**快递物流系统**

**(Express Logistics System)**

**软件详细设计描述**



**学 院：南京大学软件学院**

**团 队：伪装者W小组**

**成 员：张词校 张家盛 王丽莉 魏彦淑**

**完成日期：2015年11月05日**

# 1.引言

## 1.1编制目的

本报告详细完成对进销存系统的概要设计，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。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ELS | 快递物流系统（Express Logistics System） | 无 |
| BL | 业务逻辑（Business　Logic） | 无 |

## 1.3参考资料

[IEEE1471-2000]

快递物流系统用例文档和快递物流系统需求规格说明。

# 2.产品概述

参考快递物流系统用例文档和快递物流系统需求规格说明对产品的概括描述。

# 3.体系结构设计概述

参考快递物流系统体系结构设计文档中对体系结构设计的概述。

# 4.结构视角

## 4.1展示层的分解

### 4.1.1模块职责

如表1所示为用户界面层模块的职责。

**表1 用户界面层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 界面Frame，负责界面的显示和跳转 |
| ExpressFrame | 快递员部分界面Frame，负责快递员部分界面的显示与跳转 |
| BusinessFrame | 营业厅部分界面Frame，负责营业厅部分界面的显示和跳转 |
| IntermediateFrame | 中转中心部分界面Frame，负责中转中心部分界面的显示和跳转 |
| ReceiptFrame | 单据部分界面Frame，负责单据部分界面的显示 |
| FinanceFrame | 财务部分界面Frame，负责财务部分界面的显示和跳转 |
| RepertoryFrame | 仓库部分界面Frame，负责仓库部分界面的显示与跳转 |
| ManageFrame | 总经理部分界面Frame，负责总经理部分界面的显示和跳转 |
| UserFrame | 用户部分界面Frame，负责用户登录界面和管理员增删改查用户界面的显示和跳转 |

### 4.1.2接口规范

用户界面层模块的接口规范如表2所示。

**表2 用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | Frameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示Frame |
| ExpressFrame | 语法 | ExpressFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示快递员部分界面的Frame以及订单新建，订单完成，订单查询和订单确认收费的panel |
| BusinessFrame | 语法 | BusinessFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示营业厅部分界面的Frame以及装车管理，车辆和司机的信息管理，派件和收费汇总的panel |
| IntermediateFrame | 语法 | IntermediateFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示中转中心部分界面的Frame以及飞机，火车，汽车装运管理和中转接收的panel |
| ReceiptFrame | 语法 | ReceiptFrameFrameini（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示单据部分界面的Frame以及各种单据查增删改的panel |
| FinanceFrame | 语法 | FinanceFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示财务界面的Frame以及表单新建，报表查看，期初建账和账户管理的panel |
| RepertoryFrame | 语法 | RepertoryFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示仓库界面的Frame以及商品出、入库管理,库存初始化，库存报警，库存查看和库存盘点的panel |
| ManageFrame | 语法 | ManageFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示总经理界面的Frame以及机构管理，人员管理，基础数据设置，财务报表查看的panel |
| UserFrame | 语法 | UserLogInFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示用户登录界面的Frame以及跳转 |
| UserFrame | 语法 | UserManageFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示管理员界面的Frame以及账号管理的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**

|  |  |
| --- | --- |
| **服务名** | **服务** |
| businesslogicservice.userblservice.UserBLService | 用户信息的增删改查的业务逻辑接口和登录界面的业务逻辑接口 |
| businesslogicservice.businessblservice.DriverManagerBLService  businesslogicservice.businessblservice.VehicleManagerBLService  businesslogicservice.businessblservice.EnVehicleBLService  businesslogicservice.businessblservice.GatheringBLService  businesslogicservice.businessblservice.DistributeOrderBLService  businesslogicservice.businessblservice.AcceptCargoBLService | 营业厅司机信息的增删改查的业务逻辑接口  营业厅车辆信息的增删改查的业务逻辑接口  营业厅车辆装车安排及的业务逻辑接口  营业厅每日收款单列表提交的业务逻辑接口  营业厅派件的业务逻辑接口  营业厅接收货物的业务逻辑接口 |
| businesslogicservice.expressblservice.LogisticQueryBLService  businesslogicservice.expressblservice.AddOrderBLService  businesslogicservice.expressblservice.ChargeCollectionBLService  businesslogicservice.expressblservice.ReceiptOrderBLService | 快递员查询订单的业务逻辑接口  快递员输入订单的业务逻辑接口  快递员进行收费信息汇总的业务逻辑接口  快递员收件信息输入的业务逻辑接口 |
| businesslogicservice.financeblservice.CollectionReceiptBLService  businesslogicservice.financeblservice.PaymentReceiptBLService  businesslogicservice.financeblservice.CostIncomeReceiptBLService  businesslogicservice.financeblservice.BusinessstatementReceiptBLService  businesslogicservice.financeblservice.InitialStockBLService  businesslogicservice.financeblservice.AccountBLService | 结算管理的业务逻辑接口  新建付款单的业务逻辑接口  新建成本收益表的业务逻辑接口  查看经营情况表的业务逻辑接口  期初建账的业务逻辑接口  账户的增删改查的逻辑接口 |
| businesslogicservice.intermediateblservice.EnplaningBLService  businesslogicservice.intermediateblservice.EntrainingBLService  businesslogicservice.intermediateblservice.EntruckingBLService  businesslogicservice.intermediateblservice.TransferingBLService | 飞机装运管理的业务逻辑接口  火车装运管理的业务逻辑接口  汽车装运管理的业务逻辑接口  中转接收的业务逻辑接口 |
| businesslogicservice.repertoryblservice.RepertoryBLService | 出库、入库、库存查看、库存盘点、库存信息初始化的业务逻辑接口 |
| businesslogicservice.manageblservice.OrganizationBLService  businesslogicservice.manageblservice.PerWageBLService  businesslogicservice.manageblservice.BasicSalaryBLService  businesslogicservice.manageblservice.CityDistanceBLService  businesslogicservice.manageblservice.CostBLService | 机构信息的增删改查的业务逻辑接口  每次工资信息的增删改查的业务逻辑接口  基础月薪信息的增删改查的业务逻辑接口  城市距离信息的增删改查的业务逻辑接口  运费系数信息的增删改查的业务逻辑接口 |

## 4.2业务逻辑层的分解

### 4.2.1expressbl模块

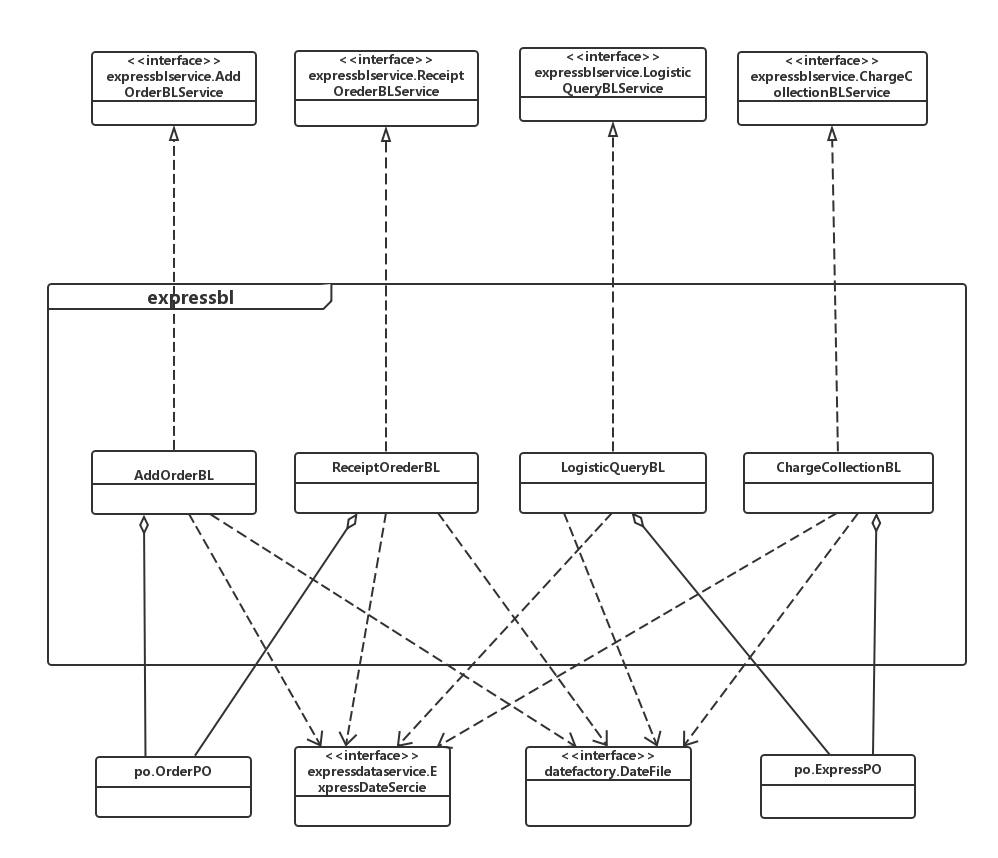
#### 4.1.1.1模块概述

expressbl模块实现了快递员的新增订单，完成订单，收费记录，订单查询功能，具体功能需求和非功能需求请参见需求规格说明文档和体系结构设计文档。

#### 4.1.1.2整体结构

expressbl模块实现了展示层与业务逻辑层之间的expressblservice接口以及业务逻辑层和数据层之间的expressdataservice接口。提供了查询订单的calculate和addOrder方法,收件信息输入的getOrderInfo和receiptOrder方法,收费信息汇总的getChargeInfo和chargeCollection方法以及查询订单的query方法。查询订单，收件信息输入，收费信息汇总和查询订单用例分别对应类LogisticQueryBL,ReceiptOrderBL,ChargeCollectionBL和AddOrderBL类，

expressbl模块的设计如图1-1所示



**图1-1 expressbl模块各个类的设计**

expressbl各个类的职责如表1-1所示

**表1-1 expressbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| LogisticQueryBL | 负责实现快递员进行物流查询的服务 |
| ReceiptOrderBL | 负责实现快递员进行收件信息输入的服务 |
| ChargeCollectionBL | 负责实现快递员进行收费信息汇总的服务 |
| AddOrderBL | 负责实现快递员新增订单的服务 |

#### 4.1.1.3模块内部类的接口规范

LogisticQueryBL,ReceiptOrderBL,ChargeCollectionBL和AddOrderBL的接口规范如表1-2，表1-3，表1-4和表1-5所示

**表1-2 LogisticQueryBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| LogisticQueryBL.query | 语法 | public OrderVO query(long orderNum) | |
|  | 前置条件 | 快递员经过验证有查询订单的权利，输入了有效的订单号 | |
| 后置条件 | 系统查找订单是否存在，如果找到就返回订单信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DataFactory.getOrderInfo(long orderNum) | | | 得到订单信息 |

**表1-3 ReceiptOrderBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ReceiptOrderBL.getOrderInfo | 语法 | public OrderVO getOrderInfo(long orderNum) | |
| 前置条件 | 快递员经过验证有确认订单的权利，并要求查看订单信息 | |
| 后置条件 | 系统根据输入的 | |
| ReceiptOrderBL.receiptOrder | 语法 | public boolean receiptOrder(OrderVO vo) | |
| 前置条件 | 快递员经过验证有确认订单的权利，并要求确认订单 | |
| 后置条件 | 系统根据输入的订单接收信息更新订单，并返回更新结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressDateService.receiptOrder(OrderPO po) | | | 完成订单 |

**表1-4 ChargeCollectionBL的接口规范**

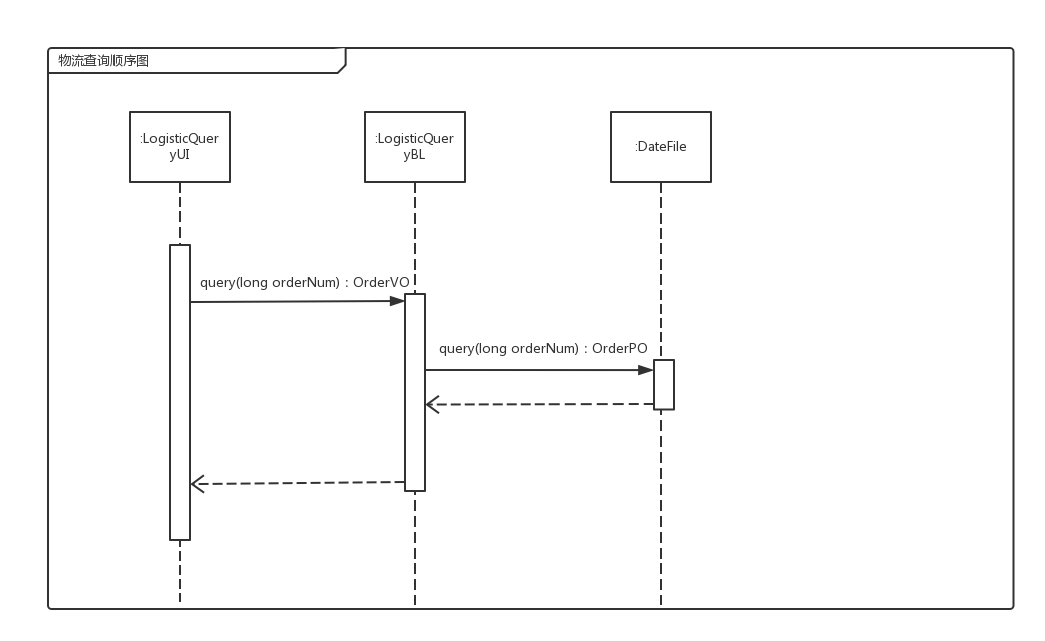
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ChargeCollectionBL.getChargeInfo | 语法 | public ExpressVO getChargeInfo() | |
| 前置条件 | 快递员经过验证有汇总收费的权利，并要求查看当前收费情况 | |
| 后置条件 | 系统更新快递员信息，并更具更新返回结果 | |
| ChargeCollectionBL.chargeCollection | 语法 | public boolean chargeCollection(ExpressVO vo) | |
| 前置条件 | 快递员经过验证有汇总收费的权利，并输入了订单号 | |
| 后置条件 | 系统根据输入的订单号返回对应的订单信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressDataService.chargeCollection(ExpressPO po) | | | 更新快递员的收费信息总额 |

**表1-5 AddOrderBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| AddOrderBL.addOrder | 语法 | public boolean addOrder(OrderVO vo) | |
| 前置条件 | 快递员经过验证有添加订单的权利，输入了有效的待添加的订单信息 | |
| 后置条件 | 系统更新对应的订单信息，并根据更新结果返回 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DataFactory.addOrder(OrderPO po) | | | 新增订单 |

