrayList<TotalGiftVO> getTotalGift(double totalMoney) |
| 前置条件 | | 已创建一个TotalGiftPromotionDataService领域对象 |
| 后置条件 | | 调用TotalGiftPromotionDataService领域对象的getTotalGift方法 |
| TotalGiftPromotion.addTotalGift | 语法 | | public String addTotalGift(TotalGiftVO vo); |
| 前置条件 | | 已创建一个TotalGiftPromotionDataService领域对象 |
| 后置条件 | | 调用TotalGiftPromotionDataService领域对象的addTotalGift方法 |
| TotalGiftPromotion.editTotalGift | 语法 | | public String editTotalGift(TotalGiftVO vo); |
| 前置条件 | | 已创建一个TotalGiftPromotionDataService领域对象 |
| 后置条件 | | 调用TotalGiftPromotionDataService领域对象的updateTotalGift方法 |
| TotalGiftPromotion.delTotalGift | 语法 | | public String delTotalGift(TotalGiftVO vo); |
| 前置条件 | | 已创建一个TotalGiftPromotionDataService领域对象 |
| 后置条件 | | 调用TotalGiftPromotionDataService领域对象的delTotalGift方法 |
| TotalGiftPromotion.delTotalGiftByMoney | 语法 | | public String delTotalGiftByMoney (double money); |
| 前置条件 | | 已创建一个TotalGiftPromotionDataService领域对象 |
| 后置条件 | | 调用TotalGiftPromotionDataService领域对象的delTotalGiftByMoney方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| TotalGiftPromotionDataService.getTotalGift(double totalMoney) | | 根据当前总价额获得赠品 | |
| TotalGiftPromotionDataService.updateTotalGift(double totalMoney ,TotalGiftPO po) | | 更新对应总价额的赠品 | |
| TotalGiftPromotionDataService.addTotalGift(double totalMoney ,TotalGiftPO po) | | 增加对应总价额的赠品 | |
| TotalGiftPromotionDataService.delTotalGift(double totalMoney ,TotalGiftPO po) | | 删除对应的赠品 | |
| TotalGiftPromotionDataService.delTotalGiftByMoney (double money) | | 删除对应总价额的赠品 | |

**VipPromotionMethod的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipPromotionMethod.updateVipMethod | 语法 | | public String updateTotalMethod(int level,VipPromotionVO vo) |
| 前置条件 | | 已创建一个VipPromotionMethodDataService领域对象 |
| 后置条件 | | 调VipPromotionMethodDataService领域对象的updateVipMethod方法 |
| VipPromotionMethod.getVipMethod | 语法 | | public VipPromotionVO getVipMethod(int vipLevel) |
| 前置条件 | | 已创建一个VipPromotionMethodDataService领域对象 |
| 后置条件 | | 调用VipPromotionMethodDataService领域对象的getVipMethod方法 |
| VipPromotionMethod.getVipMethod | 语法 | | public ArrayList<VipPromotionVO> getVipMethod() |
| 前置条件 | | 已创建一个VipPromotionMethodDataService领域对象 |
| 后置条件 | | 调用VipPromotionMethodDataService领域对象的getVipCouppon方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipPromotionMethodDataService.getVipMethod(int level) | | 根据VIP等级获得当前优惠策略 | |
| VipPromotionMethodDataService.getVipMethod() | | 根据VIP等级获得当前优惠策略列表 | |
| VipPromotionMethodDataService.updateVipMethod(int level,VipPromotionPO po) | | 更新对应VIP等级的优惠策略 | |

**VipCouponPromotion的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| VipCouponPromotion.getVipCoupon | 语法 | | | public ArrayList<VipCouponVO>getVipCoupon(int level); |
| 前置条件 | | | 已创建一个VipCouponPromotionDataService领域对象 |
| 后置条件 | | | 调用VipCouponPromotionDataService领域对象的getVipCoupon方法 |
| VipCouponPromotion.getVipCouponList | 语法 | | public ArrayList<VipCouponVO>getVipCouponList(); | |
| 前置条件 | | 已创建一个VipCouponPromotionDataService领域对象 | |
| 后置条件 | | 调用VipCouponPromotionDataService领域对象的getVipCouponList方法 | |
| VipCouponPromotion.addVipCoupon | 语法 | | public String addVipCoupon(VipCouponVO vo) | |
| 前置条件 | | 已创建一个VipCouponPromotionDataService领域对象 | |
| 后置条件 | | 调用VipCouponPromotionDataService领域对象的addVipCoupon方法 | |
| VipCouponPromotion.delVipCoupon | 语法 | | public String delVipCoupon(int level) | |
| 前置条件 | | 已创建一个VipCouponPromotionDataService领域对象 | |
| 后置条件 | | 调用VipCouponPromotionDataService领域对象的delVipCoupon方法 | |
| VipCouponPromotion.editVipCoupon | 语法 | | public String editVipCoupon(VipCouponVO vo) | |
| 前置条件 | | 已创建一个VipCouponPromotionDataService领域对象 | |
| 后置条件 | | 调用VipCouponPromotionDataService领域对象的editVipCoupoon方法 | |
| **需要的服务（需接口）** | | | | |
| 服务名 | | 服务 | | |
| VipCouponPromotion.getVipCoupon(int level) | | 根据当前VIP等级获得代金券 | | |
| VipCouponPromotion.getVipCoupon() | | 获得所有等级的代金券列表 | | |
| VipCouponPromotion.editVipCoupon(int level,VipCouponPO po) | | 更新对应VIP等级的代金券 | | |
| VipCouponPromotion.addVipCoupon(VipCouponPO po) | | 增加对应VIP等级的代金券 | | |
| VipCouponPromotion.delVipCoupon(int level) | | 删除对应VIP等级的代金券 | | |

**VipGiftPromotion的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipGiftPromotion.getVipGift | 语法 | | public ArrayList<VipGiftVO> getVipGift(int level) |
| 前置条件 | | 已创建一个VipGiftPromotionDataService领域对象 |
| 后置条件 | | 调用VipGiftPromotionDataService领域对象的getVipGift方法 |
| VipGiftPromotion.addVipGift | 语法 | | public String addVipGift(VipGiftVO vo); |
| 前置条件 | | 已创建一个VipGiftPromotionDataService领域对象 |
| 后置条件 | | 调用VipGiftPromotionDataService领域对象的addVipGift方法 |
| VipGiftPromotion.editVipGift | 语法 | | public String editVipGift(VipGiftVO vo); |
| 前置条件 | | 已创建一个VipGiftPromotionDataService领域对象 |
| 后置条件 | | 调用VipGiftPromotionDataService领域对象的editVipGift方法 |
| VipGiftPromotion.delVipGift | 语法 | | public String delVipGift(VipGiftVO vo); |
| 前置条件 | | 已创建一个VipGiftPromotionDataService领域对象 |
| 后置条件 | | 调用VipGiftPromotionDataService领域对象的delVipGift方法 |
| VipGiftPromotion.delVipGiftByLevel | 语法 | | public String delVipGiftByLevel (int level); |
| 前置条件 | | 已创建一个VipGiftPromotionDataService领域对象 |
| 后置条件 | | 调用VipGiftPromotionDataService领域对象的delVipGiftByLevel方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipGiftPromotionDataService.getVipGift(int level); | | 根据VIP等级获得当前赠品 | |
| VipGiftPromotionDataService.editVipGift(int level,VipGiftPO po) | | 更新对应VIP等级的赠品 | |
| VipGiftPromotionDataService.addVIPGift(int level,VipGiftPO po) | | 增加对应VIP等级的赠品 | |
| VipGiftPromotionDataService.delVIPGift(int level,VipGiftPO po) | | 删除对应的赠品 | |
| VipGiftPromotionDataService.delVIPGiftByLevel(int level) | | 删除对应VIP等级的赠品 | |

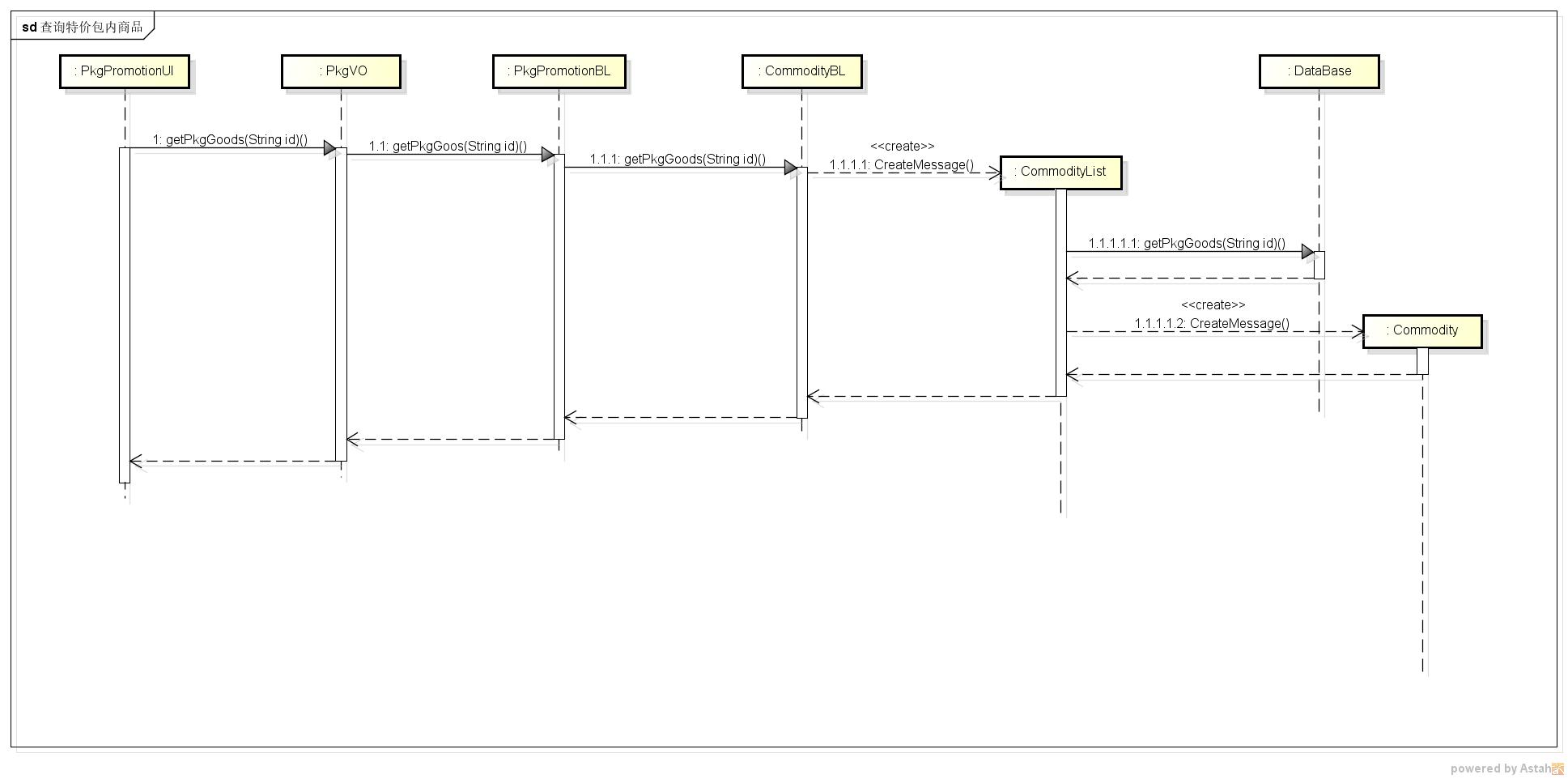
**VipDiscountPromotion的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipDiscountPromotion.getDiscount | 语法 | | public ArrayList<DiscountVO> getDiscount(int level) |
| 前置条件 | | 已创建一个VipDiscountPromotionDataService领域对象 |
| 后置条件 | | 调用VipDiscountPromotionDataService领域对象的getDiscount方法 |
| VipDiscountPromotion.getDiscount | 语法 | | public DiscountVO getDiscount(int vipLevel) |
| 前置条件 | | 已创建一个VipDiscountPromotionDataService领域对象 |
| 后置条件 | | 调用VipDiscountPromotionDataService领域对象的getDiscount方法 |
| VipDiscountPromotion.addDiscount | 语法 | | public String addDiscount(DiscountVO vo) |
| 前置条件 | | 已创建一个VipDiscountPromotionDataService领域对象 |
| 后置条件 | | 调用VipDiscountPromotionDataService领域对象的addDiscount方法 |
| VipDiscountPromotion.delDiscount | 语法 | | public String delDiscount(int level) |
| 前置条件 | | 已创建一个VipDiscountPromotionDataService领域对象 |
| 后置条件 | | 调用VipDiscountPromotionDataService领域对象的delDiscount方法 |
| VipDiscountPromotion.editDiscount | 语法 | | public String editDiscount(DiscountVO vo) |
| 前置条件 | | 已创建一个VipDiscountPromotionDataService领域对象 |
| 后置条件 | | 调用VipDiscountPromotionDataService领域对象的editDiscount方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipDiscountPromotionDataService.getDiscount(int level) | | 取得对应VIP等级的折扣 | |
| VipDiscountPromotionDataService.getDiscount() | | 取得各个VIP等级的折扣列表 | |
| VipDiscountPromotionDataService.addDiscount(DiscountPO po) | | 增加折扣 | |
| VipDiscountPromotionDataService.delDiscount(int level) | | 删除折扣 | |
| VipDiscountPromotionDataService.editDiscount(DiscountPO po) | | 编辑折扣 | |

