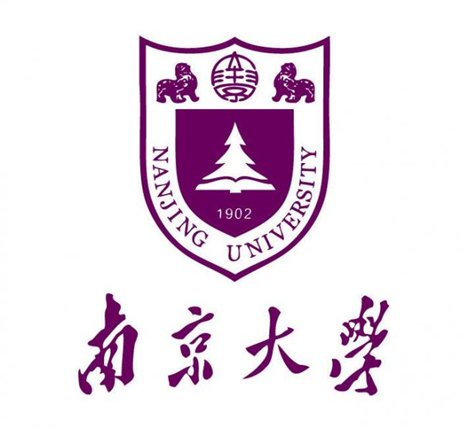
|  |  |  |  |
| --- | --- | --- | --- |
| 创建者 | 丁银门 | 创建日期 | 2015/11/15 |
| 最后一次更新者 | 丁银门 | 最后一次更新日期 | 2015/11/16 |

# 物流管理系统LMS软件详细设计描述文档



第六小组

邓又铭 丁银门

段正谋 胡成毅

目录

1. 引言…………………………………………………………2

1.1 编制目的…………………………………………2

1.2 词汇表……………………………………………2

1. 产品概述……………………………………………………2
2. 体系概述……………………………………………………3
3. 结构视角……………………………………………………3
   1. 客户端启动模块的分解…………………………4
   2. 服务器端启动模块的分解………………………5
   3. 用户界面层………………………………………7
   4. 业务逻辑层的分解………………………………9
   5. 数据层的分解……………………………………33
4. 依赖视角…………………………………………………35

**1.引言**

**1.1编制目的**

本报告详细完成对物流管理系统的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员及用户的沟通。本报告面向开发人员、测试人员及最终用户的编写，是了解系统的导航。

**1.2 词汇表**

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| LMS | 物流管理系统 | 无 |
| Warehouse | 仓库 | 无 |
| StockOut | 出库 | 无 |
| Stockin | 入库 | 无 |
| inventoryAlarm | 库存报警 | 无 |
| inventoryCheck | 库存查看 | 无 |
| inventoryCount | 库存盘点 | 无 |
| ruleChange | 改变用户权限 | 无 |
| ruleApply | 申请权限 | 无 |
| costManage | 付款单管理 | 无 |
| getcostControl | 获得成本收益信息 | 无 |
| getOperationState | 获得经营情况 | 无 |
| queryInstititionInfo | 返回一个机构列表 | 无 |
| infoHelper | 生成单据时，根据订单类型，获取生成单据所需要的额外信息 | 无 |
| SendReceipt | 寄件单 | 快递员 |
| AcceptReceipt | 收件单 | 快递员 |
| CollectionReceipt | 收件单 | 营业厅业务员 |
| DeliveryReceipt | 派件单 | 营业厅业务员 |
| EarningReceipt | 收款单 | 无 |
| loadingReceipt | 装车单 | 无 |
| transitReceipt | 装运单 | 无 |
| PaymentReceipt | 付款单 | 无 |
| ImportReceipt | 入库单 | 无 |
| ExportReceipt | 出库单 | 无 |

**1.3 参考资料**

《软件工程与计算（卷二）——软件开发的技术基础》骆斌 主编，丁二玉 刘钦 编著

**2.产品概述**

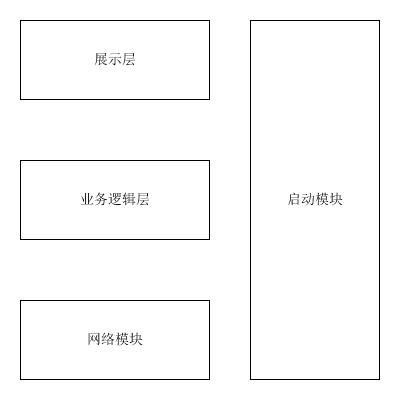
物流管理系统是为一本地民营物流企业开发的业务管理系统，开发的目标是帮助该物流企业处理日常的重点业务，包括物流货物信息管理、车辆信息管理、司机信息管理、装运管理、单据管理、仓库管理、人员和机构管理、账目管理、账户管理模块及其下属功能。

**3.体系概述**

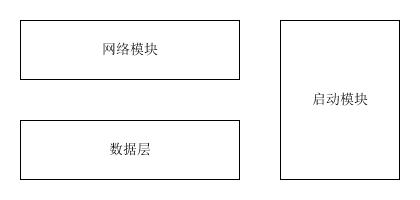
物流管理系统中，选择了分层体系结构风格，将系统分为3层（展示层，业务逻辑层）能够很好地示意整个高层抽象。展示层包含GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。

**4.结构视角**

模块视角



**客户端模块视图**



**服务器端模块视图**

**客户端各层的职责**

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 用户界面层 | 基于窗口的连锁商店客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行相应并进行业务处理逻辑 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

**服务器端各层的职责**

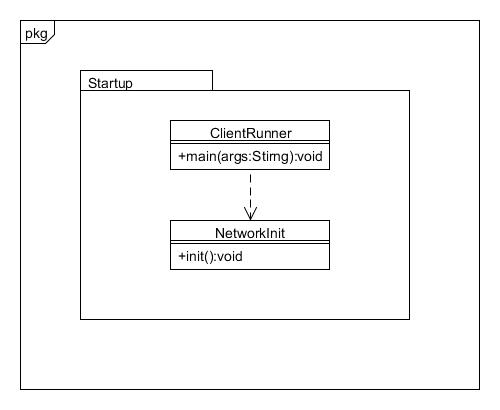
|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

**4.1 客户端启动模块的分解**

**4.1.1 客户端启动模块的静态模型**

**一、整体结构**

客户端启动模块主要包含主程序类ClientRunner和网络初始化类NetworkInit。主程序类ClientRunner是整个程序的入口，负责调用NetworkInit类的方法完成初始化工作，并且启动用户界面。而NetworkInit类主要负责在客户端查找到服务器端的RMI服务。



**客户端启动模块的结构**

**二、客户端启动模块子模块的职责**

**客户端启动层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ClientRunner | 程序入口。根据配置初始化好相应的网络连接。初始化之后，进入用户登录界面。 |
| NetworkInit | 在客户端查找RMI服务 |

**三、客户端启动模块子模块的接口规范**

**客户端启动模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ClientRunner | 语法 | main(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 创建NetworkInit对象，并调用其init方法初始化相应的网络连接。之后进入登录界面 |
| NetworkInit | 语法 | Init() |
| 前置条件 | 存有网络配置文件 |
| 后置条件 | 在客户端查找RMI服务，将服务设置为其成员变量 |

**客户端启动模块需要的服务接口**

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| Presentation.Mainframe.init() | 初始化窗口，显示登陆界面 |
| Java.rmi.Naming.lookup() | 查找需要的RMI服务 |

**4.1.2 客户端启动模块设计原理**

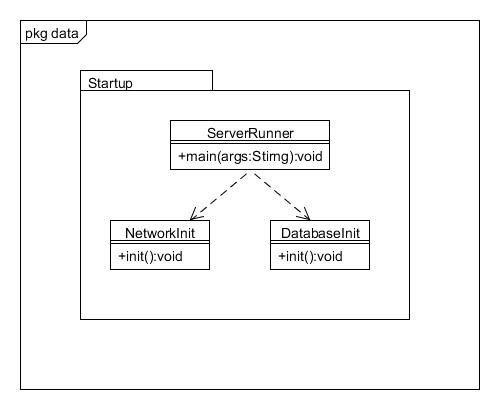
启动模块是一个独立于分层体系之外的模块，可以直接访问各层的接口，完成系统启动的需求。

**4.2 服务器端启动模块的分解**

**4.2.1 服务器端启动模块的静态模型**

**一、整体结构**

服务器端启动模块逻辑上是一个横跨服务器端各层的模块，为服务器端各层提供初始化启动服务。服务器端启动模块与客户端类似，也包含主程序类ServerRunner和网络初始化类NetworkInit。主程序类ServerRunner同样是整个程序的入口，负责调用NetworkInit类的方法完成网络初始化工作，并启动用户界面。只是NetworkInit类的职责和客户端有区别，主要负责在服务器端启动RMI注册服务，创建远程对象的实例和注册涉及的RMI服务。



**服务器端启动模块的结构**

**二、服务器端启动模块子模块的职责**

**表 0-6服务器端启动层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ServerRunner | 程序入口。根据配置初始化好相应的网络连接。初始化之后，进入用户登录界面 |
| NetworkInit | 在服务器端启动RMI服务，创建远程对象的实例和注册涉及的RMI服务 |
| DatabaseInit | 初始化数据库 |

**三、启动模块子模块的接口规范**

**表0-7 服务器端启动模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **需要的服务（需接口）** | | |
| ServerRunner | 语法 | main(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 创建NetworkInit对象，并调用Init方法初始化相应的网络连接。之后进入用户登录界面 |
| NetworkInit | 语法 | Init() |
| 前置条件 | 存有网络配置文件 |
| 后置条件 | 在服务器端启动RMI注册服务，创建远程对象的实例和注册涉及的RMI服务。 |
| DatabaseInit | 语法 | Init() |
| 前置条件 | 存有数据库和文件 |
| 后置条件 | 初始化数据库 |

**表0-8 服务器端启动模块需要的接口**

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| Presentation.MainFrame.init() | 初始化窗口，显示登陆界面 |
| java.rmi.LocateRegistry.createRegistry() | 启动RMI注册服务 |
| Java.rmi.Naming.rebind() | 注册RMI服务 |

