**物流管理系统（ELMS）**

**软件体系结构**

**描述文档**



**南京大学 软件学院**

**GGS.DDU**

张海涛 郑闻昊 张文玘 周颖婷

**2015年10月23日**

[1.引言 3](#_Toc433444115)

[1.1编制目的 3](#_Toc433444116)

[1.2词汇表 3](#_Toc433444117)

[1.3参考资料 3](#_Toc433444118)

[2.产品概述 3](#_Toc433444119)

[3.逻辑视角 3](#_Toc433444120)

[4.2运行时进程 7](#_Toc433444121)

[4.3物理部署 8](#_Toc433444122)

[5.接口视角 8](#_Toc433444123)

[5.1模块的职责 8](#_Toc433444124)

[5.2用户界面层的分解 10](#_Toc433444125)

[5.2.1用户界面层模块的职责 12](#_Toc433444126)

[5.2.2用户界面层模块的接口规范 14](#_Toc433444127)

[5.2.3用户界面模块设计原理 20](#_Toc433444128)

[5.3业务逻辑层的分解 20](#_Toc433444129)

[5.3.1业务逻辑层模块的职责 20](#_Toc433444130)

[5.3.2业务逻辑层模块的接口规范 21](#_Toc433444131)

[5.4数据层的分解 30](#_Toc433444132)

[5.4.1数据层模块的职责 33](#_Toc433444133)

[5.4.2 数据层模块的接口规范 36](#_Toc433444134)

[6信息视角 42](#_Toc433444135)

[6.1 数据持久化对象 42](#_Toc433444136)

[6.2 Txt持久化格式 57](#_Toc433444137)

[6.3 数据库表 57](#_Toc433444138)

# 1.引言

## 1.1编制目的

本报告详细完成对物流管理系统的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员以及用户的沟通。

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 |
| ELMS | 物流管理系统 | …… |
| …… | …… | …… |

## 1.3参考资料

# 2.产品概述

参考物流管理系统用例文档和物流管理系统软件需求规格说明中对产品的概括描述。

# 3.逻辑视角

物流管理系统中，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3-1和图3-2所示。

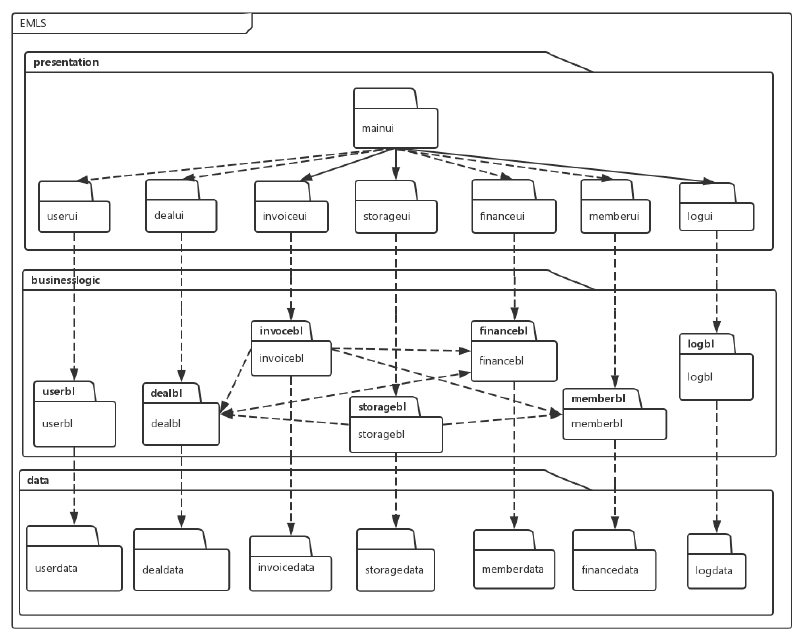
4.组合视角

4.1开发包图

物流管理系统的最终开发包设计如表4.1所示。



**图3-1 参照体系结构风格的包图表达逻辑视角**

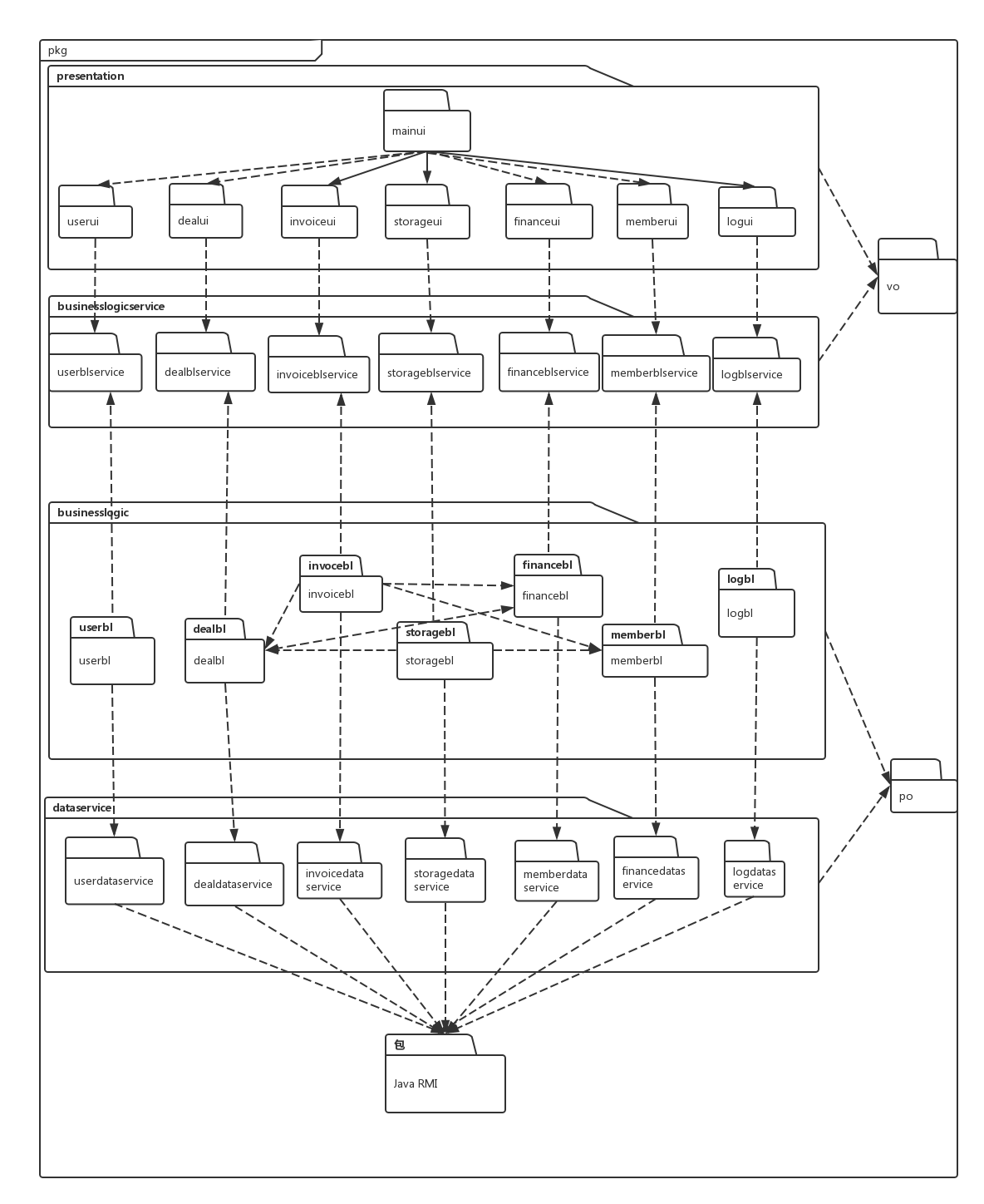


**图3-2 体系结构初始逻辑设计包图表达逻辑视角**

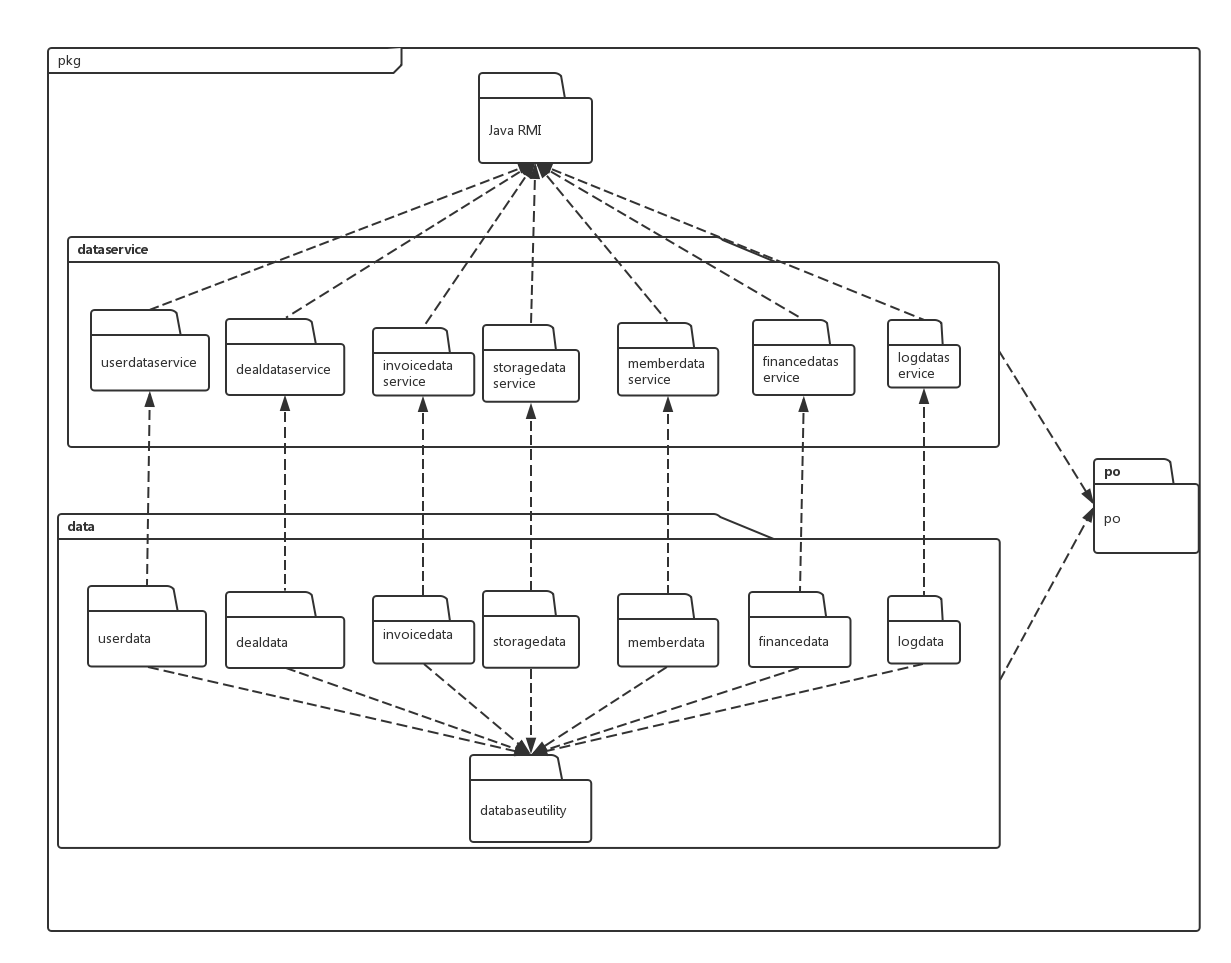
**表4.1物流管理系统的最终开发包设计**

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| mainui | userui,dealui,invoiceui,storageui,financeui,memberui,logui,vo |
| userui | userblservice,界面类库包,vo |
| userblservice |  |
| userbl | UserInterface,UserDataClient,UserPO |
| userdataservice | Java RMI,po |
| userdata | RIM,po,databaseutility |
| dealui | dealblservice,界面类库包,vo |
| dealblservice |  |
| dealbl | dealblservice,dealdataservice,financebl,po |
| dealdataservice | Java RMI,po |
| dealdata | Java RIM,po,databaseutility |
| invoiceui | invoiceblservice,界面类库包,vo |
| invoiceblservice |  |
| invoicebl | invoiceblservice,dealbl,financebl,memberbl,invoicedataservice,po |
| invoicedataservice | Java RMI,po |
| invoicedata | Java RIM,po,databaseutility |
| storageui | storageblservice,界面类库包,vo |
| storageblservice |  |
| storagebl | storageblservice,dealbl,memberbl,storagedataservice,po |
| storagedataservice | Java RMI,po |
| storagedata | Java RIM,po,databaseutility |
| financeui | financeblservice,界面类库包,vo |
| financeblservice |  |
| financebl | financeblservice,dealbl,financedataservice,po |
| financedataservice | Java RMI,po |
| financedata | Java RIM,po,databaseutility |
| memberui | memberblservice,界面类库包,vo |
| memberblservice |  |
| memberbl | memberblservice,memberdataservice |
| memberdataservice | Java RMI,po |
| memberdata | Java RIM,po,databaseutility |
| logui | logblservice,界面类库包,vo |
| logblservice |  |
| logbl | logblservice,logdataservice,po |
| logdataservice | Java RMI,po |
| logdata | Java RIM,po,databaseutility |
| vo |  |
| po |  |
| utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| databaseutility | JDBC |

