快递管理系统 MSE(Management System of Expressage)

体系结构描述

V1.0

完整版

一组

成员：吕德超、李昊朔、阙俊杰、罗铉斌

2015-10-21

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 全体人员 | 2015-10-21 | 最初草稿（不完整） | V1.0 草稿 |
| 全体人员 | 2015-10-12 | 补充完整 | V1.0 完整版 |
| 李昊朔 | 2015-10-12 | 相关功能需求完善 | V1.1 测试版 |
| 吕德超 | 2015-10-12 | UI和功能需求修改 | V1.2 测试版 |

# 引言

## 编制目的

本报告详细完成对快递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。

## 词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| MSE | 快递管理系统 |  |
| checklist | 订单查询 | 订单查询 |
| listin | 单据输入 | 订单输入、收件人信息输入 |
| transfer | 中转接收 | 接收与派件、中转接收、录入中转单 |
| load | 装运管理 | 车辆装车管理、装运管理 |
| vehicle | 车辆管理 | 司机信息管理、车辆信息管理 |
| finance | 财务管理 | 收款单管理，结算管理，成本管理，查看成本收益表，统计报表，查看经营情况 |
| store | 仓库管理 | 入库管理，出库管理，库存管理 |
| bill | 期初建账 | 期初建账 |
| account | 账户管理 | 账户管理，帐户查询 |
| staff | 员工管理 | 人员机构管理，用户信息管理 |
| policy | 策略管理 | 策略管理 |
| approve | 审批单据 | 审批单据 |
| user | 用户 |  |
| order | 订单 |  |
| receiver | 收件人信息 |  |
| loadorder | 装车单 |  |
| driver | 司机信息 |  |
| arriveorder | 到达单 | 中转中心到达单 |
| receiveorder | 接收单 | 营业厅接收单 |
| sendorder | 派件单 |  |
| payeeorder | 收款单 |  |
| changeorder | 中转单 |  |
| storeinorder | 入库单 |  |
| storeoutorder | 出库单 |  |
| payorder | 付款单 |  |
| startinfo | 期初信息 |  |
| salary | 薪水信息 |  |
| constant | 常量信息 |  |

## 参考资料

1. IEEE标准
2. 快递管理管理系统MSE用例文档
3. 快递管理系统MSE结构逻辑设计文档

# 产品描述

×××民营物流企业，希望通过对城市网点建设，扩大快运服务通达的片区。企业在每个市设立中转中心、营业厅、快递员三个层级快递模式。快递员收件送往营业厅，同市间快递在营业厅通过汽车运输，跨市之间快递一律由营业厅发往中转中心，通过飞机、火车、汽车流通发往收件人所属中转中心。再由收件人所属中转中心发向所属的营业厅。再有快递员在营业厅提件，送达给收件人。随着现公司规模扩大，企业业务量、办公场所、员工数都发生增长，为适应新的环境，提高工作效率和用户满意度， 该公司聘请Q公司开发一套快递物流系统系统。

# 逻辑视角

快递管理系统中，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实际方案如图1和图2所示。

presentation

businesslogic

data

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

# 组合视角

## 开发包图

|  |  |
| --- | --- |
| **开发（物理）包** | **依赖的其他开发包** |
| mainui | checklistui, listinui, transferui, loadui, vehicleui, financeui, storeui, billui, accountui, staffui, policyui, approveui, vo |
| checklistui | checklistblservice, 界面类库包, vo |
| checklistblservice |  |
| checklistbl | checklistblservice, checklistdataservice, po, listinbl |
| checklistdataservice | Java RMI, po |
| checklistdata | Java RMI, po, databaseutility |
| listinui | listinblservice, 界面类库包, vo |
| listinblservice |  |
| listinbl | listinblservice, listindataservice, po, policybl |
| listindataservice | Java RMI, po |
| listindata | Java RMI, po, databaseutility |
| transferui | transferblservice, 界面类库包, vo |
| transferblservice |  |
| transferbl | transferblservice, transferdataservice, po, listinbl, vehiclebl |
| transferdataservice | Java RMI, po |
| transferdata | Java RMI, po, databaseutility |
| loadui | loadblservice, 界面类库包, vo |
| loadblservice |  |
| loadbl | loadblservice, loaddataservice, po, listinbl, vehiclebl |
| loaddataservice | Java RMI, po |
| loaddata | Java RMI, po, databaseutility |
| vehicleui | vehicleblservice, 界面类库包, vo |
| vehicleblservice |  |
| vehiclebl | vehicleblservice, vehicledataservice, po |
| vehicledataservice | Java RMI, po |
| vehicledata | Java RMI, po, databaseutility |
| financeui | financeblservice, 界面类库包, vo |
| financeblservice |  |
| financebl | financeblservice, financedataservice, po, listinbl, transferbl, loadbl, vehiclebl, staffbl |
| financedataservice | Java RMI, po |
| financedata | Java RMI, po, databaseutility |
| storeui | storeblservice, 界面类库包, vo |
| storeblservice |  |
| storebl | storeblservice, storedataservice, po, listinbl |
| storedataservice | Java RMI, po |
| storedata | Java RMI, po, databaseutility |
| billui | billblservice, 界面类库包, vo |
| billblservice |  |
| billbl | billblservice, billdataservice, po, vehiclebl, storebl, accountbl, staffbl |
| billdataservice | Java RMI, po |
| billdata | Java RMI, po, databaseutility |
| accountui | accountblservice, 界面类库包, vo |
| accountblservice |  |
| accountbl | accountblservice, accountdataservice, po, financebl |
| accountdataservice | Java RMI, po |
| accountdata | Java RMI, po, databaseutility |
| staffui | staffblservice, 界面类库包, vo |
| staffblservice |  |
| staffbl | staffblservice, staffdataservice, po, policybl |
| staffdataservice | Java RMI, po |
| staffdata | Java RMI, po, databaseutility |
| policyui | policyblservice, 界面类库包, vo |
| policyblservice |  |
| policybl | policyblservice, policydataservice, po |
| policydataservice | Java RMI, po |
| policydata | Java RMI, po, databaseutility |
| approveui | approveblservice, 界面类库包, vo |
| approveblservice |  |
| approvebl | approvelblservice, approvedataservice, po, listinbl, transferbl, loadbl, financebl, storebl |
| approvedataservice | Java RMI, po |
| approvedata | Java RMI, po, databaseutility |
| vo |  |
| po |  |
| utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| databaseutility | JDBC |

快递管理系统客户端开发包图和服务器端开发包图如图所示。



图3 快递管理系统客户端开发包图



图4 快递管理系统服务器端开发包图

## 运行时进程

在快递管理系统中，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进行。



图5 进程图

## 物理部署

快递管理系统中客户端构件是放在客户端服务上，服务器端构件是放在服务器端机器上。在客户端节点上，还要不熟RMIStub构件。由于Java RMI 构件属于JDK 6.0的一部分。所以在系统JDK环境已经设置好的情况下，不需要在独立部署。部署图如图6所示。



图6 部署图

# 接口视角

## 模块的职责

|  |
| --- |
| 1. 展示层 |

|  |
| --- |
| 启动模块 |

|  |
| --- |
| 业务逻辑层 |

|  |
| --- |
| 网络模块 |

**客户端模块视图**

|  |
| --- |
| 网络模块 |

|  |
| --- |
| 启动模块 |

|  |
| --- |
| 网络模块 |

**服务器端模块视图**

**客户端各层职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 展示层 | 给予窗口的物流管理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 |

