**进销存系统软件体系结构描述文档**

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概要设计，达到知道详细设计和开发的目的，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 | 词汇定义 | 备注 |
| 进销存系统 | 主要负责进货销售及库存的系统 | …… |
| …… | …… | …… |

**1.3参考资料**

**2.产品概述**

参考进销存系统管理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.逻辑视角**

进销存系统中，选择了分层体系结构风格，将系统分为3层(展示层，业务逻辑层，数据层)能够很好地示意整个高层抽象。展示层包含GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示。

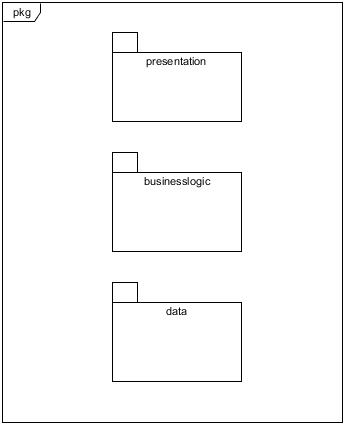


图1 参照体系结构风格的包图表达逻辑视角

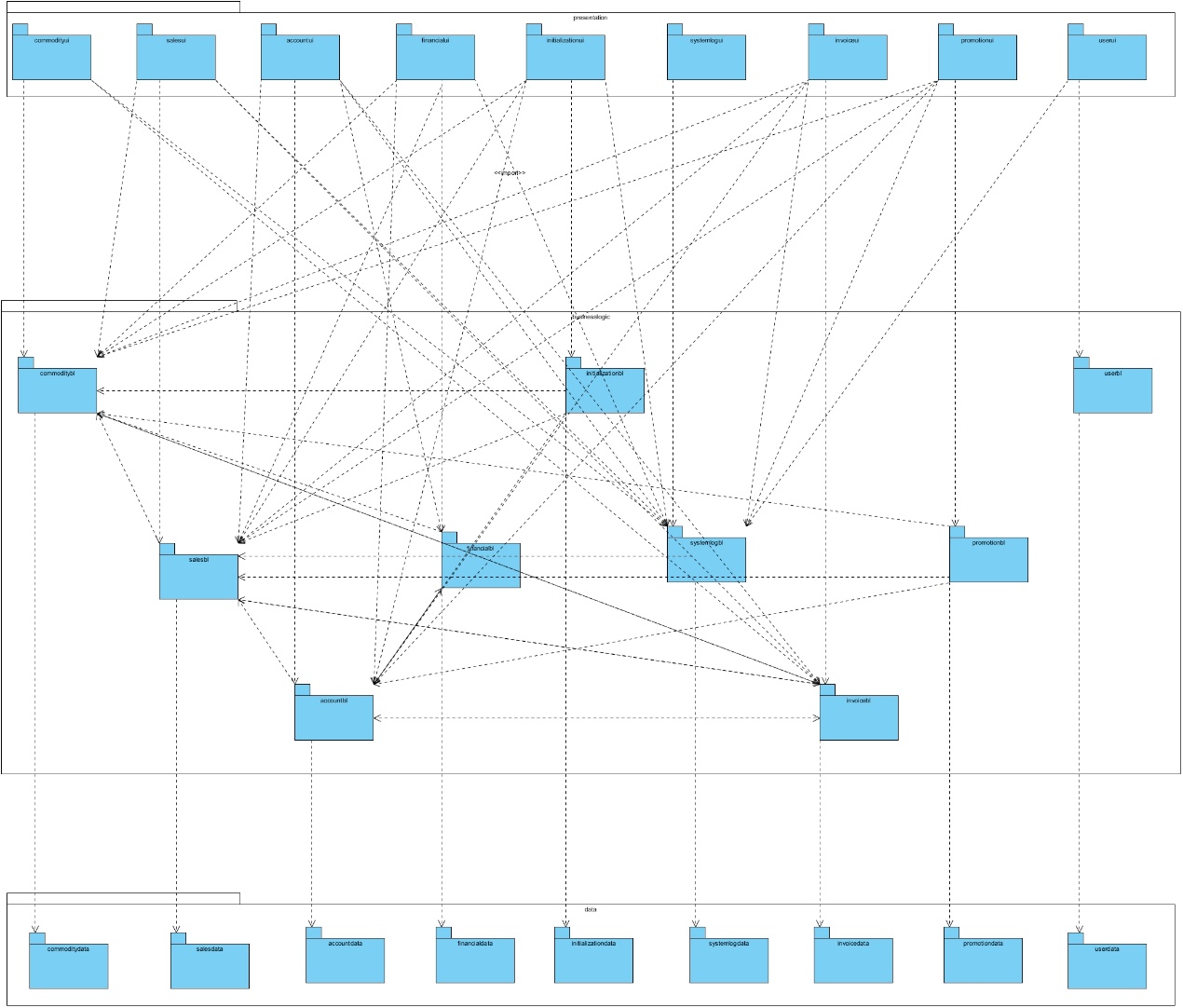


图2 软件体系结构逻辑设计方案

**4.组合视角**

**4.1开发包图**

进销存系统的最终开发包设计如表1所示。

表1 进销存系统的最终开发包设计

|  |  |
| --- | --- |
| **功能** | **对应逻辑包** |
| Mainui | Commodityui, Salesui, Accountui, Financialui, Initializaionui, Systemlogui, Invoiceui, Promotionui, Userui |
| Commodityui | Commodityblservice,界面类库包,vo |
| Commodityblservice |  |
| Commoditybl | Commodityblservice, Commoditydataservice, po, Invoicebl, Systemlogbl |
| Commoditydataservice | JavaRMI, po |
| Commoditydata | JavaRMI, po, databaseutility |
| Salesui | Salesblservice,界面类库包,vo |
| Salesblservice |  |
| Salesbl | Salesblservice, Salesdataservice, po, Commoditybl, Invoicebl, Systemlogbl |
| Salesdataservice | JavaRMI, po |
| Salesdata | JavaRMI, po, databaseutility |
| Accountui | Accountblservice,界面类库包,vo |
| Accountblservice |  |
| Accountbl | Accountblservice, Accountdataservice, po, Financialbl, Invoicebl, Systemlogbl, Salesbl |
| Accountdataservice | JavaRMI, po |
| Accountdata | JavaRMI, po, databaseutility |
| Financialui | Financialblservice,界面类库包,vo |
| Financialblservice |  |
| Financialbl | Financialblservice, Financialdataservice, po, Accountbl, Systemlogbl, Commoditybl, Salesbl, Salesbl |
| Financialdataservice | JavaRMI, po |
| Financialdata | JavaRMI, po, databaseutility |
| Initializationui | Initializationblservice,界面类库包,vo |
| Initializationblservice |  |
| Initializationbl | Initializationblservice, Initializationdataservice, po, Commoditybl, Accountbl, Systemlogbl, Salesbl |
| Initializationdataservice | JavaRMI, po |
| Initializationdata | JavaRMI, po, databaseutility |
| Systemlogui | Systemlogblservice,界面类库包,vo |
| Systemlogblservice |  |
| Systemlogbl | Systemlogblservice, Systemlogdataservice, po |
| Systemlogdataservice | JavaRMI, po |
| Systemlogdata | JavaRMI, po, databaseutility |
| Invoiceui | Invoiceblservice,界面类库包,vo |
| Invoiceblservice |  |
| Invoicebl | Invoiceblservice, Invoicedataservice, po, Commoditybl, Salesbl, Accountbl, Systemlogbl |
| Invoicedataservice | JavaRMI, po |
| Invoicedata | JavaRMI, po, databaseutility |
| Promotionui | Promotionblservice,界面类库包,vo |
| Promotionblservice |  |
| Promotionbl | Promotionblservice, Promotiondataservice, po, Commoditybl, Salesbl, Systemlogbl |
| Promotiondataservice | JavaRMI, po |
| Promotiondata | JavaRMI, po, databaseutility |
| Userui | Userblservice,界面类库包,vo |
| Userblservice |  |
| Userbl | Userblservice, Userdataservice, po, Systemlogbl |
| Userdataservice | JavaRMI, po |
| Userdata | JavaRMI, po, databaseutility |
| Vo |  |
| Po |  |
| Utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| Databaseutility | JDBC |

