**灯具公司进销存系统**

LSLS(Lighting company Longevity system)

软件详细设计描述文档

V3.0 正式版

南京大学SIF4工作组

2017-11-14

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| SIF4 | 2017-11-10 | 最初设计 | V1.0 正式版 |
| SIF4 | 2017-11-12 | 模块完善 | V2.0 正式版 |
| SIF4 | 2017-11-14 | 模块完善 | V3.0 正式版 |

目录

[更新历史 2](#_Toc498507011)

[1.引言 2](#_Toc498507012)

[1.1编制目的 2](#_Toc498507013)

[1.2词汇表 2](#_Toc498507014)

[1.3参考资料 2](#_Toc498507015)

[2.产品概述 2](#_Toc498507016)

[3.体系结构设计概述 3](#_Toc498507017)

[4.结构视角 3](#_Toc498507018)

[4.1业务逻辑层的分解 3](#_Toc498507019)

[4.1.1 userbl模块 3](#_Toc498507020)

[4.1.2 salesbl模块 7](#_Toc498507021)

[4.1.3 commoditybl模块 13](#_Toc498507022)

[4.1.4 bookbl模块 18](#_Toc498507023)

[4.1.5 accountbl模块 29](#_Toc498507024)

[4.1.6 docbl模块 34](#_Toc498507025)

[4.1.7 promotionbl模块 37](#_Toc498507026)

[5.依赖视角 40](#_Toc498507027)

# 1.引言

### 1.1编制目的

本报告详细完成对灯具公司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 |
| LSLS | 灯具公司进销存系统 | …… |
| …… | …… | …… |

### 1.3参考资料

# 2.产品概述

参考灯具公司进销存系统用例文档和灯具公司进销存系统软件需求规格说明文档中对产品的概括描述。

# 3.体系结构设计概述

参考灯具公司进销存系统概要设计文档中对体系结构设计的概述。

# 4.结构视角

### 4.1业务逻辑层的分解

业务逻辑层的开发包图参见软件体系结构文档图3。

### 4.1.1 userbl模块

（1）模块概述

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

Userbl模块的职责及接口参见软件系统结构描述文档对应表。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间通过添加接口增加灵活性。比如展示层和业务逻辑层之间，我们添加businesslogicservice.UserBLService接口。业务逻辑层和数据层之间，我们增加了DatabaseFactory.UserDatabase接口。UserPO是作为用户信息的持久化对象被添加到设计模型中去的。UserDatabase封装了关于User的数据集合的数据结构的秘密和信息管理的职责。

Userbl模块的设计如图1所示。

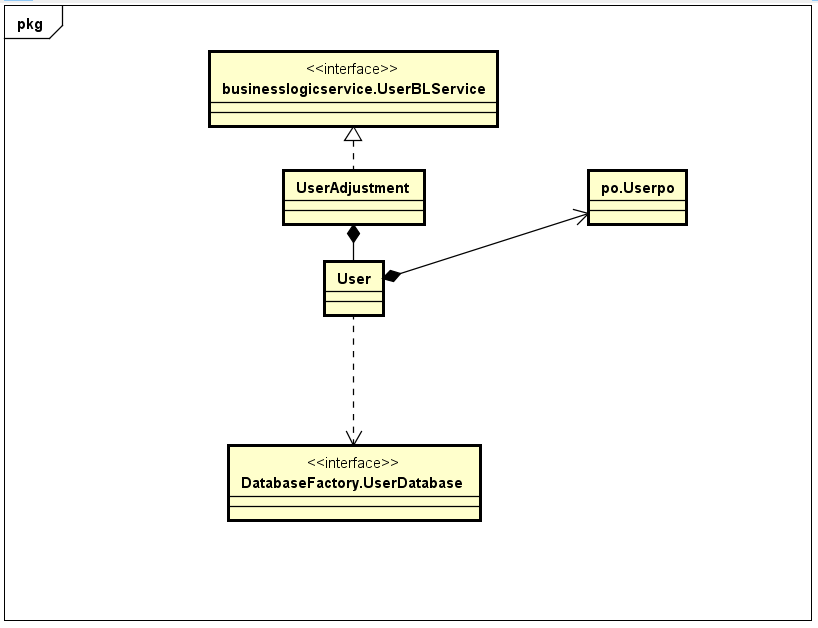


图1 userbl模块各个类的设计

Userbl模块各个类的职责如表1所示。

表1 userbl模块各个类的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| Login | 负责实现对应于登录界面所需要的服务 |
| UserAdjustment | 负责实现用户调整界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的姓名和密码，可以解决登录问题 |

（3）模块内部类的接口规范

User的接口规范如表2所示。

**表 userbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User.login | 语法 | | public boolean login(String id,String password); |
| 前置条件 | | password符合输入规则 |
| 后置条件 | | 查找是否存在相应的User，根据输入的password返回登陆验证的结果 |
| User.new | 语法 | | public static boolean userNew(); |
| 前置条件 | | 管理员成功登陆 |
| 后置条件 | | 新建成功则在数据库中添加新的User记录，否则返回相应出错信息 |
| User.edit | 语法 | | public static boolean userEdit(User opUser)； |
| 前置条件 | | 管理员已经选定某一User |
| 后置条件 | | 若修改操作正常进行则根据管理员的修改对相应User信息进行修改，否则返回出错信息 |
| User.delete | 语法 | | public static boolean userDel(User delUser); |
| 前置条件 | | 管理员已经选定某一User |
| 后置条件 | | 在数据库中将对应用户记录删除 |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| DatabaseFactory.getUserDatabase | | 得到User数据库的服务的引用 | |
| UserDataService.check(UserPO po) | | 在数据库中查找UserPO对象 | |
| UserDataService.new(UserPO po) | | 在数据库中新建UserPO对象 | |
| UserDataService.edit(UserPO po) | | 在数据库中修改引用的UserPO对象 | |

（4）业务逻辑层的动态模型

图2为管理员新增用户时，用户业务逻辑处理相关对象协作的顺序图。

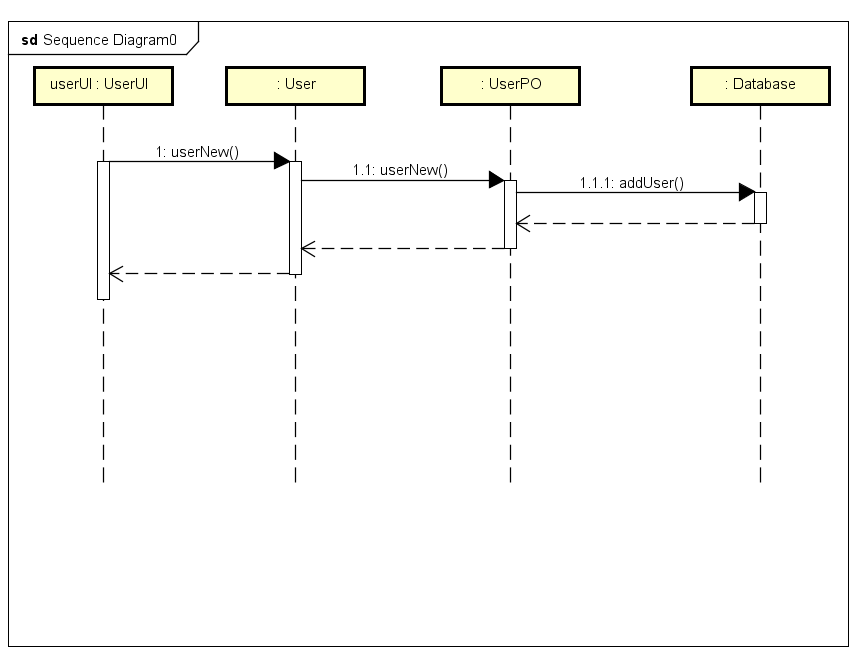
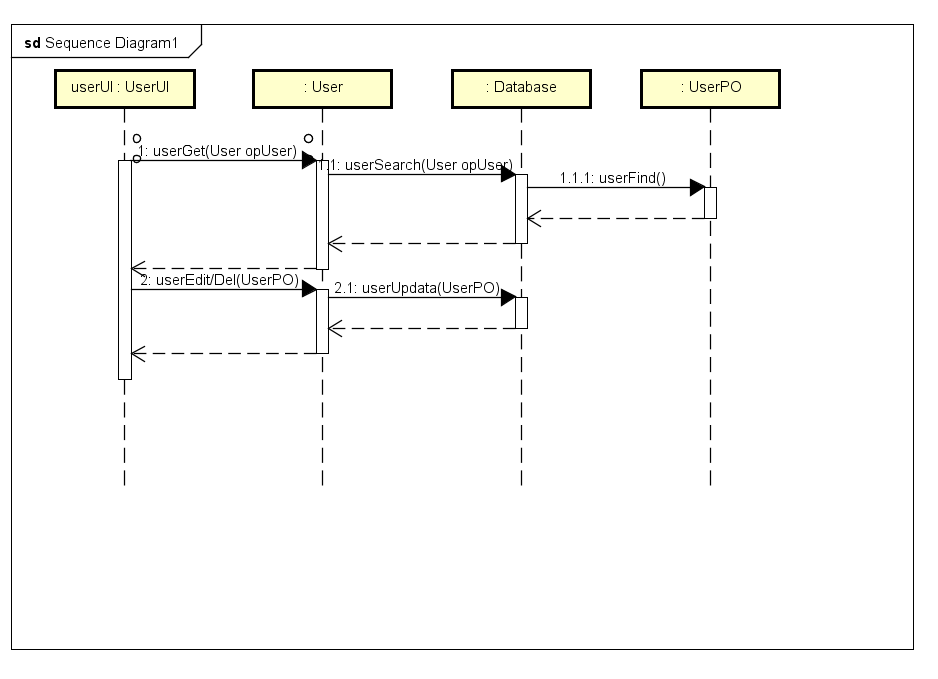


图2 新增用户的顺序图

图3为管理员修改/删除用户时，用户业务逻辑处理相关对象协作的顺序图。

图3 修改/删除用户的顺序图

如图4所示的状态图描述了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add/findUser方法被调用，引入User；之后通过edit/delUser进入可编辑User状态。管理员也可不对User进行编辑。

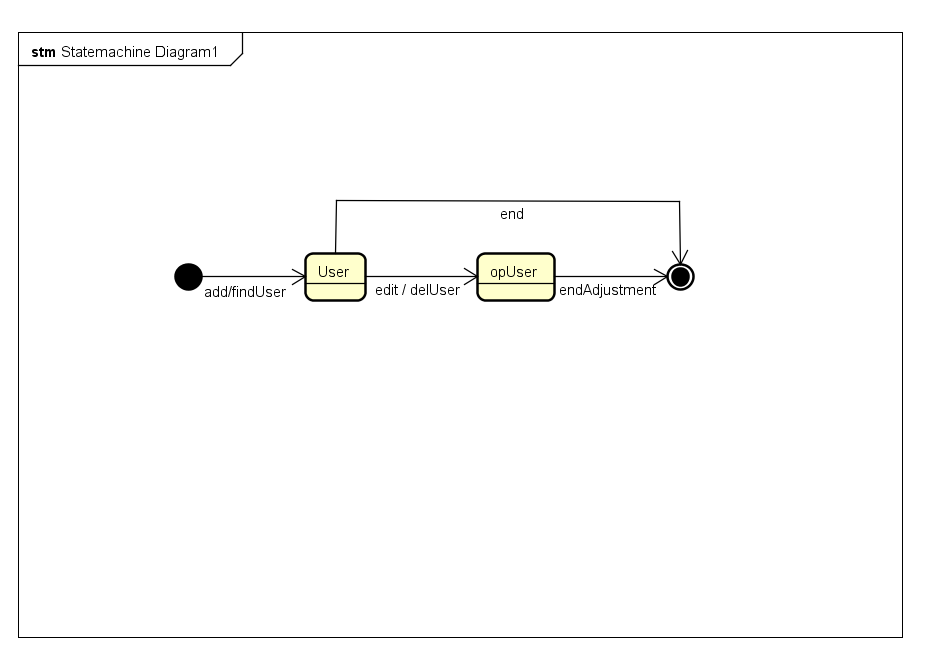


图4 User对象状态图

（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将委托给相应的领域对象。

其他略。

### 4.1.2 salesbl模块

（1）模块概述

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

Salesbl模块的职责及接口参见软件系统结构描述文档对应表。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间通过添加接口增加灵活性。

