目录

[更新历史： 3](#_Toc434442796)

[**1.** **引言** 4](#_Toc434442797)

[**1.1编制目的** 4](#_Toc434442798)

[**1.2参考文献** 4](#_Toc434442799)

[**2.** **产品描述** 4](#_Toc434442800)

[3.体系结构设计概述 4](#_Toc434442801)

[4.结构视角 4](#_Toc434442802)

[4.1业务逻辑层分解 4](#_Toc434442803)

[4.1.1Accountbl模块 5](#_Toc434442804)

[4.1.2ConfigurationblController模块 12](#_Toc434442805)

[4.1.3Deliverbl模块 21](#_Toc434442806)

[4.1.4Examinebl模块 26](#_Toc434442807)

[4.1.5Logbl模块 33](#_Toc434442808)

[4.1.6Orderbl模块 34](#_Toc434442809)

[4.1.7Receivebl模块 37](#_Toc434442810)

[4.1.8Searchbl模块 41](#_Toc434442811)

[4.1.9Transportbl模块 44](#_Toc434442812)

[4.2. 数据层的分解 49](#_Toc434442818)

[4.2.1. DataFactoryService 49](#_Toc434442819)

[4.2.2. AccountDataService 50](#_Toc434442820)

[4.2.3. CompanyDataService 51](#_Toc434442821)

[4.2.4. ConfigurationDataService 53](#_Toc434442822)

[4.2.5. DeliverDataService 55](#_Toc434442823)

[4.2.6. FinanceDataService 56](#_Toc434442824)

[4.2.7. InitialDataService 58](#_Toc434442825)

[4.2.8. MemberDataService 59](#_Toc434442826)

[4.2.9. OrderDataService 60](#_Toc434442827)

[4.2.10. ReceiveDataService 61](#_Toc434442828)

[4.2.11. StoreDataService 62](#_Toc434442829)

[4.2.12. TransportDataService 63](#_Toc434442830)

[5.依赖视角 65](#_Toc434442831)

[5.1客户端包图 65](#_Toc434442832)

[5.2服务器包图 65](#_Toc434442833)

# 更新历史：

|  |  |  |
| --- | --- | --- |
| 日期 | 姓名 | 更新 |
| 2015.11.4 | 程翔 | 创建文档 |
|  |  |  |

1. **引言**

**1.1编制目的**

本报告详细完成对快递物流系统的概要设计，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编写，是了解系统的导航。

**1.2参考文献**

1、IEEE标准

2、《2333小组用例文档》、《2333小组需求规格说明文档》

3、《软件体系结构设计目标（IEEE标准）》

4、《项目实践V4》 刘钦

1. **产品描述**

快递物流系统是为本地快递物流企业开发的业务系统，开发目标是帮助该企业处理日常的重点业务，包括收发快递、包裹转运、库存管理和财务统计等。

通过该快递物流系统的使用，期望帮助该快递物流企业保持合理库存，提高用户服务体验，增加业务额，提高财务人员工作效率，并且为经理的决策做支持。

# 3.体系结构设计概述

请参考快递物流系统需求文档对体系结构设计的概述

# 4.结构视角

## 4.1业务逻辑层分解

业务逻辑层的开发包图详见体系结构文档图3。

***说明：此次为初步设计文档，缺少共计4个包***

### 4.1.1Accountbl模块

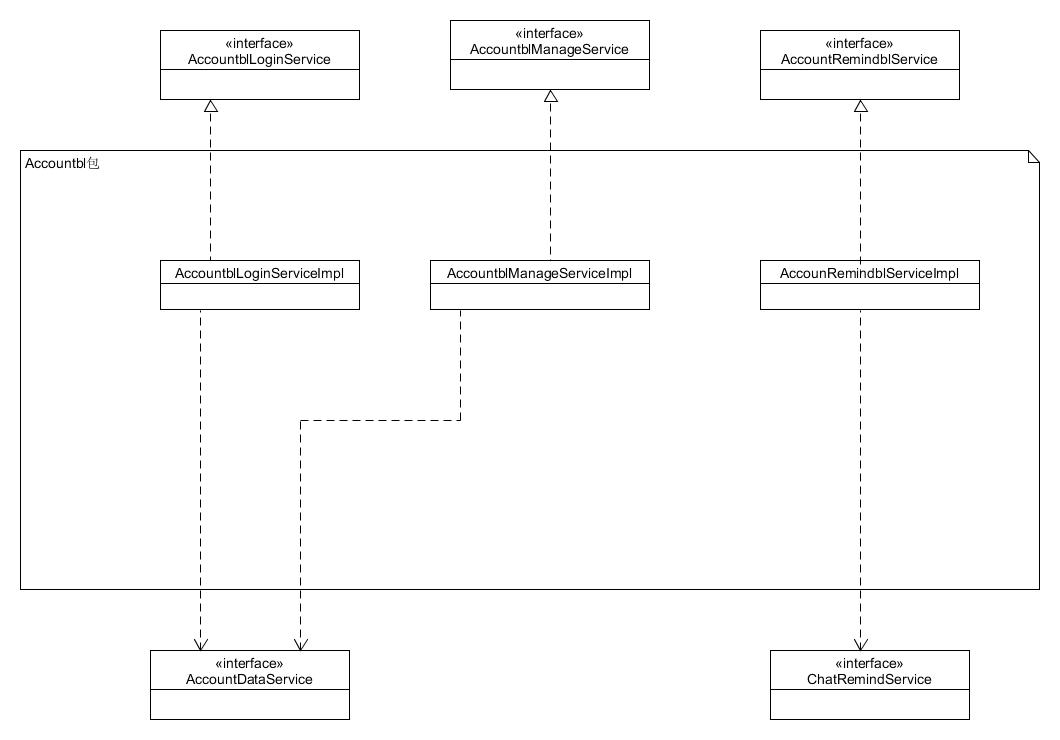
1. 模块概述

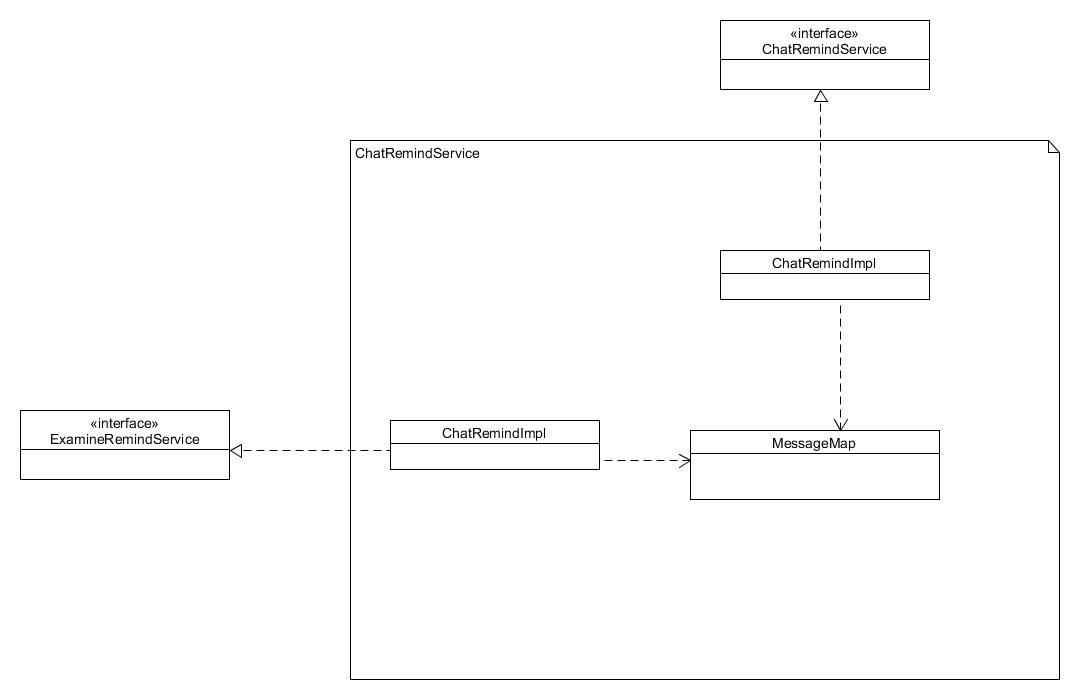
Accountbl模块负责实现人员登录、消息提醒以及管理员guanl 账户的服务

1. 整体结构

Account模块是一个横跨client、server两端的包的模块，一方面，他面向所有人提供login、chatremind服务。同时，它还额外面向管理员提供账户管理的服务

Accountbl模块的设计如图





Accountbl模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| AccountblLoginServiceImpl | 负责实现检验账户登录的服务 |
| AccountblManageServiceImpl | 负责实现管理账户的服务 |
| AccountRemindblServiceImpl | 负责实现账户消息提醒服务 |

ChatRemind模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| ChatRemindImpl | 负责实现检测是否有未读消息的服务 |
| MessageMap | 负责实现存储每个员工对应消息 |
| ExamineRemindImpl | 负责实现新的未读消息产生 |

1. 模块内部类的接口规范

**AccountblLoginServiceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountblLoginServiceImpl.  checkAccount | 语法 | public OperationMessage checkAccount(String id , String password) |
| 前置条件 | 账户的账号、密码 |
| 后置条件 | 返回检测结果 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| AccountDataLoginService.checkAccount(String id, String password) | 检查账号是否正确 | |

**AccountblManageServiceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountblManageServiceImpl.getAccountVOs | 语法 | public ArrayList<AccountVO> getAccountVOs() |
| 前置条件 | 管理员登陆成功 |
| 后置条件 | 显示所有账户名 |
| AccountblManageServiceImpl.getAccountVO | 语法 | public getAccountVO(String accountID) |
| 前置条件 | 管理员输入用户名并且已经通过检查 |
| 后置条件 | 显示该用户名对应的用户信息 |
| AccountblManageServiceImpl.addAccount | 语法 | public OperationMessage addAccount (AccountPO po) |
| 前置条件 | 管理员确认删除该用户信息 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| AccountblManageServiceImpl.deleteAccount | 语法 | public OperationMessage deleteAccount(AccountPO po) |
| 前置条件 | 管理员确认删除该用户信息 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| AccountblManageServiceImpl. modifyAccount | 语法 | public OperationMessage modifyAccount (AccountPO po) |
| 前置条件 | 管理员确认完成修改该用户信息 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| AccountDataService.find(String name) | 查找name对应的AccountPO 并返回 | |
| AccountDataService.  insert(AccountPO po) | 插入单一持久化对象 | |
| AccountDataService.  delete(AccountPO po) | 删除单一持久化对象 | |
| AccountDataService.  update(AccountPO po) | 更新单一持久化对象 | |
| AccountDataService.checkID | 检查ID是否存在 | |
| VOService.getUserVO(String ID) | 返回ID对应的UserVO | |

**AccountRemindblServiceImpl的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| AccountRemindblServiceImpl.  checkMessage | | 语法 | public OperationMessage checkMessage(String ID) |
| 前置条件 | 检查数据接口是否存在未读的系统通知,传入员工的ID |
| 后置条件 | 返回表示是否存在未读的系统通知的OperationMessage |
| AccountRemindblServiceImpl.  receive | | 语法 | public ArrayList<chatMessage> receive(String ID) |
| 前置条件 | 传入员工的ID |
| 后置条件 | 从流中取出与接受方的身份信息相关的信息列表，返回之 |
| 需要的服务（需接口） | | | |
| ChatRemindService.  checkMessage(String ID) | 检查是否有新消息 | | |
| ChatRemindService.  receive(String ID) | 获取新消息 | | |

**ChatRemindImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ChatRemindService.checkMessage | 语法 | public OperationMessage checkMessage(String ID) |
| 前置条件 | 检查数据接口是否存在未读的系统通知,传入员工的ID |
| 后置条件 | 返回表示是否存在未读的系统通知的OperationMessage |
| ChatRemindService.receive | 语法 | public ArrayList<chatMessage> receive(String ID) |
| 前置条件 | 传入员工的ID |
| 后置条件 | 从流中取出与接受方的身份信息相关的信息列表，返回之 |

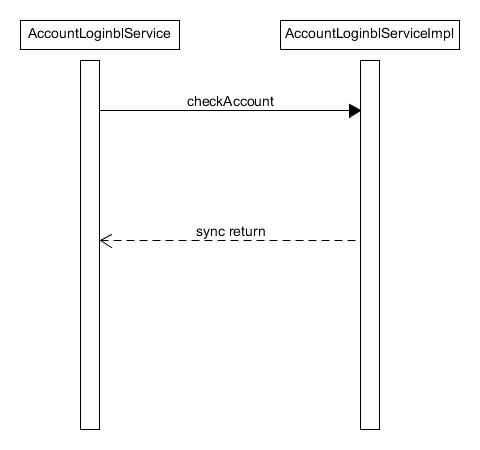
**ExamineRemindImpl的接口规范**

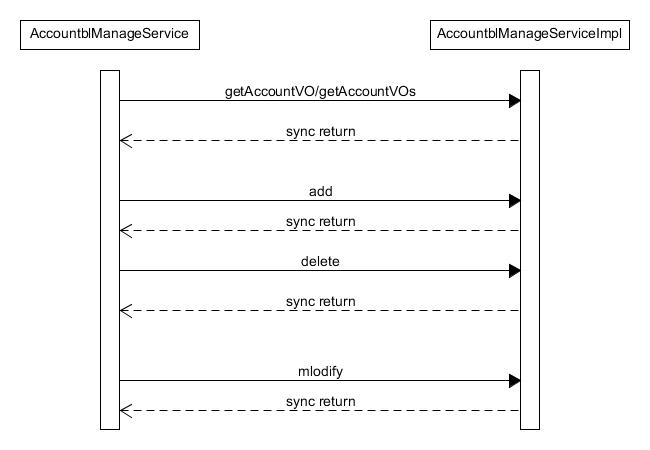
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExamineRemindImpl.transfer | 语法 | public OperationMessage transfer(ArrayList< chatMessage >mes，String ID) |
| 前置条件 | 参数为传输给用户的信息和员工的ID |
| 后置条件 | 返回数据传输是否成功,数据被传输到流中 |

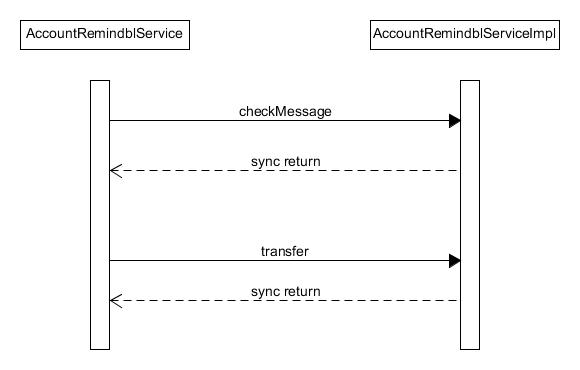
**MessageMap接口规范**

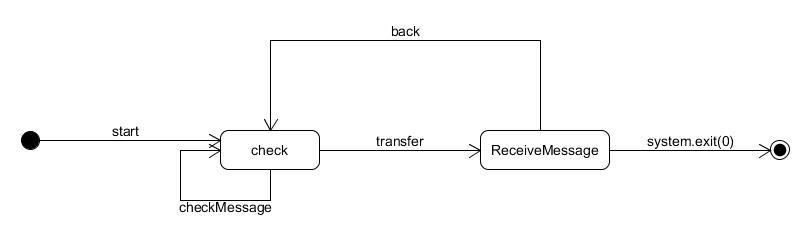
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MessageMap.checkMessage | 语法 | public OperationMessage checkMessage(String ID) |
| 前置条件 | 检查数据接口是否存在未读的系统通知,传入员工的ID |
| 后置条件 | 返回表示是否存在未读的系统通知的OperationMessage |
| MessageMap.transfer | 语法 | public OperationMessage transfer(ArrayList< chatMessage >mes，String ID) |
| 前置条件 | 参数为传输给用户的信息和员工的ID |
| 后置条件 | 返回数据传输是否成功,数据被传输到流中 |
| MessageMap.receive | 语法 | public ArrayList<chatMessage> receive(String ID) |
| 前置条件 | 传入员工的ID |
| 后置条件 | 从流中取出与接受方的身份信息相关的信息列表，返回之 |

