详细设计描述文档

物流信息管理系统

更新历史：

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 修改原因 | 版本号 |
| 栾志远 | 2015/10/26 | 完成文档 | V1.0 |
| 全体人员 | 2015/11/16 | 完善文档 | V1.1 |
| 栾志远、孙婧 | 2015/12/28 | 完善文档 | V1.2 |

目录

[详细设计描述文档 1](#_Toc435469943)

[1.引言 4](#_Toc435469944)

[1.1编制目的 4](#_Toc435469945)

[2.产品概述 4](#_Toc435469946)

[3.体系结构设计概述 4](#_Toc435469947)

[4.结构视角 4](#_Toc435469948)

[4.1业务逻辑层的分解 4](#_Toc435469949)

[4.1.1 informationbl模块 5](#_Toc435469950)

[4.1.2 sendbl模块 8](#_Toc435469951)

[4.1.3 Judegementbl模块 11](#_Toc435469952)

[4.1.4 shipmentbl模块 16](#_Toc435469953)

[4.1.5 receivementbl模块 20](#_Toc435469954)

[4.1.6 commoditybl模块 24](#_Toc435469955)

[4.1.7 moneybl模块 27](#_Toc435469956)

[4.1.8 reportbl模块 36](#_Toc435469957)

[4.1.9 staffbl模块 38](#_Toc435469958)

[4.2数据层的分解 43](#_Toc435469959)

[4.2.1 InformationData模块 44](#_Toc435469960)

[4.2.2 SendData模块 44](#_Toc435469961)

[4.2.3 shipmentdata模块 45](#_Toc435469962)

[4.2.4 receivementdata模块 47](#_Toc435469963)

[4.2.5 commoditydata模块 48](#_Toc435469964)

[4.2.6 moneydata模块 48](#_Toc435469965)

[4.2.7 StaffData模块 49](#_Toc435469966)

[5.依赖视角 51](#_Toc435469967)

# 1.引言

## 1.1编制目的

本报告详细完成对物流信息管理系统的概要设计，达到指导详细设计和开发的目的，同

时实现和开发人员及用户的沟通。

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。

# 2.产品概述

参考物流信息管理系统用例文档和物流信息管理系统软件需求规格说明中对产品的概

括描述。

# 3.体系结构设计概述

参考物流信息管理系统概要设计文档中对体系结构设计的概述。

# 4.结构视角

## 4.1业务逻辑层的分解



### 4.1.1 informationbl模块

（1）模块概述

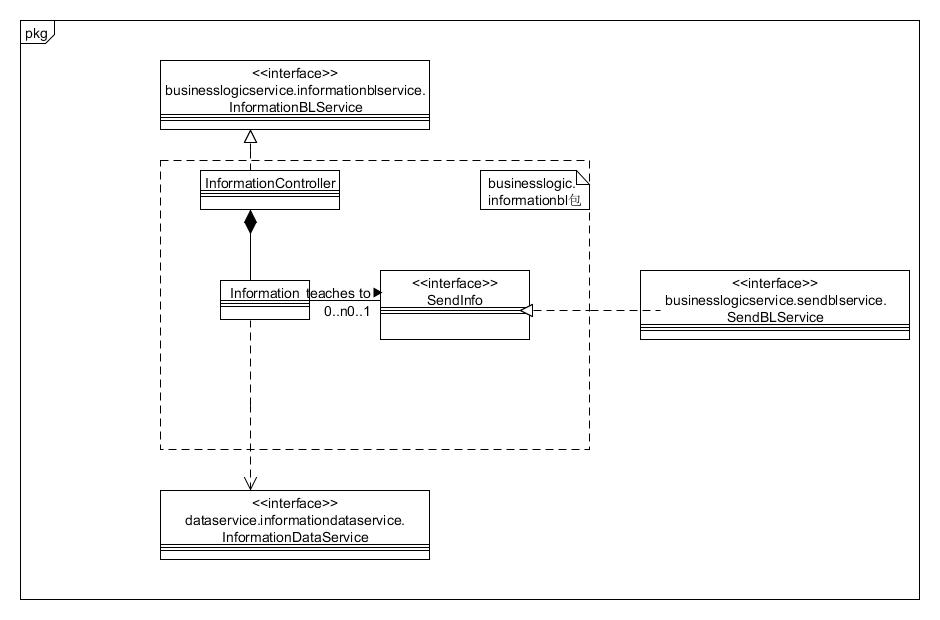
Informationbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Informationbl模块的职责及接口参见软件系统结构描述文档。

（2）整体结构

根据体系结构的设计，我们将系统分成展示层，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。比如在展示层与业务逻辑层之间，我们添加了businesslogicservice.informationblservice.InformationBLService接口。业务逻辑层与数据层之间，我们添加了dataservice.informationdataservice.InformationDataService接口。为了隔离业务逻辑职责与逻辑控制职责，我们增加了InformationController，这样对物流信息查询的业务逻辑处理委托给information对象。

Informationbl模块的设计如图所示



Informationbl模块各个类的职责如表所示

|  |  |
| --- | --- |
| 模块 | 职责 |
| InformationController | 负责实现查询界面需要的服务 |
| Information | 物流信息查询的领域模型对象，帮助实现查询界面需要的服务 |

（3）模块内部类的接口规范

InformationController和information的接口规范如表所示

InformationController的接口规范

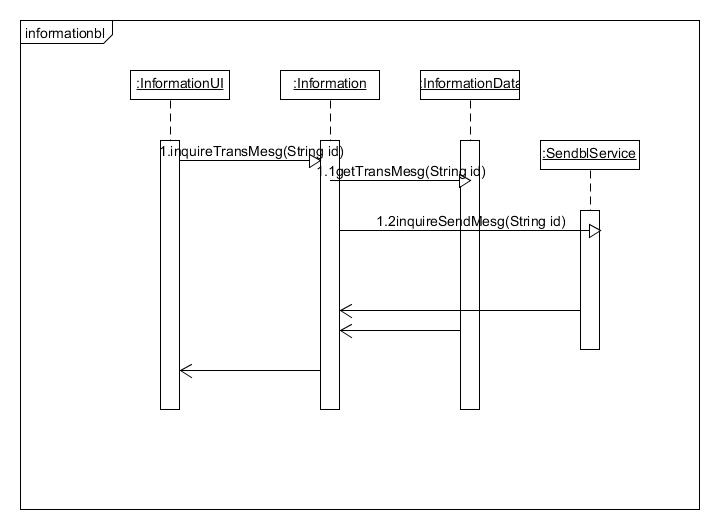
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InformationController  .InquireTransMesg | 语法 | Public PackageVO inquireTransMesg(String id) |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用information对象的inquireTransMesg方法 |
| InformationController.refreshMesg | 语法 | Public resultmessage refreshMesg(String id,String position,String time) |
| 前置条件 | 输入符合规则 |
| 后置条件 | 更新物流信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Information.InquireTransMesg | 查询快递信息 | |
| Imformationdata.refreshTransMesg | 更新物流信息 | |

Information的接口规范

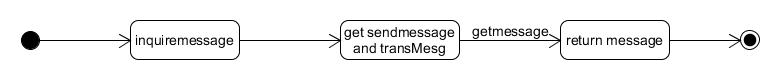
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Information.inquireTransMesg | 语法 | Public PackageVO inquireTransMesg(String id) |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 返回快递物流信息 |
| Information.refreshMesg | 语法 | Public boolean refreshMesg(String id,String position,String time) |
| 前置条件 | 输入符合规则 |
| 后置条件 | 更新物流信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| InformationDataService  .getTransMesg(String id) | 返回物流信息 | |
| InformationDataService  .refreshTransMesg(TransMesgPO po) | 更新物流信息 | |
| Sendbl.inquireSendMesg(String id) | 查询物流信息 | |

（4）业务逻辑层的动态模型

下图表明了快递物流系统中，当用户输入查询的单号之后，查询业务逻辑处理的相关对象之间的协作



下图显示information对象生存期间的状态序列。开始时进入inquiremessage状态，随后进入getmessage状态，得到信息之后，进入返回信息状态，最后退出。



（5）业务逻辑层的设计原理

利用委托式风格，将不同的任务由各自的控制器交给不同的领域对象处理。

### 4.1.2 sendbl模块

（1）模块概述

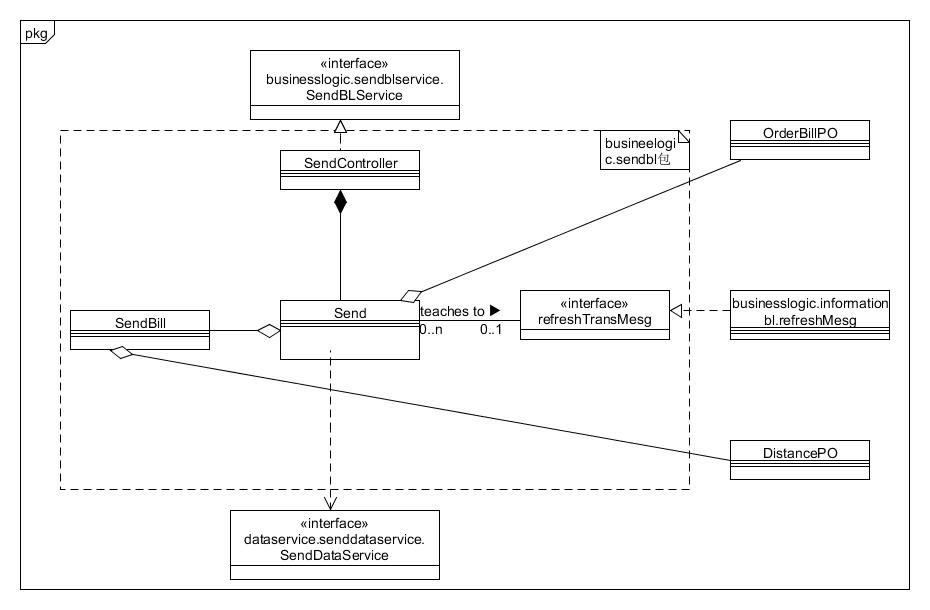
Sendbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Sendbl模块的职责及接口参见软件系统结构描述文档。

（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.sendblservice.SendBLService接口。业务逻辑层和数据层之间添加dataservice.salesdataservice.SalesDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加SendController，这样SendController把对业务逻辑处理委托给Send对象。OrderBillPO是作为订单信息的持久化对象被添加到设计模型中去的。SendBill封装了计算总价的职责，DistancePO是各地之间距离信息的持久化对象。

Informationbl模块的设计如图所示



sendbl模块各个类的职责如表所示

|  |  |
| --- | --- |
| 模块 | 职责 |
| SendController | 负责实现订单输入界面需要的服务 |
| Send | 订单输入的领域模型对象，帮助实现订单输入需要的服务 |
| SendBill | 负责实现计算时间以及报价 |