物流管理系统客户端开发包如图4.1-1所示，服务器端开发包图如图4.1-2所示。



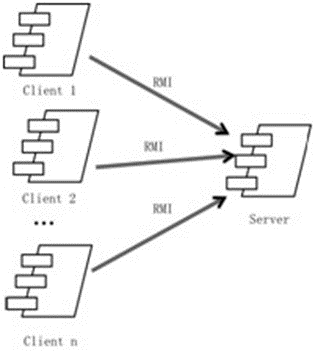
**图4.1-1物流管理系统客户端开发包图**



**图4.1-2物流管理系统服务器端开发包图**

## 4.2运行时进程

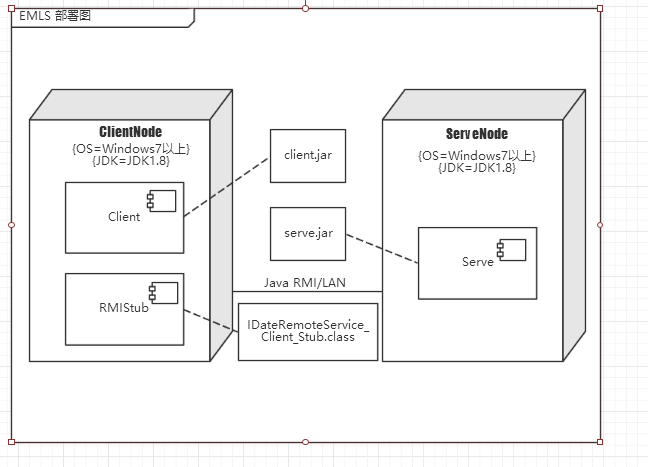
在物流管理系统中，会有多个客户端进程和一个服务器端进程，其进程图如图4.2所示。结合部署图，客户端进程是在客户端机器上运行，服务器端进程是在服务器端机器上运行。



**图4.2 进程图**

## 4.3物理部署

物流管理系统中客户端构件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4.3所示。

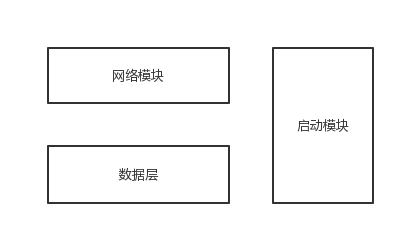
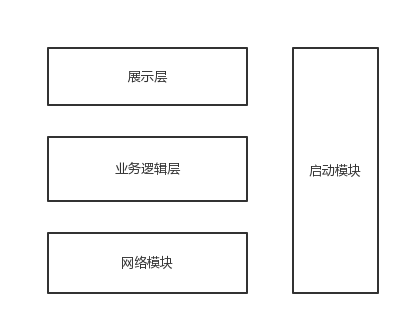


**图4.3 部署图**

# 5.接口视角

## 5.1模块的职责

客户端模块和服务器端模块视图分别如图5.1-1和5.1-2所示。客户端各层的职责分别如表5.1-3和表5.1-4所示。



**图5.1-1 客户端模块视图 图5.1-2 服务器端模块视图**

**表5.1-3 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 | 基于窗口的物流系统客户端用户界面 |
| 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5.1-4 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。层之间调用的接口如表5.1-5所示。

**表5.1-5 层之间调用的接口**

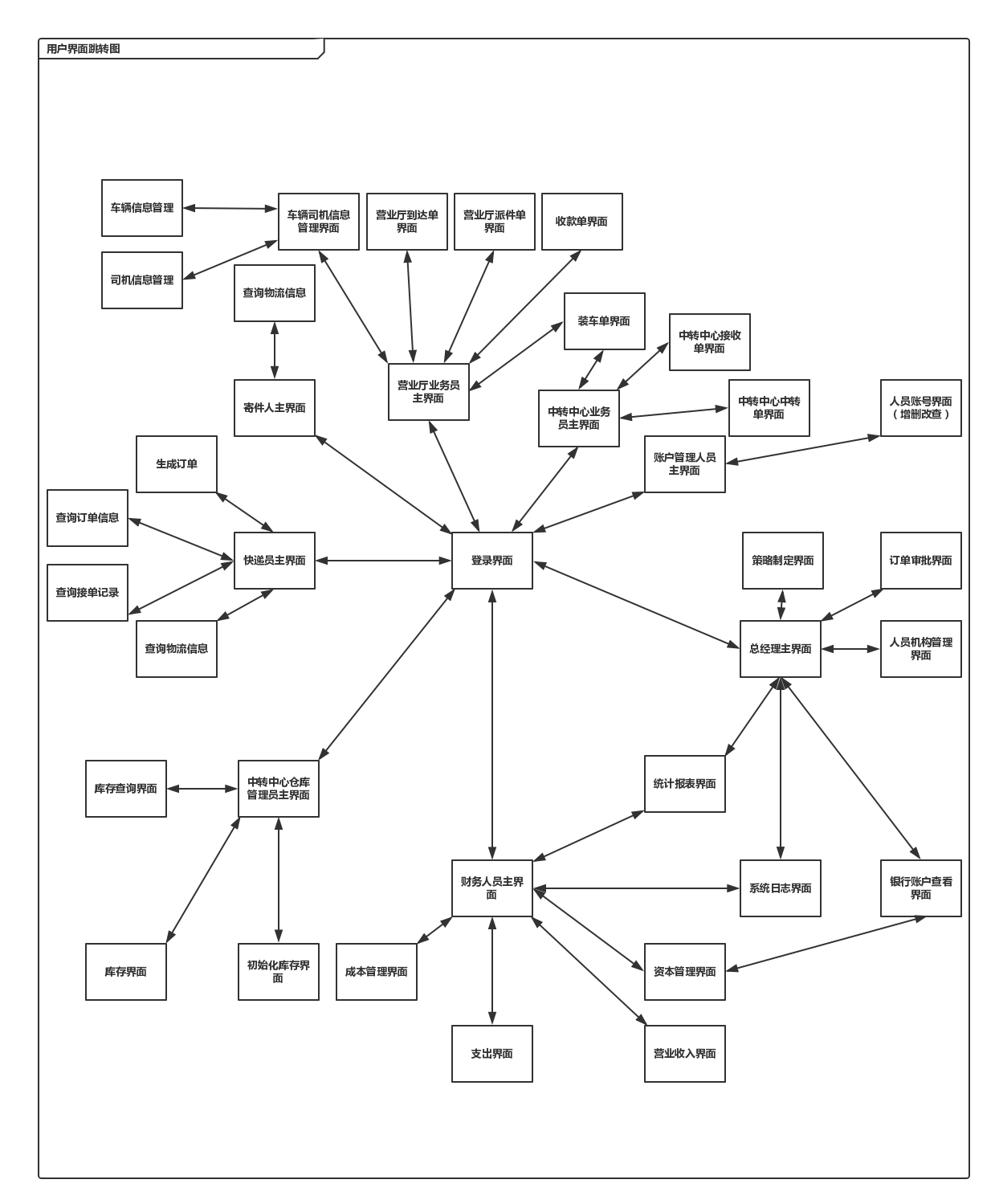
|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| userblservice  dealblservice  invoiceblservice  storageblservice  financeblservice  memberblservice  logblservice | 客户端展示层 | 客户端业务逻辑层 |
| userdataservice  dealdataservice  invoicedataservice  storagedataservice  financedataservice  memberdataservice  logdataservice | 客户端业务逻辑层 | 服务器端数据层 |

借用

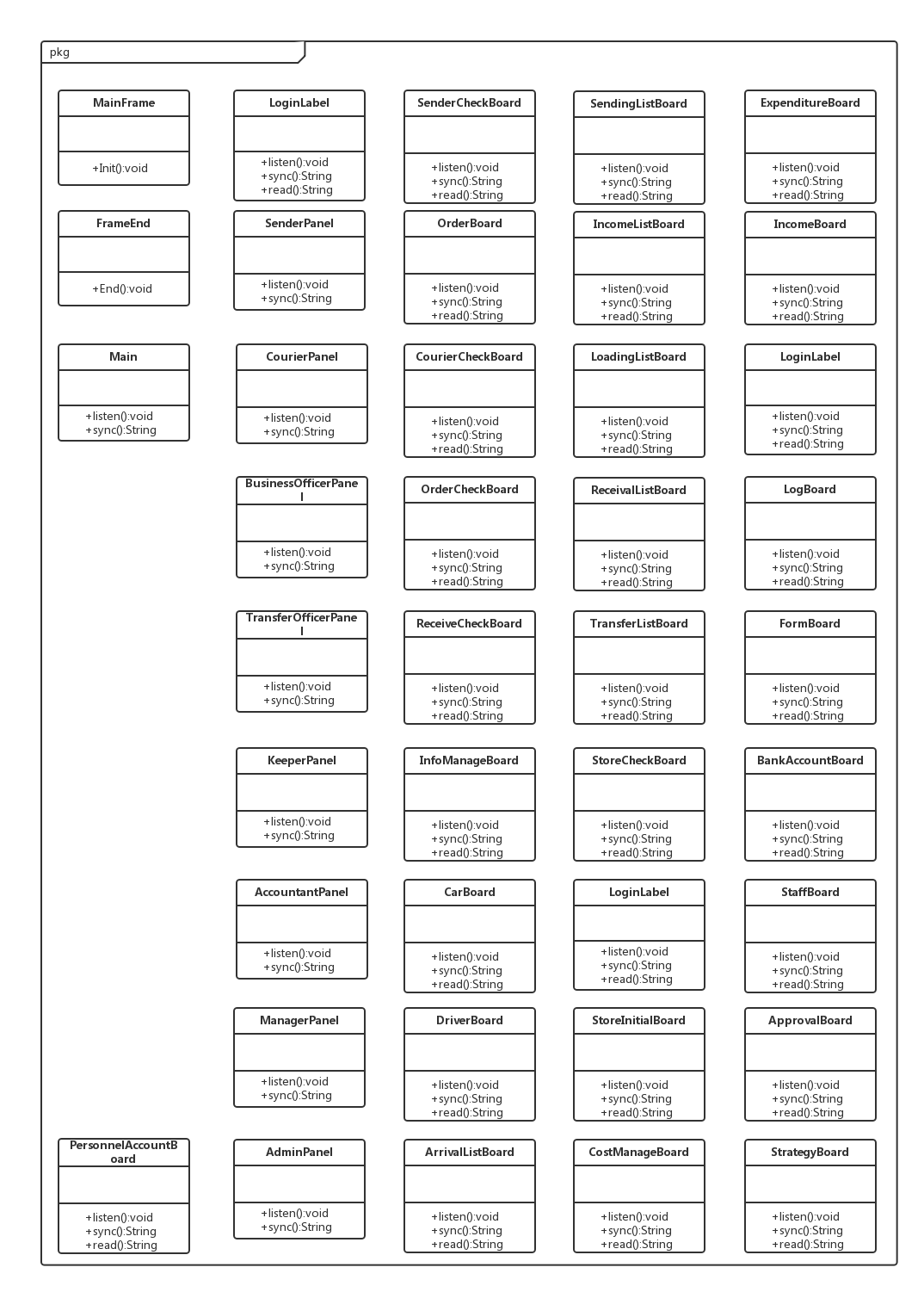
## 5.2用户界面层的分解

根据需求，系统存在37个用户界面：登录界面、寄件人主界面、快递员主界面、营业厅业务员主界面、中转中心业务员主界面、中转中心仓库管理员主界面、财务人员主界面、总经理主页面、账户管理人员主界面、寄件人查询物流信息、生成订单界面、快递员查询物流信息、查询订单信息、查询接单记录、车辆司机信息管理界面、车辆信息管理界面、司机信息管理界面、营业厅到达单界面、营业厅派件单界面、收款单界面、装车单界面、中转中心接收单界面、中转中心中转单界面、库存查询界面、库存界面、初始化库存界面、成本管理界面、支出界面、营业收入界面、资本管理界面、系统日志界面、统计报表界面、银行账户查看界面、人员机构管理界面、订单审批界面、策略制定界面、人员账号界面。界面跳转如图5.2-1所示。