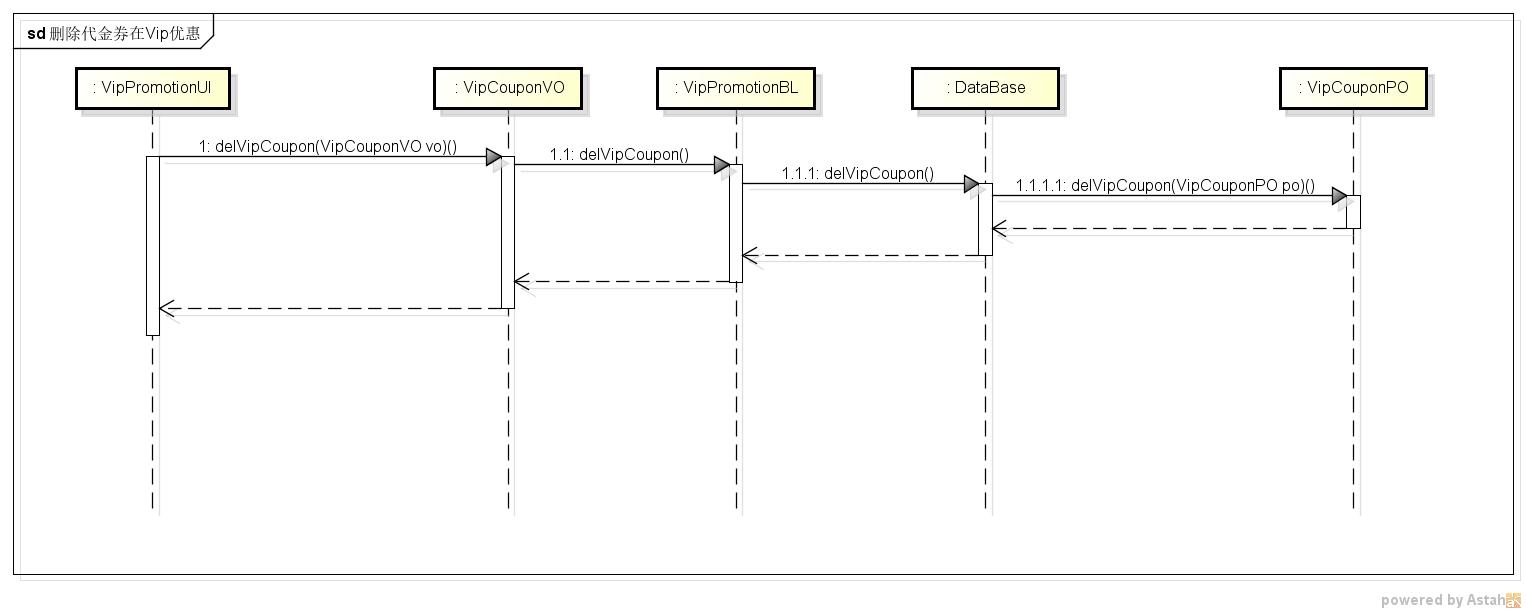
**（4）**业务逻辑层的动态模型

下列图表明了在制定促销策略中涉及的相关对象间的协作

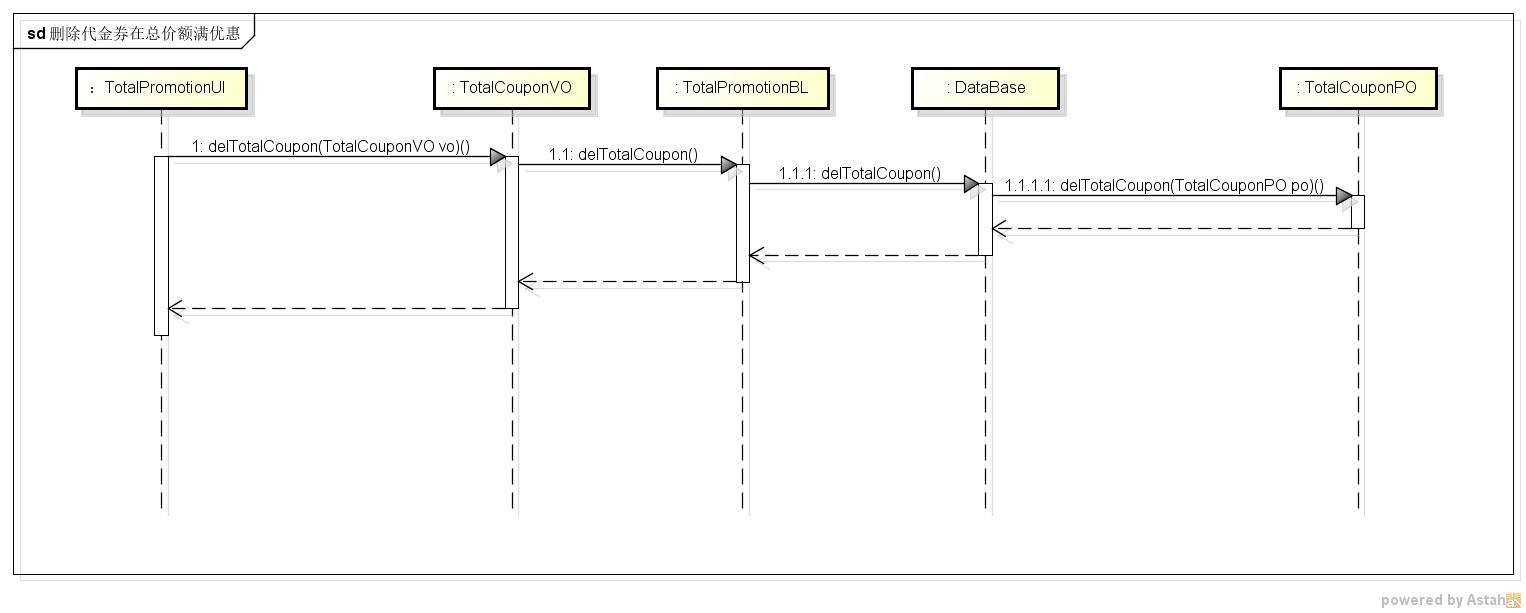
**查询特价包内商品的顺序图**



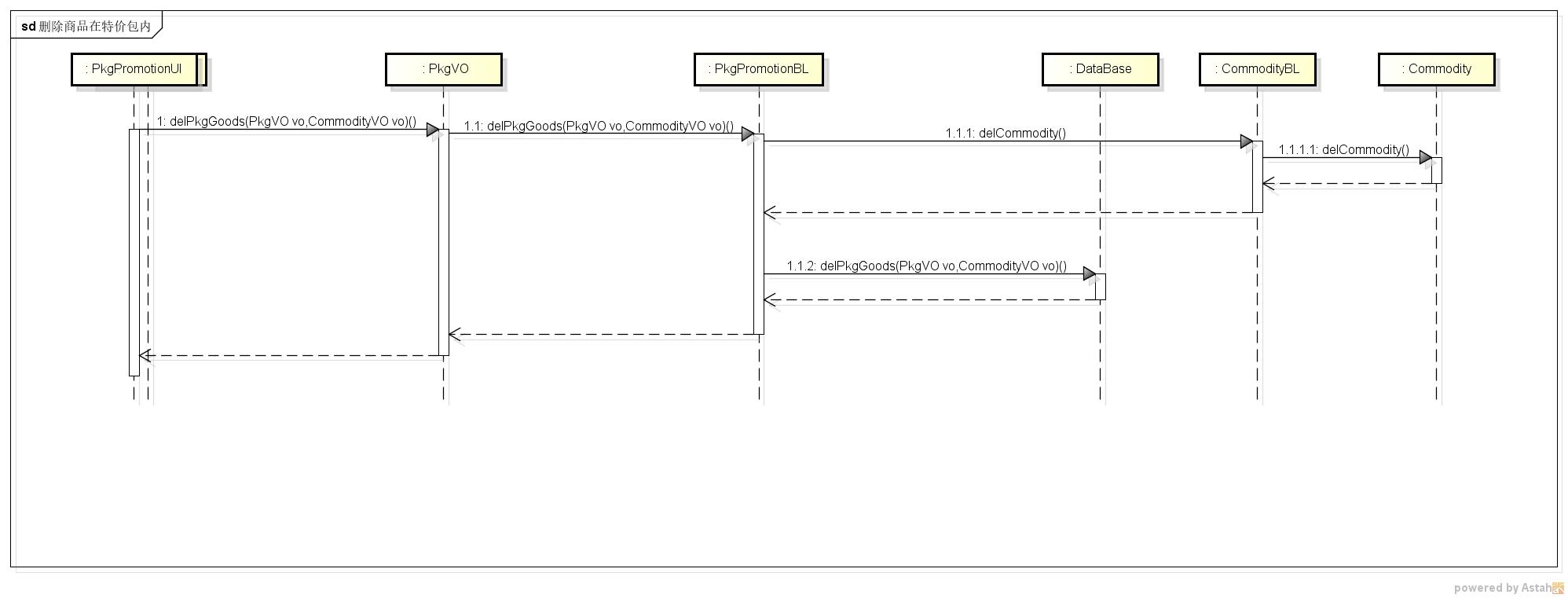
**删除Vip优惠内代金券的顺序图**



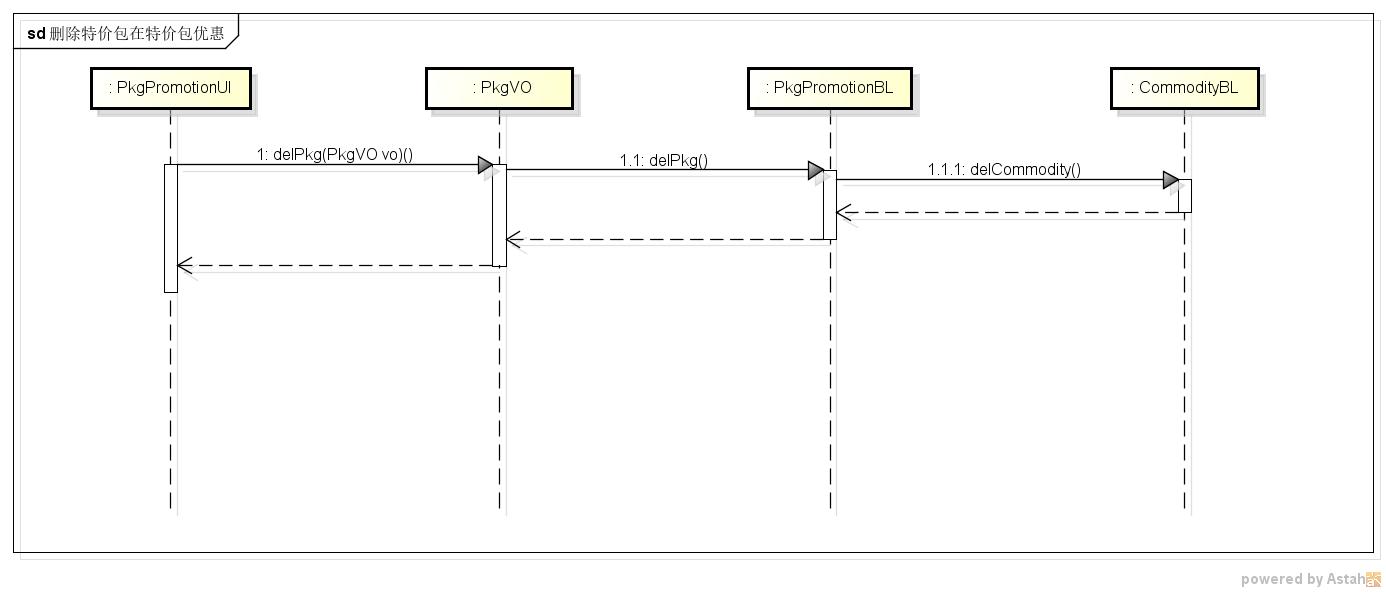
**删除总价额满优惠内代金券的顺序图**



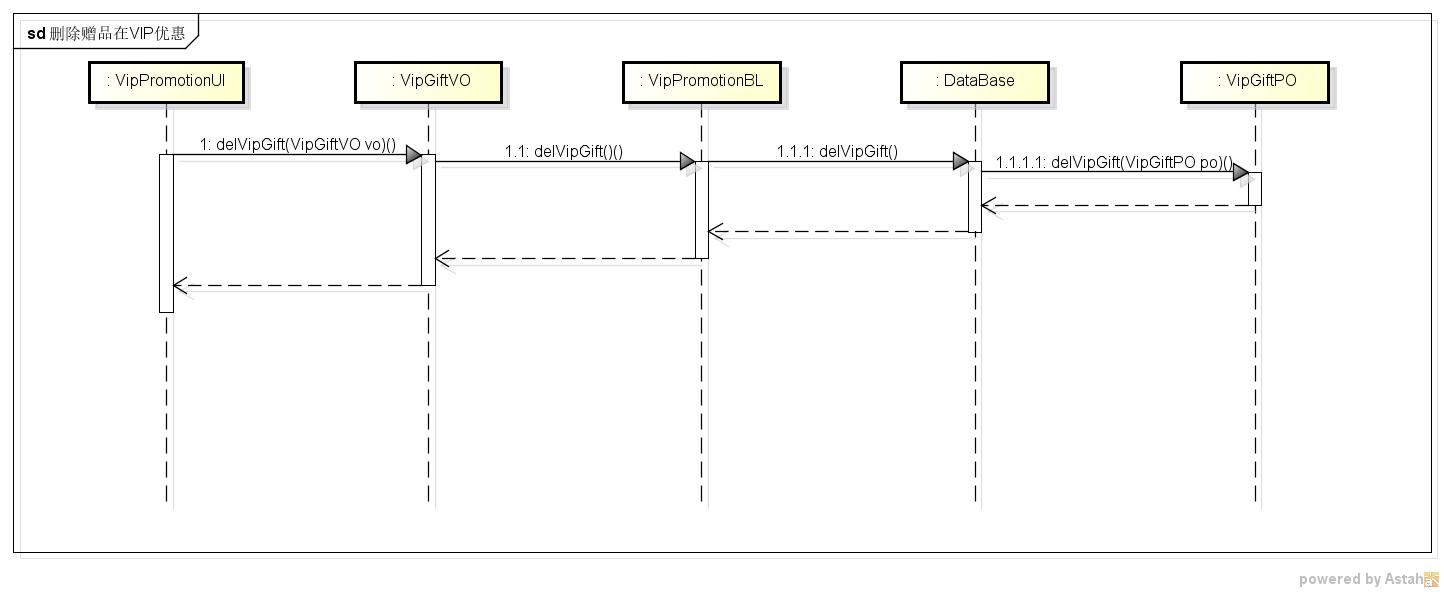
**删除特价包内商品的顺序图**



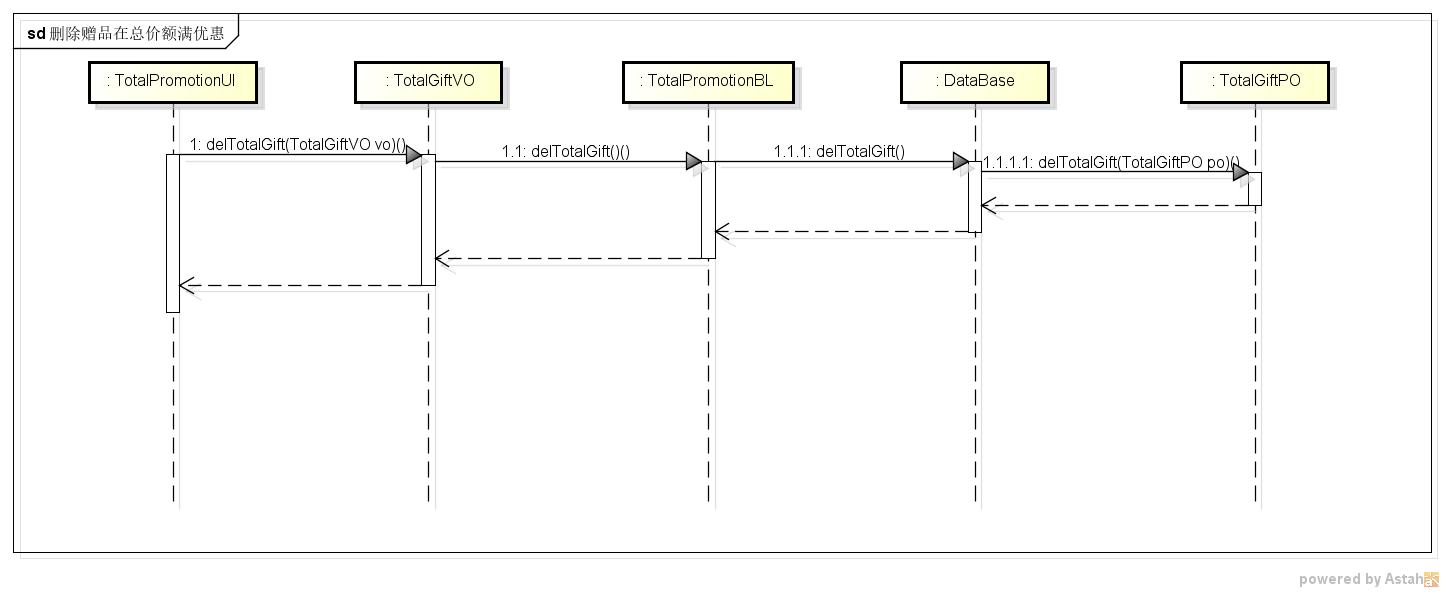
**删除特价包优惠内特价包的顺序图**



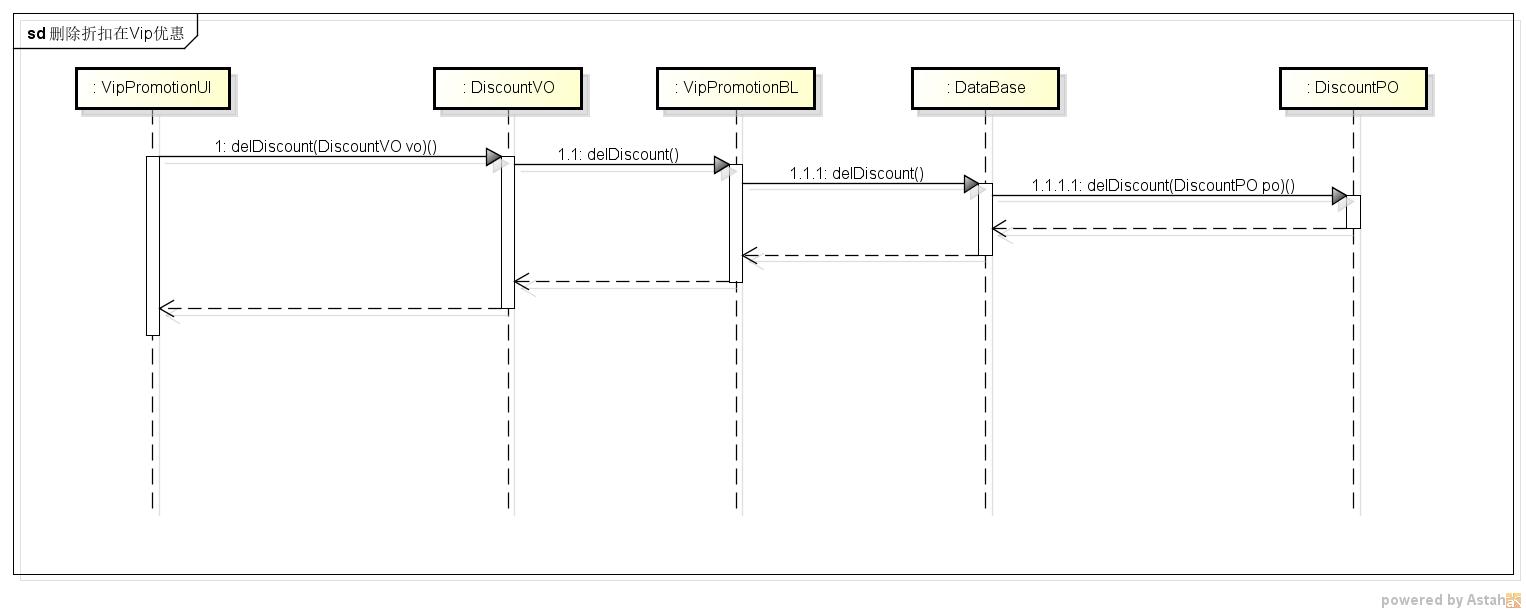
**删除Vip优惠内赠品的顺序图**



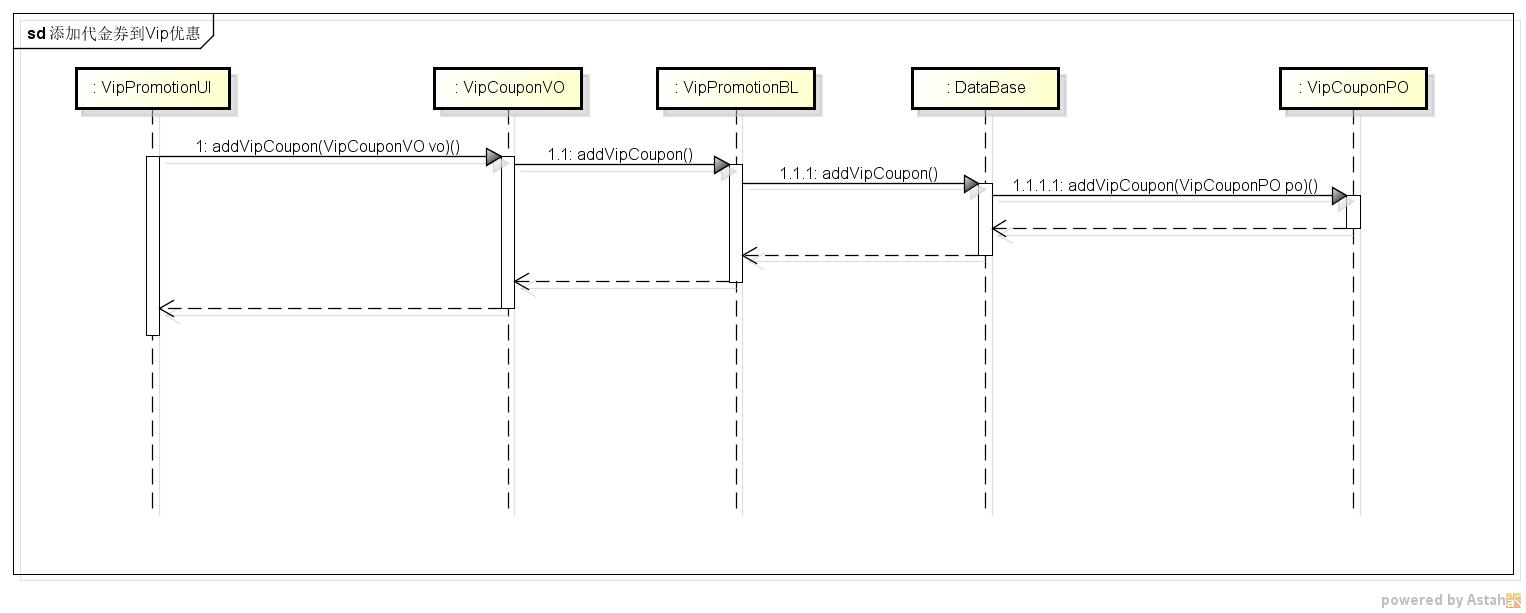
**删除总价额满优惠内赠品的顺序图**



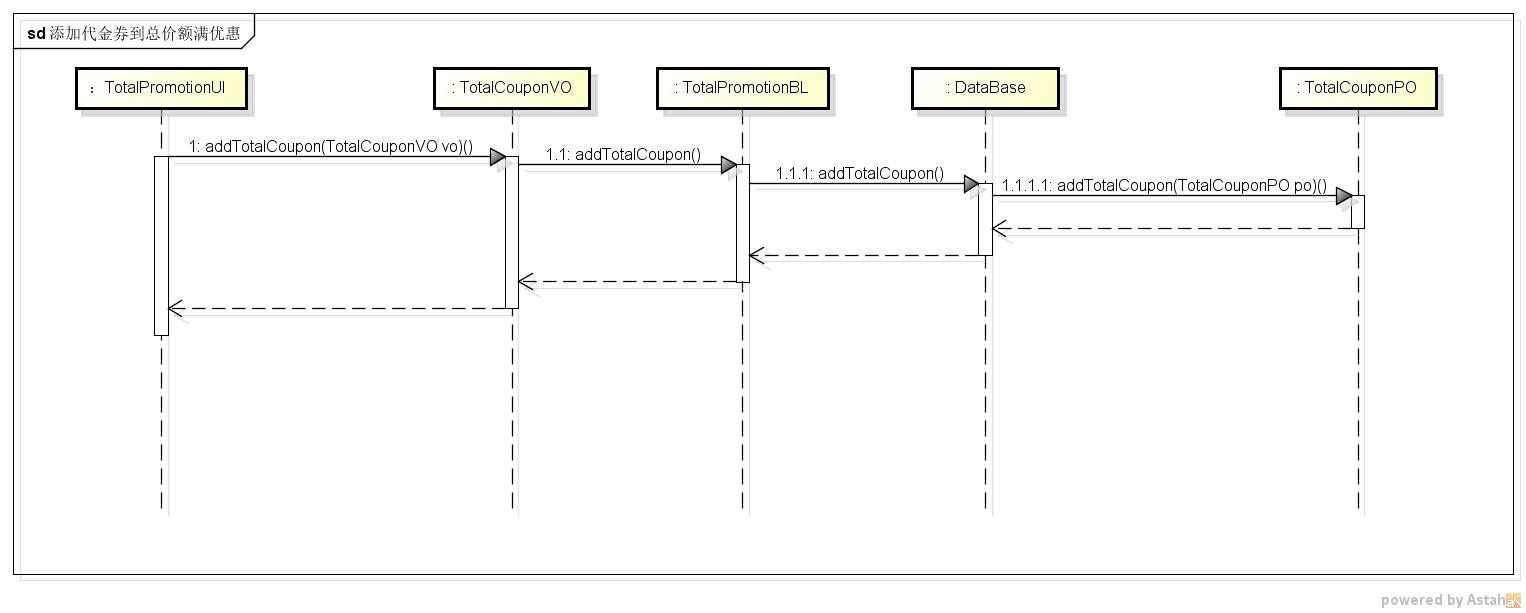
**删除Vip优惠内折扣的顺序图**



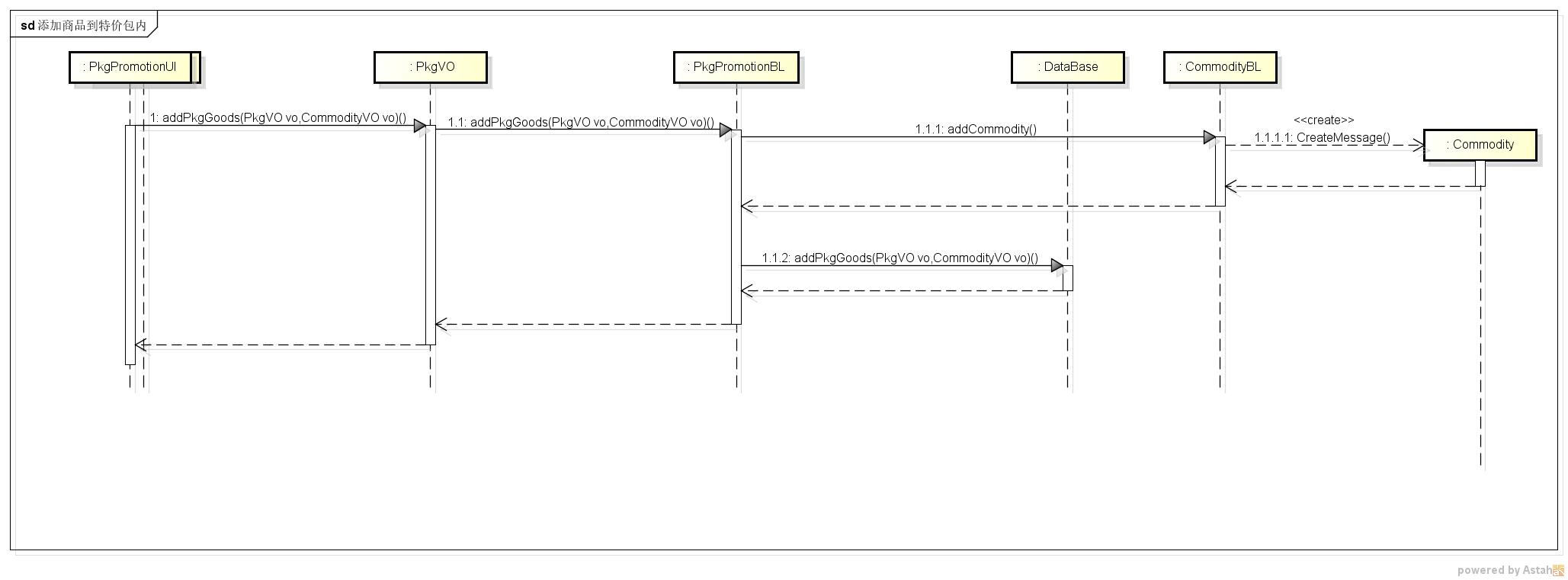
**添加代金券到Vip优惠的顺序图**



**添加代金券到总价额满优惠的顺序图**



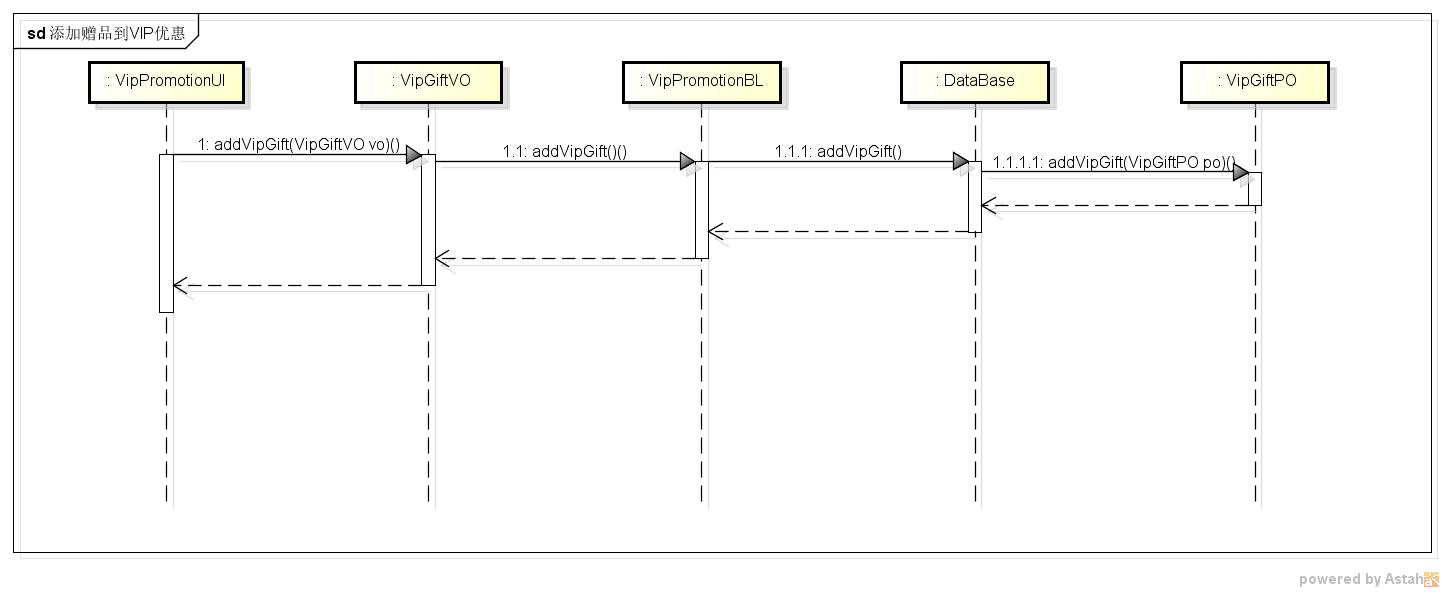
**添加商品到特价包内的顺序图**



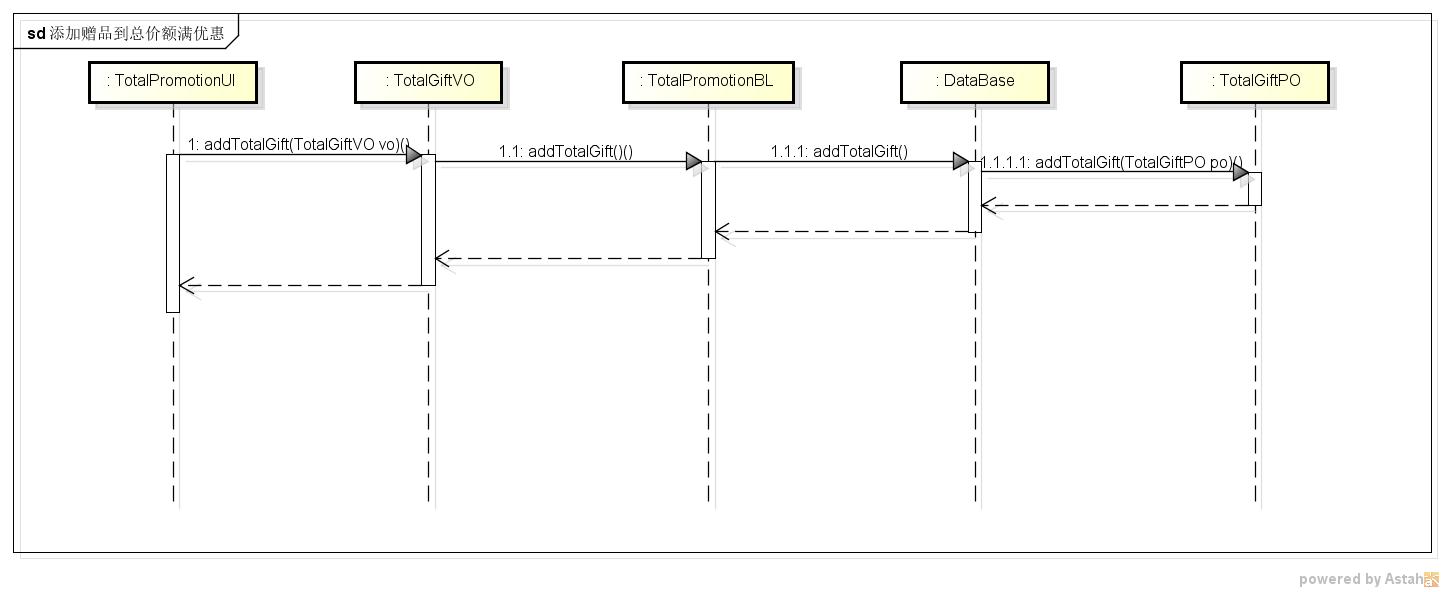
**添加特价包到特价包优惠的顺序图**



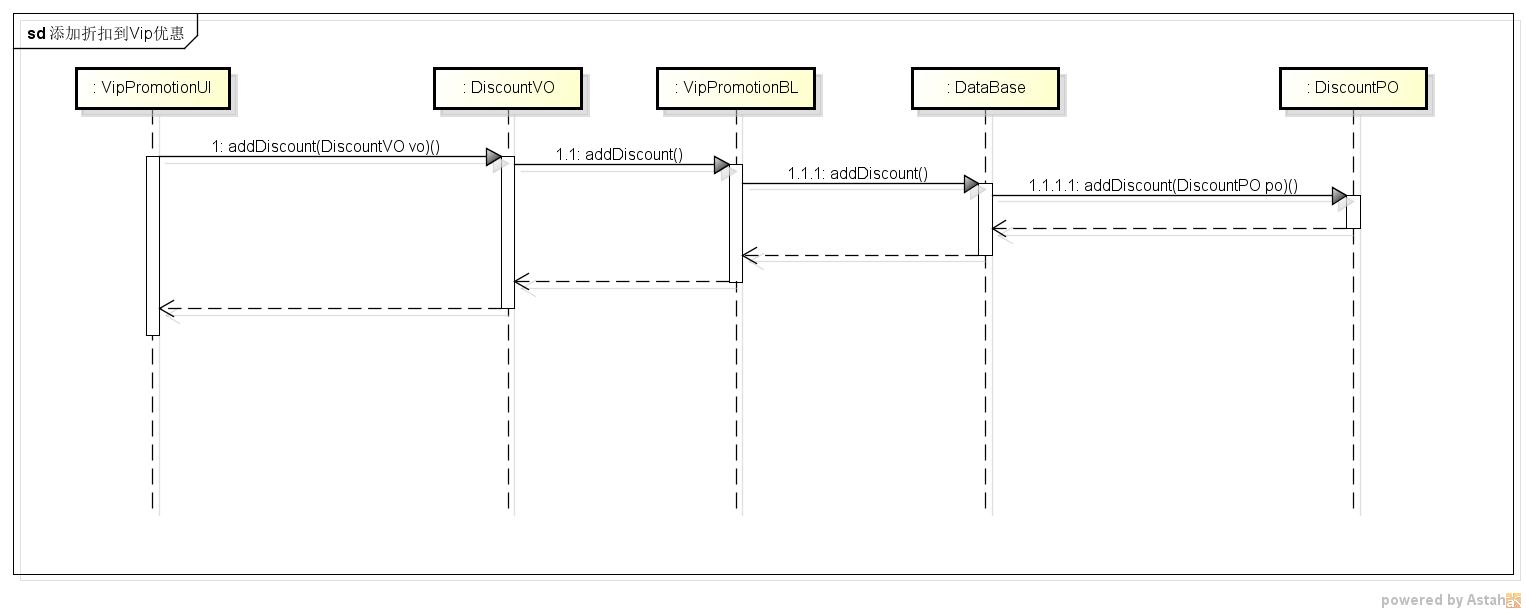
**添加赠品到Vip优惠的顺序图**



**添加赠品到总价额满优惠的顺序图**



**添加折扣到Vip优惠的顺序图**



**（5）**业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自控制器委托给不同的领域对象。

# 4.2数据层层结构视角

## 4.2.1 KindData模块

### （1）模块概述

KindData模块功能较为简单，只是承担了对分类树根节点对象的读，写，清空等操作。其他复杂逻辑聚在逻辑层实现。

### (2)整体结构

由于功能十分简单，只有KindData一个类，模块设计图和模块动态模型在此省略。

|  |  |
| --- | --- |
| 模块 | 职责 |
| KindData | 负责实现对分类树根节点对象的读，写，清空等操作 |

### (3)模块内部类接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供借口）** | | |
| **kindData.read\_\_root** | **语法** | **public** KindPo read\_root() |
| **前置条件** | **成功启动服务器** |
| **后置条件** | **返回根节点Po对象** |
| **kindData.init** | **语法** | **public** **void** init() |