（3）模块内部类的接口规范

SendController的接口规范如表所示

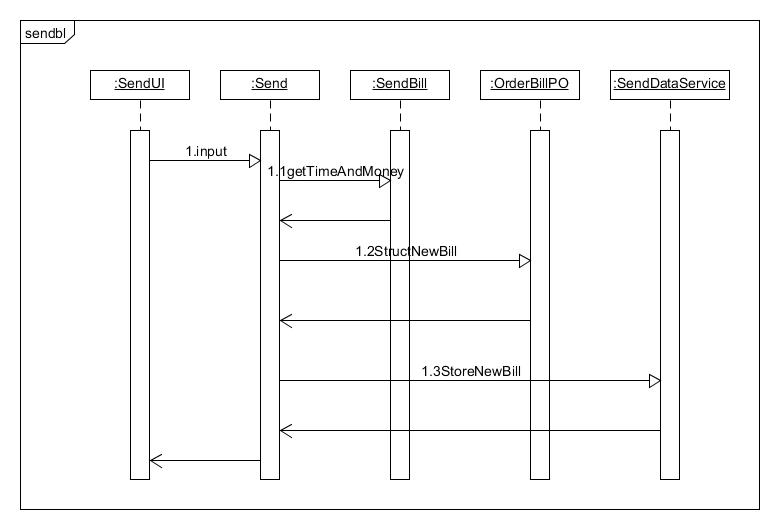
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SendController  .push | 语法 | public Boolean push(OrderBillPO) |
| 前置条件 | 使用者有输入订单的权限 |
| 后置条件 | 系统申请存储订单信息 |
| SendController  .inquireSendMesg | 语法 | Public OrderBillPO inquireSendMesg(String id) |
| 前置条件 | 使用者有查看订单信息的权限 |
| 后置条件 | 无 |
| SendController  .getUnmarkingBills | 语法 | Public ArrayList<OrderBillPO> getUnmarkingBills () |
| 前置条件 | 输入符合规则 |
| 后置条件 | 系统获得未审判的单据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Send.InquireTransMesg | 查询订单信息 | |
| Send.push | 输入订单信息 | |
| Send. getUnmarkingBills | 生成未审判的单据 | |

Send的接口规范如表所示

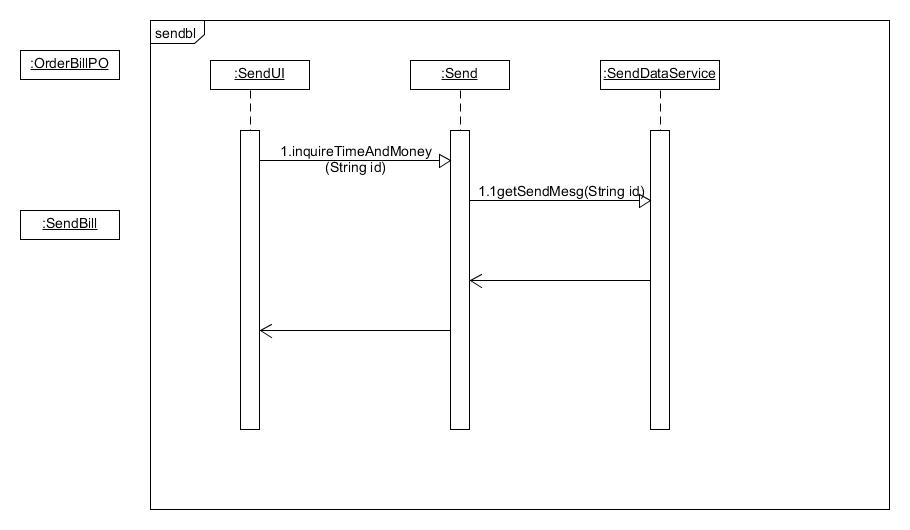
|  |  |  |
| --- | --- | --- |
| 提供的接口（供接口） | | |
| Send.inquireSendMesg | 语法 | Public OrderBillPO inquireSendMesg(String id) |
| 前置条件 | 使用者有查看订单信息的权限 |
| 后置条件 | 无 |
| Send.push | 语法 | public boolean push(OrderBillPO order) |
| 前置条件 | 使用者有输入订单的权限 |
| 后置条件 | 系统申请存储订单信息 |
| Send.getUnmarkingBills | 语法 | public ArrayList<OrderBillPO> getUnmarkingBills () |
| 前置条件 | 输入符合规则 |
| 后置条件 | 系统获得未审判的单据 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| Senddata.getSendMesg | 返回单一持久化对象（物流订单信息） | |
| Senddata.getInput | 写入单一持久化对象（订单信息） | |
| Informationbl.refreshMesg | 写入单一持久对象（物流信息） | |

（4）业务逻辑层的动态模型

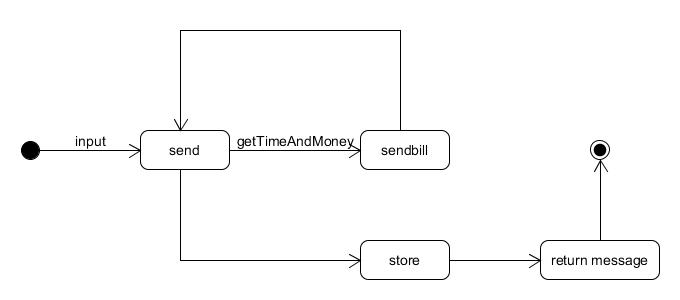
下图表明了快递物流系统中，当快递员输入订单信息之后，订单输入逻辑处理的相关对象之间的协作



下图表明了快递物流系统中，当快递员输入查询的单号之后，时间和报价查询的相关对象之间的协作



下图显示send对象生存期间的状态序列。开始时进入send状态，随后接收输入信息后进入sendbill状态，得到信息之后，返回send状态，随后进入store状态，然后进入return message状态，最后退出。



（5）业务逻辑层的设计原理

利用委托式风格，将不同的任务由各自的控制器交给不同的领域对象处理。

### 4.1.3 Judegementbl模块

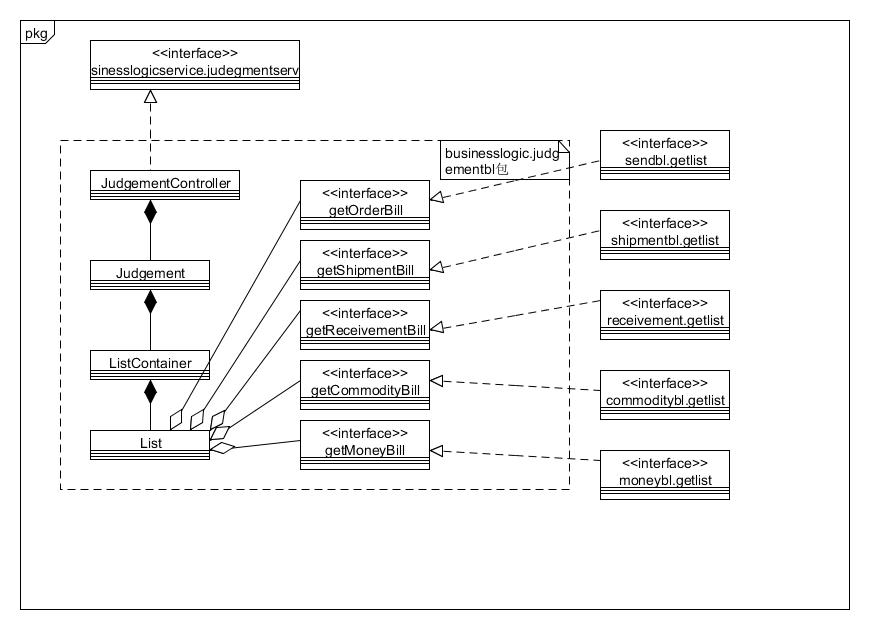
（1）模块概述

Judegementbl模块负责实现总经理单据审批功能。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

（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。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.JudegmentBLService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了JudgementContriller,这样JudgementContriller将单据审批的业务委托给judgement类。List 保存了需要审批的单据，ListContainer是list的容器类。getOrderBill,getShipmentBill,getreceivementBill,getCommodityBill,getMoneyBill是为了消除循环依赖的接口。

Judgementbl模块的设计如图



Judegmentbl模块各个类的职责

|  |  |
| --- | --- |
| MarkingBillstController | 负责实现单据审判界面需要的服务 |
| MarkingBills | 单据审批的领域模型对象，帮助实现单据审判界面需要的服务 |
| ListContainer | List的容器类 |
| List | 需要审批的表单 |

（3）模块内部类的接口规范

Judgement和judgementController的接口规范如下

MarkingBillsController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MarkingBillsController.  getBills | 语法 | Public ArrayList<T> getBills(String type) |
| 前置条件 | 经理身份识别获得权限 |
| 后置条件 | 无 |
| MarkingBillsController.  updateRecord | 语法 | Public boolean updateRecord(ArrayList<T> bills) |
| 前置条件 | 经理身份识别获得权限 |
| 后置条件 | 无 |
| MarkingBillsController.  submitRecord | 语法 | Public boolean submitRecord (ArrayList<T> bills) |
| 前置条件 | 经理身份识别获得权限 |
| 后置条件 | 无 |
| 需要的服务（需接口） | | |
| MarkingBills.getBills | 获取单据对象 | |
| MarkingBills.updateRecord | 更新单据对象 | |
| MarkingBills.submitRecord | 提交单据对象 | |

MarkingBills的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MarkingBills.  getBills | 语法 | Public ArrayList<T> getBills(String type) |
| 前置条件 | 经理身份识别获得权限 |
| 后置条件 | 无 |
| MarkingBills.  updateRecord | 语法 | Public boolean updateRecord(ArrayList<T> bills) |
| 前置条件 | 经理身份识别获得权限 |
| 后置条件 | 无 |
| MarkingBills.  submitRecord | 语法 | Public boolean submitRecord (ArrayList<T> bills) |
| 前置条件 | 经理身份识别获得权限 |
| 后置条件 | 无 |

ListContainer的接口规范

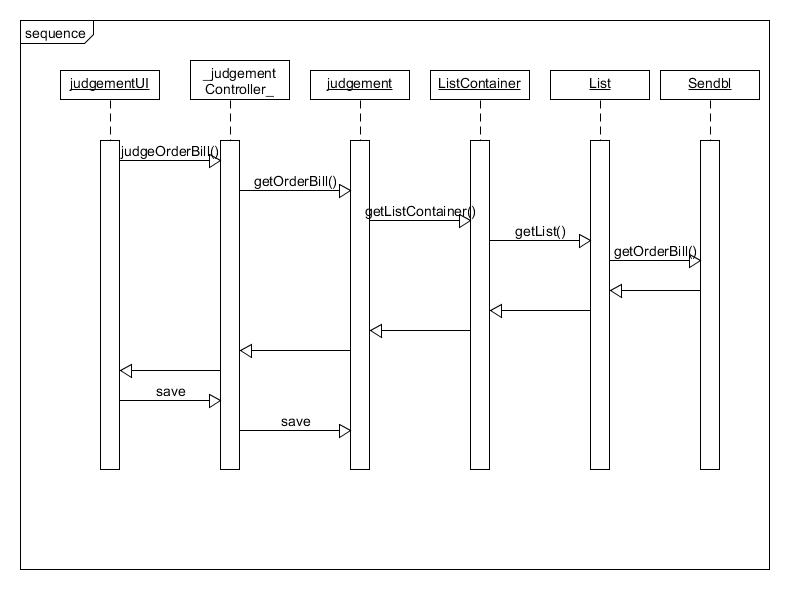
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ListContainer.getListCollect  ion | | 语法 | Public resultmessage getListCollection |
| 前置条件 | 经理身份识别获得权限 |
| 后置条件 | 无 |
| 需要的服务（需接口） | | | |
| List.getList | 返回单据对象 | | |