服务器端和客户端的用户界面设计接口是一致的，只是具体的页面不一样。用户界面类如图5.2-2所示。



**图5.2-1 用户界面跳转**



**图5.2-2 用户界面类**

### 5.2.1用户界面层模块的职责

如表5.2.1所示位用户界面层模块的职责

**表5.2.1 用户界面层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| MainFrame | 界面Frame，负责界面的显示和界面的跳转。 |
| FrameEnd | 负责界面的关闭 |
| Main | 系统主界面，负责主界面的监听和信息传递 |
| LoginLabel | 登录界面，负责输入信息的传递和同步 |
| SenderPanel | 寄件人板块负责该板块的监听和信息传递 |
| CourierPanel | 快递员板块，负责该板块的监听和信息传递 |
| BusinessOfficerPanel | 营业厅业务员板块，负责该板块的监听和信息传递 |
| TransferOfficerPanel | 中转中心业务员板块，负责该板块的监听和信息传递 |
| KeeperPanel | 中转中心仓库管理员板块，负责该板块的监听和信息传递 |
| AccountantPanel | 财务人员板块，负责该板块的监听和信息传递 |
| ManagerPanel | 总经理板块，负责该板块的监听和信息传递 |
| AdminPanel | 管理员板块，负责该板块的监听和信息传递 |
| SenderCheckBoard | 寄件人查询物流信息界面，负责该界面的监听和信息的传递与同步 |
| OrderBoard | 生成订单界面，负责该界面的监听和信息传递 |
| CourierCheckBoard | 快递员查询物流信息，负责该界面的监听和信息的传递与同步 |
| OrderCheckBoard | 查询订单信息，负责该界面的监听和信息传递 |
| ReceiveCheckBoard | 查询接单记录，负责该界面的监听和信息传递 |
| InfoManageBoard | 车辆司机信息管理界面，负责该界面的监听和信息传递 |
| CarBoard | 车辆信息管理界面，负责该界面的监听和信息传递 |
| DriverBoard | 司机信息管理界面，负责该界面的监听和信息传递 |
| ArrivalListBoard | 营业厅到达单界面，负责该界面的监听和信息传递 |
| SendingListBoard | 营业厅派件单界面，负责该界面的监听和信息传递 |
| IncomeListBoard | 收款单界面，负责该界面的监听和信息传递 |
| LoadingListBoard | 装车单界面，负责该界面的监听和信息传递 |
| ReceivalListBoard | 中转中心接收单界面，负责该界面的监听和信息传递 |
| TransferListBoard | 中转中心中转单界面，负责该界面的监听和信息传递 |
| StoreCheckBoard | 库存查询界面，负责该界面的监听和信息传递 |
| StoreBoard | 库存界面，负责该界面的监听和信息传递 |
| StoreInitialBoard | 初始化库存界面，负责该界面的监听和信息传递 |
| CostManageBoard | 成本管理界面，负责该界面的监听和信息传递 |
| ExpenditureBoard | 支出界面，负责该界面的监听和信息传递 |
| IncomeBoard | 营业收入界面，负责该界面的监听和信息传递 |
| CapitalManageBoard | 资本管理界面，负责该界面的监听和信息传递 |
| LogBoard | 系统日志界面，负责该界面的监听和信息传递 |
| FormBoard | 统计报表界面，负责该界面的监听和信息传递 |
| BankAccountBoard | 银行账户查看界面，负责该界面的监听和信息传递 |
| StaffBoard | 人员机构管理界面，负责该界面的监听和信息传递 |
| ApprovalBoard | 订单审批界面，负责该界面的监听和信息传递 |
| StrategyBoard | 策略制定界面，负责该界面的监听和信息传递 |
| PersonnelAccountBoard | 人员账号界面，负责该界面的监听和信息传递 |

### 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5.2.2-1所示

**表5.2.2-1 用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | init(String args []) |
| 前置条件 | 收到界面跳转请求 |
| 后置条件 | 显示Frame以及LoginPanel |

|  |  |  |
| --- | --- | --- |
| FrameEnd | 语法 | End(String args []) |
| 前置条件 | 收到关闭某界面的请求 |
| 后置条件 | 关闭相应界面 |

|  |  |  |
| --- | --- | --- |
| Main | 语法 | listen();sync() |
| 前置条件 | 系统开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或MainEnd |

|  |  |  |
| --- | --- | --- |
| LoginLabel | 语法 | listen();read();sync(); |
| 前置条件 | 被Main请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交userbl和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| SenderPanel | 语法 | listen();sync(); |
| 前置条件 | 由loginbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| CourierPanel | 语法 | listen();sync(); |
| 前置条件 | 由loginbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| BusinessOfficerPanel | 语法 | listen();sync(); |
| 前置条件 | 由loginbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| TransferOfficerPanel | 语法 | listen();sync(); |
| 前置条件 | 由loginbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| KeeperPanel | 语法 | listen();sync(); |
| 前置条件 | 由loginbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| AccountantPanel | 语法 | listen();sync(); |
| 前置条件 | 由loginbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| ManagerPanel | 语法 | listen();sync(); |
| 前置条件 | 由loginbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| AdminPanel | 语法 | listen();sync(); |
| 前置条件 | 由loginbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| SenderCheckBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Senderbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| OrderBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Courierbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| CourierCheckBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Courierbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| OrderCheckBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Courierbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| ReceiveCheckBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Courierbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值并提交MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| InfoManageBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由BusinessOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| CarBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由BusinessOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| DriverBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由BusinessOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| ArrivalListBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由BusinessOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| SendingListBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由BusinessOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| IncomeListBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由BusinessOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| LoadingListBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由BusinessOfficerbl或TransferOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| ReceivalListBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由TransferOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| TransferListBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由TransferOfficerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| StoreCheckBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Keeperbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| StoreBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Keeperbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| StoreInitialBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Keeperbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| CostManageBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Accountantbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| ExpenditureBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Accountantbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| IncomeBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Accountantbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| CapitalManageBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Accountantbl或Managerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| LogBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Accountantbl或Managerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| FormBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Accountantbl或Managerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| BankAccountBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Accountantbl或Managerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| StaffBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Managerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| ApprovalBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Managerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| StrategyBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Managerbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

|  |  |  |
| --- | --- | --- |
| PersonnelAccountBoard | 语法 | listen();read();sync(); |
| 前置条件 | 由Adminbl请求开启，通过MainFrame初始化 |
| 后置条件 | 返回有效监听值和输入值并提交面板层对应逻辑层和MainFrame或FrameEnd |

用户界面层需要的服务接口如表5.2.2-2所示

**表5.2.2-2 用户界面层模块需要的服务接口**

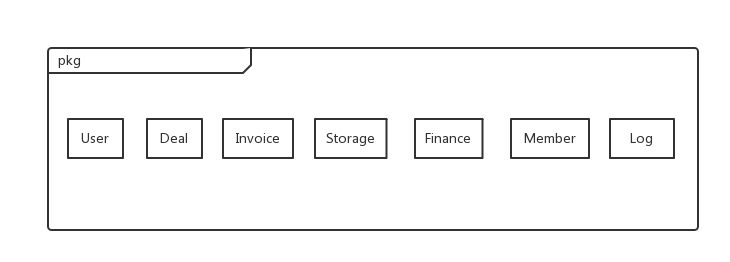
|  |  |
| --- | --- |
| **服务名** | **服务** |
| businesslogicservice.LoginBLService | 登陆界面的业务逻辑接口 |
| businesslogicservice.\*BLService | 每个界面都有一个相应的业务逻辑接口  （此处不一一赘述） |

### 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页面的业务逻辑处理对象。例如，User对象负责处理登录界面的业务逻辑；Deal对象负责快递员交易界面的业务逻辑。业务逻辑的设计如图5.3所示。



**图5.3 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5.3.1所示

**表5.3.1 业务逻辑层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| userbl | 负责实现对应与登录界面所需要的服务 |
| dealbl | 负责实现快递员交易界面所需要的服务 |
| invoicebl | 负责实现单据界面所需要的服务 |
| storagebl | 负责实现库存管理界面所需要的服务 |
| financebl | 负责实现经济管理界面所需要的服务 |
| memberui | 负责实现人事管理界面所需要的服务 |
| logui | 负责实现查询系统日志界面所需要的服务 |

### 5.3.2业务逻辑层模块的接口规范

#### 5.3.2.1.userbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.login | 语法 | Public ResultMessage 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. getChange | 语法 | Public resultMessage getChange( UserRole role) |
| 前置条件 | 启动一个信息更改操作 |
| 后置条件 | 对该员工的账号信息（权限）进行更改 |
| User. addAccount | 语法 | Public resultMessage addAccount(string id, long password, UserRole role) |
| 前置条件 | 启动一个账号增加操作 |
| 后置条件 | 新建一个新的账号，并在其中增加新的账号信息 |
| User. deleteAccount | 语法 | Public resultMessage deleteAccount(string id) |
| 前置条件 | 启动一个账号删除操作 |
| 后置条件 | 删除员工id对应账号 |
| User. inquiryAccount | 语法 | Public resultMessage inquiryAccount(string id) |
| 前置条件 | 启动一个账号查找工作 |
| 后置条件 | 查找是否存在对应的User，根据该员工信息返回员工账号信息 |
| User. endAccountOpt | 语法 | Public void endAccountOpt() |
| 前置条件 | 已结束所有对账户的操作 |
| 后置条件 | 结束此次账号管理，持久化更新涉及的领域对象的数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory. geUserDatabase | 得到User数据库的服务的引用 | |
| UserDataService. insert(UserPO po) | 在数据库中插入UserPO对象 | |
| UserDataService. delete(UserPO po) | 在数据库中删除UserPO对象 | |
| UserDataService. update(UserPO po) | 更新数据库中的UserPO对象 | |
| UserDataService. find( int id) | 根据ID进行查找UserPO对象 | |