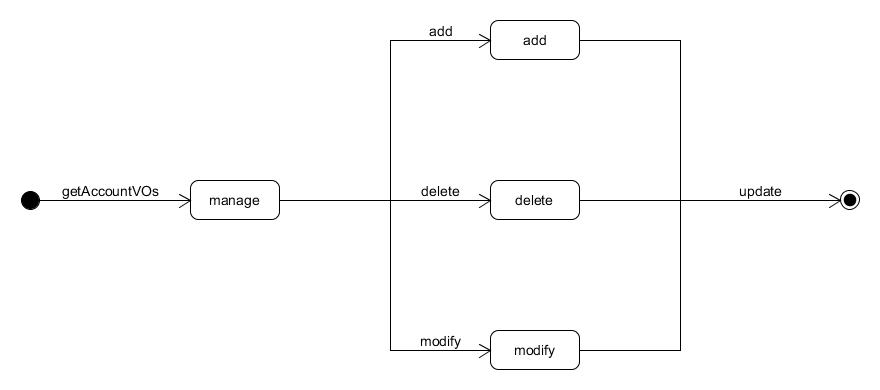
1. 业务逻辑层的动态模型











(5)控制逻辑

采用集中式控制逻辑，各个子模块间相对独立

### 4.1.2ConfigurationblController模块

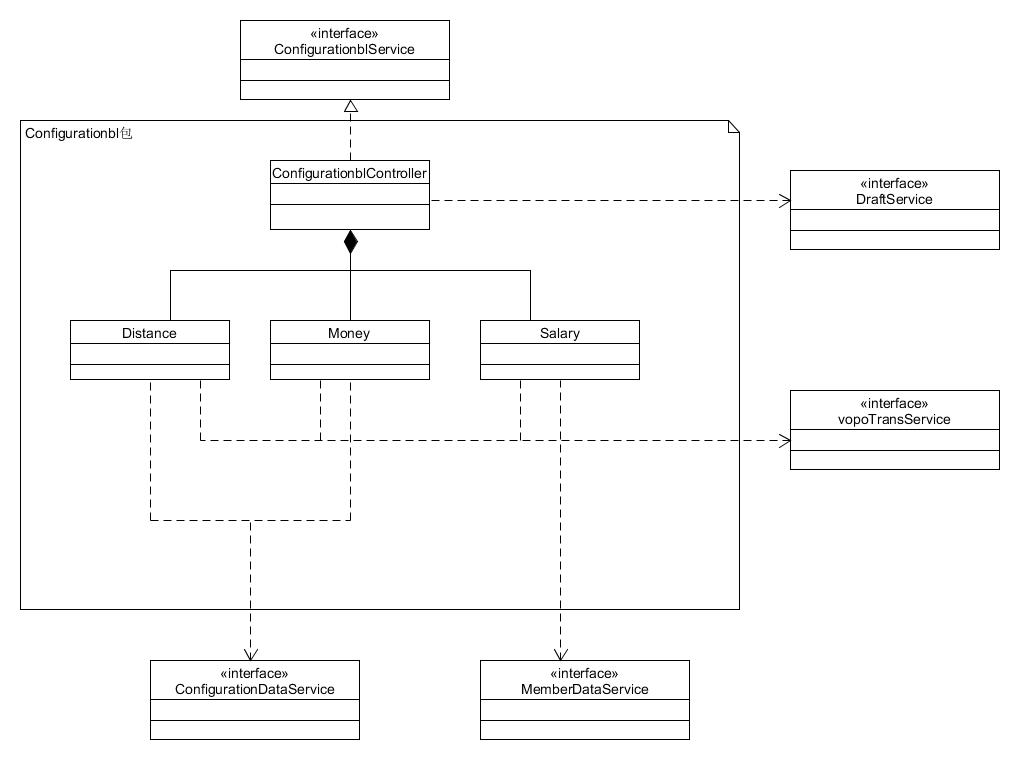
(1)模块概述

ConfigurationblController模块负责实现系统常量的增删改查工作

（2）整体结构

将包分为了salary、distance、money三个模块，其中money中包含pack、price、proportion

ConfigurationblController模块的设计如图



ConfigurationblController模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| ConfigurationblControllerController | 负责实现控制的服务 |
| Distance | 负责实现城市距离的增删改查 |
| Money | 负责实现收费的增删改查 |
| Salary | 负责实现薪水策略的增删改查 |

（3）模块内部类的接口规范

**ConfigurationblControllerController接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ConfigurationblControllerController.getCityDistance | 语法 | public ArrayList<CityDistanceVO> getCityDistance() |
| 前置条件 | 无 |
| 后置条件 | 返回城市间距离列表 |
| ConfigurationblControllerController.modifyCityDistance | 语法 | public OperationMessage modifyCityDistance(CityDistanceVO after) |
| 前置条件 | after为新修改的城市距离数据；已经获取所有城市距离的列表 |
| 后置条件 | 若修改成功则返回成功；否则返回失败原因，系统提示修改失败原因 |
| ConfigurationblController.getSalaryStrategy | 语法 | public ArrayList<SalaryStrategyVO> getSalaryStrategy() |
| 前置条件 | 无 |
| 后置条件 | 获取员工薪水策略列表 |
| ConfigurationblController.modifySalaryStrategy | 语法 | public OperationMessage modifySalaryStrategy(ArrayList<SalaryStrategyVO> after) |
| 前置条件 | after为所有修改后的数据列表 |
| 后置条件 | 若修改成功返回成功；否则返回失败原因 |
| ConfigurationblController.getPack | 语法 | public PackVO getPack() |
| 前置条件 | 无 |
| 后置条件 | 返回包装费用的信息 |
| ConfigurationblController.modifyPack | 语法 | public OperationMessage modifyPack(PackVO after) |
| 前置条件 | after为修改后的包装价格；已经获取所有的包装费用信息 |
| 后置条件 | 若修改成功，返回成功；否则，返回失败原因 |
| ConfigurationblController.getPrice | 语法 | public PriceVO getPrice() |
| 前置条件 | 无 |
| 后置条件 | 返回当前普通快递的价格 |
| ConfigurationblController.modifyPrice | 语法 | public OperationMessage modifyPrice(PriceVO after) |
| 前置条件 | after为修改后的运费价格；已经获取运费价格信息 |
| 后置条件 | 若修改成功，返回成功；否则，返回失败原因 |
| ConfigurationblController.getProportion | 语法 | public ProportionPO getProportion() |
| 前置条件 | 无 |
| 后置条件 | 获取不同快递收费比例 |
| ConfigurationblController.modifyProportion | 语法 | public OperationMessage modifyProportion(ProportionVO after) |
| 前置条件 | after为修改后的比例信息；已经获取价格比例 |
| 后置条件 | 若修改成功，返回成功；否则返回失败原因 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Distance.  getCityDistance() | 获取城市距离信息 | |
| Distance.  modifyCityDistance(CityDistancePO after) | 修改城市距离信息 | |
| Salary.  getSalaryStrategy() | 获取员工薪水策略 | |
| Salary.  modifySalaryStrategy(SalaryPOafter) | 修改员工薪水策略 | |
| Money.  getPack() | 获取包装费用 | |
| Money.  modifyPack(PackPO after) | 修改包装费用 | |
| Money.  getPrice() | 获取快递收费价格 | |
| Money.  modifyPrice(PricePO after) | 修改快递收费价格，并重新计算其它价格 | |
| Money.  getProportion() | 获取不同快递收费比例 | |
| Money.  modifyProportion(ProportionPO after) | 修改不同快递收费比例，并重新计算其它价格 | |

**Distance接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Distance.getCityDistance | 语法 | public ArrayList<CityDistanceVO> getCityDistance() |
| 前置条件 | 无 |
| 后置条件 | 返回城市间距离列表 |
| Distance.modifyCityDistance | 语法 | public OperationMessage modifyCityDistance(CityDistanceVO after) |
| 前置条件 | after为新修改的城市距离数据；已经获取所有城市距离的列表 |
| 后置条件 | 若修改成功则返回成功；否则返回失败原因，系统提示修改失败原因 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| configurationDataService.  getCityDistance() | 获取城市距离信息 | |
| configurationDataService.  modifyCityDistance(CityDistancePO after) | 修改城市距离信息 | |

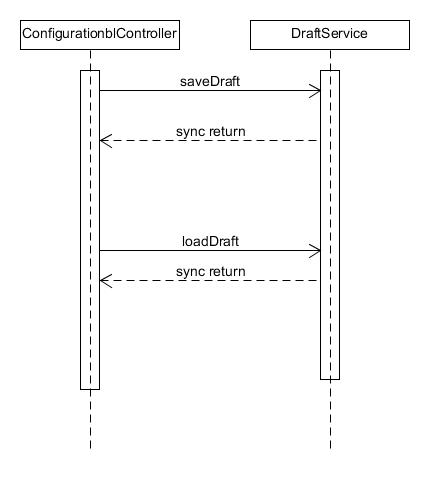
**Money接口规范**

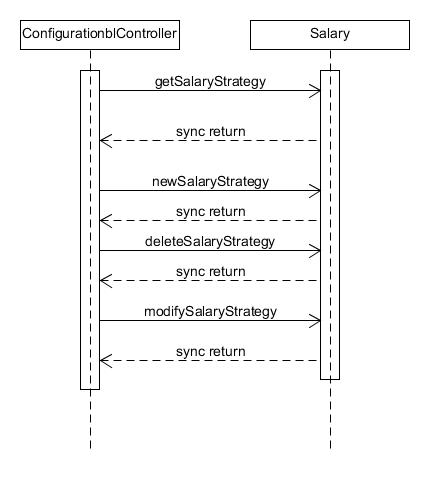
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Money.getPack | 语法 | public PackVO getPack() |
| 前置条件 | 无 |
| 后置条件 | 返回包装费用的信息 |
| Money.modifyPack | 语法 | public OperationMessage modifyPack(PackVO after) |
| 前置条件 | after为修改后的包装价格；已经获取所有的包装费用信息 |
| 后置条件 | 若修改成功，返回成功；否则，返回失败原因 |
| Money.getPrice | 语法 | public PriceVO getPrice() |
| 前置条件 | 无 |
| 后置条件 | 返回当前普通快递的价格 |
| Money.modifyPrice | 语法 | public OperationMessage modifyPrice(PriceVO after) |
| 前置条件 | after为修改后的运费价格；已经获取运费价格信息 |
| 后置条件 | 若修改成功，返回成功；否则，返回失败原因 |
| Money.getProportion | 语法 | public ProportionPO getProportion() |
| 前置条件 | 无 |
| 后置条件 | 获取不同快递收费比例 |
| Money.modifyProportion | 语法 | public OperationMessage modifyProportion(ProportionVO after) |
| 前置条件 | after为修改后的比例信息；已经获取价格比例 |
| 后置条件 | 若修改成功，返回成功；否则返回失败原因 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| configurationDataService.  getPack() | 获取包装费用 | |
| configurationDataService.  modifyPack(PackPO after) | 修改包装费用 | |
| configurationDataService.  getPrice() | 获取快递收费价格 | |
| configurationDataService.  modifyPrice(PricePO after) | 修改快递收费价格，并重新计算其它价格 | |
| configurationDataService.  getProportion() | 获取不同快递收费比例 | |
| configurationDataService.  modifyProportion(ProportionPO after) | 修改不同快递收费比例，并重新计算其它价格 | |

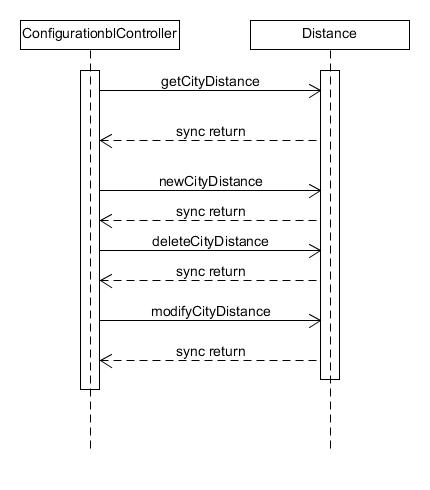
**Salary接口规范**

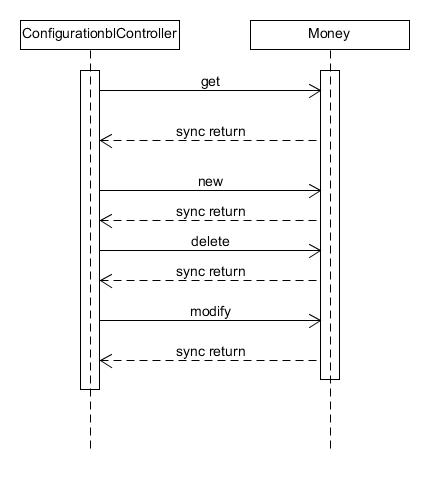
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Salary.getSalaryStrategy | 语法 | public ArrayList<SalaryStrategyVO> getSalaryStrategy() |
| 前置条件 | 无 |
| 后置条件 | 获取员工薪水策略列表 |
| Salary.modifySalaryStrategy | 语法 | public OperationMessage modifySalaryStrategy(ArrayList<SalaryStrategyVO> after) |
| 前置条件 | after为所有修改后的数据列表 |
| 后置条件 | 若修改成功返回成功；否则返回失败原因 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| configurationDataService.  getSalaryStrategy() | 获取员工薪水策略 | |
| configurationDataService.  modifySalaryStrategy(SalaryPOafter) | 修改员工薪水策略 | |