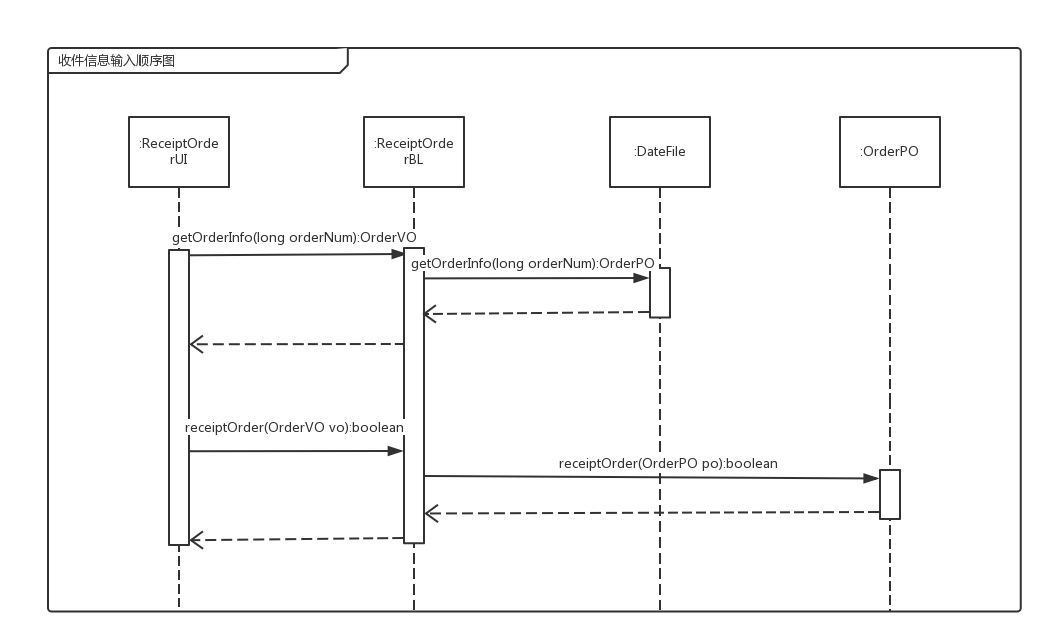
#### 4.1.1.4业务逻辑层的动态模型

图1-2为LogisticQueryBL领域对象想要新增订单时的顺序图



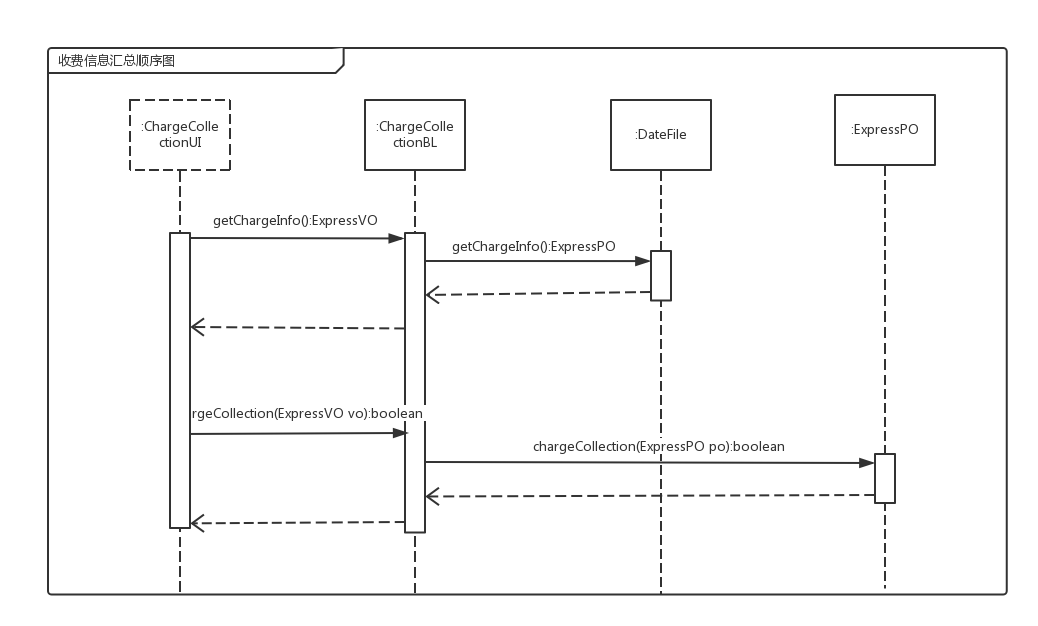
**图1-2物流查询的顺序图**

图1-3为ReceiptOrderBL领域对象想要进行收件信息输入时的顺序图



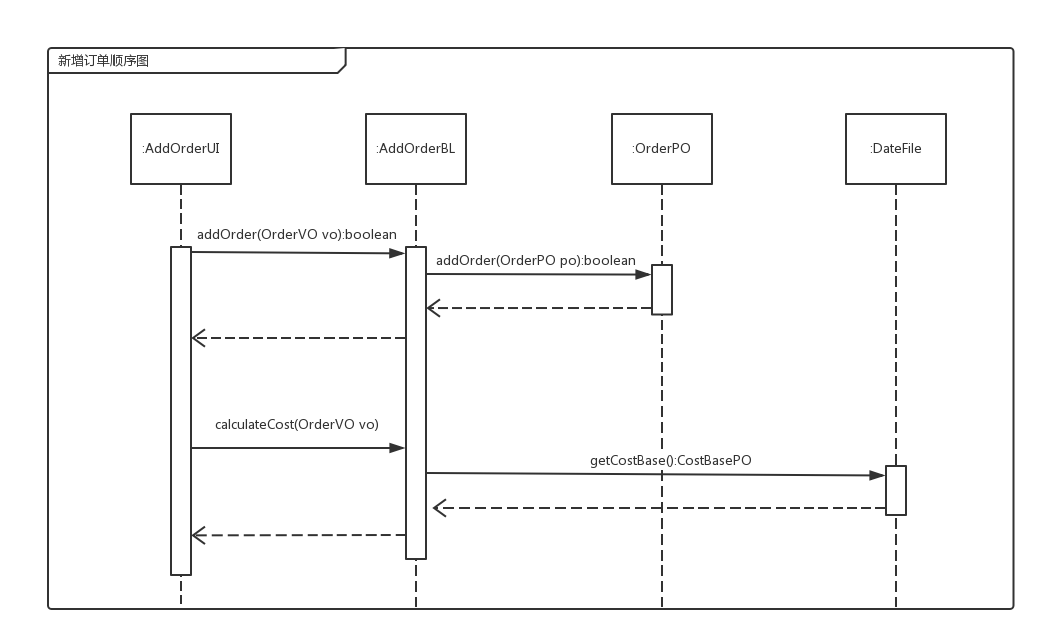
**图1-3 收件信息输入的顺序图**

图1-4为ChargeCollectionBL领域对象想要进行收费信息汇总时的顺序图



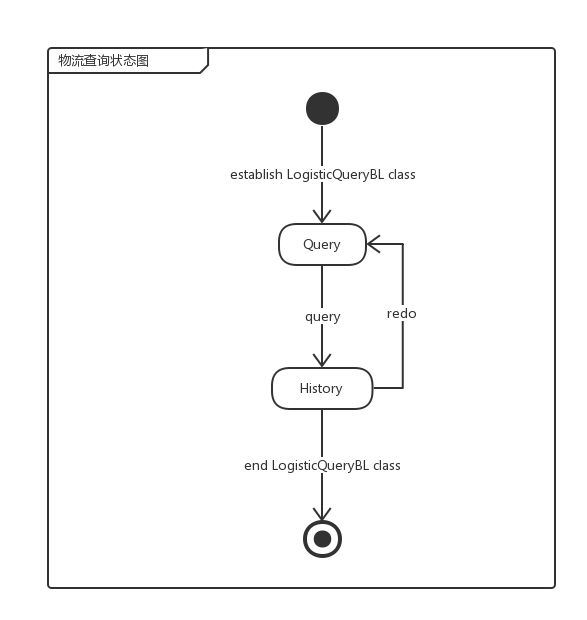
**图1-4 收费信息汇总的顺序图**

图1-5为AddOrderBL领域对象想要新增订单时的顺序图



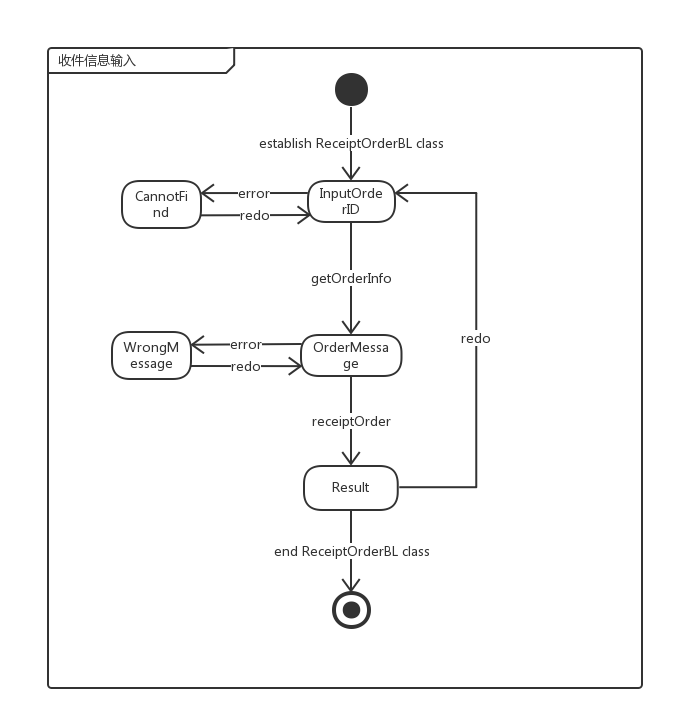
**图1-5新增订单的顺序图**

如图1-6所示，AddOrderBL对象状态图描述了AddOrderBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishAddOrderBL方法被 UI调用，AddOrderBL根据不同的指令进入不同状态。



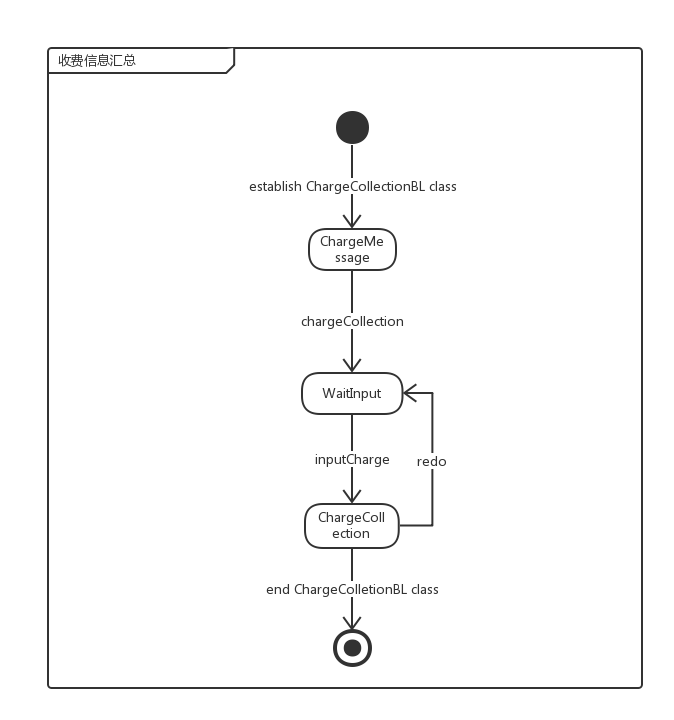
**图1-6 物流查询的状态图**

如图1-7所示，ReceiptOrderBL对象状态图描述了ReceiptOrderBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishReceiptOrderBL方法被 UI调用，ReceiptOrderBL根据不同的指令进入不同状态。



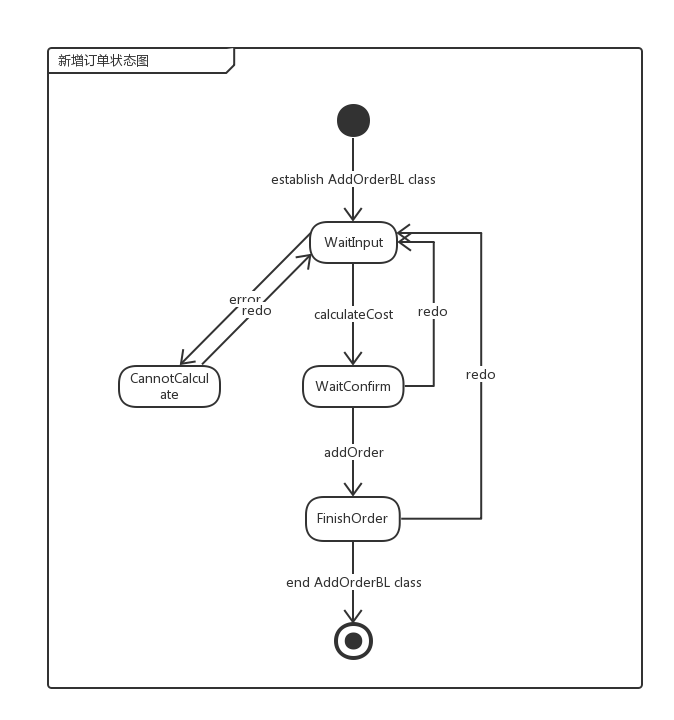
**图1-7 收件信息输入的状态图**

如图1-8所示，ChargeCollectionBL对象状态图描述了ChargeCollectionBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishChargeCollectionBL方法被 UI调用，ChargeCollectionBL根据不同的指令进入不同状态。



**图1-8 收费信息输入的状态图**

如图1-9所示，AddOrderBL对象状态图描述了AddOrderBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishAddOrderBL方法被 UI调用，AddOrderBL根据不同的指令进入不同状态。



**图1-9 新增订单的状态图**

### 4.2.2businessbl模块

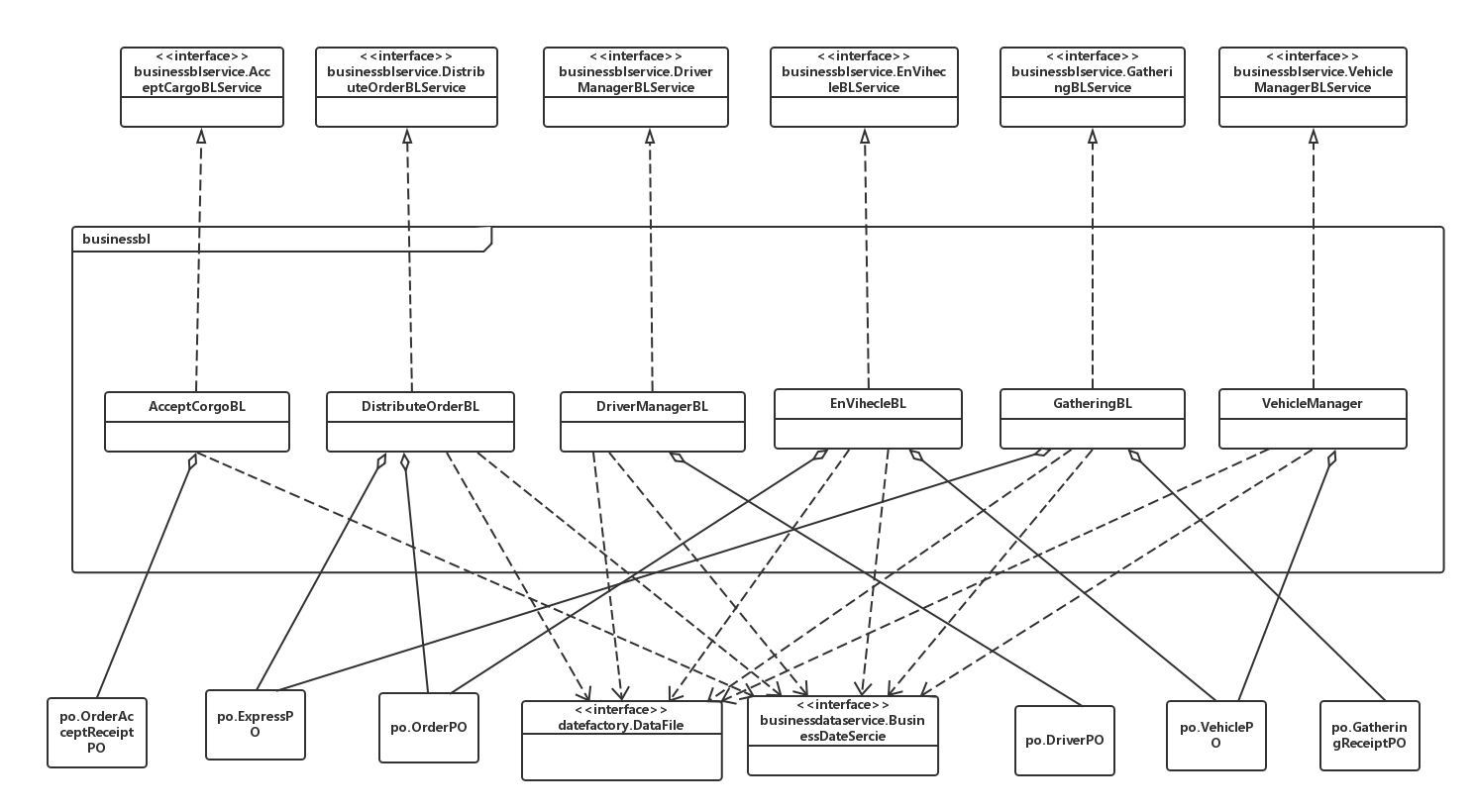
#### 4.2.2.1模块概述

businessbl模块实现了营业厅业务员的接收货物，装车管理，司机信息管理，车辆信息管理，派件，收款汇总功能，具体功能需求和非功能需求请参见需求规格说明文档和体系结构设计文档。

#### 4.2.2.2整体结构

businessbl模块实现了展示层和业务逻辑层之间的businessblservice接口以及业务逻辑层和数据层之间的businessdataservice接口。提供了接收货物的acceptCargo方法，装车管理的autoTruckLoading方法，司机信息管理的addDriver，deleteDriver，modifyDriver方法，车辆信息管理的addVehicle，deleteVehicle，modifyVehicle方法，派件的distributeOrder方法以及收款汇总的gathering方法，接收货物，装车管理，司机信息管理，车辆信息管理，派件，收款汇总用例分别对应类AcceptCargoBL，EnVehicleBL，DriverManagerBL，VehicleManagerBL，DistributeOrderBL，GatheringBL类。

businessbl模块的设计如图2-1所示



**图2-1 businessbl模块各个类的设计**

businessbl各个类的职责如表2-1所示

**表2-1 expressbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| AcceptCargoBL | 负责实现营业厅业务员接收货物的服务 |
| EnVehicleBL | 负责实现营业厅业务员的进行装车管理服务 |
| DriverManagerBL | 负责实现营业厅业务员的司机信息管理服务 |
| VehicleManagerBL | 负责实现营业厅业务员的车辆信息管理服务 |
| DistributeOrderBL | 负责实现营业厅业务员派件的服务 |
| GatheringBL | 负责实现营业厅业务员的收款汇总服务 |