**4.2.2 启动模块设计原理**

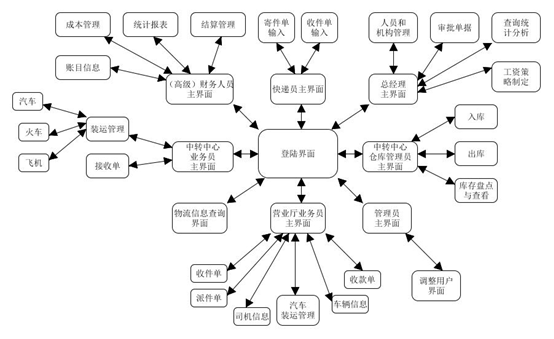
利用单一职责原则，对ServerRunner类将网络初始化和数据库初始化职责分给NetworkInit和DatabaseInit类。

**4.3用户界面层**

**4.3.1用户界面层的静态模型**

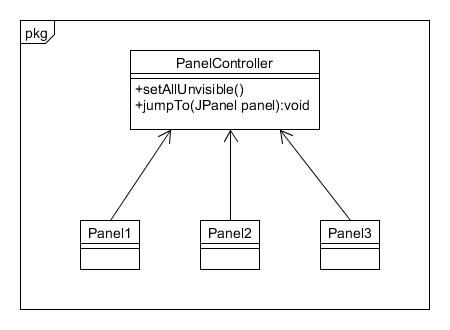
**一、整体结构**

根据要求，系统存在34个用户界面：登陆界面、快递员主界面、总经理主界面、中转中心仓库管理员主界面、管理员主界面、营业厅业务员主界面、物流信息查询界面、中转中心业务员主界面、账目信息界面、成本管理界面、统计报表界面、结算管理界面、寄件单输入界面、收件单输入界面、人员和机构管理界面、审批单据界面、查询统计分析界面、工资策略制定界面、入库界面、出库界面、库存盘点与查看界面、调整用户界面、收款单界面、车辆信息界面、汽车装运管理界面、司机信息界面、派件单界面、收件单界面、接收单界面、装运管理界面、汽车界面、火车界面、飞机界面。其中界面跳转如下：



**用户界面跳转**

用户界面的分解如下图：

**用户界面层的分解**

**二、用户界面层的职责**

如表所示为用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 界面Frame，负责界面的显示和界面的跳转 |
| Panel | Panel的抽象类 |
| LoginPanel | 登录界面Panel |
| \*Panel | 每一个界面都有一个Panel继承于Panel |

**三、用户界面层模块的接口规范**

用户界面层模块的接口规范如表所示

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | Init(args:String()) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame以及LoginPanel |

用户界面层需要的服务接口如表所示

|  |  |
| --- | --- |
| 服务名 | 服务 |
| bisinesslogicservice.LoginBLService | 登陆界面的业务逻辑接口 |
| Businesslogicservice.\*BLService | 每个界面的有一个相应的业务逻辑接口 |

**4.4 业务逻辑层的分解**

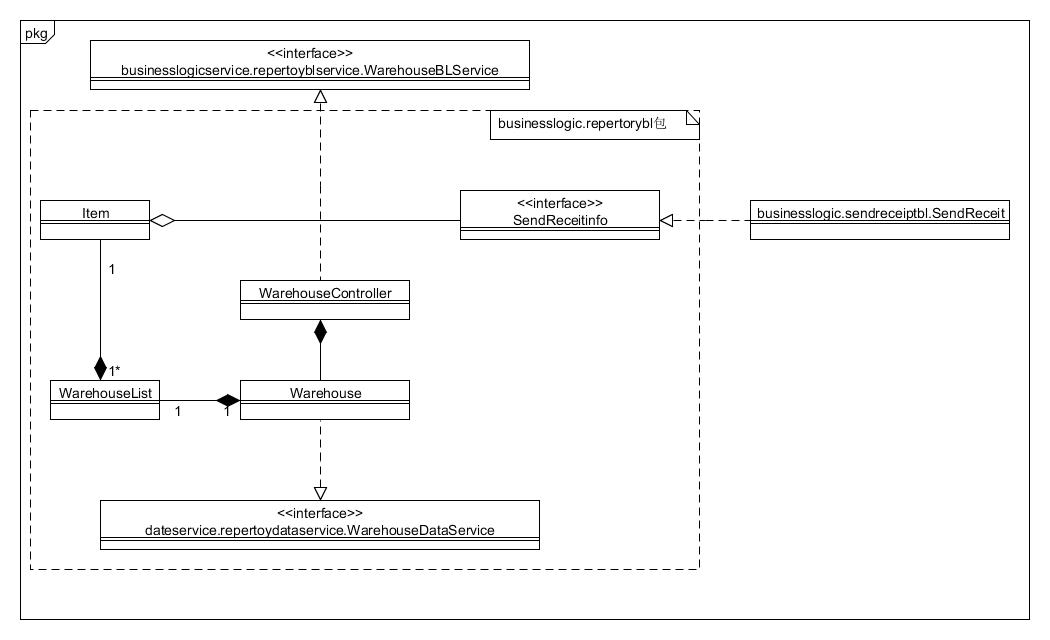
**4.4.1 业务逻辑层的静态模型**

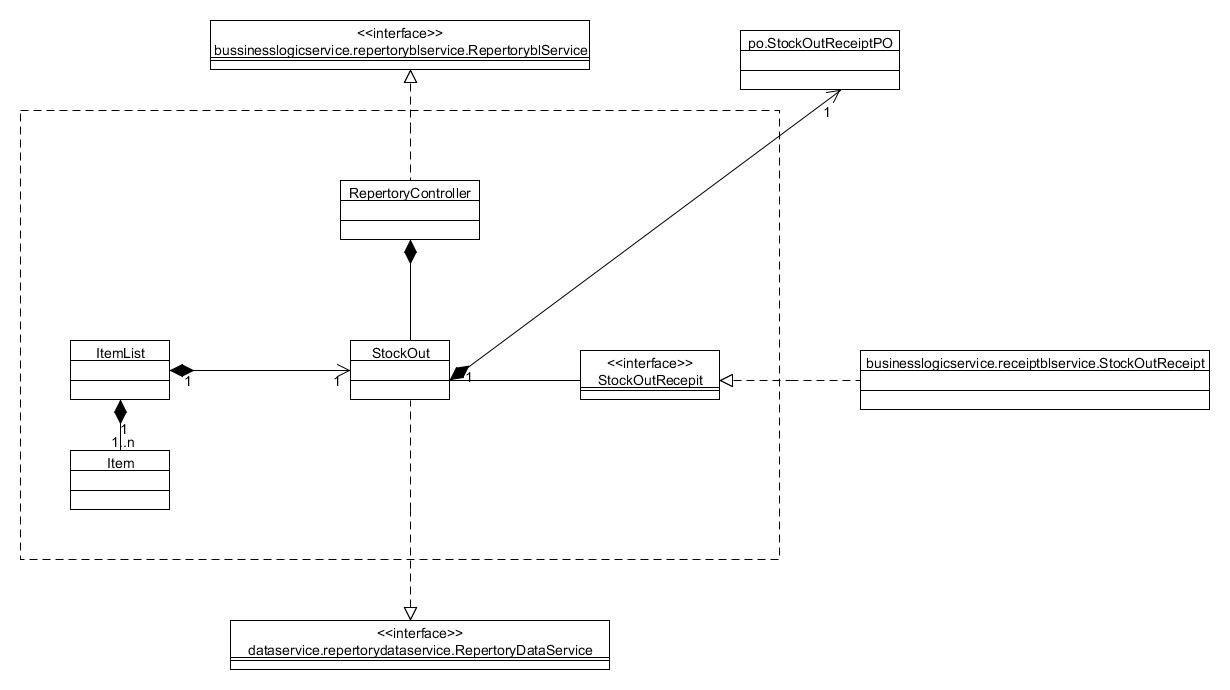
**一、整体结构**

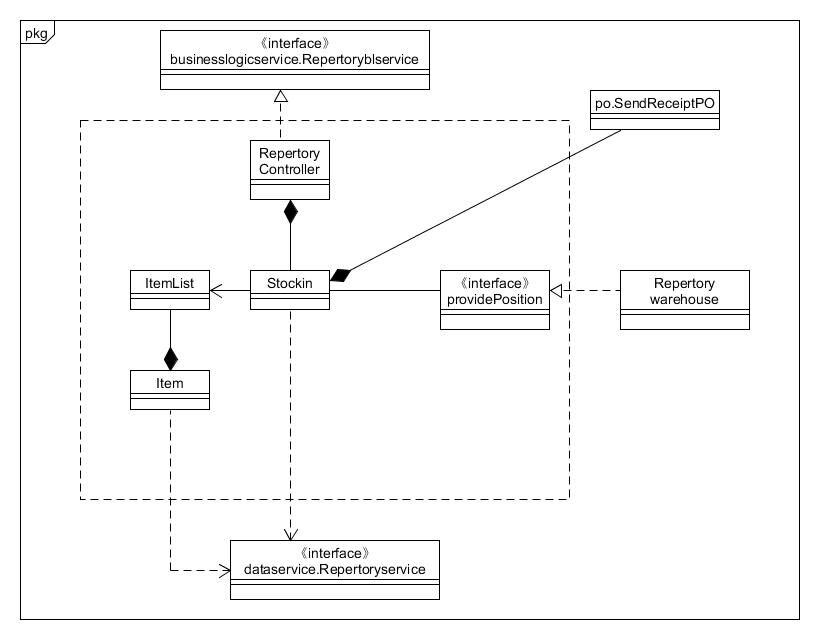
业务逻辑层分为两个子层，一个是controller层，一个是domain object层。Controlle主要负责向界面层提供服务，并利用具体的domain object层来实现各个服务。domain object层 是业务逻辑层中的各个领域模型。