**服务器端各层职责**

|  |  |
| --- | --- |
| **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 |

**层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| Userblservice  Checklistblservice  Listinblservice  Transferblservice  Loadblservice  Vehicleblservice  Financeblservice  Storeblservice  Billblservice  Accountblservice  Staffblservice  Policyblservice  approveblservice | 客户端展示层 | 客户端业务逻辑层 |
| Userdataservice  Checklistdataservice  Listindataservice  Transferdataservice  vehicledataservice  loaddataservice  financedataservice  storedataservice  billdataservice  accountdataservice  staffdataservice  policydataservice  approvedataservice | 客户端业务逻辑层 | 服务器端数据层 |

## 5.2业务逻辑层的分解

根据需求，系统存在77个用户界面。界面跳转如图。



## 5.3

## 5.3.1业务逻辑层模块的职责

**业务逻辑层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| checklistbl | 负责实现订单查询界面所需要的服务 |
| listinbl | 负责实现输入订单和收件人信息界面所需要的服务 |
| transferbl | 负责实现中转接收界面所需要的服务 |
| loadbl | 负责实现装运管理界面所需要的服务 |
| vehiclebl | 负责实现车辆管理界面所需要的服务 |
| financebl | 负责实现财务管理界面所需要的服务 |
| storebl | 负责实现仓库管理界面所需要的服务 |
| billbl | 负责实现期初建账界面所需要的服务 |
| accountbl | 负责实现账户管理界面所需要的服务 |
| Staffbl | 负责实现员工管理界面所需要的服务 |
| policybl | 负责实现策略管理界面所需要的服务 |
| approvebl | 负责实现审批单据界面所需要的服务 |

## 5.3.2业务逻辑层模块的接口规范

## 5.3.2.1登陆用户

**表1 user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返回登录验证结果 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| DatabaseFacory.getUserData | 得到User数据库的服务的引用 | |
| UserDataService.find（String id,String password） | 在数据库中查找相应的 | |

**表2 用户管理用例的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| System.new | 语法 | public ResultMessage new(String name,String key,String limit); |
| 前置条件 | 启动一个新建用户的任务 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| System.updata | 语法 | public ResultMessage updata(String oldname,String newname,Sting key,String limit); |
| 前置条件 | 已选定目标修改的用户，并启动修改用户的任务 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| System.delete | 语法 | public ResultMessage delete(String name); |
| 前置条件 | 已选定目标删除的用户，并启动删除用户的任务 |
| 后置条件 | 删除涉及的领域对象的数据 |
| System.check | 语法 | public ResultMessage check(UserPO po); |
| 前置条件 | 启动一个查看用户信息的任务 |
| 后置条件 | 显示选定用户的信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserDataService.insert(UserPO po) | 插入单一持久化对象 | |
| UserDataService.updata(UserPO po) | 更新单一持久化对象 | |
| UserDataService.delete(UserPO po) | 删除单一持久化对象 | |
| UserDataService.find(String name) | 根据用户名（name）查找单一持久化对象 | |
| DatabaseFactory.getUserDatabase | 得到User数据库的服务的引用 | |

## 5.3.2.2订单查询

**表3订单查询模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Check.getOrder | 语法 | public ResultMessage getOrder(String id) |
| 前置条件 | 启动一个订单查询回合 |
| 后置条件 | 在一个订单查询回合中返回所查询的订单信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| ListinDataService.find(String id) | 根据ID进行查找单一持久化对象 | |

## 5.3.2.3单据输入

**表4 listinbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Listin.addOrder | 语法 | Public ResultMessage addOrder（String courier, String id, int state, String sender,String address1, String position1, String phone1,  String cellphone1, String receiver, String address2,String position2, String phone2, String cellphone2, String count,int weight, int length, int width, int height, String items,int express, int pack, double bill, String time,String transformState） |
| 前置条件 | 启动一个新建订单任务 |
| 后置条件 | 界面显示一份新建的订单 |
| Listin.getTotalMoney | 语法 | Public ResultMessage getTotalMoney（String address1，String address2，int long，int width，int height，enumExpress，enumPackage） |
| 前置条件 | 地址，体积，快递种类和包装种类的选择，均符合输入规范 |
| 后置条件 | 返回此订单输入中需要支付的金额 |
| Listin. getTotalTime | 语法 | Public ResultMessage getTotalTime（String address1，String address2， enumExpress） |
| 前置条件 | 地址，快递种类的选择，均符合输入规范 |
| 后置条件 | 返回此次货运预计到达的时间 |
| Listin. endAddOrder | 语法 | Public void endAddOrder（） |
| 前置条件 | 确认新建订单 |
| 后置条件 | 结束新建订单，持续化更新涉及的领域对象的数据 |
| Listin. addReceiver | 语法 | Public void addReceiver（String courier, String phoneOfcourier, String id, String receiver, String address, String time, String cellphone） |
| 前置条件 | 启动新建一份收件人信息任务 |
| 后置条件 | 查询是否存在相应的订单号，据此判断是否新建收件人信息单 |
| Listin. endAddReceiver | 语法 | Public void endAddReceiver（） |
| 前置条件 | 确认新建收件人信息单 |
| 后置条件 | 结束新建收件人信息单，持续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 | |
| ListinDataService.insert(OrderPO po) | 插入单一持久化对象 | |
| DatabaseFacory.getListinDatabase | 得到Listin数据库的服务的引用 | |
| ListinDataService.insertReceiver(String id) | 根据订单号查找单一持久化对象 | |
| ListinDataServcie.getPrice(String address1,String address2) | 根据两地地址得到相应的价格 | |

## 5.3.2.4中转接收

**表5 接收与派件模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceiAndSend.addReceiveOrder | 语法 | public ResultMessasge addReceiveOrder (String data, String changeOrderNum, String departPlace,String arriveState, ArrayList<String> orderNum, String name ) |
| 前置条件 | 启动一个接收回合 |
| 后置条件 | 在一个接收回合中，新建一张接收单 |
| ReceiAndSend.addSendOrder | 语法 | public ResultMessasge addSendOrder (String data, ArrayList<String> orderNum, String sender ) |
|  | 前置条件 | 启动一个派件回合 |
|  | 后置条件 | 在一个派件回合中，新建一张派件单 |
| ReceiAndSend.endRecei | 语法 | public void endRecei() |
|  | 前置条件 | 已确认信息 |
|  | 后置条件 | 结束此次接收，持久化更新涉及的对象数据 |
| ReceiAndSend.endSend | 语法 | public void endSend() |
|  | 前置条件 | 已确认信息 |
|  | 后置条件 | 结束此次派件，持久化更新涉及的对象数据 |
| ReceiAndSend.exportReceiOrder | 语法 | public ResultMessage exportReceiOrder(ReceiveorderPO po) |
|  | 前置条件 | 填写接收单完成且确认保存后 |
|  | 后置条件 | 导出excel文件 |
| ReceiAndSend.exportSendOrder | 语法 | public ResultMessage exportSendOrder(SendorderPO po) |
|  | 前置条件 | 填写派件单完成且确认保存后 |
|  | 后置条件 | 导出excel文件 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| ReceiDataService.insert(ReceiveorderPO po) | 增加单一持久化对象 | |
| SendDataService.insert(SendorderPO po) | 增加单一持久化对象 | |
| VehicleDataService.updata(String key) | 更新车辆信息持久化对象中的车辆状态及所在地 | |
| DriverDataService.updata(String key) | 更新司机信息持久化对象中的车辆状态及所在地 | |
| ListinDataService.updata(String key) | 更新订单物流状态 | |

