进销存系统

软件详细设计描述文档

**V2.2正式版**

南京大学软件学院

吮指黄金柠檬脆皮鸡工作组

王宁(79)黄涵倩金翠严顺宽

2014-11-2

**目录**

[更新历史](#_更新历史)………………………………………………………………………………3

[1、引言](#_1、引言)……………………………………………………………………………….3

1.1编制目的……………………………………………………………………3

1.2词汇表………………………………………………………………………3

1.3参考目录……………………………………………………………………3

[2、产品概述](#_2、产品概述)…………………………………………………………………………4

[3、体系结构设计概述](#_3、体系结构设计概述)………………………………………………………………4

[4、中层设计](#_4、中层设计)…………………………………………………………………………4

[4.1客户端展示层模块的静态结构和动态职责](#_4.1客户端展示层模块的静态结构和动态职责)………………………………5

4.1.1客户端展示层模块局部模块的职责…………………………………5

4.1.2客户端展示层模块局部模块的接口规范……………………………6

4.1.3客户端展示层模块的设计原理………………………………………8

[4.2客户端业务逻辑层模块的静态结构和动态职责](#_4.2客户端业务逻辑层模块的静态结构和动态职责)…………………………8

4.2.1客户端业务逻辑层模块局部模块的职责……………………………8

4.2.2客户端业务逻辑层模块局部模块的接口规范及局部行为…………8

4.2.3客户端业务逻辑层模块的设计原理………………………………62

[4.3客户端网络模块的静态结构和动态职责](#_4.3客户端网络模块的静态结构和动态职责)…………………………………62

4.3.1客户端网络模块局部模块的职责………………………………62

[4.4服务器端网络模块的静态结构和动态职责](#_4.4服务器端网络模块的静态结构和动态职责)……………………………63

4.4.1服务器端网络模块局部模块的职责………………………………63

4.4.2服务器端网络模块局部模块的接口规范…………………………64

[4.5服务器端数据层模块的静态结构和动态职责](#_4.5服务器端数据层模块的静态结构和动态职责)…………………………65

4.5.1服务器端数据层模块局部模块的职责……………………………65

4.5.2服务器端数据层模块局部模块的接口规范………………………65

4.5.3服务器算数据层模块的设计原理…………………………………71

[5、依赖关系](#_5、依赖关系)…………………………………………………………………………71

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 黄涵倩严顺宽王宁(79)金翠 | 2014/10/28 | 各模块内容合并 | V1.0 |
| 黄涵倩严顺宽 | 2014/10/31 | 修改期初建账接口规范 | V1.1 |
| 王宁(79) | 2014/11/1 | 修改Member对象的状态图的跳转事件 | V1.2 |
| 黄涵倩 | 2014/11/2 | 修改Finance.BSLBL,Finance.initBL,Finance.listBL类结构设计图的部分类的关系 | V2.0 |
| 黄涵倩 | 2014/11/8 | 修改FinanceAccountBL模块find方法的传入参数 | V2.1 |
| 黄涵倩,金翠 | 2014/11/13 | Receiptbl类设计图重构，补充ReceiptController接口规范,统一文档样式 | V2.2正式版 |

# 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** |
| bl | 业务逻辑（business logic） | 无 |
|  |  |  |

**1.3 参考资料**

[Bin2012]骆斌，丁二玉，刘钦.软件工程与计算（卷二）——软件开发的技术基础.北京：机械工程出版社，2012.12;

进销存系统用例文档、进销存系统软件需求规格说明和进销存系统体系结构设计文档。

# 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中对产品的概括描述。

# 3、体系结构设计概述

参考进销存系统体系结构设计文档中对体系结构设计的概述。

# 4、中层设计

中层设计的基本思路依托了软件体系的结构视角。

在高校选课系统中，可以分别在客户端和服务器端都添加系统启动模块专门负责系统的初始化启动工作，并使它横跨系统的分层结构，可以直接访问到各层。客户端则具体分为展示层、业务逻辑层和网络模块，而服务器端则包含网络模块和数据层。具体的客户端和服务器端模块视图分别如图1和图2所示。客户端各层和服务器端各层的职责分别如表1和表2所示。



**图1 客户端模块视图**



**图2 服务器端模块视图**

**表1 客户端各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 用户界面层 | 基于窗口的进销存客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

**表2 服务器端各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。

下面，根据以上对客户端和服务器的分解对各个模块进行分析与设计。

# 4.1客户端展示层模块的静态结构和动态职责

客户端展示层由八个模块组成，分别是mainui, userui, memberui,receiptui, promotionui, financeui, stockui,salesui。mainui模块负责界面的跳转；userui模块包含用户登录,修改密码以及增删改用户的界面。memberui模块包含客户的增删改查界面。receiptui模块包含单据审批与查看经营历程表界面。promotionui模块包含制定促销策略的界面。financeui模块包含财务管理各项业务，查看经营情况表和期初建账界面。salesui模块包含销售进货各项业务，以及销售明细表界面。stockui包含商品及其分类管理，库存报溢，报损，报警，及库存赠送的界面

**4.1.1客户端展示层模块局部模块的职责**

客户端展示层模块的职责如下表所示。

**表3客户端展示层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| mainui | 负责界面的跳转 |
| Userui | 用户登录,修改密码以及增删改用户的界面 |
| Memberui | 客户的增删改查界面 |
| Receiptui | 单据审批与查看经营历程表界面 |
| Promotionui | 制定促销策略的界面 |
| Financeui | 财务管理各项业务与查看经营情况表界面 |
| stockui | 商品及其分类管理，库存报溢，报损，报警，及库存赠送的界面 |
| salesui | 销售进货各项业务以及查看销售明细表界面 |

**4.1.2客户端展示层模块局部模块的接口规范**

**4.1.2.1 mainui模块的接口规范**

mainui模块的接口规范如下表所示。

**表4 mainui模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的接口（供接口）** | | |
| mainui.init | 语法 | public void init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 根据逻辑进行界面跳转。 |

**4.1.2.2 userui模块的接口规范**

userui模块的接口规范如下表所示。

**表5 userui模块的接口规范**

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| **服务名** | **服务** |
| businesslogicservice.UserBLService | 登录界面的业务逻辑（身份验证）接口  用户的增删改查的业务逻辑接口 |

**4.1.2.3memberui模块的接口规范**

memberui模块的接口规范如下表所示。

**表6memberui模块的接口规范**

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| **服务名** | **服务** |
| businesslogicservice.MemberBLService | 客户的增删改查的业务逻辑接口 |

**4.1.2.4receiptui模块的接口规范**

receiptui模块的接口规范如下表所示。

**表7receiptui模块的接口规范**

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| **服务名** | **服务** |
| Businesslogicservice.ReceiptBLService | 单据审批和单据增改查的业务逻辑接口 |

**4.1.2.5promotionui模块的接口规范**

promotionui模块的接口规范如下表所示。

**表8promotionui模块的接口规范**

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| **服务名** | **服务** |
| businesslogicservice.promotionBLService | 制定促销策略的业务逻辑接口 |

**4.1.2.6financeui模块的接口规范**

financeui模块的接口规范如下表所示。

**表9 financeui模块的接口规范**

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| **服务名** | **服务** |
| Businesslogicservice.financeBLService.listblservice | 制定收付款单，现金费用单的业务逻辑接口 |
| Businesslogicservice.financeBLService.accountblservice | 账户管理的业务逻辑接口 |
| Businesslogicservice.financeBLService.BSLblservice | 查看经营情况表的业务逻辑接口 |
| Businesslogicservice.financeBLService.intiblservice | 期初建账的业务逻辑接口 |

**4.1.2.7stockui模块的接口规范**

stockui模块的接口规范如下表所示。

**表10 stockui模块的接口规范**

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| **服务名** | **服务** |
| businesslogicservice.stockBLService.goodsblservice | 商品增删改查的业务逻辑接口 |
| businesslogicservice.stockBLService.goodsclassblservice | 商品分类增删改查的业务逻辑接口 |
| businesslogicservice.stockBLService.controlblservice | 库存报溢，报损，报警的业务逻辑接口 |
| businesslogicservice.stockBLService.giftblservice | 库存赠送的业务逻辑接口 |

**4.1.2.8salesui模块的接口规范**

salesui模块的接口规范如下表所示。

**表11 salesui模块的接口规范**

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| **服务名** | **服务** |
| businesslogicservice.salesBLService | 制定进货单，进货退货单，销售单，销售退货单，查看销售明细表的业务逻辑接口 |

**4.1.3客户端展示层模块的设计原理**

客户端展示层模块使用了集中式的控制风格。每一个界面的职责从mainui之中分离并通过mainui模块实现跳转。

# 4.2客户端业务逻辑层模块的静态结构和动态职责

客户端业务逻辑层由7个模块组成，分别是userbl, membersbl, receiptbl, promotionbl, financebl, stockbl, salesbl。userbl模块负责用户的登录以及修改密码的安全性需求及系统用户的增删改查；memberbl模块负责销售人员对客户的增删改查。receiptbl模块负责总经理的单据审批和查看经营历程表以及财务人员的红冲单据。promotionbl负责促销策略的制定，修改，查看。financebl模块负责财务人员制定收付款单，现金费用单，期初建账，账户管理，查看经营情况表。stockbl负责库存管理人员对商品及其分类管理，库存报溢，报损，报警，及库存赠送的处理。salesbl负责销售人员对销售单，销售退货单，进货单，进货退货单的制定和销售明细表查看。

**4.2.1客户端业务逻辑层模块局部模块的职责**

客户端业务逻辑层层模块的职责如下表所示。

**表12客户端业务逻辑层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| userbl | 负责实现对应于用户登录，管理员增删改查用户界面所需要的服务 |
| memberbl | 负责实现对应于会员增删改查所需要的服务 |
| receiptbl | 负责实现所有单据的审批，修改，红冲，查看经营历程表的各项单据业务所需的服务 |
| promotionbl | 负责实现对应于促销策略制定，修改，发布，查看界面所需的服务 |
| financebl | 负责实现所有对应于创建收付款单，现金费用单，账户管理，期初建账，查看经营情况表所需要的服务 |
| salesbl | 负责实现所有关于进货，进货退货，销售，销售退货，查看销售明细表界面所需要的服务 |
| stockbl | 负责实现所有关于商品及其分类管理,库存报溢,报损，报警及库存赠送界面所需要的服务 |

**4.2.2客户端业务逻辑层模块局部模块的接口规范及局部行为**

**4.2.2.1 receiptbl模块**

1. **模块概述**

receiptbl模块实现了审批，修改，红冲复制单据和查看经营历程表的功能具体功能需求和非功能需求请参见需求规格说明文档和体系结构设计文档。

**(2)整体结构**

receiptbl模块实现了展示层与业务逻辑层之间的receiptblservice接口以及业务逻辑层和数据层之间的receiptdataservice接口。提供了对单据的Add,Modify,Approve,Send,Reply,Batch,View,RedExtru方法。ReceiptController隔离业务逻辑职责和逻辑控制职责，将对单据审批的业务逻辑处理委托给Review对象。MemberInfo和SaleInfo都是依据依赖倒置原则，为消除循环依赖而产生的接口。receiptbl关键类的属性如图3,4所示