List的接口规范

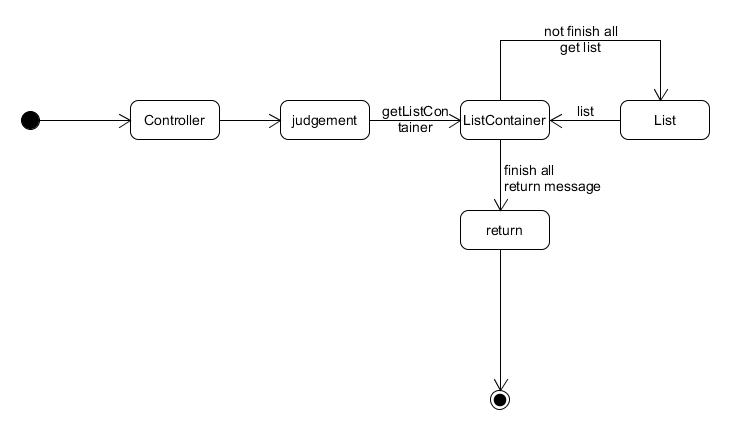
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| List.getList | 语法 | Public resultmessage getList() |
| 前置条件 | 经理身份识别获得权限 |
| 后置条件 | 无 |

（4）业务逻辑层的动态模型

下图表示了总经理审批订单的过程中，各个对象之间的协作，其他情况类似



下图为总经理审批订单过程中系统的状态图



系统先调用controller，接着调用MarkingBills，接着getListContainer，如果没有完成提取所有的单据，则继续提取，如果提取完所有的单据，返回信息，系统结束。

（5）业务逻辑层设计原理

使用委托式风格，将不同任务委托给不同对象处理。

### 4.1.4 shipmentbl模块

（1）模块概述

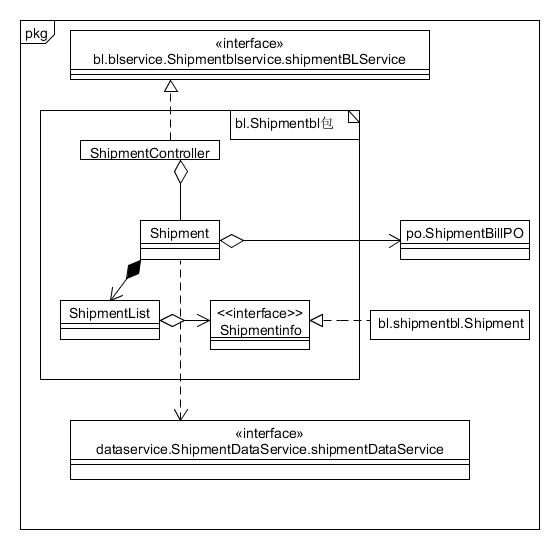
shipmentbl模块承担的需求详见需求规格说明文档功能需求及相关非功能需求。

shipmentbl模块的职责及接口参见软件系统结构描述文档。

（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bl.blservice.Shipmentblservice接口。业务逻辑层和数据层之间添加dataservice.ShipmentDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了ShipmentController，这样ShipmentController会将对销售的业务逻辑处理委托给Shipment对象。ShipmentBillPO是作为单据的持久化对象被添加到设计模型中去的。而ShipmentList和的添加是Shipmentinfo的容器类。ShipmentList封装了装运信息的数据。Shipmentinfo是根据依赖倒置原则，为了消除循环依赖而产生的接口。

shipmentbl模块的设计如图所示。



**Shipmentbl模块各个类的设计**

shipmentbl模块各个类的职责如下表所示。

**shipmentbl模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| ShipmentController | 负责实现装运界面所需要的服务 |
| Shipment | 装运的领域模型对象，拥有一次装运所持有的单据信息，可以帮助完成装运界面所需要的服务 |

（3）模块内部类的接口规范

ShipmentController和Shipment的接口规范如表所示。

ShipmentController**的接口规范**

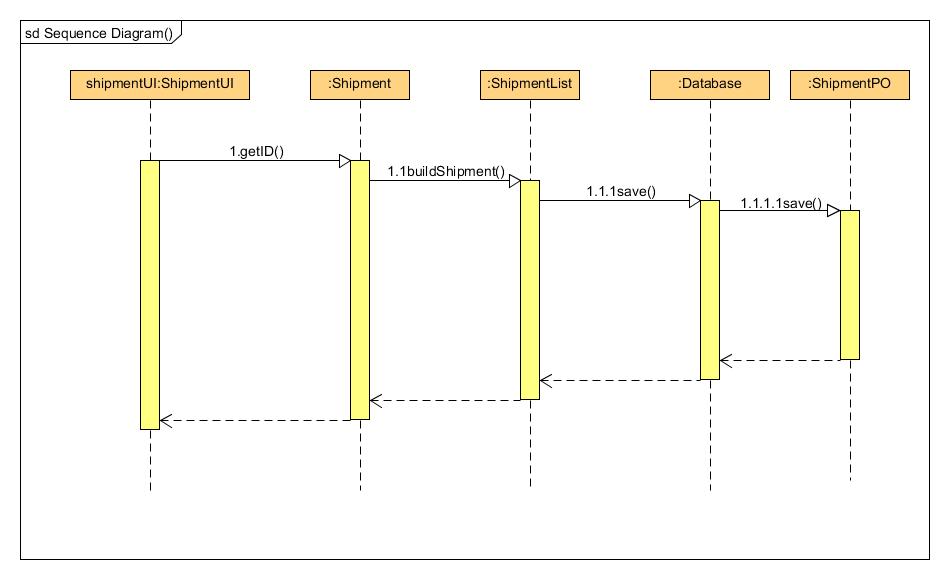
|  |  |  |  |
| --- | --- | --- | --- |
| **提供的接口（供接口）** | | | |
| **ShipmentController. submitBills** | 语法 | | public double submitBills (Transport bill) |
| 前置条件 | | 已创建一个Shipment领域对象 |
| 后置条件 | | 调用Shipment领域对象的submitBills方法 |
| **需要的接口（需接口）** | | | |
| **服务名** | | **服务** | |
| **Shipment. SubmitBills** | | 添加一个单据对象 | |

**Shipment的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的接口（供接口）** | | | |
| **Shipment. submitBills** | 语法 | | public String getTruckID () |
| 前置条件 | | 创建一个单据 |
| 后置条件 | | 保存此单据 |
| **需要的接口（需接口）** | | | |
| **服务名** | | **服务** | |
| **ShipmentDataService. getTruck**(String id) | | 根据ID进行查找单一持久化对象 | |
| **ShipmentDataService.insertTruck**(BusShipmentBill\_Center bill) | | 插入单一持久化对象 | |
| **ShipmentDataService.deleteTruck** (String id) | | 删除单一持久化对象 | |
| **ShipmentDataService.changeTruck**(BusShipmentBill\_Center bill) | | 更新单一持久化对象 | |
| **ShipmentDataService.getFreight** (String id) | | 根据ID进行查找单一持久化对象 | |
| **ShipmentDataService.insertFreight**(TransportBill\_Plane bill) | | 插入单一持久化对象 | |
| **ShipmentDataService.deleteFreight**(String id) | | 删除单一持久化对象 | |
| **ShipmentDataService.changeFreight**(TransportBill\_Plane bill) | | 更新单一持久化对象 | |
| **ShipmentDataService.init** | | 初始化持久数据 | |
| **ShipmentDataService. getReceive**(String id) | | 根据ID进行查找单一持久化对象 | |
| **ShipmentDataService.insertReceive**(ArrivementBill\_Center bill) | | 插入单一持久化对象 | |
| **ShipmentDataService.deleteReceive** (String id) | | 删除单一持久化对象 | |
| **ShipmentDataService.changeReceive**(ArrivementBill\_Center bill) | | 更新单一持久化对象 | |
| **DatabaseFacory.getShipmentDatabase** | | 得到Shipment数据库的引用 | |

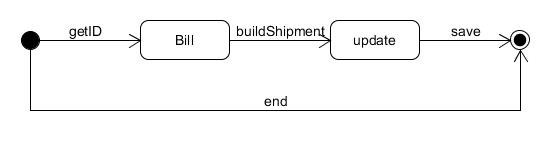
（4）业务逻辑层的动态模型

下图表明了物流信息管理系统中，当用户输入装运信息后，装运业务逻辑处理的相关对象之间的协作。



**新建装运单的顺序图**

如下图所示的状态图描述了Shipment对象的生存期间的状态序列、引起转移到事件，以及因状态转移而伴随的动作。随着getID方法被UI调用，Shipment进入Bill状态；之后通过修改数据进入update状态。



**Shipment对象状态图**

### receivementbl模块

(1)模块概述

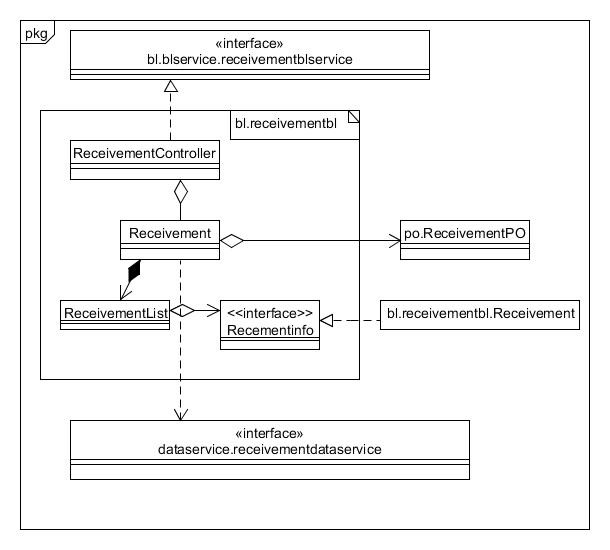
receivementbl模块承担的需求参见需求规格说明文档功能需求和相关非功能需求

receivementbl模块的职责及接口参见软件系统结构描述文档

(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bl.blservice.receivementblservice接口。业务逻辑层和数据层之间添加dataservice.receivementdataservice接口，为了隔离业务逻辑职责和逻辑控制职责，我们增加了ReceivementController，这样ReceivementController会将对收件的业务逻辑处理委托给Receivement对象。CommodityPO是作为收件货物信息记录的持久化对象被添加到设计模型中去的。而ReceivementList的添加是Receivementinfo的容器类。ReceivementList封装了收件货物信息的数据集合的数据结构的秘密。

Receivementbl模块的设计如图



receivementbl模块各个类的职责如表所示

**receivementbl模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| ReceivementController | 负责收件信息界面所需要的服务 |
| Receivement | 收件信息的领域模型对象，拥有收到的所有货物的详细信息，可以帮助收件业务所需要的服务 |