**表6 中转接收的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Receive.build | 语法 | public void build(LoadorderVO lv,ArriveState as) |
| 前置条件 | 启动一个新建到达单回合 |
| 后置条件 | 系统将新建的到达单提交给总经理审批 |
| Receive.setRemind | 语法 | public void setRemind(LoadorderVO lv) |
| 前置条件 | 有货物到达中转中心 |
| 后置条件 | 增加一条提醒消息 |
| Receive.checkRemind | 语法 | public ResultMessage checkRemind() |
| 前置条件 | 启动一个新建到达单回合 |
| 后置条件 | 删除该条提醒消息 |
| Receive.saveOrder | 语法 | public void save(ArriverorderVO av) |
| 前置条件 | 到达单已通过总经理审核 |
| 后置条件 | 将到达单存储 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Approval.checkOrder(PO) | 提交给总经理审核单据 | |
| ReceiveDataService.insert(ArriverorderPO av) | 插入一个持久化对象ArriverorderPO | |

**表7 录入中转单的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Transfer.build | 语法 | public void build(int[][][][] location,String car\_number,String monitor) |
| 前置条件 | 启动一个录入中转单回合 |
| 后置条件 | 系统将中转单提交总经理审批 |
| Transfer.setRemind | 语法 | public void setRemind() |
| 前置条件 | 有库区库存值达到提醒值 |
| 后置条件 | 增加一条提示消息 |
| Transfer.checkRemind | 语法 | public ResultMessage checkRemind() |
| 前置条件 | 启动一个录入中转单回合 |
| 后置条件 | 删除该条消息 |
| Transfer.checkInventory | 语法 | public ResultMessage checkInventory() |
| 前置条件 | 启动一个录入中转单回合 |
| 后置条件 | 返回各个库区的库存信息 |
| Transfer.saveOrder | 语法 | public void save(ChangeorderVO cv) |
| 前置条件 | 中转单通过总经理审批 |
| 后置条件 | 将中转单存储 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Approval.checkOrder(PO) | 提交给总经理审核单据 | |
| Inventory\_management.checkInventory\_now | 查看此时中转中心仓库库存信息 | |
| TransferDataService.insert(ChangeorderPO cp) | 插入一个持久化对象ChangeorderPO | |

## 5.3.2.5装运管理

**表8 车辆装车管理模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Load.addLoadOrder | 语法 | public ResultMessasge addLoadOrder(String data, String shop, String c  arShippingNum,String arrival, String vehicleNum, String monitor，String transfer, ArrayList<String> orderNum ) |
| 前置条件 | 启动一个车辆装车管理回合 |
| 后置条件 | 在一个车辆装车管理中，新建一张装车单 |
| Load.getTotal | 语法 | public double getTotal(String arrival) |
|  | 前置条件 | 装车单填写过程中填写完成到达地 |
|  | 后置条件 | 返回运费总价 |
| Load.endLoad | 语法 | public void endLoad() |
|  | 前置条件 | 已确认信息 |
|  | 后置条件 | 结束此次车辆装车管理，持久化更新涉及的对象数据 |
| Load.export | 语法 | public ResultMessage export(LoadorderPO po) |
|  | 前置条件 | 填写装车单完成且确认保存后 |
|  | 后置条件 | 导出excel文件 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| LoadDataService.add(LoadorderPO po) | 增加单一持久化对象 | |
| VehicleDataService.updata(String key) | 更新车辆信息持久化对象中的车辆状态及所在地 | |
| DriverDataService.updata(String key) | 更新司机信息持久化对象中的车辆状态及所在地 | |
| ListinDataService.updata(String key) | 更新订单物流状态 | |

**表9 装运管理的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Shipping.build | 语法 | public void build(ChangeorderVO cv,String car\_number,String monitor,String transfer) |
| 前置条件 | 启动一个新建装运单回合 |
| 后置条件 | 系统将装运单提交给总经理审批 |
| Shipping.setRemind | 语法 | public void setRemind(ChangeorderVO cv) |
| 前置条件 | 需要装运的中转单上的货物已出库完毕 |
| 后置条件 | 增加一条提示消息 |
| Shipping.checkRemind | 语法 | public ResultMessage checkRemind() |
| 前置条件 | 启动一个新建装运单回合 |
| 后置条件 | 删除该条消息 |
| Shipping.saveOrder | 语法 | public void save(LoadorderVO) |
| 前置条件 | 到达单已经通过总经理审核 |
| 后置条件 | 将中转单存储 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Approval.checkOrder(PO) | 提交给总经理审核单据 | |
| ShippingDataService.insert(LoadorderPO lp) | 插入一个持久化对象LoadorderPO | |

## 5.3.2.6车辆管理

**表10 车辆信息管理模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Vehicle.addVehicle | 语法 | public ResultMessasge addVehicle (String carNum, String engineNum, String car, String baseNum, String buyData, String useData, String carState) |
| 前置条件 | 启动一个车辆信息管理回合 |
| 后置条件 | 在一个车辆信息管理回合中，新建一个车辆信息 |
| Vehicle.updataVehicle | 语法 | public ResultMessage updataVehicle(String carNum, String engineNum, String car, String baseNum, String buyData, String useData, String carState)) |
|  | 前置条件 | 启动一个车辆信息管理回合 |
|  | 后置条件 | 在一个车辆信息管理回合中，更改一个车辆信息 |
| Vehicle.deleteVehicle | 语法 | public ResultMessage deleteVehicle(String carNum) |
|  | 前置条件 | 启动一个车辆信息管理回合 |
|  | 后置条件 | 在一个车辆信息管理回合中，删除一个车辆信息 |
| Vehicle.end | 语法 | public void end() |
|  | 前置条件 | 已确认信息 |
|  | 后置条件 | 结束此次车辆信息管理，持久化更新涉及的对象数据 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| VehicleDataService.find(String carNum) | 根据车辆代号查找单一持久化对象 | |
| VehicleDataService.insert(VehiclePO po) | 插入单一持久化对象 | |
| VehicleDataService.delete(VehiclePO po) | 删除单一持久化对象 | |
| VehicleDataService.updata(VehiclePO po) | 更新单一持久化对象 | |

**表11 司机信息管理模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Driver.addDriver | 语法 | public ResultMessasge addDriver (String driverNum, String name, String shopNum, String birthdata, String idNum, String phoneNum, String sex, String licenseDue,String driverState) |
| 前置条件 | 启动一个司机信息管理回合 |
| 后置条件 | 在一个司机信息管理回合中，新建一个司机信息 |
| Driver.updataDriver | 语法 | public ResultMessage updataDriver(String driverNum, String name, String shopNum, String birthdata, String idNum, String phoneNum, String sex, String licenseDue,String driverState) |
|  | 前置条件 | 启动一个司机信息管理回合 |
|  | 后置条件 | 在一个司机信息管理回合中，更改一个司机信息 |
| Driver.deleteDriver | 语法 | public ResultMessage deleteDriver(String name) |
|  | 前置条件 | 启动一个司机信息管理回合 |
|  | 后置条件 | 在一个司机信息管理回合中，删除一个司机信息 |
| Driver.end | 语法 | public void end() |
|  | 前置条件 | 已确认信息 |
|  | 后置条件 | 结束此次司机信息管理，持久化更新涉及的对象数据 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| DriverDataService.find(String name) | 根据姓名查找单一持久化对象 | |
| DriverDataService.insert(DriverPO po) | 插入单一持久化对象 | |
| DriverDataService.delete(DriverPO po) | 删除单一持久化对象 | |
| DriverDataService.updata(DriverPO po) | 更新单一持久化对象 | |