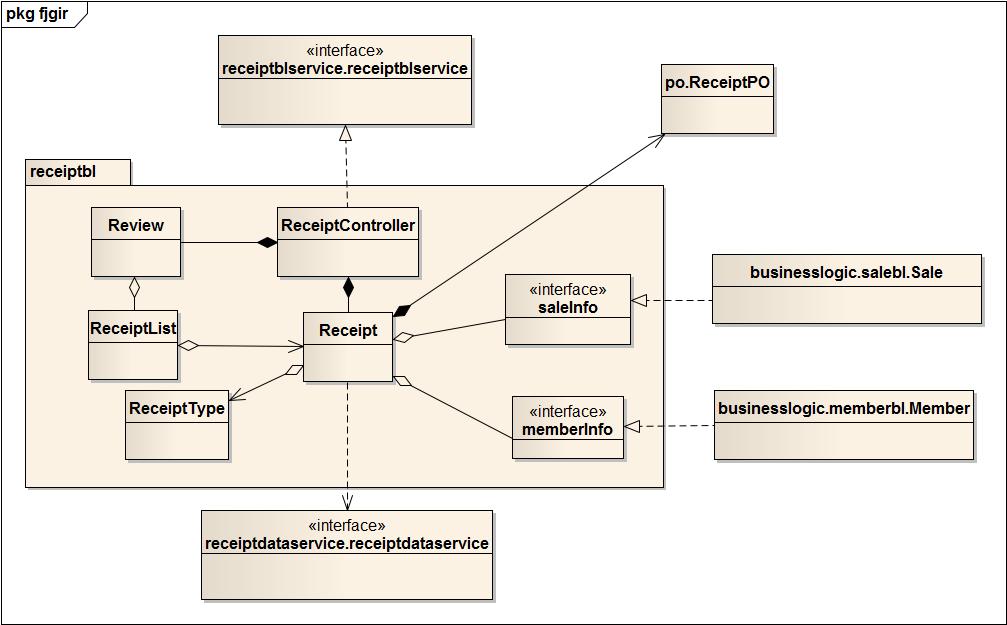
,

**图3 receipt关键类属性**



**图4review类的关键类属性**

receiptbl模块各个类的设计如图5所示

****

**图5 receiptbl各个类的设计**

**（3）模块内部类的接口规范**

ReceiptController和receipt,review的接口规范如表所示

**表13 receiptController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| ReceiptController.modifyReceipt | **语法** | | Public int modifyReceipt(ReceiptVO vo) |
| **前置条件** | | 已创建一个Receipt领域对象,且有单据可修改 |
| **后置条件** | | 调用Receipt的modify方法 |
| ReceiptController.approveReceipt | **语法** | | Public int approveReceipt(String id) |
| **前置条件** | | 已创建一个Receipt领域对象 |
| **后置条件** | | 调用Review的Approve方法 |
| ReceiptController.SendReceipt | **语法** | | Public void sendReceipt(ReceiptVO vo) |
| **前置条件** | | 已创建一个Receipt领域对象,且输入完整，符合规则 |
| **后置条件** | | 调用Review的Send方法 |
| ReceiptController.viewReceipt | **语法** | | Public ReceiptVO viewReceipt(String id) |
| **前置条件** | | 已创建一个Receipt领域对象,且id输入符合输入规则 |
| **后置条件** | | 调用Review的view方法 |
| ReceiptController.showAll | **语法** | | Public ArrayList<ReceiptVO> showAll() |
| **前置条件** | | 已创建一个Receipt领域对象 |
| **后置条件** | | 调用Review的view方法 |
| ReceiptController.redExtru | **语法** | | Public ReceiptVO redExtru(String id,int isCopy) |
| **前置条件** | | 已创建一个Receipt领域对象,输入合法 |
| **后置条件** | | 调用Review的RedExtru方法 |
| ReceiptController.Batch | **语法** | | Public int Batch(String[] id) |
| **前置条件** | | 已创建一个Receipt领域对象,当前有可审批单据 |
| **后置条件** | | 调用Review的Batch方法 |
| ReceiptController.Reply | **语法** | | Public void Reply(String id) |
| **前置条件** | | 已创建一个Receipt领域对象,且有单据审批完成 |
| **后置条件** | | 调用Receipt的Reply方法 |
| ReceiptController.refresh | **语法** | | Public ArrayList<ReceiptVO> refresh() |
| **前置条件** | | 已创建一个Receipt领域对象 |
| **后置条件** | | 调用Review的refresh方法 |
| ReceiptController.export | **语法** | | Public void export() |
| **前置条件** | | 已创建一个Receipt领域对象,当前单据列表有单据存在 |
| **后置条件** | | 调用ReceiptList的export方法 |
| **需要的服务(需接口)** | | | |
| **服务名** | | **服务** | |
| ReView.Approve(String id) | | 审批一条单据 | |
| Review.Batch(String[] id) | | 批量审批单据 | |
| Review.view(String id) | | 查看一条单据的详细信息 | |
| Review.view() | | 查看所有单据 | |
| Review.RedExtru(Stringid,int isCopy) | | 红冲或红冲并复制一条单据 | |
| Review.Send(ReceiptVOvo) | | 发送一条单据信息 | |
| Review.refresh() | | 更新显示单据列表信息 | |
| Receipt.Modify(ReceiptVO vo) | | 修改一条单据的信息 | |
| Receipt.Reply(String id) | | 返送单据审批结果 | |
| ReceiptList.export() | | 导出所有已审批单据列表 | |

表14 receipt的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Receipt.Add | **语法** | Public int Add(ReceiptVo vo) | |
| **前置条件** | User经验证，可以创建单据 | |
| **后置条件** | 向待审批单据列表库中添加该单据 | |
| Receipt.Modify | **语法** | Public int Modify(String id) | |
| **前置条件** | User选中了有效单据 | |
| **后置条件** | 更新该单据信息 | |
| Receipt.Reply | **语法** | Public void Reply(String id) | |
| **前置条件** | 该单据审批完成 | |
| **后置条件** | 该单据制单人收到审批结果信息 | |
| **需要的服务(需接口)** | | | |
| **服务名** | | | **服务** |
| ReceiptDataService.Add(ReceiptPo po) | | | 插入单一持久化对象 |
| ReceiptDataService.Modify(ReceiptPo po) | | | 更新单一持久化对象 |
| FileFactory.getReceiptData | | | 得到Receipt文件的引用 |

**表15 review的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| review.Approve | **语法** | Public int approve(String id) | |
| **前置条件** | 有单据提交待审批 | |
| **后置条件** | 更新该单据信息 | |
| review.View | **语法** | Public ArrayList<ReceiptVO> view() | |
| **前置条件** | 已有单据记录 | |
| **后置条件** | 返回所有单据信息 | |
| review.Send | **语法** | Public void Send(ReceiptVo vo) | |
| **前置条件** | 该业务单据创建成功 | |
| **后置条件** | 持久化更新涉及的领域对象的数据 | |
| review.Batch | **语法** | Public int Batch(String[] id) | |
| **前置条件** | 当前有待审批单据，且user已选中单据 | |
| **后置条件** | 更新该单据组信息 | |
| review.RedExtru | **语法** | Public ReceiptVO RedExtru(String id ,int isCopy) | |
| **前置条件** | User已经验证，且有可红冲单据 | |
| **后置条件** | 返回红冲并复制的新单据信息 | |
| review.Refresh | **语法** | Public ArrayList<ReceiptVO> refresh() | |
| **前置条件** | 已有单据记录 | |
| **后置条件** | 返回所有待审批单据信息 | |
| **需要的服务(需接口)** | | | |
| **服务名** | | | 服务 |
| ReceiptDataService.Modify(ReceiptPo po) | | | 更新单一持久化对象 |
| ReceiptDataService.Modify(ReceiptPo[] po) | | | 更新单一持久化对象组 |
| ReceiptDataService.find(Type type) | | | 根据类型type查找ReceiptPo对象 |
| ReceiptDataService.find(Date date) | | | 根据日期date查找ReceiptPo对象 |
| ReceiptDataService.Show() | | | 得到所得的ReceiptPo对象 |
| FileFactory.getReceiptData | | | 得到Receipt文件的引用 |

表16 ReceiptList的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ReceiptList.Add | **语法** | Public int Add(Receipt vo) | |
| **前置条件** | User经验证，可以创建单据 | |
| **后置条件** | 向待审批单据列表库中添加该单据 | |
| ReceiptList.export | **语法** | Public int Export() | |
| **前置条件** | User选中了有效单据 | |
| **后置条件** | 更新该单据信息 | |
| ReceiptList.find | **语法** | Public ReceiptPO findById(String id) | |
| **前置条件** | User经验证，并启动一个单据审批回合 | |
| **后置条件** | 返回对应id的单据信息 | |
| **需要的服务(需接口)** | | | |
| **服务名** | | | **服务** |
| ReceiptDataService.find(String id) | | | 根据类型id查找ReceiptPo对象 |
| ReceiptDataService.Add(ReceiptPo po) | | | 插入单一持久化对象 |
| FileFactory.getReceiptData | | | 得到Receipt文件的引用 |

**(4)业务逻辑层的动态模型**

如图6所示的顺序图表明了在进销存系统中，当总经理选择要审批的单据后，单据审批业务逻辑处理的相关对象之间的协作

****

**图6 receiptbl模块单据审批顺序图**

如图7所示，receipt对象状态图描述了receipt对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着viewReceipt或batch方法被 UI调用，Receipt进入receiptList状态；之后通过选中单据查看详细或修改单据进入receiptContent状态。随后，随着Approve方法被UI调用，Receipt回到receiptList状态。



**图7receiptbl模块单据审批状态图**

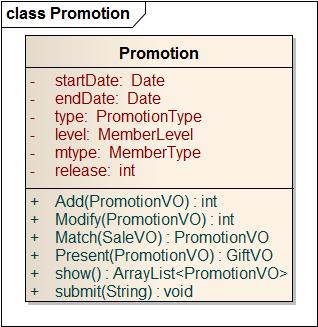
**4.2.2.2promotionbl模块**

**（1）模块概述**

promotionbl模块实现了制定、发布促销策略的功能具体功能需求和非功能需求请参见需求规格说明文档和体系结构设计文档。

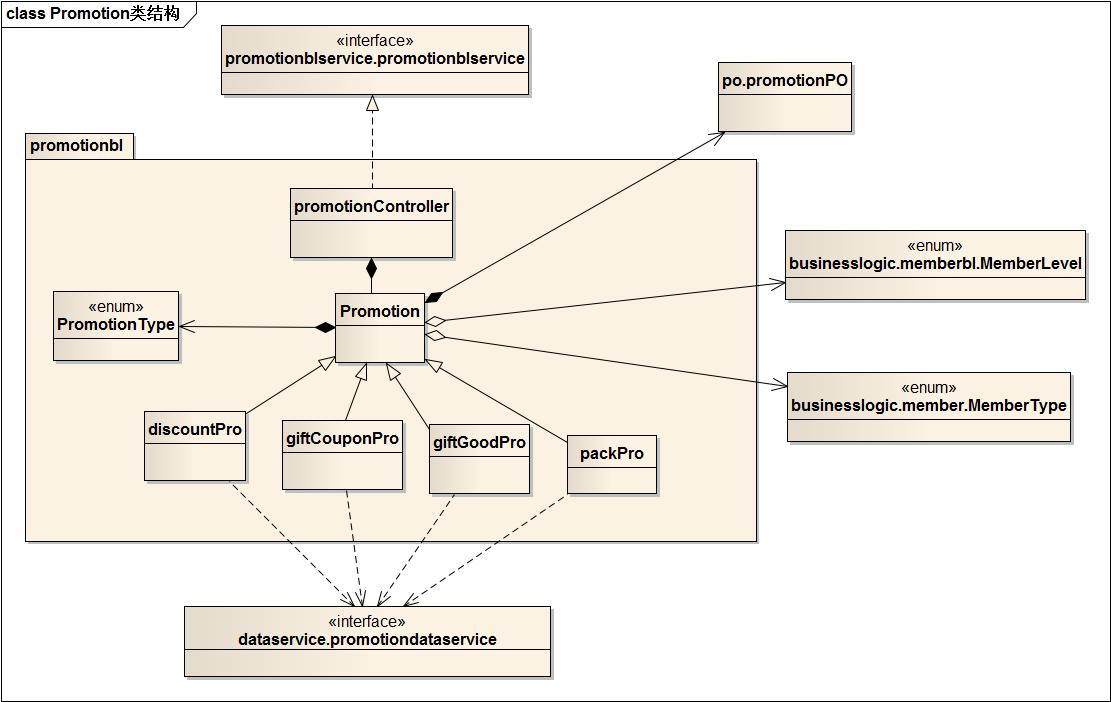
**(2)整体结构**

promotionbl模块实现了展示层与业务逻辑层之间的promotionblservice接口以及业务逻辑层和数据层之间的promotiondataservice接口。提供了对促销策略的Add,Modify,show,submit方法以及匹配促销策略的match和present方法。PromotionController隔离业务逻辑职责和逻辑控制职责，将对制定促销策略的业务逻辑处理委托给Promotion对象。giftGoodsPro,giftCouponPro,packPro,discountPro均由promotion派生而来，处理具体的不同类型的促销策略业务。promotionbl关键类的属性如图8所示