Salesbl模块的设计如图5、6、7、8所示。

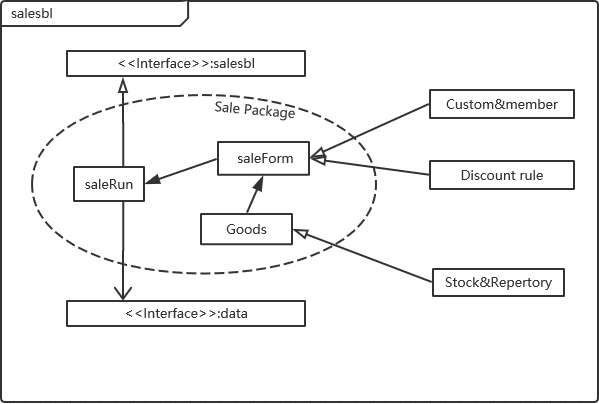


图5 salesbl模块sales包各个类的设计

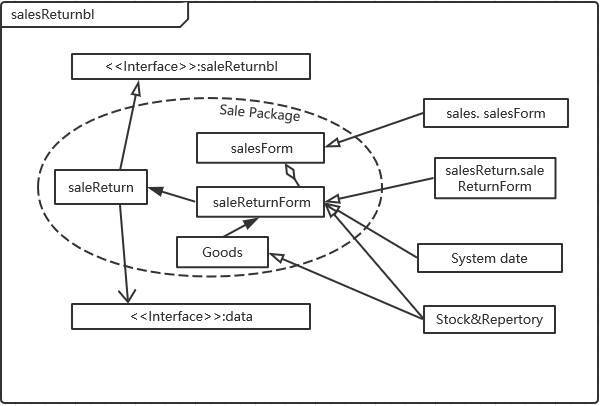


图6 salesbl模块salesReturn包各个类的设计

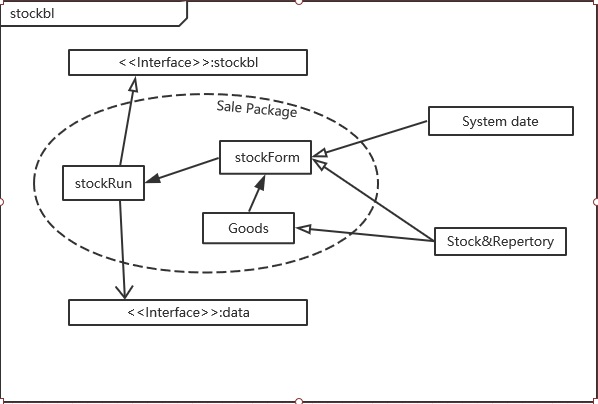


图7 salesbl模块stock包各个类的设计

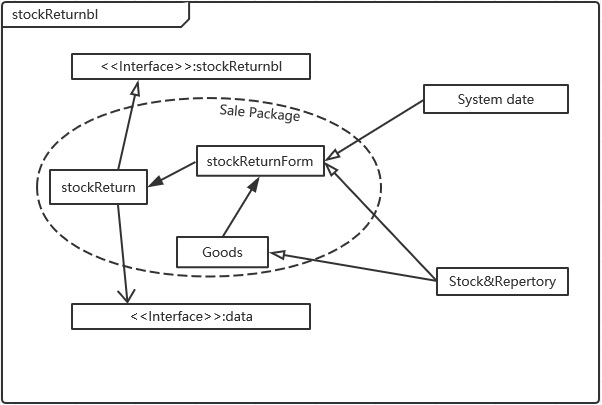


图8 salesbl模块stockReturn包各个类的设计

Salesbl模块各个类的职责如表3所示。

表3 salesbl模块各个类的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| Sales | 负责实现销售界面所需要的服务 |
| SalesReturn | 负责实现销售退货界面所需要的服务 |
| Stock | 负责实现进货界面所需要的服务 |
| StockReturn | 负责实现进货退货界面所需要的服务 |

（3）模块内部类的接口规范

Salesbl的接口规范如表4所示。

**表4 salesbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| saleRun.play | 语法 | public boolean Play(); |
| 前置条件 | 进货销售人员对销售单商品列表进行了操作 |
| 后置条件 | 商品列表所添加商品仓库存在 |
| saleRun.confirm | 语法 | public boolean Confirm()； |
| 前置条件 | 进货销售人员完成了销售操作并准备提交 |
| 后置条件 | 销售单核实无误 |
| stockRun.play | 语法 | public boolean Play(); |
| 前置条件 | 进货销售人员对进货单商品列表进行了操作 |
| 后置条件 | 商品列表所添加商品类别合法 |
| stockRun.confirm | 语法 | public boolean Confirm()； |
| 前置条件 | 进货销售人员完成了进货操作并准备提交 |
| 后置条件 | 进货单核实无误 |
| saleReturnRun.play | 语法 | public boolean Play(); |
| 前置条件 | 进货销售人员对销售退货单商品列表进行了操作 |
| 后置条件 | 商品列表符合初始销售单 |
| saleReturnRun.confirm | 语法 | public boolean Confirm()； |
| 前置条件 | 进货销售人员完成了销售退货操作并准备提交 |
| 后置条件 | 销售退货单核实无误 |
| saleReturnRun.findSaleForm | 语法 | public boolean findSaleForm(String ID) |
| 前置条件 | 销售过程发生过，销售单记录存在 |
| 后置条件 | ID号正确 |
| saleReturnRun.findReturnForm | 语法 | public boolean findReturnForm(String ID)； |
| 前置条件 | 销售过程发生过，退货单记录存在 |
| 后置条件 | 销售退货单ID正确 |
| stockReturnRun.play | 语法 | public boolean Play(); |
| 前置条件 | 进货销售人员对退货单商品列表进行了操作 |
| 后置条件 | 商品列表所添加退货商品仓库存在且充足 |
| stockReturnRun.confirm | 语法 | public boolean Confirm()； |
| 前置条件 | 进货销售人员完成了退货记录并准备提交 |
| 后置条件 | 退货单核实无误 |
| 需要的服务 | | |
| 服务名 | 服务 | |
| GetGoods(String ID); | 输入商品ID号获取商品的包括单价、类别、型号的详细信息 | |
| GetStock(String ID); | 输入商品ID号获取商品的在仓库的库存数量以及仓库库号 | |
| ChangeStock(String ID;Dword number); | 更改仓库库存 | |

（4）业务逻辑层的动态模型

图9为进货销售人员FindSaleForm时，Sales业务逻辑处理相关对象协作的顺序图。

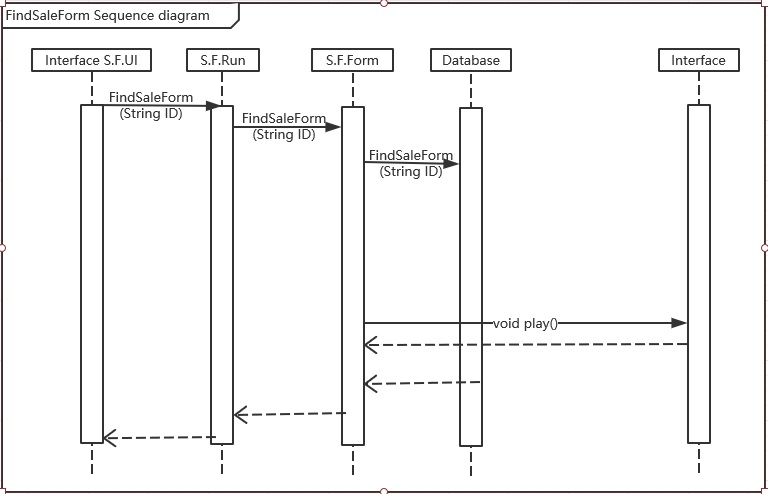


图9 FindSaleForm的顺序图

图10为进货销售人员对单据进行操作时，Sales业务逻辑处理相关对象协作的顺序图。

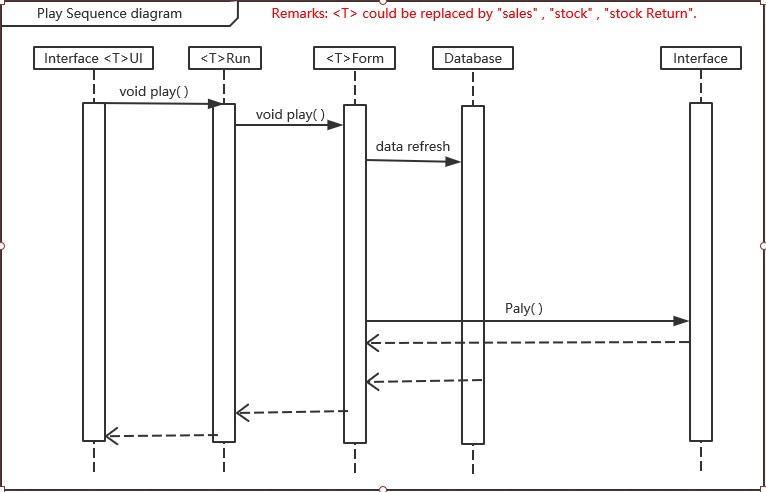
图10 play的顺序图

图11为进货销售人员进行销售时，Sales业务逻辑处理相关对象协作的顺序图。

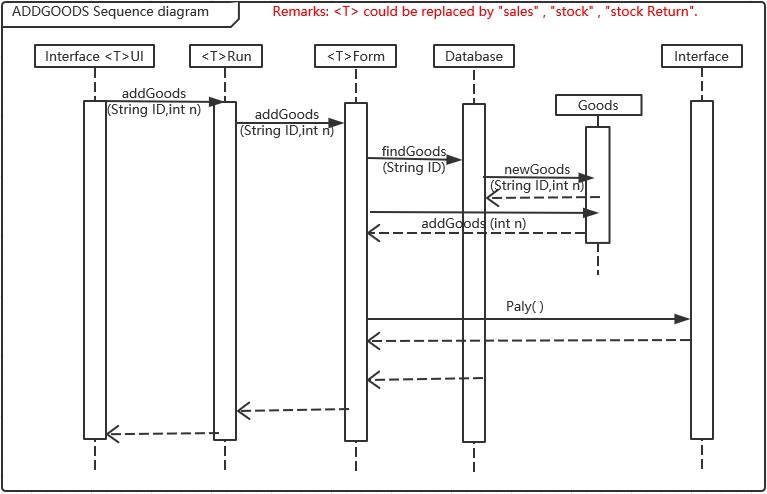
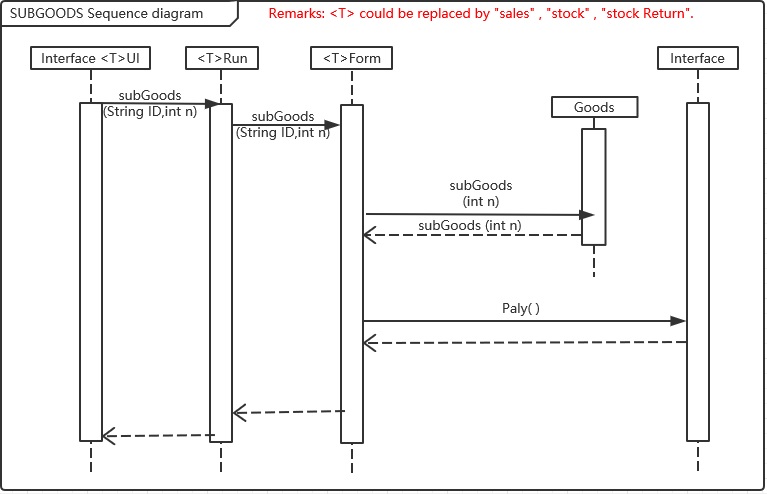
图11 销售的顺序图

图12为进货销售人员进行销售退货时，Sales业务逻辑处理相关对象协作的顺序图。

图12 销售退货的顺序图

（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将委托给相应的领域对象。

其他略。

### 4.1.3 commoditybl模块

（1）模块概述

commoditybl模块承担的需求参见需求规格文档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系统结构描述文档对应表。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间通过添加接口增加灵活性。

Commoditybl模块的设计如图13所示。

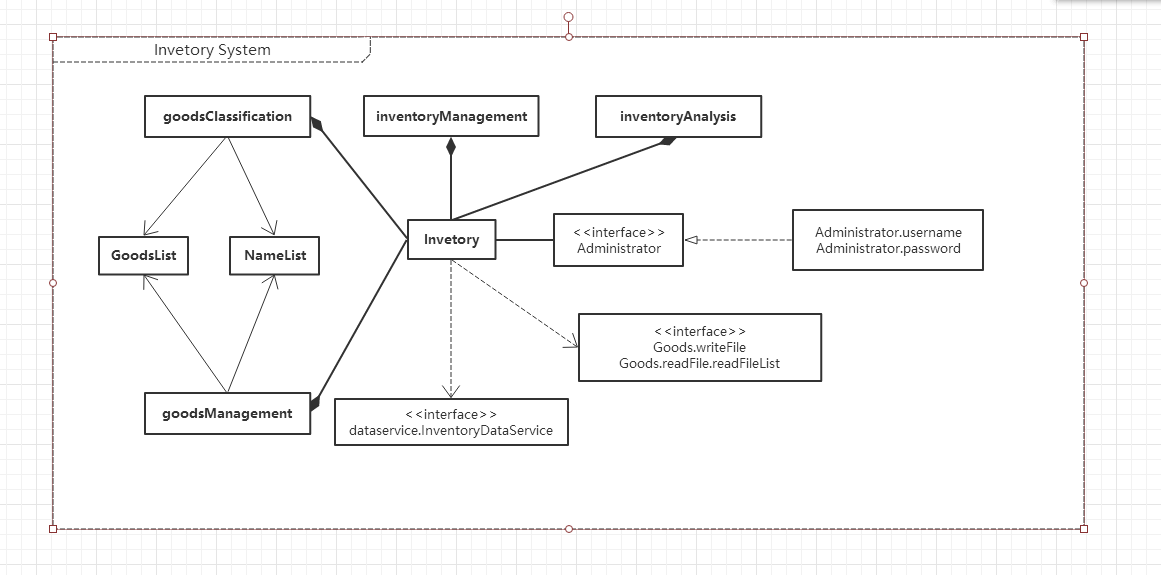


图13 commoditybl模块各个类的设计

Commoditybl模块各个类的职责如表5所示。

表5 commoditybl模块各个类的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| InventoryManagement | 负责实现库存管理界面所需要的服务 |
| InventoryAnalysis | 负责实现库存分析界面所需要的服务 |
| Inventory | 系统库存的领域模型对象 |
| goodsClassification | 负责实现商品分类界面所需要的服务 |
| goodsManagement | 负责实现商品管理界面所需要的服务 |