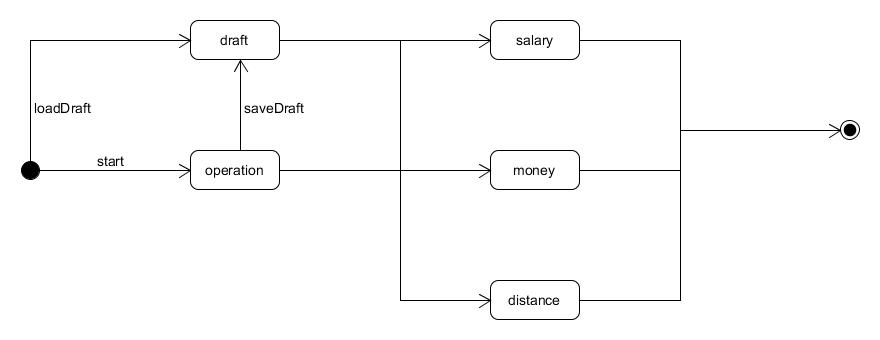
（4）业务逻辑层的动态模型











1. 业务逻辑层的设计原理

采用集中式的控制

### 4.1.3Deliverbl模块

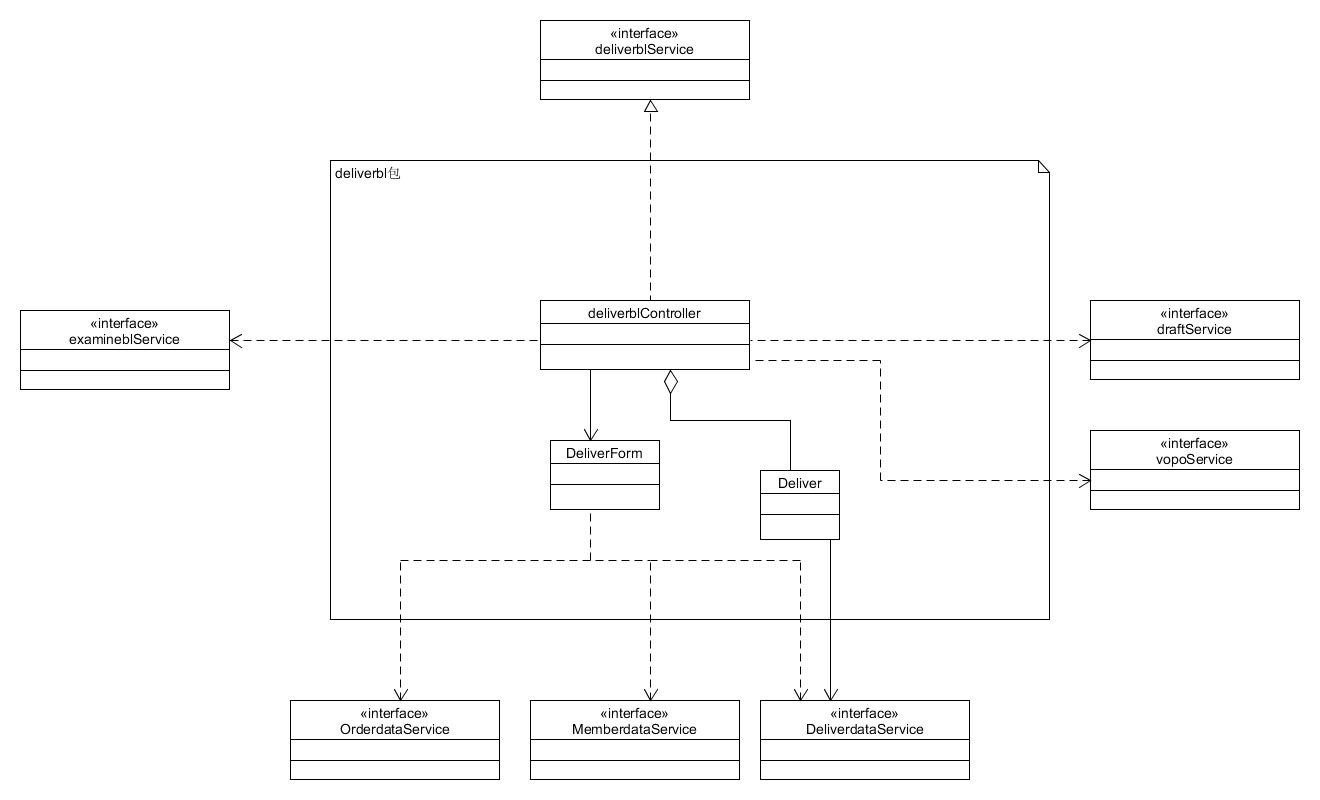
1. 模块概述

Deliverbl模块负责实现营业厅派件单的新建、提交以及查看工作。

1. 整体结构

一方面，包本身的逻辑比较简单；同时，它所需要实现的功能与其他新建订单包类似，因此采用了统一的工具实现。因此包内部的组织结构就相对较为简单

Deliverbl模块的设计如图



Deliverbl模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| DeliverblController | 负责实现控制的服务 |
| DeliverForm | 负责实现新建派件单 |
| Deliver | 负责实现查看派件单 |

1. 模块内部类的接口规范

**DeliverblController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DeliverblController.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的派件单编号 |
| DeliverblController.getOrderByHall | 语法 | public ArrayList<OrderVO> getOrderByHall(String hallID) |
| 前置条件 | 传入营业厅编号 |
| 后置条件 | 返回营业厅尚未派件的订单 |
| DeliverblController.getAvaliableDeliver | 语法 | public ArrayList<DeliverVO> getAvaliableDeliver(String hallID) |
| 前置条件 | 营业厅编号 |
| 后置条件 | 当前可派遣的快递员 |
| DeliverblController.submit | 语法 | public OperationMessage submit(DeliverVO form) |
| 前置条件 | DeliverVO已经通过检查 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| DeliverblController.saveDraft | 语法 | public OperationMessage saveDraft(DeliverVO draft) |
| 前置条件 | 输入的DeliverVO不为空 |
| 后置条件 | 将DeliverVO序列化成文件存储 |
| DeliverblController.loadDraft | 语法 | public DeliverVO loadDraft() |
| 前置条件 | 有对应的序列化文件存在 |
| 后置条件 | 将序列化文件解序列化成对应的DeliverVO对象返回 |
| DeliverblController.getDeliverFormByID(String ID) | 语法 | public ArrayList<DeliverVO> getDeliverFormByID( String ID) |
| 前置条件 | 传入员工ID |
| 后置条件 | 返回其对应的派件单 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ExamineService.  submit(FormVO VO) | 提交表格审批 | |
| DeliverForm.newID | 得到新的派件单号 | |
| DeliverForm.getOrderByHall | 获得订单信息 | |
| DeliverForm.getAvaliableDeliver | 获得可用人员信息 | |
| DeliverForm.getDeliverVO | 获得派件单信息 | |
| Deliver.getDeliverFormByID(String ID) | 根据员工ID获得派件单 | |

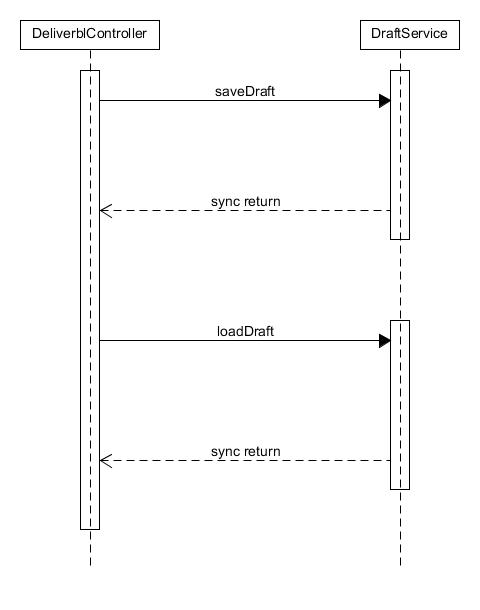
**DeliverForm接口规范**

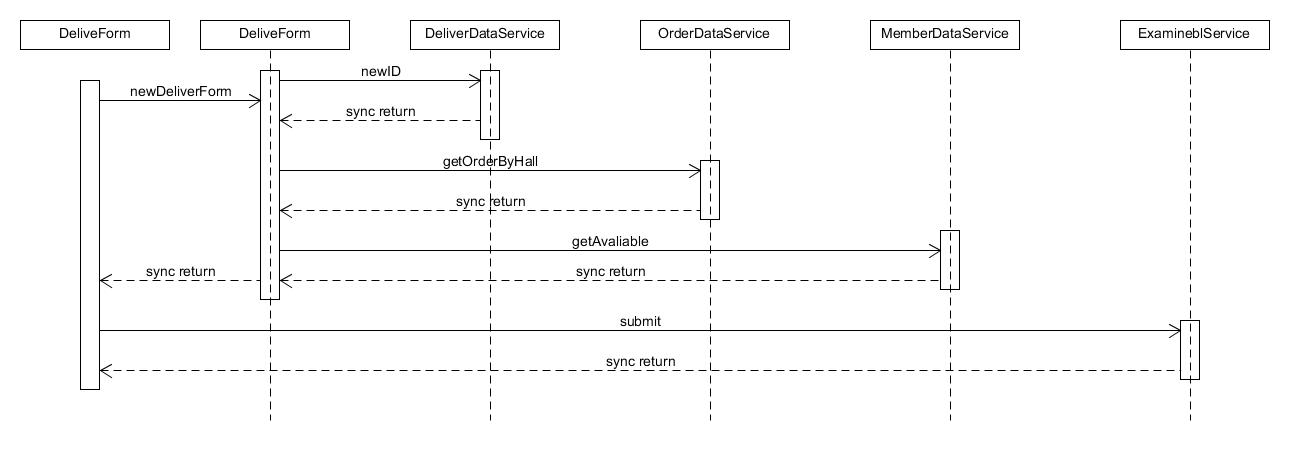
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DeliverForm.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的派件单编号 |
| DeliverForm.getOrderByHall | 语法 | public ArrayList<OrderVO> getOrderByHall(String hallID) |
| 前置条件 | 传入营业厅编号 |
| 后置条件 | 返回营业厅尚未派件的订单 |
| DeliverForm.getAvaliableDeliver | 语法 | public ArrayList<DeliverVO> getAvaliableDeliver(String hallID) |
| 前置条件 | 营业厅编号 |
| 后置条件 | 当前可派遣的快递员 |
| DeliverForm.submit | 语法 | public OperationMessage submit(DeliverVO form) |
| 前置条件 | DeliverVO已经通过检查 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| DeliverForm.saveDraft | 语法 | public OperationMessage saveDraft(DeliverVO draft) |
| 前置条件 | 输入的DeliverVO不为空 |
| 后置条件 | 将DeliverVO序列化成文件存储 |
| DeliverForm.loadDraft | 语法 | public DeliverVO loadDraft() |
| 前置条件 | 有对应的序列化文件存在 |
| 后置条件 | 将序列化文件解序列化成对应的DeliverVO对象返回 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ExamineService.  submit(FormVO VO) | 提交表格审批 | |
| DeliverDataService.newID | 得到新的派件单号 | |
| OrderDataService.getOrderByHall | 获得订单信息 | |
| MemberDataService.getAvaliableDeliver | 获得可用人员信息 | |
| DeliverDataService.getDeliverVO | 获得派件单信息 | |

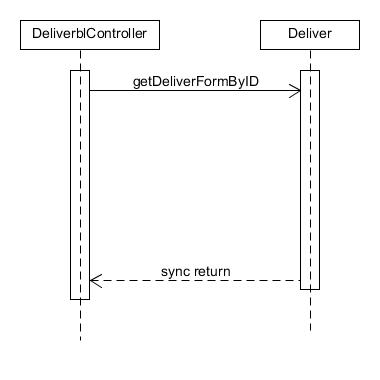
**Deliver接口规范**

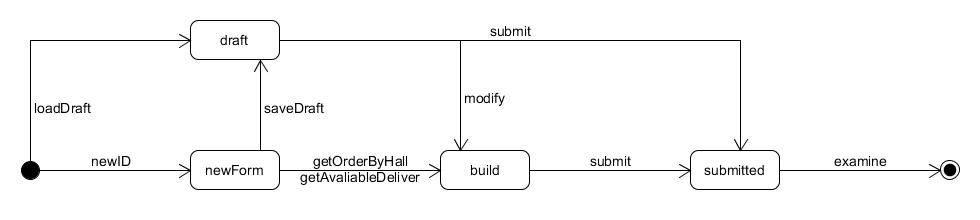
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Deliver.getDeliverFormByID | 语法 | public ArrayList<DeliverVO> getDeliverFormByID(String ID) |
| 前置条件 | 员工ID |
| 后置条件 | 返回该员工对应的派件单 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DeliverDataService.  getDeliverFormByID | 根据员工ID返回对应的派件单 | |