进销存系统客户端开发包如图3所示，服务器端开发包图如图4所示。

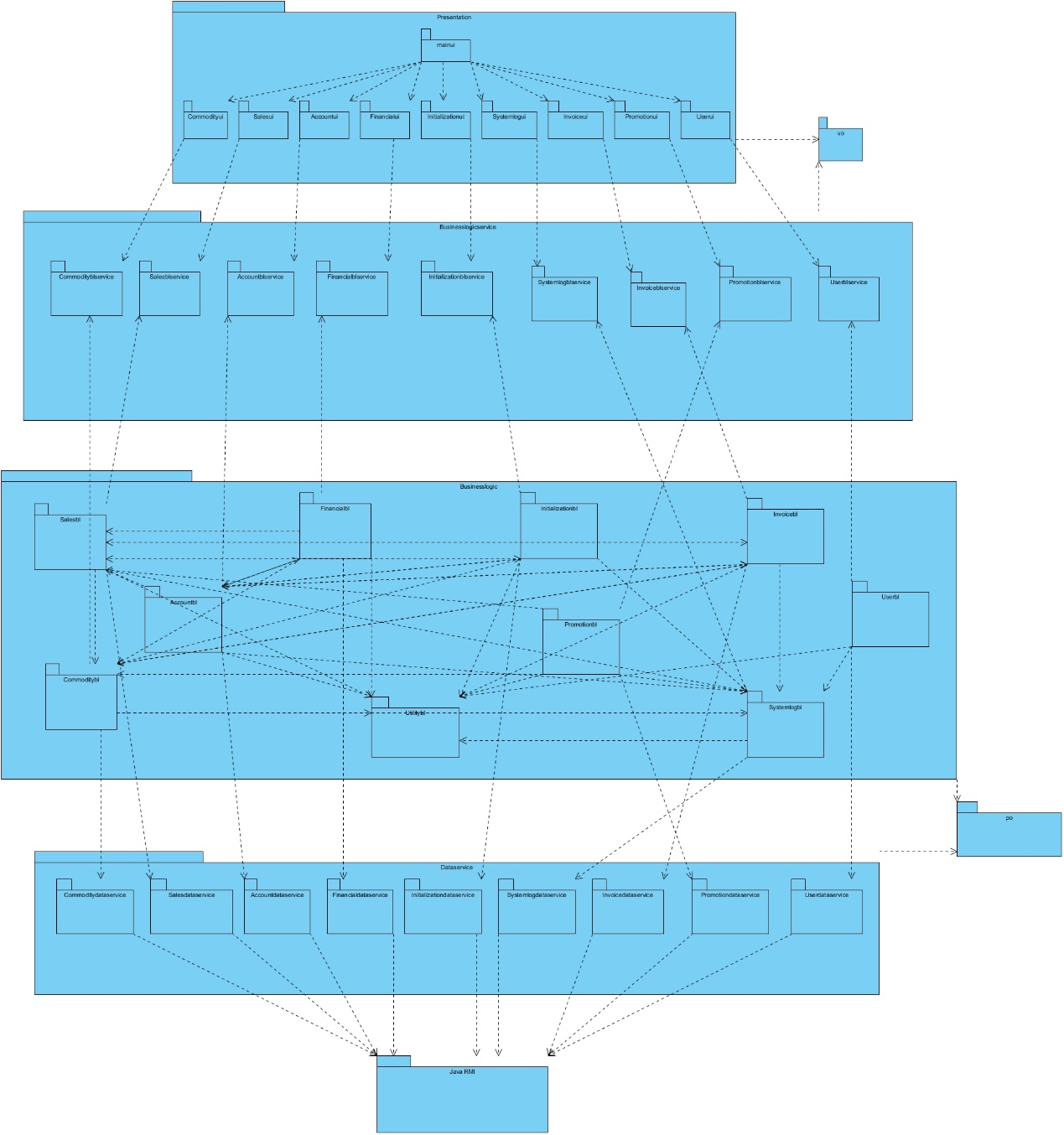


图3 进销存系统客户端开发包图

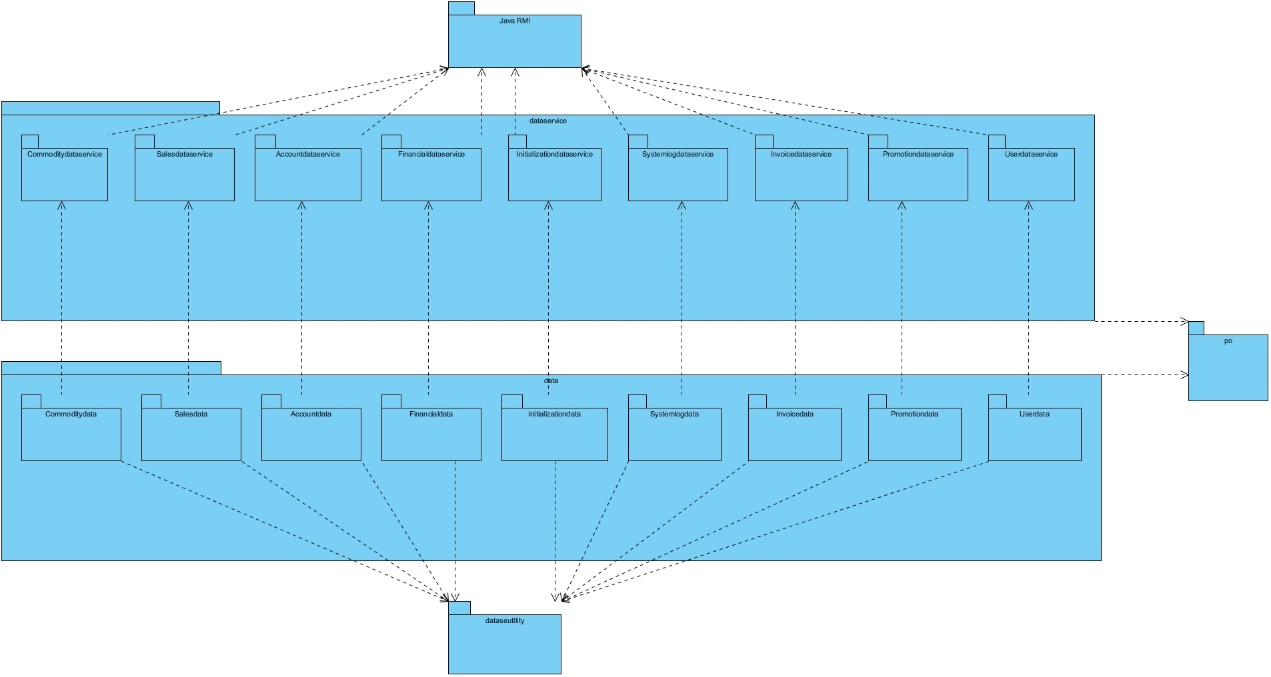


图4 进销存系统服务器端开发包图

**4.2运行时进程**

在进销存系统中，会有多个客户端进程和一个服务器端进程，其进程图如图5所示。结合部署图，客户端进程是在客户端机器上运行，服务器端进程在服务器端机器上运行。

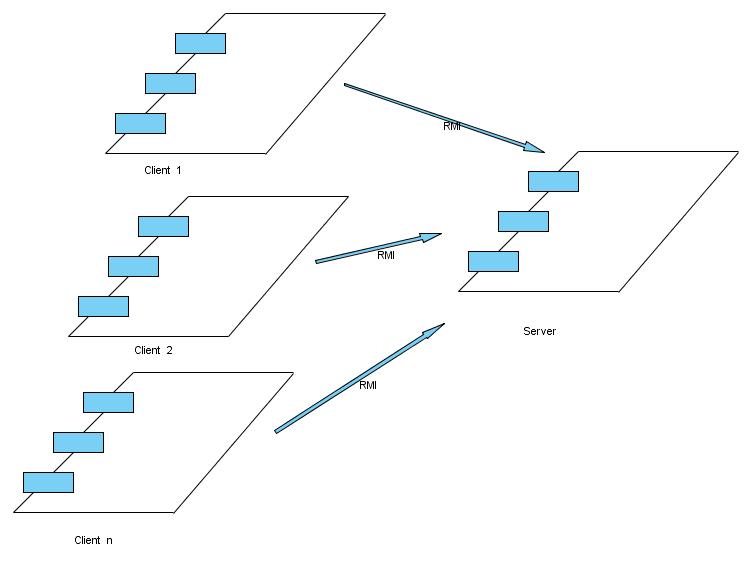


图5 进程图

**4.3物理部署**

进销存系统中客户端构建是放在客户端机器上，服务器端构件是放在服务器端机器上。在客户端节点上，还要部署RMIStub构件。由于Java RMI构件属于JDK6.0的一部分。所以，在系统JDK环境已经设置好的情况下，不需要再独立部署。部署图如图6所示。

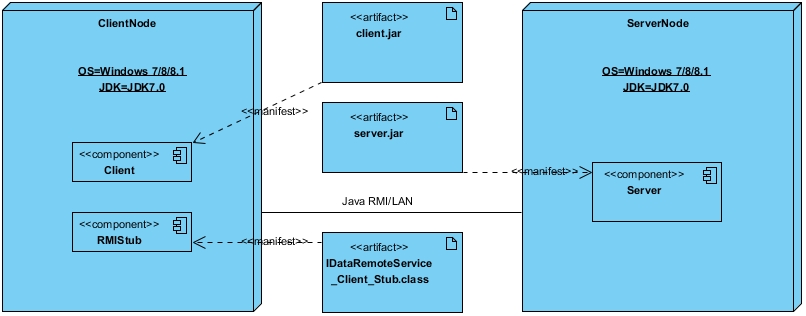


图6 部署图

**5.接口视角**

**5.1模块的指责**

客户端模块和服务器端模块试图分别如图7和图8所示。客户端各层和服务器端各层的职责分别如表2和表3所示。

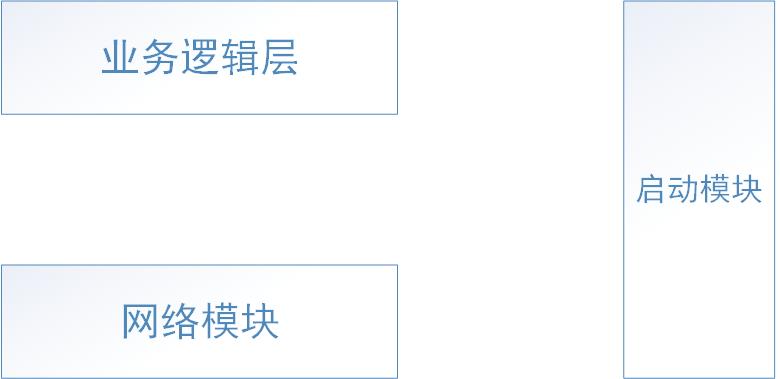


图7 客户端模块视图

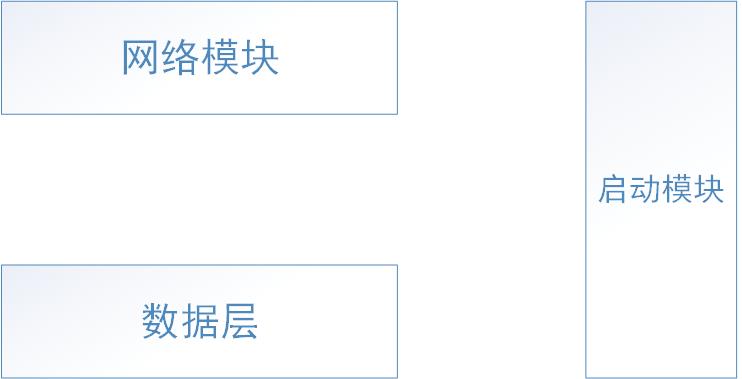


图8 服务器端模块视图

表2 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 | 基于窗口的连锁商店客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行相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3 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 |
| 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。层之间调用的接口如表4所示。

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| CommodityBlService  InvoidceBlService  SalesBlService  AccountBlService  CustomerBlService  FinancialBlService  SystemlogBlService  InitializationBlService  PromotionBlService  UserBlService | 客户端展示层 | 客户端业务逻辑层 |
| CommodityDataService  SalesDataService  AccountDataService  FinancialDataService  InitializationDataService  PromotionDataService  InvoiceDataServoice  SystemlogDataService  UserDataService | 客户端业务逻辑层 | 服务器端数据层 |

借用销售用例来说明层之间的调用，如图9所示。每一层之间都是由上层依赖了一个借口（需接口），而下层实现这个接口（供接口）SalesBLService提供了Sales界面所需要的所有业务逻辑功能。SalesDataService提供了对数据库的增，删，改，查等操作。这样的实现就大大降低了层与层之间的耦合。

图9 销售用例层之间调用的接口