（3）模块内部类的接口规范

Commoditybl的接口规范如表6所示。

表6 commoditybl**模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| goodsClassification. Attribute. addClassification | 语法 | public void addClassification(String num, String str) |
| 前置条件 | 库存管理人员进行增加分类的操作 |
| 后置条件 | 根据分类名称确定是否已存在该类，若不存在则根据输入进行添加操作 |
| goodsClassification.Attribute. modClassification | 语法 | public void modClassification(int n, String num, String str) |
| 前置条件 | 库存管理人员进行修改分类操作 |
| 后置条件 | 根据类别名称或编号确定是否存在该类，并根据需求进行更改操作 |
| goodsClassification.Attribute. subClassification | 语法 | public void subClassification(int n) |
| 前置条件 | 库存管理人员进行删除分类操作 |
| 后置条件 | 根据类别名称或编号确定是否存在该类，若存在则进行删除分类操作 |
| goodsClassification.SubClass | 语法 | public boolean Judgement(String name) |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 根据商品分类的主类判断子类的存在是否合理 |
| goodsClassification.SuperClass | 语法 | public void SuperClass |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 将商品分类的主类数据进行存储 |
| goodsManagement.Main.Searching | 语法 | public void Searching(String name, String category, String num) |
| 前置条件 | 库存管理人员进行查看操作 |
| 后置条件 | 根据商品的名称、编号或类别确定是否存在该商品，若存在则查看商品信息 |
| goodsManagement.Main.addGoods | 语法 | public void addGoods(String name, String type, int stock, double bprice, double sprice) |
| 前置条件 | 库存管理人员进行增加商品的操作 |
| 后置条件 | 根据商品名称或编号确定是否已存在该商品，若不存在则输入商品信息进行添加操作 |
| goodsManagement.Main.subGoods | 语法 | public void subGoods(String name, String num) |
| 前置条件 | 库存管理人员进行删除商品的操作 |
| 后置条件 | 根据商品名称或编号确定是否已存在该商品，若存在则进行删除操作 |
| goodsManagement.Main.modGoods | 语法 | public void modGoods(String name, String num, int stock, double bprice, double sprice) |
| 前置条件 | 库存管理人员进行修改商品的操作 |
| 后置条件 | 根据商品名称或编号确定是否已存在该商品，若存在则进行修改操作 |
| goodsManagement.Main. ErrorJudgement | 语法 | public boolean ErrorJudgement() |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 进行输入判断，包含非法输入判断、重复查询判断、中断查询 |
| inventoryAnalysis.Analysis. Searching | 语法 | public boolean Searching |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 库存管理人员进行查看库存状况的操作 |
| inventoryAnalysis.Analysis.Date | 语法 | public void Date |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 系统根据管理员输入的时间段显示该时间段内的库存变化情况 |
| inventoryAnalysis.Documents. GivingDocument | 语法 | public void GivingDocument |
| 前置条件 | 库存管理人员根据库存分析提供库存赠送单 |
| 后置条件 | 系统生成库存赠送单并显示 |
| inventoryAnalysis.Documents. RemainingDocument | 语法 | public void RemainingDocument |
| 前置条件 | 库存管理人员根据库存分析提供库存报溢单 |
| 后置条件 | 系统生成库存报溢单并显示 |
| inventoryAnalysis.Documents. LackingDocument | 语法 | public void LackingDocument |
| 前置条件 | 库存管理人员根据库存分析提供库存报损单 |
| 后置条件 | 系统生成库存报损单并显示 |
| inventoryAnalysis.Documents. WarningDocument | 语法 | public void WarningDocument |
| 前置条件 | 库存管理人员根据库存分析提供库存报警单 |
| 后置条件 | 系统生成库存报警单并显示 |
| inventoryManagement.Viewing. Searching | 语法 | public void Searching |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 库存管理人员进行查看库存状况的操作 |
| inventoryManagement.Viewing. Date | 语法 | public void Date |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 系统根据管理员输入的时间段显示该时间段内的库存变化情况 |
| inventoryManagement.Checking | 语法 | public void checking |
| 前置条件 | 库存管理人员查看当日的库存快照 |
| 后置条件 | 系统显示当日的库存快照 |

（4）业务逻辑层的动态模型

图14为库存管理人员增加分类时，Commodity业务逻辑处理相关对象协作的顺序图。

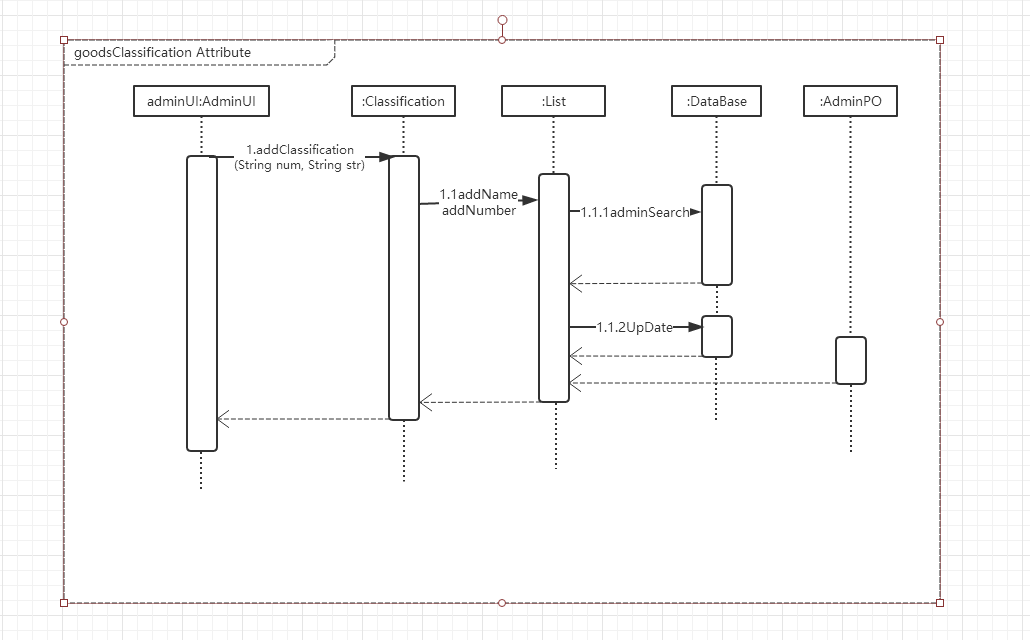


图14 增加分类的顺序图

图15为库存管理人员删除分类时，Commodity业务逻辑处理相关对象协作的顺序图。

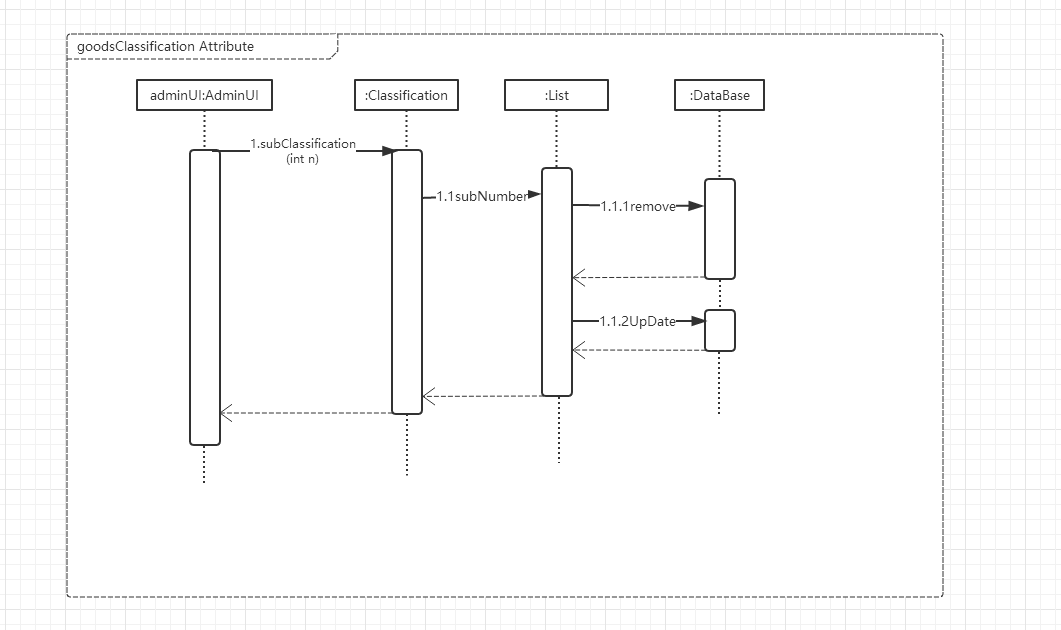


图15 删除分类的顺序图

图16为库存管理人员修改分类时，Commodity业务逻辑处理相关对象协作的顺序图。

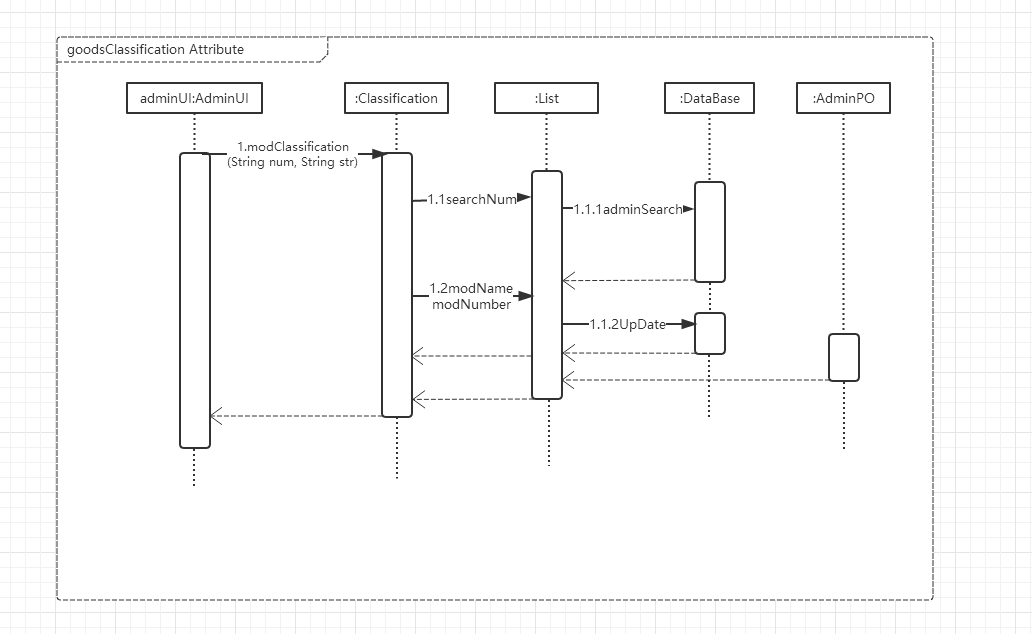


图16 修改分类的顺序图

如图17、18、19所示的状态图描述了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。

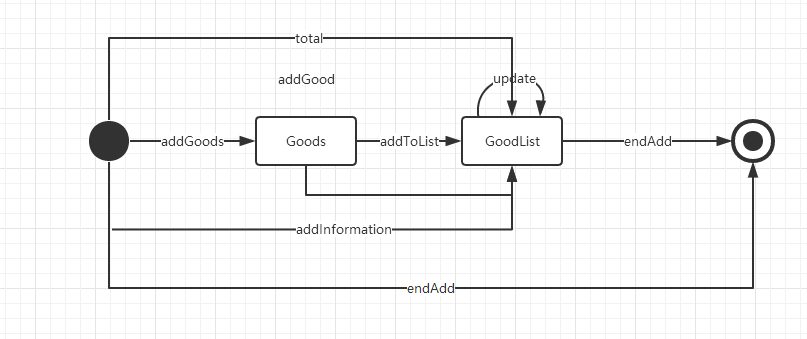


图17 Goods对象状态图（添加）

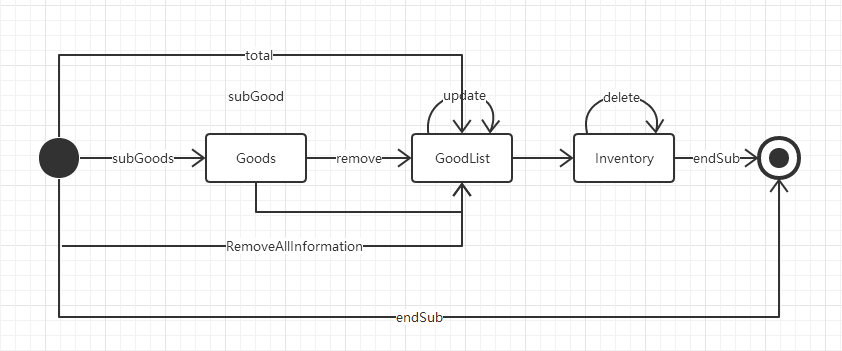


图18 Goods对象状态图（删除）

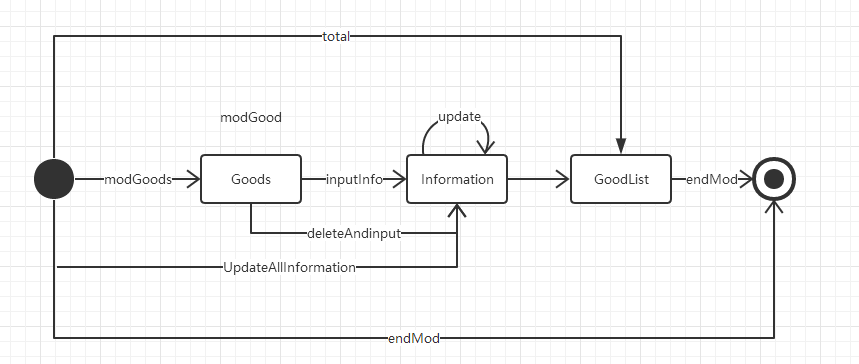


图19 Goods对象状态图（修改）

（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将委托给相应的领域对象。

其他略。

### 4.1.4 bookbl模块

（1）模块概述

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

Bookbl模块的职责及接口参见软件系统结构描述文档对应表。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间通过添加接口增加灵活性。