****

**图8 Promotion类的关键类属性**

promotionbl模块各个类的设计如图9所示



**图9 receiptbl各个类的设计**

**（3）模块内部类的接口规范**

promotionController和promotion的接口规范如表所示

**表16 promotionController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| promotionControlle  r.addgiftGoodsPro | **语法** | Public int addgiftGoodsPO(giftProVO vo ) |
| **前置条件** | 已创建一个promotion领域对象，且促销类型为商品赠送，输入符合输入规则 |
| **后置条件** | 调用giftGoodsPro的Add方法 |
| promotionControlle  r.addgiftcouponPro | **语法** | Public int addgiftcouponPro(giftProVO vo) |
| **前置条件** | 已创建一个promotion领域对象，且促销类型为代金券赠送，输入符合输入规则 |
| **后置条件** | 调用giftCouponPro的Add方法 |
| promotionControlle  r.adddiscountPro | **语法** | Public int adddiscountPro(discountProVO vo) |
| **前置条件** | 已创建一个promotion领域对象，且促销类型为折让，输入符合输入规则 |
| **后置条件** | 调用discountPro的Add方法 |
| promotionControlle  r.addpackPro | **语法** | Public int addpackPro(packProVO vo) |
| **前置条件** | 已创建一个promotion领域对象，且促销类型为特价包，输入符合输入规则 |
| **后置条件** | 调用packPro的Add方法 |
| promotionControlle  r.Modify | **语法** | Public int Modify(String id) |
| **前置条件** | 已创建一个promotion领域对象，且有单据可修改 |
| **后置条件** | 调用promotion的Modify方法 |
| promotionControlle  r.submit | **语法** | Public void submit(String id) |
| **前置条件** | 已创建一个promotion领域对象，且输入完整，符合输入规则 |
| **后置条件** | 调用promotion的submit方法 |
| promotionControlle  r.showAll | **语法** | public ArrayList<promotionVO> showAll() |
| **前置条件** | 已创建一个promotion领域对象 |
| **后置条件** | 调用promotion的view方法 |
| promotionControlle  r.getPro | **语法** | public promotionVO match(SaleVO vo) |
| **前置条件** | 已创建一个promotion领域对象，当前有销售发生 |
| **后置条件** | 调用promotion的match方法 |
| promotionControlle  r.getGift | **语法** | public giftVO getGift(promotionVO vo) |
| **前置条件** | 已创建一个promotion领域对象，且存在已发布促销策略 |
| **后置条件** | 调用promotion的present方法 |
| **需要的服务(需接口)** | | |
| **服务名** | **服务** | |
| Promotion.Add(promotionVO vo) | 创建一条促销策略 | |
| Promotion.Modify(PromotionVO vo) | 修改一条促销策略 | |
| Promotion.Submit(String id) | 发布一条促销策略 | |
| Promotion.view() | 查看所有的促销策略 | |
| Promotion.Match(SaleVO vo) | 查找当前销售所适用的促销策略 | |
| Promotion.Present(PromotionVO vo) | 根据促销策略生成一张库存赠送单，并发布 | |

**表17 promotion接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| promotion.Add | **语法** | Public int Add(PromotionVo vo) | |
| **前置条件** | 启动一个策略制定会合 | |
| **后置条件** | 保存该促销策略信息 | |
| promotion  .Modify | **语法** | Public int Modify(PromotionVo vo) | |
| **前置条件** | 系统中有可修改促销策略 | |
| **后置条件** | 更新该促销策略信息 | |
| promotion  .view | **语法** | Public ArrayList<PromotionVO> view() | |
| **前置条件** | 存在创建成功的促销策略 | |
| **后置条件** | 持久化更新涉及的领域对象的数据 | |
| Promotion.submit | **语法** | Public int submit(packProVO vo) | |
| **前置条件** | 该库存赠送单创建成功 | |
| **后置条件** | 持久化更新涉及的领域对象的数据 | |
| promotion  .Match | **语法** | Public PromotionVo Match(SaleVO vo) | |
| **前置条件** | 有销售发生 | |
| **后置条件** | 返回符合的促销策略信息 | |
| promotion  .Present | **语法** | Public GiftVo Present(PromotionVo vo) | |
| **前置条件** | 有符合促销策略的销售发生 | |
| **后置条件** | 返回库存赠送但信息 | |
| **需要的服务(需接口)** | | | |
| **服务名** | | | **服务** |
| PromotionDataService.Add(PromotionPo po) | | | 插入单一持久化对象 |
| PromotionDataService.Modify(PromotionPo po) | | | 修改一个PromotionPo对象 |
| PromotionDataService.Show () | | | 得到所有的PromotionPo对象 |
| PromotionDataService.find(String id) | | | 根据ID查找单一持久化对象 |
| FileFactory.getPromotionData | | | 得到Promotion文件的引用 |

**(4)业务逻辑层的动态模型**

如图10所示的顺序图表明了在进销存系统中，当总经理输入促销策略信息后后，制定促销策略业务逻辑处理的相关对象之间的协作。

****

**图10 promotionbl模块制定促销策略顺序图**

如图11所示，promotion对象状态图描述了promotion对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着show或addPromotion方法被 UI调用，promotion进入promotionlist状态；之后通过添加促销策略基础信息，根据促销类型的选择，进入packPro或giftGoodsPro或discountPro状态。随着addGoods方法被ui调用，promoiton进入Goods状态。最后,ui调用submit方法，发布策略，promotion跳转到promotionList状态。

****

**图11promotionbl模块制定促销策略状态图**

**4.2.2.3userbl模块**

**(1)模块概述**

userbl模块承担用户登录和增删改查用户信息的功能需求和安全性的非功能需求，具体可参见需求规格说明文档功能需求和相关非功能需求。

**(2)整体结构**

根据体系结构的设计，我们在展示层和业务逻辑层之间添加了userblservice接口。在业务逻辑层和数据层之间添加了userdataservice接口。Userblservice有login、addUser、deleteUser、modifyUser和showUser方法。其余类是User类的子类或子类的子类。

User类的属性如图12所示。

Image2

**图12user类的关键属性**

userbl模块的设计如图所示

Image2

**图13 userbl各个类的设计**

**(3)模块内部类的接口规范**

userbl的接口规范如表18所示

**表18 userbl的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| User.login | **语法** | public int login(String ID,String password) | |
| **前置条件** | password符合输入规则 | |
| **后置条件** | 查找是否存在相同的User，根据输入的password返回登录验证的结果 | |
| User.addUser | **语法** | public int addUser(UserVO vo) | |
| **前置条件** | User经过验证有添加新用户的权利，输入了有效的新用户信息 | |
| **后置条件** | 向系统中输入新增的用户账户信息 | |
| User.deleteUser | **语法** | public int deleteUser(UserVO vo) | |
| **前置条件** | User经过验证有删除用户的权利，输入了有效的待删除用户信息 | |
| **后置条件** | 系统查找对应的用户账户信息，并删除该用户对象 | |
| User.modifyUser | **语法** | public int modifyUser(UserVO vo) | |
| **前置条件** | User经过验证有修改用户的权利，输入了有效的用户修改信息 | |
| **后置条件** | 系统根据输入的用户信息和用户编号更新用户信息 | |
| User.showUser | **语法** | public UserPO showUser(String UserID) | |
| **前置条件** | User经过验证有查看所有用户的权利，输入了有效的用户ID信息 | |
| **后置条件** | 根据输入的ID查找用户，并返回用户信息 | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| FileFactory.getUserData | | | 得到User文件的引用 |
| UserDataService.add(UserPO po) | | | 在文件记录中插入UserPO对象 |
| UserDataService.check(UserPO po) | | | 得到根据输入的ID和password验证的结果 |
| UserDataService.delete(UserPO po) | | | 在文件中删除一个po对象 |
| userDataService.modify(UserPO po) | | | 在文件中更新一个po对象 |
| userDataService.showUser(long ID) | | | 得到根据输入的ID返回对应的po对象 |

**(4)业务逻辑层的动态模型**

如图14所示的顺序图表明了在选课系统中，当用户输入用户ID和password后，登录业务逻辑处理的相关对象之间的协作。

Image2

**图14 userbl模块的顺序图**

如图15 所示，User对象状态图描述了User对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着login方法被 UI调用，User进入administrator、inventoryManager、salesman或manager状态；之后administrator通过增删改用户、修改密码进入userManage状态。

Image2

**图15 userbl模块的状态图**

**4.2.2.4 stockbl模块**

**(1)模块概述**

stockbl模块承担商品及其分类管理、库存管理和库存赠送的功能需求和安全性的非功能需求，具体可参见需求规格说明文档功能需求和相关非功能需求。

**(2)整体结构**

根据体系结构的设计，我们在展示层和业务逻辑层之间添加了stockgoodsclassblservice、stockgoodsblservice、 stockerrorblservice和giftblservice接口。在业务逻辑层和数据层之间添加了stockgoodsclassdatalservice、stockgoodsdataservice和stockerrordataservice接口。Stockgoodsclassblservice有addGoodsClass、deleteGoodsClass和modifyGoodsClass方法，stockgoodsblservice有addGoods、deleteGoods、modifyGoods、findGoods、addGift、showStock和checkStock方法，stockerrorblservice有addStockOverOrLow和addStockError方法。StockGoodsClassController和goods、 stockGoodsController和stockGoods、stockErrorController和stockError、GiftController和gift是将业务的实现和控制逻辑分开，将每种任务委托给相应的处理对象去处理。

stockgoodsclass类的属性如图16所示。

Image2

**图16stockgoodsclass类的关键属性**

stockgoods类的属性如图17所示。

Image2

**图17stockgoods类的关键属性**

stockerror类的属性如图18所示。

Image2

**图18stockerror类的关键属性**

gift类的属性如题图19所示。

Image2

**图19gift类的关键属性**

stockbl模块的设计如图所示

****

**图20stockbl各个类的设计**

**(3)模块内部类的接口规范**

stockGoodsClassController的接口规范如表所示

**表19stockGoodsClassController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| StockGoodsClassController.addGoodsClass | **语法** | | Public int addGoodsClass(GoodsClassVO vo) |
| **前置条件** | | 操作人员请求新建一个商品分类 |
| **后置条件** | | 调用stockGoodsClass的add方法 |
| StockGoodsClassController.deleteGoodsClass | **语法** | | Public int deleteGoodsClass(GoodsClassVO vo) |
| **前置条件** | | 有一个商品分类需要被删除 |
| **后置条件** | | 调用stockGoodsClass的deleteGoodsClass方法 |
| StockGoodsClassController.modifyGoodsClass | **语法** | | Public void modifyGoodsClass(GoodsClassVO vo) |
| **前置条件** | | 操作人员想修改某个商品分类的相关信息 |
| **后置条件** | | 调用stockGoodsClass的modifyGoodsClass方法 |
| **需要的服务(需接口)** | | | |
| **服务名** | | **服务** | |
| stockGoodsClass.addGoodsClass(GoodsClassVO vo) | | 新增一个商品分类 | |
| stockGoodsClass.deleteGoodsClass(GoodsClassVO vo) | | 删除一个商品分类 | |
| stockGoodsClass.modifyGoodsClass(GoodsClassVO vo) | | 修改一个商品分类 | |

**表20stockGoodsController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| StockGoodsController.addGoods | **语法** | | Public int addGoods(GoodsVO vo) |
| **前置条件** | | 操作人员请求新增加一个商品 |
| **后置条件** | | 调用stockGoods的addGoods方法 |
| StockGoodsController.deleteGoods | **语法** | | Public int deleteGoods(GoodsVO vo) |
| **前置条件** | | 有一个商品需要被删除 |
| **后置条件** | | 调用stockGoods的deleteGoods方法 |
| StockGoodsController.modifyGoods | **语法** | | Public void modifyGoods (GoodsVO vo) |
| **前置条件** | | 操作人员想修改某个商品的相关信息 |
| **后置条件** | | 调用stockGoodss的modifyGoods方法 |
| **需要的服务(需接口)** | | | |
| **服务名** | | **服务** | |
| stockGoods.addGoods (GoodsVO vo) | | 新增一个商品 | |
| stockGoodsClass.deleteGoods (GoodsVO vo) | | 删除一个商品 | |
| stockGoodsClass.modifyGoods (GoodsVO vo) | | 修改一个商品 | |

**表21 stockErrorController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockErrorController.addStockOverOrLow | **语法** | public int addStockOverOrLow(StockOverOrLowVO vo) |
| **前置条件** | 操作人员需要创建一个库存报溢或者报损单 |
| **后置条件** | 调用stockError的addStockOverOrLow方法 |
| StockErrorController.addStockError | **语法** | Public int addStockError(StockErrorVO vo) |
| **前置条件** | 库存量低于警戒数量 |
| **后置条件** | 调用stockError的addError方法 |
| StockErrorController.getOverAccount | **语法** | Public double getOverAccount(StockOverOrLowPO po) |
| **前置条件** | 财务人员需要获得报溢收入时 |
| **后置条件** | 调用stockError的getOverAccount方法 |
| StockErrorController.getLowAccount | **语法** | Public double getLowAccount(StockOverOrLowPO po) |
| **前置条件** | 财务人员需要获得报损支出时 |
| **后置条件** | 调用stockError的getLowAccount方法 |
| **需要的服务(需接口)** | | |
| **服务名** | | **服务** |
| stockError.addStockOverOrLow (StockOverOrLowVO vo) | | 创建一个报溢或者报损单 |
| stockError.addError(StockErrorVO vo) | | 创建一个报警单 |
| StockErrorController.getOverAccount(StockOverOrLowPO po) | | 计算库存报溢收入 |
| StockErrorController.getLowAccount(StockOverOrLowPO po) | | 计算库存报损支出 |