（3）模块内部类的接口规范

**ReceivementController模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceivementController.billFilled | 语法 | public boolean billFilled (ArrayList<ReceiveInformationVO> info) |
| 前置条件 | 已创建一个Receivement领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Receivement领域对象的billFilled方法 |
| ReceivementContoller.distribution | 语法 | public boolean distribution(ReceivementPO po) |
| 前置条件 | 已创建一个Receivement领域对象，输入符合输入规则 |
| 后置条件 | 调用Receivement领域对象的distribution方法 |
| ReceivementController.receive | 语法 | public boolean receive(ArrivementBill\_Shop arrive) |
| 前置条件 | 已创建一个Receiement领域对象，输入符合输入规则 |
| 后置条件 | 调用Receivement领域对象的receive方法 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Receivement.billFilled | 加入一个收件单对象信息 | |
| Receivement.distribution | 更新一个派件单对象信息 | |
| Receivement.receive | 更新一个营业厅到达单对象信息 | |

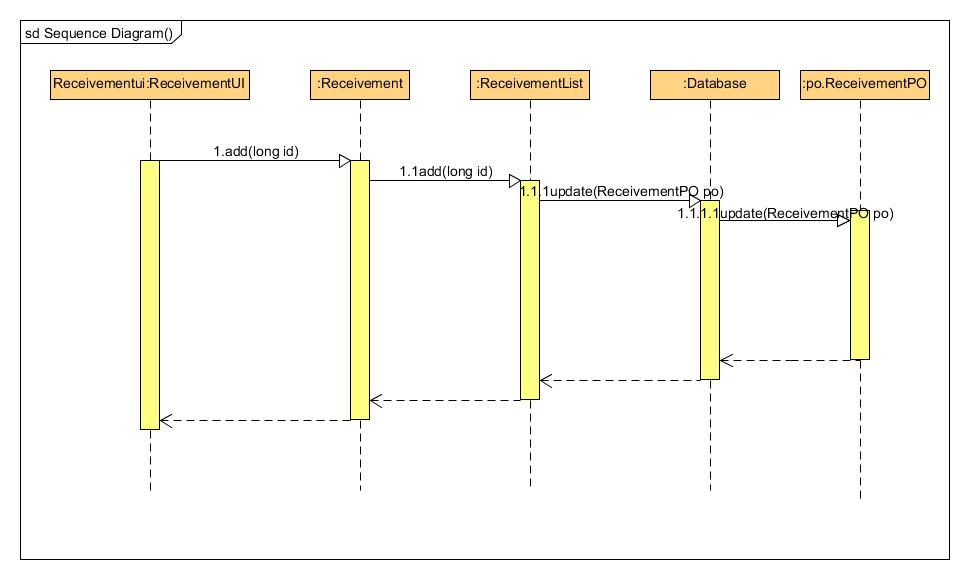
**Receivement的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Receivement.billFilled | 语法 | public boolean billFilled (ArrayList<ReceiveInformationVO> info) |
| 前置条件 | 已输入收件单信息 |
| 后置条件 | 系统持久化更新收件单信息 |
| Receivement.distribution | 语法 | Public boolean distribution(ReceivementPO po) |
| 前置条件 | 已输入派件单信息 |
| 后置条件 | 系统持久化更新派件单信息 |
| Receivement.receive | 语法 | public boolean receive(ArrivementBill\_Shop arrive) |
| 前置条件 | 已输入营业厅到达单信息 |
| 后置条件 | 系统持久化更新营业厅到达单信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Receivementdataservice.insert(ReceiveBill bill) | 插入单一持久化对象 | |
| receivementdataservice.init() | 初始化持久化数据 | |
| receivementdataservice.delete(String id) | 删除单一持久化对象 | |
| receivementdataservice.update(String id) | 更新单一持久化对象 | |
| receivementdataservice.get(String id) | 查找单一持久化对象 | |

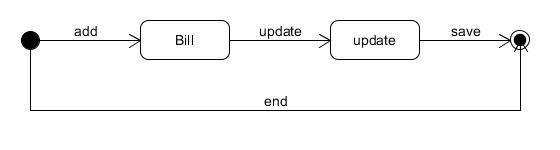
（4）业务逻辑层的动态模型

在物流信息管理系统中，当用户输入收件信息后，库存业务逻辑处理相关对象之间的协作。

得到收件信息的顺序图



如下图所示的状态图描述了Receivement对象的生存期间的状态序列、引起转移到事件，以及因状态转移而伴随的动作。随着add方法被UI调用，Receivement进入Bill状态；之后通过修改数据进入update状态。



(5)利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### commoditybl模块

(1)模块概述

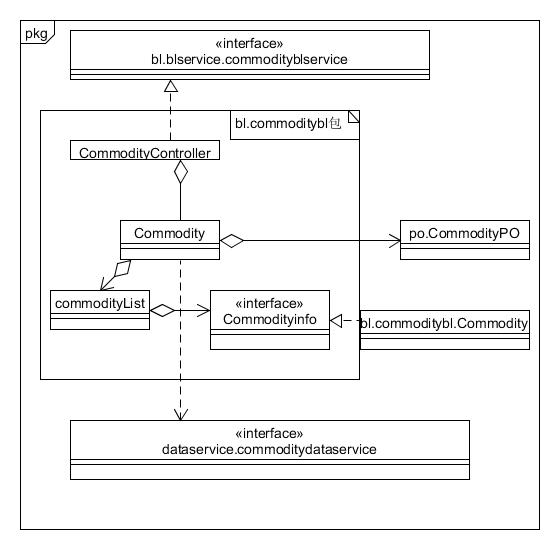
commoditybl模块承担的需求参见需求规格说明文档功能需求和相关非功能需求

commoditybl模块的职责及接口参见软件系统结构描述文档

(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，添加businesslogicservice.commodityblservice接口。为了隔离业务逻辑职责和逻辑控制职责，添加了CommodityController，这样CommodityController会将对库存管理的业务逻辑处理交给Commodity对象。CommodityPO是作为库存货物记录的持久化对象被添加到设计模型中去的。CommodityList的添加是CommodityInfo的容器类。CommodityList封装了库存货物的数据信息和数据结构的秘密。CommodityInfo是根据依赖倒置原则，为消除循环依赖而产生的接口。

commoditybl模块的设计如图。



commoditybl模块各个类的职责如表

**commoditybl模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| CommodityController | 负责库存信息界面所需要的服务 |
| Commodity | 库存信息的领域模型对象，拥有库存的所有货物的详细信息，可以帮助库存管理所需要的服务 |

(3)模块内部类的接口规范

**CommodityController的接口规范**

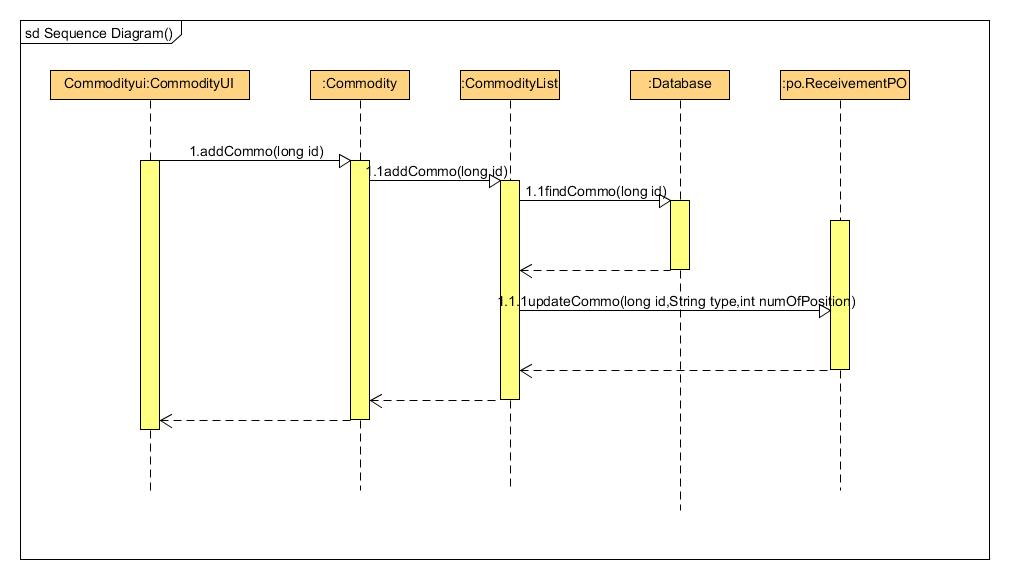
|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| CommodityController.inDepot | 语法 | public StockBill\_In inDepot(StockBill\_In stock\_in) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的inDepot方法 |
| CommodityContoller.getUnmarkingBills | 语法 | public ArrayList<stockBill> getUnmarkingBills () |
| 前置条件 | 已创建一个Commodity领域对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的getUnmarkingBills方法 |
| CommodityContoller. outDepot | 语法 | public StockBill\_Out outDepot(StockBill\_Out stock\_out) |
| 前置条件 | 已创建一个Commodity领域对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的outDepot方法 |
| CommodityContoller.areaAdjust | 语法 | public boolean areaAdjust(int[] rows, String type) |
| 前置条件 | 已创建一个Commodity领域对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的areaAdjust方法 |
| CommodityContoller.init | 语法 | public boolean init (int[] motorTransport, int[] trainTransport, int[] airTransport, int[] balnkSpace) |
| 前置条件 | 已创建一个Commodity领域对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的init方法 |
| CommodityContoller.getStockSnapShot | 语法 | public StockSnapShotSVO getStockSnapShot() |
| 前置条件 | 已创建一个Commodity领域对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的getStockSnapShot方法 |
| CommodityContoller.export | 语法 | public Boolean export() |
| 前置条件 | 已创建一个Commodity领域对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的export方法 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Commodity.inDepot | 加入一个入库单对象信息 | |
| Commodity.getUnmarkingBills | 加入未审判的单据信息 | |
| Commodity. outDepot | 加入一个出库单对象信息 | |
| Commodity.areaAdjust | 加入库区调整信息 | |
| Commodity.init | 库存信息初始化 | |
| Commodity.getStockSnapShot | 得到库存盘点信息 | |
| Commodity.export | 导出库存盘点 | |