Bookbl模块的设计如图20所示

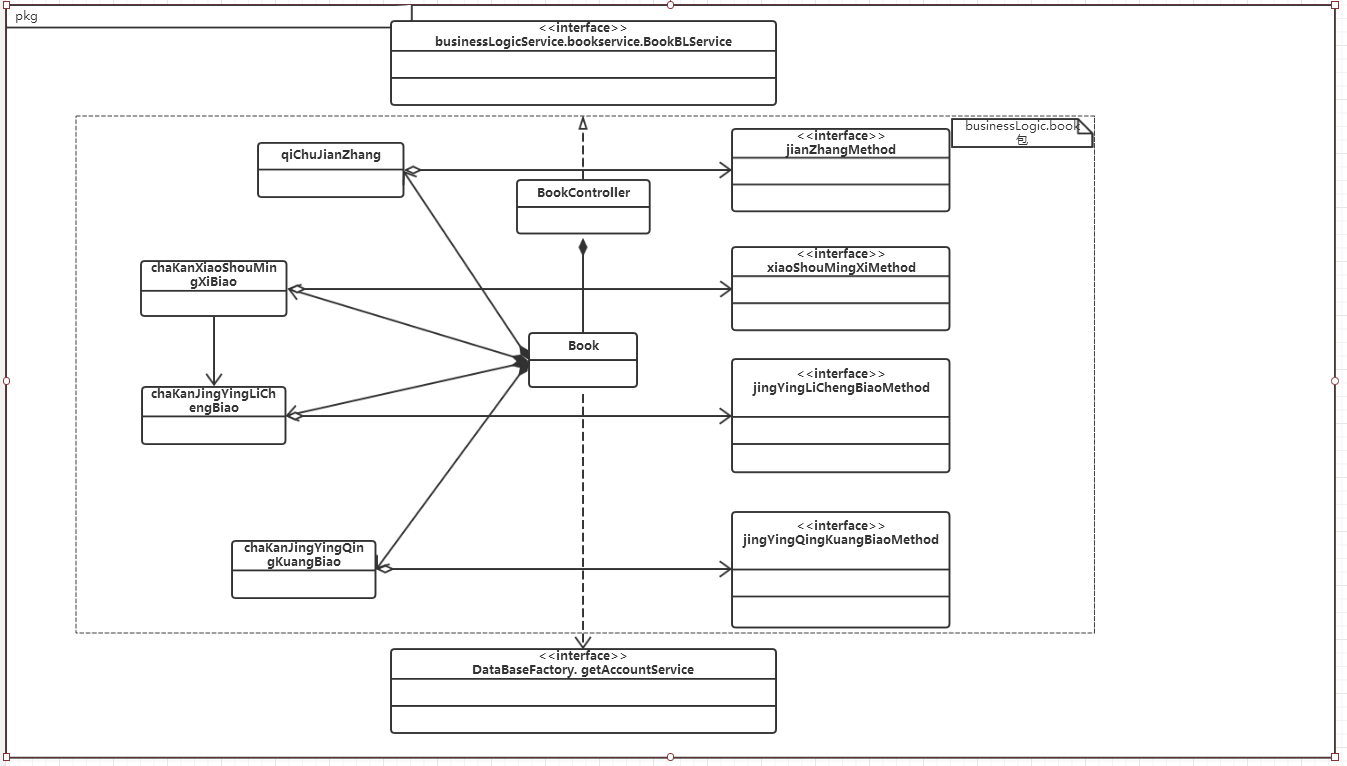


图20 bookbl模块各个类的设计

Bookbl模块各个类的职责如表7所示。

表7 bookbl模块各个类的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| chaKanXiaoShouMingXiBiao | 负责实现查看销售明细表界面所需要的服务 |
| chaKanJingYingLiChengBiao | 负责实现查看经营历程表界面所需要的服务 |
| chaKanJingYingQingKuangBiao | 负责实现查看经营情况表界面所需要的服务 |
| qiChuJianZhang | 负责实现期初建账界面所需要的服务 |

（3）模块内部类的接口规范

Bookbl的接口规范如表8所示。

**表8 bookbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| chaKanXiaoShouMingXiBiao. Timelog | 语法 | public void Timelog(String Time); |
| 前置条件 | 财务人员按时间区间进行筛选查询 |
| 后置条件 | 系统显示有关时间区间的类 |
| chaKanXiaoShouMingXiBiao.goodsName | 语法 | public void goodsName(String Goods); |
| 前置条件 | 财务人员按商品名进行筛选查询 |
| 后置条件 | 系统显示有关商品名的类 |
| chaKanXiaoShouMingXiBiao. Client | 语法 | public void Client(String Name); |
| 前置条件 | 财务人员按客户进行筛选查询 |
| 后置条件 | 系统显示有关客户的类 |
| chaKanXiaoShouMingXiBiao.Operater | 语法 | public void Operater(String opName); |
| 前置条件 | 财务人员按业务员进行筛选查询 |
| 后置条件 | 系统显示有关业务员的类 |
| chaKanXiaoShouMingXiBiao. cangKu | 语法 | public void cangKu(String place); |
| 前置条件 | 财务人员按仓库进行筛选查询 |
| 后置条件 | 系统显示有关仓库的类 |
| chaKanXiaoShouMingXiBiao. out | 语法 | public void out(); |
| 前置条件 | 财务人员按时间区间，商品名，客户，业务员，仓库五种筛选条件进行查询 |
| 后置条件 | 系统显示符合上述条件的商品销售记录，以列表形式显示时间，商品名，型号，数量，单价，总额 |
| chaKanJingYingLiChengBiao. Xiao  ShouChuHuoDan | 语法 | public void xiaoShouChuHuoDan(); |
| 前置条件 | 财务人员选择销售类单据并选择销售出货单 |
| 后置条件 | 系统显示销售出货单的单据内容 |
| chaKanJingYingLiChengBiao. Xiao  ShouTuiHuoDan | 语法 | public void xiaoShouTuiHuoDan(); |
| 前置条件 | 财务人员选择销售类单据并选择销售退货单 |
| 后置条件 | 系统显示销售退货单的单据内容 |
| chaKanJingYingLiChengBiao. jinHuo  Dan | 语法 | public void jinHuoDan(); |
| 前置条件 | 财务人员选择销售类单据并选择进货单 |
| 后置条件 | 系统显示进货单的单据内容 |
| chaKanJingYingLiChengBiao. jinHuoT  uiHuoDan | 语法 | public void jinHuoTuiHuoDan(); |
| 前置条件 | 财务人员选择销售类单据并选择进货退货单 |
| 后置条件 | 系统显示进货退货单的单据内容 |
| chaKanJingYingLiChengBiao. fuKuan  Dan | 语法 | public void fuKuanDan(); |
| 前置条件 | 财务人员选择销售类单据并选择付款单 |
| 后置条件 | 系统显示付款单的单据内容 |
| chaKanJingYingLiChengBiao. ShouKu  anDan | 语法 | public void shouKuanDan(); |
| 前置条件 | 财务人员选择销售类单据并选择收款单 |
| 后置条件 | 系统显示收款单的单据内容 |
| chaKanJingYingLiChengBiao. xianJin  FeiYongDan | 语法 | public void xianJinFeiYongDan(); |
| 前置条件 | 财务人员选择销售类单据并选择现金费用单 |
| 后置条件 | 系统显示现金费用单的单据内容 |
| chaKanJingYingLiChengBiao. baoYi  Dan | 语法 | public void baoYiDan(); |
| 前置条件 | 财务人员选择销售类单据并选择报溢单 |
| 后置条件 | 系统显示报溢单的单据内容 |
| chaKanJingYingLiChengBiao. baoShun  Dan | 语法 | public void baoShunDan(); |
| 前置条件 | 财务人员选择销售类单据并选择报损单 |
| 后置条件 | 系统显示报损单的单据内容 |
| chaKanJingYingLiChengBiao. zengSon  gDan | 语法 | public void zengSongDan(); |
| 前置条件 | 财务人员选择销售类单据并选择赠送单 |
| 后置条件 | 系统显示销售赠送单的单据内容 |
| chaKanJingYingQingKuangBiao.xiaoShouShouRu() | 语法 | public void xiaoShouShouRu(); |
| 前置条件 | 财务人员查询销售收入 |
| 后置条件 | 系统显示销售收入内容 |
| chaKanJingYingQingKuangBiao.shangPinBaoYiShouRu() | 语法 | public void shangPinBaoYiShouRu(); |
| 前置条件 | 财务人员选择商品报溢收入 |
| 后置条件 | 系统显示商品报溢收入内容 |
| chaKanJingYingQingKuangBiao.  chengBanTiaoJiaShouRu() | 语法 | public void chengBanTiaoJiaShouRu(); |
| 前置条件 | 财务人员选择成本调价收入 |
| 后置条件 | 系统显示成本调价收入内容 |
| chaKanJingYingQingKuangBiao.jinHuoTuiHuoChaJia() | 语法 | public void jinHuoTuiHuoChaJia(); |
| 前置条件 | 财务人员选择进货退货差价 |
| 后置条件 | 系统显示进货退货差价内容 |
| chaKanJingYingQingKuangBiao. daiJinQuan() | 语法 | public void daiJinQuan(); |
| 前置条件 | 财务人员选择代金券 |
| 后置条件 | 系统显示代金券相关内容 |
| chaKanJingYingQingKuangBiao. shiJiShouKuanChaE  ShouRu() | 语法 | public void shiJiShouKuanChaE  ShouRu(); |
| 前置条件 | 财务人员选择实际收款差额收入 |
| 后置条件 | 系统显示实际收款差额收入内容 |
| chaKanJingYingQingKuangBiao. xiaoShouChengBen() | 语法 | public void xiaoShouChengBen(); |
| 前置条件 | 财务人员选择销售成本 |
| 后置条件 | 系统显示销售成本内容 |
| chaKanJingYingQingKuangBiao. shangPinBaoShun() | 语法 | public void shangPinBaoShun(); |
| 前置条件 | 财务人员选择商品报损 |
| 后置条件 | 系统显示商品报损内容 |
| chaKanJingYingQingKuangBiao. shangPinZengChu() | 语法 | public void shangPinZengChu(); |
| 前置条件 | 财务人员选择商品赠出 |
| 后置条件 | 系统显示商品赠出内容 |
| chaKanJingYingQingKuangBiao. zongZhiChu() | 语法 | public void zongZhiChu(); |
| 前置条件 | 财务人员选择总支出 |
| 后置条件 | 系统显示总支出内容 |
| chaKanJingYingQingKuangBiao. zongShouRu() | 语法 | public void zongShouRu(); |
| 前置条件 | 财务人员选择总收入 |
| 后置条件 | 系统显示总收入内容 |
| chaKanJingYingQingKuangBiao. liRun() | 语法 | public void liRun(); |
| 前置条件 | 财务人员选择利润 |
| 后置条件 | 系统显示利润内容 |
| qiChuJianZhang. getShangPinXinXi(String shangPinFenLei,String mouYiShangPinMingCheng,String leiBie,String xingHao,int jinJia,int shouJia,int zuiJinJinJia,int zuiJinShouJiaLiukong) | 语法 | public void getShangPinXinXi(String shangPinFenLei,String mouYiShangPinMingCheng,String leiBie,String xingHao,int jinJia,int shouJia,int zuiJinJinJia,int zuiJinShouJiaLiukong); |
| 前置条件 | 财务人员添加商品信息，包括商品分类，某一商品的名称，类别，型号，进价/售价，最近进价和最近售价留空 |
| 后置条件 | 系统以表格形式存储商品信息，包括商品分类，某一商品的名称，类别，型号，进价/售价，最近进价和最近售价留空 |
| qiChuJianZhang. getKeHuXinXi(String keHuFenLei,String mouYiKeHuMingCheng,String lianXiFnagShi,int yingShou,int yingFu) | 语法 | public void getKeHuXinXi(String keHuFenLei,String mouYiKeHuMingCheng,String lianXiFnagShi,int yingShou,int yingFu); |
| 前置条件 | 财务人员添加客户信息，包括客户分类，某一个客户的名称，联系方式，应收应付 |
| 后置条件 | 系统以表格形式存储客户信息，包括客户分类，某一个客户的名称，联系方式，应收应付 |
| qiChuJianZhang. getYingHangZhangHuXinXi(String mingCheng,int yuE) | 语法 | public void getYingHangZhangHuXinXi(String mingCheng,int yuE); |
| 前置条件 | 财务人员输入银行账户信息，包括名称，余额 |
| 后置条件 | 系统以表格形式存储银行账户信息，包括名称，余额 |
| qiChuJianZhang. out() | 语法 | public void out(); |
| 前置条件 | 财务人员点击完成期初建账 |
| 后置条件 | 系统生成期初建账账单并将期初单独存储起来 |