#### 5.3.2.2.dealbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Deal. BuildOrder | 语法 | Public booleanBuildOrder(DealPO po) |
| 前置条件 | 启动一个生成订单的操作 |
| 后置条件 | 检验生成的订单是否正确且未用 |
| Deal. getOrder | 语法 | Public ResultMessage getOrder(string orderID) |
| 前置条件 | 输入订单号进行查询且订单号确认无误 |
| 后置条件 | 查找是否存在相应的未送达订单（格式、是否存在、是否已送达），根据输入的订单号返回结果 |
| Deal. recipients | 语法 | PublicResultMessage recipients(string name, string time) |
| 前置条件 | 已输入收件信息 |
| 后置条件 | 检验输入的收件人信息以及收件信息是否正确，根据检验结果返回结果 |
| Deal. achievement | 语法 | Public resultMessage achievement(string time1, string time2) |
| 前置条件 | 选择查询送货记录 |
| 后置条件 | 检验输入的时间段是否正确且合理，并根据时间段信息返回该时间段的送货记录 |
| Deal. endDealOpt | 语法 | Public void endDealOpt() |
| 前置条件 | 已结束本次操作 |
| 后置条件 | 结束此次交易处理/查询操作，持久化更新涉及的领域对象的数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory. getDealDatabase | 得到deal数据库的服务的引用 | |
| DealDataService.find(long id) | 根据ID查找DealPO对象 | |
| DealDataService.findbyCourier(string courier) | 按照快递员姓名查找所属的DealPO结果 | |
| DealDataService.findbyHall(string hall) | 查找并返回该营业厅所有的DealPO | |
| DealDataService.insert(DealPO po) | 在数据库中插入DealPO对象 | |
| DealDataService.delete(DealPO po) | 在数据库中删除DealPO对象 | |
| DealDataService. Update(DealPO po) | 在数据库中更新这个DealPO对象 | |

#### 5.3.2.3.invocebl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Invoice. List | 语法 | Public resultMessage List(invoiceType type, invoiceVO vo) |
| 前置条件 | 选择单据的类别并输入单据的各项数据 |
| 后置条件 | 检验数据是否正确且合理，并根据检查结果返回对应的单据生成界面 |
| Invoice. inquiry | 语法 | Public invoiceVO inquiry(string invoiceID) |
| 前置条件 | 启动一个单据查看操作 |
| 后置条件 | 返回该单据的具体信息 |
| Invoice. change | 语法 | Public invoicePO change(string invoiceID) |
| 前置条件 | 启动一个单据修改操作 |
| 后置条件 | 将该单据状态修改为草稿状态，并退回给对应提交单位进行修改 |
| Invoice. delete | 语法 | Public void delete(invoicePO po) |
| 前置条件 | 启动一个单据删除操作 |
| 后置条件 | 将该单据信息删除 |
| Invoice. approval | 语法 | Public ResultMessage approval(string invoiceID) |
| 前置条件 | 单据信息正确且合理 |
| 后置条件 | 将该单据状态修改为审批后状态 |
| Invoice. endOpt | 语法 | Public void endOpt() |
| 前置条件 | 对单据的操作已经结束 |
| 后置条件 | 结束此次单据审批操作，持久化更新涉及的领域对象的数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory. getInvoiceDatabase | 得到invoice数据库的引用 | |
| invoiceDataService. Find(string id) | 根据ID查找invoicePO对象 | |
| invoiceDataService. Insert(invoicePO po) | 在数据库中插入invoicePO对象 | |
| invoiceDataService.delete(invoicePO po) | 在数据库中删除invoicePO对象 | |

注：：写了invoiceType的类，存储单据类别

(ArrivalListPO(营业厅到达单)，SendingListPO（营业厅派件单）, IncomeListPO(收款单),LoadingListPO(装车单),receivalListPO(中转中心接受单)，,transferListPO(中转中心中转单),Store\_InListPO(入库单)，Store\_OutListPO(出库单),均继承自InvoicePO类。

#### 5.3.2.4.storagebl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Storage. Init | 语法 | PublicArrayList<StoragePO> init(ArrayList<StorageVO vo>) |
| 前置条件 | 启动一次库存信息初始化操作 |
| 后置条件 | 返回新输入的库存信息列表 |
| Storage. Storage\_inChoose | 语法 | Public Boolean storage\_inChoose(string area, int row, int line, int shelf) |
| 前置条件 | 选择入库位置 |
| 后置条件 | 检验该商品是否能入库 |
| Storage. storage\_inRecord | 语法 | Public ResultMessage storage\_inRecord(storageVO vo) |
| 前置条件 | 该商品可以入库 |
| 后置条件 | 将对应商品信息及入库时间存入库存信息中 |
| Storage. Storage\_out | 语法 | Public void storage\_out(string orderID) |
| 前置条件 | 当前库存中存在商品ID对应的货物 |
| 后置条件 | 在库存信息中删除该商品，调用出库单 |
| Storage. inquiryByTime | 语法 | Public ResultMessage inquiryByTime(string time1, string time2, string center) |
| 前置条件 | 时间范围正确 |
| 后置条件 | 返回该中转仓库对应时间段内商品的库存信息 |
| Storage.inquiryAll | 语法 | Public resultMessage inquiryAll(string center) |
| 前置条件 | 输入当前操作的中转中心业务员所属中转仓库 |
| 后置条件 | 返回当前中转仓库内的所有库存信息 |
| Storage. inquiry | 语法 | Public resultMessage inquiry(string id) |
| 前置条件 | 输入要查询的订单ID |
| 后置条件 | 检验库存中是否有该订单，如果有，则显示该订单的详细信息 |
| Storage. endStoreOpt | 语法 | Public void endStoreOpt() |
| 前置条件 | 所有库存操作已完成 |
| 后置条件 | 结束此次库存操作，持久化更新涉及的领域对象的数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory. getStorageDatabase | 得到Storage数据库的服务的引用 | |
| StorageDataService. Find(string id) | 根据订单id在数据库中查找storagePO对象 | |
| StorageDataService. Findall(string center) | 在数据库中查找该中转仓库中的所有storagePO对象 | |
| StorageDataService. findByTime(string time1, string time2, string center) | 在数据库中查找该中转仓库中time1到time2时间段内的所有storagePO对象 | |
| StorageDataService. Insert(StoragePO po) | 在数据库中插入StoragePO对象 | |
| StorageDataService. Delete(StoragePO po) | 在数据库中删除storagePO对象 | |
| storageDataService. Update(storagePOpo) | 在数据库中更新storagePO对象 | |

#### 5.3.2.5Financebl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Finance. inquiry | 语法 | Public resultMessage inquiry(string id) |
| 前置条件 | 输入该操作的编号 |
| 后置条件 | 返回该操作的详细信息(financePO) |
| Finance. inquiryHallIncome | 语法 | Public resultMessage inquiryHallIncome(string center) |
| 前置条件 | 输入营业厅编号 |
| 后置条件 | 返回该营业厅的所有收入项（IncomePO）列表 |
| Finance. inquiryInByTime | 语法 | Public resultMessage inquiryInByTime(String time1, string time2,string hall) |
| 前置条件 | 输入要查询的时间段和营业厅编号 |
| 后置条件 | 返回该营业厅中在该时间段内的所有收入项列表 |
| Finance. inquiryIncome | 语法 | Public resultMessage inquiryIncome(string time1, string time2) |
| 前置条件 | 输入要查询的时间段 |
| 后置条件 | 返回该时间段内的所有收入项 |
| Finance. inquiryExpense | 语法 | Public resultMessage inquiryExPense(string time1, string time2) |
| 前置条件 | 输入要查询的时间段 |
| 后置条件 | 返回列表中在该时间段内的所有支出项列表 |
| Finance. Init | 语法 | Public ResultMessage init(string operator) |
| 前置条件 | 启动一次账户备份 |
| 后置条件 | 备份现在系统内的所有账户，并新建一个空账 |
| Finance. addAccount | 语法 | Public Boolean addAccount( BankAccountVO vo) |
| 前置条件 | 启动一次新建账户操作，并从键盘输入新建账户的信息 |
| 后置条件 | 新建一个银行账户 |
| Finance. deleteAccount | 语法 | Public Boolean deleteAccount( BankAccountPO po) |
| 前置条件 | 选择一个账户并进行删除 |
| 后置条件 | 删除一个银行账户 |
| Finance. changeAccount | 语法 | Public Boolean changeAccount( string name) |
| 前置条件 | 输入修改的账户名 |
| 后置条件 | 对账户名进行修改 |
| Finance. inquiryAccount | 语法 | Public resultMessage inquiryAccount( BankAccountPO po) |
| 前置条件 | 选择一个银行账户进行查找 |
| 后置条件 | 返回一个银行账户信息(vo) |
| Finance. inquiryAccountByBank | 语法 | Public resultMessage inquiryAccountByBank( string bankName) |
| 前置条件 | 选择一个银行名称 |
| 后置条件 | 返回该公司在该银行名下的所有账户 |
| Finance. setFreight | 语法 | Public resultMessage setFreight(FinanceStrategyVO vo) |
| 前置条件 | 对运费制定方法进行更改 |
| 后置条件 | 更改运费制定方法 |
| Finance. getConditionForm | 语法 | Public resultMessage getConditionForm( string time1, string time2) |
| 前置条件 | 选择生成经营情况表并输入对应时间 |
| 后置条件 | 系统生成经营情况表 |
| Finance. getIncomeForm | 语法 | Public resultMessage getIncomeForm(string time1,string time2) |
| 前置条件 | 选择生成成本收益表，并输入计算时间段 |
| 后置条件 | 系统生成成本收益表 |
| Finance. endFinanceOpt | 语法 | Public void endFinanceOpt() |
| 前置条件 | 结束所有财务操作 |
| 后置条件 | 系统持久化更新所有涉及领域的对象 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory.getFinanceDatabase | 得到finance数据库的服务的引用 | |
| FinanceDataService.find( string id) | 根据Id返回对应的FinancePO对象 | |
| FinanceDataService.findbyHall(string hall) | 按照营业厅的名字返回所属的所有IncomePO | |
| FinanceDataService.findIncome(string time1, string time2) | 按照输入的时间段查找并返回该段时间内所有的IncomePO. | |
| FinanceDataService.findExpense(string time1, string time2) | 按照输入的时间段查找并返回该段时间内所有的ExpensePO | |
| FinanceDataService.findHallIncome(string time1, string time2, string hall) | 按照输入的时间段和营业厅名字查找该时间段内属于该营业厅的所有IncomePO | |
| FinanceDataService.insert(FinancePo po) | 在数据库中增加一个Financepo记录 | |
| FinanceDataService.delete（FinancePO po） | 在数据库中删除一个FinancePO记录 | |
| FinanceDataService.update(FinancePO po) | 在数据库中更新这个FianncePO记录 | |