| **前置条件** | **成功启动服务器** |
| **后置条件** | **将根节点root初始化为没有子节点的空节点** |
| **kindData.write** | **语法** | **Public void** write\_root(KindPo root) |
| **前置条件** | **成功启动服务器** |
| **后置条件** | **更新根节点对象** |

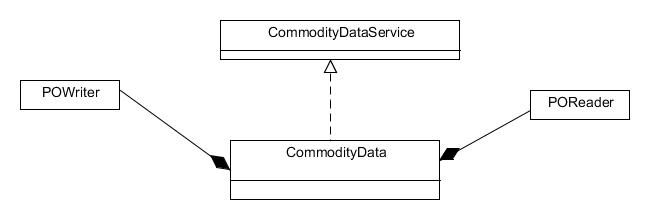
## 4.2.2 CommodityData模块

### （1）模块概述

**CommodityData模块承担了对库存商品数据的增删改和库存单据数据的管理。**

### （2）整体结构

**运用泛型，PoReader和PoWriter对底层数据进行读取，避免了频繁地读写序列化对象的操作。**



### （3）模块内部接口规范

|  |  |  |
| --- | --- | --- |
|  | | |
| **commodityData.** addCommodity | **语法** | **public** **int** addCommodity(CommodityPo po) |
| **前置条件** | **获得一个CommodityPo** |
| **后置条件** | **在库存数据中加入1个CommodityPo对象** |
| **commodityData** ．delCommodity | **语法** | **public** **int** delCommodity(**int** loc) |
| **前置条件** | **获得摸个ｐｏ在Ｐｏ信息列表中的位置** |
| **后置条件** | **完成删除** |
| **commodityData**．changeCommodity | **语法** | changeCommodity(CommodityPo comdPo,**int** loc) |
| **前置条件** | **获得一个ｐｏ及其要替换的位置** |
| **后置条件** | **完成ｐｏ的替换** |
| **commodityData**．findCommodity | **语法** | **public** **int** findCommodity(String name) |
| **前置条件** | **获得某个商品的名称** |
| **后置条件** | **返回该Ｐｏ的位置** |
| **commodityData** | **语法** | **public** Ex\_StockPo examineStock(**double** beginTime, **double** endTime) |
| **前置条件** | **获得一个合理时间区间** |
| **后置条件** | **返回一个完整的Ｅｘ＿ＳｔｏｃｋＰｏ** |
| **commodityData．**getAllCommoditys | **语法** | **public** ArrayList<CommodityPo> getAllCommoditys() |
| **前置条件** | **无** |
| **后置条件** | **返回所有库存数据** |
| **commodityData．**getCommoditys | **语法** | **public** ArrayList<CommodityPo> getCommoditys(String name) |