1. 业务逻辑层的动态模型









1. 业务逻辑层的设计原理

采用集中式的控制

### 4.1.4Examinebl模块

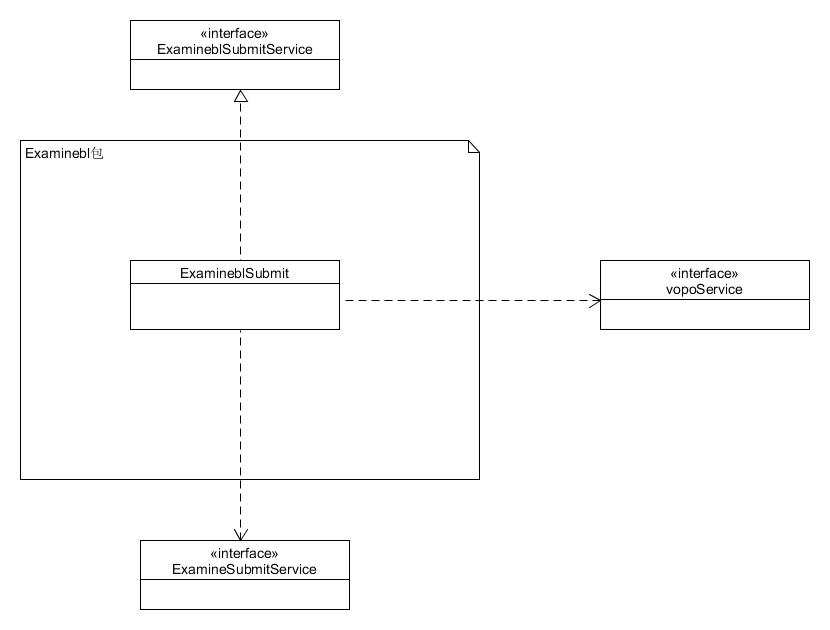
1. 模块概述

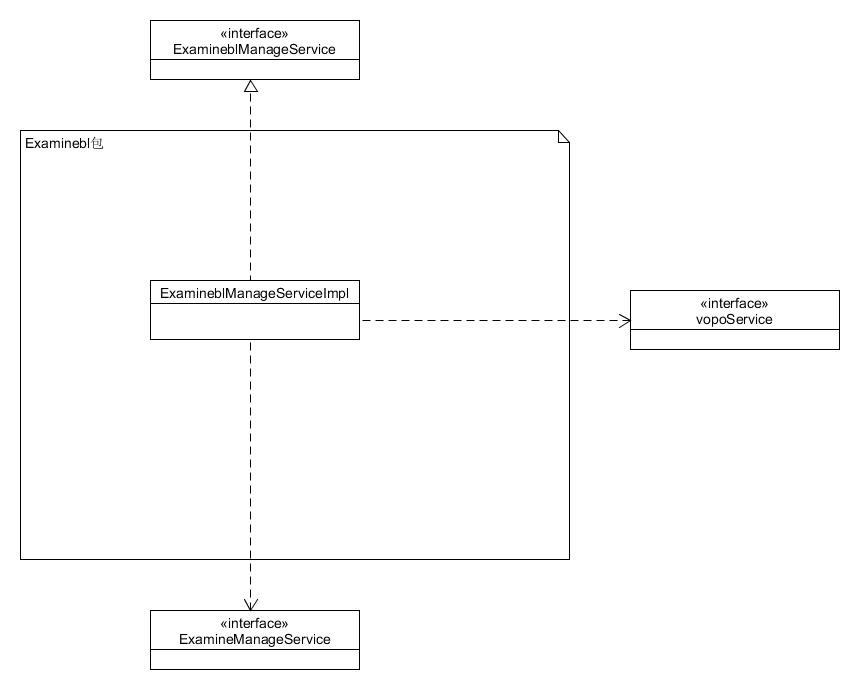
Examinebl模块负责实现单据的提交和审批

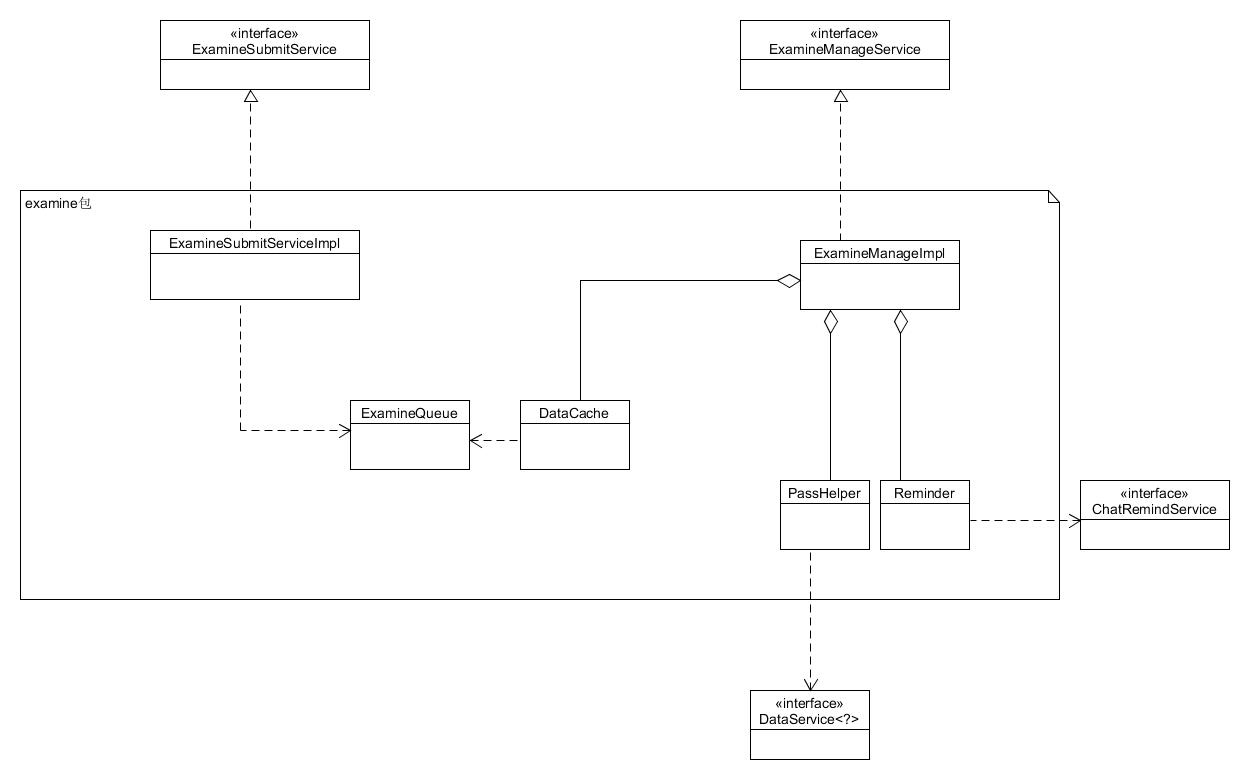
1. 整体结构

Examine模块是一个横跨client、server两端的包的模块，一方面，他面向除总经理外几乎所有人提供submit服务，而总经理通过它来审批、修改单据。

Examinebl模块的设计如图







Examinebl模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| ExamineSubmitServiceImpl | 负责实现审核流程提交的服务 |
| ExamineManageServiceImpl | 负责实现审核流程审批的服务 |
| ExamineQueue | 负责审批队列的实现 |
| DataCache | 负责审批信息的缓存 |
| PassHelper | 负责实现审核通过后的服务 |
| Reminder | 负责实现系统通知服务 |

1. 模块内部类的接口规范

**ExamineSubmitService的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| examineSubmitService.submit | 语法 | public OperationMessage submit(FormVO form) |
| 前置条件 | form为提交状态的单据 |
| 后置条件 | resultmessage为结果信息 |

**ExamineManageService的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| examineManageService.getForms | 语法 | public ArrayList<FormVO> getForms() |
| 前置条件 | 无 |
| 后置条件 | 返回提交状态的表单 |
| examineManageService.passForm | 语法 | public OperationMessage passForm(ArrayList<FormVO> form) |
| 前置条件 | form为通过审批的单据列表 |
| 后置条件 | 若审批成功则返回true；否则返回false |
| examineManageService.deleteForm | 语法 | public OperationMessage deleteForm(ArrayList<FormVO> form) |
| 前置条件 | form为要删除的单据列表 |
| 后置条件 | 若删除成功则返回true；否则返回原因 |
| examineManageService.getForm | 语法 | public FormVO getForm(FormVO form) |
| 前置条件 | 传入查询单据的信息 |
| 后置条件 | 返回查询结果 |
| examineManageService.getFormHistory | 语法 | public ArrayList<FormVO> getFormHistory() |
| 前置条件 | 无 |
| 后置条件 | 获取所有审批通过的单据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DataCache.deCache | 从Cache获得信息 | |
| PassHelper.pass | 表单通过的服务 | |

**ExamineQueue的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExamineQueue.enqueue | 语法 | Public OperationMessage enqueue(ArrayList < FormVO > forms) |
| 前置条件 | 传入表单VO集合 |
| 后置条件 | 返回操作结果 |
| ExamineQueue.dequeue | 语法 | Public ArrayList < FormVO > dequeue(String Type) |
| 前置条件 | 传入表单类型 |
| 后置条件 | 返回相应类型的表单VO集合 |
| ExamineQueue.empty | 语法 | Public boolean empty() |
| 前置条件 | 无 |
| 后置条件 | 返回队列现在是否为空 |

**DataCache的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DataCache.getFromQueue | 语法 | public OperationMessage getFromQueue(String Type) |
| 前置条件 | 传入表单类型 |
| 后置条件 | 返回相应类型的表单VO集合作为缓存 |
| DataCache.deCache | 语法 | public ArrayList<FormVO> deCache(String Type) |
| 前置条件 | 传入表单类型 |
| 后置条件 | 返回相应类型的表单VO集合 |
| DataCache.clear | 语法 | public OperationMessage clear() |
| 前置条件 | 无 |
| 后置条件 | 返回操作结果，Cache被清空 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ExamineQueue.dequeue | 出队列功能 | |

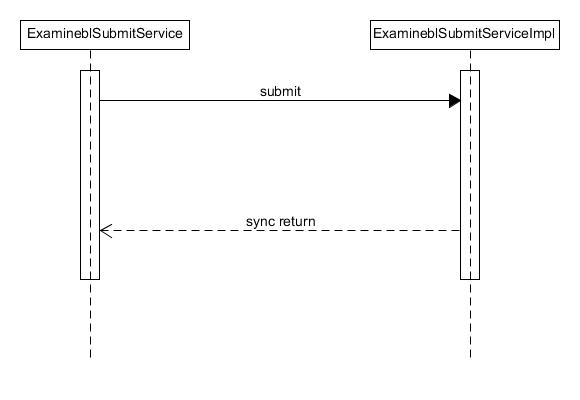
**PassHelper的接口规范**

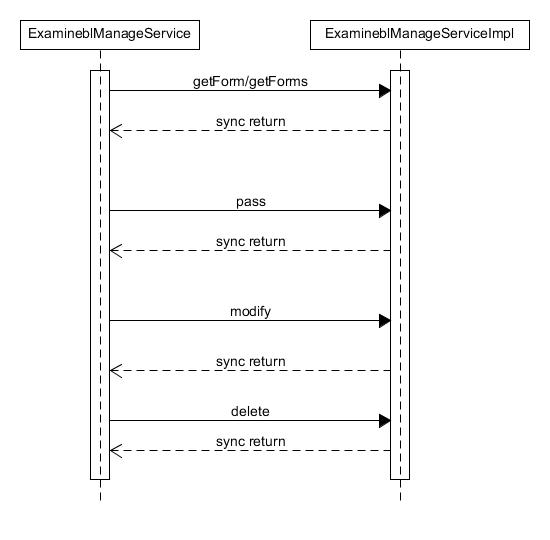
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PassHelper.pass | 语法 | public OperationMessage pass(ArrayList<FormVO>，String Type) |
| 前置条件 | 传入表单集合,表单类型 |
| 后置条件 | 返回操作结果，新通过的单据被添加 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| transportDataService.  insert(FormPO form) | 添加新通过的单据 | |
| receiveDataService.  insert(FormPO form) | 添加新通过的单据 | |
| orderDataService.  insert(FormPO form) | 添加新通过的单据 | |
| deliverDataService.  insert(FormPO form) | 添加新通过的单据 | |
| financeDataService.  insert(FormPO form) | 添加新通过的单据 | |
| storeDataService.  insert(FormPO form) | 添加新通过的单据 | |

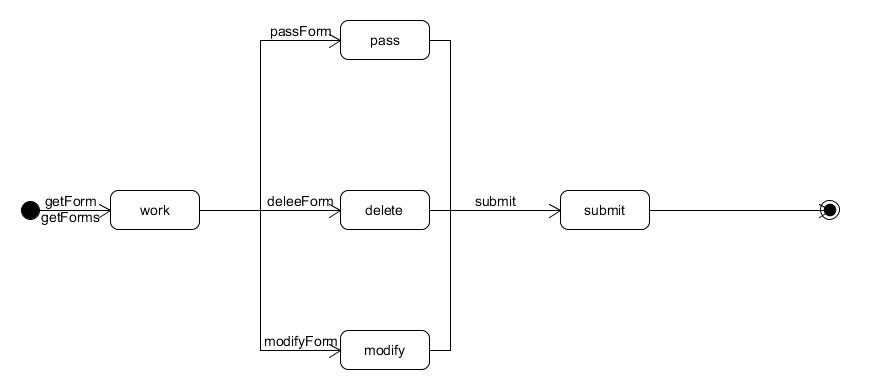
**Reminder的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Reminder.remind | 语法 | public OperationMessage remind(ArrayList< chatMessage >mes，String ID) |
| 前置条件 | 参数为传输给用户的信息和员工的ID |
| 后置条件 | 返回操作结果 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| chatRemindService.  transfer | 提醒用户 | |

1. 业务逻辑层的动态模型







### 4.1.5Logbl模块

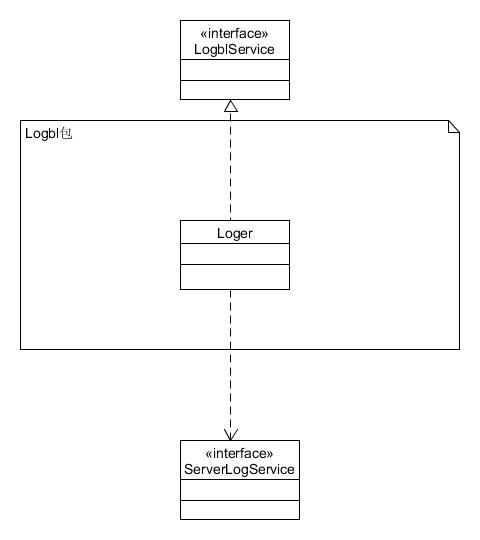
1. 模块概述

Logbl模块负责实现获取、导出系统日志的工作

1. 整体结构

包的逻辑很少，几乎不需要设计

Logbl模块的设计如图



Logbl模块各个类的职责

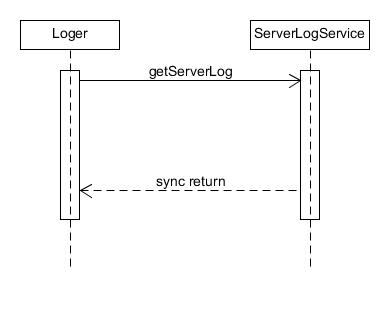
|  |  |
| --- | --- |
| 模块 | 职责 |
| Loger | 负责实现获取日志 |

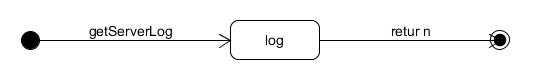
1. 模块内部类的接口规范

**Loger的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Loger.getServerLog | 语法 | Public ArrayList<LogOperation> getServerLog() |
| 前置条件 | 无 |
| 后置条件 | 导出系统服务器日志 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Service.  getServerLog | 获取服务器日志 | |

1. 业务逻辑层的动态模型





1. 业务逻辑层的设计原理

这个包的逻辑太简单，不需要控制

### 4.1.6Orderbl模块

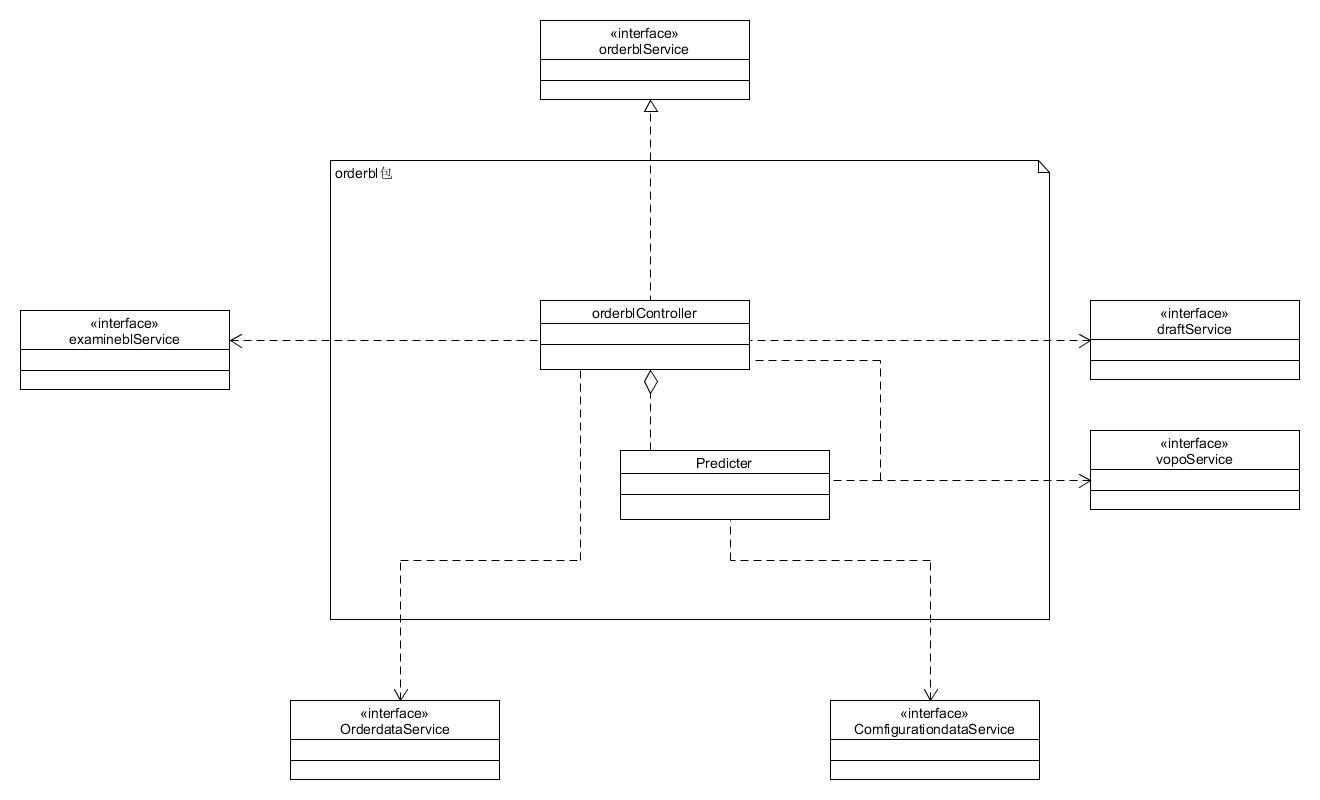
1. 模块概述

Orderbl模块负责实现订单新建、提交工作。

1. 整体结构

一方面，包本身的逻辑比较简单；同时，它所需要实现的功能与其他新建订单包类似，因此采用了统一的工具实现。因此包内部的组织结构就相对较为简单

Orderbl模块的设计如图



Orderbl模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| OrderblController | 负责实现中转/装车单的服务 |
| Predicter | 负责计算运费和到达时间 |