#### 5.3.2.6Memberbl模块的接口规范

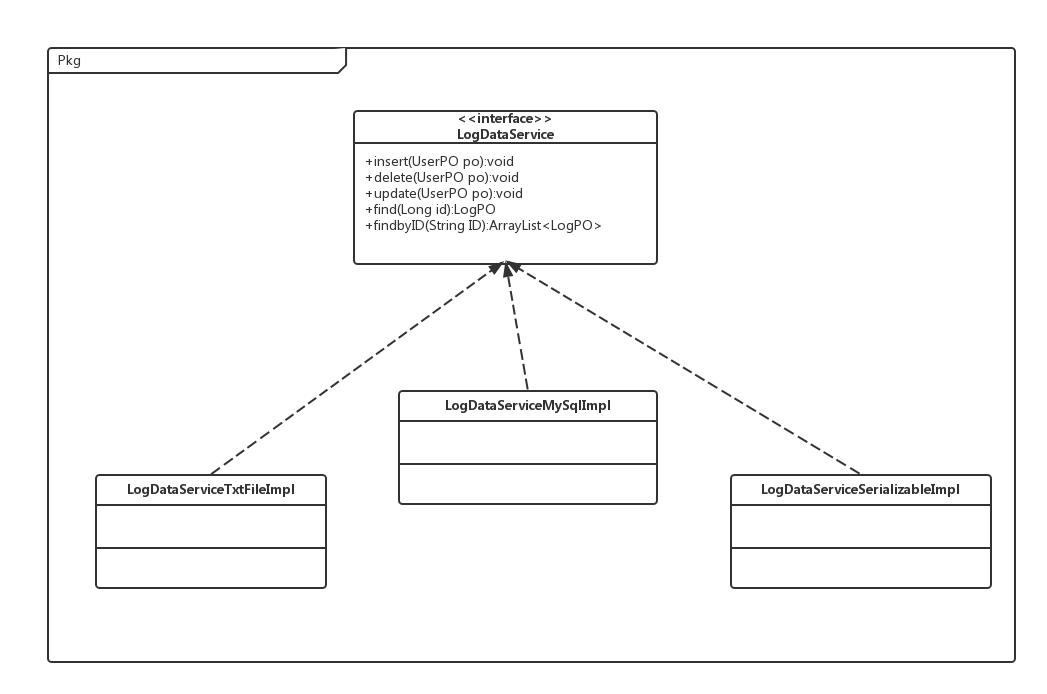
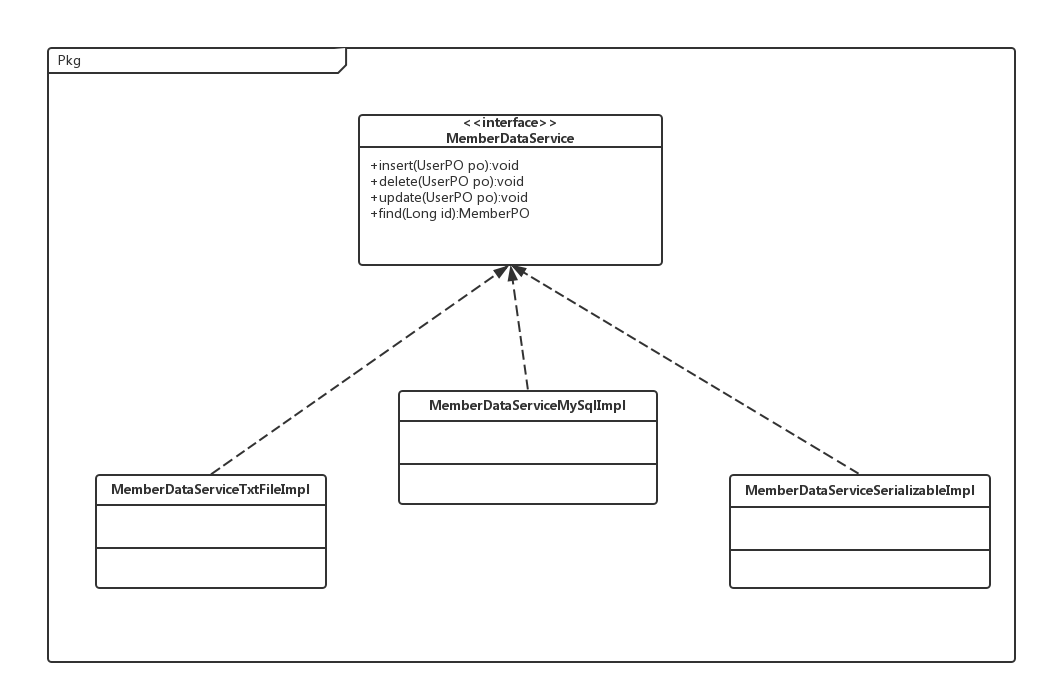
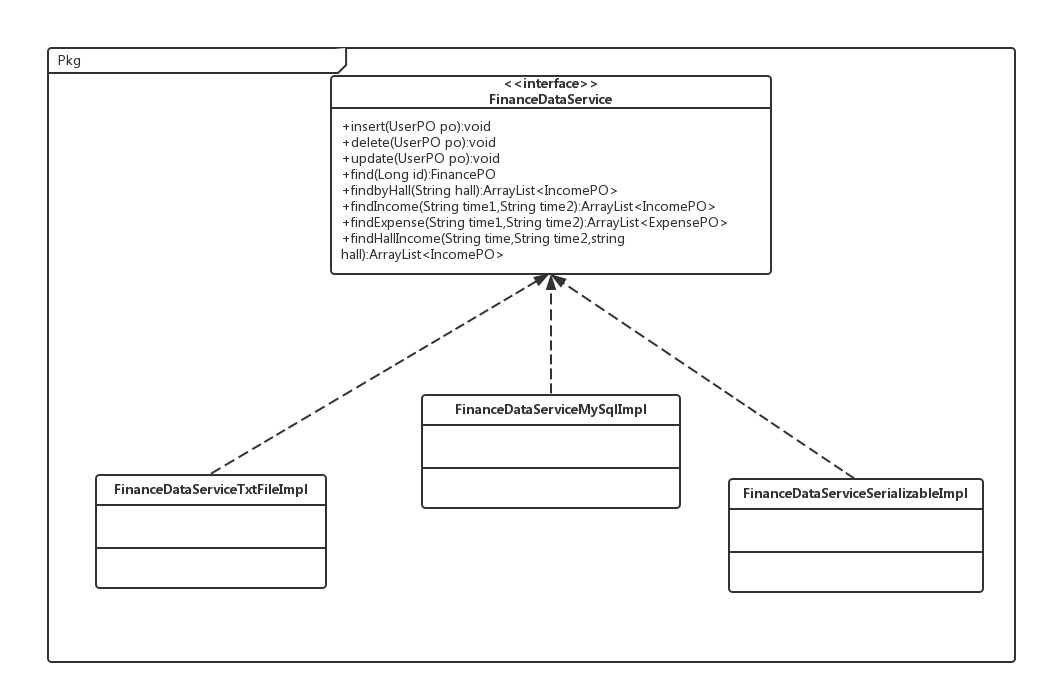
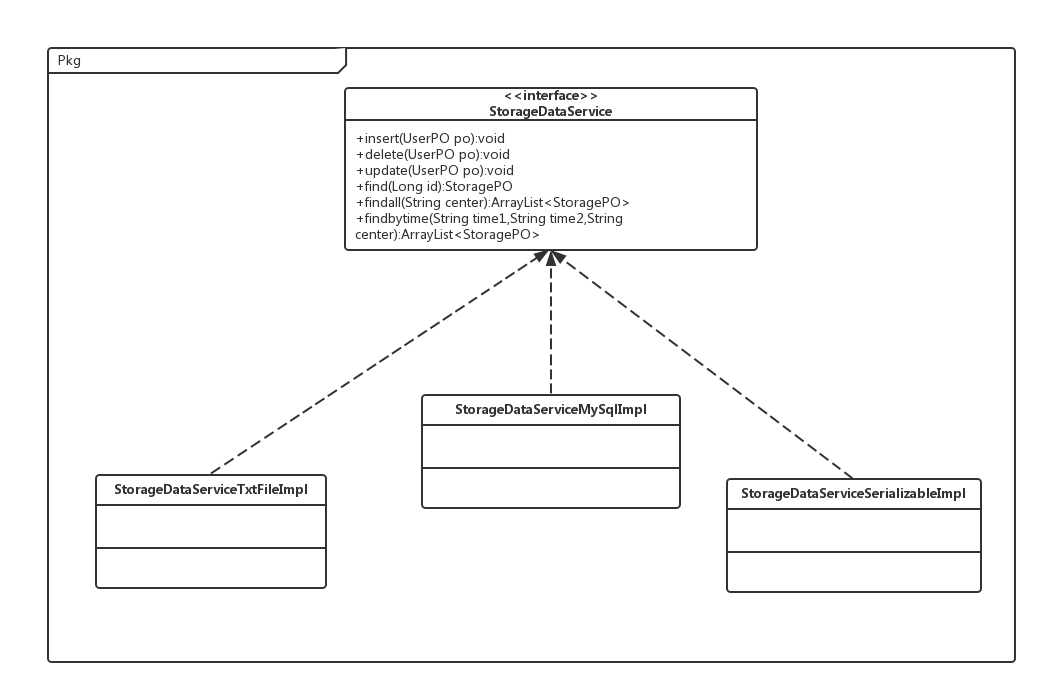
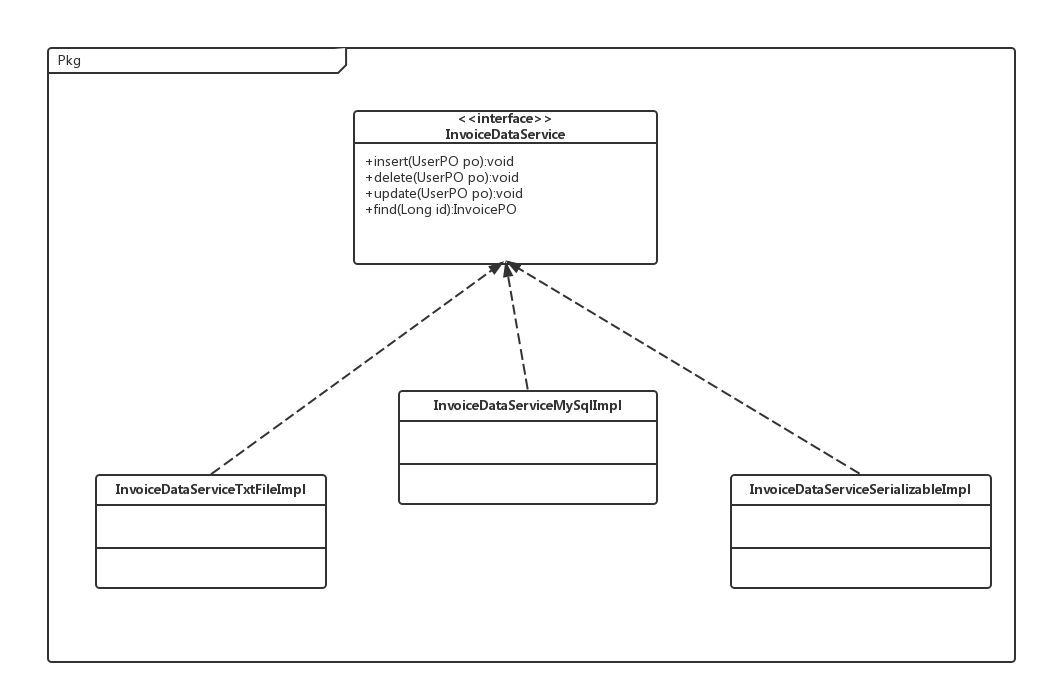
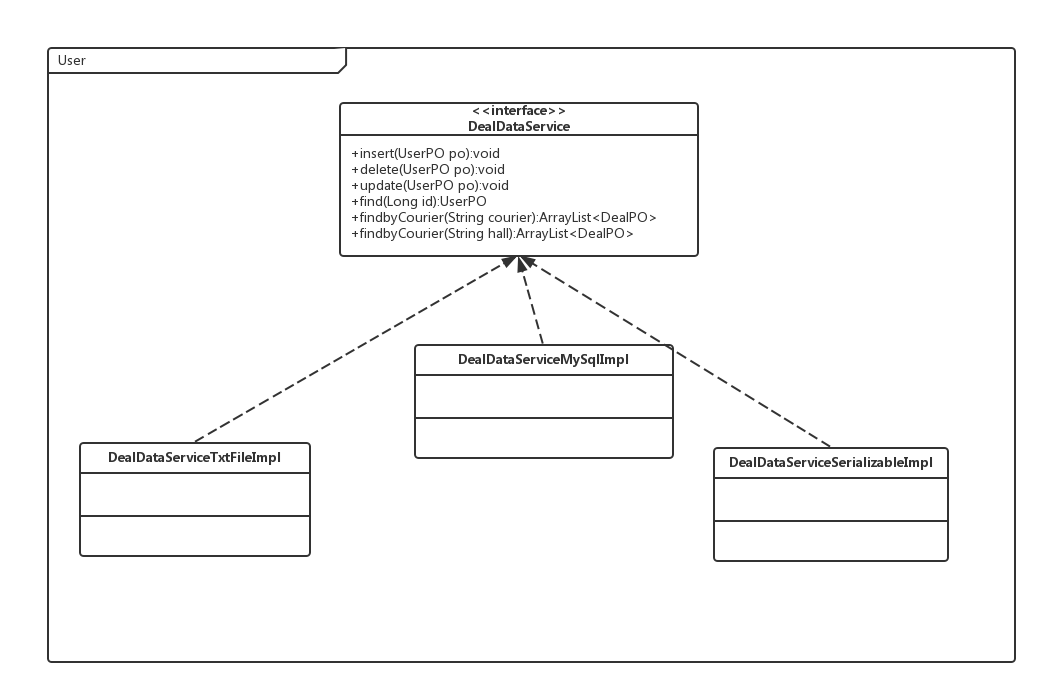
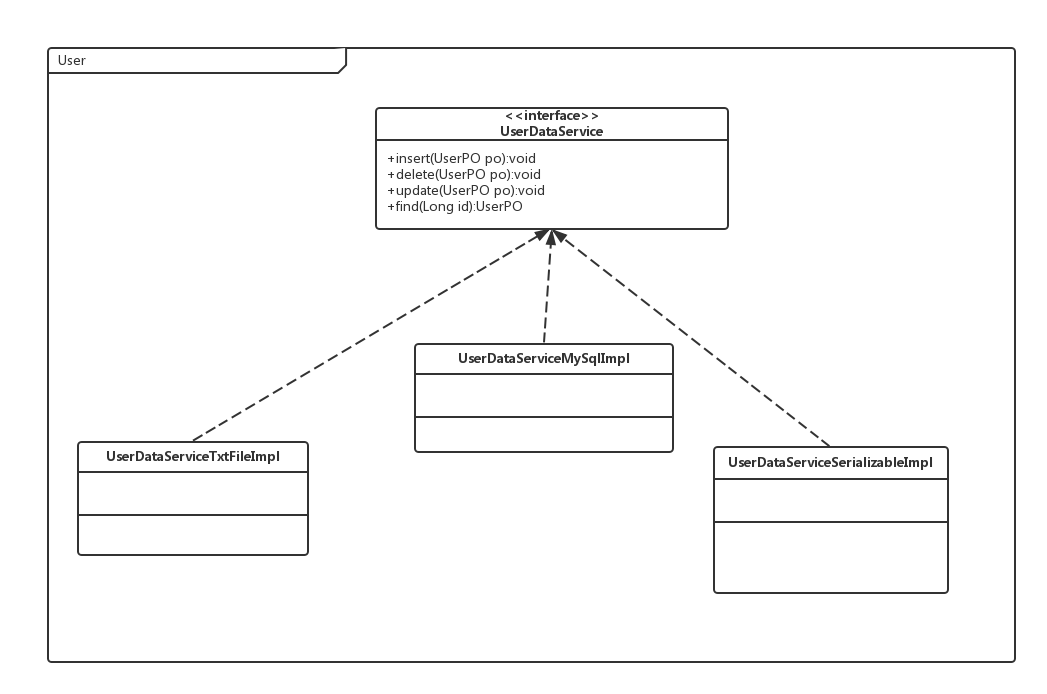
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Member. find | 语法 | Public resultMessage find( stringID) |
| 前置条件 | 输入id |
| 后置条件 | 返回该员工/司机/车辆的详细信息（memberPO） |
| Member. addStaff | 语法 | Public Boolean addStaff( StaffVO vo) |
| 前置条件 | 输入员工基本信息 |
| 后置条件 | 系统新建一个员工 |
| Member. deleteStaff | 语法 | Public void deleteStaff( staffPO po) |
| 前置条件 | 选择删除员工 |
| 后置条件 | 系统删除该员工 |
| Member. changeStaff | 语法 | Public resultMessage changeStaff( staffVo vo) |
| 前置条件 | 选择更改员工信息，输入更改后的员工vo |
| 后置条件 | 系统对该员工的信息进行更改 |
| Member. addCar | 语法 | Public boolean addCar(carVO vo) |
| 前置条件 | 选择增加车辆信息，输入车辆基本信息 |
| 后置条件 | 系统新建一个车辆信息 |
| Member. deleteCar | 语法 | Public void deleteCar(CarVO vo) |
| 前置条件 | 选择删除车辆信息 |
| 后置条件 | 系统删除该车辆信息 |
| Member. changeCar | 语法 | Public resultMessage changeCar( carVO vo) |
| 前置条件 | 选择修改车辆信息 |
| 后置条件 | 系统修改该车辆信息 |
| Member. addDriver | 语法 | Public Boolean addDriver( DriverVO vo) |
| 前置条件 | 输入司机基本信息 |
| 后置条件 | 系统新建一个司机 |
| Member. deleteDriver | 语法 | Public void deleteDriver(DriverPO po) |
| 前置条件 | 选择删除司机信息 |
| 后置条件 | 系统删除该司机信息 |
| Member. changeDriver | 语法 | Public resultMessage changeDriver(DriverVo vo) |
| 前置条件 | 选择修改司机信息 |
| 后置条件 | 系统修改该司机信息 |
| Member. setWages | 语法 | Public resultMessage setWages( string job, int wages, int cycles) |
| 前置条件 | 选择修改薪水策略，输入岗位，更改薪水及结算周期 |
| 后置条件 | 系统修改该岗位的薪水策略 |
| Member. endMemberOpt | 语法 | Public void endMemberOpt() |
| 前置条件 | 已完成操作 |
| 后置条件 | 系统持久化更新涉及领域的对象信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory.getMemberDatabase | 获得member数据库的引用 | |
| memberDataService. Find( string id) | 根据输入的id在数据库中查找MemberPO对象 | |
| MemberDataService. Insert(MemberPO po) | 在数据库中插入MemberPO对象 | |
| MemberDataService. Delete( MemberPO po) | 在数据库中删除MemberPO对象 | |
| MemberDataService. Update(MemberPO po) | 在数据库中更新MemberPO对象 | |
|  |  | |
|  |  | |