## 5.3.2.7财务管理

**表12 管理收款单模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceiForm.addReceiForm | 语法 | public ResultMessasge addReceiveOrder (String data, ArrayList<Integer> price, ArrayList<String> orderNum, String carrierName, String recorder ) |
| 前置条件 | 启动一个管理收款单回合 |
| 后置条件 | 在一个管理收款单回合中，新建一张收款单 |
| ReceiForm.getTotal | 语法 | public double getTotal(ArrayList<String> orderNum) |
|  | 前置条件 | 收款单填写过程中填写完成订单 |
|  | 后置条件 | 返回运费总价 |
| ReceiForm.endReceiForm | 语法 | public void endReceiForm() |
|  | 前置条件 | 已确认信息 |
|  | 后置条件 | 结束此次收款单管理，持久化更新涉及的对象数据 |
| ReceiForm.exportPayeeorder | 语法 | public ResultMessage exportPayeeorder (PayeeorderPO po) |
|  | 前置条件 | 填写管理收款单完成且确认保存后 |
|  | 后置条件 | 导出excel文件 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| ReceiFormDataService.insert(PayeeorderPO po) | 增加单一持久化对象 | |
|  |  | |

**表13 成本管理用例的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Cost.new | 语法 | public ResultMessage new(int paymoney,String data,String payname,String payaccount,String list,String comment); |
| 前置条件 | 启动一个新建付款单的任务 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Cost.check | 语法 | public ResultMessage check(); |
| 前置条件 | 启动一个查看付款单的任务 |
| 后置条件 | 显示所有的付款信息 |
| Cost.profit | 语法 | public double profit(doubl[]e pay,double[] payee); |
| 前置条件 | 已计算出总收入、总支出和总利润 |
| 后置条件 | 显示总收入、总支出和总利润 |
| Cost.excel | 语法 | public void excel(String data,double income,double outcome,double benefit); |
| 前置条件 | 已启动查看付款信息任务，并创建成本收益表 |
| 后置条件 | 将成本收益表导出到电脑 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| PayorderDataService.insert(PayorderPO po) | 插入单一持久化对象 | |
| PayorderDataService.finds() | 查找涉及的领域的所有持久化对象 | |
| FinancebaseFactory.gePayorderDatabase | 得到Payorder数据库的服务的引用 | |

**表14 结算管理用例的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Calculate.check | 语法 | public ResultMessage check(String data,String shop); |
| 前置条件 | 启动一个查看结算信息任务，并选定日期与营业厅 |
| 后置条件 | 显示选定日期选定营业厅的收款信息 |
| Calculate.total | 语法 | public double total(double[] payee); |
| 前置条件 | 已启动查看结算信息任务，并已计算总收款金额 |
| 后置条件 | 显示总收款金额 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| DatabaseFactory.getCalculateDatabase | 得到Calculatee数据库的服务的引用 | |
| CalculateDataService.finds(String data,String shop) | 根据字段名和值进行查找多个持久化对象 | |

**表15 统计报表用例的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Statistics.check | 语法 | public ResultMessage check(String start,String end); |
| 前置条件 | 启动一个查看统计报表任务，并输入起始和结束日期 |
| 后置条件 | 显示选定日期内所有的入款单和收款单信息 |
| Statistics.excel | 语法 | public ResultMessage excel(String start,String end); |
| 前置条件 | 已启动查看统计报表任务，并已创建经营情况统计表 |
| 后置条件 | 将经营情况表导出到电脑 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| DatabaseFactory.ge StatisticsDatabase | 得到Statistics数据库的服务的引用 | |
| StatisticsDataService.finds(String start,String end) | 根据字段名和值进行查找多个持久化对象 | |

## 5.3.2.8仓库管理

**表16 入库管理的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Warehouse-in.build | 语法 | public void build(ArriverorderVO av,int jia,int wei) |
| 前置条件 | 启动一个入库管理回合 |
| 后置条件 | 系统将入库单提交总经理审批 |
| Warehouse-in.setRemind | 语法 | public void setRemind(ArriverorderVO av) |
| 前置条件 | 货物录入到达单 |
| 后置条件 | 增加一条消息提示 |
| Warehouse-in.checkRemind | 语法 | public ResultMessage checkRemind() |
| 前置条件 | 启动一个入库管理回合 |
| 后置条件 | 删除该条消息 |
| Warehouse-in.saveOrder | 语法 | public void save(StoreinorderVO sv) |
| 前置条件 | 入库单通过总经理审批 |
| 后置条件 | 将入库单存储 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Approval.checkOrder(PO) | 提交给总经理审核单据 | |
| Warehouse-in DataService.insert(StoreinorderPO sp) | 插入一个持久化对象StoreinorderPO | |

**表17 出库管理的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Warehouse-out.build | 语法 | public void build(ChangeorderVO,String number) |
| 前置条件 | 启动一个出库管理回合 |
| 后置条件 | 系统将出库单提交总经理审批 |
| Warehouse-out.setRemind | 语法 | public void setRemind(ChangeorderVO cv) |
| 前置条件 | 货物录入中转单 |
| 后置条件 | 增加一条消息提示 |
| Warehouse-out.checkRemind | 语法 | public ResultMessage checkRemind() |
| 前置条件 | 启动一个出库管理回合 |
| 后置条件 | 删除该条消息 |
| Warehouse-out.saveOrder | 语法 | public void save(StoreoutorderVO sv) |
| 前置条件 | 出库单通过总经理审批 |
| 后置条件 | 将出库单存储 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Approval.checkOrder(PO) | 提交给总经理审核单据 | |
| Warehouse-out rDataService.insert(StoreoutorderPO sp) | 插入一个持久化对象StoreoutorderPO | |

**表18 库存管理的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Inventory\_management.checkInventory\_now | 语法 | public ResultMessage checkInventory(int store\_qu) |
| 前置条件 | 中转中心业务员启动一个录入中转单回合 |
| 后置条件 | 系统显示各库区库存量 |
| Inventory\_management.checkInventory | 语法 | public ResultMessage checkInventory(Data startdata,Data overdata) |
| 前置条件 | 启动一个查看库存回合 |
| 后置条件 | 系统显示各个库区出入库数量和金额 |
| Inventory\_management.take\_stock | 语法 | public void setRemind(int store\_number,int store\_qu, Data startdata,Data overdata,string way) |
| 前置条件 | 启动一个盘点库存回合 |
| 后置条件 | 根据查看方式，显示或输出制定库区所有出入库的订单信息 |
| Inventory\_management.setWarnData | 语法 | public void setWarnData (int store\_number, int warn\_value) |
| 前置条件 | 启动一个管理警戒值回合 |
| 后置条件 | 系统更新警戒值 |
| Inventory\_management.setRemindData | 语法 | public void setRemindData (int store\_number, int remind\_value) |
| 前置条件 | 启动一个管理提醒值回合 |
| 后置条件 | 系统更新提醒值 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Inventory\_managementDataService.check(int store\_number) | 查看当前中转中心最新的StorePO | |
| Inventory\_managementDataService.  updata(StorePO sp，int store\_number) | 更新当前中转中心StorePO | |
| Inventory\_managementDataService.  checknumber(int store\_number, Data startdata,Data overdata) | 查看StoreNumber.txt | |
| Inventory\_managementDataService.  addnumber(int store\_number,Data,int storein\_number,int storeout\_number,int storein\_money,int storeout\_money) | 更新StoreNumber.txt | |
| Inventory\_managementDataService.  checkorder(int store\_number,int store\_qu, Data startdata,Data overdata) | 查看StoreOrder.txt | |
| Inventory\_managementDataService.  addorder(int store\_number,OrderPO op) | 更新StoreOrder.txt | |