**二、业务逻辑层模块的职责**

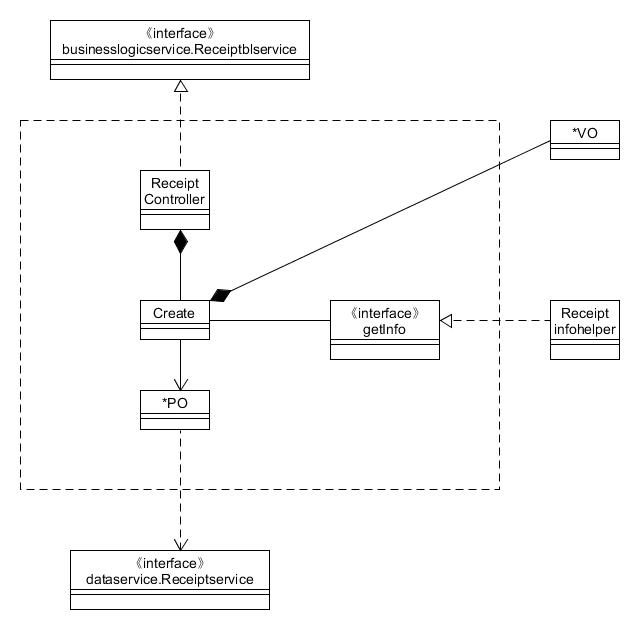
**1.Repertory模块**

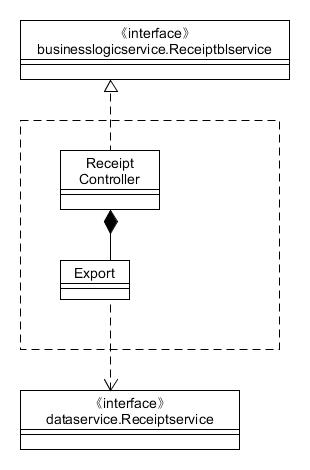
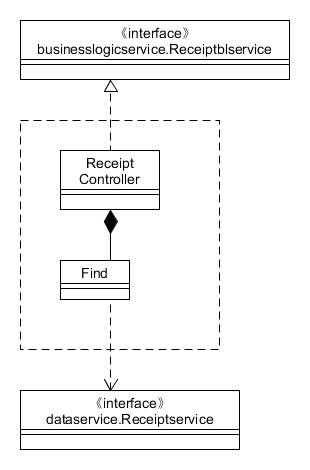
****

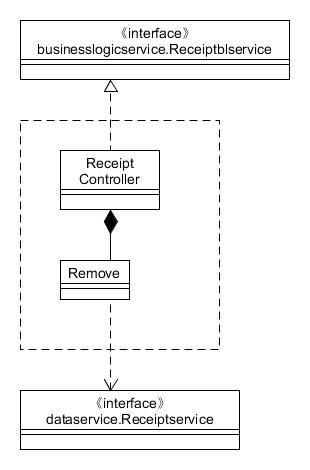
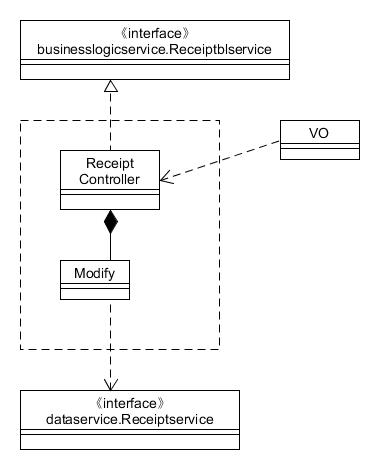
****

****

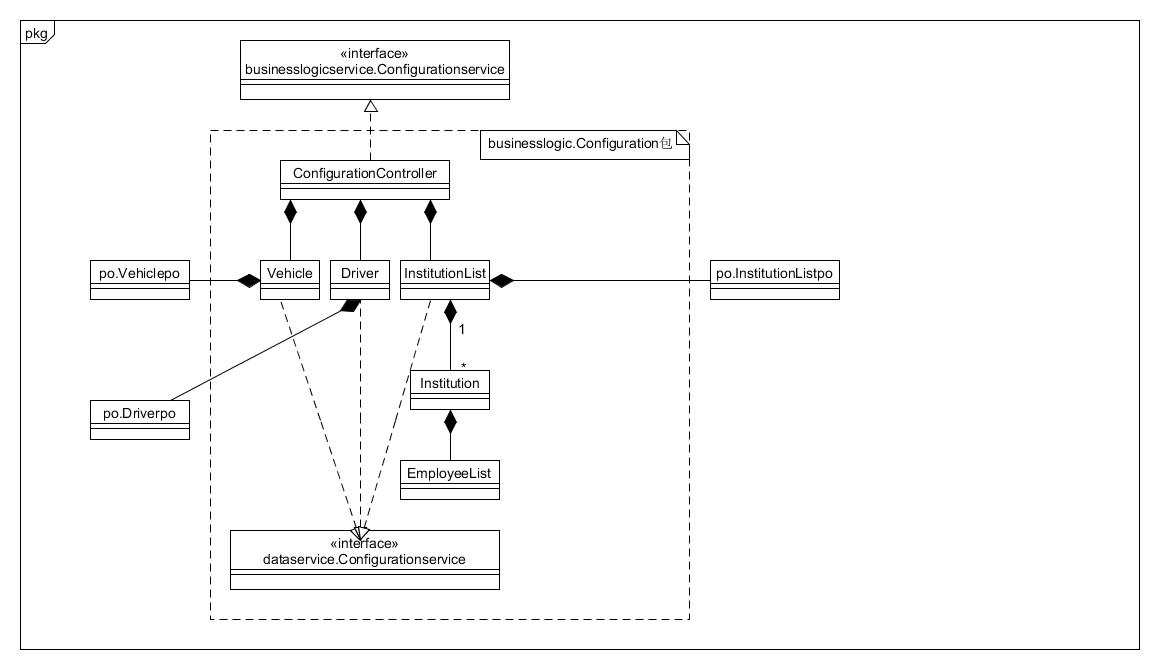
**repertory模块的各个类的设计**

****

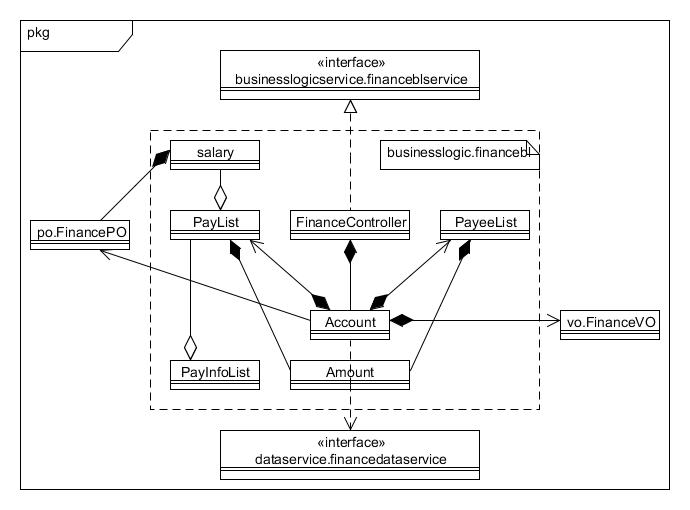
****

****

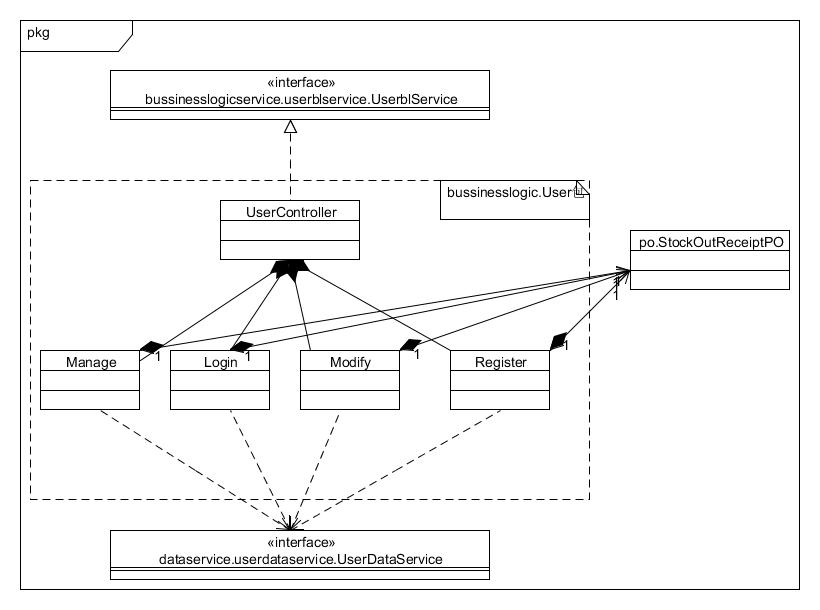
**Receipt模块的各个类的设计**

****

**Configuration模块的各个类的设计**

****

**Finance模块的各个类的设计**

****

**User模块的各个类的设计**

**repertory模块的职责**

|  |  |
| --- | --- |
| 类 | 职责 |
| RepertoryController | 负责实现仓库管理界面所需要的服务 |
| Warehouse | 仓库管理的领域模型对象，拥有一个仓库所持有的出/入库数量/金额，存储位置，各区快递的信息（快递编号、入库日期、目的地、区号、排号、架号、位号），可以帮助完成仓库管理界面所需要的服务 |
| StockOut | 出库管理的领域模型对象，拥有一次出库操作所持有的快递编号、出库日期、目的地、装运形式（火车、飞机、汽车）、中转单编号或者汽运编号 |
| Stockin | 入库管理的领域模型对象，拥有一次入库操作所持有的快递编号、入库日期、目的地、区号、排号、架号、位号 |