#### 4.2.2.3businessbl模块内部类的接口规范

AcceptCargoBL，EnVehicleBL，DriverManagerBL，VehicleManagerBL，DistributeOrderBL，GatheringBL的接口规范如表2-2，表2-3，表2-4，表2-5，表2-6和表2-7所示

**表2-2 AcceptCargoBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| AcceptCargoBL.acceptCargo | 语法 | public boolean acceptCargo( OrderAcceptReceiptVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行接收货物的权利，并输入了所有货物的订单号 | |
| 后置条件 | 系统生成新的营业厅到达单并更新营业厅到达单列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BusinessDataService.acceptCargo(OrderAcceptReceiptPO po) | | | 生成营业厅接收单 |

**表2-3 EnVehicleBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| EnVehicleBL.autoTruckLoading | 语法 | public String autoTruckLoading() | |
| 前置条件 | 营业厅业务员经过验证有进行装车管理的权利，并要求进行装车管理任务 | |
| 后置条件 | 系统自动根据已有信息生成装车单，并更新到装车单列表 | |
| EnVehicleBL.getFreeVehicles | 语法 | public ArrayList<VehicleVO> getFreeVehicles() | |
| 前置条件 | 营业厅业务员经过验证有进行装车管理的权利，并要求查看当前的空闲车辆 | |
| 后置条件 | 系统返回当前的空闲车辆的信息 | |
| EnVehicleBL.getTransferOrders | 语法 | public ArrayList<OrderVO>getTransferOrders() | |
| 前置条件 | 营业厅业务员经过验证有进行装车管理的权利，并要求查看当前待转发的订单 | |
| 后置条件 | 系统返回订单待转发的订单信息 | |
| EnVehicleBL.addEnVehicleReceiptPO | 语法 | public boolean addEnVehicleReceiptPO(EnVehicleReceiptPO po) | |
| 前置条件 | 系统完成自动装车任务，将生成的装车单添加到 | |
| 后置条件 | 系统返回更新结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DateFactory.getFreeVehicle() | | | 得到所有空闲的车辆的信息 |
| DateFactory.getSendOrder() | | | 得到所有待派送的订单信息 |

**表2-4 DriverManagerBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| DriverManagerBL.addVehicle | 语法 | public boolean addVehicle(Vehicle vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入了正确的车辆信息 | |
| 后置条件 | 系统根据输入的车辆信息更新车辆列表，并返回更新结果 | |
| DriverManagerBL.getDriverInfo | 语法 | Public ArrayList<VehicleVO>getVehicleInfo(String ID) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入正确的车辆编号 | |
| 后置条件 | 系统返回已有车辆的信息 | |
| DriverManagerBL.deleteDriver | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求删除指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| DriverManagerBL.modifyDriver | 语法 | public boolean modifyVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求修改 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DataFactory.getDriverInfo() | | | 得到ID对应的司机信息 |
| BusinessDataService.addDriver(DriverPO po) | | | 增加司机信息 |
| BusinessDataService.deleteDriver(DriverPO po) | | | 删除司机信息 |
| BusinessDataService.modifyDriver(DriverPO po) | | | 修改司机信息 |

**表2-5 VehicleManager的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| VehicleManagerBL.modifyVehicle | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求改正指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| VehicleManagerBL.addVehicle | 语法 | public boolean addVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入了正确的车辆信息 | |
| 后置条件 | 系统根据输入的车辆信息更新车辆列表，并返回更新结果 | |
| VehicleManagerBL.getVehicleInfo | 语法 | public ArrayList<VehicleVO>getVehicleInfo(String ID) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入了车辆编号 | |
| 后置条件 | 系统返回已有车辆的信息 | |
| VehicleManagerBL.deleteVehicle | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求删除指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BusinessDataService.addVehicle(VehiclePO po) | | | 增加车辆信息 |
| BusinessDataService.deleteVehicle(VehiclePO po) | | | 删除车辆信息 |
| BusinessDataService.modifyVehicle(VehiclePO po) | | | 修改车辆信息 |
| DataFactory.getVehicleInfo(String ID) | | | 得到车辆信息 |

**表2-6 DistributeOrderBL的接口规范**

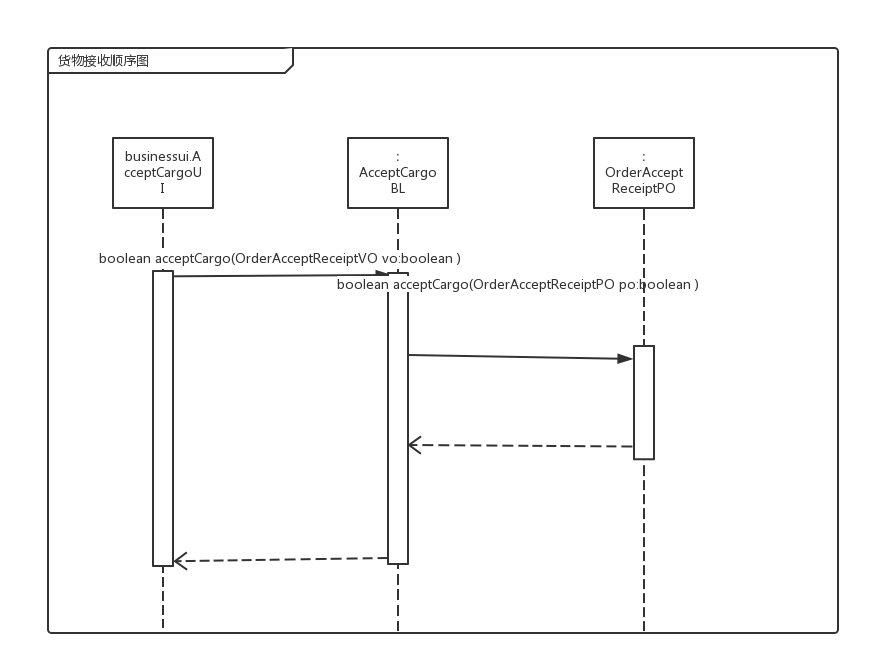
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| DistributeOrderBL.distributeOrder | 语法 | public String distributeOrder() | |
| 前置条件 | 营业厅业务员经过验证有进行分配订单的权利，并要求进行分配订单任务 | |
| 后置条件 | 系统根据已有信息生成装车单，并返回分配结果 | |
| DistributeOrderBL.getExpressInfos | 语法 | public ArrayList<ExpressVO> getExpressInfos() | |
| 前置条件 | 营业厅业务员经过验证有进行分配订单的权利，并要求查看该营业厅所有快递员的信息 | |
| 后置条件 | 系统返回该营业厅所有快递员的信息 | |
| DistributeOrderBL.getSendOrde | 语法 | public ArrayList<OrderVO>getSendOrder() | |
| 前置条件 | 营业厅业务员经过验证有进行分配订单的权利，并要求查看该营业厅所有待派发的订单信息 | |
| 后置条件 | 系统返回该营业厅所有待派发的订单信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DateFactory.getExpressInfos() | | | 获得该营业厅所有快递员信息 |
| DateFactory.getSendOrder() | | | 得到所有待派送的订单信息 |

**表2-7 GatheringBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| GatheringBL.getChargeInfo | 语法 | public ArrayList<ExpressVO>getChargeInfo() | |
| 前置条件 | 营业厅业务员经过验证有进行收款汇总的权利，并要求查看所有营业厅业务员的收款与订单信息 | |
| 后置条件 | 系统返回该营业厅所有快递员的信息（订单与收款） | |
| GatheringBL.gathing | 语法 | public double gathering(GatheringReceiptVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行收款汇总的权利，并要求进行收款汇总任务 | |
| 后置条件 | 系统汇总收款并返回结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DataFactory.getExpressInfos() | | | 获得该营业厅所有快递员信息 |
| BusinessDataService.gathering(GatheringReceiptPO po) | | | 将快递员的收费汇总并生成收款单 |