（4）业务逻辑层的动态模型

图21为财务人员查看销售明细表时，Book业务逻辑处理相关对象协作的顺序图。

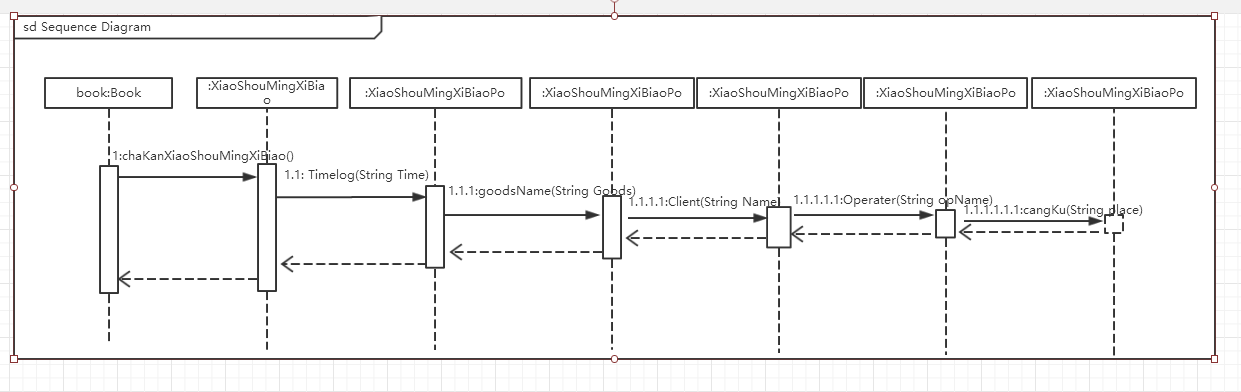


图21 查看销售明细表的顺序图

图22、23、24为财务人员查看经验历程表时，Book业务逻辑处理相关对象协作的顺序图。

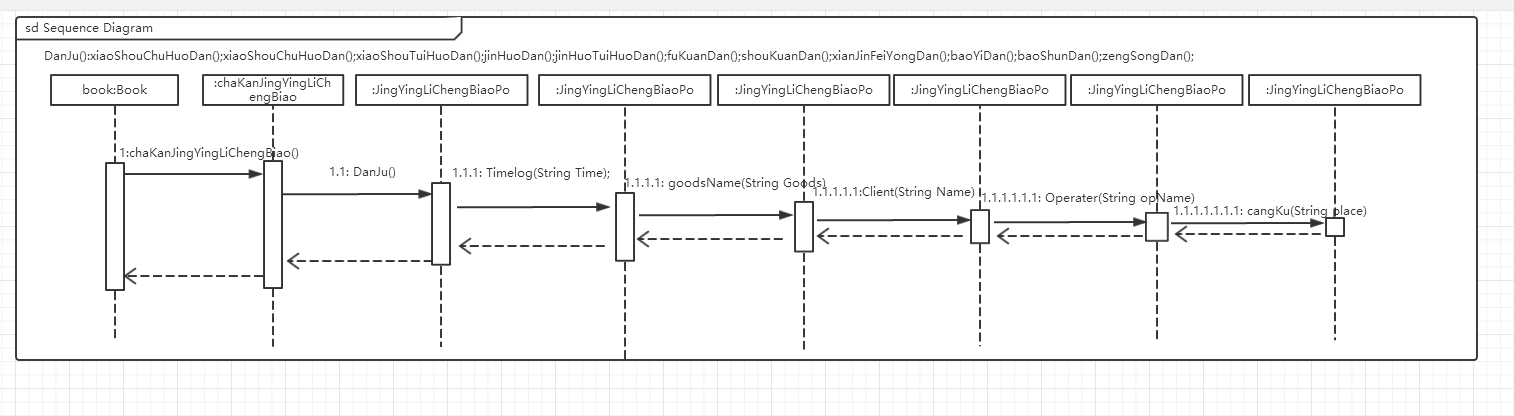


图22 单据的顺序图

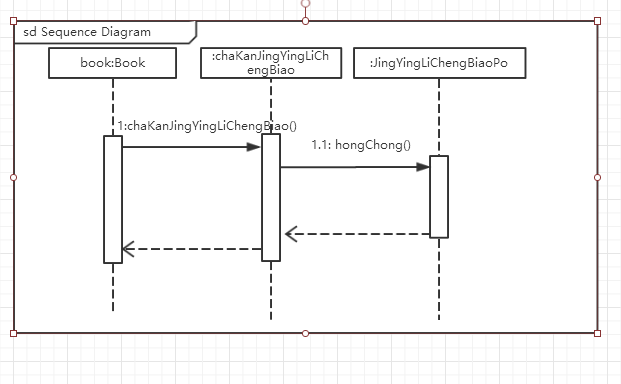


图23 红冲的顺序图

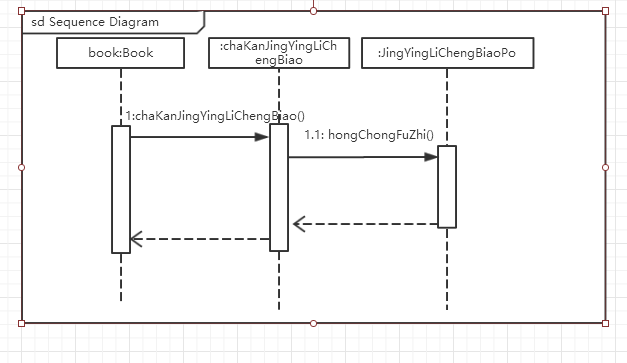


图24 红冲复制的顺序图

图25-36为财务人员查看经营情况表时，Book业务逻辑处理相关对象协作的顺序图。

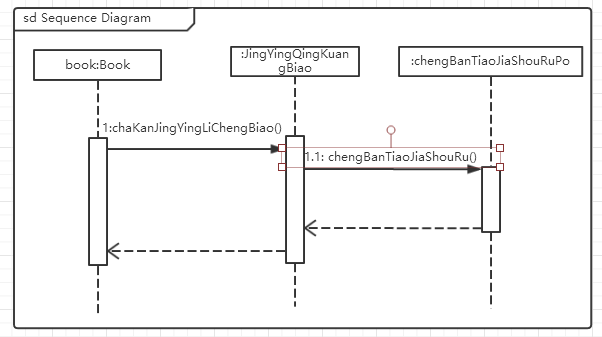


图25成本调价收入的顺序图

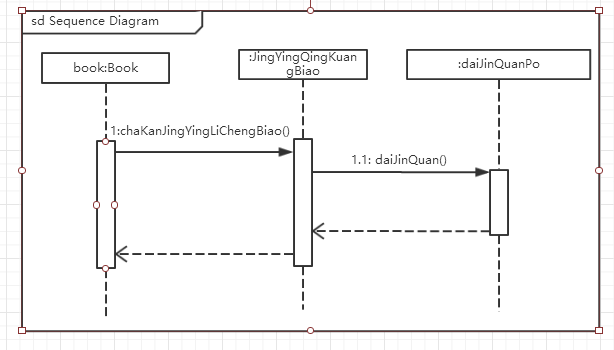


图26 代金券的顺序图

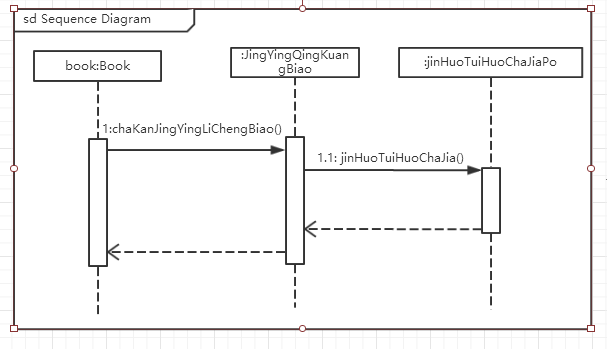


图27 进货退货差价的顺序图

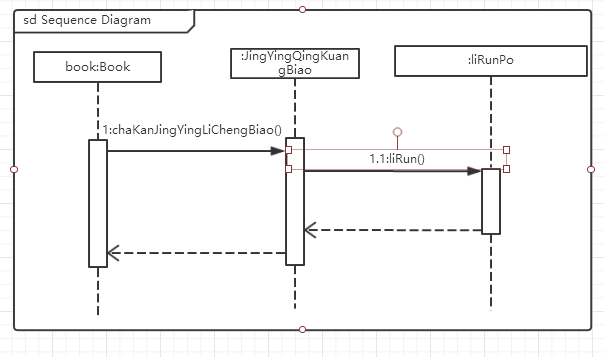


图28 利润的顺序图

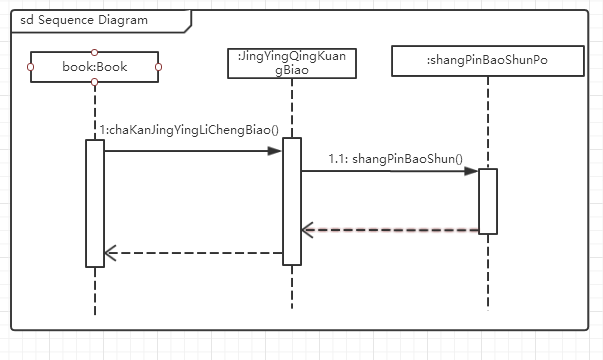


图29 商品报损的顺序图

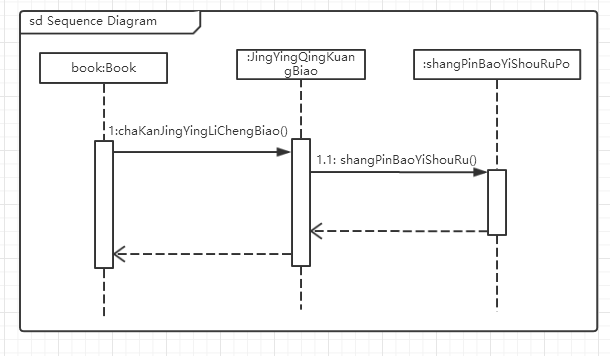


图30 商品报溢收入的顺序图

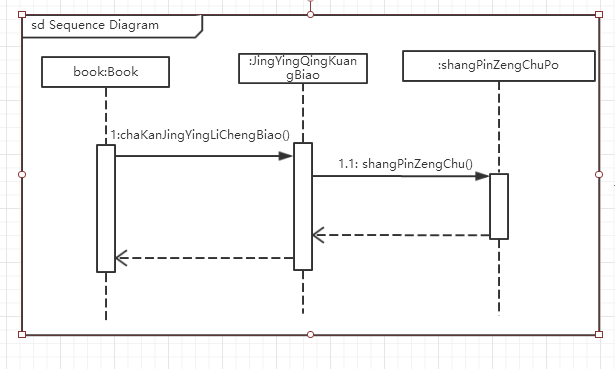


图31 商品赠出的顺序图

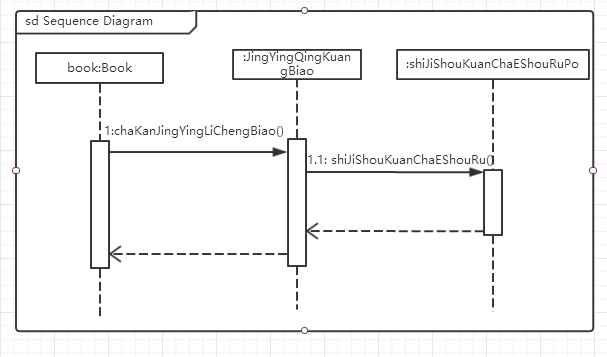


图32 实际收款差额收入的顺序图

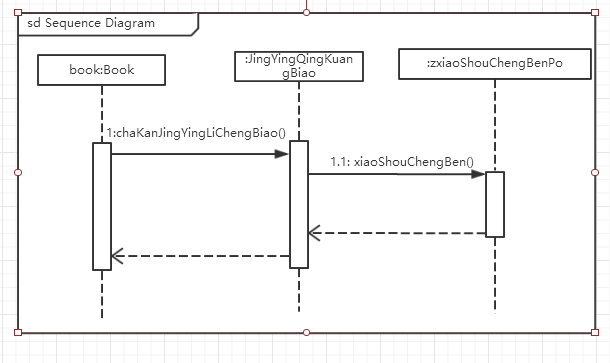


图33 销售成本的顺序图

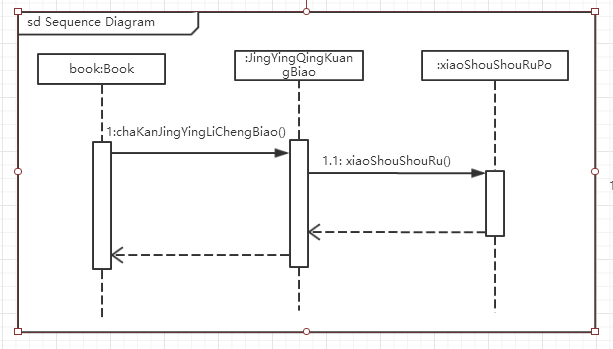


图34 销售收入的顺序图

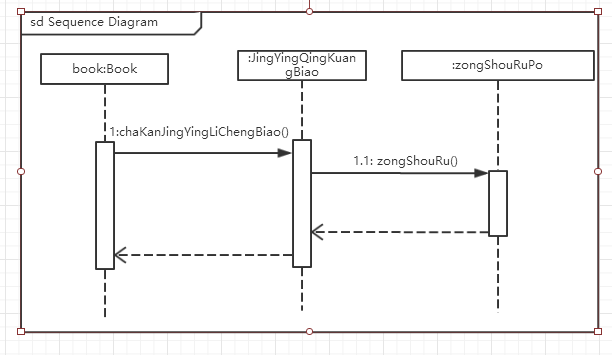


图35 折让后总收入的顺序图

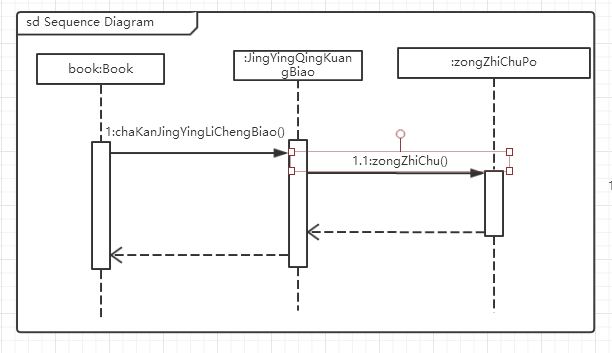


图36 总支出的顺序图

图37为财务人员期初建账时，Book业务逻辑处理相关对象协作的顺序图。

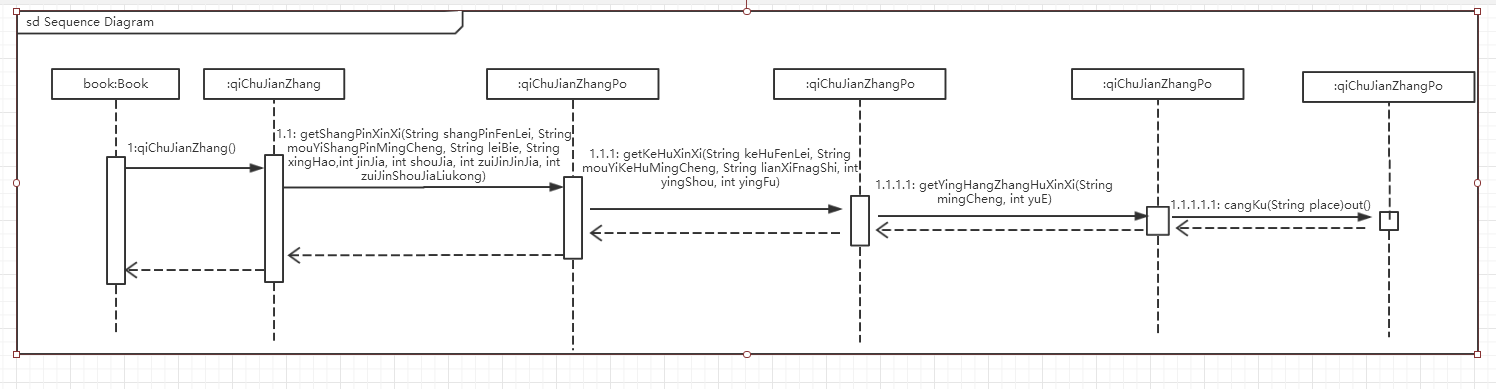


图37 期初建账的顺序图

如图38、39所示的状态图描述了Book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。

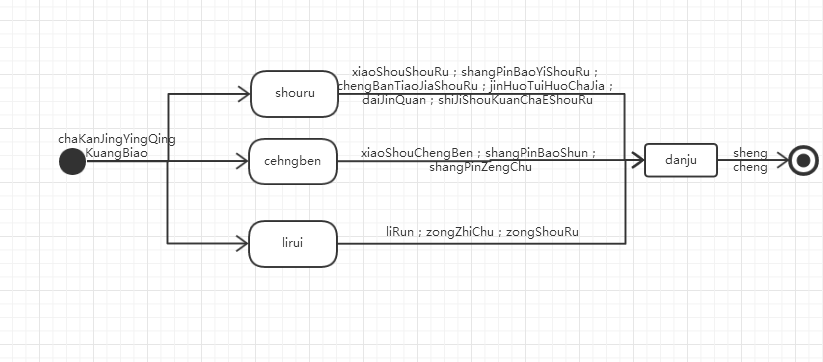


图38 查看经营历程期间状态图

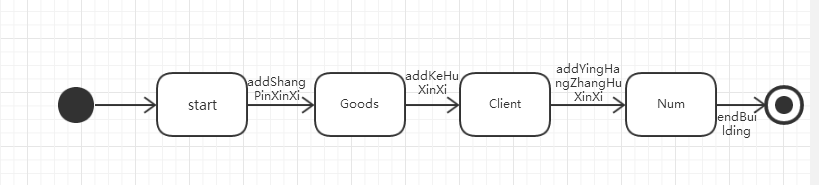


图39期初建账期间状态图

（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将委托给相应的领域对象。

其他略。

### 4.1.5 accountbl模块

（1）模块概述

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

Accountbl模块的职责及接口参见软件系统结构描述文档对应表。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间通过添加接口增加灵活性。

account模块的设计如图40所示。