**2.User模块**

**User模块的职责**

|  |  |
| --- | --- |
| 类 | 职责 |
| UserController | 负责实现用户界面所需要的服务 |
| Login | 用户登录的领域模型对象，拥有一次登录操作所持有的ID和密码 |
| Register | 用户注册的领域模型对象，拥有一次注册操作所持有的ID、密码、申请的权限、姓名、所工作的机构地点、职位 |
| Modify | 修改用户信息的领域模型对象，拥有一次修改操作所持有的ID、密码、权限、姓名、所工作的机构地点、职位 |
| Manage | 用户管理的领域模型对象，拥有一次管理操作所持有的ID、权限、姓名 |

**3.Finance模块**

**Finance模块的职责**

|  |  |
| --- | --- |
| FinanceController | 负责实现财务管理界面所需要的服务 |
| Finance | 财务的领域模型对象 ，拥有一次财务所拥有的薪水和账户等信息，可以帮助完成财务界面所需要的服务 |

**4.Receipt模块**

**Receipt模块的职责**

|  |  |
| --- | --- |
| 类 | 职责 |
| ReceiptController | 负责实现各个涉及单据的增删改查输出所需要的服务 |
| Create | 增加单据的领域模型对象，可以接受包含相应单据信息的vo对象，创建相应的单据并存储 |
| Remove | 删除单据的领域模型对象，根据单据编号执行对相应单据的持久化对象的删除 |
| Modify | 修改单据的领域模型对象，根据单据编号执行对相应单据的持久化对象的修改 |
| Find | 查找单据的领域模型对象，根据单据编号查找相应单据的持久化对象 |
| infoHelper | 获取额外信息的领域模型对象，生成单据时，根据订单类型，获取生成单据所需要的额外信息 |
| Export | 输出单据的领域模型对象，将相应单据输出为excel格式 |

**5.Configuration模块**

**Configuration模块的职责**

|  |  |
| --- | --- |
| ConfigurationController | 负责配置管理（人员，机构，车辆）所需要的服务 |
| Vehicle | 车辆信息管理的领域模型对象，拥有一辆车所持有的车辆代号、发动机号、车辆号、底盘号、购买时间、服役时间，可以帮助 |
| Driver | 司机信息管理的领域模型对象，拥有一名司机所持有的司机编号、姓名、出生日期、身份证号、手机、车辆单位、性别、行驶证期限 |
| InstitutionList | 人员机构管理的领域模型对象，拥有所有机构的引用 |
| Institution | 人员机构管理的领域模型对象，拥有一个机构所持有的机构编号，员工信息 |
| EmployeeList | 人员机构管理的领域模型对象，拥有一个机构所有员工的信息 |

**三、业务逻辑层的接口规范**

**1.repertory模块**

**repertoryController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| RepertoryController.  getItemList | 语法 | Public ItemList get ItemList(expressType type,String destination) |
| 前置条件 | 存在符合对应类型、目的地的需要出库的货物 |
| 后置条件 | 调用StockOut领域对象的getItemList方法 |
| RepertoryController.  chooseItem | 语法 | Public void chooseItem () |
| 前置条件 | ItemList里包含该Item |
| 后置条件 | 调用StockOut领域对象的chooseItem方法 |
| RepertoryController.  addStockOutReceipt | 语法 | Public void addStockOutReceipt(ItemList list) |
| 前置条件 | 已经创建了一个完整的ItemList |
| 后置条件 | 调用StockOut领域对象的addStockOutReceipt方法 |
| RepertoryController.  addInfo | 语法 | Public Item addInfo(String ID) |
| 前置条件 | 无 |
| 后置条件 | 调用Stockin的addInfo方法 |
| RepertoryController.  Stockin | 语法 | Public void addStockinReceipt(ItemList list) |
| 前置条件 | 已经创建了一个包含完整信息的ItemList |
| 后置条件 | 调用Stockin的AddStockinReceipt方法 |
| RepertoryController.  inventoryAlarm | 语法 | Public Boolean inventoryAlarm() |
| 前置条件 | 已创建一个Warehouse领域对象 |
| 后置条件 | 调用Warehouse领域对象的inventoryAlarm方法 |
| RepertoryController.  ProvidePosition | 语法 | Public String providePosition(expressType type) |
| 前置条件 | 已创建一个Warehouse领域对象 |
| 后置条件 | 调用Warehouse领域对象的providePosition方法 |
| RepertoryController.  store | 语法 | Public void store(String position) |
| 前置条件 | 已创建一个Warehouse领域对象 |
| 后置条件 | 调用Warehouse领域对象的store方法 |
| RepertoryController.  exportItem | 语法 | Public Item[] exportItem(String date,String arrival,expressType type) |
| 前置条件 | 已创建一个Warehouse领域对象 |
| 后置条件 | 调用Warehouse领域对象的exportItem方法 |
| RepertoryController.  inventoryCheck | 语法 | Public inventorycheckVO inventoryCheck(String date) |
| 前置条件 | 已创建一个Warehouse领域对象 |
| 后置条件 | 调用Warehouse领域对象的inventoryCheck方法 |
| RepertoryController.  inventoryCount | 语法 | Public InventoryCountVO inventoryCount() |
| 前置条件 | 已创建一个Warehouse领域对象 |
| 后置条件 | 调用Warehouse领域对象的inventoryCount方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| Stockin.addInfo | 添加一个包含入库信息的Item |
| Stockin.addStockinReceipt | 存储入库单的持久化对象 |
| StockOut.getItemList | 得到一个包含符合条件的货物列表 |
| StockOut.chooseItem | 将该Item加入到出库单的ItemList中 |
| StockOut.addStockOutReceipt | 将该出库单保存到数据层中 |
| Warehouse.inventoryAlarm | 检查库存是否报警 |
| Warehouse.ProvidePosition | 提供货物的位置 |
| Warehouse.store | 在指定位置存放货物 |
| Warehouse.exportItem | 提供满足指定时间、目的地和类型的可出库的货物 |
| Warehouse.inventoryCheck | 进行库存查看 |
| Warehouse.inventoryCount | 进行库存盘点 |

**Warehouse模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Warehouse.inventoryAlarm | 语法 | Public Boolean inventoryAlarm() |
| 前置条件 | 启动一个入库回合 |
| 后置条件 | 在一个入库回合中，检查库存是否报警 |
| Warehouse.ProvidePosition | 语法 | Public String[] providePosition(expressType type) |
| 前置条件 | 启动一个入库回合 |
| 后置条件 | 在一个入库回合中，提供空位置 |
| Warehouse.store | 语法 | Public void store(String position) |
| 前置条件 | 启动一个入库回合 |
| 后置条件 | 在一个入库回合中，存放货物 |
| Warehouse.exportItem | 语法 | Public ItemList exportItem(String date,String arrival,expressType type) |
| 前置条件 | 启动一个出库回合 |
| 后置条件 | 在一个出库回合中，提供指定时间和目的地的可出库的货物 |
| Warehouse.inventoryCheck | 语法 | Public inventorycheckVO inventoryCheck(String date) |
| 前置条件 | 启动一个库存查看回合 |
| 后置条件 | 返回一个库存查看值对象，包括此时间段内的出/入库数量/金额，存储位置及库存数量的合计 |
| Warehouse.inventoryCount | 语法 | Public InventoryCountVO inventoryCount() |
| 前置条件 | 启动一个库存盘点回合 |
| 后置条件 | 返回一个库存盘点值对象，包括当天的各区快递的信息（快递编号、入库日期、目的地、区号、排号、架号、位号），并生成批次（日期）批号（序号） |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| RepertoryDataService.find(String id) | 根据ID进行查找单一持久化对象 |
| RepertoryDataService.insert(RepertoryPO po) | 插入单一持久化对象 |
| RepertoryDataService.delete(RepertoryPO po) | 删除单一持久化对象 |
| RepertoryDataService.update(RepertoryPO po) | 更新单一持久化对象 |
| DataFactory.getRepertoryDatabase | 得到Repertory数据库的引用 |

**StockOut的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockOut.getItemList | 语法 | Public ItemList get ItemList(expressType type,String destination) |
| 前置条件 | 存在符合对应类型、目的地的需要出库的货物 |
| 后置条件 | 得到一个包含符合条件的货物列表 |
| StockOut.chooseItem | 语法 | Public void chooseItem () |
| 前置条件 | ItemList里包含该Item |
| 后置条件 | 将该Item加入到出库单的ItemList中 |
| StockOut.addStockOutReceipt | 语法 | Public void addStockOutReceipt(ItemList list) |
| 前置条件 | 已经创建了一个完整的ItemList |
| 后置条件 | 将该出库单保存到数据层中 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| Warehouse.exportItem | 在一个出库回合中，提供类型、指定时间和目的地的可出库的货物 |