**表22 giftController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| GiftController.addGift | **语法** | | Public int addGift(GiftVO vo) |
| **前置条件** | | 操作人员需要创建一个库存赠送单 |
| **后置条件** | | 调用gift的addGift方法 |
| GiftController.addGift | **语法** | | Public int dealGift (GiftVO vo) |
| **前置条件** | | 处理一个赠送单 |
| **后置条件** | | 调用gift的dealGift方法 |
| GiftController.getAccount(GiftVO vo) | **语法** | | Public double getAccount (GiftVO vo) |
| **前置条件** | | 财务人员想知道一个赠送单的总价 |
| **后置条件** | | 调用gift的getAccount方法 |
| **需要的服务(需接口)** | | | |
| **服务名** | | **服务** | |
| gift.addgift (GiftVO vo) | | 创建一个库存赠送单 | |
| gift.addError(GiftVO vo) | | 处理一个库存赠送单 | |
| gift.getAccount(GiftVO vo) | | 计算赠送商品的总价 | |

stockbl的接口规范如表23所示

**表23stockbl的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Stock.addGoodsClass | **语法** | public int addGoodsClass (GoodsClassVO vo) | |
| **前置条件** | User经过验证有添加新商品分类的权利，输入了有效的新分类信息 | |
| **后置条件** | 向系统中输入新增的商品分类信息 | |
| Stock.deleteGoodsClass | **语法** | public int deleteGoodsClass (GoodsClassVO vo) | |
| **前置条件** | User经过验证有删除分类的权利，输入了有效的待删除的信息 | |
| **后置条件** | 系统查找对应的商品分类信息，并删除该商品分类对象 | |
| Stock.modifyGoodsClass | **语法** | public int modifyGoodsClass (GoodsClassVO vo) | |
| **前置条件** | User经过验证有修改商品分类的权利，输入了有效的商品分类信息 | |
| **后置条件** | 系统根据输入的分类名称和商品上级分类更新商品分类信息 | |
| Stock.addGoods | **语法** | public int addGoods (GoodsVO vo) | |
| **前置条件** | User经过验证有添加新商品的权利，输入了有效的新商品信息 | |
| **后置条件** | 向系统中输入新增的商品信息 | |
| Stock.deleteGoods | **语法** | public int deleteGoods (GoodsVO vo) | |
| **前置条件** | User经过验证有删除商品信息的权利，输入了有效的待删除商品信息 | |
| **后置条件** | 系统查找对应的商品信息，并删除该商品对象 | |
| Stock.modifyGoods | **语法** | public int modifyGoods (GoodsVO vo) | |
| **前置条件** | User经过验证有修改商品属性的权利，输入了有效的新商品属性 | |
| **后置条件** | 系统根据输入的商品的各个属性更新商品信息 | |
| Stock.findGoods | **语法** | public ArrayList<GoodsPO> findGoods (String message) | |
| **前置条件** | User经过验证有查找商品的权利，输入了有效的商品关键字信息 | |
| **后置条件** | 系统根据输入的商品的相关属性显示查询到的商品信息 | |
| Stock.showGoods | **语法** | public ArrayList<GoodsPO> showGoods () | |
| **前置条件** | User经过验证有查找商品的权利，要查看所有商品信息 | |
| **后置条件** | 系统显示所有商品信息 | |
| Stock.isEnough | **语法** | public Boolean (String ID, int number) | |
| **前置条件** | User经过验证有查找商品的权利，要了解某种商品库存是否满足销售时的数量 | |
| **后置条件** | 系统返回库存足够与否 | |
| Stock.showStock | **语法** | public ArrayList<GoodsPO> showStock (String beginDate,String endDate) | |
| **前置条件** | User经过验证有查看库存的权利，输入了有效的时间段 | |
| **后置条件** | 系统根据输入的时间段显示查询到的该时间段内的库存商品信息 | |
| Stock.checkStock | **语法** | public ArrayList<GoodsPO>checkStock () | |
| **前置条件** | User经过验证有盘点库存的权利 | |
| **后置条件** | 系统显示当天的库存快照 | |
| Stock.add addStockOverOrLow | **语法** | public int addStockOverOrLow (StockOverOrLowVO vo) | |
| **前置条件** | 当实际库房中的商品数量比系统中的多或少 | |
| **后置条件** | 系统创建并存储库存报溢（损）单，返回操作成功与否 | |
| Stock.getOverAccount | **语法** | public double getOverAccount (StockOverOrLowPO po) | |
| **前置条件** | 财务人员想了解库存报溢的收入 | |
| **后置条件** | 系统计算报溢收入并返回 | |
| Stock.getLowAccount | **语法** | public double getLowAccount (StockOverOrLowPO po) | |
| **前置条件** | 财务人员想了解库存报损的收入 | |
| **后置条件** | 系统计算报损收入并返回 | |
| Stock.addGift | **语法** | public int addGift (GiftVO vo) | |
| **前置条件** | User经过验证有新建库存赠送单的权利，输入了有效的赠品信息，客户尚未因满足促销条件而获得库存赠送 | |
| **后置条件** | 系统创建并存储库存赠送单，并返回操作成功与否 | |
| Stock.dealGift | **语法** | public int dealGift (GiftVO vo) | |
| **前置条件** | User经过验证有新建库存赠送单的权利，取得了要处理的赠送单 | |
| **后置条件** | 系统处理库存赠送单，并返回操作成功与否 | |
| Stock.getAccount | **语法** | public double getAccount (GiftVO vo) | |
| **前置条件** | 财务人员需要了解一个库存赠送单的总价 | |
| **后置条件** | 返回库存赠送单的总价 | |
| Stock.addStockError | **语法** | public int addStockError (StockErrorVO vo) | |
| **前置条件** | 无 | |
| **后置条件** | 系统创建并存储库存报警单，并返回操作成功与否 | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| FileFactory.getStockData | | | 得到Stock文件的引用 |
| StockDataService.addGoodsClass (GoodsClassPO po) | | | 向外部文件中插入一个GoodsClassPo对象 |
| StockDataService.deleteGoodsClass (GoodsClassPO po) | | | 从文件中删除一个GoodsClassPO对象 |
| StockDataService.modifyGoodsClass (GoodsClassPO po) | | | 修改文件中对应的GoodsClassPo对象 |
| StockDataService.addGoods(GoodsPO po) | | | 向外部文件插入一个GoodsPo对象 |
| StockDataService. deleteGoods (GoodsPO po) | | | 从文件中删除一个GoodsPo对象 |
| StockDataService.modifyGoods (GoodsPO po) | | | 修改文件中对应的GoodsPo对象 |
| StockDataService.findGoods(String message) | | | 根据message返回文件中响应的商品的详细信息 |
| StockDataService.showStock(String beginDate,String endDate) | | | 根据beginDate和endDate，返回该时间段内的商品库存信息 |
| StockDataService.checkStock ( ) | | | 返回当天的库存快照 |
| StockDataService.addStockOverOrLow（StockOverOrLowPO po） | | | 向外部文件中插入一个StockOverOrLowPo对象 |
| StockDataService .addGift(GiftPO po) | | | 向外部文件中插入一个GiftPo对象 |
| StockDataService.addStockError（StockErrorPO po） | | | 向外部文件中插入一个StockErrorPo对象 |

**(4)业务逻辑层的动态模型**

如图21所示的顺序图表明了在进销存系统中，当库存管理人员进行商品分类管理时，商品分类的相关对象之间的协作。

Image2

**图21 stockbl模块新增商品分类顺序图**

如图22所示的顺序图表明了在进销存系统中，当库存管理人员进行商品管理时，商品的相关对象之间的协作。

Image2

**图22 stockbl模块新增商品顺序图**

如图23所示的顺序图表明了在进销存系统中，当库存管理人员进行库存报溢、报损管理时，库存报溢、报损的相关对象之间的协作。

Image2

**图23 stockbl模块库存报溢顺序图**

如图24所示的顺序图表明了在进销存系统中，当库存管理人员进行库存赠送时，库存赠送的相关对象之间的协作。

Image2

**图24 stockbl模块库存赠送顺序图**

如图25所示，goodsClass对象状态图描述了goodsClass对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishGoodsClass方法被 UI调用，establishGoodsClass根据不同的指令进入不同状态；之后通过更新计划进入goodsClassTree状态。



**图25 stockbl模块商品分类管理状态图**

如图26所示，stockGoods对象状态图描述了stockGoods对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishStockGoods方法被 UI调用，establishStockGoods根据不同的指令进入不同状态。

Image2

**图26 stockbl模块库存管理状态图**

如图27所示，stockError对象状态图描述了stockError对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishStockError方法被 UI调用，establishStockError根据不同的指令进入不同状态。



**图27 stockbl模块库存报溢、报损状态图**

如图28所示，stockError对象状态图描述了stockError对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishStockError方法被 UI调用，establishStockError根据不同的指令进入不同状态。



**图28 stockbl模块库存赠送状态图**

**4.2.2.5Finance.AccountBL模块**

**(1)模块概述**

Finance.AccountBL模块承担增删改查账户的功能需求和安全性的非功能需求，具体可参见需求规格说明文档功能需求和相关非功能需求。

**(2)整体结构**

根据体系结构的设计，我们将系统分为展示层、业务层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们添加businesslogicservice.financeblservice.Account接口。业务逻辑层和数据层之间添加dataService.financedataservice.Account接口。因为业务逻辑处理较少，根据专家模式没有添加controller类。AccountPO是作为销售记录的持久化对象被添加到设计模型中去的。

Account类的属性如图29所示。

****

**图29Account类的关键属性**

Finance.AccountBL模块的设计如图30所示



**图30Finance.AccountBL各个类的设计**

**(3)模块内部类的接口规范**

Account的接口规范如表24所示

**表24Account的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Account.addAccount | **语法** | | public intaddAccount(AccountVO a) |
| **前置条件** | | 用户启动一个增加账户任务 |
| **后置条件** | | 返回添加账户的结果 |
| Account.deleteAccount | **语法** | | public intdeleteAccount(AccountVO a) |
| **前置条件** | | 用户启动删除账户任务 |
| **后置条件** | | 返回删除账户的结果 |
| Account.modifyAccount | **语法** | | public intdeleteAccount(AccountVO a) |
| **前置条件** | | 用户启动修改账户任务 |
| **后置条件** | | 返回修改账户的结果 |
| Account.findAccount | **语法** | | public AccountVOfindAccount(String a) |
| **前置条件** | | 用户启动查找账户任务 |
| **后置条件** | | 返回查找到的账户，未查到返回null |
| Account.getName | **语法** | | public StringgetName() |
| **前置条件** | | 已添加该账户信息 |
| **后置条件** | | 返回此账户的名称 |
| Account.getMoney | **语法** | | public doublegetMoney() |
| **前置条件** | | 已添加该账户信息 |
| **后置条件** | | 返回此账户的余额 |
| Account.modifyMoney | **语法** | | public intmodifyMoney(char c,double money) |
| **前置条件** | | 已添加该账户信息 |
| **后置条件** | | 返回增加或减少余额成功与否 |
| Account.showAccount | **语法** | | Public ArrayList<AccountVO>showAccount() |
| **前置条件** | | 需要查看账户列表时 |
| **后置条件** | | 返回账户列表 |
| **需要的服务（需接口）** | | | |
| **服务名** | | **服务** | |
| financedataservice.Account.addAccount(AccountPO a) | | 插入单一持久化对象AccountPO | |
| financedataservice.Account.deleteAccount(AccountPO a) | | 删除单一持久化对象AccountPO | |
| financedataservice.Account.modifyAccount(AccountPO a) | | 修改单一持久化对象AccountPO的名称 | |
| financedataservice.Account.findAccount(String a) | | 查找单一持久化对象AccountPO | |
| financedataservice.Account.modifyMoney (AccountPO a) | | 修改单一持久化对象AccountPO的余额 | |
| financedataservice.Account.showAccount () | | 得到单一化持久对象AccountPO的列表 | |
| DataFactory.getAccountDatafile | | 得到Account序列化文件的引用 | |

**(4)业务逻辑层的动态模型**

图31标明了进销存系统中，当用户输入账户的名称之后，账户业务逻辑处理的相关对象之间的协作。



**图31添加账户的顺序图**

如图32所示，Account对象状态图描述了Account对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着addAccount方法被 UI调用，Account进入Account状态；之后通过添加账户进入findAccount状态。随后，进入addAccount状态；随着submit方法被UI调用，Account进入endAccount状态。