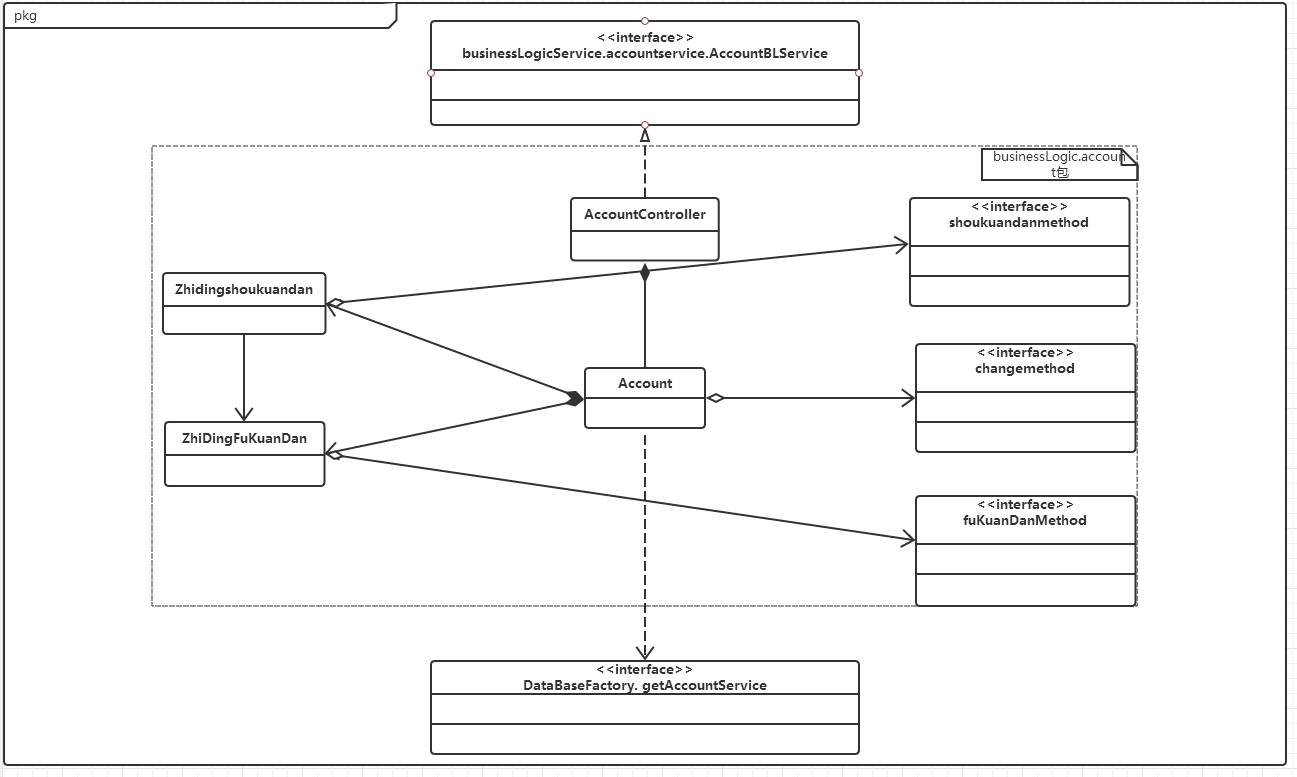


图40 accountbl模块各个类的设计

Accountbl模块各个类的职责如表9所示。

表9 accountbl模块各个类的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| Zhanghuguanli | 负责实现对账户管理界面所需要的服务 |
| Zhidingshoukuandan | 负责实现制定收款单界面所需要的服务 |
| ZhiDingFuKuanDan | 负责实现制定付款单界面所需要的服务 |

（3）模块内部类的接口规范

Accountbl的接口规范如表11所示。

**表11 accountbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Zhanghuguanli.NameChange | 语法 | public void NameChange(String a) ; |
| 前置条件 | 财务人员进行更改账户名称操作 |
| 后置条件 | 根据账户名称确定是否存在账户，并根据反馈进行更改操作 |
| Zhanghuguanli.NameDelete | 语法 | public void NameDelete(String a) ; |
| 前置条件 | 财务人员进行删除账户操作 |
| 后置条件 | 根据账户名称确定是否存在账户，若存在则删除该用户相关内容 |
| Zhanghuguanli. NameAdd | 语法 | public void NameAdd(String a) ; |
| 前置条件 | 财务人员进行增加账户操作 |
| 后置条件 | 根据账户名称确定是否存在账户，若不存在则增加用户 |
| Zhanghuguanli.NameCheck | 语法 | public void NameCheck(String a) ; |
| 前置条件 | 财务人员进行查看账户操作 |
| 后置条件 | 根据账户名称确定账户，并显示账户名称，余额等内容 |
| Zhanghuguanli. JinEchange | 语法 | public void JinEchange(float a) ; |
| 前置条件 | 财务人员进行更改金额操作 |
| 后置条件 | 根据账户名称确定金额数目，并根据更改数据进行更改金额 |
| Zhanghuguanli. YuEchange | 语法 | public void YuEchange(float a) ; |
| 前置条件 | 付款单和收款单导入分析成功 |
| 后置条件 | 根据账户名称确定账户，系统计算并显示该账户余额 |
| Zhidingshoukuandan. GetS | 语法 | Public void GetS(String Num,String keHu,String Oprater,String zhuanZhangLieBiao,float jinE); |
| 前置条件 | 财务人员必须已经被识别和授权,并进行制定操作 |
| 后置条件 | 收款单通过审批后更改客户的应付数据，同时财务人员会收到一条消息，根据该信息财务人员完成现实银行账户操作。 |
| ZhiDingFuKuanDan. GetF | 语法 | public void GetF(String Num,String zhangHu,String Oprater,String tiaoMuQingDan,float jinE); |
| 前置条件 | 财务人员必须已经被识别和授权,并进行制定操作 |
| 后置条件 | 付款单通过审批后更改客户的应收数据，同时财务人员会收到一条消息，根据该信息财务人员完成现实银行账户操作。 |