**Stockin的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Stockin.addInfo | 语法 | Public Item addInfo(String ID) |
| 前置条件 | 无 |
| 后置条件 | 得到一个货物包含将被存储在入库单中的信息的Item对象 |
| Stockin.addStockinReceipt | 语法 | Public void addStockinReceipt(ItemList list) |
| 前置条件 | 已经创建了一个包含完整信息的ItemList |
| 后置条件 | 将入库单存储（到数据库中） |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| Warehouse.providePosition | 根据货物的类型计算并返回可以存放的位置 |

**2.User模块**

**UserController接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserController.  getID | 语法 | Public String getID () |
| 前置条件 | 已创建Login领域对象 |
| 后置条件 | 调用Login领域对象的getID方法 |
| UserController.getPassword | 语法 | Public String getPassword () |
| 前置条件 | 已创建Login领域对象 |
| 后置条件 | 调用Login领域对象的getPassword方法 |
| UserController .verify | 语法 | Public boolean verify() |
| 前置条件 | 已创建Login领域对象 |
| 后置条件 | 调用Login领域对象的verify方法 |
| UserController.addUser | 语法 | Public boolean addUser () |
| 前置条件 | 已创建Register领域对象 |
| 后置条件 | 调用Register领域对象的addUser方法 |
| UserController.ruleApply | 语法 | Public void ruleApply () |
| 前置条件 | 已创建Register领域对象 |
| 后置条件 | 调用Register领域对象的ruleApply方法 |
| UserController.change | 语法 | Public boolean change (String ID) |
| 前置条件 | 已创建Modify领域对象 |
| 后置条件 | 调用Modify领域对象的change方法 |
| UserController.check | 语法 | Public boolean check (String ID) |
| 前置条件 | 已创建Manage领域对象 |
| 后置条件 | 调用Manage领域对象的check方法 |
| UserController.ruleChange | 语法 | Public boolean ruleChange (String ID) |
| 前置条件 | 已创建Manage领域对象 |
| 后置条件 | 调用Manage领域对象的ruleChange方法 |
| UserController.delete | 语法 | Public boolean delete (String ID) |
| 前置条件 | 已创建Manage领域对象 |
| 后置条件 | 调用Manage领域对象的delete方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| Login.getID | 获得ID |
| Login.getPassword | 获得密码 |
| Login.verify | 验证用户名和密码 |
| Register.addUser | 增加新用户，通过注册实现 |
| Register.ruleApply | 新用户申请权限 |
| Modify.change | 修改用户信息 |
| Manage.check | 管理员验证新用户的权限 |
| Manage.ruleChange | 管理员修改用户的权限 |
| Manage.delete | 管理员删除用户 |

**Login的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Login.getID | 语法 | Public String getID () |
| 前置条件 | 用户输入了ID |
| 后置条件 | 得到用户输入的ID |
| Login.getPassword | 语法 | Public String getPassword () |
| 前置条件 | 用户输入了密码 |
| 后置条件 | 得到用户输入的密码 |
| Login.verify | 语法 | Public boolean verify() |
| 前置条件 | 已经输入了ID和密码 |
| 后置条件 | 验证ID和密码是否匹配，成功则跳转到对应的页面，否则提示错误 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| UserDataService.find(String id) | 根据ID进行查找单一持久化对象 |

**Register的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Register.addUser | 语法 | Public boolean addUser () |
| 前置条件 | 必要信息（密码、姓名、申请的权限、所工作的机构地点、职位）填写完毕 |
| 后置条件 | 新增一个用户，并将用户的ID、密码、姓名、申请的权限、所工作的机构地点、职位保存到UserPO中 |
| Register.ruleApply | 语法 | Public void ruleApply () |
| 前置条件 |  |
| 后置条件 | 将申请的权限提交给管理员审核 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| UserDataService.insert(UserPO po) | 插入单一持久化对象 |

**Modify接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Modify.change | 语法 | Public boolean change (String ID) |
| 前置条件 | 该ID存在 |
| 后置条件 | 更新修改的信息，并保存 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| UserDataService.update(UserPO po) | 更新单一持久化对象 |
| UserDataService.find(String id) | 根据ID进行查找单一持久化对象 |

**Manage接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Manage.check | 语法 | Public boolean check (String ID) |
| 前置条件 | 该ID存在 |
| 后置条件 | 验证该ID对应的用户的权限，改变该ID的权限 |
| Manage.ruleChange | 语法 | Public boolean ruleChange (String ID) |
| 前置条件 | 该ID存在 |
| 后置条件 | 修改该ID对应的用户的权限，并保存 |
| Manage.delete | 语法 | Public boolean delete (String ID) |
| 前置条件 | 该ID存在 |
| 后置条件 | 删除该ID对应的用户，并删除对应的PO |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| UserDataService.update(UserPO po) | 更新单一持久化对象 |
| UserDataService.find(String id) | 根据ID进行查找单一持久化对象 |
| UserDataService.delete(UserPO po) | 删除单一持久化对象 |

**3.Finance模块**

**FinanceController接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FinanceController.getSalary | 语法 | Public int getSalary(String id) |
| 前置条件 | 已创建一个Finance领域对象，并且输入符合规则 |
| 后置条件 | 调用Finance领域对象的getSalary方法 |
| FinanceController.getAccount | 语法 | Public ResultMessage getAccount(String id) |
|  | 前置条件 | 已创建一个Finance领域对象，并且输入符合规则 |
| 后置条件 | 调用Finance领域对象的getAccount方法 |
| FinanceController.initialization | 语法 | Public void initialization(String name, String date, String people) |
| 前置条件 | 已创建一个Finance领域对象 |
| 后置条件 | 调用Finance领域对象的initialization方法 |
| FinanceController.getInfo | 语法 | Public FinancePO initialization(String date, String ID) |
| 前置条件 | 已创建一个Finance领域对象 |
| 后置条件 | 调用Finance领域对象的getInfo方法 |
| FinanceController.amount | 语法 | Public void amout() |
| 前置条件 | 已创建一个Finance领域对象 |
| 后置条件 | 调用Finance领域对象的amount方法 |
| FinanceController.costManage | 语法 | Public FinancePO costManage(String name, String date, String ID) |
| 前置条件 | 已创建一个Finance领域对象，输入符合规范 |
| 后置条件 | 调用Finance领域对象的costManage方法 |
| FinanceController.getcostControl | 语法 | Public FinancePO costControl() |
| 前置条件 | 已创建一个Finance领域对象 |
| 后置条件 | 调用Finance领域对象的costControl方法 |
| FinanceController.getOperationState | 语法 | Public FinancePO getOperationState(String time1, String time2) |
| 前置条件 | 已创建一个Finance领域对象，输入符合规范 |
| 后置条件 | 调用Finance领域对象的getOperationState方法 |

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| Finance.getSalary | 返回一个工资数值 |
| Finance.getAccount | 返回一个账户信息 |
| Finance.initialization | 新增一个账目信息 |
| Finance.getInfo | 返回一个收款单信息 |
| Finance.amount | 新增一个合计收款信息表单 |
| Finance.costManage | 返回一个付款单信息 |
| Finance.getcostControl | 返回一个成本收益信息 |
| Finance.getOperationState | 返回一个经营状况信息 |

**Finance接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的接口（供接口）** | | |
| Finance.getSalary | 语法 | Public int stockin(long id) |
| 前置条件 | Salary经过工资策略中的计算得出，ID符合规范 |
| 后置条件 | 返回该账户的工资 |
| Finance.getAccount | 语法 | Public ResultMessage getAccount(String id) |
| 前置条件 | 此登录人员拥有最高权限 |
| 后置条件 | 返回相关账户的姓名与账号 |
| Finance.initialization | 语法 | Public void initialization() |
| 前置条件 | 此登录人员拥有最高权限 |
| 后置条件 | 完成一个期初建账操作 |
| Finance.getInfo | 语法 | Public FinancePO getInfo(String date, String ID) |
| 前置条件 | 进入一个结算管理回合 |
| 后置条件 | 从FinancePO中根据输入查询对应的持久化对象 |
| Finance.amount | 语法 | Public void amout() |
| 前置条件 | 已进行收款单的查询 |
| 后置条件 | 根据收款单生成合计信息表单并将其插入FinancePO中 |
| Finance.costManage | 语法 | Public FinancePO costManage(String name, String date, String ID) |
| 前置条件 | 进入一个成本管理回合，输入符合规范 |
| 后置条件 | 根据输入返回一个ReiceptPO |
| Finance.getcostControl | 语法 | Public FinancePO costControl() |
| 前置条件 | 进入一个成本管理回合 |
| 后置条件 | 根据ReiceptPO计算出当前时间与上次生成的时间段内的总收入，总支出与总利润并存入数据库 |
| Finance.getOperationState | 语法 | Public FinancePO getOperationState(String time1, String time2) |
| 前置条件 | 进入一个统计报表回合 |
| 后置条件 | 根据输入的时间显示FinancePO对象，并符合可以导出的前置条件 |

|  |  |
| --- | --- |
| **需要的接口（需接口）** | |
| FinanceDataService.find(int id) | 根据ID进行查找单一持久化对象 |
| FinanceDataService.insert(FinancePO po) | 插入单一持久化对象 |
| FinanceDataService.delete(FinancePO po) | 删除单一持久化对象 |
| FinanceDataService.update(FinancePO po) | 更新单一持久化对象 |
| FinanceDataService.modify(FinancePO po) | 修改单一持久化对象 |
| FinanceDataService.finds(String field, int value) | 根据字段和值进行模糊搜索多个对象 |