## 5.3.2.9期初建账

**表19 期初建账用例的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Start.new | 语法 | public ResultMessage new(String institution,String staff,String vehicle,String store,String account,String data,String creator); |
| 前置条件 | 启动一个期初建账任务 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Start.check | 语法 | public ResultMessage check(String billname); |
| 前置条件 | 启动一个查看期初信息的任务 |
| 后置条件 | 显示选定的帐的期初信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| StartinfoDataService.insert(StartinfoPO po) | 插入单一持久化对象 | |
| StartinfoDataService.finds(String billname) | 根据账名（billname）查找单一持久化对象 | |
| DatabaseFactory.getStartinfoDatabase | 得到Startinfo数据库的服务的引用 | |

## 5.3.2.10账户管理

**表20 账户管理用例的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Account.new | 语法 | public ResultMessage new(String name,double pay,double payee,String creator); |
| 前置条件 | 启动一个新建账户任务 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Account.updata | 语法 | public ResultMessage updata(String oldname,String newname); |
| 前置条件 | 已选定目标修改的账户，并启动修改账户的任务 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Account.delete | 语法 | public ResultMessage delete(String aimname); |
| 前置条件 | 已选定目标删除的账户，且账户余额为0，并启动删除账户的任务 |
| 后置条件 | 删除涉及的领域对象的数据 |
| Account.search | 语法 | public ResultMessage search(String key); |
| 前置条件 | 已输入关键字，并启动查询账户的任务 |
| 后置条件 | 查找是否存在与输入的关键字相关的账户，显示可能的账户的信息或提示输入更准确的关键字 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| AccountDataService.insert(AccountPO po) | 插入单一持久化对象 | |
| AccountDataService.updata(AccountPO po) | 更新单一持久化对象 | |
| AccountDataService.delete(AccountPO po) | 删除单一持久化对象 | |
| AccountDataService.find(String name) | 根据账户名（name）查找单一持久化对象 | |
| AccountDataService.finds(String key) | 根据字段名和值进行查找多个持久化对象 | |
| DatabaseFactory.getAccountDatabase | 得到Account数据库的服务的引用 | |

## 5.3.2.11员工管理

**表21 staffbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| staff.showStaff | 语法 | Public ResultMessage showStaff（） |
| 前置条件 | 启动一个人员管理任务 |
| 后置条件 | 界面显示多份人员信息 |
| staff.checkStaff | 语法 | Public ResultMessage checkStaff（String name） |
| 前置条件 | 点击人员信息的其中一项 |
| 后置条件 | 界面显示人员具体信息 |
| staff.addStaff | 语法 | Public ResultMessage addStaff（String name,String address,String position,String phone,String cellphone） |
| 前置条件 | 启动一个新建人员信息表任务 |
| 后置条件 | 界面显示新建的一份人员信息表 |
| staff.searchStaff | 语法 | Public ResultMessage searchStaff（String name） |
| 前置条件 | 启动一个查询人员信息任务 |
| 后置条件 | 查找是否存在相应姓名的人员信息，返回查询验证的结果 |
| staff.editStaff | 语法 | Public ResultMessage editStaff（StaffPO po） |
| 前置条件 | 启动一个修改人员信息表任务 |
| 后置条件 | 持续化更新涉及的领域对象的数据 |
| staff.deleteStaff | 语法 | Public ResultMessage removeStaff（StaffPO po） |
| 前置条件 | 启动一个删除人员信息表任务 |
| 后置条件 | 持续化更新涉及的领域对象的数据 |
| staff.endStaff | 语法 | Public ResultMessage endStaff（） |
| 前置条件 | 结束人员管理任务 |
| 后置条件 | 结束人员管理，持续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 | |
| StaffDataService.gets() | 获得多个持久化对象 | |
| StaffDataService.get(StaffPO po) | 获得单一持久化对象 | |
| StaffDataService.insert(String name) | 插入单一持久化对象 | |
| StaffDataService. delete(StaffPO po) | 删除单一持久化对象 | |
| StaffDataService. updata(StaffPO po) | 修改单一持久化对象 | |
| DatabaseFacory.getStaffDatabase | 得到Policy数据库的服务的引用 | |

## 5.3.2.12策略管理

**表22 policybl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| policy.checkSalary | 语法 | Public ResultMessage checkSalary（String typeOfStaff, double salary, int typeOfStrategy） |
| 前置条件 | 点击人员薪水表的其中一项 |
| 后置条件 | 界面薪水表项对应的具体信息 |
| policy.editSalary | 语法 | Public ResultMessage editStaff（SalaryPO po） |
| 前置条件 | 启动一个编辑人员信息任务 |
| 后置条件 | 持续化更新涉及的领域对象的数据 |
| policy.endSalary | 语法 | Public ResultMessage endSalary（） |
| 前置条件 | 结束薪水管理任务 |
| 后置条件 | 结束人员管理，持续化更新涉及的领域对象的数据 |
| policy.searchConstant | 语法 | Public ResultMessage searchConstant（String address1, String address2, double distance, double price） |
| 前置条件 | 启动一个查询两地价格和距离信息任务，并且两地地址输入规范 |
| 后置条件 | 查找是否存在相应两地，返回查询验证的结果 |
| policy.editConstant | 语法 | Public ResultMessage editConstant（ConstantPO po） |
| 前置条件 | 启动一个修改常量表任务 |
| 后置条件 | 持续化更新涉及的领域对象的数据 |
| policy.endConstant | 语法 | Public ResultMessage endConstant（） |
| 前置条件 | 结束常量管理任务 |
| 后置条件 | 结束人员管理，持续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 | |
| PolicyDataService.getSalary(String name) | 获得单一持久化对象 | |
| PolicyDataService.findConstant(String address1，String address2) | 根据String查找单一持久化对象 | |
| PolicyDataService.updataSalary(PolicyPO po) | 修改单一持久化对象 | |
| PolicyDataService.updataConstant(PolicyPO po) | 修改单一持久化对象 | |
| DatabaseFacory.getPolicyDatabase | 得到Policy数据库的服务的引用 | |