（4）业务逻辑层的动态模型

图41-44为财务人员进行账户管理时，account业务逻辑处理相关对象协作的顺序图。

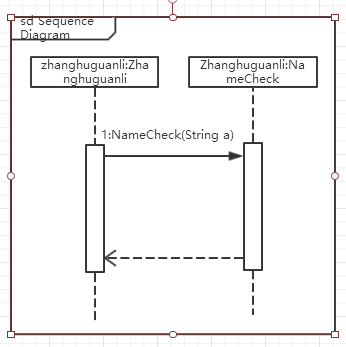


图41 账户查看的顺序图

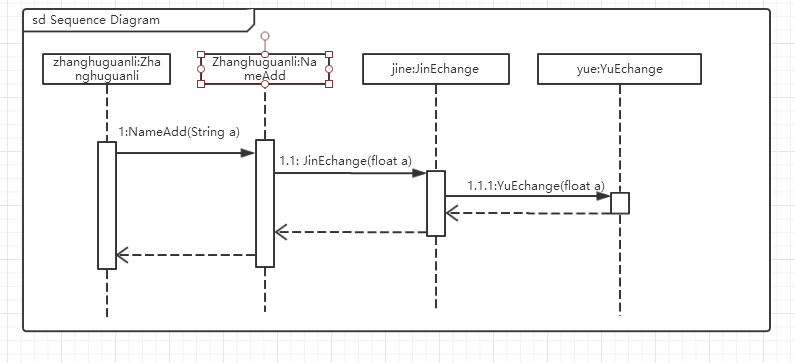


图42 账户增加的顺序图

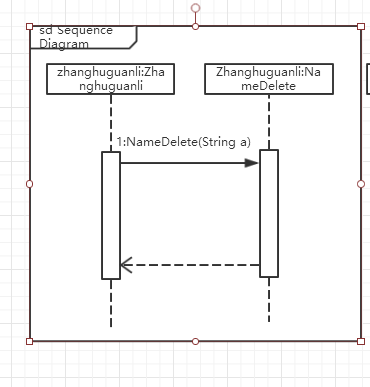


图43 账户删除的顺序图

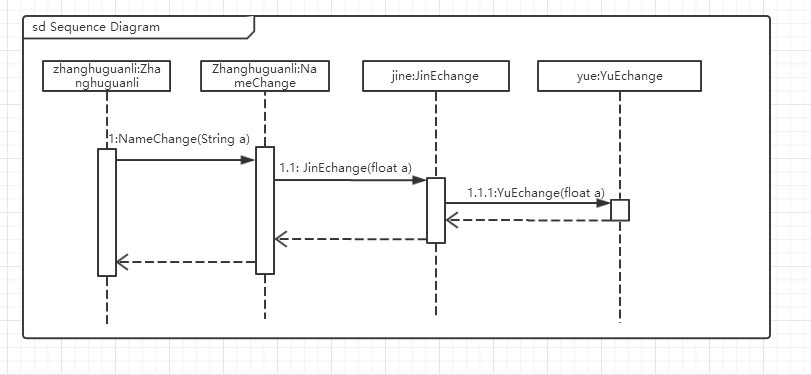


图44 账户修改的顺序图

图45为财务人员制定收款单时，account业务逻辑处理相关对象协作的顺序图。

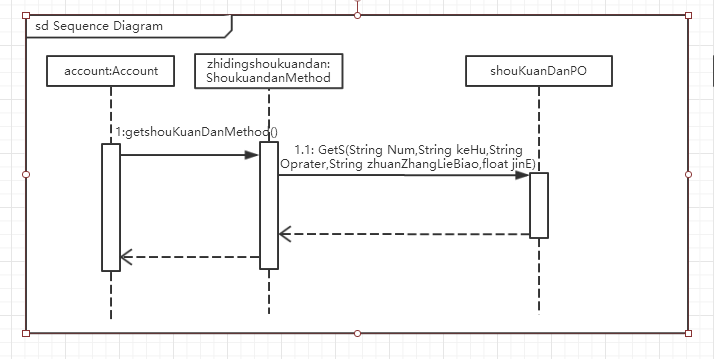


图45制定收款单的顺序图

图46为财务人员制定付款单时，account业务逻辑处理相关对象协作的顺序图。

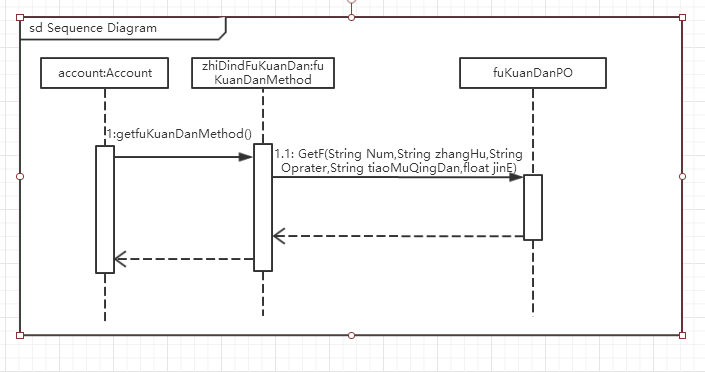


图46 制定付款单的顺序图

如图47、48、49所示的状态图描述了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。

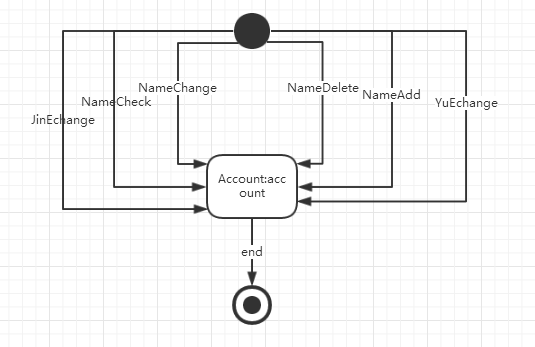


图47 账户管理期间的状态图

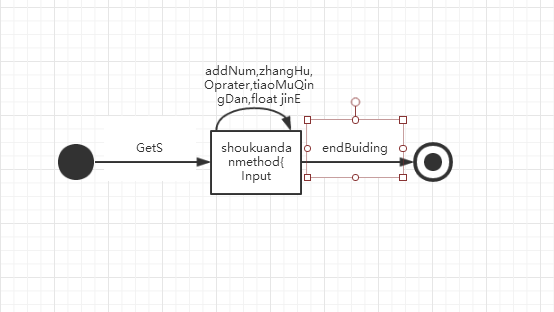


图48 制定收款单期间的状态图

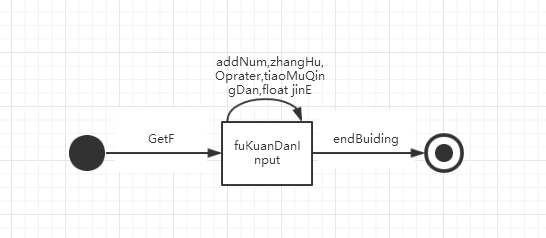


图49 制定付款单期间的状态图

（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将委托给相应的领域对象。

其他略。

### 4.1.6 docbl模块

（1）模块概述

docbl模块承担的需求参见需求规格文档说明文档功能需求及相关非功能需求。

Docbl模块的职责及接口参见软件系统结构描述文档对应表。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间通过添加接口增加灵活性。比如展示层和业务逻辑层之间，我们添加businesslogicservice.DocBLService接口。业务逻辑层和数据层之间，我们增加了DatabaseFactory.DocDatabase接口。DocPO是作为用户信息的持久化对象被添加到设计模型中去的。DocDatabase封装了关于Doc的数据集合的数据结构的秘密和信息管理的职责。

Doc模块的设计如图50所示。

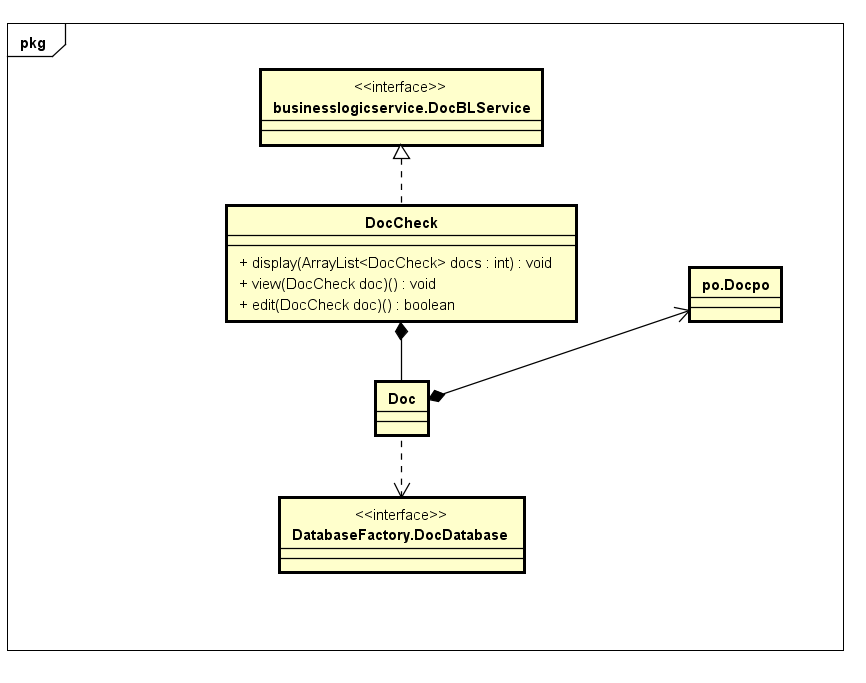


图50 docbl模块各个类的设计

Docbl模块各个类的职责如表11所示。

表11 docbl模块各个类的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| DocCheck | 负责实现单据审批界面所需要的服务 |
| Doc | 系统单据的领域模型对象，拥有单据类型及单据详细信息 |

（3）模块内部类的接口规范

Doc的接口规范如表12所示。

**表12 docbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| DocCheck.display | 语法 | | public void display(ArrayList<DocCheck> docs); |
| 前置条件 | | 总经理成功登陆系统 |
| 后置条件 | | 列表显示待审批的单据 |
| DocCheck. view | 语法 | | public void view(DocCheck doc); |
| 前置条件 | | 总经理制定某一待审批单据 |
| 后置条件 | | 显示指定单据的详细信息 |
| DocCheck. edit | 语法 | | public boolean edit(DocCheck doc); |
| 前置条件 | | 界面显示着某一单据的详细信息 |
| 后置条件 | | 若修改操作正常进行则根据总经理的修改对相应单据信息进行修改，否则返回false |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| DatabaseFactory.getDocDatabase | | 得到Doc数据库的服务的引用 | |
| UserDataService.check(DocPO po) | | 在数据库中查找DocPO对象 | |
| UserDataService.edit(DocPO po) | | 在数据库中修改引用的DocPO对象 | |

（4）业务逻辑层的动态模型

图51为总经理审批单据时，业务逻辑层处理相关对象协作的顺序图。

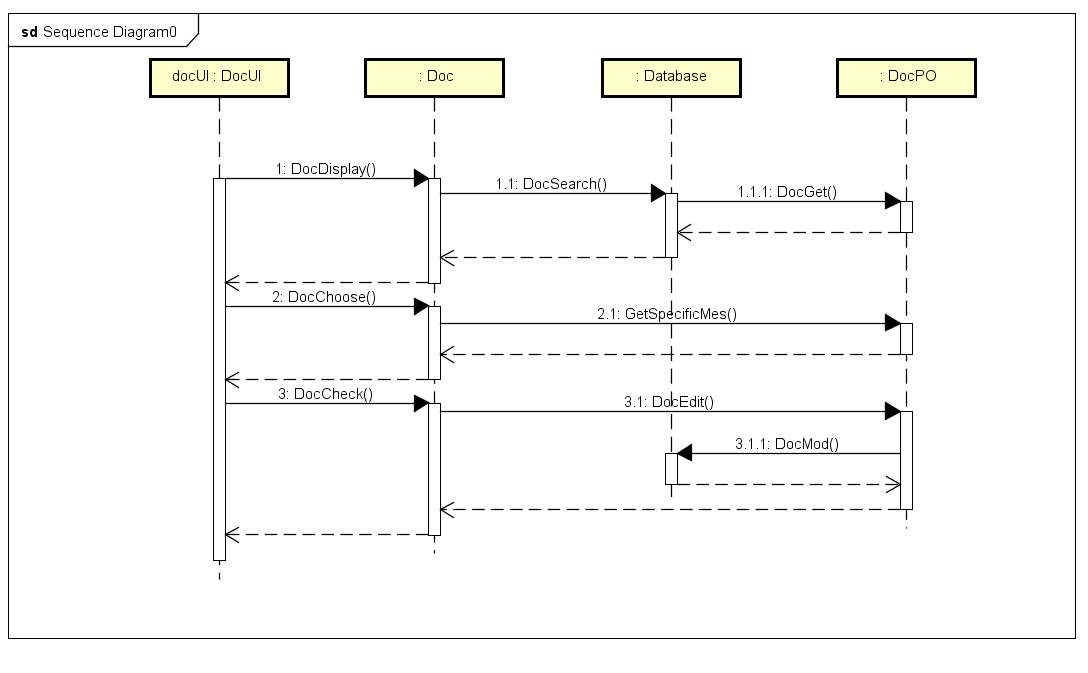


图51 审批单据的顺序图

如图52所示的状态图描述了Doc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check方法被调用，引入待审批的Doc，之后通过Check分为failed和permit两种状态，failed的Doc可通过Modify进入Permit状态，单据通过审批进入Fin状态后，系统根据单据信息修改数据库并向对应用户发送信息。

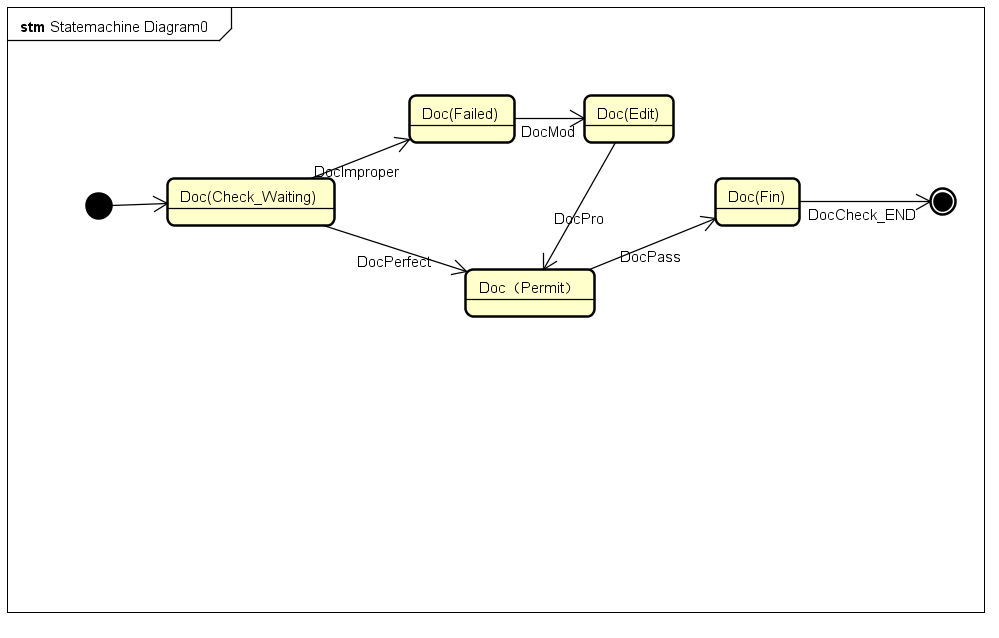


图52 Doc对象的状态图

（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将委托给相应的领域对象。

其他略。

### 4.1.7 promotionbl模块

（1）模块概述

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

Promotionbl模块的职责及接口参见软件系统结构描述文档对应表。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间通过添加接口增加灵活性。比如展示层和业务逻辑层之间，我们添加businesslogicservice.PromotionBLService接口。业务逻辑层和数据层之间，我们增加了DatabaseFactory.PromotionDatabase接口。PromotionPO是作为促销策略信息的持久化对象被添加到设计模型中去的。PromotionDatabase封装了关于Promotion的数据集合的数据结构的秘密和信息管理的职责。