**4.Receipt模块**

**ReceiptController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceiptController.add | 语法 | Public boolean add(\*PO receiptPO) |
| 前置条件 | 无 |
| 后置条件 | 调用Create的add方法 |
| ReceiptController.delete | 语法 | Public boolean delete(String ID) |
| 前置条件 | 无 |
| 后置条件 | 调用Remove的delete方法 |
| ReceiptController.modify | 语法 | Public boolean modify(String ID) |
| 前置条件 | 无 |
| 后置条件 | 调用Modify的modify方法 |
| ReceiptController.find | 语法 | Public boolean find(String ID) |
| 前置条件 | 无 |
| 后置条件 | 调用Find的find方法 |
| ReceiptController.getInfo\* | 语法 | \*根据单据类型调用不同的getinfo方法，参见下表infohelper的接口规范 |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| Creat.add | 存储单据的持久化对象 |
| Remove.delete | 删除单据的持久化对象 |
| Modify.modify | 修改单据的持久化对象 |
| Find.find | 查找单据的信息 |
| infohelper.getInfo | 获得创建单据持久化对象的其他信息 |

**Infohelper的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| infohelper.getSendReceiptInfo | 语法 | Public SendReceiptPO getSendReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.getAcceptReceiptInfo | 语法 | Public AcceptReceiptPO getAcceptReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.getCollectionReceiptInfo | 语法 | Public CollectionReceiptPO getCollectionReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.getDeliveryReceiptInfo | 语法 | Public DeliveryReceiptPO getDeliveryReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.getEarningReceiptInfo | 语法 | Public EarningReceiptPO getEarningReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.getloadingReceiptInfo | 语法 | Public loadingReceiptPO infohelper.getloadingReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.gettransitReceiptInfo | 语法 | Public transitReceipPO gettransitReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.getPaymentReceiptInfo | 语法 | Public PaymentReceiptPO getPaymentReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.getImportReceiptInfo | 语法 | Public ImpotReceiptPO getImportReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |
| infohelper.getExportReceiptInfo | 语法 | Public ExportReceiptPO getExportReceiptInfo() |
| 前置条件 | 无 |
| 后置条件 | 生成一个包含所需要的其余信息的相应PO对象 |

**Create的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Create.add | 语法 | Public boolean add(\*PO ReceiptPO) |
| 前置条件 | 无 |
| 后置条件 | 存储相应单据的持久化对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| ReceiptDataService.insert(ReceiptPO po) | 插入单一持久化对象 |
| DataFactory.getReceiptDatabase | 得到Repertory数据库的引用 |

**Remove的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Create.delete | 语法 | Public boolean delete(String ID) |
| 前置条件 | 无 |
| 后置条件 | 删除相应单据的持久化对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| ReceiptDataService.delete(ReceiptPO po) | 删除单一持久化对象 |
| DataFactory.getReceiptDatabase | 得到Repertory数据库的引用 |

**Modify的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Modify.modify | 语法 | Public boolean modify(String ID) |
| 前置条件 | 无 |
| 后置条件 | 修改相应单据的持久化对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| ReceiptDataService.update(ReceiptPO po) | 更新单一持久化对象 |
| DataFactory.getReceiptDatabase | 得到Repertory数据库的引用 |

**Find的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Find.find | 语法 | Public boolean find(String ID) |
| 前置条件 | 无 |
| 后置条件 | 修改相应单据的持久化对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| ReceiptDataService.find(int id) | 根据ID进行查找单一持久化对象 |
| DataFactory.getReceiptDatabase | 得到Repertory数据库的引用 |

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Export.export | 语法 | Public boolean export(String ID) |
| 前置条件 | 无 |
| 后置条件 | 输出相应单据的持久化对象 |

**5.Configuration模块**

**ConfigurationController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ConfigurationController.  addVehicleInfo | 语法 | Public void addVehicleInfo () |
| 前置条件 | 存在新增的车辆 |
| 后置条件 | 调用vehicle领域对象的addVehicleInfo方法 |
| ConfigurationController.  addDriverInfo | 语法 | Public void addDriverInfo () |
| 前置条件 | 存在新增的司机 |
| 后置条件 | 调用Driver领域对象的addDriverInfo方法 |
| ConfigurationController.  queryInstititionInfo | 语法 | Public InstitutionListVO queryInstititionInfo () |
| 前置条件 | 已经创建了一个完整的InstitutionList |
| 后置条件 | 调用InstitutionList领域对象的queryInstititionInfo 方法 |
| ConfigurationController.  addInstititionInfo | 语法 | Public void addInstititionInfo () |
| 前置条件 | 已经创建了一个完整的InstitutionList |
| 后置条件 | 调用InstitutionList的addInstititionInfo方法 |
| RepertoryController.  addIEmployeeInfo | 语法 | Public void addIEmployeeInfo () |
| 前置条件 | 已经创建了一个包含完整信息的EmployeeList |
| 后置条件 | 调用EmployeeList的addIEmployeeInfo t方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| vehicle.addVehicleInfo | 添加一个新的车辆信息 |
| Driver.addDriverInfo | 添加一个新的司机信息 |
| InstitutionList.queryInstititionInfo | 返回一个机构列表 |
| InstitutionList .addInstititionInfo | 添加一个新的机构信息 |
| EmployeeList.addIEmployeeInfo | 添加一个新的员工信息 |

**Vehicle的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Vehicle. addVehicleInfo | 语法 | Public Boolean addVehicleInfo() |
| 前置条件 | 启动一个新增车辆信息回合 |
| 后置条件 | 在一个新增车辆信息回合中，生成新的车辆信息 |

**Driver的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Driver.addDriverInfo | 语法 | Public Boolean addDriverInfo () |
| 前置条件 | 启动一个新增司机信息回合 |
| 后置条件 | 在一个新增司机信息回合中，生成新的司机信息 |

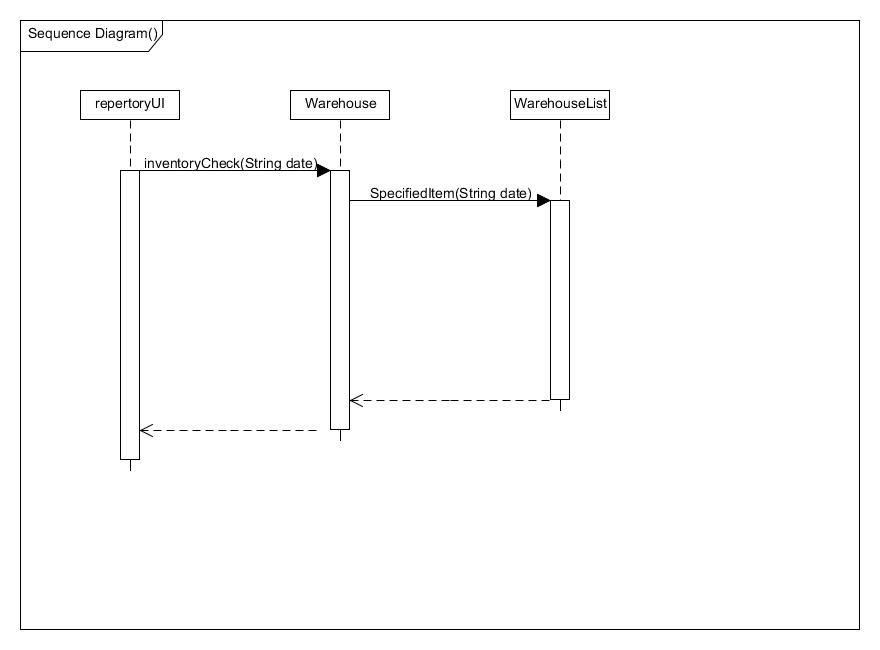
**InstitutionList的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| InstitutionList.queryInstititionInfo | 语法 | Public InstitutionListVO queryInstititionInfo () |
| 前置条件 |  |
| 后置条件 | 得到一个机构列表 |
| InstitutionList.addInstititionInfo | 语法 | Public void addInstititionInfo (Institution) |
| 前置条件 |  |
| 后置条件 | 将该Institution加入到配置管理的InstitutionList中 |
| EmployeeList. addIEmployeeInfo | 语法 | Public void addIEmployeeInfo() |
| 前置条件 |  |
| 后置条件 | 将该员工信息加入到指定的EmployeeList中 |