| **前置条件** | **获得一个商品名称** |
| **后置条件** | **返回同名的商品ｐｏ** |
| **commodityData** | **语法** | **public** CommodityPo getCommodity(String kindNum) |
| **前置条件** | **获得一个商品编号** |
| **后置条件** | **返回符合编号的商品ＰＯ** |
| **commodityData．getCommoditys\_ByKindName** | **语法** | **Public ArrayList<CommodityＰo> getCommoditys\_ByKindName(String kindName);** |
| **前置条件** | **系统需要获得某一分类的所有商品** |
| **后置条件** | **返回同属kindName的所有商品项目** |
| **commodityData．**addBill | **语法** | **public** **int** addBill(StockBillsPo po) |
| **前置条件** | **获得一个库存单据对象** |
| **后置条件** | **加入到单据数据中** |
| **commodityData．**getAlarm | **语法** | **public** ArrayList<StockBillsPo> getAlarm() |
| **前置条件** | **无** |
| **后置条件** | **获取所有报警单** |
| **commodityData．**resume\_NumOfCommodity | **语法** | **public** **int** resume\_NumOfCommodity(String name,String modelNum,**int** fact\_num) |
| **前置条件** | **获得库存数量与实际不符的商品名称，型号和具体数量** |
| **后置条件** | **将数据恢复为ｆａｃｔ＿ｎｕｍ** |
| **commodityData** | **语法** | **public** **void** checkAlarm() |
| **前置条件** | **无** |
| **后置条件** | **检查报警的商品项目，加入但ａｌａｒｍＬｉｓｔ** |
| **commodityData** | **语法** | **public** ArrayList<BillPO> getBills(infoPO po) |
| **前置条件** | **获得ｉｎｆｏＰｏ** |
| **后置条件** | **返回符合查询标准的库存单据** |
| **commodityData**．changeStock | **语法** | **public** String changeStock(String num, **int** number, **double** inPrice,**double** outPrice) |
| **前置条件** | **无** |
| **后置条件** | **根据参数对库存信息进行正确修改** |
| **commodityData**．MarketFind | **语法** | **public** CommodityPo MarketFind(String numberOfkind) |