#### 5.3.2.7Logbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Log.buildLog | 语法 | public boolean buildLog(Logvo vo) |
| 前置条件 | 有需要记录的系统操作发生 |
| 后置条件 | 生成本次操作的系统日志 |
| Log.inquiry | 语法 | Public LogVO inquiry(String id) |
| 前置条件 | 输入某次操作编号 |
| 后置条件 | 显示该次操作详情 |
| Log.inquirybyID | 语法 | Public ArrayList<LogVO> inquirybyID() |
| 前置条件 | 输入员工编号 |
| 后置条件 | 显示该员工操作 |
| Log.inquiryAll | 语法 | Public ArrayList<LogVO> inquiryAll() |
| 前置条件 | 选择查看所有系统日志 |
| 后置条件 | 显示该账户所有系统日志 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory.getLogDatabase | 获得Log数据库的引用 | |
| LogDataService. Find( string id) | 根据输入的id在数据库中查找Log对象 | |
| LogDataService. Insert(MemberPO po) | 在数据库中插入LogPO对象 | |
| MemberDataService. Update(MemberPO po) | 在数据库中更新MemberPO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、差。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下图所示。



### 5.4.1数据层模块的职责

数据层模块的职责如下表所示。

#### 5.4.1.1

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserDataService | 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 |
| UserDataServiceTxtFileImpl | 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 |
| UserDataServiceSeralizableFileImpl | 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 |
| UserDataServiceMySqlImpl | 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 |

#### 5.4.1.2

|  |  |
| --- | --- |
| 模块 | 职责 |
| DealDataService | 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 |
| DealDataServiceTxtFileImpl | 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 |
| DealDataServiceSeralizableFileImpl | 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 |
| DealDataServiceMySqlImpl | 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 |

#### 5.4.1.3

|  |  |
| --- | --- |
| 模块 | 职责 |
| InvoiceDataService | 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 |
| InvoiceDataServiceTxtFileImpl | 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 |
| InvoiceDataServiceSeralizableFileImpl | 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 |
| InvoiceDataServiceMySqlImpl | 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 |

#### 5.4.1.4

|  |  |
| --- | --- |
| 模块 | 职责 |
| StorageDataService | 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 |
| StorageDataServiceTxtFileImpl | 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 |
| StorageDataServiceSeralizableFileImpl | 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 |
| StorageDataServiceMySqlImpl | 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 |

#### 5.4.1.5

|  |  |
| --- | --- |
| 模块 | 职责 |
| FinanceDataService | 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 |
| FinanceDataServiceTxtFileImpl | 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 |
| FinanceDataServiceSeralizableFileImpl | 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 |
| FinanceDataServiceMySqlImpl | 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 |

#### 5.4.1.6

|  |  |
| --- | --- |
| 模块 | 职责 |
| MemberDataService | 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 |
| MemberServiceTxtFileImpl | 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 |
| MemberDataServiceSeralizableFileImpl | 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 |
| MemberDataServiceMySqlImpl | 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 |

#### 5.4.1.7

|  |  |
| --- | --- |
| 模块 | 职责 |
| LogDataService | 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 |
| LogDataServiceTxtFileImpl | 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 |
| LogDataServiceSeralizableFileImpl | 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 |
| LogDataServiceMySqlImpl | 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

数据层模块的接口规范如下表所示。

#### 5.4.2.1 UserDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService.find | 语法 | Public UserPO find(Long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的UserPO结果 |
| UserDataService.findall | 语法 | Public ArrayList<UserPo>findall() throws RemoteExcepiton |
| 前置条件 | 无 |
| 后置条件 | 返回所有UserPO |
| UserDataService.insert | 语法 | Public void insert(UserPO po) throws RemoteException |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| UserDataService.delete | 语法 | Public void delete(UserPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除这个po |
| UserDataService.update | 语法 | Public void update(UserPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新这个po |
| UserDataService.init | 语法 | Public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| UserDataService.finish | 语法 | Public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

#### 5.4.2.2 DealDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DealDataService.find | 语法 | Public DealPO find(Long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的DealPO结果 |
| DealDataService.findall | 语法 | Public ArrayList<DealPO> findall() throws RemoteExcepiton |
| 前置条件 | 无 |
| 后置条件 | 返回数据库中所有DealPO |
| DealDataService.findbyCourier | 语法 | Public ArrayList<DealPO> findbyCourier(String courier)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按揽件员（快递员）姓名查找所属的DealPO结果 |
| DealDataService.findbyHall | 语法 | Public ArrayList<DealPO> findbyHall(String hall>throws RemoteExceptionw |
| 前置条件 | 无 |
| 后置条件 | 查找并返回该营业厅所有的DealPO |
| DealDataService.insert | 语法 | Public void insert(DealPO po) throws RemoteException |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| DealDataService.delete | 语法 | Public void delete(DealPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除这个po |
| DealDataService.update | 语法 | Public void update(DealPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新这个po |
| DealDataService.init | 语法 | Public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| DealDataService.finish | 语法 | Public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

5.4.2.3InvoiceDataService(ArrivalListPO(营业厅到达单)，SendingListPO（营业厅派件单）, IncomeListPO(收款单),LoadingListPO(装车单),receivalListPO(中转中心接受单)，,transferListPO(中转中心中转单),Store\_InListPO(入库单)，Store\_OutListPO(出库单),均继承自InvoicePO类。

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InvoiceDataService.find | 语法 | Public InvoicePO find(Long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回对应的单据 |
| InvoiceDataService.insert | 语法 | Public void insert(Invoice po) throws RemoteException |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| InvoiceDataService.delete | 语法 | Public void delete(Invoice po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除这个po |
| InvoiceDataService.update | 语法 | Public void update(Invoicepo) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新这个po |
| InvoiceDataService.init | 语法 | Public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| InvoiceDataService.finish | 语法 | Public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

#### 5.4.2.4 StorageDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageDataService.find | 语法 | Public StoragePO find(Long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的StoragePO结果 |
| StorageDataService.findall | 语法 | Public ArrayList<StoragePO> (String center)findall throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回该中转中心当前时间点时所有的库存记录（状态为入库的）。 |
| StorageDataService.findbytime | 语法 | Public ArrayList<StoragePO>findbytime(String time1,String time2,String center)throws RemoteExcepition |
| 前置条件 | 无 |
| 后置条件 | 返回该中转中心输入的时间段内所有的出入库记录 |
| StorageDataService.insert | 语法 | Public void insert(StoragePO po) throws RemoteException |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| StorageDataService.delete | 语法 | Public void delete(StoragePO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除这个po |
| StorageDataService.update | 语法 | Public void update(Storage PO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新这个po |
| StorageDataService.init | 语法 | Public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| StorageDataService.finish | 语法 | Public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