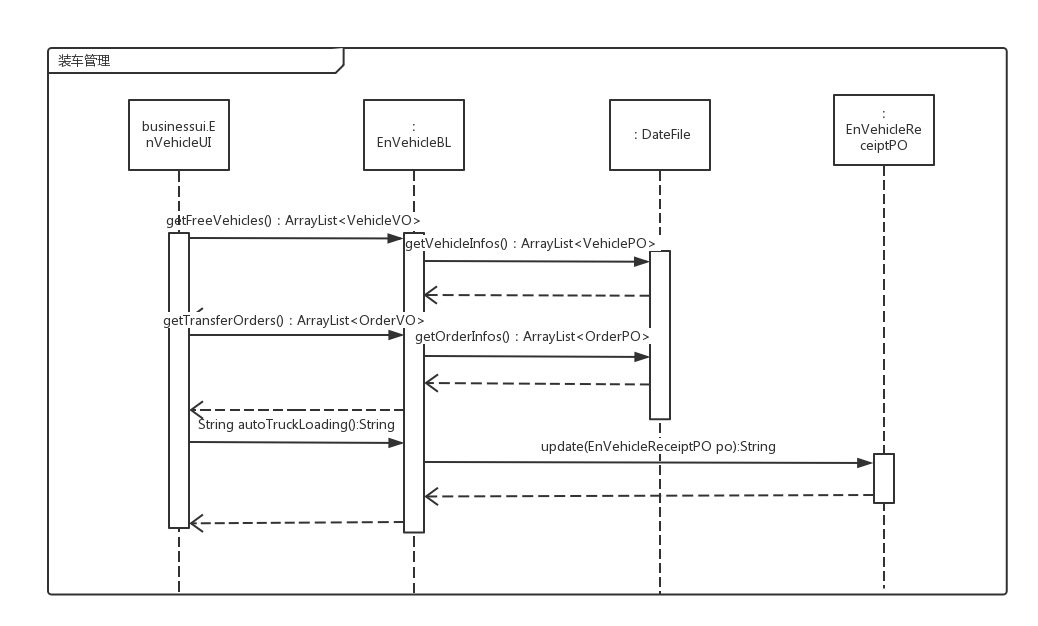
#### 4.2.2.4业务逻辑层的动态模型

图2-2为 AcceptCargoBL领域对象想要进行货物接收时的顺序图

****

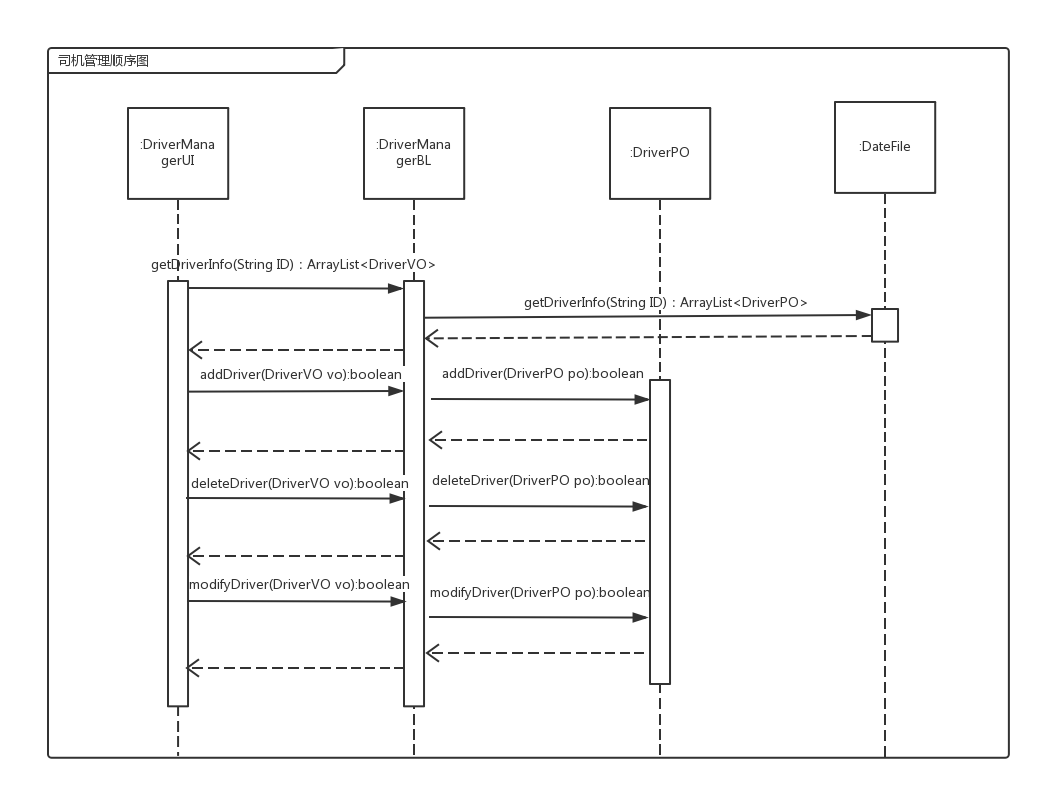
**图2-2 货物接收的顺序图**

图2-3为 EnVehicleBL领域对象想要进行装车管理时的顺序图



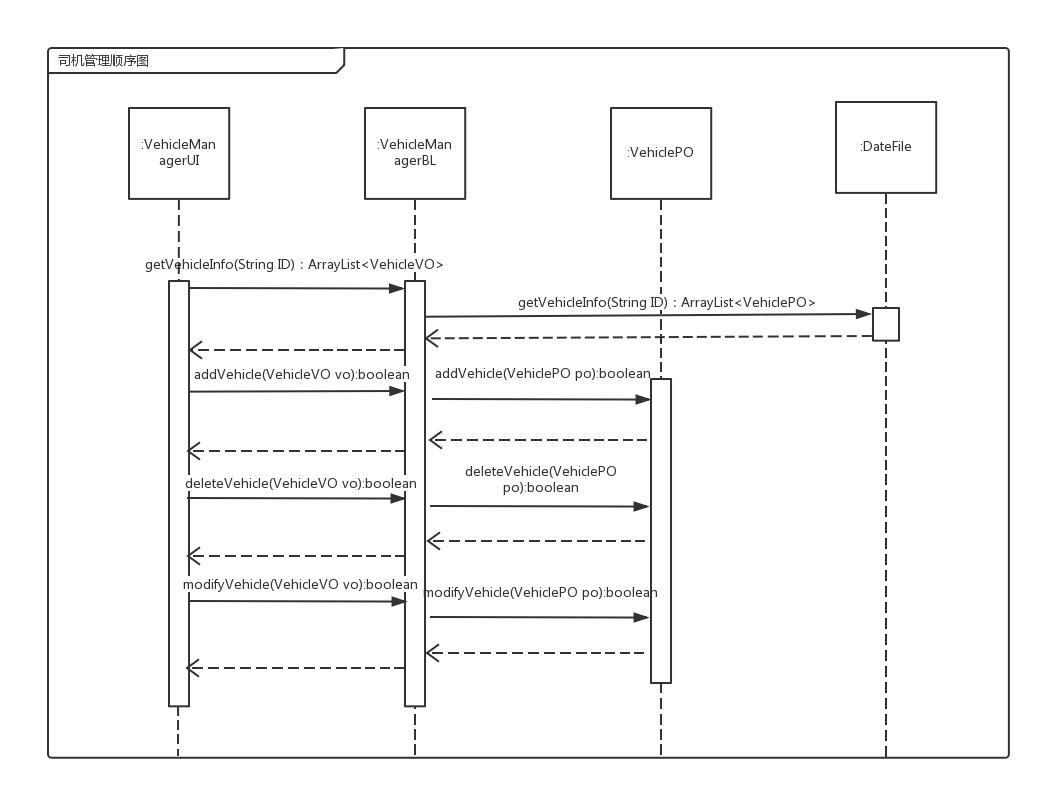
**图2-3 装车管理的顺序图**

图2-4为 DriverManagerBL领域对象想要进行司机信息管理时的顺序图



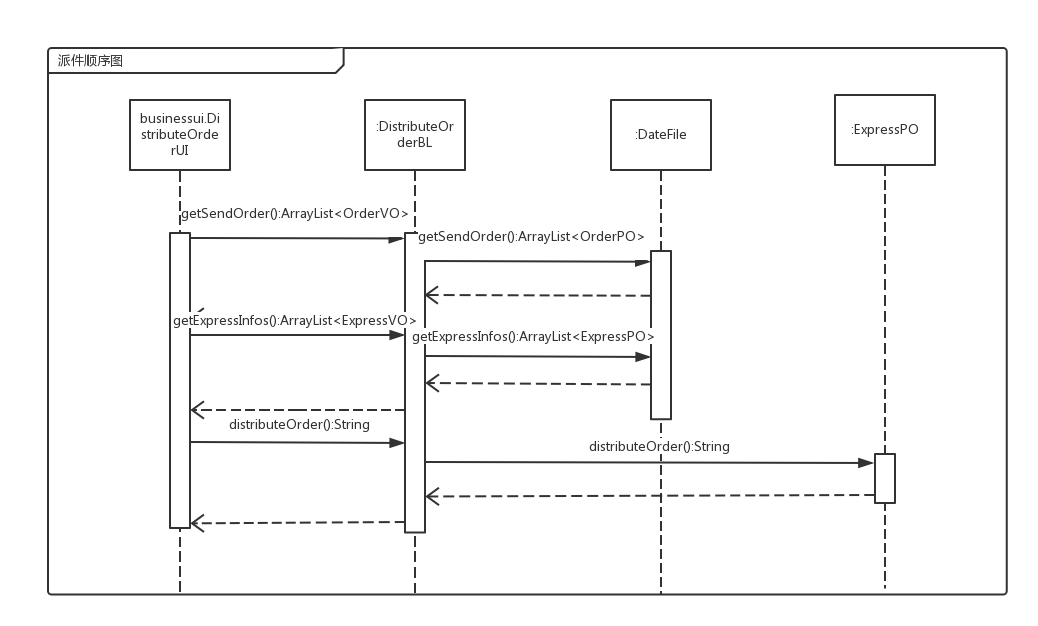
**图2-4 司机信息管理的顺序图**

图2-5为 VehicleManagerBL领域对象想要车辆信息管理时的顺序图



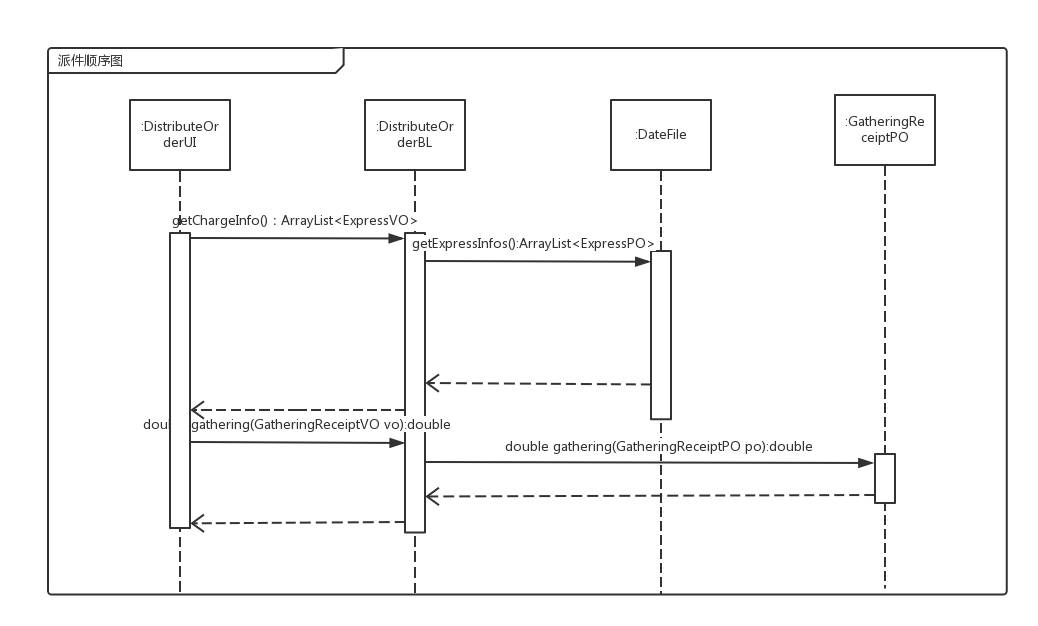
**图2-5 车辆信息管理的顺序图**

图2-6为 DistributeOrderBL领域对象想要派件时的顺序图



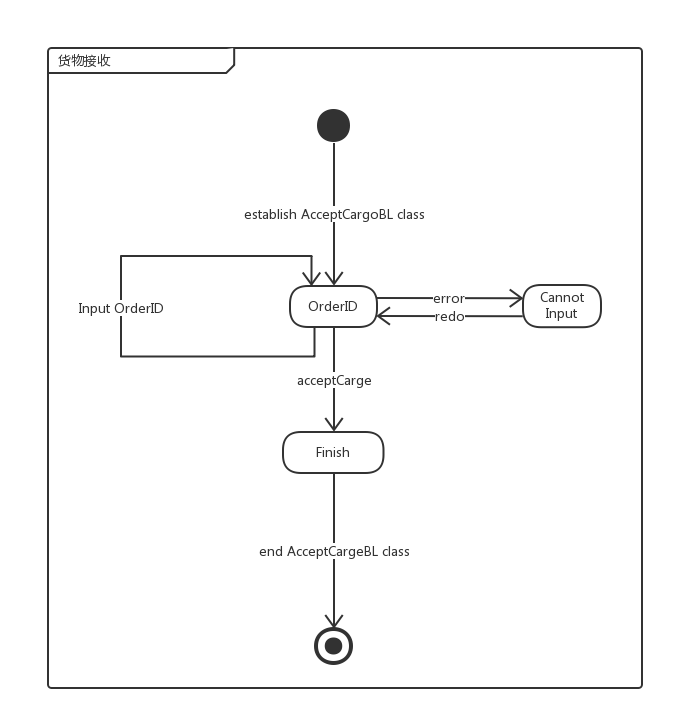
**图2-6 派件的顺序图**

图2-7为 GatheringBL领域对象想要进行收款汇总时的顺序图



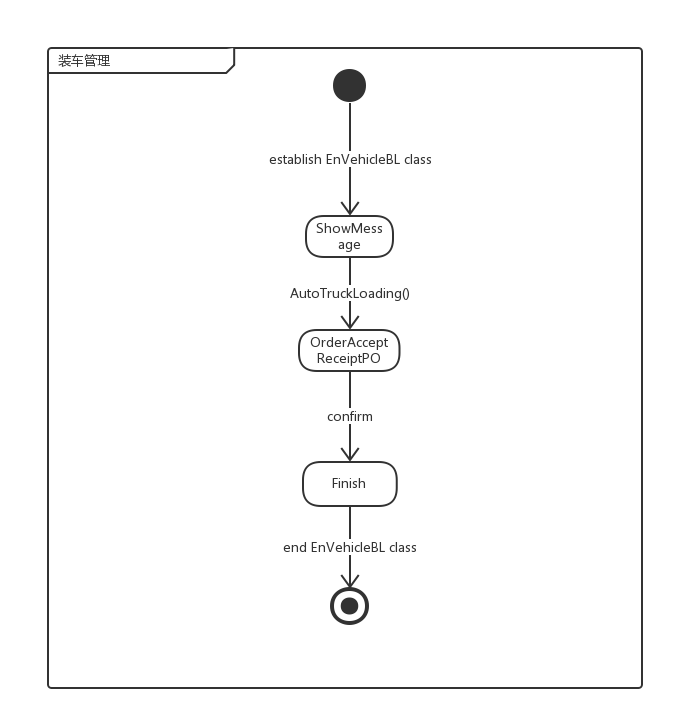
**图2-7 收款汇总的顺序图**

如图1-8所示，AcceptCargoBL对象状态图描述了AcceptCargoBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishAcceptCargoBL方法被 UI调用，AcceptCargoBL根据不同的指令进入不同状态。

****

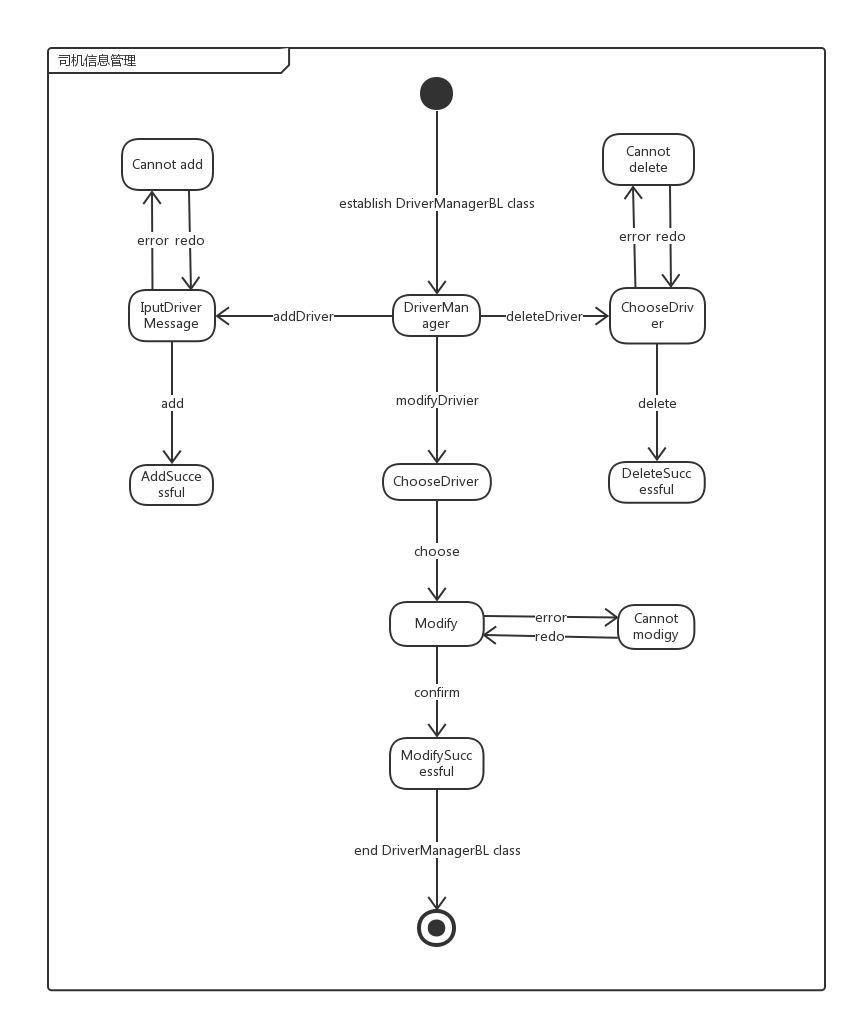
**图1-8 接收货物的状态图**

如图1-9所示，EnVehicleBL对象状态图描述了EnVehicleBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishEnVehicleBL方法被 UI调用，EnVehicleBL根据不同的指令进入不同状态。



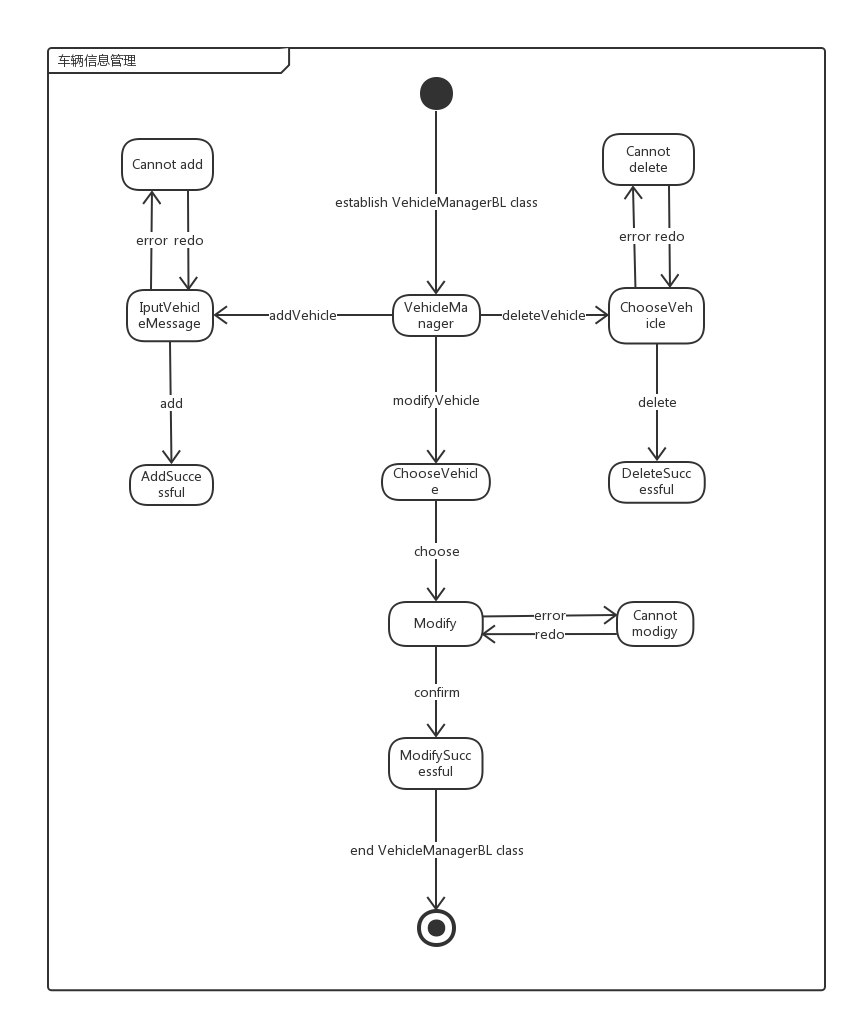
**图1-9 装车管理的状态图**

如图1-10所示，DriverManagerBL对象状态图描述了DriverManagerBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishDriverManagerBL方法被 UI调用，DriverManagerBL根据不同的指令进入不同状态。



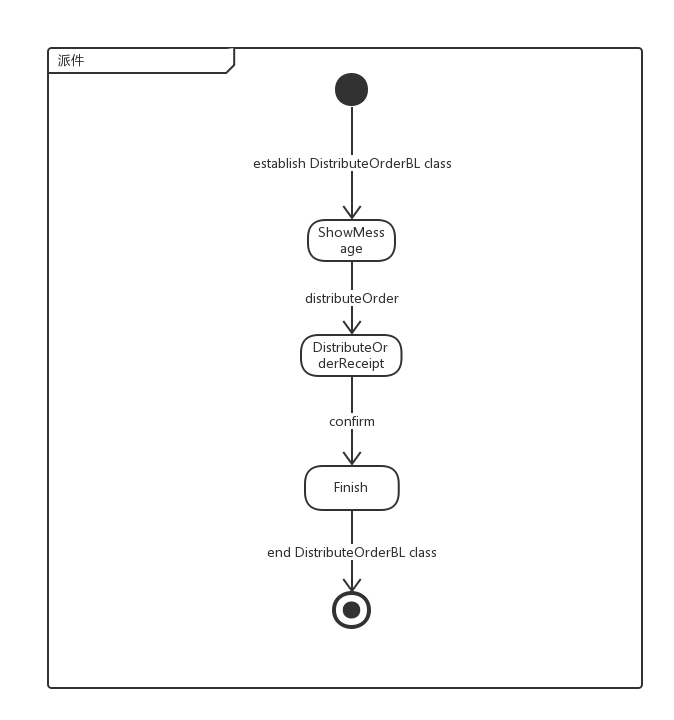
**图1-10 司机信息管理的状态图**

如图1-11所示，VehicleManagerBL对象状态图描述了VehicleManagerBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishVehicleManagerBL方法被 UI调用，VehicleManagerBL根据不同的指令进入不同状态。



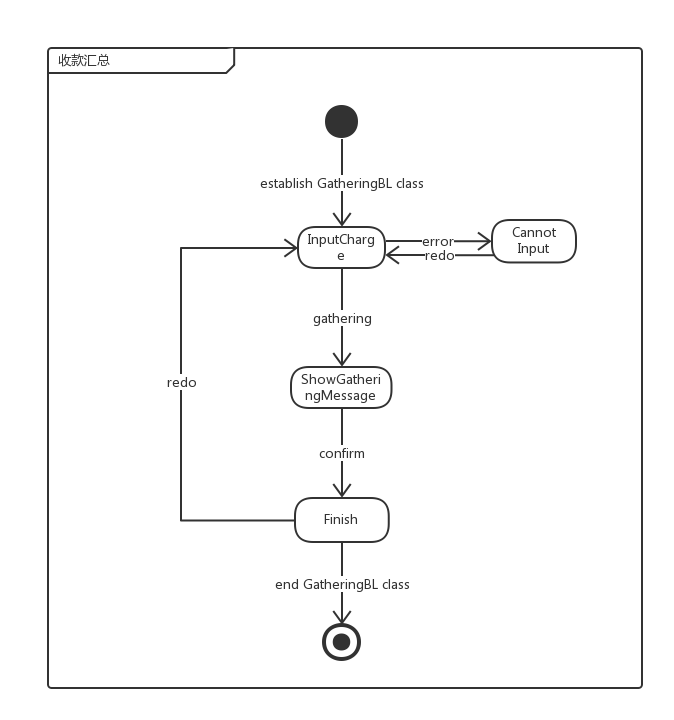
**图1-11 车辆信息管理的状态图**

如图1-12所示，DistributeOrderBL对象状态图描述了DistributeOrderBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishDistributeOrderBL方法被 UI调用，DistributeOrderBL根据不同的指令进入不同状态。