| **前置条件** | **无** |
| **后置条件** | **返回符合条件的ＣｏｍｍｏｄｉｔｙＰｏ** |
| **commodityData．addPkg** | **语法** | **Public int addPkg(PkgＰO ＰＯ);** |
| **前置条件** | **系统需要添加一个新的特价包** |
| **后置条件** | **返回更新是否成功** |
| **commodityData．delPkg** | **语法** | **Public int delPkg（PkgＰO ＰＯ）;** |
| **前置条件** | **系统需要删除一个特价包** |
| **后置条件** | **返回是否删除成功** |
| **commodityData．ｄｅｌＰｋｇ** | **语法** | **public int delPkg(String id)** |
| **前置条件** | **获得Pkg的Id** |
| **后置条件** | **返回是否删除成功** |
| **commodityData．changePkg** | **语法** | **Public int changePkg(PkgＰO ＰO);** |
| **前置条件** | **系统需要更改一个特价包** |
| **后置条件** | **返回是否更改成功** |
| **commodityDatａ．ｇetAllPkgs** | **语法** | **Public ArrayList<PkgＰO>**  **getAllPkgs()** |
| **前置条件** | **系统需要所有特价包的信息** |
| **后置条件** | **返回所有特价包的信息** |
| **commodityDatａ．**matchPkg | **语法** | **public** **int** matchPkg (PkgPO po) |
| **前置条件** | **无** |
| **后置条件** | **返回特价包的位置** |
| **commodityDatａ．getOneCommodity** | **语法** | **Public CommodityＰo getOneCommodity(int loc);** |
| **前置条件** | **系统需要获得在商品列表中的第loc个位置的商品信息** |
| **后置条件** | **返回处在第loc个位置的商品项目** |
| **commodityDatａ．**delCommodityByName | **语法** | **public** **int** delCommodityByName(String name) |
| **前置条件** | **无** |
| **后置条件** | **根据名字删除商品** |
| **commodityDatａ** | **语法** | **public** **int** delCommodityByKindName(String kindName) |
| **前置条件** | **无** |
| **后置条件** | **删除属于ｋｉｎｄＮａｍｅ的商品** |
| **commodityDatａ** | **语法** | **public** **void** init() **throws** RemoteException |
| **前置条件** | **无** |
| **后置条件** | **清空库存数据** |