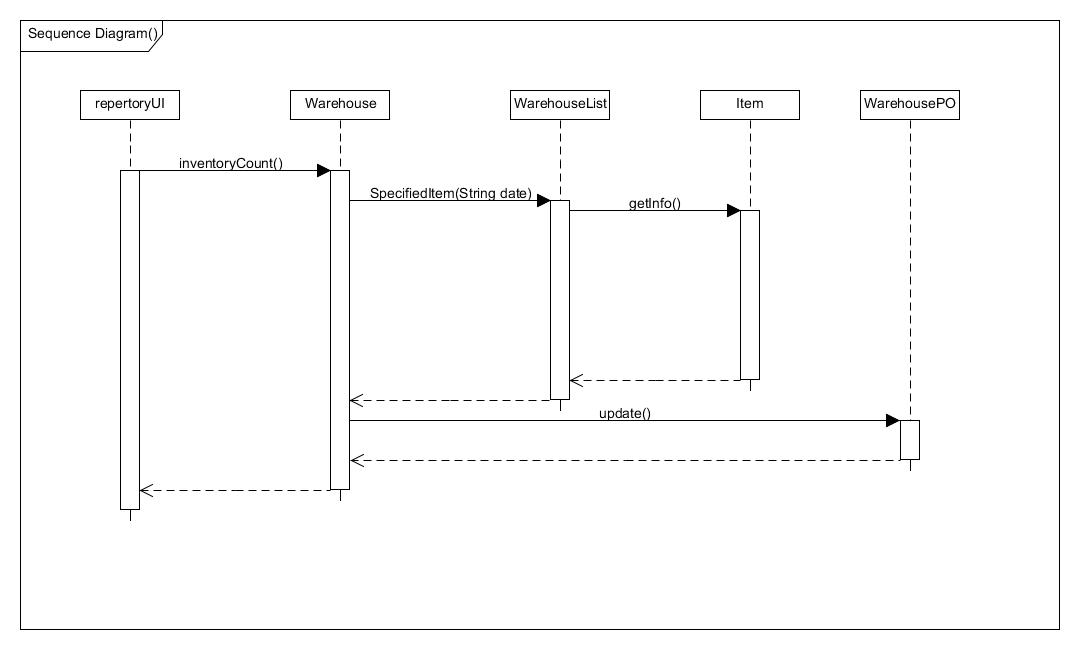
|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| ConfigurationDataService.find(int id) | 根据ID进行查找单一持久化对象 |
| ConfigurationDataService.insert(ConfigurationPO po) | 插入单一持久化对象 |
| ConfigurationDataService.delete(ConfigurationPO po) | 删除单一持久化对象 |
| ConfigurationDataService.update(ConfigurationPO po) | 更新单一持久化对象 |
| DataFactory.getConfigurationDatabase | 得到Repertory数据库的引用 |