**5.2 用户界面层的分解**

根据需求，系统存在 个用户界面。界面跳转如图10所示。

图10 用户界面跳转

服务器端和客户端的用户设计接口是一致的，只是具体的界面不一样。用户界面类如图11所示。



图11 用户界面类

**5.2.1用户界面层模块的职责**

如表5所示为用户界面层的职责

表5 用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| ShowPanel | 界面Frame,负责界面的显示和界面的跳转 |
| HidePanel | 负责隐藏界面 |

**5.2.2用户界面层模块的接口规范**

用户界面层模块的接口规范如表6所示。

表6 用户界面层模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| ShowPanel | 语法 | Show(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame与Panel |
| HidePanel | 语法 | Hide(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 隐藏Frame与Panel |

用户界面层需要的服务接口如表7所示。

表7 用户界面层模块需要的服务接口

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| ShowPanel | 语法 | Show(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame与Panel |
| HidePanel | 语法 | Hide(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 隐藏Frame与Panel |
| 需要的服务 | | |
| 服务名 | 服务 | |
| Businesslogicservice.LoginBLService | 登陆界面的业务逻辑接口 | |
| Businesslogicservice.CommodityBLService | 库存界面的业务逻辑接口 | |
| Businesslogcservice.SalesBLService | 销售界面的业务逻辑接口 | |
| Businesslogcservice.AccountBLService | 账户管理界面的业务逻辑接口 | |
| Businesslogcservice.FinancialBLService | 销售表单管理界面的业务逻辑接口 | |
| Businesslogcservice.InitializationBLService | 期初建账界面的业务逻辑接口 | |
| Businesslogcservice.SystemlogBLService | 系统日志界面的业务逻辑接口 | |
| Businesslogcservice.InvoiceBLService | 单据审批界面的业务逻辑接口 | |
| Businesslogcservice.PromotionBLService | 促销界面的业务逻辑接口 | |
| Businesslogcservice.UserBLService | 用户管理界面的业务逻辑接口 | |

**5.2.3 用户界面模块设计原理**

用户界面利用Java的Swing和AWT库来实现。

**5.3业务逻辑层的分解**

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，User对象负责处理登陆界面的业务逻辑；Sales对象负责销售界面的业务逻辑。业务逻辑层的设计如图12所示。