## 4.2.3 memberData模块

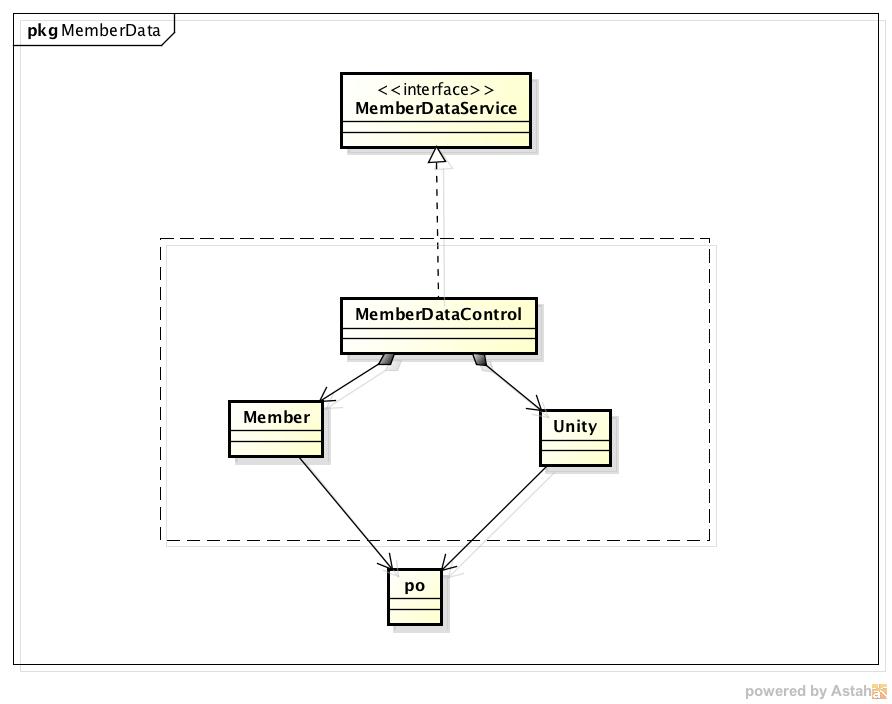
### （1）模块概述

memberData模块承担了需求参见需求规格说明文档的功能需求以及非功能需求。

### （2）整体结构

根据体系结构的设计，我们将系统分为展示层，逻辑层，数据层。每层之间有接口。为了增加每个类的内聚我们将其分为了两个类。为了隔离业务逻辑职责和业务控制职责，我们添加了MemberDataControl。

memberData模块设计：



MemberData模块各个类的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| MemberDataControl | 负责实现客户管理逻辑层所需要的服务 |
| Member | 客户管理数据层的领域模型对象，拥有一次客户管理所持有的客户信息，可以帮助完成客户管理界面所需服务 |

### （3）模块内部接口规范

MemberDataControl的接口规范：

|  |  |  |
| --- | --- | --- |
| **提供的服务（供借口）** | | |
| **memberDataControl.get** | **语法** | **public** ArrayList<MemberPO> get(String id, String name) |
| **前置条件** | **已创建一个Member和Unity领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberDataControl.del** | **语法** | **public** String del(String id, String name) |
| **前置条件** | **已创建一个Member和Unity领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberDataControl.insert** | **语法** | **public** String insert(MemberPO member) |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberDataControl.upd** | **语法** | **public** String upd(MemberPO member) |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberDataControl.payMoney** | **语法** | **public** String payMoney(String id, **double** b) |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberDataControl.getMoney** | **语法** | **public** String getMoney(String id, **double** b) |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **memberDataControl.getId** | **语法** | **public** String getId() |
| **前置条件** | **已创建一个Member领域对象，且输入符合规范** |
| **后置条件** | **调用Member领域的方法** |
| **需要的服务（需接口）** | | |
| **MemberData.** get(String id, String name) | **获得一个客户** | |
| **MemberData.** del(String id, String name) | **删除一个客户** | |
| **MemberData.** insert(MemberPO member) | **新增一个客户** | |
| **MemberData.** upd(MemberPO member) | **更新一个客户** | |
| **MemberData.** payMoney(String id, **double** b) | **减少该客户的应付金额** | |
| **MemberData.** getMoney(String id, **double** b) | **减少该客户的应收金额** | |
| **Unity.** getid() | **自动生成新客户的id** | |

MemberData的借口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供借口）** | | |
| **MemberData.** get(String id, String name) | **语法** | **public ArrayList<MemberPO>** get(String id, String name) |
| **前置条件** | **无** |
| **后置条件** | **根据PO查找是否存在该客户，返回所有包含关键字的客户** |
| **MemberData.** del(String id, String name) | **语法** | **public String MemberData.** del(String id, String name) |
| **前置条件** | **无** |
| **后置条件** | **根据PO查找是否存在该客户，有则删除用户** |
| **MemberData.** insert(MemberPO member) | **语法** | **public String**  insert(MemberPO member) |
| **前置条件** | **PO包含所有信息** |
| **后置条件** | **根据PO查找是否存在该客户，有则添加用户** |
| **MemberData.** upd(MemberPO member) | **语法** | **public String upd（MemberPO member）** |
| **前置条件** | **该用户存在，PO包含所有信息** |
| **后置条件** | **更新客户信息** |
| **MemberData.** payMoney(String id, **double** b) | **语法** | **public String** payMoney(String id, **double** b) |
| **前置条件** | **该用户存在，财务人员收了用户钱** |
| **后置条件** | **减少客户的应付金额** |
| **MemberData.** getMoney(String id, **double** b) | **语法** | **public String** getMoney(String id, **double** b) |
| **前置条件** | **该用户存在，财务人员付了用户钱** |
| **后置条件** | **减少客户的应收金额** |

Unity的借口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供借口）** | | |
| **Unity.getId** | **语法** | **public String getId();** |
| **前置条件** | **无** |
| **后置条件** | **根据保存的id创建新的客户id** |

### （4）业务逻辑层的动态模型

由于Memberdata层的顺序图均为control调用Memberdata后返回，或者调用unity后返回，较为简单，略去。

由于Memberdata层的状态图也十分简单，略去。

### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

## 4.2.4MarketData模块

### （1）模块概述

MarketData模块承担了需求参见需求规格说明文档的功能需求以及非功能需求。

### （2）整体结构

根据体系结构的设计，我们将系统分为展示层，逻辑层，数据层。每层之间有接口。为了隔离业务逻辑职责和业务控制职责，我们添加了MarketControl，这样就将客户管理的业务逻辑委托给Market对象。我们将生成id，初始化等功能分离出MarketData，增加了聚合。Market主要负责单据的增 和 查，Unity负责获取id及初始化

MarketData的模块设计：



MarketData模块各个类的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| MarketControl | 负责对应销售逻辑层所需的服务 |
| MarketData | 销售任务数据层的领域模型对象， 拥有销售任务的类型，商品的清单总价格这些信息，可以帮助完成销售页面所需服务 |
| Unity | 负责初始化，自动生成id |

### （3）模块内部的接口规范

MarketControl的接口规范：

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **MarketControl.** findSale | **语法** | **public** ArrayList<SaleListPO> findSale(String start, String end,  String thingid, String memberid, String salerid, String house) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的find方法** |
| **MarketControl.** findUnsale | **语法** | **public** ArrayList<UnsaleListPO> findUnsale(String start, String end,  String thingid, String memberid, String salerid, String house) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的find方法** |
| **MarketControl.** findStack | **语法** | **public** ArrayList<StackListPO> findStack(String start, String end,  String thingid, String memberid, String salerid, String house) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的find方法** |
| **MarketControl.** findUnstack | **语法** | **public** ArrayList<UnstackListPO> findUnstack(String start, String end,  String thingid, String memberid, String salerid, String house) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的find方法** |
| **MarketControl.** findGift | **语法** | **public** ArrayList<GiftListPO> findGift(String start, String end,  String thingid, String memberid, String salerid, String house) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的find方法** |
| **MarketControl.insertSale** | **语法** | **public** String insertSale(SaleListPO po,**int** ok) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的insert方法** |
| **MarketControl.insertUnsale** | **语法** | **public** String insertUnsale(UnsaleListPO po,**int** ok) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的insert方法** |
| **MarketControl.insertStack** | **语法** | **public** String insertStack(StackListPO po,**int** ok) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的insert方法** |
| **MarketControl.insertUnstack** | **语法** | **public** String insertUnstack(UnstackListPO po,**int** ok) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的insert方法** |
| **MarketControl.getId** | **语法** | **public** String getId(int i) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Unity的getId方法** |
| **MarketControl.check** | **语法** | **public int[] check** |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的check方法** |
| **MarketControl.showList** | **语法** | **public** ArrayList<MarketListPO> showList(**int** i) |
| **前置条件** | **已创建一个Market领域对象** |
| **后置条件** | **调用Market的showList方法** |
| **需要的接口（需接口）** | | |
| **MarketData.insert(T extend MarketListPO,int ok,int kind)** | **添加不同的销售单，并保存是否通过审批** | |
| **MarketData.** find(String start, String end,  String thingid, String memberid, String salerid, String house,int kind) | **查找不同种类的销售清单按照时间，仓库，销售员，客户，等不同的查找条件** | |
| **Unity.getId(int kind)** | **自动生成不同销售单的id** | |
| **MarketData.showList** | **查看刚通过审批的单据详情** | |
| **MarketData.check()** | **查看刚刚通过审批的单据有多少通过审批，多少没有通过审批，（0为通过，1为不通过）** | |

MarketData的接口规范：

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **MarketData.find** | **语法** | **public** <T **extends** MarketListPO> String start, String end,String thingid, String memberid, String salerid, String house,int kind) |
| **前置条件** | **无** |
| **后置条件** | **返回符合要求的销售清单** |
| **MarketData.insert** | **语法** | **private** <T **extends** MarketListPO> String insert(T m,int kind) |
| **前置条件** | **无** |
| **后置条件** | **将该种类的销售单加入刚通过单据中，如果通过了单据，将其保存** |
| **MarketData.check** | **语法** | **public int[] check()** |
| **前置条件** | **无** |
| **后置条件** | **返回刚通过审批的单据的数量，0为通过审批的单据数量，1为未通过的** |
| **MarketData.showList** | **语法** | public ArrayList<MarketListPO> showList() |
| **前置条件** | **无** |
| **后置条件** | **返回刚通过审批的单据的详细信息。** |

Unity接口规范

|  |  |  |
| --- | --- | --- |
| **Unity.getId(int kind)** | **语法** | **Public String getId(int kind)** |
| **前置条件** | **无** |
| **后置条件** | **返回生成的id，并且知道增加id计数** |

### （4） 业务逻辑层的动态模型

由于Memberdata层的顺序图均为control调用Memberdata后返回，或者调用unity后返回，较为简单，略去。

由于Memberdata层的状态图也十分简单，略去。

### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

## 4.2.5.paymentdata模块

模块内仅有三个类，分别对应payin，payout，paycash，见体系结构说明文档paymentindataservice，paymentoutdataservice，paymenticashdataservice