5.4.2.5 FinanceDataService (BankAccountPO、 ExpensePO和IncomePO、FinanceStrategyPO)继承自FinancePO)

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinanceDataService.find | 语法 | Public FinancePO find(Long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的po结果 |
| FinanceDataService.findbyHall | 语法 | Public ArrayList<IncomePO>findbyHall(String hall) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按照营业厅的名字查找所属的所有IncomePO |
| FinanceDataService.findIncome | 语法 | Public ArrayList<IncomePO> findIncome(String time1,String time2) throws RemoteExcepiton |
| 前置条件 | 无 |
| 语法 | 按照输入的时间段查找并返回该段时间内所有的IncomePO. |
| FinanceDataService.findExpense | 语法 | Public ArrayList<ExpensePO> findExpense(String time1,String time2) throws RemoteExcepiton |
| 前置条件 | 无 |
| 后置条件 | 按照输入的时间段查找并返回该段时间内所有的ExpensePO |
| FinanceDataService.findHallIncome | 语法 | Public ArrayList<IncomePO>findHallIncome(String time1,String time2,String hall) throws RemoteExcepiton |
| 前置条件 | 无 |
| 后置条件 | 按照输入的时间段和营业厅名字查找该时间段内属于该营业厅的所有IncomePO |
| FinanceDataService.insert | 语法 | Public void insert(FinancePO po) throws RemoteException |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| FinanceDataService.delete | 语法 | Public void delete(FinancePO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除这个po |
| FinanceDataService.update | 语法 | Public void update(FinancePO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新这个po |
| FinanceDataService.init | 语法 | Public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| FinanceDataService.finish | 语法 | Public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

5.4.2.6 MemberDataService (StaffPO、CarPO、DriverPO均继承自MemberPO)

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MemberDataService.find | 语法 | Public MemberPO find(Long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的po结果 |
| MemberDataService.findall | 语法 | Public ArrayList<MemberPO> throws RemoteExcepiton |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回响应的po结果 |
| MemberDataService.insert | 语法 | Public void insert(MemberPO po) throws RemoteException |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| MemberDataService.delete | 语法 | Public void delete(MemberPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除这个po |
| MemberDataService.update | 语法 | Public void update(MemberPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新这个po |
| MemberDataService.init | 语法 | Public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| MemberDataService.finish | 语法 | Public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

#### 5.4.2.7 LogDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LogDataService.find | 语法 | Public LogPO find(Long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的po结果 |
| LogDataService.findall | 语法 | Public ArrayList<LogPO> findall() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有的LogPO |
| LogDataService.findbyID | 语法 | Public ArrayList<LogPO> findbyID(String ID)throws RemoteException |
| 前置条件 | 无 |
|  | 根据操作者的ID找出所对应的LogPO并返回 |
| LogDataService.init | 语法 | Public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| LogDataService.finish | 语法 | Public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |
| LogDataService.insert | 语法 | Public void insert(LogPO) throws RemoteException |
| 前置条件 | 产生系统操作记录 |
| 后置条件 | 系统日志更新 |

# 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，在此作介绍。

●UserPO类包含用户的用户名、用户类别、用户密码、用户权限。

●DealPO类包含订单的编号、订单的寄件人信息、收件人信息、寄件时间、邮件类别、货物信息、包装方式、邮费、揽件员、物流信息、所属营业厅信息。

●InvoicePO类包含到达单：到达单编号、中转单编号（中转单）、到达日期、到达状态、出发地；派件单：到达信息、托运订单号、派件员；收款单：收款日期、金额、收费快递员、订单号；装车单：装车单编号、订单号、装车日期、营业厅编号、车辆代号、到达地、监装员、押运员、运输成本、运送方式；中转单：装车日期、中转单编号、航班/汽运/火车编号、到达地、货柜号、监装员、托运单号、运费；中转中心接收单：中转中心编号、到达时间、中转单编号、出发地、货物到达状态；中转中心入库单：库存项信息；中转中心出库单：库存项信息；

●StoragePO类包含库存项编号、库存分区、库存位置、订单编号、入库时间、出库时间、商品状态、所属中转中心信息。

●FinanceIncomePO类包含收入编号、收入时间、收入金额、收入营业厅、财收操作人员。

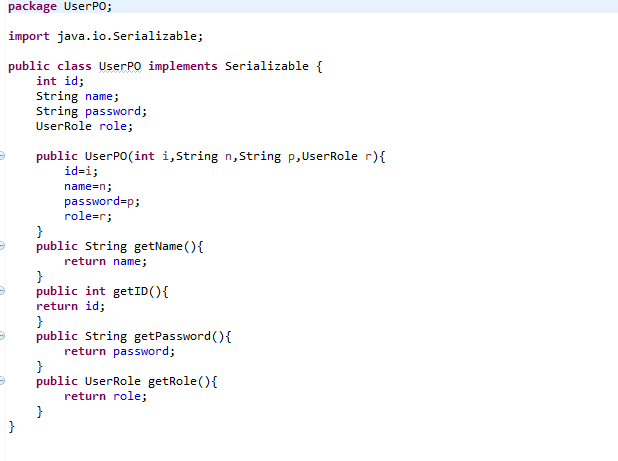
●FinanceExpensePO类包括支出编号、支出类别、支出时间、支出金额、支出人员、操作财务人员。

●FinanceBankAccountPO类包括银行账户名、银行账户金额、开户银行信息。

●FinanceStrategyPO类包括当前的邮费制定方法。

●MemberPO类包含工作人员：人员编号、姓名、性别、职务、身份证编号、薪水、联系方式。车辆信息：车辆代号（城市编号（电话号码区号南京025）+营业厅编号（000三位数字）+000三位数字）、车牌号（苏A 00000））、服役时间、车辆图片。司机信息：司机编号（城市编号（电话号码区号南京025）+营业厅编号（000三位数字）+000三位数字）、姓名、出生日期、身份证号、手机、性别、行驶证期限。

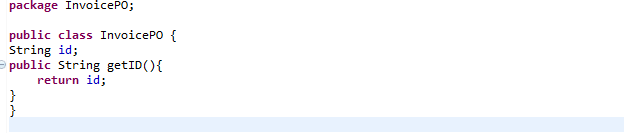
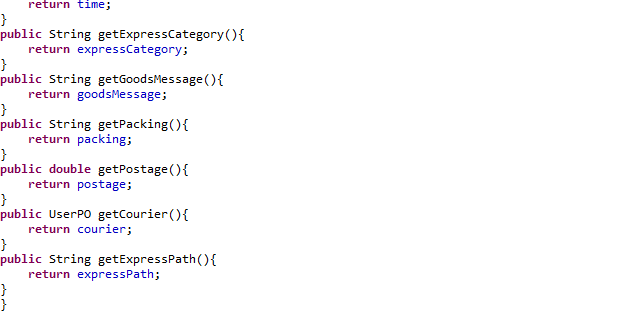
●LogPO类包含系统操作编号、系统操作类别、系统操作人员、系统操作时间。



持久化用户对象UserPO的定义

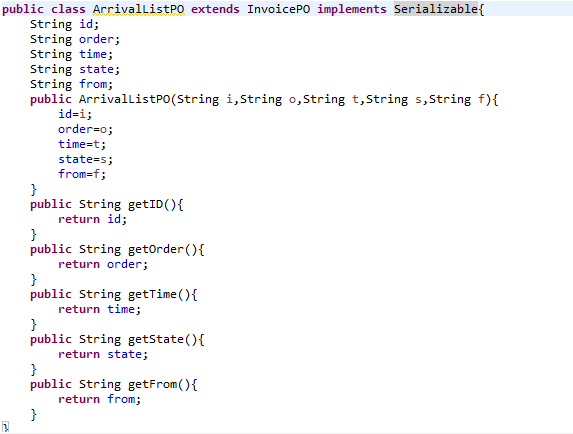


持久化订单对象dealPO的定义

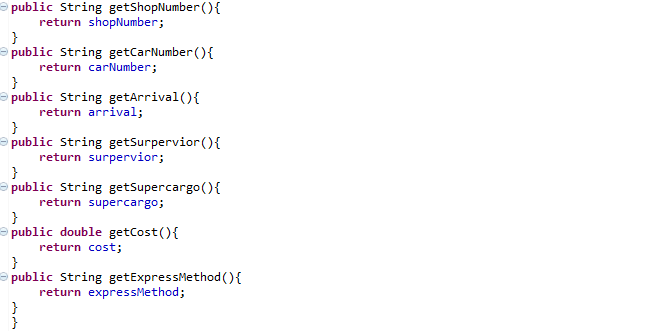
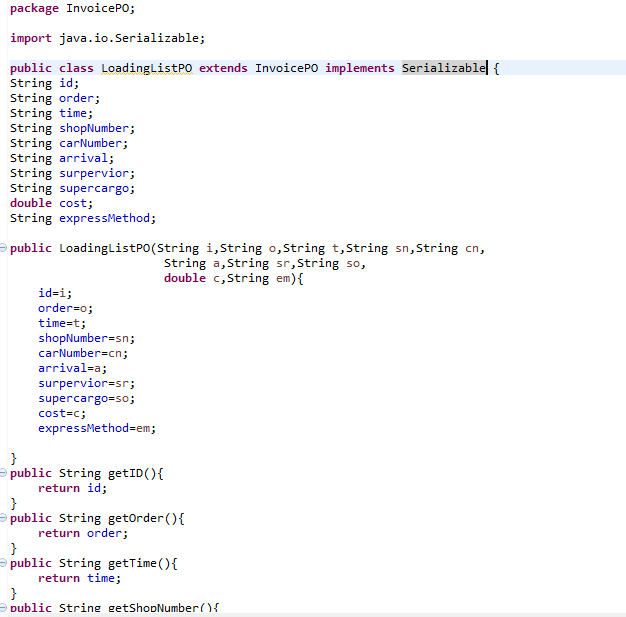
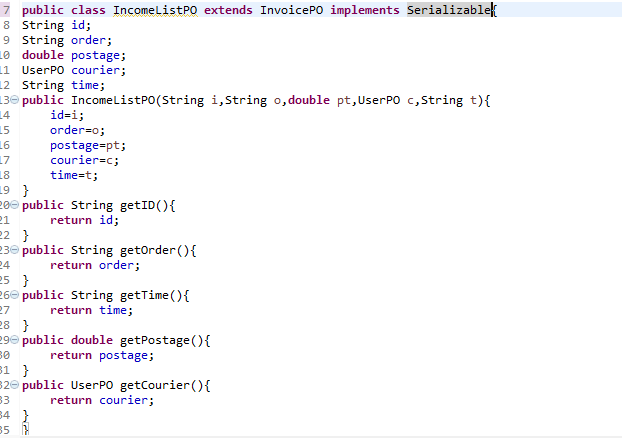