**图1-12 派件的状态图**

如图1-13所示，GatheringBL对象状态图描述了GatheringBL对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishGatheringBL方法被 UI调用，GatheringBL根据不同的指令进入不同状态。

****

**图1-13 收款汇总的状态图**

### 4.2.3intermediatebl模块

4.2.3.1模块概述

4.2.3.2整体结构

4.2.3.3模块内部类的接口规范

4.2.3.4业务逻辑层的动态模型

4.2.3.5业务逻辑层的设计原理

### 4.2.4receiptbl模块

4.2.4.1模块概述

4.2.4.2整体结构

4.2.4.3模块内部类的接口规范

4.2.4.4业务逻辑层的动态模型

4.2.4.5业务逻辑层的设计原理

### 4.2.5repertorybl模块

4.2.5.1模块概述

4.2.5.2整体结构

4.2.5.3模块内部类的接口规范

4.2.5.4业务逻辑层的动态模型

4.2.5.5业务逻辑层的设计原理

### 4.2.6financebl模块

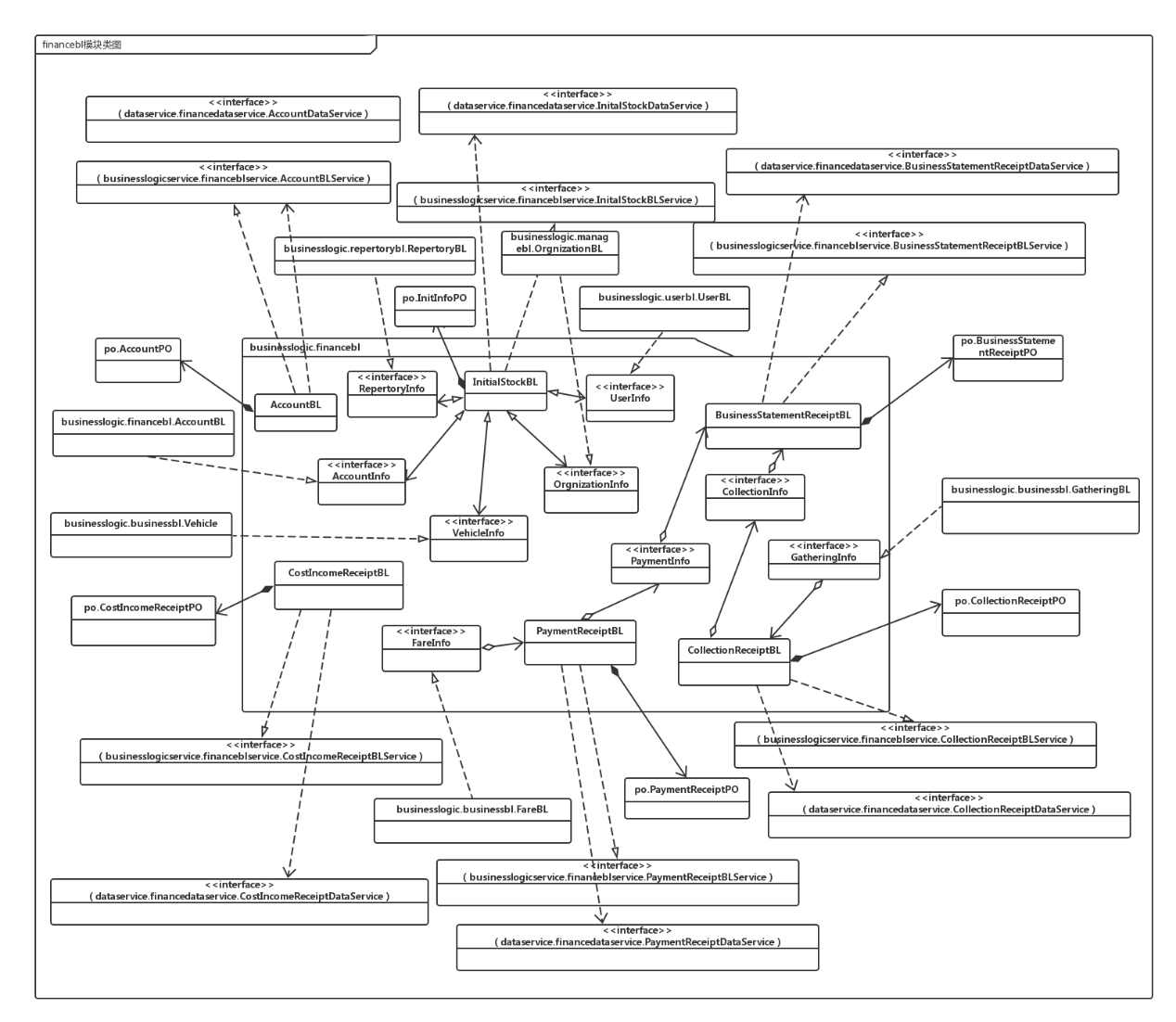
#### 4.2.6.1模块概述

financebl模块承担的需求参见ELS需求规格说明文档功能需求及相关非功能性需求。

financebl模块的职责及接口参见ELS体系结构设计文档。

#### 4.2.6.2整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们添加businesslogicservice.financeblservice.CollectionReceiptBLService，businesslogicservice.financeblservice.PaymentReceiptBLService，businesslogicservice.financeblservice.CostIncomeReceiptBLService，businesslogicservice.financeblservice.BusinessStatementReceiptBLService，businesslogicservice.financeblservice.InitalStockBLService和businesslogicservice.financeblservice.AccountBLService接口。因为业务逻辑处理较少，根据专家模式没有添加controller类。业务逻辑层和数据层之间添加dataservice.financedataservice. CollectionReceiptDataService，dataservice.financedataservice. PaymentReceiptDataService，dataservice.financedataservice. CostIncomeReceiptDataService，dataservice.financedataservice. BusinessStatementReceiptDataService，dataservice.financedataservice. InitalStockDataService，dataservice.financedataservice. AccountDataService接口。CollectionReceiptPO,PaymentReceiptPO,BusinessStatementReceiptPO，CostIncomeReceiptPO，AccountPO和InitInfoPO是作为财务管理的持久化对象被添加到设计模型中去的。



**图6-1 financebl模块各个类的设计**

Financebl各个类的职责如表6-1所示

**表6-1 financebl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| CollectionReceiptBL | 入款单的领域模型对象，拥有一个入款单所持有的编号，操作员，入款金额等信息，可帮助完成结算管理界面所需要的服务。 |
| PaymentReceiptBL | 付款单的领域模型对象，拥有一个付款单所持有的编号，操作员，付款金额等信息，可帮助完成新建付款单界面所需要的服务。 |
| CostIncomeReceiptBL | 成本收益表的领域模型对象，拥有一个成本收益表所持有的编号，操作员，收入，支出金额等信息，可帮助完成新建成本收益表界面所需要的服务。 |
| BusinessStatementReceiptBL | 经营情况表的领域模型对象，拥有一个经营情况表所持有的入款记录，付款记录，编号等信息，可以帮助完成经营情况表界面所需要的服务。 |
| InitialStockBL | 期初建账的领域模型对象，拥有一个套账所持有的人员、机构、车辆、库存和账户等信息，可以帮助完成期初建账界面所需要的服务。 |
| AccountBL | 账户的领域模型对象，拥有账户的名称、金额等信息，可以帮助完成账户管理界面所需要的服务。 |

#### 4.2.6.3模块内部类的接口规范

CollectionReceiptBL，PaymentReceiptBL，CostIncomeReceiptBL，BusinessStatementReceiptBL，InitialStockBL和AccountBL的接口规范如表所示。

**表6-2 CollectionReceiptBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CollectionReceiptBL.  creatCollection | 语法 | | public int creatCollection (CollectionReceiptVO vo); |
| 前置条件 | | 用户要求新建入款单并输入有效的信息 |
| 后置条件 | | 向系统输入新的入款单信息 |
| CollectionReceiptBL.  getCollection | 语法 | | public CollectionReceiptVO getCollection (String s); |
| 前置条件 | | 用户具有查看入款单的权限 |
| 后置条件 | | 系统显示符合条件的入款单 |
| CollectionReceiptBL.  getAllCollection | 语法 | | public ArrayList<CollectionReceiptVO> getAllCollection(); |
| 前置条件 | | 用户有查看入款单的权限 |
| 后置条件 | | 系统显示所有入款单列表 |
| CollectionReceiptBL.  getGathering | 语法 | | public ArrayList<GatheringVO> getGathering(String HallID,String Time,GatheringVO vo); |
| 前置条件 | | 用户输入日期和营业厅编号并要求查看收款单 |
| 后置条件 | | 系统显示符合条件的收款单列表 |
| CollectionReceiptBL.  getMoney | 语法 | | public double[] getMoney(GatheringVO vo); |
| 前置条件 | | 所有符合条件的收款单已经获取 |
| 后置条件 | | 系统返回收款单的金额 |
| CollectionReceiptBL.  getTotalMoney | 语法 | | public double getTotalMoney(int[] money); |
| 前置条件 | | 用户要求合计金额 |
| 后置条件 | | 系统返回合计后的金额 |
| CollectionReceiptBL. getCollectionListID | 语法 | | public String getCollectionListID(); |
| 前置条件 | | 用户制定入款单时需要根据今天已经制定的入款单来自动生成入款单编号 |
| 后置条件 | | 系统显示自动生成的入款单ID |
| 需要的服务（需接口） | | | |
| FinanceDataService.creatCollection(CollectionReceiptPO po) | | 插入单一持久化对象CollectionReceiptPO | |
| FinanceDataService.getAllCollection() | | 得到持久化对象CollectionReceiptPO的ArrayList | |
| FinanceDataService.getGathering(String HallID,String Time) | | 得到持久化对象GatheringReceiptPO的ArrayList | |
| FinanceDataService.getMoney(GatheringReceiptPO po) | | 得到持久化对象GatheringReceiptPO中的金额数 | |
| FinanceDataService.getTotalMoney(int[] money) | | 得到持久化对象GatheringReceiptPO中的金额总和 | |
| FinanceDataService.getNum() | | 得到当天存储的持久化对象CollectionReceiptPO的个数 | |
| FinanceDataService.findByID(String ID) | | 根据ID返回对应的一个持久化对象CollectionReceiptPO | |
| FinanceDataService.modify(CollectionReceiptPO po) | | 修改单一持久化对象CollectionReceiptPO的信息 | |

**表6-3 PaymentReceiptBL的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PaymentReceiptBL.  createPayment | 语法 | public int creatPayment(PaymentReceiptVO vo); |
| 前置条件 | 用户要求新建付款单并输入有效信息 |
| 后置条件 | 向系统输入新的付款单信息 |
| PaymentReceiptBL.  getPayment | 语法 | Public PaymentReceiptVO getPayment(String s); |
| 前置条件 | 用户有查看付款单的权限 |
| 后置条件 | 系统显示符合条件的付款单列表 |
| PaymentReceiptBL.  getAllPayment | 语法 | public ArrayList<PaymentReceiptVO> getAllPayment(); |
| 前置条件 | 用户有查看付款单的权限 |
| 后置条件 | 系统显示所有付款单列表 |
| PaymentReceiptBL.  getPaymentID | 语法 | public String getPaymentListID(); |
| 前置条件 | 用户制定付款单时需要根据今天已经制定的付款单来自动生成付款单编号 |
| 后置条件 | 系统显示自动生成的付款单ID |
| 需要的服务（需接口） | | |
| FinanceDataService. creatPaymentReceipt  (PaymentReceiptPO po) | | 插入单一持久化对象 PaymentReceiptPO |
| FinanceDataService. getAllPaymentReceipt() | | 得到持久化对象 PaymentReceiptPO的ArrayList |
| FinanceDataService.  getNum() | | 得到当天存储的持久化对象PaymentReceiptPO的个数 |
| FinanceDataService. findByID(String ID) | | 根据ID返回对应的一个持久化对象PaymentReceiptPO |
| FinanceDataService. modify(PaymentReceiptPO po) | | 修改单一持久化对象PaymentReceiptPO的信息 |

**表6-4 CostIncomeReceiptBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CostIncomeReceiptBL.  createCostIncomeList | 语法 | | public int creatCostIncomeList(CostIncomeReceiptVO vo); |
| 前置条件 | | 用户要求新建成本收益表并输入有效信息 |
| 后置条件 | | 向系统中输入新的成本收益表信息 |
| CostIncomeReceiptBL.  getCostIncmeList | 语法 | | public CostIncomeReceiptVO getCostIncomeList(String s); |
| 前置条件 | | 用户有查看成本收益表的权限 |
| 后置条件 | | 系统返回符合条件的成本收益表 |
| CostIncomeReceiptBL.  getAllCostIncomeList | 语法 | | public ArrayList<CostIncomeReceiptVO> getAllCostIncomeList(); |
| 前置条件 | | 用户有查看成本收益表的权限 |
| 后置条件 | | 系统返回所有成本收益表列表 |
| CostIncomeReceiptBL.  getCollection | 语法 | | public ArrayList<CollectionReceiptVO> getCollection(); |
| 前置条件 | | 用户有查看入款单的权限 |
| 后置条件 | | 系统返回所有入款单列表 |
| CostIncomeReceiptBL.  getPayment | 语法 | | public ArrayList<PaymentReceiptVO>getPayment(); |
| 前置条件 | | 用户有查看付款单的权限 |
| 后置条件 | | 系统返回所有付款单列表 |
| CostIncomeReceiptBL.  getCost | 语法 | | public double getCost(PaymentReceiptVO vo); |
| 前置条件 | | 已获取付款单 |
| 后置条件 | | 系统返回付款单金额总和 |
| CostIncomeReceiptBL.  getIncome | 语法 | | public double getIncome(CollectionReceiptVO vo); |
| 前置条件 | | 已获取入款单 |
| 后置条件 | | 系统返回入款单金额总和 |
| CostIncomeReceiptBL.  getProfit | 语法 | | public double getProfit(double income,double cost); |
| 前置条件 | | 已获取付款单金额之和,入款单金额之和 |
| 后置条件 | | 系统返回利润（即入款单付款单金额之差） |
| CostIncomeReceiptBL.  getCostIncomeListID | 语法 | | public String getCostIncomeListID(); |
| 前置条件 | | 用户制定成本收益表时需要根据今天已经制定的成本收益表来自动生成编号 |
| 后置条件 | | 系统显示自动生成的成本收益表ID |
| 需要的服务（需接口） | | | |
| FinanceDataService.creatCostIncomeList(CostIncomeReceiptPO po) | | 插入单一持久化对象 CostIncomeReceiptPO | |
| FinanceDataService. getAllCostIncomeList() | | 得到持久化对象CostIncomeReceiptPO的ArrayList | |
| FinanceDataService. getCollection() | | 得到持久化对象CollectionReceiptPO的ArrayList | |
| FinanceDataService. getPayment() | | 得到持久化对象PaymentReceiptPO的ArrayList | |
| FinanceDataService. getNum() | | 得到当天存储的持久化对象CostIncomeReceiptPO的个数 | |
| FinanceDataService. findByID(String ID) | | 根据ID返回对应的一个持久化对象CostIncomeReceiptPO | |
| FinanceDataService. modify(CostIncomeReceiptPO po) | | 修改单一持久化对象CostIncomeReceiptPO的信息 | |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| BusinessStatementReceiptBL.showBSL | 语法 | | public BusinessstatementReceiptVO show(String beginTime,String endTime); |
| 前置条件 | | 用户输入起始日期并要求查看经营情况表 |
| 后置条件 | | 系统显示该段时间内所有入款单和付款单信息 |
| BusinessStatementReceiptBL.showAllBSL | 语法 | | public ArrayList<BusinessstatementReceiptVO> showAllBSList(); |
| 前置条件 | | 用户经验证有查看经营情况表的权限 |
| 后置条件 | | 系统显示所有经营情况的列表 |
| BusinessStatementReceiptBL.export | 语法 | | public int export(BusinessstatementReceiptVO vo); |
| 前置条件 | | 用户要求导出经营情况表的excel |
| 后置条件 | | 系统生成经营情况表的excel |
| 需要的服务（需接口） | | | |
| FinanceDataService.showBSL(String beginTime,String endTime) | | 返回并输出符合时间条件的单一持久化对象BusinessstatementReceiptPO | |
| public ArrayList<BusinessstatementReceiptPO> showAllBSList() | | 返回并输出所有的单一持久化对象BusinessstatementReceiptPO | |
| public int export(BusinessstatementReceiptPO po) | | 导出单一持久化对象BusinessstatementReceiptPO到excel | |