## 4.2.6.accountdata模块

模块内仅有一个类，见体系结构说明文档accountdataservice

## 4.2.7.promotiondata模块

### （1）模块概述

需求详见需求规格说明文档，职责与接口规范详见下表。

### （2）整体结构

promotiondata模块各个类的职责如下表所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| TotalMethodDataController | 负责实现制定总价额满优惠的对应具体总价额促销策略所需要的服务 |
| TotalCouponDataController | 负责实现制定总价额满优惠的代金券促销策略所需要的服务 |
| TotalGiftDataController | 负责实现制定总价额满优惠的赠品促销策略所需要的服务 |
| VipMethodDataController | 负责实现制定vip优惠的具体等级对应促销策略所需要的服务 |
| VipCouponDataController | 负责实现制定vip优惠的代金券促销策略所需要的服务 |
| VipGiftDataController | 负责实现制定vip优惠的赠品促销策略所需要的服务 |
| VipDiscountDataController | 负责实现制定vip优惠的折扣促销策略所需要的服务 |
| TotalMethodData | 总价额满优惠的关于各个总价额的领域模型对象，拥有各个总价额项对应的促销策略po对象，可以帮助完成制定总价额满优惠所需要的服务。 |
| TotalCouponData | 总价额满优惠的关于代金券的领域模型对象，拥有各个总价额项对应的代金券促销策略po对象，可以帮助完成制定总价额满优惠的代金券所需要的服务。 |
| TotalGiftData | 总价额满优惠的关于赠品的领域模型对象，拥有各个总价额项对应的赠品促销策略po对象，可以帮助完成制定总价额满优惠的赠品所需要的服务。 |
| VipMethodData | vip优惠的关于各个vip等级的领域模型对象，拥有各个vip等级对应促销策略po对象，可以帮助完成制定vip优惠所需要的服务。 |
| VipCouponData | vip优惠的关于代金券的领域模型对象，拥有各个vip等级对应的代金券促销策略po对象，可以帮助完成制定vip优惠的代金券所需要的服务。 |
| VipGiftData | vip优惠的关于赠品的领域模型对象，拥有各个vip等级对应的赠品促销策略po对象，可以帮助完成制定vip优惠的赠品所需要的服务。 |
| VipDiscountData | vip优惠的关于折扣的领域模型对象，拥有各个vip等级对应的折扣促销策略po对象，可以帮助完成制定vip优惠的折扣所需要的服务。 |

### （3）模块内部类的接口规范

**TotalMethodDataController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| TotalMethodDataController.updateTotalMethod | 语法 | | public String updateTotalMethod(double totalMoney, TotalPromotionPO po) |
| 前置条件 | | 已创建一个TotaMethodData领域对象 |
| 后置条件 | | 调用TotaMethodData领域对象的updateTotalMethod方法 |
| TotalMethodDataController.addTotalMethod | 语法 | | public String addTotalMethod(double totalMoney, TotalPromotionPO po) |
| 前置条件 | | 已创建一个TotaMethodData领域对象 |
| 后置条件 | | 调用TotaMethodData领域对象的addTotalMethod方法 |
| TotalMethodDataController.delTotalMethod | 语法 | | public String delTotalMethod(double totalMoaney) |
| 前置条件 | | 已创建一个TotaMethodData领域对象 |
| 后置条件 | | 调用TotaMethodData领域对象的delTotalMethod方法 |
| TotalMethodDataController.getTotalMethod | 语法 | | public TotalPromotionVO getTotalMethod(double totalMoney) |
| 前置条件 | | 已创建一个TotaMethodData领域对象 |
| 后置条件 | | 调用TotaMethodData领域对象的getTotalMethod方法 |
| TotalMethodDataController.getTotalMethod | 语法 | | public ArrayList<TotalPromotionPO> getTotalMethod() |
| 前置条件 | | 已创建一个TotaMethodData领域对象 |
| 后置条件 | | 调用TotaMethodData领域对象的getTotalCouppon方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| TotaMethodData.getTotalMethod(double totalMoney) | | 根据总价额项获得当前优惠策略 | |
| TotaMethodData.getTotalMethod() | | 取得总价额优惠列表 | |
| TotaMethodData.updateTotalMethod(double totalMoney, TotalPromotionPO po) | | 更新对应总价额项的优惠策略 | |
| TotaMethodData.addTotalMethod(double totalMoney, TotalPromotionPO po) | | 增加对应总价额项的优惠策略 | |
| TotaMethodData.delTotalMethod(double totalMoney) | | 删除对应总价额项的优惠策略 | |

**TotalCouponDataController的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| TotalCouponDataController.getTotalCoupon | 语法 | | | Public TotalCouponPO getTotalCoupon(int level); |
| 前置条件 | | | 已创建一个TotalCouponData领域对象 |
| 后置条件 | | | 调用TotalCouponData领域对象的getTotalCoupon方法 |
| TotalCouponDataController.getTotalCouponList | 语法 | | public ArrayList<TotalCouponPO>getTotalCouponList(); | |
| 前置条件 | | 已创建一个TotalCouponData领域对象 | |
| 后置条件 | | 调用TotalCouponData领域对象的getTotalCouponList方法 | |
| TotalCouponDataController.addTotalCoupon | 语法 | | public String addVipCoupon(double totalMoney,TotalCouponPO vo) | |
| 前置条件 | | 已创建一个TotalCouponData领域对象 | |
| 后置条件 | | 调用TotalCouponData领域对象的addTotalCoupon方法 | |
| TotalCouponDataController.delTotalCoupon | 语法 | | public String delTotalCoupon(double totalMoney) | |
| 前置条件 | | 已创建一个TotalCouponData领域对象 | |
| 后置条件 | | 调用TotalCouponData领域对象的delTotalCoupon方法 | |
| TotalCouponDataController.updateTotalCoupon | 语法 | | public String updateTotalCoupon(double totalMoney,TotalCouponPO po) | |
| 前置条件 | | 已创建一个TotalCouponData领域对象 | |
| 后置条件 | | 调用TotalCouponData领域对象的editTotalCoupoon方法 | |
| **需要的服务（需接口）** | | | | |
| 服务名 | | 服务 | | |
| TotalCouponData.getTotalCoupon(int totalMoney) | | 根据当前总价额获得代金券 | | |
| TotalCouponData.getTotalCouponList() | | 获得代金券列表 | | |
| TotalCouponData.addTotalCoupon(double totalMoney,TotalCouponPO po) | | 增加对应总价额的代金券 | | |
| TotalCouponData.delTotalCoupon(TotalCouponPO po); | | 删除对应总价额的代金券 | | |
| TotalCouponData.updateTotalCoupon(double totalMoney,TotalCouponPO po); | | 编辑对应总价额的代金券 | | |

**TotalGiftDataController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| TotalGiftDataController.getTotalGift | 语法 | | public ArrayList<TotalGiftPO> getTotalGift(double totalMoney) |
| 前置条件 | | 已创建一个TotalGiftData领域对象 |
| 后置条件 | | 调用TotalGiftData领域对象的getTotalGift方法 |
| TotalGiftDataController.addTotalGift | 语法 | | public String addTotalGift(double totalMoney,TotalGiftPO po); |
| 前置条件 | | 已创建一个TotalGiftData领域对象 |
| 后置条件 | | 调用TotalGiftData领域对象的addTotalGift方法 |
| TotalGiftDataController.updateTotalGift | 语法 | | public String updateTotalGift(double totalMoney,TotalGiftPO po); |
| 前置条件 | | 已创建一个TotalGiftData领域对象 |
| 后置条件 | | 调用TotalGiftData领域对象的editTotalGift方法 |
| TotalGiftDataController.delTotalGift | 语法 | | public String delTotalGift(double totalMoney,TotalGiftPO po); |
| 前置条件 | | 已创建一个TotalGiftData领域对象 |
| 后置条件 | | 调用TotalGiftData领域对象的delTotalGift方法 |
| TotalGiftDataController.delTotalGiftByMoney | 语法 | | public String delTotalGiftByMoney (double money); |
| 前置条件 | | 已创建一个TotalGiftData领域对象 |
| 后置条件 | | 调用TotalGiftData领域对象的delTotalGiftByMoney方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| TotalGiftData.getTotalGift(double totalMoney) | | 根据当前总价额获得赠品 | |
| TotalGiftData.updateTotalGift(double totalMoney,TotalGiftPO po) | | 更新对应总价额的赠品 | |
| TotalGiftData.addTotalGift(double totalMoney,TotalGiftPO po) | | 增加对应总价额的赠品 | |
| TotalGiftData.delTotalGift(double totalMoney,TotalGiftPO po) | | 删除对应的赠品 | |
| TotalGiftData.delTotalGiftByMoney (double money) | | 删除对应总价额的赠品 | |

**VipMethodDataController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipMethodDataController.updateVipMethod | 语法 | | public String updateTotalMethod(int level,VipPromotionPO po) |
| 前置条件 | | 已创建一个VipMethodData领域对象 |
| 后置条件 | | 调用VipMethodData领域对象的updateVipMethod方法 |
| VipMethodDataController.getVipMethod | 语法 | | public VipPromotionPO getVipMethod(int vipLevel) |
| 前置条件 | | 已创建一个VipMethodData领域对象 |
| 后置条件 | | 调用VipMethodData领域对象的getVipMethod方法 |
| VipMethodDataController.getVipMethod | 语法 | | public ArrayList<VipPromotionPO> getVipMethod() |
| 前置条件 | | 已创建一个VipMethodData领域对象 |
| 后置条件 | | 调用VipMethodData领域对象的getVipCouppon方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipMethodData.getVipMethod(int level) | | 根据VIP等级获得当前优惠策略 | |
| VipMethodData.getVipMethod() | | 根据VIP等级获得当前优惠策略列表 | |
| VipMethodData.updateVipMethod(int level,VipPromotionVO vo) | | 更新对应VIP等级的优惠策略 | |

**VipCouponDataController的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| VipCouponDataController.getVipCoupon | 语法 | | | public ArrayList<VipCouponPO>getVipCoupon(int level); |
| 前置条件 | | | 已创建一个VipCouponData领域对象 |
| 后置条件 | | | 调用VipCouponData领域对象的getVipCoupon方法 |
| VipCouponDataController.getVipCouponList | 语法 | | public ArrayList<VipCouponPO>getVipCouponList(); | |
| 前置条件 | | 已创建一个VipCouponData领域对象 | |
| 后置条件 | | 调用VipCouponData领域对象的getVipCouponList方法 | |
| VipCouponDataController.addVipCoupon | 语法 | | public String addVipCoupon(int level,VipCouponPO po) | |
| 前置条件 | | 已创建一个VipCouponData领域对象 | |
| 后置条件 | | 调用VipCouponData领域对象的addVipCoupon方法 | |
| VipCouponDataController.delVipCoupon | 语法 | | public String delVipCoupon(int level) | |
| 前置条件 | | 已创建一个VipCouponData领域对象 | |
| 后置条件 | | 调用VipCouponData领域对象的delVipCoupon方法 | |
| VipCouponDataController.updateVipCoupon | 语法 | | public String updateVipCoupon(int level,VipCouponPO po) | |
| 前置条件 | | 已创建一个VipCouponData领域对象 | |
| 后置条件 | | 调用VipCouponData领域对象的editVipCoupoon方法 | |
| **需要的服务（需接口）** | | | | |
| 服务名 | | 服务 | | |
| VipCouponData.getVipCoupon(int level) | | 根据当前VIP等级获得代金券 | | |
| VipCouponData.getVipCoupon() | | 获得所有等级的代金券列表 | | |
| VipCouponData.updateVipCoupon(int level,VipCouponPO po) | | 更新对应VIP等级的代金券 | | |
| VipCouponData.addVipCoupon(int level,VipCouponPO po) | | 增加对应VIP等级的代金券 | | |
| VipCouponData.delVipCoupon(int level) | | 删除对应VIP等级的代金券 | | |

**VipGiftDataController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipGiftDataController.getVipGift | 语法 | | public ArrayList<VipGiftPO> getVipGift(int level) |
| 前置条件 | | 已创建一个VipGiftData领域对象 |
| 后置条件 | | 调用VipGiftData领域对象的getVipGift方法 |
| VipGiftDataController.addVipGift | 语法 | | public String addVipGift(int level,VipGiftPO po); |
| 前置条件 | | 已创建一个VipGiftData领域对象 |
| 后置条件 | | 调用VipGiftData领域对象的addVipGift方法 |
| VipGiftDataController.updateVipGift | 语法 | | public String updateVipGift(int level,VipGiftPO po); |
| 前置条件 | | 已创建一个VipGiftData领域对象 |
| 后置条件 | | 调用VipGiftData领域对象的editVipGift方法 |
| VipGiftDataController.delVipGift | 语法 | | public String delVipGift(int level,VipGiftPO po); |
| 前置条件 | | 已创建一个VipGiftData领域对象 |
| 后置条件 | | 调用VipGiftData领域对象的delVipGift方法 |
| VipGiftDataController.delVipGiftByLevel | 语法 | | public String delVipGiftByLevel (int level); |
| 前置条件 | | 已创建一个VipGiftData领域对象 |
| 后置条件 | | 调用VipGiftData领域对象的delVipGiftByLevel方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipGiftData.getVipGift(int level); | | 根据VIP等级获得当前赠品 | |
| VipGiftData.updateVipGift(int level,VipGiftPO po) | | 更新对应VIP等级的赠品 | |
| VipGiftData.addVIPGift(int level,VipGiftPO po) | | 增加对应VIP等级的赠品 | |
| VipGiftData.delVIPGift(int level,VipGiftPO po) | | 删除对应的赠品 | |
| VipGiftData.delVIPGiftByLevel(int level) | | 删除对应VIP等级的赠品 | |