## 5.3.2.13审批单据

**表23 approvebl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| approve.showList | 语法 | Public ResultMessage showSList（enum state,String type） |
| 前置条件 | 选择单据种类和状态后，请求显示相关单据 |
| 后置条件 | 界面显示相应的所有单据 |
| approve.checkList | 语法 | Public ResultMessage checkList（String type） |
| 前置条件 | 点击订单的其中一项 |
| 后置条件 | 界面对应的单据具体信息 |
| approve.editList | 语法 | Public ResultMessage editList（ListPO po） |
| 前置条件 | 启动一个编辑单据信息任务 |
| 后置条件 | 持续化更新涉及的领域对象的数据 |
| approve.endApprove | 语法 | Public ResultMessage endApprove（） |
| 前置条件 | 结束审批单据任务 |
| 后置条件 | 结束单据审批，持续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 | |
| ApproveDataService.gets(enum state,String type) | 获得多个持久化对象 | |
| ApproveDataService.get(String name) | 根据string查找单一持久化对象 | |
| ApproveDataService.updata(T po) | 修改单一持久化对象 | |
| DatabaseFacory.getApproveDatabase | 得到Policy数据库的服务的引用 | |

## 5.4数据层的分解

//编写

#### 5.4.1数据层的分解

|  |  |
| --- | --- |
| **模块** | **职责** |
| SalesDataService | 持续化数据库的接口，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接口，提供集体载入、集体保存、增、删、改、查服务 |
| SalesDataServiceSerializableImpl | 基于序列化文件的持久化数据库接口，提供集体载入、集体保存、增、删、改、查服务 |
| SalesDataServiceMySqlImpl | 基于Mysql文件的持久化数据库接口，提供集体载入、集体保存、增、删、改、查服务 |
| SalesDataService | 持续化数据库的接口，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

## 5.4.2.1用户登录

用户管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| UserDataService.find | 语法 | public UserPO find(String name) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按用户名（name）进行查找返回相应的UserPO结果 |
| UserDataService.insert | 语法 | public void insert（UserPO po）throws RemoteException; |
| 前置条件 | 同样名称的po在数据库中不存在 |
| 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名称的po |
| 后置条件 | 删除一个po |
| UserDataService.updata | 语法 | public void updata(UserPO po) hrows RemoteException; |
| 前置条件 | 在数据库中存在同样名称的po |
| 后置条件 | 更新一个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订单查询

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ListinDataService.find | 语法 | public void OrderPO find()throws RemoteException; |
| 前置条件 | po在Mapper存在 |
| 后置条件 | 返回一个po记录 |

## 5.4.2.3单据输入

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ListinDataService.insert | 语法 | public void insert(OrderPO po)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper不存在 |
| 后置条件 | 在数据库增加一个po记录 |
| ListinDataService.insertReceiver | 语法 | public ReceiverPO insertReceiver(String id)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper不存在 |
| 后置条件 | 根据ID查找是否存在相应的的OrderPO，若有，返回一个的ReceiverPO |
| ListinDataService.getPrice | 语法 | public ConstantPO getPrice (String address1,String address2)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 根据ID查找是否存在相应的的OrderPO，若有，返回一个的ReceiverPO |

## 5.4.2.4中转接收

接收与派件的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceiDataService.insert | 语法 | public void insert(ReceiveorderPO po)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper不存在 |
| 后置条件 | 在数据库增加一个po记录 |
| SendDataService.insert | 语法 | public void insert (SendorderPO po)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper不存在 |
| 后置条件 | 在数据库增加一个po记录 |
| VehicleDataService.updata | 语法 | public void updata (String key)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper中存在 |
| 后置条件 | 根据ID查找是否存在相应的的VehiclePO，若有，覆盖该po |
| DriverDataService.updata | 语法 | public void updata (String key)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper中存在 |
| 后置条件 | 根据ID查找是否存在相应的的DriverPO，若有，覆盖该po |
| ListinDataService.updata | 语法 | public void updata (String key)throws RemoteException; |
|  | 前置条件 | 相同的ID的po在Mapper中存在 |
|  | 后置条件 | 根据ID查找是否存在相应的的OrderPO，若有，覆盖该po |

中转接受的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceiveDataService.insert | 语法 | public void insert(ArriverorderPO ap)throws RemoteException |
| 前置条件 | 同样的ArriverorderPO在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |

录入中转单的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| TransferDataService.insert | 语法 | public void insert(ChangeorderPO cp)throws RemoteException |
| 前置条件 | 同样的ChangeorderPO在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |

## 5.4.2.5装运管理

车辆装车管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LoadDataService.insert | 语法 | public void insert (LoadorderPO po)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper不存在 |
| 后置条件 | 在数据库增加一个po记录 |
| VehicleDataService.updata | 语法 | public void updata (String key)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper中存在 |
| 后置条件 | 根据ID查找是否存在相应的的VehiclePO，若有，覆盖该po |
| DriverDataService.updata | 语法 | public void updata (String key)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper中存在 |
| 后置条件 | 根据ID查找是否存在相应的的DriverPO，若有，覆盖该po |
| ListinDataService.updata | 语法 | public void updata (String key)throws RemoteException; |
|  | 前置条件 | 相同的ID的po在Mapper中存在 |
|  | 后置条件 | 根据ID查找是否存在相应的的OrderPO，若有，覆盖该po |

装运管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ShippingDataService.insert | 语法 | public void insert(LoadorderPO lp)throws RemoteException |
| 前置条件 | 同样的LoadorderPO在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |

## 5.4.2.6车辆管理

车辆信息管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| VehicleDataService.insert | 语法 | public void insert (VehiclePO po)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper不存在 |
| 后置条件 | 在数据库增加一个po记录 |
| VehicleDataService.updata | 语法 | public void updata (VehiclePO po)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper中存在 |
| 后置条件 | 根据ID查找是否存在相应的的VehiclePO，若有，覆盖该po |
| VehicleDataService.find | 语法 | public void find (String carNum)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper中存在 |
| 后置条件 | 根据ID查找是否存在相应的的VehiclePO，若有，返回该po |
| VehicleDataService.delete | 语法 | public void delete (VehiclePO po)throws RemoteException; |
|  | 前置条件 | 相同的ID的po在Mapper中存在 |
|  | 后置条件 | 根据ID查找是否存在相应的的VehiclePO，若有，覆盖该po |

司机信息管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| DriverDataService.insert | 语法 | public void insert (DriverPO po)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper不存在 |
| 后置条件 | 在数据库增加一个po记录 |
| DriverDataService.updata | 语法 | public void updata (DriverPO po)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper中存在 |
| 后置条件 | 根据ID查找是否存在相应的的DriverPO，若有，覆盖该po |
| DriverDataService.find | 语法 | public void find (String name)throws RemoteException; |
| 前置条件 | 相同的ID的po在Mapper中存在 |
| 后置条件 | 根据ID查找是否存在相应的的DriverPO，若有，返回该po |
| DriverDataService.delete | 语法 | public void delete (DriverPO po)throws RemoteException; |
|  | 前置条件 | 相同的ID的po在Mapper中存在 |
|  | 后置条件 | 根据ID查找是否存在相应的的DriverPO，若有，覆盖该po |

## 5.4.2.7财务管理

管理收款单的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceiFormDataService.insert | 语法 | public void insert(PayeeorderPO po)throws RemoteException |
| 前置条件 | 同样的PayeeorderPO在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |

成本管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FinanceDataService.find | 语法 | public PayorderPO find() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 查找返回相应的PayorderPO结果 |
| FinanceDataService.insert | 语法 | public void insert（PayorderPO po）throws RemoteException; |
| 前置条件 | 同样名称的po在数据库中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| 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 |

结算管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FinanceDataService.find | 语法 | public PayoeerderPO find(String data,String shop) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按日期（data）和营业厅（shop）进行查找返回相应的PayeeorderPO结果 |
| 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 |