持久化单据父类InvoicePO的定义

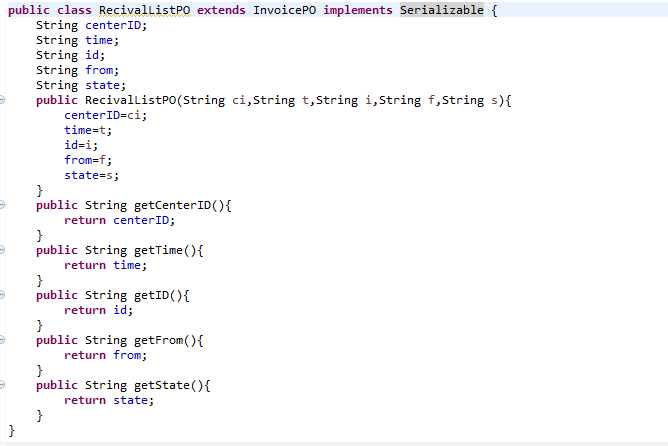
持久化单据-到达单对象ArrivalListPO的定义



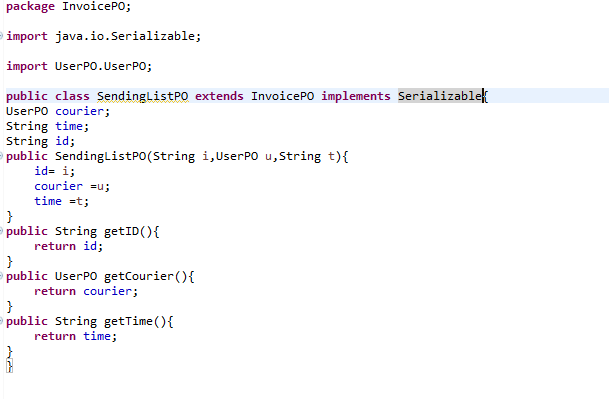
持久化单据-收款单对象IncomeListPO的定义



持久化单据-装车单LoadingListPO的定义



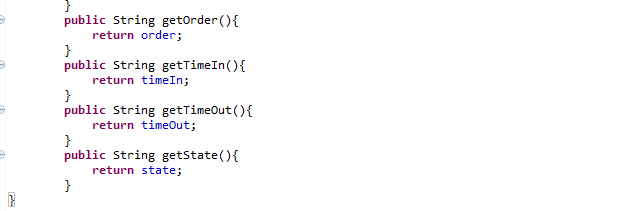
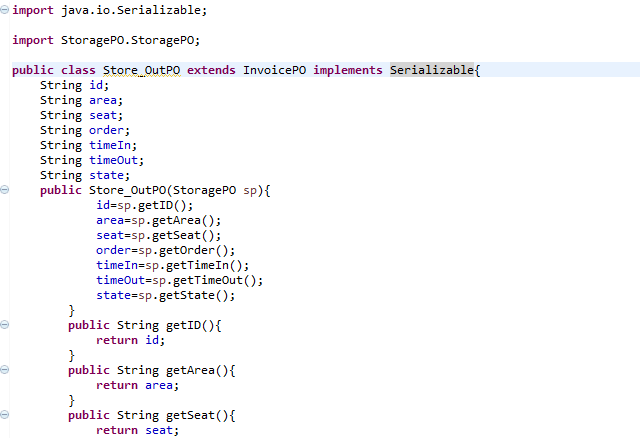
持久化单据-中转接收单RecivalListPO的定义



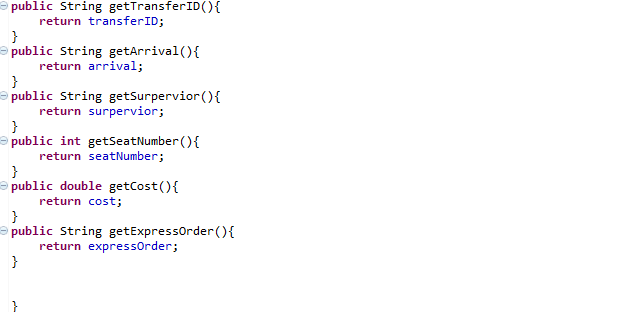
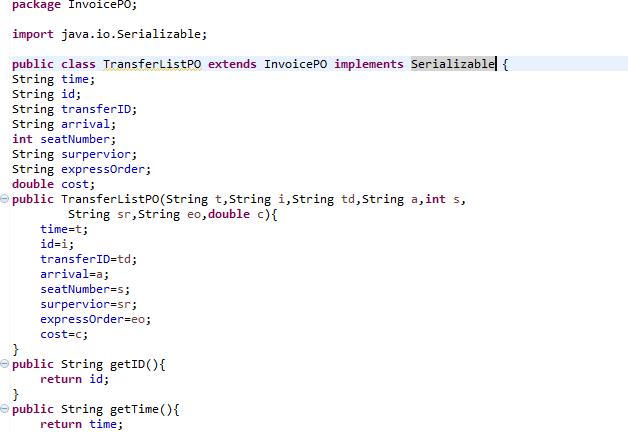
持久化单据-派送单SendingListPO的定义



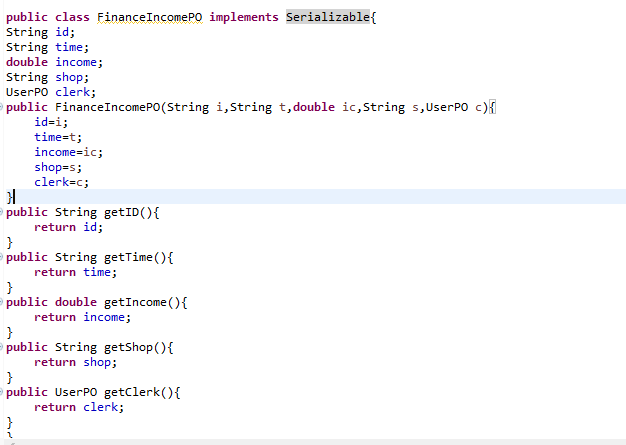
持久化单据-中转中心入库单Store\_InListPO的定义



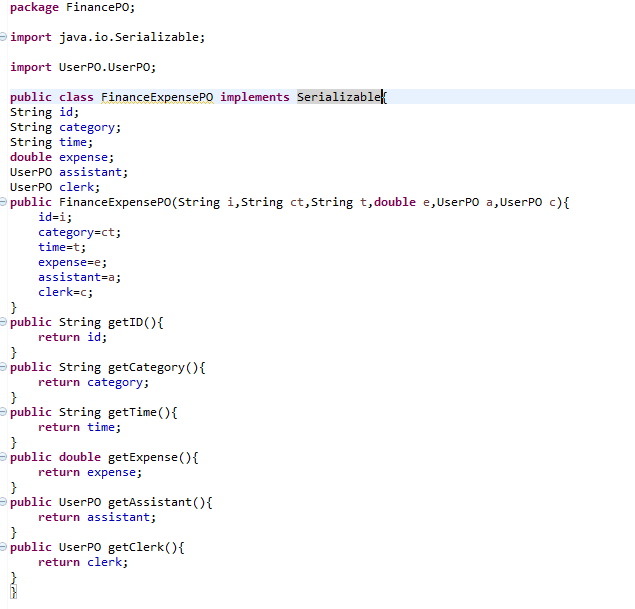
持久化单据-中转中心出库单对象 Store\_OutListPO的定义



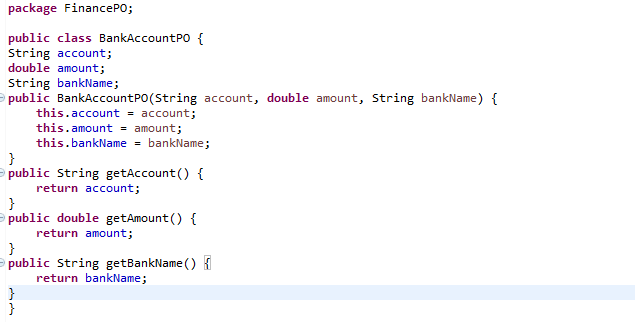
持久化单据-中转中心中转单对象 TransferListPO的定义



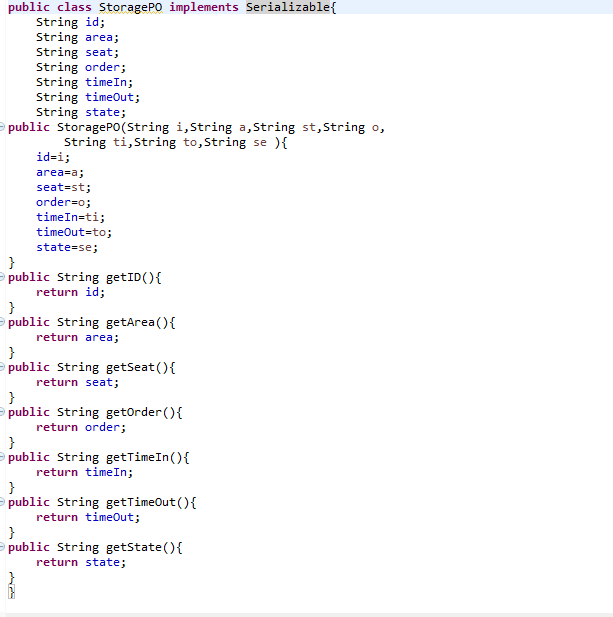
持久化财政收入对象FinanceIncomePO的定义



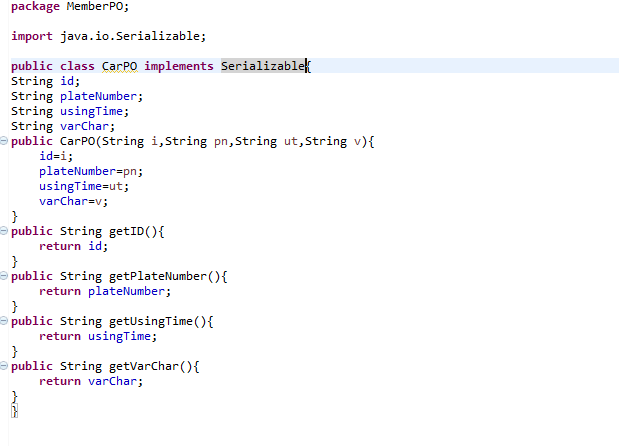
持久化财政支出对象FinanceExpensePO的定义



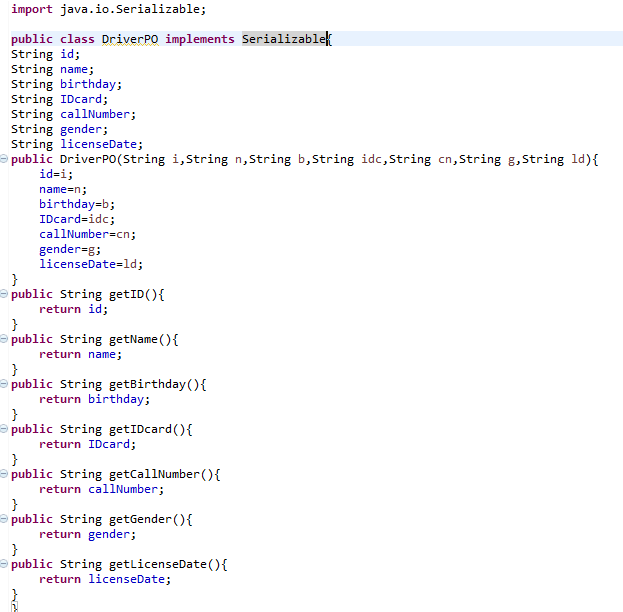
持久化银行账户对象FinanceBankAccountPO的定义



持久化库存对象StoragePO的定义



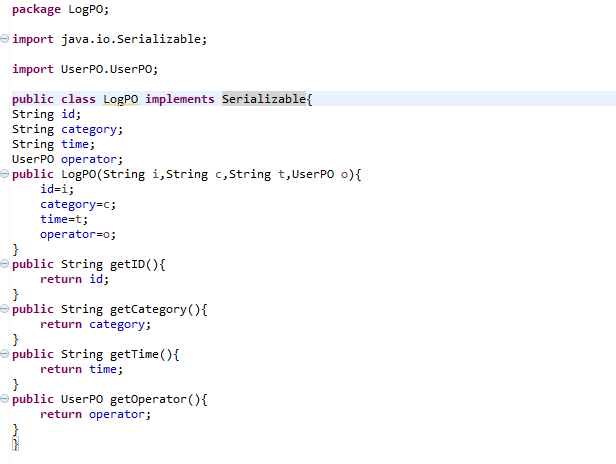
持久化车辆信息对象CarPO的定义



持久化司机对象DriverPO的定义



持久化员工对象StaffPO的定义



持久化系统日志对象LogPO的定义

## 6.2 Txt持久化格式

Txt数据保持格式以Storage.txt为例。每行分别对应库存编号、库存分区、库存位置、订单编号、入库时间、出库时间、商品状态。中间用’:’隔开，如下所示：

000001:航运区:R3L4S5:00000000001:2015-10-20-20-18:0-0-0-0-0:库存中

000002:货运区:R35L1S42:00000000002:2015-10-18-08-42:2015-10-21-15-20:出库

## 6.3 数据库表

数据库表中包含User表、Invoice表、Storage表、StorageList表、FinanceIncome表、FinanceIncomeList表、FinanceExpense表、FinanceExpenseList、Member表、Log表。