**表6-5 BusinessStatementReceiptBL的接口规范**

**表6-6 InitialStockBL的接口规范**

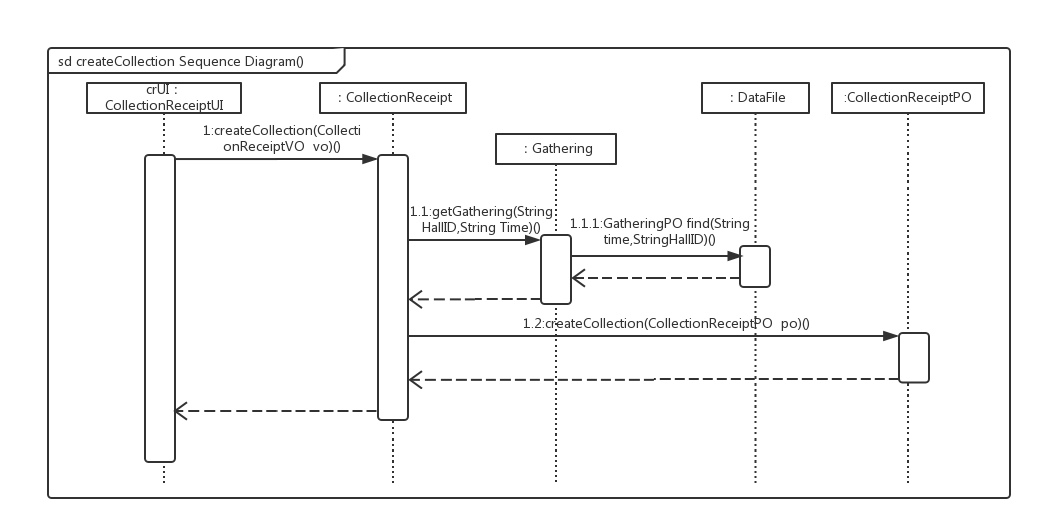
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InitialStockBL.  initInfo | 语法 | public int initInfo(InitInfoVO vo,String time); |
| 前置条件 | 用户要求期初建账 |
| 后置条件 | 系统初始化一套账 |
| InitialStockBL.  getInitInfo | 语法 | public InitInfoVO getInitInfo(String time); |
| 前置条件 | 用户要求查看期初信息 |
| 后置条件 | 返回期初信息并显示 |
| InitialStockBL.  getAllInitInfo | 语法 | public ArrayList<InitInfoVO> getAllInitInfo(); |
| 前置条件 | 用户具有查看期初信息的权限 |
| 后置条件 | 系统返回所有期初信息 |
| 需要的服务（需接口） | | |
| FinanceDataService.initInfo(InitInfoPO po) | 在数据库中添加一套账的初始化信息（InitInfoPO） | |
| FinanceDataService.getInitInfo(String time) | 查找符合时间条件的期初信息并返回 | |
| FinanceDataService.getAllInitInfo() | 返回并输出所有的持久化对象InitInfoPO | |

**图6-7 AccountBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Account.  addAccount | 语法 | public int addAccount(AccountVO vo); | |
| 前置条件 | 用户要求新建账户 | |
| 后置条件 | 在账户列表中添加该账户信息 | |
| Account.  deleteAccount | 语法 | public int deleteAccount(AccountVO vo); | |
| 前置条件 | 用户要求删除账户 | |
| 后置条件 | 从账户列表中删除该账户信息 | |
| Account.  modifyAccount | 语法 | public int modifyAccount(AccountVO vo,String name) | |
| 前置条件 | 用户要求修改账户并选中需修改的账户 | |
| 后置条件 | 在账户列表中修改该账户信息 | |
| Account.  findByName | 语法 | public AccountVO findByName(String name); | |
| 前置条件 | 用户输入账户名称并查找 | |
| 后置条件 | 查找是否存在该名称的账户并返回该账户的信息 | |
| Account.  findByKeyword | 语法 | public ArrayList<AccountVO> findByKeyword(String s); | |
| 前置条件 | 用户输入关键词并查询 | |
| 后置条件 | 查找是否存在含有该关键字的账户并返回含有该关键字的账户的信息 | |
| Account.  showAll | 语法 | public ArrayList<AccountVO> showAll(); | |
| 前置条件 | 用户要求查看所有的账户列表 | |
| 后置条件 | 返回所有账户列表 | |
| 需要的服务（需接口） | | | |
| DataFactory.getAccountData | | | 得到Account数据库的引用 |
| FinanceDataService.addAccount(AccountPO po) | | | 插入单一持久化对象AccountPO |
| FinanceDataService.deleteAccount(AccountPO po) | | | 删除单一持久化对象AccountPO |
| FinanceDataService.modifyAccount(AccountPO po,String name) | | | 修改单一持久化对象AccountPO |
| FinanceDataService.findByName(String name) | | | 根据输入的名称查找单一持久化对象AccountPO |
| public ArrayList<AccountPO> findByKeyword(String s) | | | 根据输入的关键字查找持久化对象AccountPO |
| public ArrayList<AccountPO> showAll() | | | 输出持久化对象AccountPO |