**VipDiscountDataController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipDiscountDataController.getDiscount | 语法 | | public ArrayList<DiscountPO> getDiscount(int level) |
| 前置条件 | | 已创建一个VipDiscountData领域对象 |
| 后置条件 | | 调用VipDiscountData领域对象的getDiscount方法 |
| VipDiscountDataController.getDiscount | 语法 | | public DiscountPO getDiscount(int vipLevel) |
| 前置条件 | | 已创建一个VipDiscountData领域对象 |
| 后置条件 | | 调用VipDiscountData领域对象的getDiscount方法 |
| VipDiscountDataController.addDiscount | 语法 | | public String addDiscount(DiscountPO po) |
| 前置条件 | | 已创建一个VipDiscountData领域对象 |
| 后置条件 | | 调用VipDiscountData领域对象的addDiscount方法 |
| VipDiscountDataController.delDiscount | 语法 | | public String delDiscount(int level) |
| 前置条件 | | 已创建一个VipDiscountData领域对象 |
| 后置条件 | | 调用VipDiscountData领域对象的delDiscount方法 |
| VipDiscountDataController.editDiscount | 语法 | | public String editDiscount(DiscountPO po) |
| 前置条件 | | 已创建一个VipDiscountData领域对象 |
| 后置条件 | | 调用VipDiscountData领域对象的editDiscount方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| VipDiscountData.getDiscount(int level) | | 取得对应VIP等级的折扣 | |
| VipDiscountData.getDiscount() | | 取得各个VIP等级的折扣列表 | |
| VipDiscountData.addDiscount(DiscountPO po) | | 增加折扣 | |
| VipDiscountData.delDiscount(int level) | | 删除折扣 | |
| VipDiscountData.editDiscount(DiscountPO po) | | 编辑折扣 | |

**TotalMethodData的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| TotalMethodData.updateTotalMethod | 语法 | public String updateTotalMethod(double totalMoney, TotalPromotionPO po) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 改变对应TotalPromotionPO |
| TotalMethodData.addTotalMethod | 语法 | public String addTotalMethod(double totalMoney, TotalPromotionPO po) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 增加对应TotalPromotionPO |
| TotalMethodData.delTotalMethod | 语法 | public String delTotalMethod(double totalMoaney) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 删除对应TotalPromotionPO |
| TotalMethodData.getTotalMethod | 语法 | public TotalPromotionPO getTotalMethod(double totalMoney) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 返回对应TotalPromotionPO |
| TotalMethodData.getTotalMethod | 语法 | public ArrayList<TotalPromotionPO> getTotalMethod() |
| 前置条件 | 无 |
| 后置条件 | 返回TotalPromotionPO列表 |

**TotalCouponData的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| TotalCouponData.getTotalCoupon | 语法 | | Public TotalCouponPO getTotalCoupon(int level); |
| 前置条件 | | 输入有效信息 |
| 后置条件 | | 返回对应TotalCouponPO |
| TotalCouponData.getTotalCouponList | 语法 | public ArrayList<TotalCouponPO>getTotalCouponList(); | |
| 前置条件 | 无 | |
| 后置条件 | 返回TotalCouponPO列表 | |
| TotalCouponData.addTotalCoupon | 语法 | public String addVipCoupon(double totalMoney,TotalCouponPO po) | |
| 前置条件 | 输入有效信息 | |
| 后置条件 | 增加对应TotalCouponPO | |
| TotalCouponData.delTotalCoupon | 语法 | public String delTotalCoupon(double totalMoney) | |
| 前置条件 | 输入有效信息 | |
| 后置条件 | 删除对应TotalCouponPO | |
| TotalCouponData.updateTotalCoupon | 语法 | public String updateTotalCoupon(double totalMoney,TotalCouponPO po) | |
| 前置条件 | 输入有效信息 | |
| 后置条件 | 改变对应TotalCouponPO | |

**TotalGiftData的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| TotalGiftData.getTotalGift | 语法 | public ArrayList<TotalGiftPO> getTotalGift(double totalMoney) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 返回TotalGiftPO列表 |
| TotalGiftData.addTotalGift | 语法 | public String addTotalGift(double totalMoney,TotalGiftPO po); |
| 前置条件 | 输入有效信息 |
| 后置条件 | 增加TotalGiftPO |
| TotalGiftData.upateTotalGift | 语法 | public String updateTotalGift(double totalMoney,TotalGiftPO po); |
| 前置条件 | 输入有效信息 |
| 后置条件 | 改变对应TotalGiftPO |
| TotalGiftData.delTotalGift | 语法 | public String delTotalGift(double totalMoney,TotalGiftPO po); |
| 前置条件 | 输入有效信息 |
| 后置条件 | 删除对应TotalGiftPO |
| TotalGiftData.delTotalGiftByMoney | 语法 | public String delTotalGiftByMoney (double money); |
| 前置条件 | 输入有效信息 |
| 后置条件 | 删除对应TotalGiftPO |

**VipPMethodData的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| VipMethodData.updateVipMethod | 语法 | public String updateTotalMethod(int level,VipPromotionPO po) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 改变对应VipPromotionPO |
| VipMethodData.getVipMethod | 语法 | public VipPromotionPO getVipMethod(int vipLevel) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 返回对应VipPromotionPO |
| VipMethodData.getVipMethod | 语法 | public ArrayList<VipPromotionPO> getVipMethod() |
| 前置条件 | 输入有效信息 |
| 后置条件 | 返回VipPromotionPO列表 |

**VipCouponData的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| VipCouponData.getVipCoupon | 语法 | | public VipCouponPO getVipCoupon(int level); |
| 前置条件 | | 输入有效信息 |
| 后置条件 | | 返回对应VipCouponPO |
| VipCouponData.getVipCouponList | 语法 | public ArrayList<VipCouponPO>getVipCouponList(); | |
| 前置条件 | 无 | |
| 后置条件 | 返回VipCouponPO列表 | |
| VipCouponData.addVipCoupon | 语法 | public String addVipCoupon(int level,VipCouponPO po) | |
| 前置条件 | 输入有效信息 | |
| 后置条件 | 增加对应VipCouponPO | |
| VipCouponData.delVipCoupon | 语法 | public String delVipCoupon(int level) | |
| 前置条件 | 输入有效信息 | |
| 后置条件 | 删除对应VipCouponPO | |
| VipCouponData.updateVipCoupon | 语法 | public String updateVipCoupon(int level,VipCouponPO po) | |
| 前置条件 | 输入有效信息 | |
| 后置条件 | 改变对应VipCouponPO | |

**VipGiftData的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| VipGiftData.getVipGift | 语法 | public ArrayList<VipGiftPO> getVipGift(int level) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 返回对应VipGiftPO列表 |
| VipGiftData.addVipGift | 语法 | public String addVipGift(int level,VipGiftPO po); |
| 前置条件 | 输入有效信息 |
| 后置条件 | 增加对应VipGiftPO |
| VipGiftData.editVipGift | 语法 | public String editVipGift(int level,VipGiftPO po); |
| 前置条件 | 输入有效信息 |
| 后置条件 | 改变对应VipGiftPO |
| VipGiftData.delVipGift | 语法 | public String delVipGift(int level,VipGiftPO po); |
| 前置条件 | 输入有效信息 |
| 后置条件 | 删除对应VipGiftPO |
| VipGiftData.delVipGiftByLevel | 语法 | public String delVipGiftByLevel (int level); |
| 前置条件 | 输入有效信息 |
| 后置条件 | 删除对应VipGiftPO |

**VipDiscountData的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| VipDiscountData.getDiscount | 语法 | public ArrayList<DiscountPO> getDiscount() |
| 前置条件 | 无 |
| 后置条件 | 返回DiscountPO列表 |
| VipDiscountData.getDiscount | 语法 | public DiscountPO getDiscount(int vipLevel) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 返回对应DiscountPO |
| VipDiscountData.addDiscount | 语法 | public String addDiscount(DiscountPO po) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 增加对应DiscountPO |
| VipDiscountData.delDiscount | 语法 | public String delDiscount(int level) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 删除对应DiscountPO |
| VipDiscountData.editDiscount | 语法 | public String editDiscount(DiscountPO po) |
| 前置条件 | 输入有效信息 |
| 后置条件 | 改变对应DiscountPO |

### （4） 业务逻辑层的动态模型

由于Promotiondata层的顺序图均为controller调用对应data对象后返回，较为简单，略去。

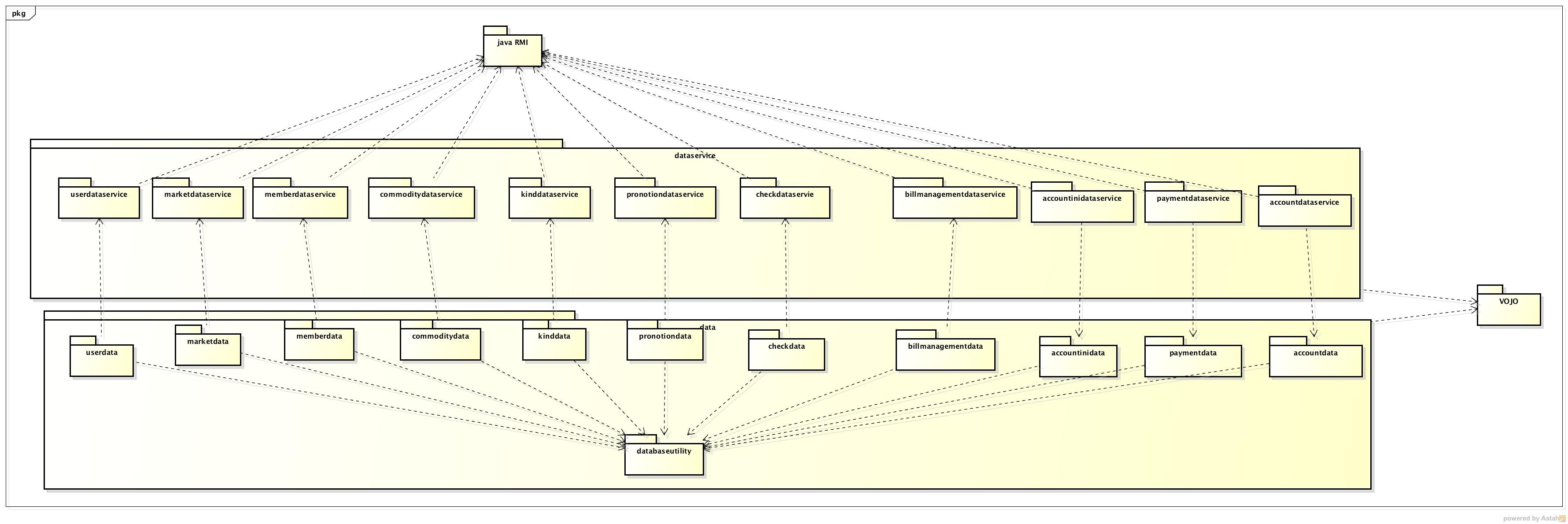
由于Promotiondata层的状态图也十分简单，略去。

### （5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

# 5.依赖视角

**服务器端包图：**



**客户端包图：**