统计报表的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FinanceDataService.find | 语法 | public PayorderPO find(String start,String end) throws RemoteException;  public PayeeorderPO find(String start,String end) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按起始和结束日期进行查找返回相应的PayorderPO和PayeeorderPO结果 |
| 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 |

## 5.4.2.8仓库管理

入库管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Warehouse-in DataService.insert | 语法 | public void insert(StoreinorderPO sp)throws RemoteException |
| 前置条件 | 同样的StoreinorderPO在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |

出库管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Warehouse-out rDataService.insert | 语法 | public void insert(StoreoutorderPO sp)throws RemoteException |
| 前置条件 | 同样的StoreoutorderPO在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |

库存管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Inventory\_managementDataService.check | 语法 | public StorePO check(int store\_number) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回对应中转中心的StorePO |
| Inventory\_managementDataService.updata | 语法 | public void updata (StorePO sp，int store\_number) throws RemoteException |
| 前置条件 | 在数据库中存在同样中转中心的的StorePO |
| 后置条件 |  |
| Inventory\_managementDataService.  checknumber | 语法 | public void checknumber(int store\_number, Data startdata,Data overdata)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按Data进行查找StoreNumber.txt返回相应的信息 |
| Inventory\_managementDataService.  addnumber | 语法 | public void addnumber(int store\_number,Data,int storein\_number,int storeout\_number,int storein\_money,int storeout\_money)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在文件中添加一行对应天出入库数量和金额的信息 |
| Inventory\_managementDataService.  checkorder | 语法 | public void checkorder(int store\_number,int store\_qu, Data startdata,Data overdata)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按int和Data进行查找StoreOrder.txt 返回相应的信息 |
| Inventory\_managementDataService.  addorder | 语法 | public void addorder(int store\_number,OrderPO op)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在文件中添加一行快递编号、入库日期、目的地、区号、排号、架号、位号信息 |

## 5.4.2.9期初建账

期初建账的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BillDataService.find | 语法 | public StartinfoPO find(String name) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按账名（name）进行查找返回相应的StartinfoPO结果 |
| BillDataService.insert | 语法 | public void insert（StartinfoPO po）throws RemoteException; |
| 前置条件 | 同样名称的po在数据库中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| BillDataService.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 |
| BillDataService.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 |

## 5.4.2.10账户管理

账户管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| AccountDataService.find | 语法 | public AccountPO find(String name) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按账户名（name）进行查找返回相应的AccountPO结果 |
| AccountDataService.insert | 语法 | public void insert（AccountPO po）throws RemoteException; |
| 前置条件 | 同样名称的po在数据库中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| AccountDataService.delete | 语法 | public void delete（AccountPO po）throws RemoteException; |
| 前置条件 | 在数据库中存在同样名称的po |
| 后置条件 | 删除一个po |
| AccountDataService.updata | 语法 | public void updata(AccountPO po) hrows RemoteException; |
| 前置条件 | 在数据库中存在同样名称的po |
| 后置条件 | 更新一个po |
| AccountDataService.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 |
| AccountDataService.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 |

## 5.4.2.11员工管理

|  |
| --- |
| **提供的服务（供接口）** |

|  |  |  |
| --- | --- | --- |
| StaffDataService.gets | 语法 | public ArrayList<StaffPO>() gets()throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回一个StaffPO的ArrayList |
| StaffDataService.get | 语法 | public StaffPO() get(String name)throws RemoteException; |
| 前置条件 | 相同的name的po在Mapper不存在 |
| 后置条件 | 返回一个相应名字的StaffPO |
| StaffDataService.insert | 语法 | public void insert(StaffPO po)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中插入一个持久化po记录 |
| StaffDataService.delete | 语法 | public void delete(StaffPO po)throws RemoteException; |
| 前置条件 | 相同的PO在数据库中仍然存在 |
| 后置条件 | 删除一个PO |
| StaffDataService. updata | 语法 | public void updata(StaffPO po)throws RemoteException; |
| 前置条件 | 相同的PO在数据库中仍然存在 |
| 后置条件 | 更新一个PO |

## 5.4.2.12策略管理

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| PolicyDataService.getSalary | 语法 | public SalaryPO getSalary(String name)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回一个相应的SalaryPO |
| PolicyDataService.findConstant | 语法 | public ConstantPO findConstant(String address1，String address2)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 根据两地地址查找是否存在相应的的ConstantPO，若有，返回一个的ConstantPO |
| PolicyDataService.updataSalary | 语法 | public void updataSalary(PolicyPO po)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 更新一个PO |
| PolicyDataService.updataConstant | 语法 | public void updataConstant (PolicyPO po)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 更新一个PO |

## 5.4.2.13审批单据

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ApproveDataService.gets | 语法 | public ArrayList<T po> gets(enum state,String type)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 根据状态和单据类型获得多个含持久化对象的ArrayList |
| ApproveDataService.get | 语法 | public T get(String name)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 根据状态和单据类型获得多个含持久化对象的ArrayList |
| ApproveDataService. updata | 语法 | public void updata (T po)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 更新一个PO |

# 信息视角

## 数据持久化对象

UserPO（用户）类包含用户的用户名、密码、权限。

OrderPO（订单）类包含寄件人和收件人各自的姓名、住址、单位、电话，托运货物的原件数、实际重量、体积、内件品名，快递种类，包装费，合计费用，订单条形码号、快递员姓名、物流信息及审批状态。