**Commodity 的接口规范**

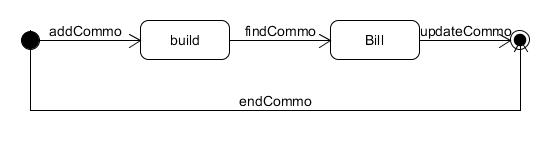
|  |  |  |  |
| --- | --- | --- | --- |
| **提供的接口（供接口）** | | | |
| Commodity.inDepot | 语法 | | public StockBill\_In inDepot(StockBill\_In stock\_in) |
| 前置条件 | | 输入合法 |
| 后置条件 | | 加入一个入库单对象信息 |
| Commodity.getUnmarkingBills | 语法 | | public ArrayList<stockBill> getUnmarkingBills () |
| 前置条件 | | 输入合法 |
| 后置条件 | | 加入未审判的单据信息 |
| Commodity. outDepot | 语法 | | public StockBill\_Out outDepot(StockBill\_Out stock\_out) |
| 前置条件 | | 输入合法 |
| 后置条件 | | 加入一个出库单对象信息 |
| Commodity.areaAdjust | 语法 | | public boolean areaAdjust(int[] rows, String type) |
| 前置条件 | | 输入合法 |
| 后置条件 | | 加入库区调整信息 |
| Commodity.init | 语法 | | public boolean init (int[] motorTransport, int[] trainTransport, int[] airTransport, int[] balnkSpace) |
| 前置条件 | | 输入合法 |
| 后置条件 | | 库存信息初始化 |
| Commodity.getStockSnapShot | 语法 | | public StockSnapShotSVO getStockSnapShot() |
| 前置条件 | | 输入合法 |
| 后置条件 | | 得到库存盘点信息 |
| Commodity.export | 语法 | | public boolean export() |
| 前置条件 | | 输入合法 |
| 后置条件 | | 导出库存盘点 |
| **需要的接口（需接口）** | | | |
| 服务名 | | 服务 | |
| Commoditydataservice.insertIndepot(StockBill\_In bill) | | 添加单一持久化对象 | |
| Commoditydataservice.deleteIndepot (String id) | | 删除单一持久化对象 | |
| Commoditydataservice.changeIndepot (StockBill\_In bill) | | 更新单一持久化对象 | |
| Commoditydataservice.getIndepot (String id) | | 根据编号查找相应持久化对象 | |
| Commoditydataservice.insertOutdepot(StockBill\_Out bill) | | 添加单一持久化对象 | |
| Commoditydataservice.deleteOutdepot (String id) | | 删除单一持久化对象 | |
| Commoditydataservice.changeOutdepot (StockBill\_Out bill) | | 更新单一持久化对象 | |
| Commoditydataservice.getOutdepot (String id) | | 根据编号查找相应持久化对象 | |
| Commoditydataservice.getCommoditydata | | 得到commodity数据的服务引用 | |
| Commoditydataservice.init() | | 初始化持久数据 | |

(4)业务逻辑层的动态模型

在物流信息管理系统中，当用户输入了货物信息后，库存业务逻辑处理相关对象之间的协作。



如下图所示的状态图描述了Commodity对象的生存期间的状态序列、引起转移到事件，以及因状态转移而伴随的动作。随着addCommo方法被UI调用，Commodity进入Bill状态；之后通过修改数据进入update状态。



(5)业务逻辑设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### 4.1.7 moneybl模块

1. 模块概述

moneybl模块承担的需求见需求规格说明文档功能需求及相关非功能需求。

moneybl模块的职责及接口见软件系统结构描述文档。

1. 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。每层之间为了增加灵活性，我们会添加接口。比如业务逻辑层和数据层之间添加dataservice.moneydataservice.MoneyDataService接口，展示层和业务逻辑层之间添加businesslogicservice.moneyblservice,MoneyBLService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了MoneyController，MoneyController会将对财务管理的业务逻辑处理委托给Money对象。MoneyPO是作为财务记录的持久化对象被添加到设计模型中去的。MoneyList和MoneyItem的添加是MoneyInfo的容器类。MoneyItem保有财务出入的详细条目，以及计算财务收益成本的职责。MoneyList封装了MoneyItem,paymentItem,expenseItem和Operating的数据集合的数据结构的秘密。MoneyInfo和AccountInfo分别保有财务的账目信息和账户信息，paymentInfo,expenseInfo分别保有财务的收款信息和付款信息，MoneyInfo和MemberInfo都是根据依赖倒置原则，为了消除循环依赖而产生的接口。

moneybl模块的设计如图

pkg

<<interface>>

businesslogicservice.moneyblservice.MoneyBLService

po.MoneyPO

Businesslogic.moneybl包

moneyController

businesslogic.moneybl.Account

<<interface>>

AccountInfo

account

money

<<interface>>

paymentInfo

businesslogic.moneybl.Payment

paymentItem

Businesslogic.moneybl.expense

expenseItem

<<interface>>

expenseInfo

Businesslogic.moneybl.money

<<interface>>

moneyInfo

moneyItem

<<interface>>

dataservice.moneydataservice.MoneyDataService

**moneybl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应登录界面所需要的服务 |
| MoneyController | 负责财务管理界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的名称和密码，可以解决登录问题 |
| Money | 财务管理的领域模型对象，拥有财务管理的所有账目的详细信息，可以帮助完成财务管理所需要的服务 |

1. 模块内部类的接口规范

**MoneyController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| MoneyController.addAccount | | 语法 | public boolean addAccount(ArrayList<Account> account) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money领域对象的addAccount方法 |
| MoneyController.deleteAccount | | 语法 | public boolean deleteAccount (ArrayList<String> ID) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money领域对象的deleteAccount方法 |
| MoneyController.changeAccountInfo | | 语法 | public boolean changeAccountInfo (ArrayList<String> ID, ArrayList<String> name) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money领域对象的changeAccountInfo方法 |
| MoneyController.getAccount | | 语法 | public ArrayList<Account> getAccount (String KeyWord) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money领域对象的getAccount方法 |
| MoneyController.getBills | | 语法 | public ReceiveMoneyBill[] getBills (String[] Date, String bussinessHall) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money领域对象的getBills方法 |
| MoneyController.getTotalMoney | | 语法 | Public double getTotalMoney () |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money领域对象的getTotalMoney方法 |
| MoneyController.addPaidmentBill | | 语法 | Public boolean addPaidmentBill (ReceiveMoneyBill[] bills) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money 领域对象的addPaidmentBill方法 |
| MoneyController. getPaidmentBill | | 语法 | Public ArrayList<ReceiveMoneyBill> getPaidmentBill (String[] time) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money 领域对象的getPaidmentBill方法 |
| MoneyController.getUnmarkingBills | | 语法 | Public ArrayList<ReceiveMoneyBill> getUnmarkingBills () |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money 领域对象的getUnmarkingBills方法 |
| MoneyController.addPaymentBill | | 语法 | Public boolean addPaymentBill (ArrayList<PaymentBill>] bills) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money 领域对象的addPaidmentBill方法 |
| MoneyController. getPaymentBill | | 语法 | Public ArrayList<PaymentBill> getPaymentBill (String[] endTime) |
| 前置条件 | 已创建一个Money领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Money 领域对象的getPaymentBill方法 |
| 需要的服务（需接口） | | | |
| 服务名 | 服务 | | |
| MoneyController.addAccount | 加入一个账户的信息 | | |
| MoneyController.deleteAccount | 删除一个账户的信息 | | |
| MoneyController.changeAccountInfo | 更新一个账户的信息 | | |
| MoneyController.getAccount | 查看一个账户的信息 | | |
| MoneyController.getBills | 得到当天的收款单信息 | | |
| MoneyController.getTotalMoney | 得到所查看的收款单总金额 | | |
| MoneyController.addPaidmentBill | 加入一个收款单信息 | | |
| MoneyController. getPaidmentBill | 得到收款单信息 | | |
| MoneyController.getUnmarkingBills | 添加一个未审判的单据 | | |
| MoneyController.addPaymentBill | 加入一个付款单信息 | | |
| MoneyController. getPaymentBill | 得到付款单信息 | | |

**Money的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供服务） | | | |
| Money.addAccount | 语法 | | public boolean addAccount(ArrayList<Account> account) |
|  | 前置条件 | | 输入合法 |
|  | 后置条件 | | 加入一个账户的信息 |
| Money.deleteAccount | 语法 | | public boolean deleteAccount (ArrayList<String> ID) |
|  | 前置条件 | | 输入合法 |
|  | 后置条件 | | 删除一个账户的信息 |
| Money.changeAccountInfo | 语法 | | public boolean changeAccountInfo (ArrayList<String> ID, ArrayList<String> name) |
|  | 前置条件 | | 输入合法 |
|  | 后置条件 | | 更新一个账户的信息 |
| Money.getAccount | 语法 | | public ArrayList<Account> getAccount (String KeyWord) |
|  | 前置条件 | | 输入合法 |
|  | 后置条件 | | 查看一个账户的信息 |
| Money.getBills | 语法 | | public ReceiveMoneyBill[] getBills (String[] Date, String bussinessHall) |
|  | 前置条件 | | 输入合法 |
|  | 后置条件 | | 得到当天的收款单信息 |
| Money.getTotalMoney | 语法 | | Public double getTotalMoney () |
|  | 前置条件 | | 输入合法 |
|  | 后置条件 | | 得到所查看的收款单总金额 |
| Money.addPaidmentBill | 语法 | | Public boolean addPaidmentBill (ReceiveMoneyBill[] bills) |
|  | 前置条件 | | 输入合法 |
|  | 后置条件 | | 加入一个收款单信息 |
| Money. getPaidmentBill | 语法 | | Public ArrayList<ReceiveMoneyBill> getPaidmentBill (String[] time) |
| 前置条件 | | 输入合法 |
| 后置条件 | | 得到收款单信息 |
| Money.getUnmarkingBills | 语法 | | Public ArrayList<ReceiveMoneyBill> getUnmarkingBills () |
| 前置条件 | | 输入合法 |
| 后置条件 | | 添加一个未审判的单据 |
| Money.addPaymentBill | 语法 | | Public boolean addPaymentBill (ArrayList<PaymentBill>] bills) |
| 前置条件 | | 输入合法 |
| 后置条件 | | 加入一个付款单信息 |
| Money. getPaymentBill | 语法 | | Public ArrayList<PaymentBill> getPaymentBill (String[] endTime) |
| 前置条件 | | 输入合法 |
| 后置条件 | | 得到付款单信息 |
| 需要的服务（需服务） | | | |
| 服务名 | | 服务 | |
| Moneydataservice.insert(Account account) | | 添加单一持久化对象 | |
| Moneydataservice.delete(String id) | | 删除单一持久化对象 | |
| Moneydataservice.change(Account account) | | 更新单一持久化对象 | |
| Moneydataservice.get(String id) | | 根据编号查找相应持久化对象 | |
| Moneydataservice.init() | | 初始化单一持久化对象 | |
| Moneydataservice.insertMember（String id) | | 添加单一持久化账户对象 | |
| Moneydataservice.deleteMember（String id) | | 删除单一持久化账户对象 | |
| Moneydataservice.getMember（String id) | | 查找单一持久化账户对象 | |
| Moneydataservice.getPaid | | 得到付款单ID | |
| Moneydataservice.getExpenseid | | 得到收款单ID | |

1. 业务逻辑层的动态模型

在物流信息管理系统中，加入一个新账户的动态模型如下

addAccount(String name,int money)

creatPO

addAccount(MoneyPO po)

 得到付款单的顺序图和状态图

MoneyDataBase

MoneyPO

Account

MoneyUI:addAccount

getPayment(MoneyPO po)

creatMessage()

MoneyPO

PaymentList

Payment

MoneyUI:getpayment

PaymentItem

getPayment

getPayment getPayment end

PaymentList

PaymentItem

状态图

得到收款单的顺序图和状态图如下所示

getExpense(MoneyPO po)

creatMessage()