**图32Finance.AccountBL 模块状态图**

**4.2.2.6FinanceListBL模块**

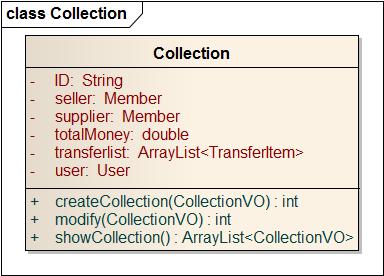
**(1)模块概述**

FinanceListBL模块承担创建单据的功能需求和安全性的非功能需求，具体可参见需求规格说明文档功能需求和相关非功能需求。

**(2)整体结构**

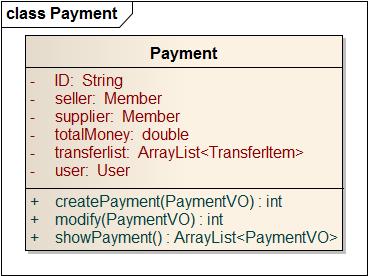
根据体系结构的设计，我们将系统分为展示层、业务层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们添加businesslogicservice.financeblservice.List接口。使用集中式控制风格。ListController和Collection,Payment,CashList将业务的实现和控制逻辑分开。业务逻辑层和数据层之间添加dataService.financedataservice.List接口。CollectionPO,PaymentPO,CashListPO是作为销售记录的持久化对象被添加到设计模型中去的。

Collection类的属性如图33所示。



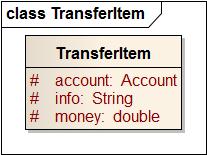
**图33 Collection类的关键类属性**

Payment类的属性如图34所示。



**图34 Payment类的关键类属性**

TransferItem是Collection和Payment的内部类，属性如图35所示。



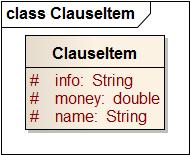
**图35TransferItem类的关键类属性**

CashList类的属性如图36所示。



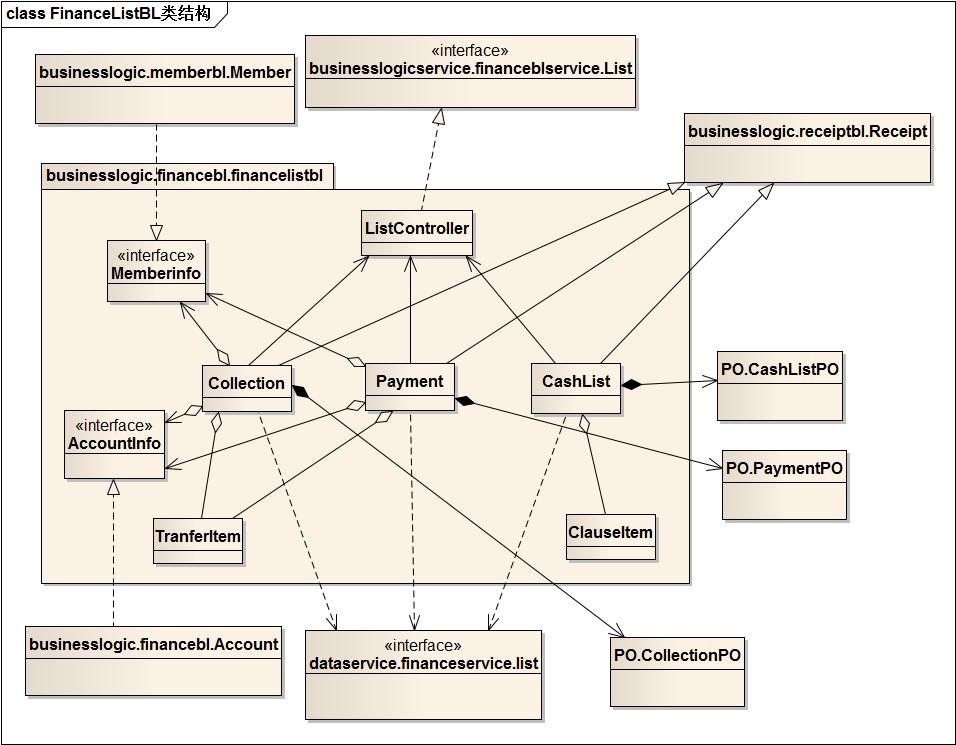
**图36 CashList类的关键类属性**

ClauseItem是CashList的内部类，属性如图37所示。



**图37ClauseItem类的关键类属性**

FinanceListBL模块各个类的设计如图38所示



**图38FinanceListBL各个类的设计**

FinanceListBL模块各个类的职责如表25所示

**表25 FinanceListBL模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ListController | 负责实现财务单据界面所需要的服务 |
| Collection | 收款单的领域模型对象，拥有一个收款单所持有的编号，操作员等信息，可帮助完成创建收款单界面所需要的服务 |
| Payment | 付款单的领域模型对象，拥有一个付款单所持有的编号，操作员等信息，可帮助完成创建付款单界面所需要的服务 |
| CashList | 现金费用单的领域模型对象，拥有一个现金费用单所持有的编号，操作员等信息，可以帮助完成创建现金费用单界面所需要的服务 |

**(3)模块内部类的接口规范**

ListController的接口规范如表所示

**表26 FinanceList的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| ListController.createCollection | **语法** | | public int createCollection(CollectionVO vo) |
| **前置条件** | | 已创建一个FinanceList领域对象，并且输入符合输入规则 |
| **后置条件** | | 调用FinanceList领域对象的createCollection方法 |
| ListController.createPayment | **语法** | | public int createPayment(PaymentVO vo) |
| **前置条件** | | 已创建一个FinancListe领域对象，并且输入符合输入规则 |
| **后置条件** | | 调用FinanceList领域对象的createPayment方法 |
| ListController..createCashList | **语法** | | public intcreateCashList(CashListVO a) |
| **前置条件** | | 已创建一个FinanceList领域对象，并且输入符合输入规则 |
| **后置条件** | | 调用FinanceList领域对象的createCashList方法 |
| ListController.createRedExtrusion | **语法** | | public int createRedExtrusion(ReceiptVO vo); |
| **前置条件** | | 已创建一个FinanceList领域对象，并且输入符合输入规则 |
| **后置条件** | | 调用FinanceList领域对象的createRedExtrusion方法 |
| ListController.creatRedExtrusionAndCopy | **语法** | | public int creatRedExtrusionAndCopy(ReceiptVO vo) |
| **前置条件** | | 已创建一个FinanceList领域对象，并且输入符合输入规则 |
| **后置条件** | | 调用FinanceList领域对象的RedExtrusionAndCopy方法 |
| ListController.showCollection | **语法** | | public ArrayList<Collection> showCollection() |
| **前置条件** | | 已创建一个FinanceList领域对象 |
| **后置条件** | | 调用FinanceList领域对象的showCollection方法 |
| ListController.showPayment | **语法** | | public ArrayList<Payment> showPayment() |
| **前置条件** | | 已创建一个FinanceList领域对象 |
| **后置条件** | | 调用FinanceList领域对象的showPayment方法 |
| ListController.showCashList | **语法** | | public ArrayList<CashList> showCashList() |
| **前置条件** | | 已创建一个FinanceList领域对象 |
| **后置条件** | | 调用FinanceList领域对象的showCashList方法 |
| **需要的服务（需接口）** | | | |
| **服务名** | | **服务** | |
| List. createCollection | | 创建一个收款单 | |
| List. createPayment | | 创建一个付款单 | |
| List. createCashlist | | 创建一个现金费用单 | |
| List.createRedExtrusion | | 创建一个红冲单对象 | |
| List.createRedExtrusionAndCopy | | 创建一个红冲单对象并且生成一张草稿单 | |
| List.showCollection | | 显示所有的收款单对象 | |
| List.showPayment | | 显示所有的付款单对象 | |
| List.showCashList | | 显示所有的现金费用单对象 | |

FinanceList的接口规范如表27所示

**表27 FinanceList的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| List.createCollection | **语法** | | public int createCollection(CollectionVO vo) |
| **前置条件** | | 用户启动一个创建收款单任务 |
| **后置条件** | | 返回创建收款单的结果，系统存储收款单信息 |
| List.createPayment | **语法** | | public int createPayment(PaymentVO vo) |
| **前置条件** | | 用户启动一个创建付款单任务 |
| **后置条件** | | 返回创建付款单的结果，系统存储付款单信息 |
| List.createCashList | **语法** | | public intcreateCashList(CashListVO a) |
| **前置条件** | | 用户启动一个创建现金费用单任务 |
| **后置条件** | | 返回创建现金费用单的结果，系统中存储现金费用单信息 |
| List. createRedExtrusion | **语法** | | public int createRedExtrusion(ReceiptVO vo); |
| **前置条件** | | 用户启动一个创建红冲单任务 |
| **后置条件** | | 返回创建红冲单的结果，系统中存储红冲单信息 |
| List.creatRedExtrusionAndCopy | **语法** | | public int creatRedExtrusionAndCopy(ReceiptVO vo) |
| **前置条件** | | 用户启动一个创建红冲单并复制任务 |
| **后置条件** | | 返回创建红冲单并复制的结果，系统中存储红冲单信息，并创建一个草稿单 |
| List.showCollection | **语法** | | public ArrayList<Collection> showCollection() |
| **前置条件** | | 用户启动查看所有收款单任务 |
| **后置条件** | | 返回所有收款单 |
| List.showPayment | **语法** | | public ArrayList<Payment> showPayment() |
| **前置条件** | | 用户启动查看所有付款单任务 |
| **后置条件** | | 返回所有付款单 |
| List.showCashList | **语法** | | public ArrayList<CashList> showCashList() |
| **前置条件** | | 用户启动查看所有现金费用单任务 |
| **后置条件** | | 返回所有现金费用单 |
| **需要的服务（需接口）** | | | |
| **服务名** | | **服务** | |
| financedataservice.List.createCollection (CollectionPO a) | | 插入单一持久化对象CollectionPO | |
| financedataservice.List.createPayment (PaymentPO a) | | 删除单一持久化对象PaymentPO | |
| financedataservice.List.createCashList (CashListPO a) | | 修改单一持久化对象CashListPO的名称 | |
| financedataservice.List.showCollection () | | 返回单一持久化对象CollectionPO | |
| financedataservice.List.showPayment () | | 返回单一持久化对象PaymentPO | |
| financedataservice.List.showCashlist () | | 返回单一持久化对象CashListPO | |
| DataFactory.getListDatafile | | 得到FinanceList序列化文件的引用 | |

**(4)业务逻辑层的动态模型**

图39标明了进销存系统中，当用户输入收款单的信息之后，财务单据业务逻辑处理的相关对象之间的协作。



**图39创建收款单的顺序图**

如图40所示，FinanceList对象状态图描述了FinanceList对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着createCollection方法被 UI调用，ListUI进入List状态；之后通过创建收款单进入Member状态。随后，通过填写账户信息进入Account状态；随着submit方法被UI调用，FinanceList进入endList状态。



**图40FinanceListBL 模块状态图**

**4.2.2.7Finance.BSLBL模块**

**(1)模块概述**

Financel.BSLBL模块承担查看经营情况表的功能需求和安全性的非功能需求，具体可参见需求规格说明文档功能需求和相关非功能需求。

**(2)整体结构**

根据体系结构的设计，我们将系统分为展示层、业务层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们添加businesslogicservice.financeblservice.BSL接口。因为业务逻辑处理较少，根据专家模式没有添加controller类。业务逻辑层和数据层之间添加dataService.financedataservice.BSL接口。BSLPO是作为销售记录的持久化对象被添加到设计模型中去的。

BSL类的属性如图所示。

****

**图41BSL类的关键属性**

BSLBL模块的设计如图42所示



**图42Finance.BSLBL各个类的设计**

Finance.BSLBL模块各个类的职责如表28所示

**表1Finance. BSLBL模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| BSL | 经营情况表的领域模型对象，拥有一个经营情况表所持有的收入，支出等信息，可以帮助完成经营情况表界面所需要的服务 |

**(3)模块内部类的接口规范**

Finance.BSL的接口规范如表29所示

**表29Finance. BSL的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BSL.viewBSL | **语法** | public BSLVOviewBSL() |
| **前置条件** | 用户启动一个查看经营情况表任务 |
| **后置条件** | 返回经营情况表 |
| BSL.export | **语法** | public int export(BSLVO a) |
| **前置条件** | 用户启动一个导出经营情况表任务 |
| **后置条件** | 系统将经营情况表导出到Excel中 |
| **需要的服务（供接口）** | | |
| **服务名** | **服务** | |
| financedataservice.BSL.viewBSL(BSLVO a) | 得到单一化持久对象BSLPO | |

**(4)业务逻辑层的动态模型**

图43标明了进销存系统中，当用户点击查看经营情况表之后，经营情况表业务逻辑处理的相关对象之间的协作。



**图43查看经营情况表的顺序图**

如图44所示，BSL对象状态图描述了BSL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着showBSL方法被 UI调用，BSLUI进入BSL状态；随着close方法被UI调用，BSL进入endBSL状态。