ReceiverPO（收件人信息）类包含收件人姓名、手机、电话、收件日期、地点，快递员姓名、电话。

LoadorderPO（装车单）类包含装车日期、本营业厅编号、汽运编号、到达地、车辆代号、监装员、押运运、运费及审批状态。

VehiclePO（车辆信息）类包含车辆代号、车牌号、服役时间。

DriverPO（司机信息）类包含司机编号、姓名、出生日期、身份证号、手机、性别、行驶证期限。

ArriverorderPO（到达单）类包含到达日期、中转单编号、出发地、货物到达状态及审批状态。

ReceiveorderPO（接收单）类接收方信息、接收日期、派送人信息及审批状态、中转单编号、到达状态、订单号、接收人姓名、出发地

SendorderPO（派件单）类包含到达日期、托运订单号、派送员及审批状态。

PayeeorderPO（收款单）类包含收款人姓名、收款日期、收款金额、付款人信息及审批状态。

ChangeorderPO（中转单）类包含装车日期、本中转中心中转单编号、航班号、出发地、到达地、货柜号、监装员、本次装箱所有托运单号、运费及审批状态。

StorePO（仓库信息）类包含库存警戒值、提醒值、各位置库存信息。

StoreinorderPO（入库单）类包含快递编号、入库日期、目的地、区号、排号、架号、位号及审批状态。

StoreoutorderPO（出库单）类包含快递编号、出库日期、目的地、装运形式、中转单编号、汽运编号及审批状态。

PayorderPO（付款单）类包含付款金额、付款日期、付款人、付款账号、条目、备注信息、审批状态。

StartinfoPO（期初信息）类包含机构、人员、车辆、库存、银行账户信息、建账日期、建账人及审批状态。

AccountPO（账户信息）类包含账户名称、余额、建账日期、建账人、审批状态。

StaffPO（员工信息）类包含员工姓名、电话、地址、职位、手机。

SalaryPO（薪水信息）类包含员工类别、各自的薪水、薪水策略。

ConstantPO（常量信息）类包含两个城市地名、城市距离、价格。

持久化对象UserPO的定义如图6.1所示

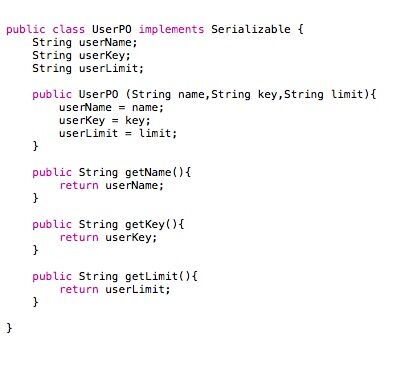


图6.1 持久化对象UserPO的定义

持久化对象AccountPO的定义如图6.2所示

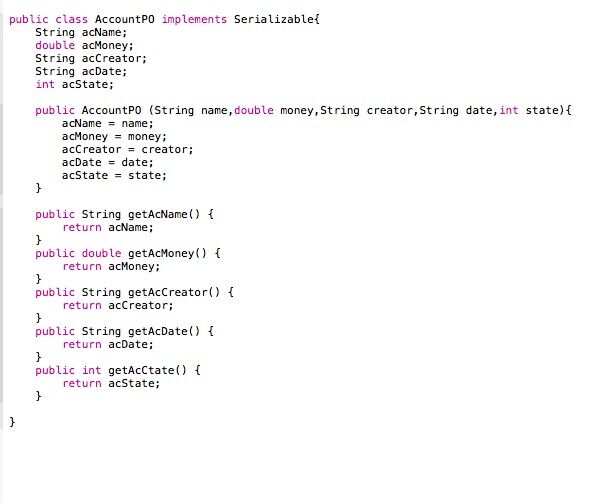


图6.2 持久化对象AccountPO的定义

持久化对象PayorderPO的定义如图6.3所示

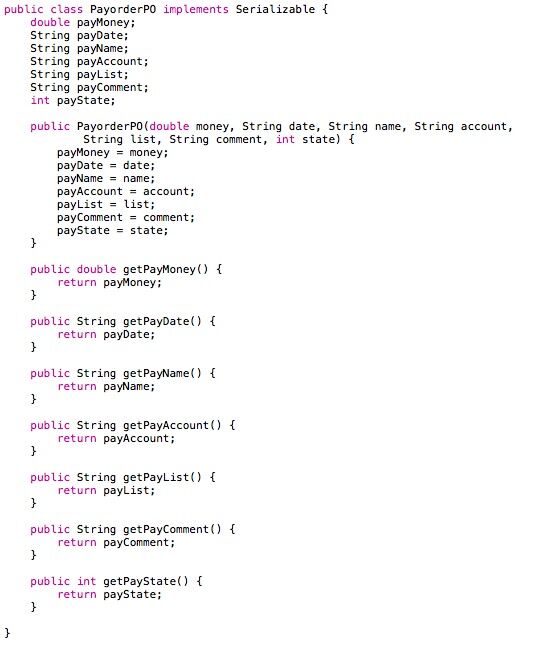


图6.3 持久化对象PayorderPO的定义

持久化对象StartinfoPO的定义如图6.4所示

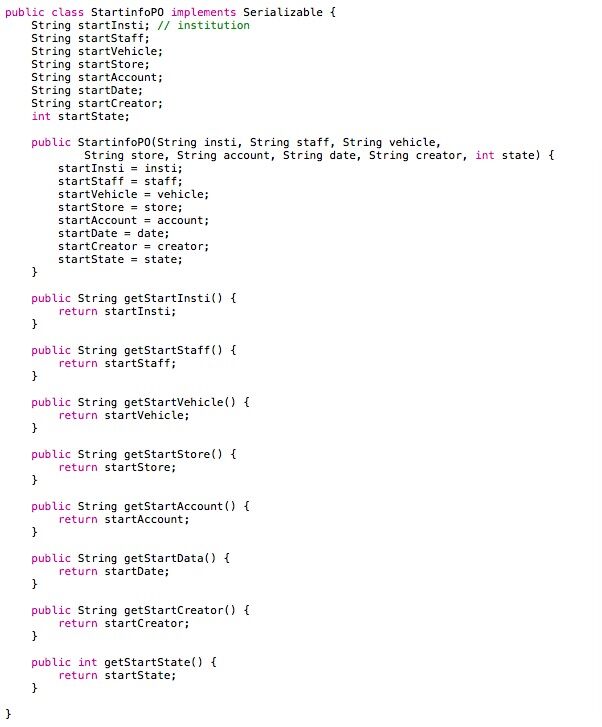


图6.4 持久化对象StartinfoPO的定义

## Txt持久化格式

1. User.txt用户信息保持格式：

用户名：密码【\n】

1. Order.txt订单信息保持格式：

订单号（10位数字）——订单建立日期【\n】

1. Receiver.txt收件人信息保持格式：

收件人姓名 收件人联系方式 收件日期 收件地址 快递员姓名【\n】

4、Loadorder.txt装运单保持格式：

0中转中心编号+00000000日期+00000编码 、五位数字【\n】

5、Vehicle.txt车辆信息保持格式：

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）【\n】

6、Driver.txt司机信息保持格式：

司机编号（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）【\n】

7、Arriverorder.txt到达单保持格式：

来自的营业厅编号——到达的营业厅编号：到达日期【\n】

8、Receiveorder.txt接收单保持格式：

接收的营业厅编号——司机姓名——日期【\n】

9、Sendorder.txt派件单保持格式：

派件的营业厅编号——目的营业厅编号——司机姓名——日期【\n】

10、Payee.txt收款单保持格式：

收款金额，快递员，日期【\n】

11、Changeorder.txt中转单保持格式：

0中转中心编号+00000000日期+00000编码 、五位数字【\n】

12、Store.txt仓库信息保持格式：

营业厅编号——订单号——入库日期【\n】

13、Storeinorder.txt入库单保持格式：

营业厅编号——订单号——仓库管理员姓名——入库日期【\n】

14、Storeoutorder.txt出库单保持格式：

营业厅编号——订单号——仓库管理员姓名——出库日期【\n】

15、Payorder.txt付款单保持格式：

付款日期，付款金额，付款人，付款账号，条目，备注信息【\n】

16、Startinfo.txt期初信息保持格式：

（季度）

（机构）

（人员信息）

（车辆信息）

（库存信息）

（银行账户信息）

（建账人姓名——建账日期）【\n】

17、Account.txt账户信息保持格式：

账户名称：余额，建账人——建账日期【\n】

18、Staff.txt员工信息保持格式：

姓名：联系方式，职位【\n】

19、Salary.txt薪水信息保持格式：

姓名：薪水【\n】

20、Constant.txt常量信息保持格式：

城市距离：价格【\n】

21、StoreNumber.txt

出库数量：入库数量：出库金额：入库金额

22、StoreOrder.txt

快递编号：入库日期：目的地：区号：排号：架号：位号信息

## 数据库表

数据库中包含User表、Order表、Receiver表、Loadorder表、Vehicle表、Driver表、Arriverorder表、Receiveorder表、Sendorder表、Payeeorder表、Changeorder表、Store表、Storeinorder表、Storeoutorder表、Payorder表、Startinfo表、Account表、Staff表、Salary表、Constant表