MoneyPO

ExpenseList

Expense

MoneyUI:getExpense

ExpenseItem

getExpense

getExpense getExpense end

ExpenseList

ExpenseItem

状态图

得到成本收益表的顺序图和状态图如下

getItem(MoneyPO po)

creatMessage()

ExpenseItem

MoneyItem

MoneyPO

MoneyUI:getItem

PaymentItem

MoneyItem

end

getItem

getItem getItem

PaymentItem

ExpenseItem

状态图

得到经营情况表的顺序图和状态图如下

getOperating(MoneyPO po)

creatMessage()

MoneyUI:getOperating

ExpenseList

Operating

MoneyPO

PaymentList

Operating

end

getOperating

getOperating getOperating

PaymentList

ExpenseList

状态图

1. 业务逻辑设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### 4.1.8 reportbl模块

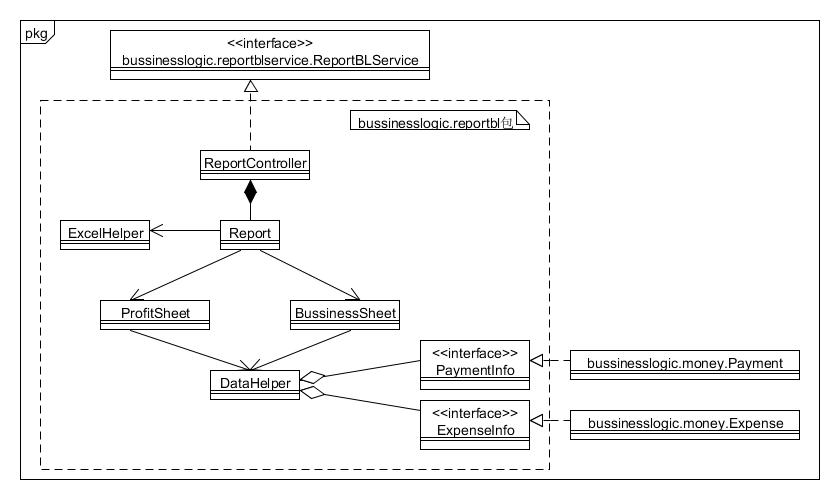
1. 模块概述

Reportbl模块负责实现对成本收益表和经营情况表的查看和导出。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

(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。展示层和业务逻辑层之间，我们添加bussinesslogicservice. ReportBLService接口。Report模块没有需要持久化的数据，所以该模块没有对应的数据层。为了隔离业务逻辑职责和逻辑控制职责，我们增加了ReportController，这样ReportController会将对查看报表的逻辑业务处理委托给Report对象。

Reportbl模块类的设计



**reportbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ReportController | 负责实现报表查看界面所需要的服务 |
| Report | 报表查看的领域模型对象，拥有经营情况表和成本收益表对象 |
| BussinessSheet | 经营情况表的对象模型，拥有对应报表的信息 |
| ProfitSheet | 成本收益表的对象模型，拥有对应报表的信息 |
| ExcelHelper | 负责导出报表为excel文件 |
| DataHelper | 负责收款单和付款单的查看 |

（3）模块内部类的接口规范

ReportController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReportController.show | 语法 | public VO show() |
| 前置条件 | 已创建一个Report领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Report 领域对象的show方法 |
| ReportController.export | 语法 | public boolean export() |
| 前置条件 | 已创建一个Report领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Report 领域对象的export方法 |
| 需要的接口(需接口) | | |
| ReportController.show | 显示一个表对象 | |
| ReportController.export | 导出一个表对象 | |

Report的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的接口（供接口） | | |
| Report.show | 语法 | public VO show() |
| 前置条件 | 输入合法 |
| 后置条件 | 显示一个表对象 |
| Report.export | 语法 | public boolean export() |
| 前置条件 | 输入合法 |
| 后置条件 | 导出一个表对象 |

### staffbl模块

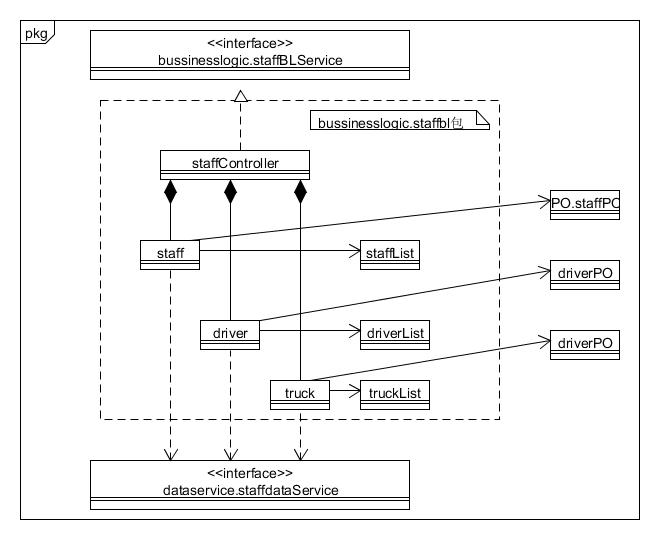
（1）模块概述

Staffbl模块负责实现对人员与机构、车辆信息、司机信息、用户的管理。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

（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。展示层和业务逻辑层之间，我们添加bussinesslogicservice. StaffBLService接口。Staff模块没有需要持久化的数据，所以该模块没有对应的数据层。为了隔离业务逻辑职责和逻辑控制职责，我们增加了StaffController，这样StaffController会将各种逻辑业务处理委托给Staff对象。

Staffbl模块类的设计



**Staffbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| StaffController | 负责实现人员与机构管理、车辆信息管理、司机信息管理、用户管理所需要的服务 |
| Staff | 人员管理的领域模型对象，拥有人员的各种信息 |
| StaffList | 人员管理中存放人员PO的容器 |
| Driver | 司机管理的领域模型对象，拥有司机的各种信息 |
| DriverList | 司机管理中存放司机PO的容器 |
| Truck | 货车管理的领域模型对象，拥有货车的各种信息 |
| TruckList | 货车管理中存放货车PO的容器 |

（3）模块内部类的接口规范

StaffController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StaffConrtoller.addNewStaff | 语法 | public boolean addNewStaff (StaffVO[] staffs) |
| 前置条件 | 已创建一个Staff领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Staff 领域对象的addNewStaff方法 |
| StaffConrtoller.deleteStaff | 语法 | public boolean deleteStaff (String[] ID) |
| 前置条件 | 已创建一个Staff领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Staff 领域对象的deleteStaff方法 |
| StaffConrtoller.changePosition | 语法 | public boolean changePosition (String[] ID, String[] newPositon, String[] newDepartment) |
| 前置条件 | 已创建一个Staff领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Staff 领域对象的changePosition方法 |
| StaffConrtoller.changesalarymodel | 语法 | public boolean changesalarymodel (String[] ID, String[] salaryModel) |
| 前置条件 | 已创建一个Staff领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Staff 领域对象的changesalarymodel方法 |
| StaffConrtoller. getStaff | 语法 | public StaffVO getStaff (String ID) |
| 前置条件 | 已创建一个Staff领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Staff 领域对象的getStaff方法 |
| StaffConrtoller.getAllStaff | 语法 | public ArrayList<StaffVO> getAllStaff (String src) |
| 前置条件 | 已创建一个Staff领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Staff 领域对象的getAllStaff方法 |
| StaffConrtoller.changeStaff | 语法 | public boolean changeStaff (String ID,String competence, String password, String name, String power) |
| 前置条件 | 已创建一个Staff领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Staff 领域对象的changeStaff方法 |
| StaffConrtoller.addNewDriver | 语法 | public boolean addNewDriver (ArrayList<DriverPO> drivers) |
| 前置条件 | 已创建一个Driver领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver 领域对象的addNewDriver方法 |
| StaffConrtoller.deleteDriver | 语法 | public boolean deleteDriver (ArrayList<String> ID) |
| 前置条件 | 已创建一个Driver领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver 领域对象的deleteDriver方法 |
| StaffController.ChangeDriverInfo | 语法 | public boolean ChangeDriverInfo (String[] ID, String[] IDCardNumber, String[] birthday, boolean[] isBoy, String[] TelNumber, String[] validate) |
| 前置条件 | 已创建一个Driver领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver领域对象的ChangeDriverInfo方法 |
| StaffController.getDriver | 语法 | public DriverPO getDriver (String ID) |
| 前置条件 | 已创建一个Driver领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver 领域对象的getDriver方法 |
| StaffConrtoller.getAllDriver | 语法 | public ArrayList<DriverPO> getAllDriver (String str) |
| 前置条件 | 已创建一个Driver领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver 领域对象的getAllDriver方法 |
| StaffConrtoller.addNewTruck | 语法 | public boolean addNewTruck (TruckPO[] trucks) |
| 前置条件 | 已创建一个Truck领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Truck 领域对象的addNewTruck方法 |
| StaffConrtoller.deleteTruck | 语法 | public boolean deleteTruck (String[] ID) |
| 前置条件 | 已创建一个Truck领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Truck 领域对象的deleteTruck方法 |
| StaffController.ChangeTruckInfo | 语法 | public boolean ChangeTruckInfo (String[] truckID, String[] licensePlate, String[] inTime) |
| 前置条件 | 已创建一个Truck领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Truck 领域对象的ChangeTruckInfo方法 |
| StaffController.getTruck | 语法 | public TruckPO getTruck (String ID) |
| 前置条件 | 已创建一个Truck领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Truck 领域对象的getTruck方法 |
| StaffConrtoller.getAllTruck | 语法 | public ArrayList<TruckPO> getAllTruck () |
| 前置条件 | 已创建一个Truck领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Truck 领域对象的getAllTruck方法 |
| 需要的接口（需接口） | | |
| StaffConrtoller.addNewStaff | 添加新员工信息 | |
| StaffConrtoller.deleteStaff | 删除员工信息 | |
| StaffConrtoller.changePosition | 更新员工职位信息 | |
| StaffConrtoller.changesalarymodel | 更新员工薪水策略 | |
| StaffConrtoller. getStaff | 查看员工信息 | |
| StaffConrtoller.getAllStaff | 查看所有员工信息 | |
| StaffConrtoller.changeStaff | 修改员工权限 | |
| StaffConrtoller.addNewDriver | 添加新司机信息 | |
| StaffConrtoller.deleteDriver | 删除司机信息 | |
| StaffController.ChangeDriverInfo | 更新司机信息 | |
| StaffController.getDriver | 查看司机信息 | |
| StaffConrtoller.getAllDriver | 查看所有司机信息 | |
| StaffConrtoller.addNewTruck | 添加新车辆信息 | |
| StaffConrtoller.deleteTruck | 删除车辆信息 | |
| StaffController.ChangeTruckInfo | 更新车辆信息 | |
| StaffController.getTruck | 查看车辆信息 | |
| StaffConrtoller.getAllTruck | 查看所有车辆信息 | |