**图44BSLBL 模块状态图**

**4.2.2.8Finance.InitBL模块**

**(1)模块概述**

Finance.InitBL模块承担期初建账的功能需求和安全性的非功能需求，具体可参见需求规格说明文档功能需求和相关非功能需求。

**(2)整体结构**

根据体系结构的设计，我们将系统分为展示层、业务层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们添加businesslogicservice.financeblservice.Init接口。因为业务逻辑处理较少，根据专家模式没有添加controller类。业务逻辑层和数据层之间添加dataService.financedataservice.Init接口。BeginInfoPO是作为销售记录的持久化对象被添加到设计模型中去的。

BeginInfo类的属性如图所示。



**图45BeginInfo类的关键属性**

InitBL模块的设计如图46所示



**图46Finance.InitBL各个类的设计**

Finance.InitBL模块各个类的职责如表1所示

**表30Finance.BSLBL模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| Init | 期初建账的领域模型对象，拥有一个套账所持有的商品，客户，账户等信息，可以帮助完成期初建账界面所需要的服务 |

**(3)模块内部类的接口规范**

Init的接口规范如表31所示

**表31 BSL的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Init.initialInfo | **语法** | public int initialInfo(BeginInfoVO vo) |
| **前置条件** | 用户启动一个期初建账任务 |
| **后置条件** | 返回期初建账的结果 |
| Init.getInfo | **语法** | public BeginInfoVO getInfo(String time) |
| **前置条件** | 用户启动查看期初套账任务 |
| **后置条件** | 返回一个期初套账 |
| **需要的服务（供接口）** | | |
| **服务名** | **服务** | |
| financedataservice.Init.initialInfo | 插入一个单一化持久对象BeginInfoPO | |
| financedataservice.Init.getInfo(String time) | 得到单一化持久对象BeginInfoPO | |

**(4)业务逻辑层的动态模型**

图47标明了进销存系统中，当用户点击期初建账之后，期初建账业务逻辑处理的相关对象之间的协作。



**图47期初建账的顺序图**

如图48所示，BeginInfo对象状态图描述了BeginInfo对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着initialInfo方法被 UI调用，InitUI进入BeginInfo状态；之后通过添加商品信息进入Goods状态。随后，通过添加客户信息进入Member状态；然后，通过添加银行账户信息进入Account状态。随着submit方法被UI调用，BeginInfo进入endInit状态。



**图48 Finance.InitBL 模块状态图**

**4.2.2.9MemberBL模块**

**(1)模块概述**

memberbl模块承担着客户管理的功能需求，具体可参见需求规格说明文档的相关描述。

**(2)整体结构**

根据体系结构的设计，我们在展示层和业务逻辑层之间添加了MemberBLService接口。在业务逻辑层和数据层之间添加了MemberDataService接口。MemberBLService有addMember,deleteMember,modifyMember,findMember和showMember方法，用来完成客户的增删改查及显示任务。

Member类的属性如图49所示。



**图49 Member类的关键属性**

memberbl模块各个类的设计如图50所示



**图50 MemberBL各个类的设计**

**(3)模块内部类的接口规范**

memberbl的接口规范如表32所示

**表32 memberbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| member. addMember | **语法** | public int addMember(MemberVO vo) | |
| **前置条件** | 需要添加客户时 | |
| **后置条件** | 向系统中添加新客户信息 | |
| member. deleteMember | **语法** | public int deleteMember(MemberVO vo) | |
| **前置条件** | 需要删除会员时 | |
| **后置条件** | 系统查找对应的客户信息，并删除MemberPO | |
| member. modifyMember | **语法** | public int modifyMember(MemberVO vo) | |
| **前置条件** | 需要修改会员信息时 | |
| **后置条件** | 系统查找对应客户信息，并修改 | |
| member. findMember | **语法** | public MemberVOfindMember(String message) | |
| **前置条件** | 需要查找客户时 | |
| **后置条件** | 根据输入的关键字查找客户，并返回客户信息 | |
| member. showMembers | **语法** | public ArrayList<MemberVO> showMembers() | |
| **前置条件** | 需要查看客户列表时 | |
| **后置条件** | 返回客户列表 | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| FileFactory.getMemberData | | | 得到Member文件的引用 |
| memberDataService.add(MemberPO po) | | | 插入一个MemberPO对象 |
| memberDataService.delete(MemberPO po) | | | 删除一个MemberPO对象 |
| memberDataService.modify(MemberPO po) | | | 修改一个MemberPO对象 |
| memberDataService.find(String message) | | | 根据message返回对应MemberPo对象 |

**(4)业务逻辑层的动态模型**

图51标明了进销存系统中，当用户选择客户管理之后，增加客户业务逻辑处理的相关对象之间的协作。

****

**图51添加客户的顺序图**

如图52所示， member对象状态图描述了member对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着addMembert方法被 UI调用，Member进入addMember状态；之后通过查找用户进入findMember状态。随后，select方法被ui调用进入MemberList状态；随着MemberInfo的输入，Member进入modifyMember或其他增删Member的状态。

****

**图52Member对象状态图**

**4.2.2.10SalesBL模块**

**(1)模块概述**

salesbl模块承担着进货销售的功能需求及相关非功能需求，具体可参见需求规格说明文档的相关描述。

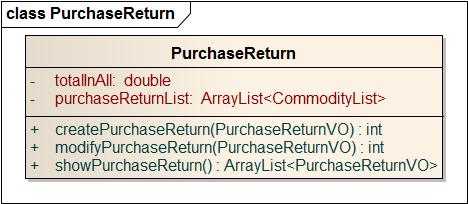
**(2)整体结构**

根据体系结构的设计，我们在展示层和业务逻辑层之间添加了SalesBLService接口。在业务逻辑层和数据层之间添加了SalesDataService接口。SalesBLService有purchase,purchaseReturn,sale,saleReturn和showSaleList等方法，用来完成进货、进货退货、销售、销售退货、查看销售明细表任务。使用集中式控制风格。SalesController和Sale,SaleReturn,Purchase,PurchaseReturn,SaleList将业务的实现和控制逻辑分开。

上述五个类的关键属性如图53-57所示。



**图53 Purchase类的关键类属性**



**图54 PurchaseReturn类的关键类属性**



**图55 Sale类的关键类属性**

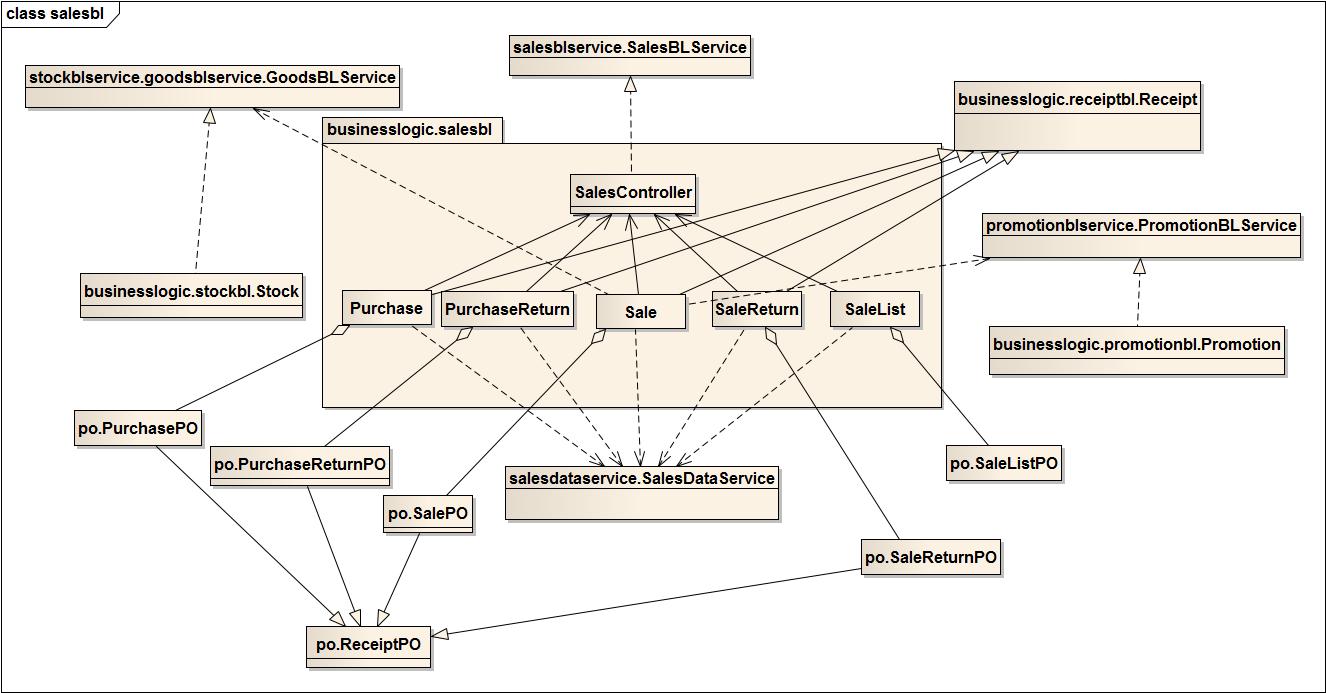


**图56 SaleReturn类的关键类属性**



**图57 SaleList类的关键类属性**

salesbl模块各个类的设计如图58所示



**图58 SaleBL模块各个类的设计**

**(3)模块内部类的接口规范**

saleController,salesbl的接口规范如表33-34所示

**表33 SalesController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| SalesController.Purchase | **语法** | public int purchase (PurchaseVO vo); | |
| **前置条件** | 销售人员使用创建进货单功能时 | |
| **后置条件** | 调用Purchase类的createPurchase方法 | |
| SalesController.PurchaseReturn | **语法** | public int purchaseReturn (PurchaseReturnVO vo); | |
| **前置条件** | 销售人员使用创建进货退货单功能时 | |
| **后置条件** | 调用PurchaseReturn类的createPurchaseReturn方法 | |
| SalesController.Sale | **语法** | public int sale (SaleVO vo); | |
| **前置条件** | 销售人员使用创建销售单功能时 | |
| **后置条件** | 调用Sale类的createSale方法 | |
| SalesController.SaleReturn | **语法** | public int saleReturn (SaleReturnVO vo); | |
| **前置条件** | 销售人员使用创建销售退货单功能时 | |
| **后置条件** | 调用SaleReturn类的createSaleReturn方法 | |
| SalesController.showSaleList | **语法** | public SaleListVO showSaleList() | |
| **前置条件** | 销售人员使用查看销售明细表功能时 | |
| **后置条件** | 调用SaleList类的showSaleList方法 | |
| SalesController.modifyPurchase | **语法** | public int modifyPurchase (PurchaseVO vo); | |
| **前置条件** | 总经理需要修改进货单 | |
| **后置条件** | 调用Purchase类的modifyPurchase方法 | |
| SalesController.modifyPurchaseReturn | **语法** | public int modifyPurchaseReturn (PurchaseReturnVO vo); | |
| **前置条件** | 总经理需要修改进货退货单 | |
| **后置条件** | 调用PurchaseReturn类的modifyPurchaseReturn方法 | |
| SalesController.modifySale | **语法** | public int modifySale (SaleVO vo); | |
| **前置条件** | 总经理需要修改销售单 | |
| **后置条件** | 调用Sale类的modifySale方法 | |
| SalesController.modifySaleReturn | **语法** | public int modifySaleReturn (SaleReturnVO vo); | |
| **前置条件** | 总经理需要修改销售退货单 | |
| **后置条件** | 调用SaleReturn类的modifySaleReturn方法 | |
| SalesController.showPurchase | **语法** | public ArrayList<PurchaseVO> showPurchase(); | |
| **前置条件** | 需要显示所有进货单 | |
| **后置条件** | 调用Purchase类的showPurchase方法 | |
| SalesController.showPurchaseReturn | **语法** | public ArrayList<PurchaseReturnVO> showPurchaseReturn(); | |
| **前置条件** | 需要显示所有进货退货单 | |
| **后置条件** | 调用PurchaseReturn类的showPurchaseReturn方法 | |
| SalesController.showSale | **语法** | public ArrayList<SaleVO> showSale(); | |
| **前置条件** | 需要显示所有销售单 | |
| **后置条件** | 调用Sale类的showSale方法 | |
| SalesController.showSaleReturn | **语法** | public ArrayList<SaleReturnVO> showSaleReturn(); | |
| **前置条件** | 需要显示所有销售退货单 | |
| **后置条件** | 调用SaleReturn类的showSaleReturn方法 | |
| SalesController.exportSaleListToExcel | **语法** | public int exportToEXCEL(); | |
| **前置条件** | 需要把销售明细表导出到Excel | |
| **后置条件** | 调用SaleList类的exportToEXCEL方法 | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| Purchase.createPurchase | | | 创建进货单 |
| Purchase.modifyPurchase | | | 修改进货单 |
| Purchase.showPurchase | | | 查看所有进货单 |
| PurchaseReturn.createPurchaseReturn | | | 创建进货退货单 |
| PurchaseReturn.modifyPurchaseReturn | | | 修改进货退货单 |
| PurchaseReturn.showPurchaseReturn | | | 查看所有进货退货单 |
| Sale.createSale | | | 创建销售单 |
| Sale.modifySale | | | 修改销售单 |
| Sale.showSale | | | 查看所有销售单 |
| SaleReturn.createSaleReturn | | | 创建销售退货单 |
| SaleReturn.modifySaleReturn | | | 修改销售退货单 |
| SaleReturn.showSaleReturn | | | 查看所有销售退货单 |
| SaleList.showSaleList | | | 查看所有销售明细表 |
| SaleList.exportToEXCEL | | | 导出到Excel |