1. 模块内部类的接口规范

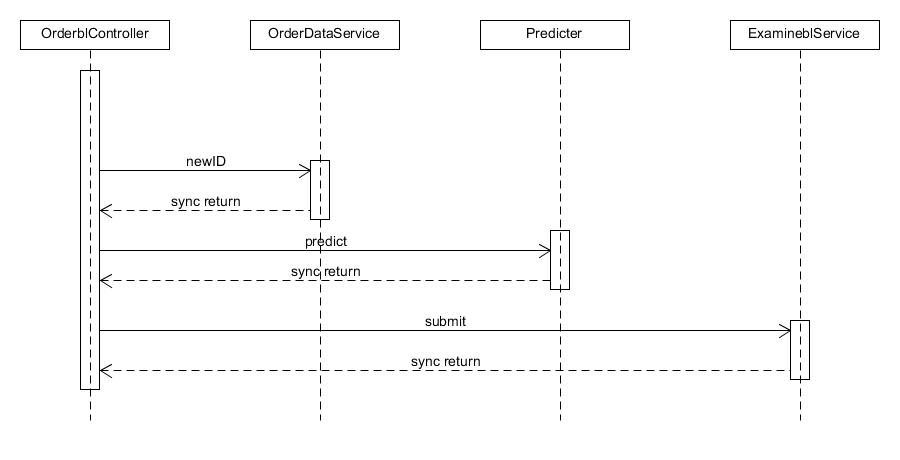
**OrderBL的接口规范**

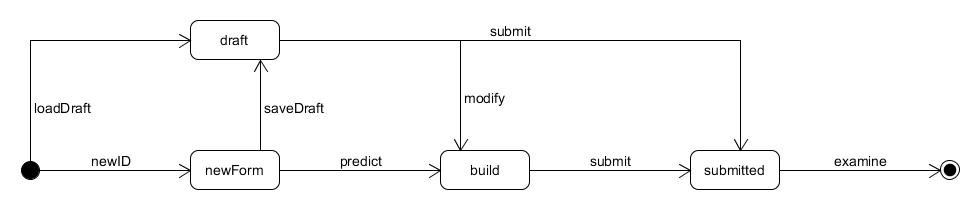
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderBL.checkFormat | 语法 | Public List < CheckFormMessage > checkFormat(String OrderID) |
| 前置条件 | 输入的OrderID不为空 |
| 后置条件 | 检查输入信息是否符合系统要求，若符合，返回的list中只有一个显示正确的CheckFormMessage，否则返回能指出错误的CheckFormMessage的列表 |
| OrderBL.submit | 语法 | Public OperationMessage submit(OrderPO form) |
| 前置条件 | OrderPO已通过检查 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| OrderBL.saveDraft | 语法 | public OperationMessage saveDraft(OrderPO draft) |
| 前置条件 | 输入的OrderPO不为空 |
| 后置条件 | 将OrderPO序列化成文件存储 |
| OrderBL.loadDraft | 语法 | public OrderPO loadDraft() |
| 前置条件 | 有对应的序列化文件存在 |
| 后置条件 | 将序列化文件解序列化成对应的OrderPO对象返回 |
| OrderBL.predict | 语法 | Public PredictPO predict(OrderPO po) |
| 前置条件 | OrderPO已通过检查 |
| 后置条件 | 返回经计算得出的运费和预计到达时间 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ExamineService.  submit(FormPO po) | 提交表格审批 | |
| Predicter.predict(OrderPO) | 进行预计和计算 | |

**Predicter的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderBL.predict | 语法 | Public PredictPO predict(OrderPO po) |
| 前置条件 | 输入的OrderPO不为空 |
| 后置条件 | 返回运费和预计到达时间 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ConfigurationDataService.getPrice | 获得单位运费 | |
| ConfigurationDataService.get | 获得距离 | |

1. 业务逻辑层的动态模型





1. 业务逻辑层的设计原理

采用集中式的控制

### 4.1.7Receivebl模块

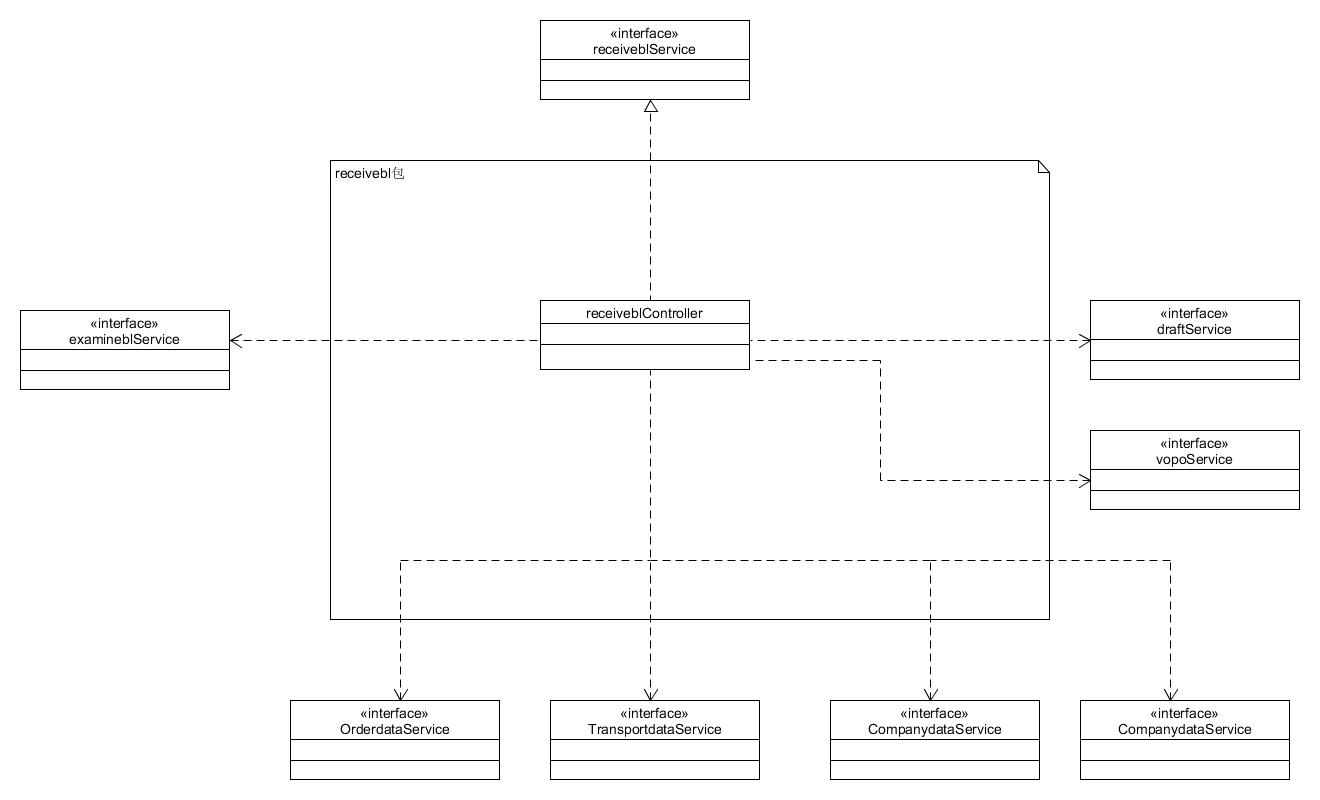
1. 模块概述

Receivebl模块负责实现营业厅和中转中心到达单的新建、提交工作。

1. 整体结构

事实上，个人感觉这个包被架空了。一方面，包本身的逻辑比较简单；同时，它所需要实现的功能与其他新建订单包类似，因此采用了统一的工具实现。因此包内部的组织结构就相对较为简单

Receivebl模块的设计如图



Receivebl模块各个类的职责

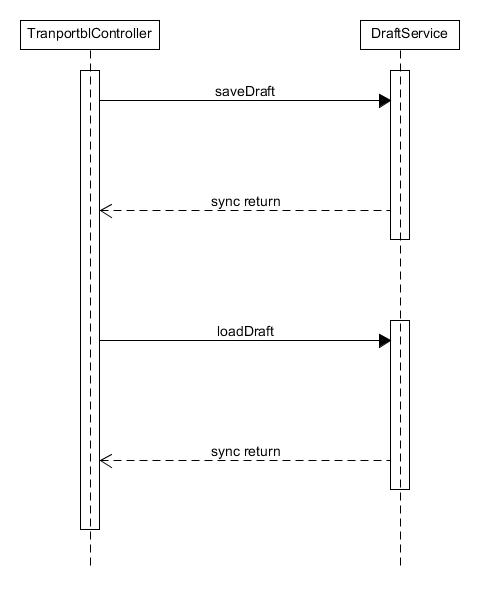
|  |  |
| --- | --- |
| 模块 | 职责 |
| ReceiveblController | 负责实现到达单的服务 |

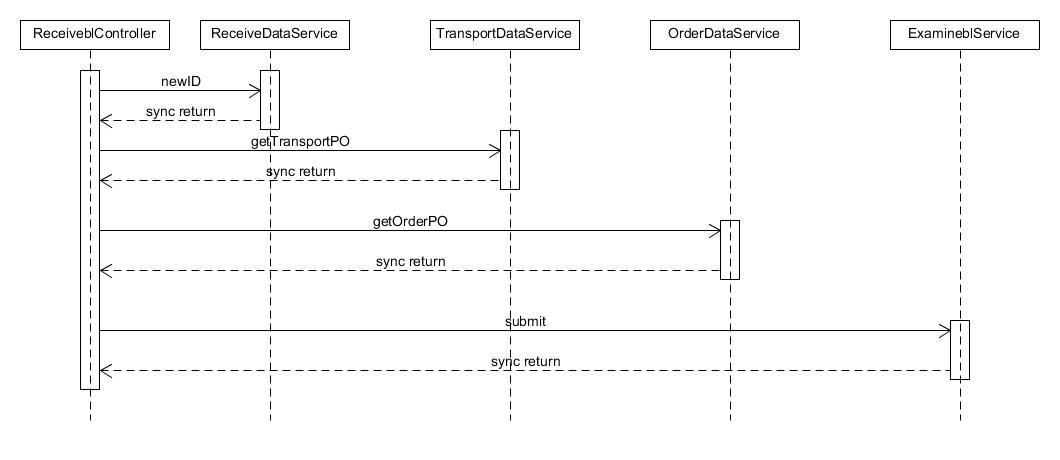
1. 模块内部类的接口规范

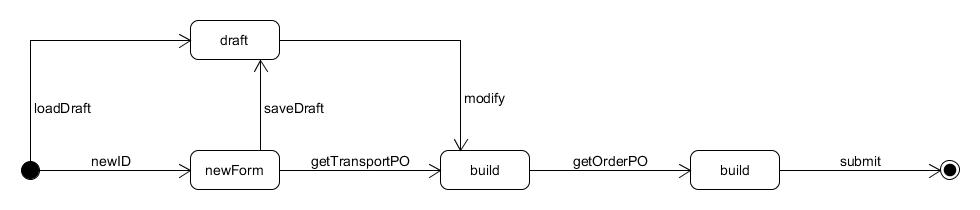
**ReceiveBL的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReceiveblController. checkFormat | 语法 | public List< CheckFormMessage > checkFormat(LoadPO form, boolean isFinal) |
| 前置条件 | 输入的LoadPO不为空,isFinal判断是否为提交表格的操作 |
| 后置条件 | 检查输入信息是否符合系统要求，若符合，返回的list中只有一个显示正确的CheckFormMessage, 否则返回能指示出错误的CheckFormMessage的列表 |
| ReceiveblController.getTransportPO | 语法 | public TransportPO getTransportPO(String transportID) |
| 前置条件 | 传入中转/装车单号 |
| 后置条件 | 返回对应的中转/装车单信息 |
| ReceiveblController.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的装车单编号 |
| ReceiveblController.submit | 语法 | public OperationMessage submit(LoadPO form) |
| 前置条件 | LoadPO已经通过检查 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| ReceiveblController.saveDraft | 语法 | public OperationMessage saveDraft(LoadPO draft) |
| 前置条件 | 输入的LoadPO不为空 |
| 后置条件 | 将LoadPO序列化成文件存储 |
| ReceiveblController.loadDraft | 语法 | public LoadPO loadDraft() |
| 前置条件 | 有对应的序列化文件存在 |
| 后置条件 | 将序列化文件解序列化成对应的LoadPO对象返回 |
| ReceiveblController. checkFormat | 语法 | public List<OperationMessage> checkFormat(CenterPO form) |
| 前置条件 | 输入的CenterPO不为空 |
| 后置条件 | 检查输入信息是否符合系统要求，若符合，返回的list中只有一个显示正确的OperationMessage, 否则返回能指示出错误的OperationMessage的列表 |
| ReceiveblService.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回合法的新中转单编号 |
| ReceiveblController.submit | 语法 | public OperationMessage submit(CenterPO form) |
| 前置条件 | CenterPO已经通过检查 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| ReceiveblController.saveDraft | 语法 | public OperationMessage saveDraft(CenterPO draft) |
| 前置条件 | 输入的CenterPO不为空 |
| 后置条件 | 将CenterPO序列化成文件存储 |
| ReceiveblController.loadDraft | 语法 | public CenterPO loadDraft() |
| 前置条件 | 有对应的序列化文件存在 |
| 后置条件 | 将序列化文件解序列化成对应的CenterPO对象返回 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ExamineService.  submit(FormPO po) | 提交表格审批 | |
| ReceiveDataService.newID | 得到新的到达单号 | |
| OrderDataService.getOrderPO | 获得订单信息 | |
| CompanyDataService.freeCar | 改变车辆车辆状态 | |
| TransportDataService.getTranspotPO | 获取对应的中转/装车单信息 | |