Promotion模块的设计如图53所示。

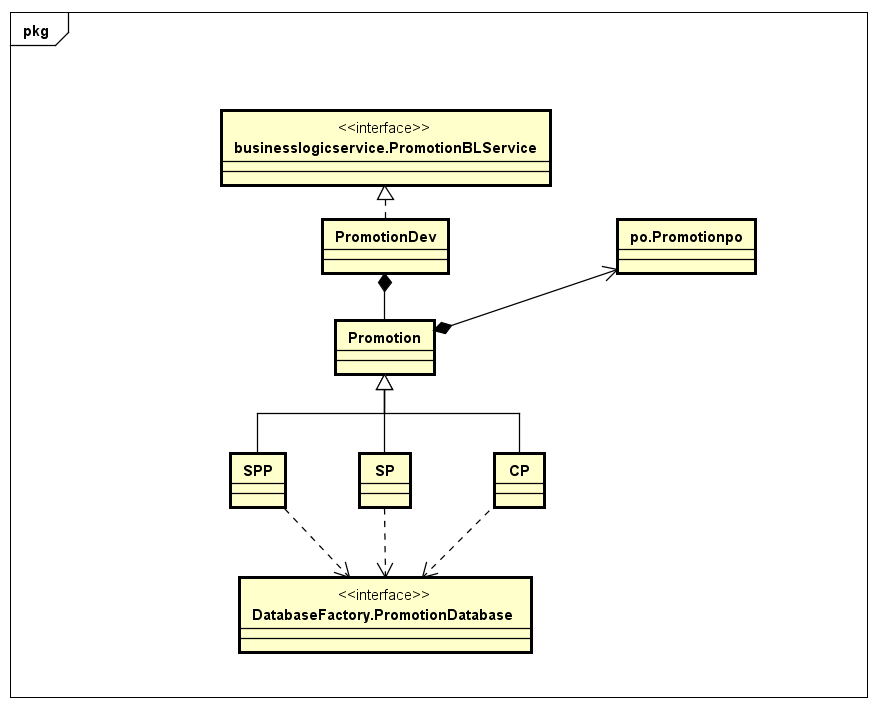


图53 promotionbl模块各个类的设计

Promotionbl模块各个类的职责如表13所示。

表13 promotionbl模块各个类的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| PromotionDev | 负责实现促销策略制定界面所需要的服务 |
| Promotion | 促销策略的领域模型对象，拥有促销策略类型及相应促销内容 |
| SPP | 商品特价包的领域模型对象 |
| SP | 总价促销策略的领域模型对象 |
| CP | 客户促销策略的领域模型对象 |

（3）模块内部类的接口规范

Promotion的接口规范如表14所示。

**表14 promotionbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| PromotionDev.SPP | 语法 | | public static boolean SPP\_Dev(ArrayList<Integer> goodsList,ArrayList<Double> SPList) |
| 前置条件 | | 总经理已被授权（成功登陆即授权） |
| 后置条件 | | 在与已有促销策略不冲突时，将新的商品特价包添加到PromotionDataBase，否则提示出错信息 |
| PromotionDev.SP | 语法 | | public static boolean SP\_Dev(double sum,ArrayList<Integer> giftsList,double voucher,double discount) |
| 前置条件 | | 总经理已被授权（成功登陆即授权） |
| 后置条件 | | 与已有促销策略不冲突时，将新的总价促销策略添加到PromotionDataBase，否则提示出错信息 |
| PromotionDev.CP | 语法 | | public static boolean CP\_Dev(int userLevel,ArrayList<Integer> giftsList,double voucher,double discount) |
| 前置条件 | | 总经理已被授权（成功登陆即授权） |
| 后置条件 | | 与已有促销策略不冲突时，将新的客户促销策略添加到PromotionDataBase，否则提示出错信息 |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| DatabaseFactory.getPromotionDatabase | | 得到Promotion数据库的服务的引用 | |
| PromotionDataService.check(PromotionPO po) | | 在数据库中查找PromotionPO对象 | |
| PromotionDataService.new(PromotionPO po) | | 在数据库中添加PromotionPO对象 | |

（4）业务逻辑层的动态模型

图54为总经理制定促销策略时，模块业务逻辑处理相关对象协作的顺序图。

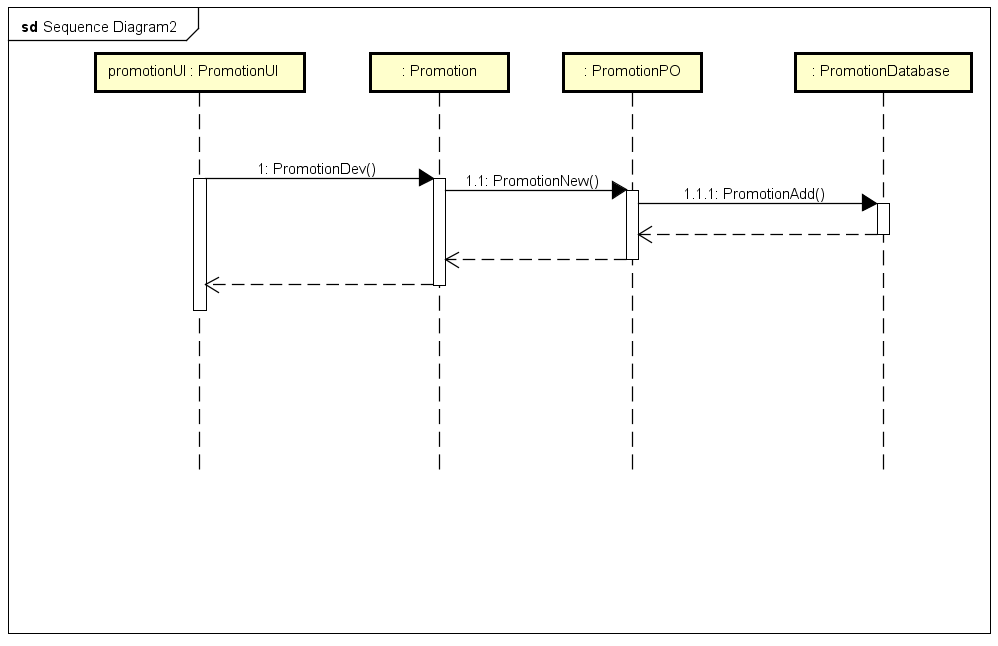


图54 制定促销策略的顺序图

如图55所示的状态图描述了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PromotionDev方法被调用，Promotion进入可编辑状态，编辑完成之后进行Check，Succ的话直接进入AfCheck状态，如与已有Promotion冲突，总经理可进行相应修改从而通过Check，最后将Promotion(AfCheck)加入PromotionDatabase，End流程PromotionDev。

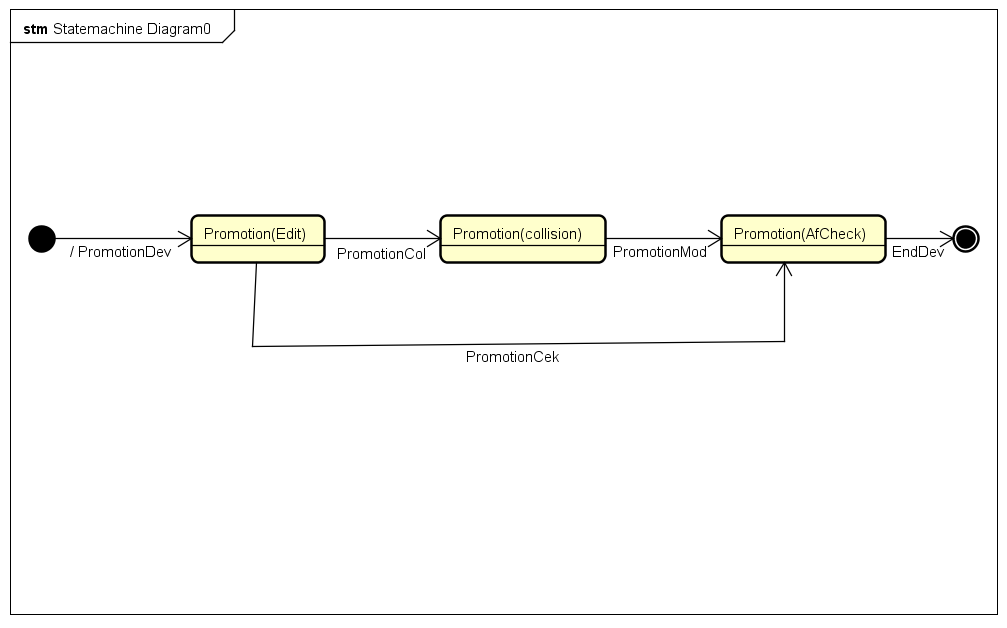


图55 Promotion对象的状态图

（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将委托给相应的领域对象。

其他略。

# 5.依赖视角

图56和图57是客户端和服务器端各自的包之间的依赖关系。

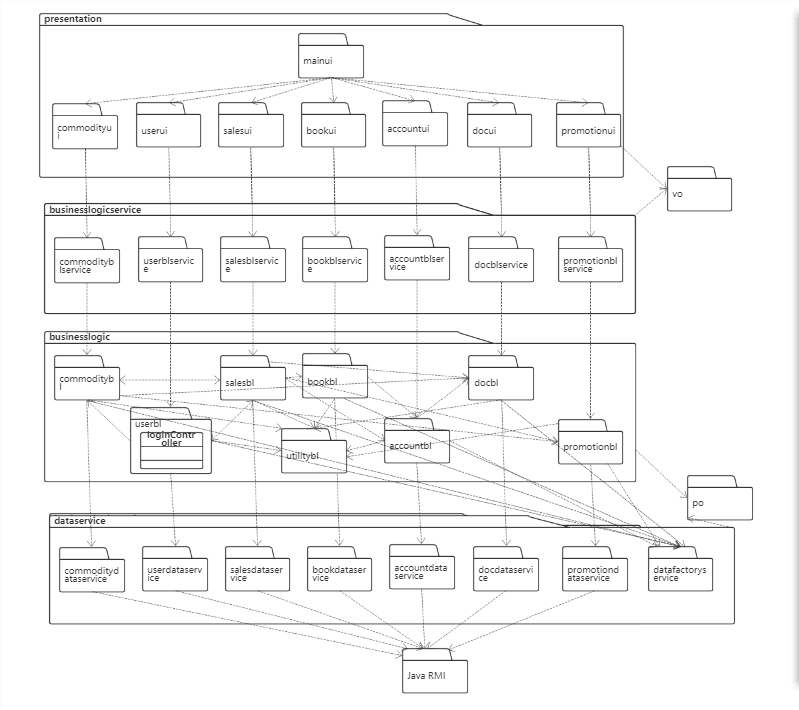


图56 客户端包图

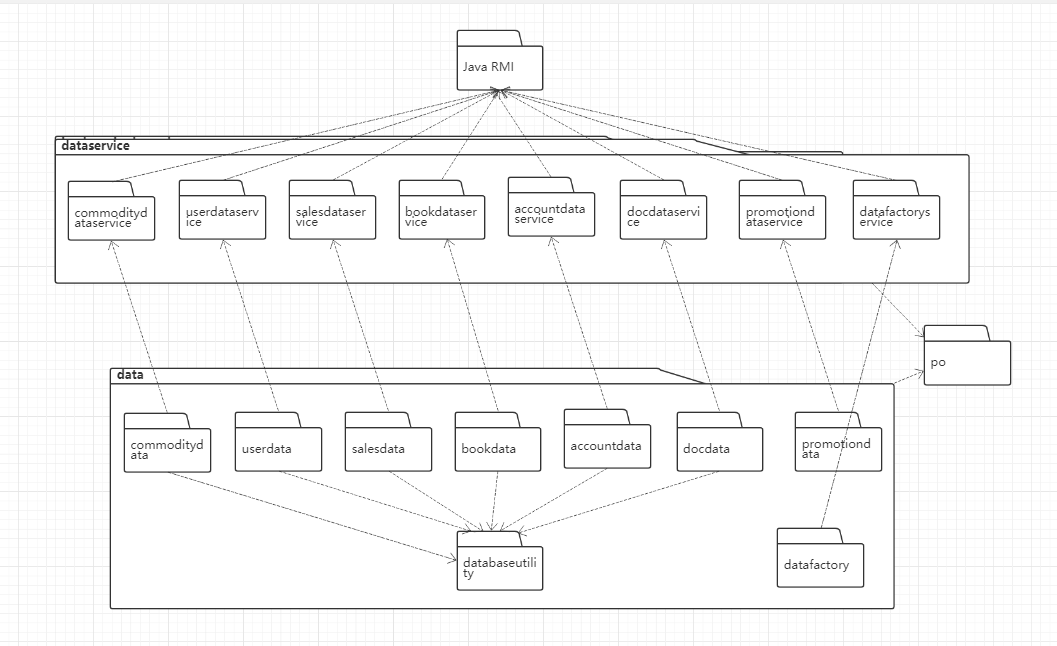


图57 服务器端包图