**表34 salesbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| sales.Purchase | **语法** | public int purchase (PurchaseVO vo); | |
| **前置条件** | 销售人员使用创建进货单功能时 | |
| **后置条件** | 系统中存储进货单信息 | |
| sales.PurchaseReturn | **语法** | public int purchaseReturn (PurchaseReturnVO vo); | |
| **前置条件** | 销售人员使用创建进货退货单功能时 | |
| **后置条件** | 系统中存储进货退货单信息 | |
| sales.Sale | **语法** | public int sale (SaleVO vo); | |
| **前置条件** | 销售人员使用创建销售单功能时 | |
| **后置条件** | 系统中存储销售单信息 | |
| sales.SaleReturn | **语法** | public int saleReturn (SaleReturnVO vo); | |
| **前置条件** | 销售人员使用创建销售退货单功能时 | |
| **后置条件** | 系统中存储销售退货单信息 | |
| sales.showSaleList | **语法** | public SaleListVO showSaleList() | |
| **前置条件** | 销售人员使用查看销售明细表功能时 | |
| **后置条件** | 系统返回销售明细表 | |
| sales.modifyPurchase | **语法** | public int modifyPurchase (PurchaseVO vo); | |
| **前置条件** | 总经理需要修改进货单 | |
| **后置条件** | 修改进货单 | |
| sales.modifyPurchaseReturn | **语法** | public int modifyPurchaseReturn (PurchaseReturnVO vo); | |
| **前置条件** | 总经理需要修改进货退货单 | |
| **后置条件** | 修改进货退货单 | |
| sales.modifySale | **语法** | public int modifySale (SaleVO vo); | |
| **前置条件** | 总经理需要修改销售单 | |
| **后置条件** | 修改销售单 | |
| sales.modifySaleReturn | **语法** | public int modifySaleReturn (SaleReturnVO vo); | |
| **前置条件** | 总经理需要修改销售退货单 | |
| **后置条件** | 修改销售退货单 | |
| sales.showPurchase | **语法** | public ArrayList<PurchaseVO> showPurchase(); | |
| **前置条件** | 需要显示所有进货单 | |
| **后置条件** | 显示所有进货单 | |
| sales.showPurchaseReturn | **语法** | public ArrayList<PurchaseReturnVO> showPurchaseReturn(); | |
| **前置条件** | 需要显示所有进货退货单 | |
| **后置条件** | 显示所有进货退货单 | |
| sales.showSale | **语法** | public ArrayList<SaleVO> showSale(); | |
| **前置条件** | 需要显示所有销售单 | |
| **后置条件** | 显示所有销售单 | |
| sales.showSaleReturn | **语法** | public ArrayList<SaleReturnVO> showSaleReturn(); | |
| **前置条件** | 需要显示所有销售退货单 | |
| **后置条件** | 显示所有销售退货单 | |
| Sales.exportSaleListToExcel | **语法** | public int exportSaleListToExcel(); | |
| **前置条件** | 导出销售明细表到Excel | |
| **后置条件** | 创建销售明细表的Excel文件 | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| FileFactory.getSalesDatabase | | | 得到Sales文件的引用 |
| salesDataService.getSaleList() | | | 返回一个SaleListPO对象 |
| salesDataService.purchase(PurchasePOpo) | | | 插入一个PurchasePO对象 |
| salesDataService.purchaseReturn(PurchaseReturnPOpo) | | | 插入一个PurchaseReturnPo对象 |
| salesDataService.Sale(SalePOpo) | | | 插入一个SalePO对象 |
| salesDataService.SaleReturn(SaleReturnPOpo) | | | 插入一个SaleReturnPO对象 |

**(4)业务逻辑层的动态模型**

图59标明了进销存系统中，当用户选择销售管理之后，创建销售单业务逻辑处理的相关对象之间的协作。



**图59 创建销售单的顺序图**

如图60所示的sale对象状态图描述了sale对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着showMembers方法被ui调用，sale进入Member状态。之后通过添加商品进入stockGoods状态，再通过匹配促销策略进入Promotion状态。最后随着checkOut方法被ui调用，sale进入submit状态，一个销售单创建完成。

****

**图60 Sale对象的状态图**

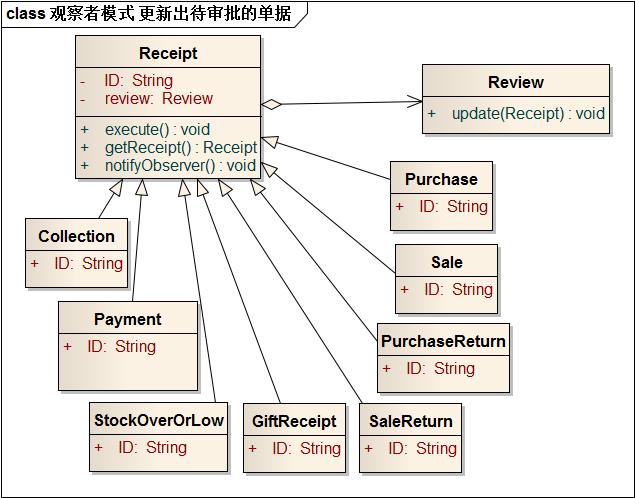
**4.2.3客户端业务逻辑层模块的设计原理**

利用委托式控制风格，每个模块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。这些对象分别承担一定的职责，做出一定的决策，从而实现大的职责。

Receipt模块单据的提交提醒和审批结果回复，使用了观察者模式，实时检测更新动态。

1. **单据的审批提醒的观察者模式**

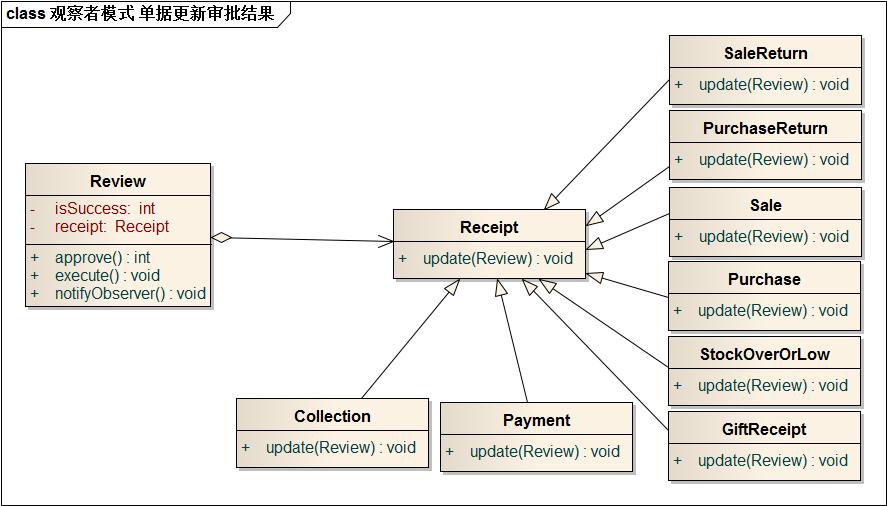
Review类需要实时更新检测是否有待审批的单据并提醒总经理，这里Review是观察者，Receipt是被观察者。当一个单据被创建，将调用notifyObserver方法，提醒Review已经被更新。

****

**图61 单据审批提醒的观察者模式图**

1. **审批结果回复的观察者模式**

Receipt需要实时检测单据是否已被审批，并提醒用户。Receipt是观察者，Review是被观察者。当一个单据被审批，将调用notifyObserver方法，提醒Receipt已经被更新。

****

**图62 审批结果回复的观察者模式图**

# 4.3客户端网络模块的静态结构和动态职责

由于数据是存储在服务器上的，客户端访问数据必须通过网络通信层，具体是由客户端的业务逻辑层和服务器端的数据层通过网络通信层进行通信。如图61所示。

具体网络通信层的实现是通过远程方法调用RMI来实现的。



**图63客户端数据通信层模块的描述**

**4.3.1客户端网络模块局部模块的职责**

其实网络通信层的实际运作代码是由RMI API所完成的，在客户端我们只需要部署RMI服务的Stub文件和Java RMI类库即可。如表35所示。

**表35 网络通信层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| IDataRemoteService\_Stub | 给业务逻辑层提供服务接口 |
| Java RMI 类库 | 实现RMI机制 |

# 4.4服务器端网络模块的静态结构和动态职责

**4.4.1服务器端网络模块局部模块的职责**

由于数据是存储在服务器上的，客户端访问数据必须通过网络通信层，具体是由客户端的业务逻辑层和服务器端的数据层通过网络通信层进行通信。

具体服务器端模块中只需要包含Java RMI类库即可。但是其数据层实现RMI的类需要符合以下条件：

****

**图64服务器数据层实现RMI的类的描述**

**4.4.2服务器端网络模块局部模块的接口规范**

SerService 是被业务逻辑层调用的接口。为了实现RMI，SerService需要继承 Remote接口，而数据层中的实现这个接口的类serServiceSerializableFileImpl都需要继承UnicastRemoteObject类。

以下是服务器网络模块的RMI详细的规范：

**表36服务器网络模块的RMI接口规范**

|  |  |
| --- | --- |
| **类/接口** | **规范** |
| 远程服务接口 | extends Remote，方法 throws RemoteException |
| 远程服务的实现类 | extendsUnicastRemoteObject |

**表37服务器网络模块的RMI接口返回值的规范**

|  |  |
| --- | --- |
| **类/接口** | **规范** |
| 作为远程引用传递 | 该类需要extends Remote，方法 throws RemoteException |
| 通过序列化进行复制 | Implements Serializable |

# 4.5服务器端数据层模块的静态结构和动态职责

**4.5.1服务器端数据层模块局部模块的职责**

服务器端数据层主要负责数据的持久化及数据访问接口。对于持久化数据的集体载入、集体保存和增、删、改、查。这些服务由DataService接口提供。本系统（进销存系统）程序的持久化数据的保存以序列化文件的形式实现。具体如图63所示。



**图65数据层模块的描述**

如表38所示为数据层模块的职责。

**表38数据层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| FileFactory | 持久化数据工厂，提供获得不同数据的工厂方法 |
| SerService | 持久化外部文件的接口，提供集体载入、集体保存、增、删、改、查服务。 |
| FileFactorySerializableFilelmpl | 基于序列化的持久化数据工厂的实现 |
| SerServicSerializableFileImpl | 基于序列化的持久化外部文件的接口，提供集体载入、集体保存、增、删、改、查服务。 |