1. 业务逻辑层的动态模型







1. 业务逻辑层的设计原理

采用集中式的控制

### 4.1.8Searchbl模块

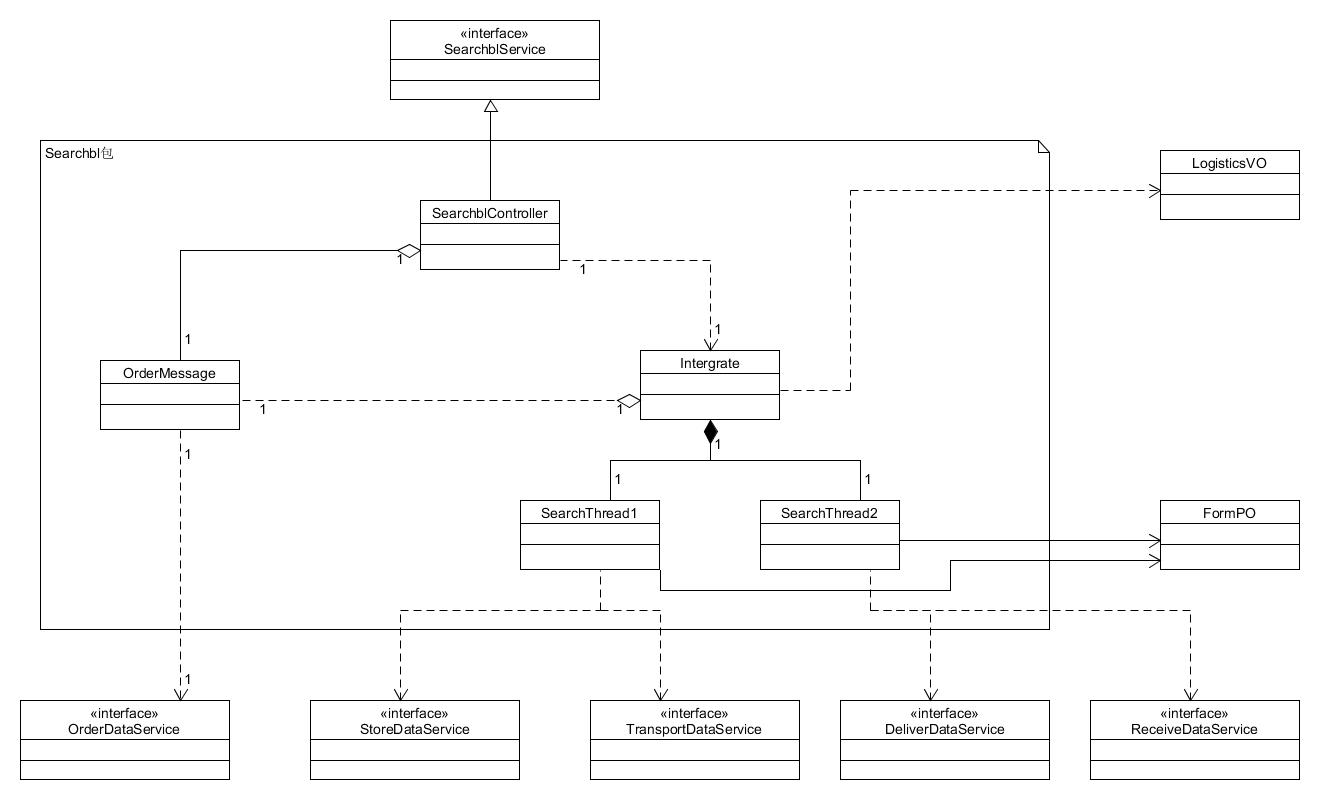
1. 模块概述

模块负责按照用户输入的订单号来查询订单状态，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。

1. 整体结构

由于该模块逻辑简单，就采用了集中控制的风格，为了追求响应速度，采用了多线程的方式。该模块会首先确认订单是否存在，之后会爆发出多线程来搜索、整合信息，事实上可能存在冲突、死锁等问题

模块的设计如图



模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| SearchblController | 负责控制、安排搜索的进度与结果 |
| OrderMessage | 负责确认、获取订单信息（我认为用户输入错订单的可能性较大，所以特地增加审核订单号的过程） |
| Intergrate | 负责搜素、整合信息，并构建LogisticsVO |

1. 模块内部类的接口规范

**SearchblController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SearchblController.  searchOrder | 语法 | Public LogisticsVO SearchOrder (String OrderID) |
| 前置条件 | 订单号格式正确 |
| 后置条件 | 对订单信息搜索；若找到，返回整合后的LogisticsVO，否则返回null |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderMessage.checkOrderID(String OrderID) | 查询ID是否存在 | |
| Intergrate.getLogistics(String OrderID) | 获取订单的状态信息 | |

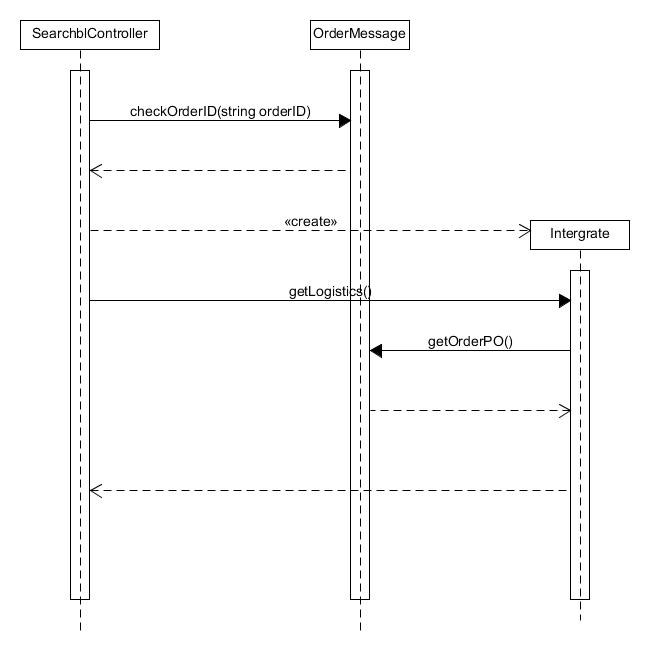
**OrderMessage的接口规范**

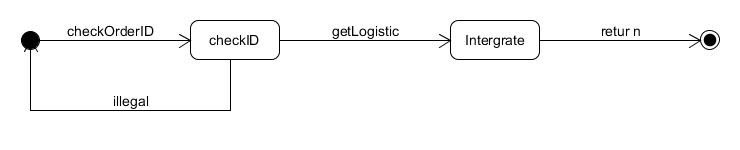
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderMessage.  checkOrderID | 语法 | public boolean checkOrderID(String OrderID) |
| 前置条件 | orderID为订单号 |
| 后置条件 | 若订单号存在，返回true，否则返回false |
| OrderMessage.  getOrderPO | 语法 | public OrderPO getOrderPO() |
| 前置条件 | 订单号存在 |
| 后置条件 | 返回订单信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDataService.  getOrderPO | 根据ID获得订单信息 | |

**Intergrate的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Intergrate.  getLogistics | 语法 | public boolean checkOrderID(String OrderID) |
| 前置条件 | orderID为订单号 |
| 后置条件 | 若订单号存在，返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| TransportDataService.  getTransportPO(String OrderID) | 获取中转/装车单的信息 | |
| ReceiveDataService.  getReceivePO(String OrderID) | 获取到达单的信息 | |
| DeliverDataService.  getDeliverPO(String OrderID) | 获得派件单信息 | |
| StoreDataService.  getStoreInPO | 获取入库单信息 | |
| StoreDataService.  getStoreOutPO | 获取出库单信息 | |
|  |  | |
| OrderMessage.  getOrderPO | 获取订单信息 | |

1. 业务逻辑层的动态模型





1. 业务逻辑层的设计原理

采用集中式控制风格，由controller决定ordermessage和integrate的存亡。

### 4.1.9Transportbl模块

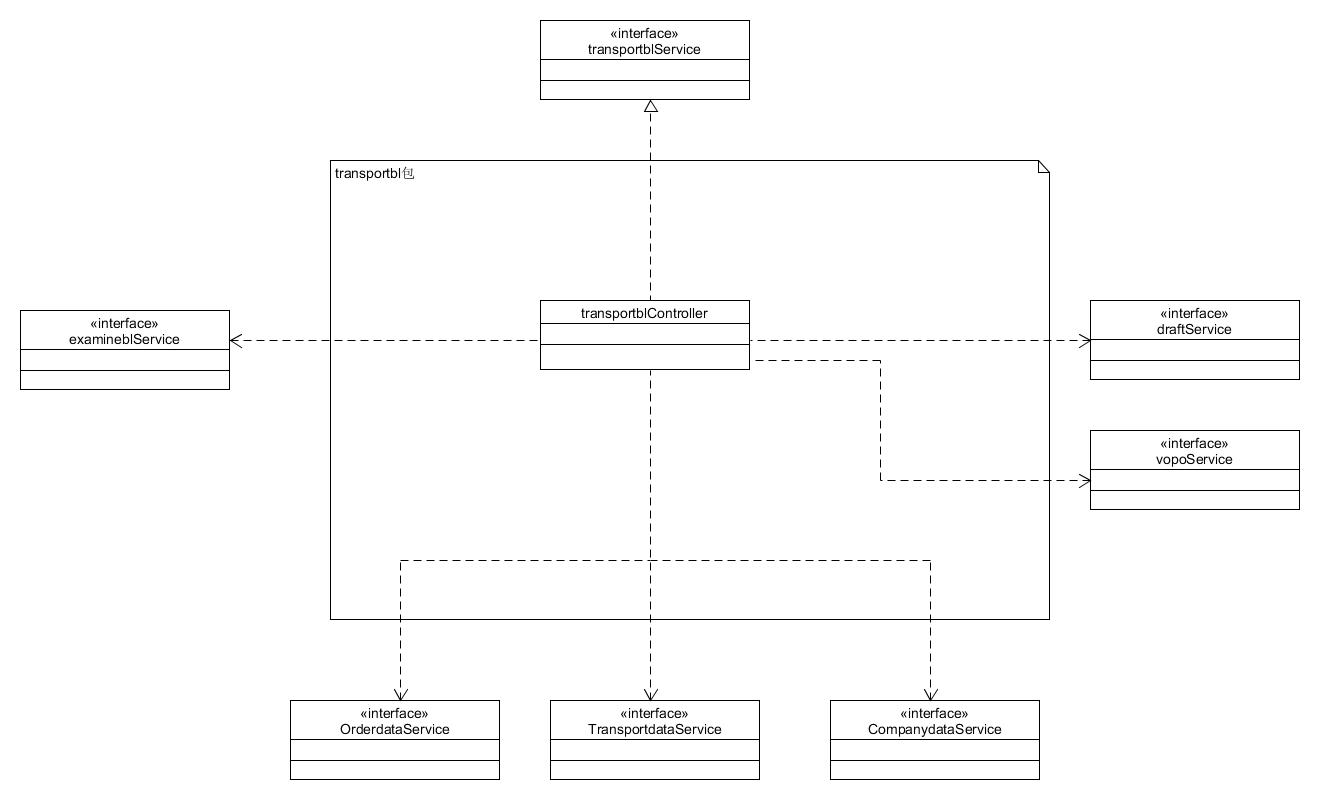
1. 模块概述

Transportbl模块负责实现营业厅装车单和中转中心中转单的新建、提交工作。

1. 整体结构

事实上，个人感觉这个包被架空了。一方面，包本身的逻辑比较简单；同时，它所需要实现的功能与其他新建订单包类似，因此采用了统一的工具实现。因此包内部的组织结构就相对较为简单

Transportbl模块的设计如图



Transportbl模块各个类的职责

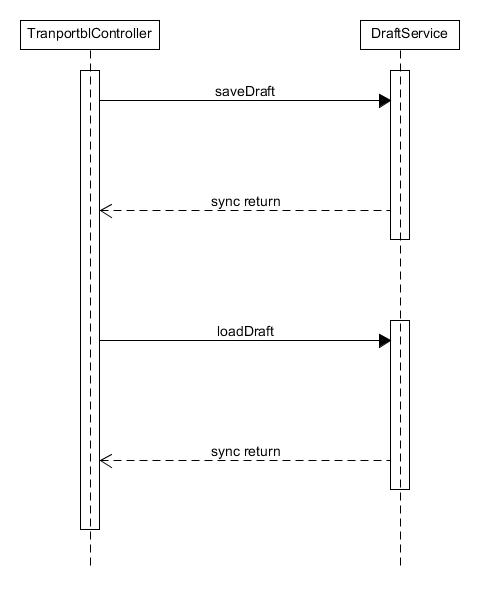
|  |  |
| --- | --- |
| 模块 | 职责 |
| TransportblController | 负责实现中转/装车单的服务 |