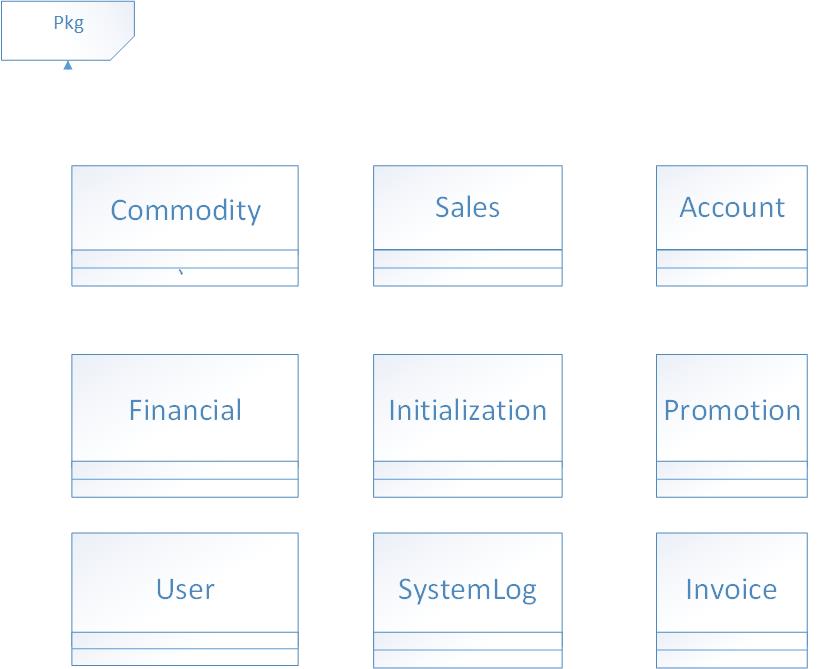


图12 业务逻辑层的设计

**5.3.1业务逻辑层模块的职责**

业务逻辑层模块的职责如表8所示。

表8 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| Commoditybl | 负责实现库存界面所需要的服务 |
| Salesbl | 负责实现销售界面所需要的服务 |
| Accountbl | 负责实现账户界面所需要的服务 |
| Financialbl | 负责实现财务界面所需要的服务 |
| Initializationbl | 负责实现建账初期界面所需要的服务 |
| Promotionbl | 负责实现促销策略界面所需要的服务 |
| Userbl | 负责实现与登陆界面所需要的服务 |
| SystemLogbl | 负责实现系统日志界面所需要的服务 |
| Invoicebl | 负责实现单据审批界面所需要的服务 |

**5.3.2业务逻辑层模块的接口规范**

接口规范如下图所示：

Commoditybl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Commodity.addCommodity | 语法 | Public ResultMessage addCommodity(String name, String type)  Public ResultMessage addCommodity(String name, String type, int in\_price, int out\_price) |
| 前置条件 | 输入的in\_price和out\_price为正整数 |
| 后置条件 | 添加商品，返回是否成功 |
| Commodity.delCommodity | 语法 | Public ResultMessage delCommodity(String name, String type) |
| 前置条件 | 无 |
| 后置条件 | 删除商品，返回是否成功 |
| Commodity.updateCommodity | 语法 | Public ResultMessage updateCommodity(String name, String type, int in\_price, int out\_price) |
| 前置条件 | 输入的in\_price和out\_price为正整数 |
| 后置条件 | 更新商品，返回是否更新成功 |
| Commodity.searchCommodity | 语法 | Public ResultMessage searchCommodity(String word) |
| 前置条件 | 无 |
| 后置条件 | 返回搜索结果 |
| Commodity.addSort | 语法 | Public ResultMessage addSort(String name) |
| 前置条件 | 无 |
| 后置条件 | 添加分类，返回是否添加成功 |
| Commodity.delSort | 语法 | Public ResultMessage delSort(String name) |
| 前置条件 | 无 |
| 后置条件 | 删除分类，返回是否删除成功 |
| Commodity.updateSort | 语法 | Public ResultMessage updateSort(String name1, String name2) |
| 前置条件 | 无 |
| 后置条件 | 修改分类，返回是否修改成功 |
| Commodity.Examine | 语法 | Public ResultMessage examine(String time1, String time2) |
| 前置条件 | 输入的时间格式正确 |
| 后置条件 | 返回查看结果 |
| Commodity.Inventory | 语法 | Public ResultMessage inventory() |
| 前置条件 | 无 |
| 后置条件 | 返回盘点结果 |
| Commodity.addGift | 语法 | Public ResultMessage addGift(String name, String type) |
| 前置条件 | 无 |
| 后置条件 | 增加赠品，返回是否成功 |
| Commodity.delGift | 语法 | Public ResultMessage delGift(String name, String type) |
| 前置条件 | 无 |
| 后置条件 | 删除赠品，返回是否成功 |
| Commodity.patch | 语法 | Public ResultMessage patch(String name, String type, int number) |
| 前置条件 | 输入的number为数字 |
| 后置条件 | 生成报溢报损单，返回是否成功 |
| Commodity.warn | 语法 | Public ResultMessage warn(String name, String type, int number) |
| 前置条件 | 输入的number为数字 |
| 后置条件 | 返回是否设置成功 |
| 需要的服务 | | |
| 服务名 | 服务 | |
| CommodityDataService.findGood(String name, String type) | 根据name和type查找商品对象 | |
| CommodityDataService.findName(String name) | 根据name查找商品对象 | |
| CommodityDataService.findType(String type) | 根据type查找商品对象 | |
| CommodityDataService.addGood(CommodityPO po1, SortPO po2) | 在分类中添加商品对象 | |
| CommodityDataService.delGood(CommodityPO po) | 删除商品对象 | |
| CommodityDataService.updateGood(CommodityPO po1, CommodityPO po2) | 更新商品对象信息 | |
| CommodityDataService.addSort(SortPO po) | 添加分类对象 | |
| CommodityDataService.delSort(SortPO po) | 删除分类对象 | |
| CommodityDataService.updateSort(SortPO po1, SortPO po2) | 更新分类对象 | |
| CommodityDataService.addGift(CommodityPO po) | 添加赠品对象 | |
| CommodityDataService.findGift(String name, String type) | 查找赠品对象 | |
| CommodityDataService.delGift(CommodityPO po) | 删除赠品对象 | |
| CommodityDataService.getAll() | 获得所有商品对象 | |
| CommodityDataService.clear() | 清空所有商品与分类对象 | |

Salesbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Sales.addCustomer | 语法 | Public ResultMessage addCustomer(String name, int level, String phone, String zip, String mail, int money, Stirng clerk) |
| 前置条件 | 输入的phone,zip和money合法 |
| 后置条件 | 返回提示是否成功添加客户 |
| Sales.delCustomer | 语法 | Public ResultMessage delCustomer(String name) |
| 前置条件 | 无 |
| 后置条件 | 返回提示是否成功删除客户 |
| Sales.updateCustomer | 语法 | Public ResultMessage updateCustomer(String name, int level, String phone, String zip, String mail, int money, Stirng clerk) |
| 前置条件 | 有客户的信息发生改变，需要更新 |
| 后置条件 | 返回提示是否成功更新客户信息 |
| Sales.searchCustomer | 语法 | Public ResultMessage searchCustomer(String word) |
| 前置条件 | 无 |
| 后置条件 | 模糊搜索，返回搜索结果 |
| Sales.addImport | 语法 | Public ResultMessage addImport(String name, int warehouse, String good\_name, String good\_type, String ps, int number) |
| 前置条件 | 发生进货时，输入的warehouse和number合法 |
| 后置条件 | 完成进货单后交由总经理审批 |
| Sales.addImport\_Return | 语法 | Public ResultMessage addImport\_Return(String name, int warehouse, String good\_name, String good\_type, String ps, int number)  Public ResultMessage addImport\_Return(String note, int number) |
| 前置条件 | 发生进货退货时，输入的warehouse和number合法 |
| 后置条件 | 完成进货退货后，交由总经理审批 |
| Sales.addExport | 语法 | Public ResultMessage addExport(String name, int warehouse, String good\_name, String good\_type, String ps, int number, int price, int discount) |
| 前置条件 | 发生销售时，输入的warehouse、number、price和discount合法 |
| 后置条件 | 完成销售单，交由总经理审批 |
| Sales.addExport\_Return | 语法 | Public ResultMessage addExport\_Return(String name, int warehouse, String good\_name, String good\_type, String ps, int number, int price, int discount)  Public ResultMessage addExport\_Return(String note, int number) |
| 前置条件 | 发生销售时，输入的warehouse、number、price和discount合法 |
| 后置条件 | 完成销售退货，交由总经理审批 |
| 需要的服务 | | |
| SalesDataService.addCustomer(CustomerPO po) | 添加客户CustomerPO对象 | |
| SalesDataService.delCustomer(CustomerPO po) | 删除客户CustomerPO对象 | |
| SalesDataService.updateCustomer(CustomerPO po1, CustomerPO po2) | 更新客户CustomerPO对象信息 | |
| SalesDataService.findCustomer(String name) | 根据name查找客户Customer对象 | |
| SalesDataService.addImport(ImportPO po) | 添加进货单ImportPO对象 | |
| SalesDataService.addImport\_Return(Import\_ReturnPO po) | 添加进货退货单Import\_ReturnPO对象 | |
| SalesDataService.addExport(ExportPO po) | 添加销售单ExportPO对象 | |
| SalesDataService.addExport\_Return(Export\_ReturnPO po) | 添加销售退货单Export\_ReturnPO对象 | |
| SalesDataService.getAllCustomer() | 获得所有的客户对象 | |
| SalesDataService.getAllImport() | 获得所有的进货单对象 | |
| SalesDataService.getAllImport\_Return() | 获得所有的进货退货单对象 | |
| SalesDataService.getAllExport() | 获得所有的销售单对象 | |
| SalesDataService.getAllExport\_Return() | 获得所有的销售退货单对象 | |
| SalesDataService.clear() | 清除所有对象 | |

Accountbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Account.checkAccount() | 语法 | Public ResultMessage checkAccount() |
| 前置条件 | 管理人员为总经理或财务经理 |
| 后置条件 | 返回账户信息 |
| Account.addAccount | 语法 | Public ResultMessage addAccount(String name, int money) |
| 前置条件 | 需要添加账户时，管理人员为总经理或财务经理 |
| 后置条件 | 返回是否添加成功 |
| Account.delAccount | 语法 | Public ResultMessage delAccount(String name) |
| 前置条件 | 需要删除账户时，管理人员为总经理或财务经理 |
| 后置条件 | 返回是否删除账户成功 |
| Account.updateAccount | 语法 | Public ResultMessage updateAccount(String name1, String name2) |
| 前置条件 | 需要更新账户信息时，管理人员为总经理或财务经理 |
| 后置条件 | 完成账户信息修改，返回是否成功修改 |
| Account.searchAccount | 语法 | Public ResultMessage searchAccount(String word) |
| 前置条件 | 操作员是总经理或财务经理 |
| 后置条件 | 返回查找结果 |
| Account.addReceipt | 语法 | Public ResultMessage addReceipt(String customer\_name, String clerk, String account\_name, int money, int all\_money, String ps) |
| 前置条件 | 收到货款时，输入的money合法 |
| 后置条件 | 完成收款单，交由总经理审批 |
| Account.addPayment | 语法 | Public ResultMessage addPayment(String customer\_name, String clerk, String account\_name, int money, int all\_money, String ps) |
| 前置条件 | 完成付款时，输入的money合法 |
| 后置条件 | 完成付款单，交由总经理审批 |
| 需要的服务 | | |
| AccountDataService.getAccount() | | 获得当前账户信息 |
| AccountDataService.addAccount(AccountPO po) | | 添加账户AccountPO对象 |
| AccountDataService.delAccount(AccountPO po) | | 删除账户AccountPO对象 |
| AccountDataService.updateAccount(AccountPO po1, AccountPO po2) | | 更新账户AccountPO对象 |
| AccountDataService.findAccount(String name) | | 根据name查找账户对象 |
| AccountDataService.getAllAccount() | | 获得所有的账户对象 |
| AccountDataService.addReceipt(ReceiptPO po) | | 添加收款单ReceiptPO对象 |
| AccountDataService.addPayment(PaymentPO po) | | 添加付款单PaymentPO对象 |
| AccountDataService.getAllReceipt() | | 获得所有的收款单对象 |
| AccountDataService.getAllPayment() | | 获得所有的付款单对象 |
| AccountDataService.clear | | 清除所有对象 |

Financialbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Financial.saleList | 语法 | Public ResultMessage saleList(String time1, String time2, String good\_name, String good\_type, String customer\_name, String clerk, int warehouse) |
| 前置条件 | 操作员是经过验证的财务人员 |
| 后置条件 | 查询销售明细，返回操作是否成功 |
| Financial.allBill | 语法 | Public ResultMessage allBill(String time1, String time2, String note\_type, String customer\_name, String clerk, int warehouse) |
| 前置条件 | 操作员是经过验证的财务人员 |
| 后置条件 | 查询单据，返回操作是否成功 |
| Financial.operatingCondition | 语法 | Public ResultMessage operatingCondition(String time1, String time2) |
| 前置条件 | 操作员是经过验证的财务人员 |
| 后置条件 | 查看经营情况，返回操作是否成功 |
| 需要的服务 | | |
| FinancialDataService.getSaleList() | 获得所有销售明细对象 | |
| FinancialDataService.getAllBill() | 获得所有经营历程对象 | |
| FinancialDataService.getOperatingCondition() | 获得所有经营情况对象 | |
| FinancialDataService.addSaleList(SaleListPO po) | 添加销售明细的单据对象 | |
| FinancialDataService.addAllBill(AllBillPO po) | 添加经营历程的单据对象 | |
| FinancialDataService.addOperatingCondition(OperatingConditionPO po) | 添加经营情况的单据对象 | |
| FinancialDataService.clear() | 清空所有对象 | |

Initializationbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Initialization.newSystem | 语法 | Public ResultMessage newSystem() |
| 前置条件 | 操作人员输入完成期初建账的信息，包括商品，客户，账户 |
| 后置条件 | 旧系统内容将被删除，新系统替换旧系统 |
| Initialization.addCommodity | 语法 | Public ResultMessage addCommodity(String name, String type, int in\_price, int out\_price) |
| 前置条件 | 操作人员为财务人员 |
| 后置条件 | 记录此商品信息，等待替换库存内商品 |
| Initialization.addCustomer | 语法 | Public ResultMessage addCustomer(String name, String phone) |
| 前置条件 | 操作人员为财务人员 |
| 后置条件 | 记录此客户信息，等待替换客户信息 |
| Initialization.addAccount | 语法 | Public ResultMessage addAccount(String name, int money) |
| 前置条件 | 操作人员为财务人员 |
| 后置条件 | 记录此账户信息，等待替换账户信息 |
| Initialization.showInformation | 语法 | Public ResultMessage showInformation() |
| 前置条件 | 操作人员为财务人员 |
| 后置条件 | 返回期初信息 |
| 需要的服务 | | |
| InitializationDataService.getInformation | 查看InitializationPO对象的信息 | |