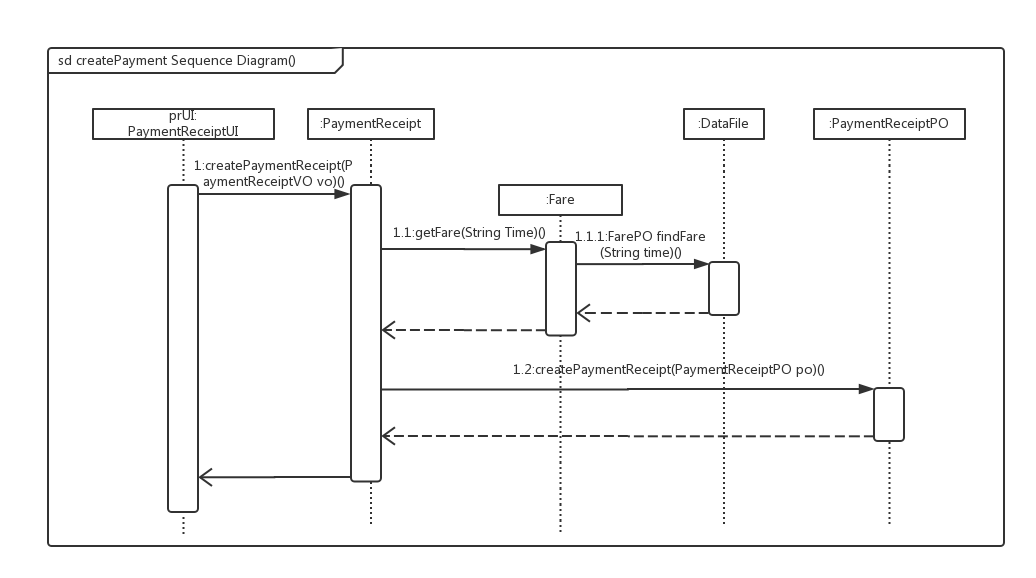
#### 4.2.6.4业务逻辑层的动态模型

图6-2表示了快递物流系统中，当财务人员进行结算管理时，业务逻辑层处理的相关对象之间的协作。



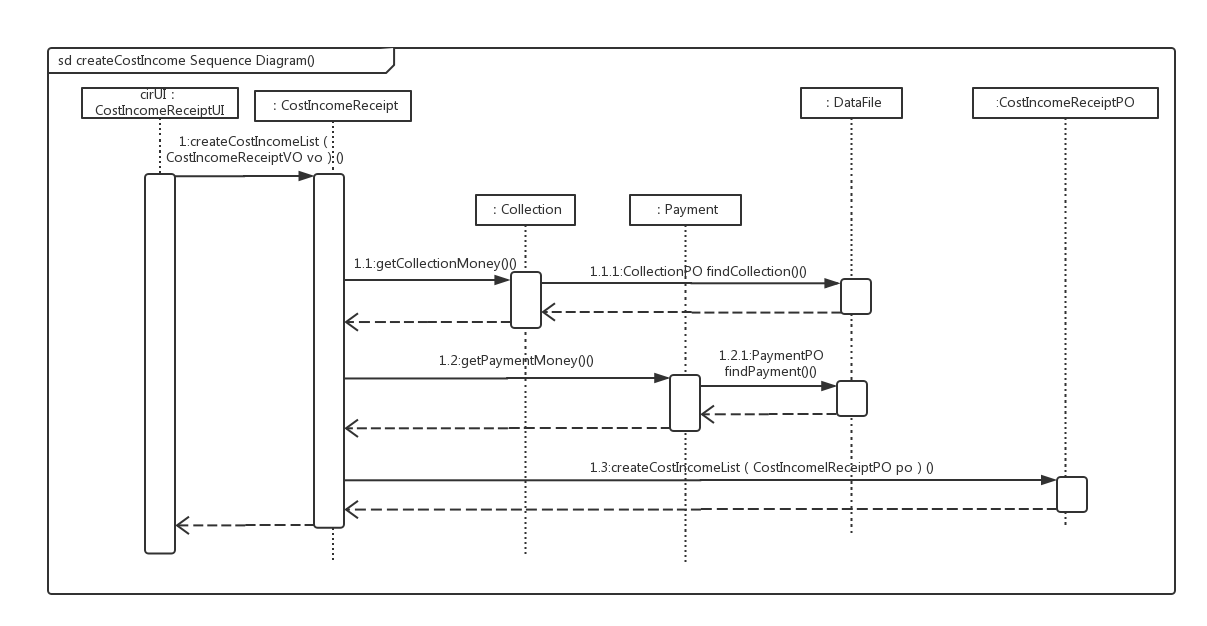
**图6-2 结算管理的顺序图**

图6-3表示了快递物流系统中，当财务人员进行新建付款单时，业务逻辑层处理的相关对象之间的协作。



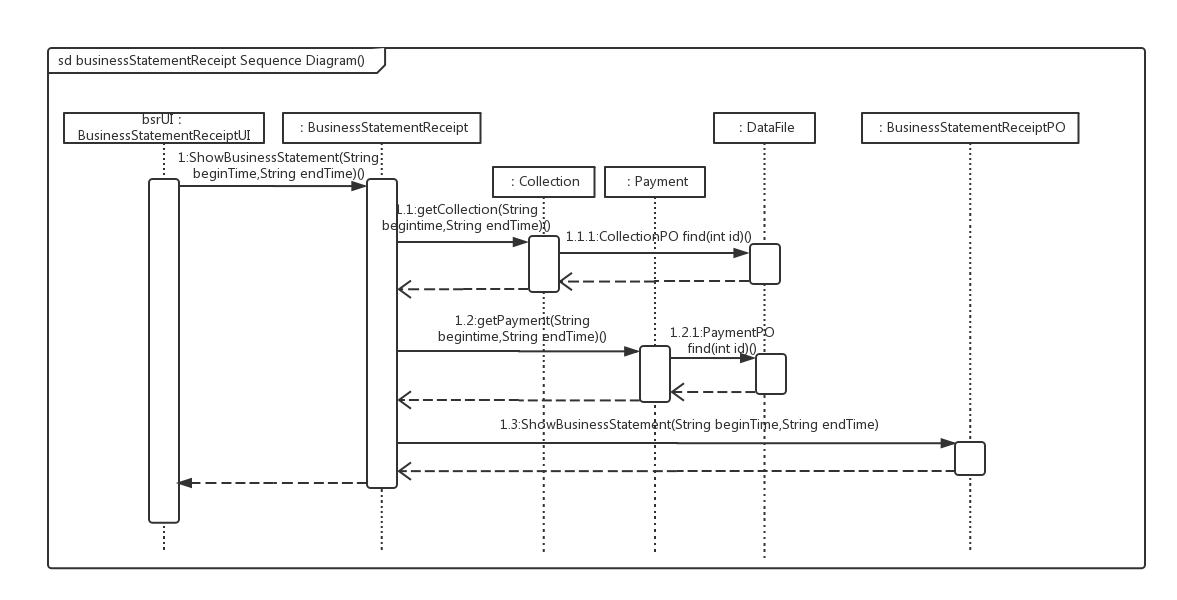
**图6-3 新建付款单的顺序图**

图6-4表示了快递物流系统中，当财务人员进行新建成本收益表时，业务逻辑层处理的相关对象之间的协作。



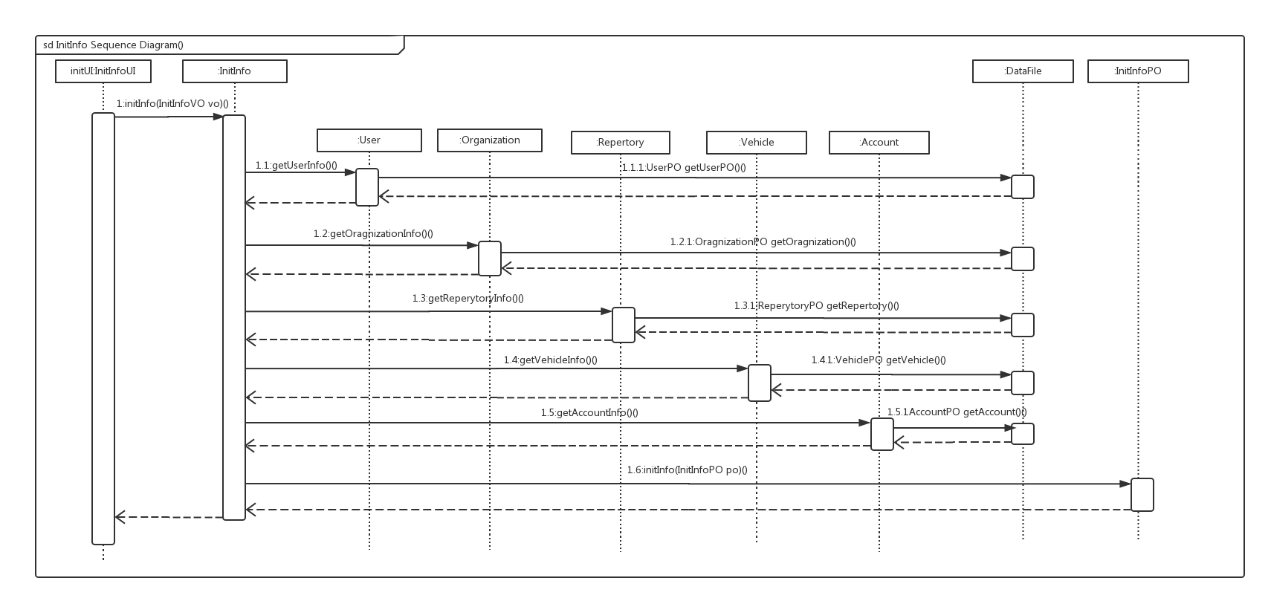
**图6-4 新建成本收益表的顺序图**

图6-5表示了快递物流系统中，当财务人员进行查看经营情况表时，业务逻辑层处理的相关对象之间的协作。



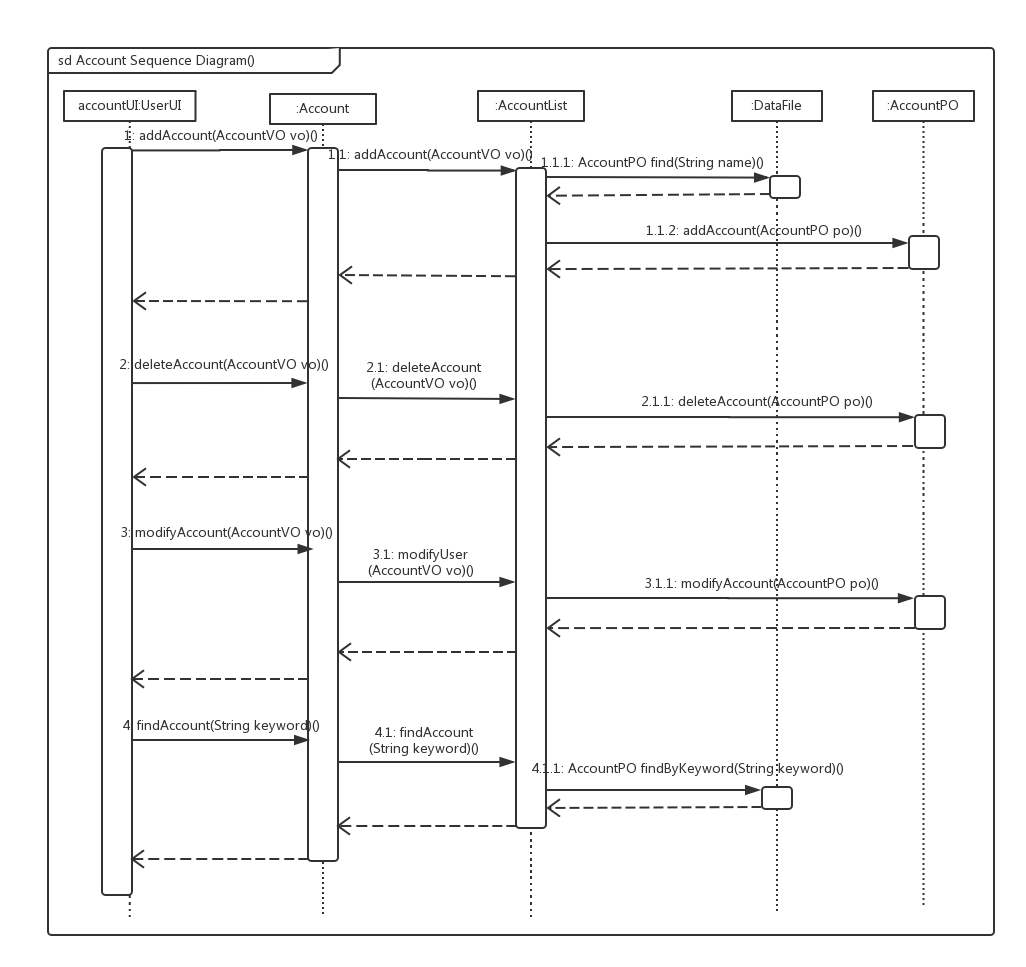
**图6-5 查看经营情况表的顺序图**

图6-6表示了快递物流系统中，当财务人员进行期初建账时，业务逻辑层处理的相关对象之间的协作。



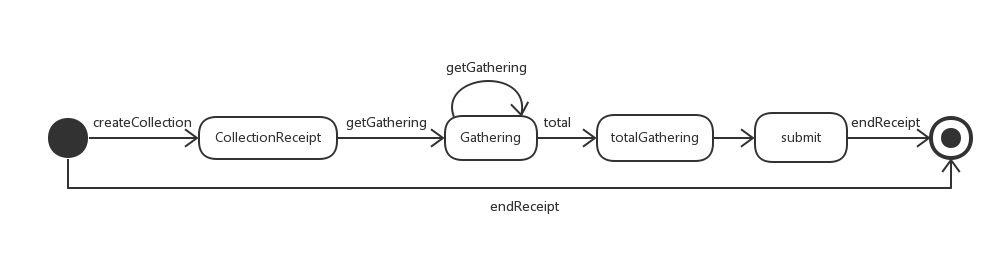
**图6-6 期初建账的顺序图**

图6-7表示了快递物流系统中，当财务人员进行账户管理时，业务逻辑层处理的相关对象之间的协作。



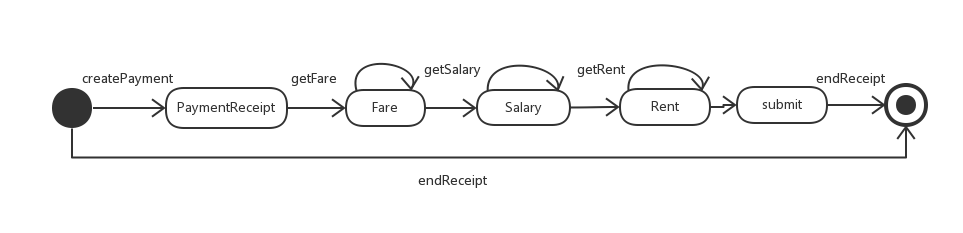
**图6-7 账户管理的顺序图**

如图6-8所示，CollectionReceipt对象状态图描述了CollectionReceipt对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着createCollection方法被 UI调用，进入CollectionReceipt状态；之后通过获取收款单进入Gathering状态。随后，通过计算收款单金额之和进入totalGathering状态；随着submit方法被UI调用，CollectionReceipt进入endReceipt状态。



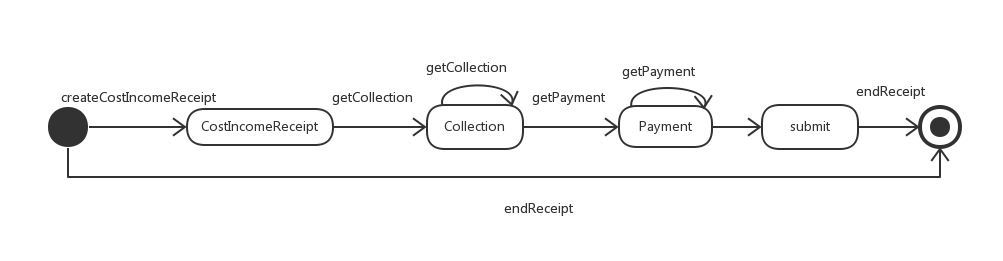
**图6-8 CollectionReceipt对象状态图**

如图6-9所示，PaymentReceipt对象状态图描述了PaymentReceipt对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着createPayment方法被 UI调用，进入PaymentReceipt状态；之后通过获取运费单进入Fare状态，通过输入薪水进入Salary状态，通过输入租金进入Rent状态；随着submit方法被UI调用，PaymentReceipt进入endReceipt状态。



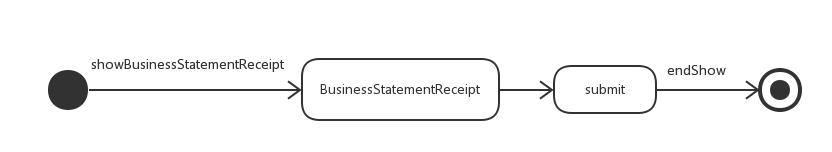
**图6-9 PaymentReceipt对象状态图**

如图6-10所示，CostIncomeReceipt对象状态图描述了CostIncomeReceipt对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着createCostIncome方法被 UI调用，进入CostIncomeReceipt状态；之后通过获取入款单进入Collection状态，通过获取付款单进入Payment状态；随着submit方法被UI调用，CostIncomeReceipt进入endReceipt状态。

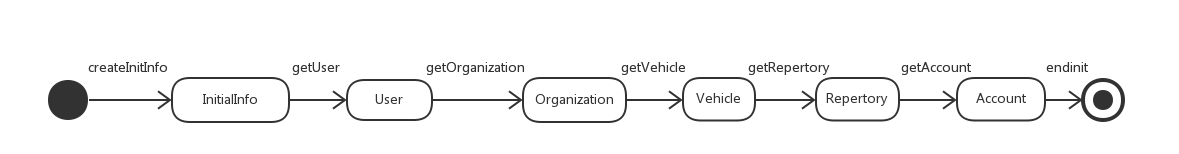


**图6-10 CostIncomeReceipt对象状态图**

如图6-11所示，BusinessStatementReceipt对象状态图描述了BusinessStatementReceipt对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着showBusinessStatementReceipt被 UI调用，进入BusinessStatementReceipt状态；随着submit方法被UI调用，BusinessStatementReceipt进入endShow状态。

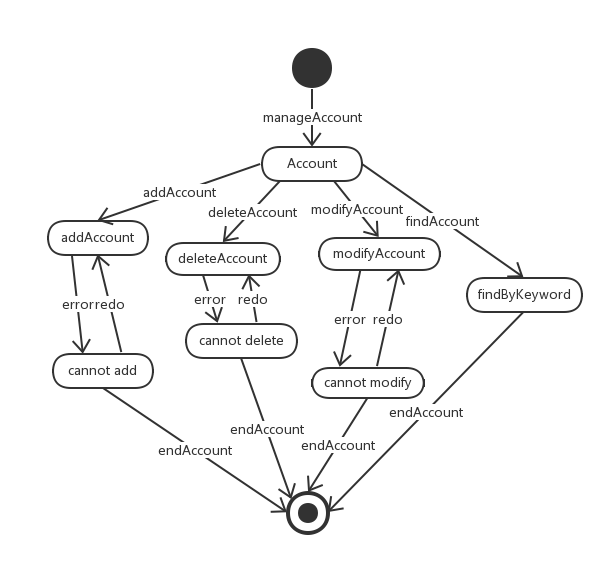


**图6-11 BusinessStatementReceipt对象状态图**

如图6-12所示, InitInfo对象状态图描述了InitInfo对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着createInitInfo方法被 UI调用，进入InitialInfo状态；之后通过添加用户信息进入User状态，通过添加机构信息进入Organization状态，通过添加车辆信息进入Vehicle状态，通过添加库存信息进入Repertory状态，通过添加账户信息进入Account状态；最后进入endInit状态。

**图6-12 InitInfo对象状态图**

如图6-13所示，Account对象状态图描述了Account对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着manageAccount方法被 UI调用，Account进入Account状态；之后通过添加账户进入addAccount状态，通过删除账户进入deleteAccount状态，通过修改账户进入modifyAccount状态，通过查询账户进入findByKeyword状态；最后进入endAccount状态。



**图6-13 Account对象状态图**

### 4.2.7managebl模块

4.2.7.1模块概述

4.2.7.2整体结构

4.2.7.3模块内部类的接口规范

4.2.7.4业务逻辑层的动态模型

4.2.7.5业务逻辑层的设计原理

### 4.2.8userbl模块

4.2.8.1模块概述

4.2.8.2整体结构

4.2.8.3模块内部类的接口规范

4.2.8.4业务逻辑层的动态模型

4.2.8.5业务逻辑层的设计原理

## 4.3数据层的分解

### 4.3.1模块职责

数据层模块的职责如表3所示。

**表3 数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ExpressDataService | 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 |
| BusinessDataService | 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 |
| IntermediateDataService | 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 |
| 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 |
| RepertoryDataService | 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 |
| ManageDataService | 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 |
| 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 |
| ReceiptDataService | 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 |

### 4.3.2接口规范

express数据层模块的接口规范如表4所示。

**表4 express数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExpressDataService.getBaseCost | 语法 | public CostBasePO getBaseCost() |
| 前置条件 | 系统中存在订单收费的价格信息信息 |
| 后置条件 | 系统返回订单收费的价格信息 |
| ExpressDataService.chargeCollection | 语法 | public boolean chargeCollection(ExpressPO po) |
| 前置条件 | ExpressPO中存在成员charge需要更改 |
| 后置条件 | 系统更新ExpressPO的charge属性 |
| ExpressDataService.getExpressInfos | 语法 | public ExpressPO getExpressInfos() |
| 前置条件 | 系统中存在ExpressPO的信息 |
| 后置条件 | 系统返回ExpressPO |

business数据层模块的接口规范如表5所示。

**表5 business数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BusinessDataService.addVehicle | 语法 | public boolean addVehicle(VehiclePO po) |
| 前置条件 | VehiclePOList中需要添加一个VehiclePO |
| 后置条件 | 系统在VehiclePOList中添加一个VehiclePO并返回操作结果 |
| BusinessDataService.getFreeVehicle | 语法 | public ArrayList<VehiclePO> getVehicleInfos() |
| 前置条件 | 系统存在空闲的车辆信息VehiclePO |
| 后置条件 | 系统返回空闲的车辆信息列表VehiclePOList |
| BusinessDataService.deleteVehicle | 语法 | public boolean BusinessDataService.deleteVehicle(VehiclePO po) |
| 前置条件 | 系统中存在改车辆信息VehiclePO |
| 后置条件 | 系统删除该车辆信息，并返回操作结果 |
| BusinessDataService.modifyVehicle | 语法 | public boolean BusinessDataService.modifyVehicle(VehiclePO po) |
| 前置条件 | 系统中存在该车辆信息VehiclePO |
| 后置条件 | 系统修改该车辆信息，并返回操作结果 |
| BusinessDataService.getDriverInfo | 语法 | public DriverPO getDriverInfos(String ID) |
| 前置条件 | 系统中存在该司机信息DriverPO |
| 后置条件 | 系统返回该司机信息DriverPO |
| BusinessDataService.addDriver | 语法 | public boolean addDriver(DriverPO po) |
| 前置条件 | DriverPOList中需要添加一个DriverPO |
| 后置条件 | 系统在DriverPOList中添加一个DriverPO并返回操作结果 |
| BusinessDataService.deleteDriver | 语法 | public boolean deleteDriver(DriverPO po) |
| 前置条件 | 系统中存在该司机信息DriverPO |
| 后置条件 | 系统删除该司机信息DriverPO，并返回操作结果 |
| BusinessDataService.modifyDriver | 语法 | public boolean modifyDriver(DriverPO po) |
| 前置条件 | 系统中存在该车辆信息VehiclePO |
| 后置条件 | 系统修改该车辆信息，并返回操作结果 |
| BusinessDataService.getTransferOrders | 语法 | public ArrayList<OrderVO>getTransferOrders() |
| 前置条件 | 系统中存在待转运的的OrderVO |
| 后置条件 | 系统返回待转运的的OrderVO的列表OrderVOList |
| BusinessDataService.getSendOrder | 语法 | public ArrayList<VehiclePO> getFreeVehicles() |
| 前置条件 | 系统中存在待派送的的OrderVO |
| 后置条件 | 系统返回待派送的的OrderVO的列表OrderVOList |

intermediate数据层模块的接口规范如表6所示。

**表6 intermediate数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| IntermediateDataService. showTransferingReceipt | 语法 | Public TransferingReceiptPO getTransferingReceipt(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TransferingReceiptPO对象 |
| IntermediateDataService.addOrder | 语法 | public boolean addOrder(String ID); |
| 前置条件 | 同样的ID在TransferingReceiptPO的对象中不存在 |
| 后置条件 | 在外部文件中增加一个OrderPO对象 |
| IntermediateDataService.deleteOrder | 语法 | public boolean deleteOrder(String ID); |
| 前置条件 | 在外部文件中存在一个ID符合的OrderPO对象需要删除 |
| 后置条件 | 删除一个OrderPO对象 |
| IntermediateDataService.modifyOrder | 语法 | public boolean modifyOrder(String ID); |
| 前置条件 | 在外部文件中存在一个ID符合的OrderPO对象需要修改 |
| 后置条件 | 更新一个OrderPO对象 |
| IntermediateDataService.updateTransferingReceipt | 语法 | public void updateTransferingReceipt(TransferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 在外部文件中增加一个TransferingReceiptPO对象 |
| IntermediateDataService.getPlaneList | 语法 | public ArrayList<PlanePO> getPlaneList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储PlanePO的ArrayList对象 |
| IntermediateDataService. getPlane | 语法 | public PlanePO getPlane(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个PlanePO对象 |
| IntermediateDataService.updateWaitingList | 语法 | public ArrayList<OrderPO> updateWaitingList(TranferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 存储OrderPO对象的ArrayList对象更新 |
| IntermediateDataService. enplane | 语法 | public EnplaningReceiptPO enplane(ArrayList<OrderPO>); |
| 前置条件 | 存储OrderPO对象的ArrayList对象更新，且不为空链表 |
| 后置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| IntermediateDataService. updateEnplaningReceiptList | 语法 | public ArrayList<EnplaningReceiptPO> updateEnplaningReceiptList(EnplaningReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EnplaningReceiptPO对象 |
| IntermediateDataService. computeFare | 语法 | Public farePO computeFare(ArrayList<EnplaningReceiptPO>); |
| 前置条件 | 同样的EnplaningReceiptPO对象存在 |
| 后置条件 | 返回一个farePO对象 |
| IntermediateDataService. updateFare | 语法 | public boolean updateFare(farePO); |
| 前置条件 | 同样的farePO对象在外部文件中不存在 |
| 后置条件 | 外部文件中新增一个farePO对象 |
| IntermediateDataService.getEnplaningReceipt | 语法 | Public boolean getEnplaningReceipt(ArrayList<EnplaningReceiptPO>); |
| 前置条件 | 存储EnplaningReceiptPO对象的ArrayList对象不为空链表，且同样的EnplaningReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. updateEnplaningReceipt | 语法 | Public boolean updateEnplaningReceipt(ArrayList<EnplaningReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EnplaningRecePO对象 |
| IntermediateDataService.getTrainList1 | 语法 | public ArrayList<TrainPO> getTrainList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储TrainPO的ArrayList对象 |
| IntermediateDataService. getTrain | 语法 | public TrainPO getTrain(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TrainPO对象 |
| IntermediateDataService.updateWaitingList | 语法 | public ArrayList<OrderPO> updateWaitingList(TranferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 存储OrderPO对象的ArrayList对象更新 |
| IntermediateDataService. entrain | 语法 | public EntrainingReceiptPO entrain(ArrayList<OrderPO>); |
| 前置条件 | 存储OrderPO对象的ArrayList对象更新，且不为空链表 |
| 后置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| IntermediateDataService. updateEntrainingReceiptList | 语法 | public ArrayList<EntrainingReceiptPO> updateEntrainingReceiptList(EntrainingReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EntrainingReceiptPO对象 |
| IntermediateDataService. computeFare | 语法 | public farePO computeFare(ArrayList<EntrainingReceiptPO>); |
| 前置条件 | 同样的EntrainingReceiptPO对象存在 |
| 后置条件 | 返回一个farePO对象 |
| IntermediateDataService. updateFare | 语法 | public boolean updateFare(farePO); |
| 前置条件 | 同样的farePO对象在外部文件中不存在 |
| 后置条件 | 外部文件中新增一个farePO对象 |
| IntermediateDataService.getEntrainingReceipt | 语法 | public boolean getEntrainingReceipt(ArrayList<EntrainingReceiptPO>); |
| 前置条件 | 存储EntrainingReceiptPO对象的ArrayList对象不为空链表，且同样的EntrainingReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. updatEntrainingReceipt | 语法 | public boolean updateEntrainingReceipt(ArrayList<EntrainingReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EntrainingRecePO对象 |
| IntermediateDataService.getTruckList | 语法 | public ArrayList<TruckPO> getTruckList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储TruckPO的ArrayList对象 |
| IntermediateDataService. getTruck | 语法 | public TruckPO getTruck(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TruckPO对象 |
| IntermediateDataService.updateWaitingList | 语法 | public ArrayList<OrderPO> updateWaitingList(TranferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 存储OrderPO对象的ArrayList对象更新 |
| IntermediateDataService. entruck | 语法 | public EntruckingReceiptPO enTruck(ArrayList<OrderPO>); |
| 前置条件 | 存储OrderPO对象的ArrayList对象更新，且不为空链表 |
| 后置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| IntermediateDataService. updateEntruckingReceiptList | 语法 | public ArrayList<EntruckingReceiptPO> updateEntruckingReceiptList(EntruckingReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EntruckingReceiptPO对象 |
| IntermediateDataService. computeFare | 语法 | public farePO computeFare(ArrayList<EntruckingReceiptPO>); |
| 前置条件 | 同样的EntruckingReceiptPO对象存在 |
| 后置条件 | 返回一个farePO对象 |
| IntermediateDataService. updateFare | 语法 | public boolean updateFare(farePO); |
| 前置条件 | 同样的farePO对象在外部文件中不存在 |
| 后置条件 | 外部文件中新增一个farePO对象 |
| IntermediateDataService.getEntruckingReceipt | 语法 | public boolean getEntruckingReceipt(ArrayList<EntruckingReceiptPO>); |
| 前置条件 | 存储EntruckingReceiptPO对象的ArrayList对象不为空链表，且同样的EntruckingReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. updatEntruckingReceipt | 语法 | public boolean updateEntruckingReceipt(ArrayList<EntruckingReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EntruckingRecePO对象 |