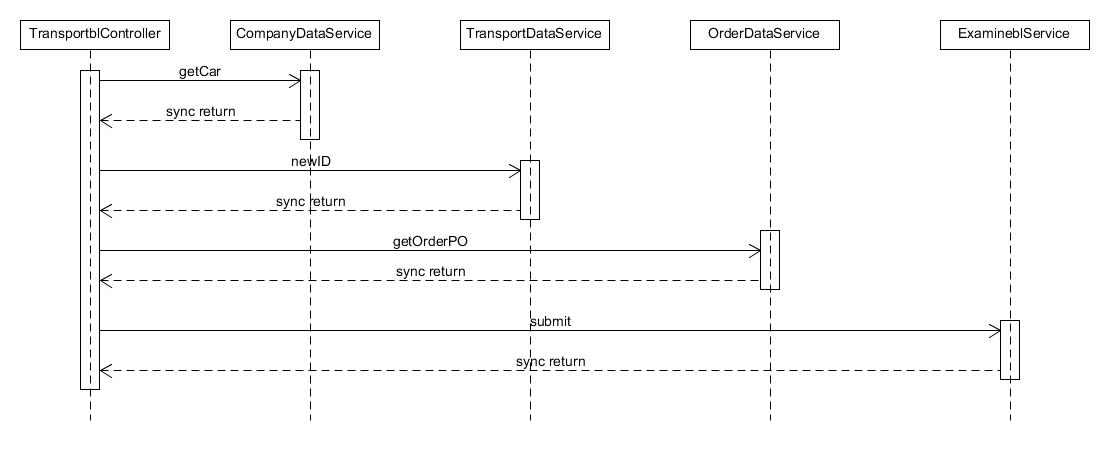
1. 模块内部类的接口规范

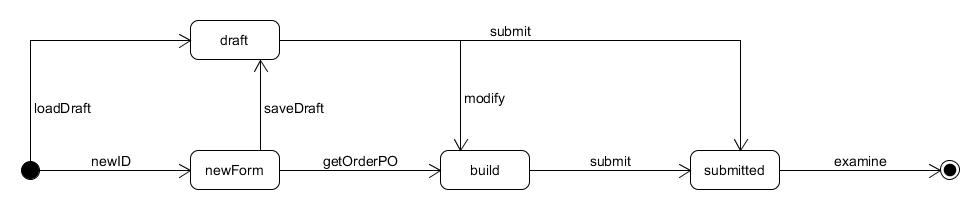
**TransportBL的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransportblController. checkFormat | 语法 | public List< CheckFormMessage > checkFormat(LoadPO form, boolean isFinal) |
| 前置条件 | 输入的LoadPO不为空,isFinal判断是否为提交表格的操作 |
| 后置条件 | 检查输入信息是否符合系统要求，若符合，返回的list中只有一个显示正确的CheckFormMessage, 否则返回能指示出错误的CheckFormMessage的列表 |
| TransportblController.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的装车单编号 |
| TransportblController.getCar | 语法 | public CarVO getCar(String hallID) |
| 前置条件 | 传入营业厅编号 |
| 后置条件 | 返回可用车辆的信息 |
| TransportblController.submit | 语法 | public OperationMessage submit(LoadPO form) |
| 前置条件 | LoadPO已经通过检查 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| TransportblController.saveDraft | 语法 | public OperationMessage saveDraft(LoadPO draft) |
| 前置条件 | 输入的LoadPO不为空 |
| 后置条件 | 将LoadPO序列化成文件存储 |
| TransportblController.loadDraft | 语法 | public LoadPO loadDraft() |
| 前置条件 | 有对应的序列化文件存在 |
| 后置条件 | 将序列化文件解序列化成对应的LoadPO对象返回 |
| TransportblController. checkFormat | 语法 | public List<OperationMessage> checkFormat(CenterPO form) |
| 前置条件 | 输入的CenterPO不为空 |
| 后置条件 | 检查输入信息是否符合系统要求，若符合，返回的list中只有一个显示正确的OperationMessage, 否则返回能指示出错误的OperationMessage的列表 |
| TransportblService.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回合法的新中转单编号 |
| TransportblController.submit | 语法 | public OperationMessage submit(CenterPO form) |
| 前置条件 | CenterPO已经通过检查 |
| 后置条件 | 若网络连接正常，返回正常的OperationMessage，否则返回相应的提示信息 |
| TransportblController.saveDraft | 语法 | public OperationMessage saveDraft(CenterPO draft) |
| 前置条件 | 输入的CenterPO不为空 |
| 后置条件 | 将CenterPO序列化成文件存储 |
| TransportblController.loadDraft | 语法 | public CenterPO loadDraft() |
| 前置条件 | 有对应的序列化文件存在 |
| 后置条件 | 将序列化文件解序列化成对应的CenterPO对象返回 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ExamineService.  submit(FormPO po) | 提交表格审批 | |
| TransportDataService.newID | 得到新的中转单/装车单号 | |
| OrderDataService.getOrderPO | 获得订单信息 | |
| CompanyDataService.getCar | 获得可用车辆信息 | |

1. 业务逻辑层的动态模型







1. 业务逻辑层的设计原理

采用集中式的控制



## 数据层的分解

数据层的开发包图请参见体系结构描述文档图4。

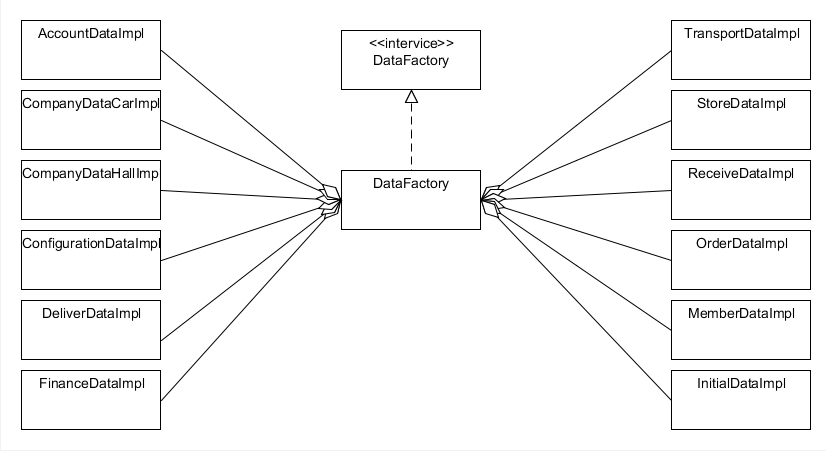
### DataFactoryService

1. 模块概述

DataFactoryService模块是一个工厂类，用来创建其他的DataService实例，并将其统一管理起来。

1. 整体结构

DataFactoryService模块设计如下图所示



1. 模块内部的接口规范

DataFactory内部的的接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DataFactory. createDataService | 语法 | public DataService<?> createDataService(String name) |
| 前置条件 | data的名字 |
| 后置条件 | 返回DataService的子类对象 |
| DataFactory. createChatRemindService | 语法 | public ChatRemindService createChatRemindService () |
| 前置条件 | 无 |
| 后置条件 | 返回ChatRemindService对象 |
| DataFactory. createExamineSubmitService | 语法 | public ExamineSubmitService createExamineSubmitService() |
| 前置条件 | 无 |
| 后置条件 | 返回ExamineSubmitService对象 |
| DataFactory. createDataService | 语法 | public ExamineManageService createExamineManageService() |
| 前置条件 | 无 |
| 后置条件 | 返回ExamineManageService对象 |

### AccountDataService

1. 模块概述

AccountDataService模块提供对系统账号的基本操作

1. 模块内类接口规范

AccountDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountDataService. getAccountPO | 语法 | public AccountPO getAccountPO (String accountID) |
| 前置条件 | AccountID在数据中存在 |
| 后置条件 | 查找返回相应的AccountPO |
| AccountDataService.getAccountPOs | 语法 | public AccountPO getAccountPOs() |
| 前置条件 | 无 |
| 后置条件 | 返回所有的AccountPO |
| AccountDataService.insert | 语法 | Public OperationMessage insert  (AccountPO po) |
| 前置条件 | 同样的po在数据中不存在，参数为插入对象 |
| 后置条件 | 增加一条po数据，返回结果信息 |
| AccountDataService.delete | 语法 | public OperationMessage delete  (String name) |
| 前置条件 | 同样的po在数据中存在,传入name |
| 后置条件 | 删除po，返回结果信息 |
| AccountDataService.update | 语法 | public OperationMessage update  (AccountPO po) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新po，返回结果信息 |
| AccountDataService.checkID | 语法 | public Boolean checkID(String accountID) |
| 前置条件 | 无 |
| 后置条件 | 检查ID是否存在于数据中，若是，则返回true，否则返回false |
| AccountDataService.checkAccount | 语法 | Public boolean checkAccount(String id , String password) |
| 前置条件 | CheckID返回true |
| 后置条件 | 若ID与password匹配，则返回true，否则，返回false |
| OrderDataService.newAccountID | 语法 | public String newAccountID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的账户名 |

### CompanyDataService

1. 模块概述

CompanyDataService模块提供对公司体系调整的操作（营业厅，中转中心，车辆的管理）

1. 模块内类接口规范

CompanyDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| companyDataService.getHall | 语法 | public ArrayList<HallPO> getHall() |
| 前置条件 | 无 |
| 后置条件 | 返回公司所有营业厅的信息 |
| companyDataService.newHallID | 语法 | public String newHallID(String centerID) |
| 前置条件 | 上属中转中心编号 |
| 后置条件 | 返回一个合法的新编号 |
| companyDataService.addHall | 语法 | public OperationMessage addHall(HallPO Hall) |
| 前置条件 | Hall为新营业厅的信息 |
| 后置条件 | 若新建成功，返回成功；否则，返回失败原因 |
| companyDataService.deleteHall | 语法 | public OperationMessage deleteHall(HallPO hall) |
| 前置条件 | hall为待删除营业厅信息；已经获取所有的营业厅信息 |
| 后置条件 | 若删除成功返回成功，否则返回失败原因 |
| companyDataService.modifyHall | 语法 | public OperationMessage modifyHall(HallPO hall) |
| 前置条件 | hall为修改后的营业厅信息 |
| 后置条件 | 若修改成功，返回成功；否则返回失败原因 |
| companyDataService.getCenter | 语法 | public ArrayList<CenterPO> getCenter() |
| 前置条件 | 无 |
| 后置条件 | 获取所有中转中心的信息 |
| companyDataService.addCenter | 语法 | public OperationMessage addCenter(CenterPO center) |
| 前置条件 | center为新建中转中心的信息 |
| 后置条件 | 若新建成功，返回成功；否则返回失败原因 |
| companyDataService.deleteCenter | 语法 | public OperationMessage deleteCenter(CenterPO center) |
| 前置条件 | center为待删除中转中心信息 |
| 后置条件 | 若删除成功，返回成功，否则返回失败原因 |
| companyDataService.modifyCenter | 语法 | public OperationMessage modifyCenter(CenterPO center) |
| 前置条件 | center为待修改中转中心信息 |
| 后置条件 | 若修改成功，返回成功；否则，返回失败原因 |
| companyDataService.getCar | 语法 | public ArrayList<CarPO> getCar() |
| 前置条件 | 无 |
| 后置条件 | 返回公司车辆信息 |
| companyDataService.newCarID | 语法 | public String newCarID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个合法的新编号 |
| companyDataService.addCar | 语法 | public OperationMessage addCar(CarPO car) |
| 前置条件 | car为新增车辆的信息 |
| 后置条件 | 若新建成功则返回成功，否则返回失败原因 |
| companyDataService.deleteCar | 语法 | public OperationMessage deleteCar(CarPO car) |
| 前置条件 | car为待删除的车辆信息 |
| 后置条件 | 若删除成功返回成功，否则返回失败原因 |
| companyDataService.modifyCar | 语法 | public OperationMessage modifyCar(CarPO car) |
| 前置条件 | car为修改后的车辆信息 |
| 后置条件 | 若修改成功则返回成功；否则返回失败原因 |

### ConfigurationDataService

1. 模块概述

ConfigurationDataService模块提供有关公司基础信息的操作的操作（常量，薪水策略等）

1. 模块内类接口规范

ConfigurationDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| configurationDataService.getCityDistance | 语法 | public ArrayList<CityDistancePO> getCityDistance() |
| 前置条件 | 无 |
| 后置条件 | 返回城市距离信息 |
| configurationDataService.modifyCityDistance | 语法 | public OperationMessage modifyCityDistance(CityDistancePO after) |
| 前置条件 | after为修改后的信息 |
| 后置条件 | 若修改成功返回成功，否则失败原因 |
| configurationDataService.getSalaryStrategy | 语法 | public ArrayList<SalaryStrategyPO> getSalaryStrategy () |
| 前置条件 | 无 |
| 后置条件 | 返回员工的薪水策略信息 |
| configurationDataService.modifySalaryStrategy | 语法 | public OperationMessage modifySalaryStrategy (SalaryStrategyPO SalaryStrategy) |
| 前置条件 | SalaryStrategy为修改的薪水策略 |
| 后置条件 | 若修改成功则返回成功；否则返回失败原因 |
| configurationDataService.getPack | 语法 | public ArrayList<PackPO>  getPack() |
| 前置条件 | 无 |
| 后置条件 | 返回包装费用信息 |
| configurationDataService.modifyPack | 语法 | public OperationMessage modifyPack(PackPO pack) |
| 前置条件 | pack为修改后的包装信息 |
| 后置条件 | 若修改成功返回成功；否则返回失败原因 |
| configurationDataService.getPrice | 语法 | public PricePO getPrice() |
| 前置条件 | 无 |
| 后置条件 | 返回当前快递价格 |
| configurationDataService.modifyPrice | 语法 | public OperationMessage modifyPrice(PricePO price) |
| 前置条件 | price为修改后价格信息 |
| 后置条件 | 若修改成功返回成功；否则返回失败原因 |
| configurationDataService.getProportion | 语法 | public ProportionPO getProportion () |
| 前置条件 | 无 |
| 后置条件 | 返回不同快递收费比例 |
| configurationDataService.modifyProportion | 语法 | public OperationMessage modifyProportion(ProportionPO proportion) |
| 前置条件 | proportion为修改后的比例 |
| 后置条件 | 若修改成功，返回成功；否则返回失败原因 |
| configurationDataService. getInstitutionDistance | 语法 | public Object getInstitutionDistance() |
| 前置条件 | 无 |
| 后置条件 | 返回各个机构的距离信息 |
| configurationDataService. newInstitutionDistanceSearch | 语法 | public Object[] newInstitutionDistanceSearch(String ID) |
| 前置条件 | 传入新建机构的编号 |
| 后置条件 | 返回需要填写的距离信息Sea |
| configurationDataService. newInstitutionDistanceInsert | 语法 | public OperationMessage newInstitutionDistanceInsert(String ID,Object[] ob) |
| 前置条件 | ID为新机构编号，ob为相关距离信息 |
| 后置条件 | 返回操作结果 |
| configurationDataService. modifyInstitutionDistance | 语法 | public OperationMessage modifyInstitutionDistance(String ID,Object ob) |
| 前置条件 | ID为修改的机构编号 |
| 后置条件 | ob为修改内容 |

### DeliverDataService

1. 模块概述

DeliverDataService模块提供货物派送的一系列操作（派件单）

1. 模块内类接口规范

DeliverDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DeliverDataService.insert | 语法 | public OperationMessage insert  (DeliverPO po) |
| 前置条件 | 同样的po在数据中不存在，参数为插入对象 |
| 后置条件 | 增加一条po数据，返回结果信息 |
| DeliverDataService.geFormPO | 语法 | public DeliverPO getFormPO(String id) |
| 前置条件 | 参数为派件单号 |
| 后置条件 | 查找返回相应的结果 |
| DeliverDataService.delete | 语法 | public OperationMessage delete  (String id) |
| 前置条件 | 同样的po在数据中存在,传入id |
| 后置条件 | 删除po，返回结果信息 |
| DeliverDataService.update | 语法 | public OperationMessage update  (DeliverPO po) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新po，返回结果信息 |
| DeliverDataService.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的派件单ID |
| DeliverDataService. clear | 语法 | public OperationMessage clear () |
| 前置条件 | 无 |
| 后置条件 | 清空DeliverPO数据 |
| DeliverDataService. getAll | 语法 | public ArrayList< DeliverPO >  getAll () |
| 前置条件 | 无 |
| 后置条件 | 返回所有派件单PO数据 |