Promotionbl的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务 | | | |
| Promotion.makeDiscount | 语法 | | Public ResultMessage makeDiscount(int start\_money, int discount, int end\_money, String time1, String time2) |
| 前置条件 | | 操作用户为总经理 |
| 后置条件 | | 制作好折价优惠之后，立即更新至销售 |
| Promotion.delPromotion | 语法 | | Public ResultMessage delPromotion(String note) |
| 前置条件 | | 存在已经制定好的销售策略，操作用户为总经理 |
| 后置条件 | | 取消优惠后，立即更新至销售 |
| Promotion.makeGift | 语法 | | Public ResultMessage makeGift(start\_money, String time1, String time2) |
| 前置条件 | | 操作用户为总经理 |
| 后置条件 | | 制作好赠品优惠之后，立即更新至销售 |
| Promotion.search() | 语法 | | Public ResultMessage search() |
| 前置条件 | | 操作用户为总经理 |
| 后置条件 | | 返回显示的促销策略信息 |
| Promotion.makeOther | 语法 | | Public ResultMessage makeOther(String word) |
| 前置条件 | | 操作用户为总经理 |
| 后置条件 | | 制作好其他优惠之后，立即更新至销售 |
| 需要的服务 | | | |
| PromotionDataService.addDiscount(DiscountPO po) | | 添加折价促销DiscountPO方案对象 | |
| PromotionDataService.addGift(ProGiftPO po) | | 添加赠品促销ProGiftPO方案对象 | |
| PromotionDataService.addRecord() | | 添加促销记录 | |
| PromotionDataService.delDiscount(DiscountPO po) | | 删除折价促销DiscountPO方案对象 | |
| PromotionDataService.delGift(ProGiftPO po) | | 删除赠品促销ProGiftPO方案对象 | |
| PromotionDataservice.getDiscount() | | 获得当前折价促销方案对象 | |
| PromotionDataService.getGift() | | 获得当前赠品促销方案对象 | |
| PromotionDataService.show() | | 获得历史所有促销方案对象 | |

Invoicebl的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务 | | | |
| Invoice.show | 语法 | | Public ResultMessage show() |
| 前置条件 | | 操作人员为总经理 |
| 后置条件 | | 开始审核单据，返回单据信息 |
| Invoice.pass | 语法 | | Public ResultMessage pass(int note) |
| 前置条件 | | 操作人员为总经理，存在正在审批的单据 |
| 后置条件 | | 按照单据内容修改库存、客户、账户信息 |
| Invoice.refuse | 语法 | | Public ResultMessage refuse(int note) |
| 前置条件 | | 操作人员为总经理 |
| 后置条件 | | 拒绝单据 |
| Invoice.add | 语法 | | Public ResultMessage add(GiftPO po)  Public ResultMessage add(ImportPO po)  Public ResultMessage add(ExportPO po)  Public ResultMessage add(Import\_ReturnPO po)  Public ResultMessage add(Export\_ReturnPO po)  Public ResultMessage add(ReceiptPO po)  Public ResultMessage add(PaymentPO po) |
| 前置条件 | | 操作人员为总经理，库存管理人员，销售人员、财务人员填写好单据 |
| 后置条件 | | 等待总经理审批 |
| 需要的服务 | | | |
| InvoiceDataServoice.getInvoice() | | 获得一个单据InvoicePO对象 | |
| InvoiceDataServoice.addInvoice(InvoicePO po) | | 添加单据InvoicePO对象 | |
| InvoiceDataServoice.delInvoice(InvoicePO po) | | 移除单据InvoicePO对象 | |
| InvoiceDataService.getAllInvoice() | | 获得所有单据InvoicePO对象 | |
| InvoiceDataService.clear() | | 清空所有单据 | |

Systemlog的接口规格

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Systemlog.show | 语法 | Public ResultMessage show() |
| 前置条件 | 操作人员为财务人员或总经理 |
| 后置条件 | 显示日志内容，返回是否成功 |
| Systemlog.add | 语法 | Public ResultMessage add(String word) |
| 前置条件 | 其他人员完成操作后，发送操作记录并存储 |
| 后置条件 | 添加日志内容，返回是否成功 |
| 需要的服务 | | |
| SystemlogDataService.get() | 获得日志SystemlogPO对象 | |
| SystemlogDataService.add(SystemlogPO po) | 添加日志SystemlogPO对象 | |
| SystemlogDataService.clear() | 清空日志对象 | |

Userbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| User.addUser | 语法 | Public ResultMessage(String name, String password, int duty) |
| 前置条件 | 管理员身份登陆系统 |
| 后置条件 | 完成新账号的添加，可以开始使用 |
| User.delUser | 语法 | Public ResultMessage(String name) |
| 前置条件 | 管理员身份登陆系统 |
| 后置条件 | 删除账号信息 |
| User.show | 语法 | Public ResultMessage show() |
| 前置条件 | 管理员身份登陆系统 |
| 后置条件 | 返回显示的身份信息 |
| User.login | 语法 | Public ResultMessage login(String name, Stirng password) |
| 前置条件 | 有人尝试登陆时 |
| 后置条件 | 验证账号与密码，返回是否成功 |
| 需要的服务 | | |
| UserDataService.addUser(UserPO po) | 添加UserPO对象 | |
| UserDataService.delUser(UserPO po) | 删除UserPO对象 | |
| UserDataService.getUser(String name) | 根据名字查找UserPO对象 | |
| UserDataService.show() | 获得所有UserPO对象 | |
| UserDataService.clear() | 清空所有对象 | |

**5.4数据层的分解**

数据层主要给业务逻辑层提供数据访问服务，包括对于持久化数据的增删改查。Sales业务逻辑需要的服务由SalesDataService接口提供。由于持久化数据的保存可能存在多种形式：Txt文件，序列化文件，数据库等，所示抽象了数据服务。数据层模块的描述具体如图13所示。

**5.4.1数据层模块的职责**

数据称模块的职责如表11所示。

表11 数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| SalesDataService | 持久化数据库的接口，提供集体载入，集体保存，增删改查服务 |
| SalesDataServiceTxtFileImpl | 基于Txt文件的持久化数据库的接口，提供集体载入，集体保存，增删改查服务 |
| SalesDataServiceSerializableFileImpl | 基于序列化文件的持久化数据库的接口，提供集体载入，集体保存，增删改查服务 |
| SalesDataServiceMySqlImpl | 基于MySql数据库的持久化数据库的接口，提供集体载入，集体保存，增删改查服务 |