Staff的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Staff.addNewStaff | | 语法 | public boolean addNewStaff (StaffVO[] staffs) |
| 前置条件 | 输入合法 |
| 后置条件 | 添加新员工信息 |
| Staff.deleteStaff | | 语法 | public boolean deleteStaff (String[] ID) |
| 前置条件 | 输入合法 |
| 后置条件 | 删除员工信息 |
| Staff.changePosition | | 语法 | public boolean changePosition (String[] ID, String[] newPositon, String[] newDepartment) |
| 前置条件 | 输入合法 |
| 后置条件 | 更新员工职位信息 |
| Staff.changesalarymodel | | 语法 | public boolean changesalarymodel (String[] ID, String[] salaryModel) |
| 前置条件 | 输入合法 |
| 后置条件 | 更新员工薪水策略 |
| Staff. getStaff | | 语法 | public StaffVO getStaff (String ID) |
| 前置条件 | 输入合法 |
| 后置条件 | 查看员工信息 |
| Staff.getAllStaff | | 语法 | public ArrayList<StaffVO> getAllStaff (String src) |
| 前置条件 | 输入合法 |
| 后置条件 | 查看所有员工信息 |
| Staff.changeStaff | | 语法 | public boolean changeStaff (String ID,String competence, String password, String name, String power) |
| 前置条件 | 输入合法 |
| 后置条件 | 修改员工权限 |
| 需要的服务（需接口） | | | |
| StaffDataService.insertStaff | 插入一条员工信息 | | |
| StaffDataService.getStaff | 查找员工信息 | | |
| StaffDataService.delete | 删除员工信息 | | |
| StaffDataService.change | 更新员工信息 | | |
| StaffDataService.showStaff | 显示员工信息 | | |
| StaffDataService.init | 初始化持久化数据 | | |

Driver的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Staff.addNewDriver | 语法 | public boolean addNewDriver (DriverPO po) |
| 前置条件 | 输入合法 |
| 后置条件 | 添加新司机信息 |
| Staff.deleteDriver | 语法 | public boolean deleteDriver(String id) |
| 前置条件 | 输入合法 |
| 后置条件 | 删除司机信息 |
| Staff.ChangeDriverInfo | 语法 | public boolean ChangeDriverInfo (String[] ID, String[] IDCardNumber, String[] birthday, boolean[] isBoy, String[] TelNumber, String[] validate) |
| 前置条件 | 输入合法 |
| 后置条件 | 更新司机信息 |
| Staff.getDriver | 语法 | public DriverPO getDriver (String ID) |
| 前置条件 | 输入合法 |
| 后置条件 | 查看司机信息 |
| Staff.getAllDriver | 语法 | **Public** ArrayList<DriverPO> getAll(String str) |
| 前置条件 | 输入合法 |
| 后置条件 | 查看所有司机信息 |
| 需要的服务（需接口） | | |
| StaffDataService.insertDriver | 插入一条司机信息 | |
| StaffDataService.getDriver | 查找司机信息 | |
| StaffDataService.delete | 删除司机信息 | |
| StaffDataService.change | 更新司机信息 | |
| StaffDataService.showDriver | 显示司机信息 | |
| StaffDataService.init | 初始化持久化数据 | |

Truck的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Staff.addNewTruck | | 语法 | public boolean addNewTruck(TruckPO po) |
| 前置条件 | 输入合法 |
| 后置条件 | 添加新车辆信息 |
| Staff.deleteTruck | | 语法 | public boolean deleteTruck (String id) |
| 前置条件 | 输入合法 |
| 后置条件 | 删除车辆信息 |
| Staff.ChangeTruckInfo | | 语法 | public boolean ChangeTruckInfo (TruckPO po) |
| 前置条件 | 输入合法 |
| 后置条件 | 更新车辆信息 |
| Staff.getTruck | | 语法 | public TruckPO getTruck (String id) |
| 前置条件 | 输入合法 |
| 后置条件 | 查看车辆信息 |
| Staff.getAllTruck | | 语法 | public ArrayList<TruckPO> getAll() |
| 前置条件 | 输入合法 |
| 后置条件 | 查看所有车辆信息 |
| 需要的服务（需接口） | | | |
| StaffDataService.insertTruck | 插入一条车辆信息 | | |
| StaffDataService.getTruck | 查找车辆信息 | | |
| StaffDataService.delete | 删除车辆信息 | | |
| StaffDataService.change | 更新车辆信息 | | |
| StaffDataService.showTruck | 显示车辆信息 | | |
| StaffDataService.init | 初始化持久化数据 | | |

## 4.2数据层的分解

下图为数据层的开发包图。

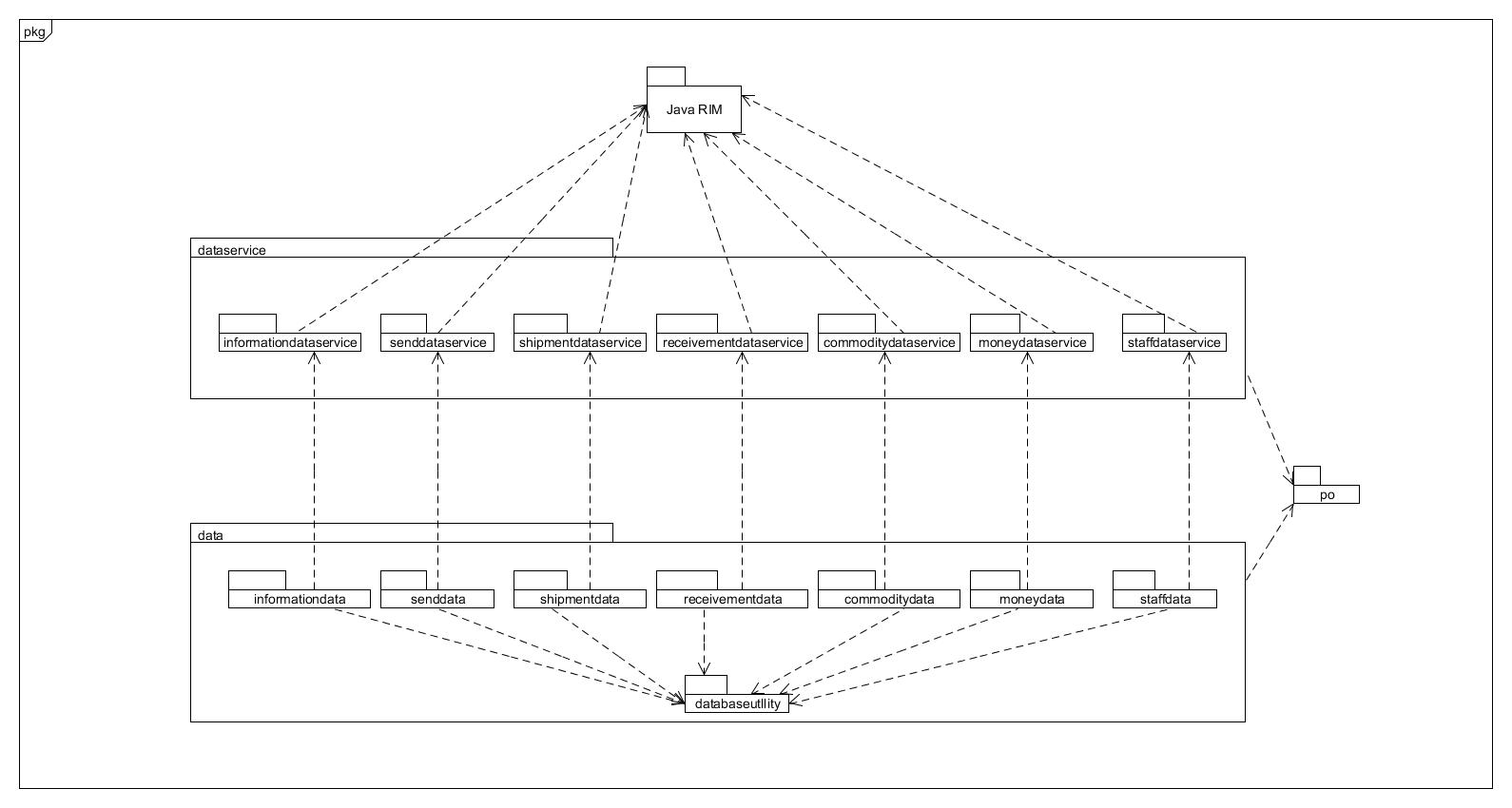


图4 物流系统服务器端开发包图

### 4.2.1 InformationData模块

（1）模块概述

InformationDataService模块提供物流信息查询的服务

（2）模块内接口规范如下

|  |  |  |
| --- | --- | --- |
| **提供的接口（供接口）** | | |
| Informationdataservice.getTransMesg | 语法 | Public ArrayList<TransMesgPO> getTransMesg(String id) |
| 前置条件 | 快递信息已经返回 |
| 后置条件 | 无 |
| Informationdataservice.refreshTransMesg | 语法 | **public** **boolean** refreshTransMesg(TransMesgPO po) |
| 前置条件 | 货物接收成功 |
| 后置条件 | 系统保存货物物流信息 |

### 4.2.2 SendData模块

（1）模块概述

Sendblservice模块提供订单信息输入、查询、报价和时间查询的服务

（2）模块内部类的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的接口（供接口）** | | |
| Senddataservice.getSendMesg | 语法 | **public** OrderBillPO getSendMesg(String id) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Senddataservice.getInput | 语法 | **public** **boolean** getInput(OrderBillPO bill) |
| 前置条件 | 使用者有写入的权限 |
| 后置条件 | 系统保存写入的订单信息 |

### 4.2.3 shipmentdata模块

（1）模块概述

ShipmentDataService模块提供装运单据数据集体载入、修改、保存

（2）模块内部类的接口规范

ShipmentDataService类的接口规范如下表所示

**shipmentdata模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的接口（供接口）** | | |
| **ShipmentDataService.getTruck** | 语法 | **public** BusShipmentBill\_Center getTruck(String id) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的ShipmentBillPO结果 |
| **ShipmentDataService.insertTruck** | 语法 | **public** **boolean** insertTruck(BusShipmentBill\_Center bill) |
| 前置条件 | 同样ID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 |
| **ShipmentDataService.deleteTruck** | 语法 | **public** **boolean** deleteTruck(String id) |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| **ShipmentDataService.changeTruck** | 语法 | **public** **boolean** changeTruck(BusShipmentBill\_Center bill) |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除一个po |
| **ShipmentDataService.initTruck** | 语法 | **public** **boolean** initTruck() |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| **ShipmentDataService.getFreight** | 语法 | **public** TransportBill\_Plane getFreight(String id) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的ShipmentBillPO结果 |
| **ShipmentDataService.insertFreight** | 语法 | **public** **boolean** **insertFreight** (TransportBill\_Plane bill) |
| 前置条件 | 同样ID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 |
| **ShipmentDataService.deleteFreight** | 语法 | **public** **boolean** deleteFreight(String id) |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| **ShipmentDataService.** **changeFreight** | 语法 | **public** **boolean** changeFreight(TransportBill\_Plane bill) |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除一个po |
| **ShipmentDataService.initFreight** | 语法 | **Public boolean initFreight()** |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| **ShipmentDataService.getReceive** | 语法 | **public** ArrivementBill\_Center getReceive(String id) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的ShipmentBillPO结果 |
| **ShipmentDataService.insertReceive** | 语法 | **public** **boolean** insertReceive(ArrivementBill\_Center bill) |
| 前置条件 | 同样ID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 |
| **ShipmentDataService.deleteReceive** | 语法 | **public** **boolean** deleteReceive(String id) |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| **ShipmentDataService.changeReceive** | 语法 | **public** **boolean** changeReceive(ArrivementBill\_Center bill) |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除一个po |
| **ShipmentDataService.initReceive** | 语法 | **public** **boolean initReceive**() |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |

### 4.2.4 receivementdata模块

（1）模块概述

ReceivementDataService提供接收单单据数据集体载入、修改、保存

（2）模块内部类的接口规范

ReceivementDataService类的接口规范如下表所示

**receivementdataservice 模块的接口规范**

|  |  |  |
| --- | --- | --- |
|  | 提供的服务 | （供接口） |
| receivementdataserv  ice.get | 语法 | **public** ReceiveBill get(String id) |
| 前置条件 | 无 |
| 后置条件 | 查找返回相应的结果 |
| receivementdataservice.delete | 语法 | **public** **boolean** delete(String id) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 删除po |
| receivementdataservice.update | 语法 | **public** **boolean** update(String id) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新po |
| receivementdataservice.init | 语法 | public boolean init() |
| 前置条件 | 无 |
| 后置条件 | 初始化持久数据 |
| receivementdataservice.insert | 语法 | **public** **boolean** insert(ReceiveBill bill) |
| 前置条件 | 同样id的po在数据中不存在 |
| 后置条件 | 在数据库中增加一条po记录 |

### 4.2.5 commoditydata模块

（1）模块概述

CommodityDataService提供库存单据数据集体载入、修改、保存

（2）模块内部类的接口规范

CommodityDataService类的接口规范如下表所示

**commoditydataservice模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| commoditydataservice.insertIndepot | 语法 | **public** **boolean** insertIndepot (StockBill\_In bill) **throws** RemoteException) |
| 前置条件 | 同样的po在数据中不存在 |
| 后置条件 | 插入一条po数据 |
| commoditydataservice.deleteIndepot | 语法 | **public** **boolean** deleteIndepot (String id) **throws** RemoteException |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 删除一条po数据 |
| commoditydataservice.changeIndepot | 语法 | **public** **boolean** changeIndepot(StockBill\_In bill) **throws** RemoteException |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新Po数据 |
| commoditydataservice.getIndepot | 语法 | **public** StockBill\_In getIndepot (String id) **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回查找相应的结果 |
| commoditydataservice.initIndepot | 语法 | **public** **boolean** initIndepot () **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久数据 |
| commoditydataservice.insertOutdepot | 语法 | **public** **boolean** insertOutdepot (StockBill\_Out bill) **throws** RemoteException |
| 前置条件 | 同样的po在数据中不存在 |
| 后置条件 | 插入一条po数据 |
| commoditydataservice.deleteOutdepot | 语法 | **public** **boolean** deleteOutdepot (String id) **throws** RemoteException |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 删除一条po数据 |
| commoditydataservice.changeOutdepot | 语法 | **public** **boolean** changeOutdepot (StockBill\_Out bill) **throws** RemoteException |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新Po数据 |
| commoditydataservice.getOutdepot | 语法 | **public** StockBill\_Out getOutdepot (String id) **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回查找相应的结果 |
| commoditydataservice.initOutdepot | 语法 | **public** **boolean** initOutdepot () **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久数据 |

### 4.2.6 moneydata模块

（1）模块概述

MoneyDataService提供财务单据数据集体载入、修改、保存

（2）模块内部类的接口规范

MoneyDataService类的接口规范如下表所示

**moneydataservicce模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| moneydataservice.insert | 语法 | **public** **boolean** insert(Account account) |
| 前置条件 | 同样的po在数据中不存在 |
| 后置条件 | 增加一条po记录 |
| moneydataservice.delete | 语法 | **public** **boolean** delete(String id) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 删除一条po数据 |
| moneydataservice.change | 语法 | **public** **boolean** change(Account account) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新一条po数据 |
| moneydataservice.get | 语法 | **public** Account get(String id) |
| 前置条件 | 无 |
| 后置条件 | 查找返回相应的结果 |
| moneydataservice.init | 语法 | **public** **boolean** init() |
| 前置条件 | 无 |
| 后置条件 | 初始化持久数据 |
| moneydataservice.getPaid | 语法 | **public** PaymentBill get getPaid (String id) |
| 前置条件 | 无 |
| 后置条件 | 返回付款单id |
| moneydataservie.getExpenseid | 语法 | **public** ReceiveMoneyBill get getExpenseid (String id) |
| 前置条件 | 无 |
| 后置条件 | 返回收款单id |
| moneydataservice. getUnjudged | 语法 | **public** ArrayList<PaymentBill> getUnjudged() **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回成本收益单id |
| moneydataservice.getMember | 语法 | **public** Account getMember (String id) **throws** RemoteException |
| 前置条件 | 账户已在系统中建立账号 |
| 后置条件 | 返回账户的账号名称 |
| moneydataservice.getAll | 语法 | **public** ArrayList<Account> getAll(String id) **throws** RemoteException |
| 前置条件 | 账户已在系统中进行交易 |
| 后置条件 | 返回查看账户的交易记录（交易金额） |

### 4.2.7 StaffData模块

（1）模块概述

StaffDataService模块提供人员、车辆、司机的信息存取

（2）模块内类接口规范

StaffDataService接口规范如下表

StaffDataService模块接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StaffDataService.insertStaff | 语法 | **public** **boolean** insertStaff (StaffPO po) **throws** RemoteException |
| 前置条件 | 同样的po在数据中不存在 |
| 后置条件 | 增加一条po数据 |
| StaffDataService.insertDriver | 语法 | **public** **boolean** insert(DriverPO po) **throws** RemoteException |
| 前置条件 | 同样的po在数据中不存在 |
| 后置条件 | 增加一条po数据 |
| StaffDataService.insertTruck | 语法 | **public** **boolean** insert(TruckPO po) **throws** RemoteException |
| 前置条件 | 同样的po在数据中不存在 |
| 后置条件 | 增加一条po数据 |
| StaffDataService.getStaff | 语法 | **public** StaffPO getStaff (String id) **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 查找返回相应的结果 |
| StaffDataService.getDriver | 语法 | **public** DriverPO get(String id) **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 查找返回相应的结果 |
| StaffDataService.getTruck | 语法 | **public** TruckPO get(String id) **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 查找返回相应的结果 |
| StaffDataService.delete | 语法 | **public** **boolean** delete(String id)**throws** RemoteException |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 删除po |
| StaffDataService.change | 语法 | **public** **boolean** change(StaffPO po) **throws** RemoteException |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新po |
| StaffDataService. showStaff | 语法 | **public** ArrayList<StaffPO> getAll(String str)**throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有人员信息 |
| StaffDataService. showDriver | 语法 | **public** ArrayList<DriverPO> getAll(String str) **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有人员信息 |
| StaffDataService. showTruck | 语法 | **public** ArrayList<TruckPO> getAll() **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有人员信息 |
| StaffDataService.init | 语法 | **public** **boolean** init() **throws** RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久数据 |

# 5.依赖视角

下图是客户端和服务器端各自的包之间的依赖关系。



图3 快递系统客户端开发包图

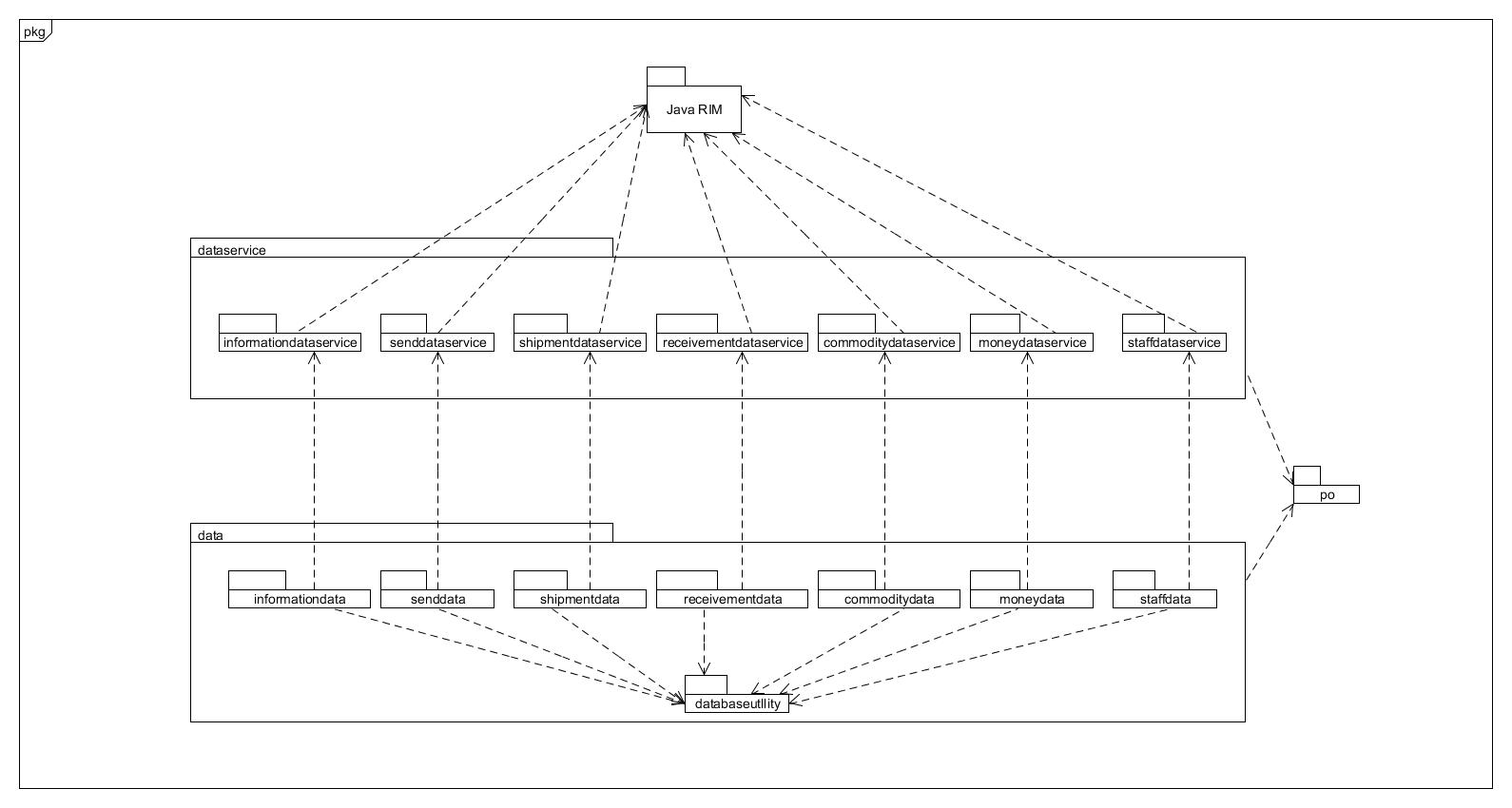


图4 物流系统服务器端开发包图