### FinanceDataService

1. 模块概述

FinanceDataService模块提供有关财务支出审查的操作（收款单，付款单）

1. 模块内类接口规范

FinanceDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinanceFormDataService. updateRevenuePOs | 语法 | public LinkedList<RevenuePO>  updateRevenuePOs(String staffID) |
| 前置条件 | 无 |
| 后置条件 | 返回服务端staffID对应的队列中的收款单数据的对象 |
| FinanceFormDataService. updatePaymentPOs | 语法 | public LinkedList <PaymentPO>  updatePaymentPOs(String staffID) |
| 前置条件 | 无 |
| 后置条件 | 返回服务端staffID对应的队列中的付款单数据的对象 |
| FinanceFormDataService. downloadAllPaymentPOs | 语法 | public LinkedList <PaymentPO>  downloadAllPaymentPOs() |
| 前置条件 | 无 |
| 后置条件 | 将所有收款单的数据下载到本地 |
| FinanceFormDataService. downloadAllRevenuePOs | 语法 | public LinkedList <RevenuePO>  downloadAllRevenuePOs() |
| 前置条件 | 无 |
| 后置条件 | 将所有付款单的数据下载到本地 |
| FinanceFormDataService. | 语法 | publicRevenuePO getRevenuePO(String formID) |
| 前置条件 | 无 |
| 后置条件 | 返回传入的收款单编号对应的RevenuePO |
| FinanceFormDataService. getPaymentPO | 语法 | public PaymentPO getPaymentPO(String formID) |
| 前置条件 | 无 |
| 后置条件 | 返回传入的入款单编号对应的PaymentPO |
| FinanceFormDataService. getNewRevenueID | 语法 | public String getNewRevenueID  (String date, String hallID) |
| 前置条件 | date和hallID格式正确 |
| 后置条件 | 返回新的未使用过的收款单ID |
| FinanceFormDataService. getNewPaymentID | 语法 | public String getNewPaymentID  (String date) |
| 前置条件 | date格式正确 |
| 后置条件 | 返回新的未使用过的收款单ID |
| BankAccountDataService. | 语法 | public BankAccountPO getBankAccount(String bankID); |
| 前置条件 | 无 |
| 后置条件 | 返回传入的银行账号id对应的银行账户 |
| BankAccountDataService. updateAccountOperations | 语法 | public List<BankAccountOperation>  updateAccountOperations(String staffID) |
| 前置条件 | 无 |
| 后置条件 | 返回服务端staffID对应的队列中未同步的对账户进行的操作 |
| BankAccountDataService. uploadAccountOperations | 语法 | public OperationMessage  uploadAccountOperations  (String staffID, LinkedList<BankAccountOperation> operations) |
| 前置条件 | 无 |
| 后置条件 | 将本地队列中未上传的对账户的操作上传到服务端 |
| BankAccountDataService. downloadAllAccounts | 语法 | public List<BankAccountPO >  downloadAllAccounts() |
| 前置条件 | 无 |
| 后置条件 | 下载所有的账户的数据 |
| BankAccountDataService. getNewBankID | 语法 | public String getNewBankID() |
| 前置条件 | 无 |
| 后置条件 | 返回未使用过的银行账户ID号 |
| BankAccountDataService.  checkIsNameUsed | 语法 | public boolean  checkIsNameUsed(String name) |
| 前置条件 | 无 |
| 后置条件 | 检查传入的账户名是否已经在使用 |

### InitialDataService

1. 模块概述

InitialDataService模块提供有关初期建账的操作

1. 模块内类接口规范

InitialDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InitialDataService.getInitialDataPO | 语法 | public InitialDataPO getInitialDataPO(String version) |
| 前置条件 | 传入版本号信息 |
| 后置条件 | 返回InitialDataPO持久化对象 |
| InitialDataService.requestInitData | 语法 | public OperationMessage requestInitData(String staffID) |
| 前置条件 | 参数为员工ID |
| 后置条件 | 返回操作结果，系统进入建账中状态，不回应期初建账外其他的操作 |
| InitialDataService.uploadInitialData | 语法 | public OperationMessage uploadInitialData(String staffID, InitialDataPO newData) |
| 前置条件 | 传入员工ID和新的初期建账信息 |
| 后置条件 | 返回操作信息,将新的期初建账信息上传 |
| InitialDataService.abortnitData | 语法 | public OperationMessage abortInitData(String staffID) |
| 前置条件 | 传入员工ID |
| 后置条件 | 返回操作信息, 终止系统建账中的状态，用之前的帐继续运行 |

### MemberDataService

1. 模块概述

MenberDataService模块提供有关人员信息的操作

1. 模块内类接口规范

MemberDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| memberDataService.getStaff | 语法 | public ArrayList<StaffVO> getStaff(StaffTypeEnum StaffTypeEnum) |
| 前置条件 | StaffTypeEnum为员工类型的枚举 |
| 后置条件 | 根据StaffTypeEnum获取相应的所有员工信息 |
| memberDataService.modifyStaff | 语法 | public OperationMessage modifyStaff(StaffPO after) |
| 前置条件 | after为修改后的信息 |
| 后置条件 | 若修改成功返回成功，否则失败原因 |
| memberDataService.addStaff | 语法 | public OperationMessage addStaff(StaffPO staff) |
| 前置条件 | Staff为新员工的信息;已经获取所有员工信息 |
| 后置条件 | 若新建成功则返回成功；否则，返回失败原因 |
| memberDataService.dismissStaff | 语法 | public OperationMessage dismissStaff(StaffPO Staff) |
| 前置条件 | Staff为待解雇员工信息；已经获取员工信息列表 |
| 后置条件 | 若删除成功则返回成功；否则返回失败原因 |
| memberDataService.newStaffID | 语法 | public String newStaffID(StaffTypeEnum type) |
| 前置条件 | type为新员工的工种 |
| 后置条件 | 返回合法的员工ID |

### OrderDataService

1. 模块概述

OrderDataService模块提供有关人员信息的操作（订单）

1. 模块内类接口规范

OrderDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderDataService.getFormPO | 语法 | public OrderPO getFormPO(String id) |
| 前置条件 | 参数为快递单号 |
| 后置条件 | 查找返回相应的结果 |
| OrderDataService.insert | 语法 | public OperationMessage insert  (OrderPO po) |
| 前置条件 | 同样的po在数据中不存在，参数为插入对象 |
| 后置条件 | 增加一条po数据，返回结果信息 |
| OrderDataService.delete | 语法 | public OperationMessage delete  (String id) |
| 前置条件 | 同样的po在数据中存在,传入id |
| 后置条件 | 删除po，返回结果信息 |
| OrderDataService.update | 语法 | public OperationMessage update  (OrderPO po) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新po，返回结果信息 |
| OrderDataService.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的订单号 |
| OrderDataService. clear | 语法 | public void clear () |
| 前置条件 | 无 |
| 后置条件 | 清空OrderPO数据 |
| OrderDataService. getAll | 语法 | public ArrayList< OrderPO >  getAll () |
| 前置条件 | 无 |
| 后置条件 | 返回所有订单PO数据 |

### ReceiveDataService

1. 模块概述

ReceiveDataService模块提供货物到达的一系列操作（到达单）

1. 模块内类接口规范

ReceiveDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReceiveDataService.insert | 语法 | public OperationMessage insert  (ReceivePO po) |
| 前置条件 | 同样的po在数据中不存在，参数为插入对象 |
| 后置条件 | 增加一条po数据，返回结果信息 |
| ReceiveDataService.getFormPO | 语法 | public ReceivePO gerFormPO(String id) |
| 前置条件 | 参数为到达单号 |
| 后置条件 | 查找返回相应的结果 |
| ReceiveDataService.delete | 语法 | public OperationMessage delete  (String id) |
| 前置条件 | 同样的po在数据中存在,传入id |
| 后置条件 | 删除po，返回结果信息 |
| ReceiveDataService.update | 语法 | public OperationMessage update  (ReceivePO po) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新po，返回结果信息 |
| ReceiveDataService.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的到达单ID |
| ReceiveDataService. clear | 语法 | public OperationMessage clear () |
| 前置条件 | 无 |
| 后置条件 | 清空ReceivePO数据 |
| ReceiveDataService. getAll | 语法 | public ArrayList< ReceivePO >  getAll () |
| 前置条件 | 无 |
| 后置条件 | 返回所有到达单PO数据 |

### StoreDataService

1. 模块概述

StoreDataService模块提供有关货物库存的一系列操作（入库单，出库单）

1. 模块内类接口规范

StoreDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StoreFormDataService.  getStoreInPO | 语法 | public StoreInPO getStoreInPO  (String id) |
| 前置条件 | 无 |
| 后置条件 | 返回id对应的入库单的数据 |
| StoreFormDataService.  getStoreOutPO | 语法 | public StoreOutPO getStoreOutPO  (String id) |
| 前置条件 | 无 |
| 后置条件 | 返回id对应的入库单的数据 |
| StoreFormDataService. updateStoreInPOs | 语法 | public List<StoreInPO>  updateStoreInPOs(String staffID) |
| 前置条件 | 无 |
| 后置条件 | 返回staffID对应的服务器队列中的所有的入库单数据 |
| StoreFormDataService. downloadAllStoreInPOs | 语法 | public List<StoreInPO> downloadAllStoreInPOs  (String centerID) |
| 前置条件 | 无 |
| 后置条件 | 返回centerID对应的所有的入库单 |
| StoreFormDataService. updateStoreOutPOs | 语法 | public List<StoreOutPO> updateStoreOutPOs  (String centerID) |
| 前置条件 | 无 |
| 后置条件 | 返回staffID对应的服务器队列中的所有的出库单数据 |
| StoreFormDataService. downloadAllStoreOutPOs | 语法 | public List<StoreOutPO> downloadAllStoreOutPOs  (String staffID) |
| 前置条件 | 无 |
| 后置条件 | 返回centerID对应的所有的出库单 |
| StoreModelDataService. updateModelOperations | 语法 | public List<StoreModelOperation>  updateModelOperations  (String centerID, String staffID) |
| 前置条件 | 无 |
| 后置条件 | 返回对应staffID队列里的未更新的对仓库做的操作 |
| StoreModelDataService. uploadModelOperations | 语法 | public OperationMessage  uploadModelOperations  (String centerID, String staffID,  List<StoreModelOperation> operations) |
| 前置条件 | 无 |
| 后置条件 | 将对仓库做的操作上传到服务器 |
| StoreModelDataService. downloadStoreModel | 语法 | public StoreModel  downloadStoreModel  (String centerID) |
| 前置条件 | 无 |
| 后置条件 | 下载仓库的数据 |

### TransportDataService

1. 模块概述

TransportDataService模块提供有关货物中转的一系列操作（中转单，装车单）

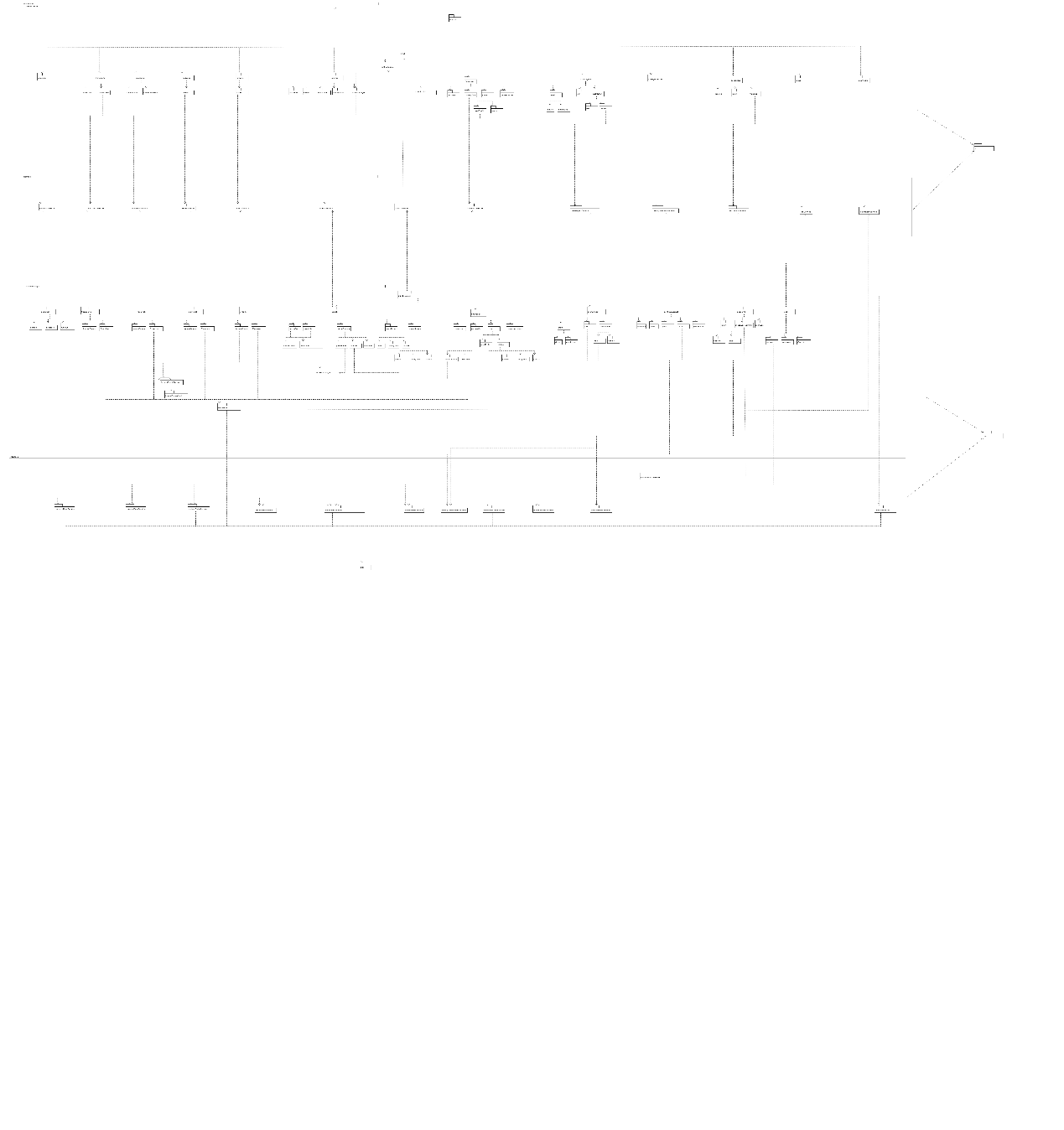
1. 模块内类接口规范

TransportDataService接口规范如下表

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransportDataService.insert | 语法 | public OperationMessage insert  (TransportPO po) |
| 前置条件 | 同样的po在数据中不存在，参数为插入对象 |
| 后置条件 | 增加一条po数据，返回结果信息 |
| TransportDataService.getFormPO | 语法 | public TransportPO getFormPO (String id) |
| 前置条件 | 参数为快递单号 |
| 后置条件 | 查找返回相应的结果 |
| TransportDataService.delete | 语法 | public OperationMessage delete  (String id) |
| 前置条件 | 同样的po在数据中存在,传入id |
| 后置条件 | 删除po，返回结果信息 |
| TransportDataService.update | 语法 | public OperationMessage update  (TransportPO po) |
| 前置条件 | 同样的po在数据中存在 |
| 后置条件 | 更新po，返回结果信息 |
| TransportDataService.newID | 语法 | public String newID() |
| 前置条件 | 无 |
| 后置条件 | 返回一个新的装车单/中转单编号 |
| TransportDataService. clear | 语法 | public void clear () |
| 前置条件 | 无 |
| 后置条件 | 清空TransportPO数据 |
| ReceiveDataService. getAll | 语法 | public ArrayList< TransportPO>  getAll () |
| 前置条件 | 无 |
| 后置条件 | 返回所有中转单PO数据 |

# 5.依赖视角

## 5.1客户端包图



## 5.2服务器包图