**5.4.2数据层模块的接口规范**

数据称模块的接口规范如表12所示

表12 数据层模块的接口规范

CommodityData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| CommodityDataService.findGood | 语法 | Public CommodityPO findGood(String name, String type) |
| 前置条件 | 无 |
| 后置条件 | 按名称和型号查找并返回相应的CommodityPO结果 |
| CommodityDataService.findName(String name) | 语法 | Public CommodityPO findName(String name) |
| 前置条件 | 无 |
| 后置条件 | 按名称查找并返回相应的CommodityPO结果 |
| CommodityDataService.findType(String type) | 语法 | Public CommodityPO findType(String type) |
| 前置条件 | 无 |
| 后置条件 | 按型号查找并返回相应的CommodityPO结果 |
| CommodityDataService.addGood(CommodityPO po1, SortPO po2) | 语法 | Public void addGood(CommodityPO po1, SortPO po2) |
| 前置条件 | 数据库中不存在相同的CommodityPO对象 |
| 后置条件 | 在数据库中添加一个CommodityPO对象 |
| CommodityDataService.delGood(CommodityPO po) | 语法 | Public void delGood(CommodityPO po) |
| 前置条件 | 数据库中存在相同的CommodityPO对象，此商品的库存数量为0 |
| 后置条件 | 数据库删除此CommodityPO对象 |
| CommodityDataService.updateGood(CommodityPO po1, CommodityPO po2) | 语法 | Public void updateGood(CommodityPO po1, CommodityPO po2) |
| 前置条件 | 数据库中存在相同名称与型号的CommodityPO，输入的两个CommodityPO的名称与型号相同 |
| 后置条件 | 更新CommodityPO信息 |
| CommodityDataService.addSort(SortPO po) | 语法 | Public void addSort(SortPO po) |
| 前置条件 | 数据库不存在相同名称的SortPO对象 |
| 后置条件 | 数据库添加一个SortPO对象 |
| CommodityDataService.delSort(SortPO po) | 语法 | Public void delSort(SortPO po) |
| 前置条件 | 数据库中存在相同名称的SortPO对象，此分类下不存在其他SortPO对象或CommodityPO对象 |
| 后置条件 | 数据库删除这个SortPO对象 |
| CommodityDataService.updateSort(SortPO po1, SortPO po2) | 语法 | Public void updateSort(SortPO po1, SortPO po2) |
| 前置条件 | 数据库存在相同名称的SortPO对象，输入的两个SortPO对象名称一致 |
| 后置条件 | 数据库更新这个SortPO对象信息 |
| CommodityDataService.addGift(CommodityPO po) | 语法 | Public void addGift(CommodityPO po) |
| 前置条件 | 数据库不存在相同名称与型号的GiftPO，数量小于库存内相同名称与型号CommodityPO对象的数量 |
| 后置条件 | 数据库添加GiftPO对象，库存中相同名称与型号的CommodityPO对象数量减少 |
| CommodityDataService.findGift(String name, String type) | 语法 | Public CommodityPO findGift(String name, String type) |
| 前置条件 | 无 |
| 后置条件 | 根据名字和型号查找并返回GiftPO |
| CommodityDataService.delGift(CommodityPO po) | 语法 | Public void delGift(CommodityPO po) |
| 前置条件 | 数据库中存在相同名称与型号的GiftPO对象 |
| 后置条件 | 数据库删除GiftPO对象，库存中相同名称与型号的CommodityPO数量增加 |
| CommodityDataService.getAll() | 语法 | Public void getAll() |
| 前置条件 | 无 |
| 后置条件 | 获得所有的CommodityPO对象 |
| CommodityDataService.clear() | 语法 | Public void clear() |
| 前置条件 | 进行期初建账之前 |
| 后置条件 | 删除所有CommodityPO与SortPO对象 |

SalesData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| SalesDataService.addCustomer(CustomerPO po) | 语法 | Public void addCustomer(CustomerPO po) |
| 前置条件 | 数据库中不存在相同名称的CustomerPO对象 |
| 后置条件 | 数据库添加CustomerPO对象 |
| SalesDataService.delCustomer(CustomerPO po) | 语法 | Public void delCustomer(CustomerPO po) |
| 前置条件 | 数据库中存在相同名称的CustomerPO对象，此客户的应付或应收为0 |
| 后置条件 | 数据库删除Customer对象 |
| SalesDataService.updateCustomer(CustomerPO po1, CustomerPO po2) | 语法 | Public void updateCustomer(CustomerPO po1, CustomerPO po2) |
| 前置条件 | 数据库中存在相同名称的CustomerPO，输入的两个CustomerPO对象名称一致 |
| 后置条件 | 数据库 |
| SalesDataService.findCustomer(String name) | 语法 | Public CustomerPO findCustomer(String name) |
| 前置条件 | 无 |
| 后置条件 | 根据名称查找并返回CustomerPO对象 |
| SalesDataService.addImport(ImportPO po) | 语法 | Public void addImport(ImportPO po) |
| 前置条件 | 进货单得到总经理审批通过 |
| 后置条件 | 数据库中添加ImportPO对象，完成进货单 |
| SalesDataService.addImport\_Return(Import\_ReturnPO po) | 语法 | Public void addImport\_Return(Import\_ReturnPO po) |
| 前置条件 | 进货退货单得到总经理审批通过，库存中相同名称与型号的CommodityPOd对象数量大于退货数量 |
| 后置条件 | 数据库中添加Import\_ReturnPO对象，完成进货的退货 |
| SalesDataService.addExport(ExportPO po) | 语法 | Public void addExport(ExportPO po) |
| 前置条件 | 销售单得到总经理审批通过，库存中相同名称与型号的CommodityPO对象数量大于销售数量 |
| 后置条件 | 数据库中添加ExportPO对象，完成销售 |
| SalesDataService.addExport\_Return(Export\_ReturnPO po) | 语法 | Public void addExport\_Return(Export\_ReturnPO po) |
| 前置条件 | 销售退货单得到总经理审批通过 |
| 后置条件 | 数据库中添加Export\_ReturnPO对象，完成销售的退货 |
| SalesDataService.getAllCustomer() | 语法 | Public CustomerPO[] getAllCustomer() |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有CustomerPO对象的数组 |
| SalesDataService.getAllImport() | 语法 | Public ImportPO[] getAllImport() |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有Import对象的数组 |
| SalesDataService.getAllImport\_Return() | 语法 | Public Import\_ReturnPO[] getAllImport\_Return() |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有Import\_ReturnPO对象的数组 |
| SalesDataService.getAllExport() | 语法 | Public ExportPO[] getAllExport() |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有ExportPO对象的数组 |
| SalesDataService.getAllExport\_Return() | 语法 | Public Export\_ReturnPO[] getAllExport\_Return() |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有Export\_ReturnPO对象的数组 |
| SalesDataService.clear() | 语法 | Public void clear() |
| 前置条件 | 进行期初建账之前 |
| 后置条件 | 清楚所有的CustomerPO，ImportPO，Import\_ReturnPO，ExportPO，Export\_ReturnPO对象 |

AccountData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| AccountDataService.getAccount() | 语法 | Public AccountPO getAccount() |
| 前置条件 | 无 |
| 后置条件 | 返回当前账户AccountPO对象 |
| AccountDataService.addAccount(AccountPO po) | 语法 | Public void addAccount(AccountPO po) |
| 前置条件 | 数据库中不存在相同名称的AccountPO对象 |
| 后置条件 | 数据库添加AccountPO对象 |
| AccountDataService.delAccount(AccountPO po) | 语法 | Public void delAccount(AccountPO po) |
| 前置条件 | 数据库中存在相同名称的AccountPO对象，此对象中应付和应收为0 |
| 后置条件 | 数据库删除AccountPO对象 |
| AccountDataService.updateAccount(AccountPO po1, AccountPO po2) | 语法 | Public void updateAccount(AccountPO po1, AccountPO po2) |
| 前置条件 | 数据库中存在相同名称的AccountPO对象，输入的两个AccountPO对象的名称相同 |
| 后置条件 | 数据库更新AccountPO对象的信息 |
| AccountDataService.findAccount(String name) | 语法 | Public AccountPO findAccount(String name) |
| 前置条件 | 无 |
| 后置条件 | 根据名称查找并返回AccountPO对象 |
| AccountDataService.getAllAccount() | 语法 | Public AccountPO[] getAllAccount() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有AccountPO对象的数组 |
| AccountDataService.addReceipt(ReceiptPO po) | 语法 | Public void addReceipt(ReceiptPO po) |
| 前置条件 | 收款单得到总经理审批通过 |
| 后置条件 | 数据库添加ReceiptPO对象 |
| AccountDataService.addPayment(PaymentPO po) | 语法 | Public void addPayment(PaymentPO po) |
| 前置条件 | 付款单得到总经理审批通过 |
| 后置条件 | 数据库添加PaymentPO对象 |
| AccountDataService.getAllReceipt() | 语法 | Public ReceiptPO[] getAllReceipt() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有ReceiptPO对象的数组 |
| AccountDataService.getAllPayment() | 语法 | Public PaymentPO[] getAllPayment() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有PaymentPO对象的数组 |
| AccountDataService.clear | 语法 | Public void clear() |
| 前置条件 | 进行期初建账之前 |
| 后置条件 | 清除所有AccountPO，ReceiptPO，PaymentPO对象 |