finance数据层模块的接口规范如表7所示。

**表7 finance数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinanceDataService.InitInfo | 语法 | public int InitInfo(InitInfoPO po)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个 InitInfoPO |
| FinanceDataService.getInitInfo | 语法 | public InitInfoPO getInitInfo（String time）throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回一个 InitInfoPO |
| FinanceDataService.addAccount | 语法 | public int addAccount(AccountPO po) throws RemoteException; |
| 前置条件 | 同样ID的AccountPO在数据库中不存在 |
| 后置条件 | 在数据库中增加一个AccountPO |
| FinanceDataService.deleteAccount | 语法 | public int deleteAccount(AccountPO po) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的AccountPO |
| 后置条件 | 删除一个AccountPO |
| FinanceDataService.modifyAccount | 语法 | public int modifyAccount(AccountPO po,String name) throws RemoteException; |
| 前置条件 | 在数据库中存在某个需要被修改的AccountPO |
| 后置条件 | 修改一个AccountPO |
| FinanceDataService.findByName | 语法 | public AccountPO findbyName(String name) throws RemoteException； |
| 前置条件 | 含有该名称的AccountPO在数据库中存在 |
| 后置条件 | 按名称进行查找返回相应的AccountPO |
| FinanceDataService.findByKeyword | 语法 | public ArrayList<AccountPO> findByKeyword(String s) throws RemoteException； |
| 前置条件 | 含有关键字的AccountPO在数据库中存在 |
| 后置条件 | 按关键字进行查找返回相应的AcountPO的ArrayList |
| FinanceDataService.showAll | 语法 | Public ArrayList<AccountPO> showAll() throws RemoteException; |
| 前置条件 | 在数据库中存在AccountPO的记录 |
| 后置条件 | 返回数据库中所有的AccountPO记录 |
| FinanceDataService.showBSL | 语法 | public BusinessstatementReceiptPO showBSL(String beginTime,String endTime) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 BusinessstatementReceiptPO |
| 后置条件 | 返回一个BusinessstatementReceiptPO |
| FinanceDataService.showAllBSList | 语法 | public ArrayList<BusinessstatementReceiptPO> showAllBSList()throws RemoteException; |
| 前置条件 | 数据库中存在 BusinessstatementReceiptPO |
| 后置条件 | 返回 BusinessstatementReceiptPO的ArrayList |
| FinanceDataService.export | 语法 | public int export(BusinessstatementReceiptPO po)throws RemoteException; |
| 前置条件 | 数据库中存在 BusinessstatementReceiptPO |
| 后置条件 | 导出 BusinessstatementReceiptPO到excel |
| FinanceDataService.creatCollection | 语法 | public int createCollection(CollectionReceiptPO po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张入款单 CollectionReceiptPO |
| FinanceDataService.getAllCollection | 语法 | public ArrayList<CollectionReceiptPO> getAllCollection() throws RemoteException; |
| 前置条件 | 数据库中存在 CollectionReceiptPO记录 |
| 后置条件 | 返回数据库中所有的 CollectionReceiptPO记录 |
| FinanceDataService.getGathering | 语法 | public ArrayList<GatheringReceiptPO> getGathering(String HallID,String Time) throws RemoteException; |
| 前置条件 | 数据库中存在 GatheringReceiptPO记录 |
| 后置条件 | 返回符合条件的 GatheringReceiptPO记录 |
| FinanceDataService.getMoney | 语法 | public double[] getMoney(GatheringReceiptPO po) throws RemoteException; |
| 前置条件 | 已经获得GatheringReceiptPO记录 |
| 后置条件 | 返回GatheringReceiptPO中所有金额 |
| FinanceDataService.getTotalMoney | 语法 | public double getTotalMoney(int[] money) throws RemoteException; |
| 前置条件 | 已经获得GatheringReceiptPO中的所有金额 |
| 后置条件 | 返回所有金额之和 |
| FinanceDataService.getNum | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已创建的入款单数 |
| FinanceDataService.findByID | 语法 | public CollectionReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的CollectionReceiptPO |
| 后置条件 | 返回相应ID的 CollectionReceiptPO记录 |
| FinanceDataService.modify | 语法 | publicCollectionReceiptPOmodify(CollectionReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 CollectionReceiptPO |
| 后置条件 | 修改相应的 CollectionReceiptPO记录 |
| FinanceDataService.creatPaymentReceipt(PaymentReceiptPO po) | 语法 | public int creatPaymentReceipt(PaymentReceiptPO vo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张付款单PaymentReceiptPO |
| FinanceDataService.getAllPaymentReceipt() | 语法 | public ArrayList<PaymentReceiptPO> getAllPaymentReceipt() throws RemoteException; |
| 前置条件 | 数据库中存在 PaymentReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 PaymentReceiptPO记录 |
| FinanceDataService. getNum() | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已创建的付款单数 |
| FinanceDataService.findByID(String ID) | 语法 | public PaymentReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合ID的 PaymentReceiptPO |
| 后置条件 | 返回相应的 PaymentReceiptPO记录 |
| FinanceDataService.modify(PaymentReceiptPO po) | 语法 | public PaymentReceiptPO modify(PaymentReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的PaymentReceiptPO |
| 后置条件 | 修改相应的PaymentReceiptPO记录 |
| FinanceDataService.creatCostIncomeList(CostIncomeReceiptPO po) | 语法 | public int creatCostIncomeList(CostIncomeReceiptPO po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张成本收益表CostIncomeReceiptPO |
| FinanceDataService.getAllCostIncomeList() | 语法 | public ArrayList<CostIncomeReceiptPO> getAllCostIncomeList() throws RemoteException; |
| 前置条件 | 数据库中存在 CostIncomeReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 CostIncomeReceiptPO记录 |
| FinanceDataService.getCollection() | 语法 | public ArrayList<CollectionReceiptPO> getCollection() throws RemoteException; |
| 前置条件 | 数据库中存在 CollectionReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 CollectionReceiptPO记录 |
| FinanceDataService.getPayment() | 语法 | public ArrayList<PaymentReceiptPO>getPayment() throws RemoteException; |
| 前置条件 | 数据库中存在 PaymentReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 PaymentReceiptPO记录 |
| FinanceDataService. getNum() | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已经创建的成本收益表数 |
| FinanceDataService.findByID(String ID) | 语法 | public CostIncomeReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合ID的CostIncomeReceiptPO记录 |
| 后置条件 | 返回相应的 CostIncomeReceiptPO记录 |
| FinanceDataService.modify(CostIncomeReceiptPO po) | 语法 | public CostIncomeReceiptPO modify(CostIncomeReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 CostIncomeReceiptPO |
| 后置条件 | 修改相应的CostIncomeReceiptPO记录 |

repertory数据层模块的接口规范如表8所示。

**表8 repertory数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RepertoryDataService. modifyRepertory | 语法 | public int modifyRepertory(RepertoryPO repertorypo) |
| 前置条件 | 需要修改仓库信息 |
| 后置条件 | 更新一个po对象 |
| RepertoryDataService. findRepertory | 语法 | public RepertoryPO findRepertory(String repertoryID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的RepertoryPO的结果 |
| RepertoryDataService. addGoods | 语法 | public int addGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 无 |
| 后置条件 | 在仓库的库存信息列表中增加一个po记录 |
| RepertoryDataService. deleteGoods | 语法 | public int deleteGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 在仓库的库存信息列表中存在某一个po需要删除 |
| 后置条件 | 删除一个po对象 |
| RepertoryDataService. modifyGoods | 语法 | public int modifyGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 在仓库的库存信息列表中存在某一个po需要更新 |
| 后置条件 | 更新一个po对象 |
| RepertoryDataService. findGoodsbyID | 语法 | public GoodsPO findGoodsbyID(String repertoryID, String JJD-ID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的GoodsPO的结果 |
| RepertoryDataService.findGoodsbyDate | 语法 | public GoodsPO findGoodsbyDate(String repertoryID, String beginDate, String endDate) |
| 前置条件 | 无 |
| 后置条件 | 按beginDate和endDate进行查找返回相应的GoodsPO的结果 |
| RepertoryDataService.findGoodsbyTime | 语法 | public GoodsPO findGoodsbyTime(String repertoryID, String time) |
| 前置条件 | 无 |
| 后置条件 | 按time进行查找返回相应的GoodsPO的结果 |

manage数据层模块的接口规范如表9所示。

**表9manage数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageDataService.addOrganization | 语法 | public int addOrganization(OrganizationPO organizationpo) |
| 前置条件 | 同样ID的organizationpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个organizationpo记录 |
| ManageDataService.deleteOrganization | 语法 | public int deleteOrganization(OrganizationPO organizationpo) |
| 前置条件 | 在外部文件中存在某一个organizationpo需要删除 |
| 后置条件 | 删除一个organizationpo对象 |
| ManageDataService.modifyOrganization | 语法 | public int modifyOrganization(OrganizationPO organizationpo) |
| 前置条件 | 在外部文件中存在某一个organizationpo需要更新 |
| 后置条件 | 更新一个organizationpo对象 |
| ManageDataService.findOrganization | 语法 | public OrganizationPO findOrganization(String organizationID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的OrganizationPO的结果 |
| ManageDataService.modifyUser | 语法 | public int modifyUser(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个userpo需要更新 |
| 后置条件 | 更新一个userpo对象 |
| ManageDataService.addPerWage | 语法 | public int addPerWage(PerWagePO perwagepo) |
| 前置条件 | 同样profession的perwagepo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个perwagepo记录 |
| ManageDataService.deletePerWage | 语法 | public int deletePerWage(PerWagePO perwagepo) |
| 前置条件 | 在外部文件中存在某一个perwagepo需要删除 |
| 后置条件 | 删除一个perwagepo对象 |
| ManageDataService.modifyPerWage | 语法 | public int modifyPerWage(PerWagePO perwagepo) |
| 前置条件 | 在外部文件中存在某一个perwagepo需要更新 |
| 后置条件 | 更新一个perwagepo对象 |
| ManageDataService.findPerWage | 语法 | public double findPerWage(int profession) |
| 前置条件 | 无 |
| 后置条件 | 按profession进行查找返回相应的PerWage的值 |
| ManageDataService.addBasicSalary | 语法 | public int addBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 同样profession的basicsalarypo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个basicsalarypo记录 |
| ManageDataService.deleteBasicSalary | 语法 | public int deleteBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 在外部文件中存在某一个basicsalarypo需要删除 |
| 后置条件 | 删除一个basicsalarypo对象 |
| ManageDataService.modifyBasicSalary | 语法 | public int modifyBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 在外部文件中存在某一个basicsalarypo需要更新 |
| 后置条件 | 更新一个basicsalarypo对象 |
| ManageDataService.findBasicSalary | 语法 | public double findBasicSalary((int profession)) |
| 前置条件 | 无 |
| 后置条件 | 按profession进行查找返回相应的BasicSalary的值 |
| ManageDataService.addCityDistance | 语法 | public int addCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 同样City A, City B的citydistancepo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个citydistancepo记录 |
| ManageDataService.deleteCityDistance | 语法 | public int deleteCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 在外部文件中存在某一个citydistancepo需要删除 |
| 后置条件 | 删除一个citydistancepo对象 |
| ManageDataService.modifyCityDistance | 语法 | public int modifyCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 在外部文件中存在某一个citydistancepo需要更新 |
| 后置条件 | 更新一个citydistancepo对象 |
| ManageDataService.findCityDistance | 语法 | public double findCityDistance(String CityA, String CityB) |
| 前置条件 | 无 |
| 后置条件 | 按CityA, CityB进行查找返回相应的CityDistance的值 |
| ManageDataService.addCost | 语法 | public int addCost(CostPO costpo) |
| 前置条件 | 同样category的costpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个costpo记录 |
| ManageDataService.deleteCost | 语法 | public int deleteCost(CostPO costpo) |
| 前置条件 | 在外部文件中存在某一个costpo需要删除 |
| 后置条件 | 删除一个costpo对象 |
| ManageDataService.modifyCost | 语法 | public int modifyCost(CostPO costpo) |
| 前置条件 | 在外部文件中存在某一个costpo需要更新 |
| 后置条件 | 更新一个costpo对象 |
| ManageDataService.findCost | 语法 | public double findCost(category) |
| 前置条件 | 无 |
| 后置条件 | 按category进行查找返回相应的Cost的值 |

user数据层模块的接口规范如表10所示。

**表10 user数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService.check | 语法 | public boolean check(String userID, String password) |
| 前置条件 | 无 |
| 后置条件 | 返回用户名和密码是否匹配 |
| UserDataService.add | 语法 | public int add(UserPO userpo) |
| 前置条件 | 同样ID的userpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个po记录 |
| UserDataService.delete | 语法 | public int delete(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个po需要删除 |
| 后置条件 | 删除一个userpo对象 |
| UserDataService.modify | 语法 | public int modify(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个po需要更新 |
| 后置条件 | 更新一个userpo对象 |
| UserDataService.find | 语法 | public UserPO find (String ID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的UserPO的结果 |
| UserDataService.showAll | 语法 | public ArrayList<UserPO> showAll() |
| 前置条件 | 无 |
| 后置条件 | 返回所有的UserPO的结果 |

receipt数据层模块的接口规范如表11所示。

**表11 receipt数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| ReceiptDataService.add | 语法 | Public int Add(ReceiptPo po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个po记录 |
| ReceiptDataService.find | 语法 | Public ReceiptPO find(String id)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的ReceiptPo()结果 |
| ReceiptDataService.modify | 语法 | Public int Modify(ReceiptPo po)throws RemoteException |
| 前置条件 | 该单据id存在 |
| 后置条件 | 修改一个po |
| ReceiptDataService.show | 语法 | Public ArrayList<ReceiptPO> show()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有的po |
| ReceiptDataService.init | 语法 | Public void init()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始持久化数据库的使用 |