**4.5.2服务器端数据层模块局部模块的接口规范**

服务器端数据层模块的接口规范如表39所示。

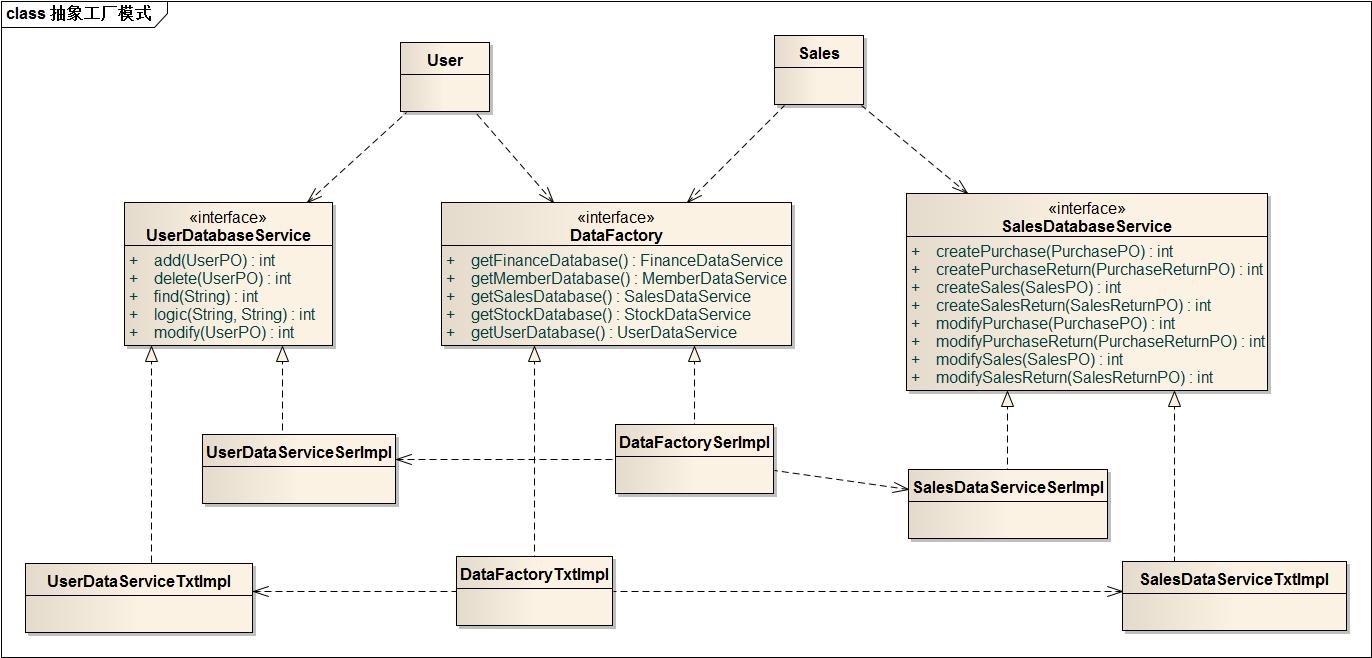
**表39数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口**） | | |
| FileFactory.getUserData | **语法** | Public SerService getUserData ()throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 得到User外部文件的引用 |
| UserDataService.add | **语法** | public int add(UserPO po) |
| **前置条件** | 同样ID的po在文件中不存在 |
| **后置条件** | 在外部文件中增加一个po记录 |
| UserDataService.check | **语法** | public boolean check(UserPO po) |
| **前置条件** | 无 |
| **后置条件** | 返回用户名和密码是否匹配 |
| UserDataService.delete | **语法** | public int delete(UserPO po) |
| **前置条件** | 在外部文件中存在某一个ID 的po需要删除 |
| **后置条件** | 删除一个po对象 |
| UserDataService.modify | **语法** | public int modify(UserPO po) |
| **前置条件** | 在外部文件中存在某一个ID 的po需要更新 |
| **后置条件** | 更新一个po对象 |
| UserDataService.showUserInfo | **语法** | public UserPO showUserInfo (String ID) |
| **前置条件** | 无 |
| **后置条件** | 按ID进行查找返回相应的UserPO的结果 |
| FileFactory.geStockData | **语法** | Public SerService getStockData ()throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 得到Stock外部文件的引用 |
| StockDataService. addGoodsClass | **语法** | public int addGoodsClass (GoodsClassPO po) |
| **前置条件** | 需要新建一个商品分类 |
| **后置条件** | 向外部文件中添加一个GoodsClassPo |
| StockDataService. deleteGoodsClass | **语法** | public int deleteGoodsClass (GoodsClassPO po) |
| **前置条件** | 外部一个商品分类需要删除 |
| **后置条件** | 删除文件中的一个GoodsClasspO |
| StockDataService. modifyGoodsClass | **语法** | public int modifyGoodsClass (GoodsClassPO po); |
| **前置条件** | 外部一个商品分类需要修改 |
| **后置条件** | 修改文件中一个GoodsClassPo |
| StockDataService. addGoods | **语法** | public int addGoods(GoodsPO po) |
| **前置条件** | 需要新增一个商品 |
| **后置条件** | 向外部文件中添加一个GoodsPO对象 |
| StockDataService. deleteGoods | **语法** | public int deleteGoods (GoodsPO po) |
| **前置条件** | 外部一个商品需要被删除 |
| **后置条件** | 删除文件中的一个GoodsPO |
| StockDataService. modifyGoods | **语法** | public int modifyGoods (GoodsPO po) |
| **前置条件** | 外部一个商品需要修改 |
| **后置条件** | 修改文件中一个GoodsPO |
| StockDataService. findGoods | **语法** | public ArrayList<GoodsPO>findGoods (String message) |
| **前置条件** | 需要根据关键字来查找商品 |
| **后置条件** | 返回对应的GoodsPO对象 |
| StockDataService. showStock | **语法** | public ArrayList<GoodsPO>showStock (String beginDate,String endDate) |
| **前置条件** | 需要查看一个时间段内的库存商品信息 |
| **后置条件** | 返回该时间段内的库存商品信息 |
| StockDataService. checkStock | **语法** | public ArrayList<GoodsPO>checkStock(SalesPO po) |
| **前置条件** | 需要盘点当天的库存 |
| **后置条件** | 返回当天的库存快照 |
| StockDataService. addStockOverOrLow | **语法** | public int addStockOverOrLow(StockOverOrLowPO po) |
| **前置条件** | 无 |
| **后置条件** | 向外部文件中添加一个StockOverOrLowPO |
| StockDataService. addGift | **语法** | public int addGift(GiftPO po) |
| **前置条件** | 需要进行一次库存赠送 |
| **后置条件** | 向外部文件中添加一个GiftPO |
| StockDataService. addStockError | **语法** | public int addStockError (StockErrorPO po) |
| **前置条件** | 无 |
| **后置条件** | 向外部文件中写入一个StockErrorPO |
| FileFactory.getSalesDatabase | **语法** | Public SerService getSalesDatabase ()throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 得到Sales外部文件的引用 |
| salesDataService. getSaleList | **语法** | public saleListPO getSaleList() |
| **前置条件** | 请求查看销售明细表 |
| **后置条件** | 返回一个销售明细表PO |
| salesDataService. Purchase | **语法** | public int purchase(PurchasePO po) |
| **前置条件** | 符合进货条件 |
| **后置条件** | 存储一个PurchasePO对象 |
| salesDataService. PurchaseReturn | **语法** | public int purchaseReturn(PurchaseReturnPO po) |
| **前置条件** | 符合退货条件 |
| **后置条件** | 存储一个PurchaseReturnPO 对象 |
| salesDataService. Sale | **语法** | public int sale(SalePO po) |
| **前置条件** | 符合销售条件 |
| **后置条件** | 存储一个SalePO对象 |
| salesDataService. SaleReturn | **语法** | public int saleReturn(SaleReturnPO po) |
| **前置条件** | 符合退货条件 |
| **后置条件** | 存储一个SaleReturnPO对象 |
| FileFactory.getMemberData | **语法** | Public SerService getMemberData ()throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 得到Member外部文件的引用 |
| memberDataService.add | **语法** | public int add(MemberPO po) |
| **前置条件** | member不重名 |
| **后置条件** | 存储一个memberPO对象 |
| memberDataService.delete | **语法** | public int delete(MemberPO po) |
| **前置条件** | 该member不存在应收应付 |
| **后置条件** | 删除一个memberPO对象 |
| memberDataService.modify | **语法** | public int modify(MemberPO po) |
| **前置条件** | member存在 |
| **后置条件** | 修改一个memberPO对象信息 |
| memberDataService. find | **语法** | public MemberPO find(String message) |
| **前置条件** | 用户输入关键词 |
| **后置条件** | 反馈出memberPO信息 |
| FileFactory.getFinanceData | **语法** | Public SerService getFinanceData ()throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 得到Finance外部文件的引用 |
| FinanceDataService.findAccount | **语法** | public AccountPO findAccount(Account PO) throws RemoteException; |
| **前置条件** | 同样名字的AccountPO在Account中存在 |
| **后置条件** | 按账户名查找并返回相应的AccountPO记录 |
| FinanceDataService.addAccount | **语法** | public int addAccount(AccountPO po) throws RemoteException; |
| **前置条件** | 同样名字的AccountPO在Account中不存在 |
| **后置条件** | 在Account中增加一个AccountPO记录 |
| FinanceDataService.deleteAccount | **语法** | public int deleteAccount(AccountPO po) throws RemoteException; |
| **前置条件** | 在Account中存在某个AccountPO需要删除 |
| **后置条件** | 删除一个AccountPO记录 |
| FinanceDataService.modifyAccount | **语法** | public int modifyAccount(AccountPO po) throws RemoteException; |
| **前置条件** | 在Account中存在某个的AccountPO需要修改 |
| **后置条件** | 修改一个AccountPO记录 |
| FinanceDataService.createCollection | **语法** | public int createCollection (CollectionPO po) throws RemoteException; |
| **前置条件** | 符合收款条件 |
| **后置条件** | 存储一张收款单CollectionPO |
| FinanceDataService.createPayment | **语法** | public int createPayment(PaymentPO po) throws RemoteException |
| **前置条件** | 符合收款条件 |
| **后置条件** | 存储一张付款单PaymentPO |
| FinanceDataService.createCashlist | **语法** | public int createCashlist (CashlistPO po) throws RemoteException |
| **前置条件** | 符合创建现金费用单条件 |
| **后置条件** | 存储一张现金费用单CashlistPO |
| FinanceDataService.viewBSL | **语法** | Public BSLPO viewBSL() throws RemoteException |
| **前置条件** | 符合查看经营情况表条件 |
|  | 返回一个BSLPO记录 |
| FinanceDataService.initInfo | **语法** | public int initInfo(BeginInfoVO vo) throws RemoteException); |
| **前置条件** | 无 |
| **后置条件** | 存储期初信息 |
| FinanceDataService.getInfo | **语法** | Public BeginInfoPO getInfo(BeginInfoVO vo) throws RemoteException |
| **前置条件** | 无 |
| **后置条件** | 返回一个BeginInfoPO记录 |
| FileFactory.getReceiptData | **语法** | Public SerService getReceiptData ()throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 得到Receipt外部文件的引用 |
| ReceiptDataService.Add | **语法** | Public int Add(ReceiptPo po) throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 在数据库中增加一个po记录 |
| ReceiptDataService.find | **语法** | Public ReceiptPO find(String id)throws RemoteException |
| **前置条件** | 无 |
| **后置条件** | 按ID进行查找返回相应的ReceiptPO()结果 |
| ReceiptDataService.modify | **语法** | Public int Modify(ReceiptPo po)throws RemoteException |
| **前置条件** | 该单据id存在 |
| **后置条件** | 修改一个po |
| ReceiptDataService.show | **语法** | Public ArrayList<ReceiptPO> show()throws RemoteException |
| **前置条件** | 无 |
| **后置条件** | 返回所有的po |
| ReceiptDataService.init | **语法** | Public void init()throws RemoteException |
| **前置条件** | 无 |
| **后置条件** | 初始持久化数据库的使用 |
| FileFactory.getPromotionData | **语法** | Public SerService getPromotionData ()throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 得到Promotion外部文件的引用 |
| PromotionDataService.Add | **语法** | Public int Add(PromotionPo po) throws RemoteException; |
| **前置条件** | 无 |
| **后置条件** | 在数据库中增加一个po记录 |
| PromotionDataService.find | **语法** | Public PromotionPO find(String id)throws RemoteException |
| **前置条件** | 无 |
| **后置条件** | 按ID进行查找返回相应的PromotionPo()结果 |
| PromotionDataService.Modify | **语法** | public int Modify(PromotionPo po) throws RemoteException; |
| **前置条件** | 存在该id的PromotionPO对象 |
| **后置条件** | 更新PromotionPo |
| PromotionDataService.show | **语法** | Public ArrayList<PromotionPO> show()throws RemoteException |
| **前置条件** | 无 |
| **后置条件** | 返回所有的po |
| PromotionDataService.init | **语法** | Public void init()throws RemoteException |
| **前置条件** | 无 |
| **后置条件** | 初始持久化数据库的使用 |

**4.5.3服务器端数据层模块的设计原理**

服务器端数据层模块采用了抽象工厂的模式，这种模式对文件存储的实现提供了很好的可修改性。

如图所示，对于各个表格数据，我们可以有不同的实现：TXT文件存储，对象序列化存储。DataFactory是抽象工厂。DataFactorySerImpl和DataFactoryTxtImpl是具体的工厂，即实现了抽象工厂的接口。DatabaseFactorySerImpl实现中利用了SalesDataServiceSerImpl、UserDataServiceTxtImpl来创建不同的序列化文件。SalesDataServiceSerImpl则提供SalesDataService的服务。Logic层的Sales可以利用DataFactory提供的接口getSalesDataBase()得到相应的Sales数据，再使用其提供的数据访问服务SalesDataService，从而达到很好的灵活性。

****

**图66 抽象工厂模式实例**

# 5、依赖关系

图64和图65是客户端和服务器端各自的包之间的依赖关系。



**图67客户端开发包图**



**图68 服务器端开发包图**