FinancialData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| FinancialDataService.getSaleList() | 语法 | Public SaleListPO[] getSaleList() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有SaleListPO对象的数组 |
| FinancialDataService.getAllBill() | 语法 | Public AllBillPO[] getAllBill() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有AllBillPO对象的数组 |
| FinancialDataService.getOperatingCondition() | 语法 | Public OperatingConditionPO[] getOperatingCondition() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有OperatingConditionPO对象的数组 |
| FinancialDataService.addSaleList(SaleListPO po) | 语法 | Public void addSaleList(SaleListPO po) |
| 前置条件 | 完成一个进货或销售单据的审批 |
| 后置条件 | 数据库添加一个SaleListPO对象 |
| FinancialDataService.addAllBill(AllBillPO po) | 语法 | Public void addAllBill(AllBillPO po) |
| 前置条件 | 完成任何单据的审批 |
| 后置条件 | 数据库添加一个AllBillPO对象 |
| FinancialDataService.addOperatingCondition(OperatingConditionPO po) | 语法 | Public void addOperationCondition(OperatingConditionPO po) |
| 前置条件 | 完成收付款单据的审批 |
| 后置条件 | 数据库添加一个OperatingConditionPO对象 |
| FinancialDataService.clear() | 语法 | Public void clear() |
| 前置条件 | 进行期初建账之前 |
| 后置条件 | 清楚所有SaleListPO，AllBillPO，OperatingConditionPO对象 |

InitializationData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| InitializationDataService.getInformation | 语法 | Public InitializationPO getInformation() |
| 前置条件 | 无 |
| 后置条件 | 返回InitializationPO对象 |

PromotionData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| PromotionDataService.addDiscount(DiscountPO po) | 语法 | Public void addDiscount(DiscountPO po) |
| 前置条件 | 无 |
| 后置条件 | 添加或替换当前的DiscountPO对象 |
| PromotionDataService.addGift(ProGiftPO po) | 语法 | Public void addGift(ProGiftPO po) |
| 前置条件 | 无 |
| 后置条件 | 添加或替换当前的ProGiftPO对象 |
| PromotionDataService.delDiscount(DiscountPO po) | 语法 | Public void delDiscount(DiscountPO po) |
| 前置条件 | 当前存在处于有效期的DiscountPO对象 |
| 后置条件 | 清除DiscountPO对象 |
| PromotionDataService.delGift(ProGiftPO po) | 语法 | Public void delGift(ProGiftPO po) |
| 前置条件 | 当前存在处于有效期的ProGiftPO对象 |
| 后置条件 | 清除ProGift对象 |
| PromotionDataservice.getDiscount() | 语法 | Public DiscountPO getDiscount() |
| 前置条件 | 无 |
| 后置条件 | 返回当前的DiscountPO对象 |
| PromotionDataService.getGift() | 语法 | Public ProGiftPO getGift() |
| 前置条件 | 无 |
| 后置条件 | 返回当前的ProGiftPO对象 |
| PromotionDataService.showDiscount() | 语法 | Public DIscountPO[] showDiscount() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有的DiscountPO对象的数组 |
| PromotionDataService.showProGift() | 语法 | Public ProGiftPO[] showProGift() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有的ProGiftPO对象的数组 |

InvoiceData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| InvoiceDataServoice.getInvoice() | 语法 | Public InvoicePO getInvoice() |
| 前置条件 | 无 |
| 后置条件 | 返回最新的待审批的InvoicePO对象 |
| InvoiceDataServoice.addInvoice(InvoicePO po) | 语法 | Public void addInvoice(InvoicePO po) |
| 前置条件 | 无 |
| 后置条件 | 添加InvoicePO对象 |
| InvoiceDataServoice.delInvoice(InvoicePO po) | 语法 | Public void delInvoice(InvoicePO po) |
| 前置条件 | 存在相同编号的InvoicePO对象 |
| 后置条件 | 删除InvoicePO对象 |
| InvoiceDataService.getAllInvoice() | 语法 | Public InvoicePO[] getAllInvoice() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有InvoicePO对象的数组 |
| InvoiceDataService.clear() | 语法 | Public void clear() |
| 前置条件 | 无 |
| 后置条件 | 清除所有InvoicePO对象 |

SystemlogData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| SystemlogDataService.get() | 语法 | Public SystemlogPO[] get() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有SystemlogPO对象的数组 |
| SystemlogDataService.add(String word) | 语法 | Public void add(SystemlogPO po) |
| 前置条件 | 无 |
| 后置条件 | 添加SystemlogPO对象 |
| SystemlogDataService.clear() | 语法 | Public void clear() |
| 前置条件 | 无 |
| 后置条件 | 清除所有SystemlogPO对象 |

UserData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| UserDataService.addUser(UserPO po) | 语法 | Public void addUser(UserPO po) |
| 前置条件 | 不存在相同名称的UserPO对象 |
| 后置条件 | 添加UserPO对象 |
| UserDataService.delUser(UserPO po) | 语法 | Public void delUser(UserPO po) |
| 前置条件 | 存在相同名称的UserPO对象 |
| 后置条件 | 删除UserPO对象，若用户正在登陆，则强制退出程序 |
| UserDataService.getUser(String name) | 语法 | Public UserPO getUser(String name) |
| 前置条件 | 无 |
| 后置条件 | 根据名称查找返回UserPO对象 |
| UserDataService.show() | 语法 | Public UserPO[] show() |
| 前置条件 | 无 |
| 后置条件 | 返回一个包含所有UserPO对象的数组 |
| UserDataService.clear() | 语法 | Public void clear() |
| 前置条件 | 无 |
| 后置条件 | 清除所有UserPO对象 |

**6.信息视角**

**6.1 数据持久化对象**

系统的PO类就是对应的相关的实体类，在此只做简单的介绍。

UserPO类包括用户的用户名、密码属性。

CommodityPO类是包含商品的变化、价格、数量和名字属性

SalesPO类是保存销售时的数据的类，包含编号、会员编号、商品列表、总价、折扣、客户支付金额、找零金额等属性

SalesLineItemPO是保持销售记录中一行的信息的类，包含商品编号、数量、小计

CommodityGiftPromotionPO包含商品ID、促销起始日、促销结束日、礼品编号、礼品数量。

持久化用户对象UserPO的定义如图14所示

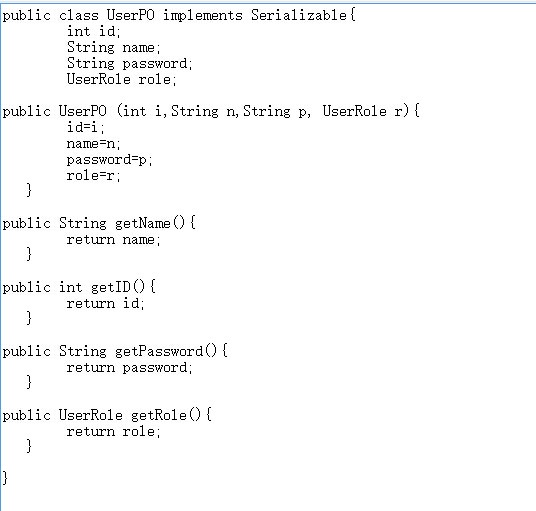


图14 持久化用户对象UserPO的定义

**6.2 Txt持久化格式**

Txt数据保持格式以Commodity.txt为例。每行分别对应货号、商品名称、价格、数量。中间用“：”隔开。如下所示：

123：杯子：10:32

456：桌子：20:22

**6.3 数据库表**

数据库中包含User表、Commodity表、Sales表、SalesLineItem表、CommodityGiftPromotion表、CommodityPricePromotion表、Account表、Initialization表、Invoice表