**4.4.2业务逻辑层的动态模型**

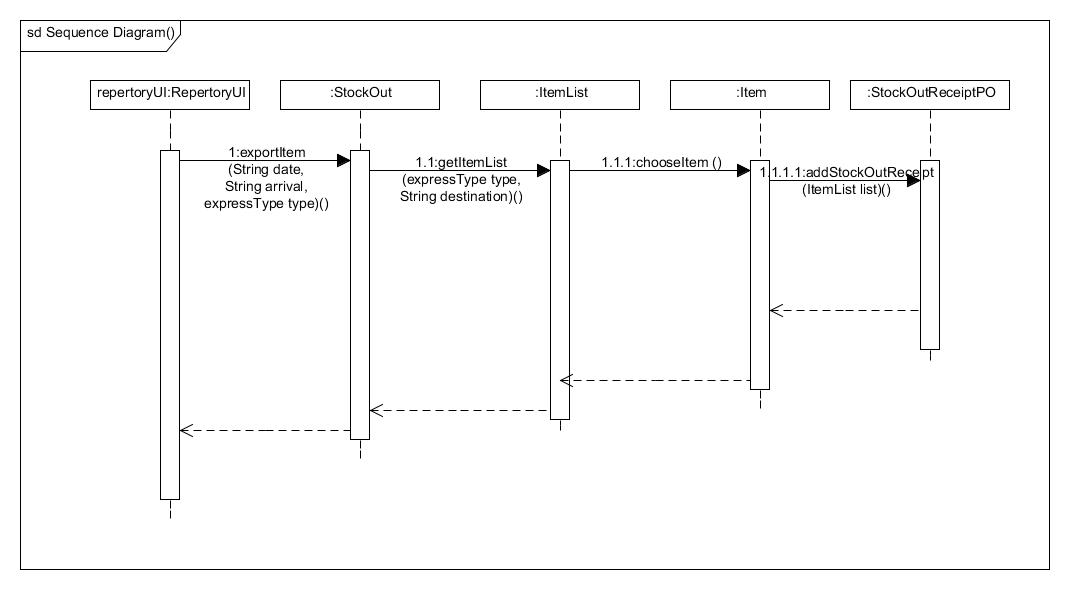
**1.repertory模块**

****

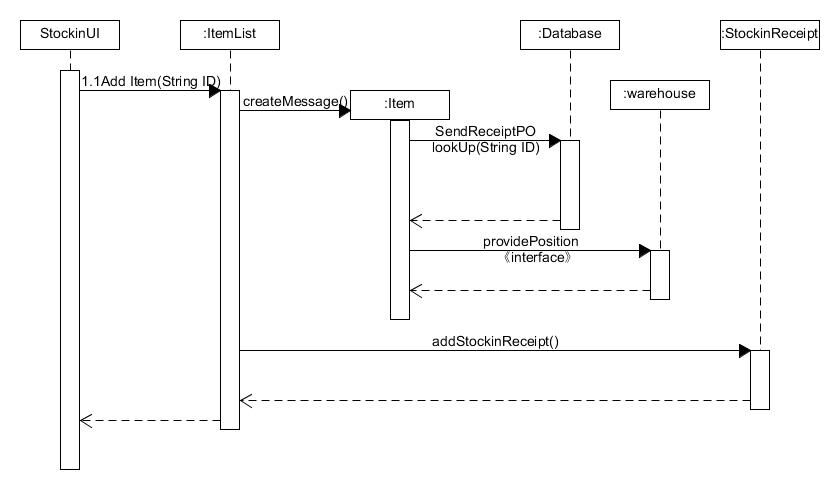
**库存查看顺序图**

****

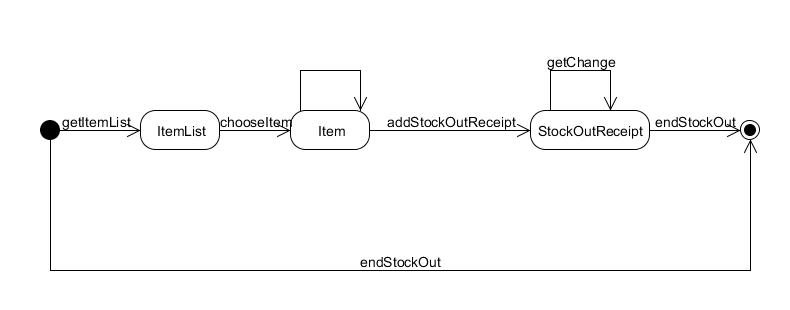
**库存盘点顺序图**

****

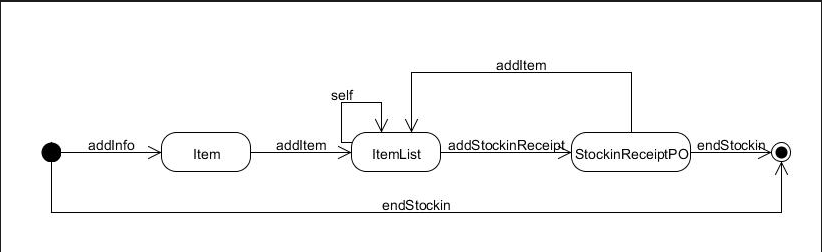
**出库顺序图**

****

**入库顺序图**

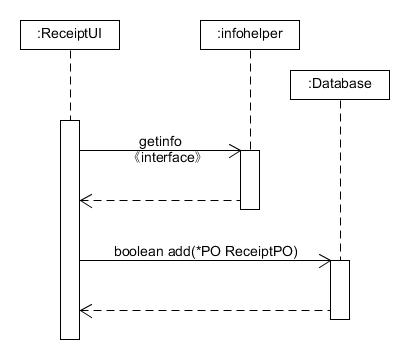
****

**Stockout状态图**

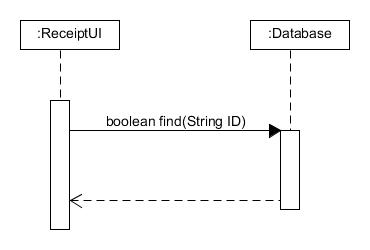
****

**Stockin状态图**

**2.receipt模块**

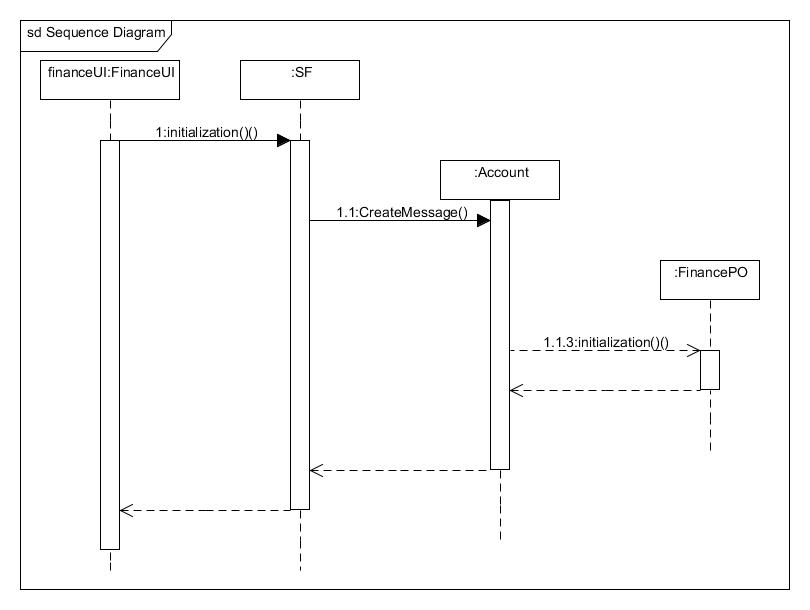
****

**Create顺序图**

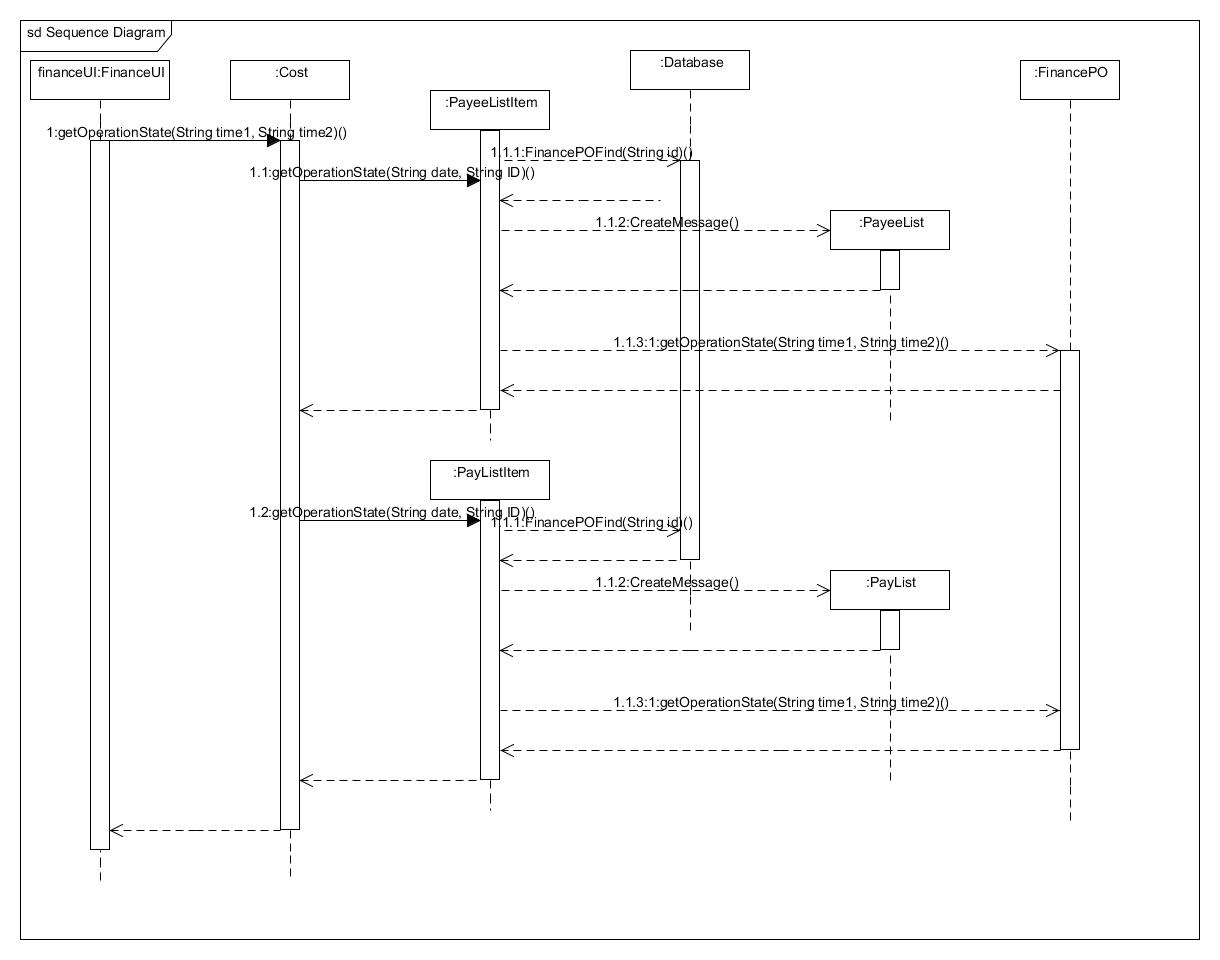
****

**Find顺序图**

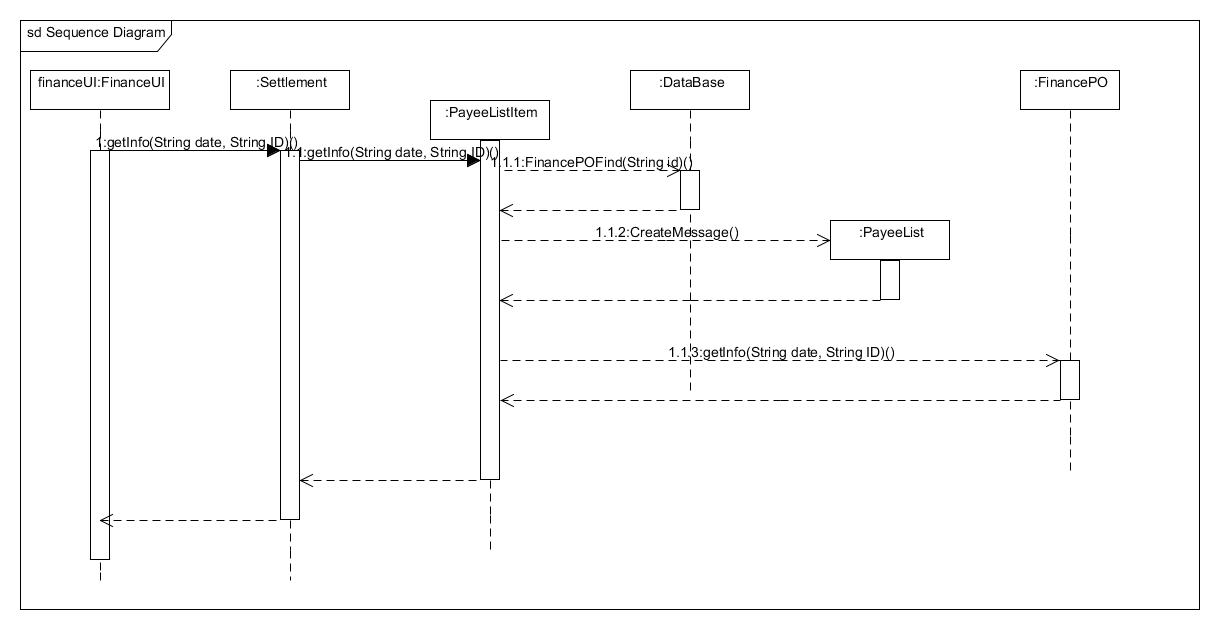
**3.Finance模块**

****

**期初建账顺序图**

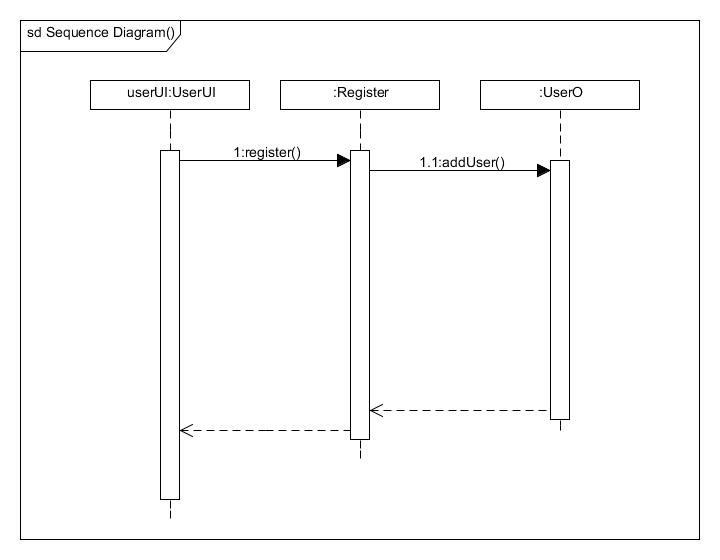
****

**统计报表顺序图**

****

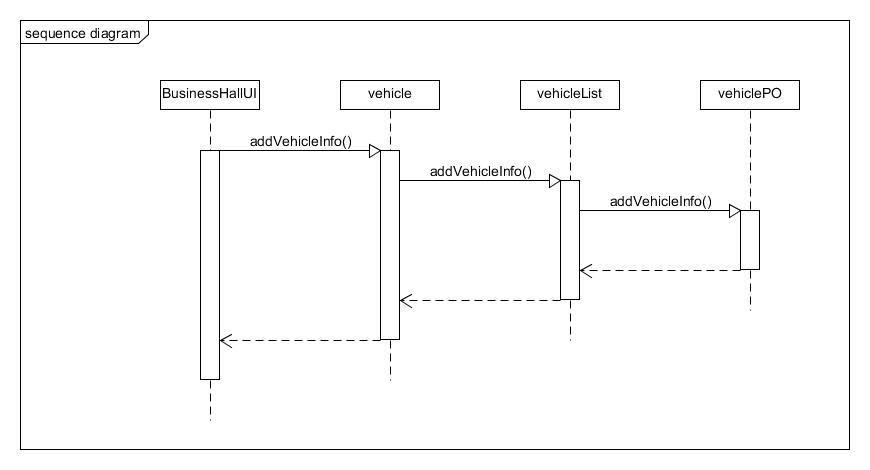
**结算管理顺序图**

**4.User模块**

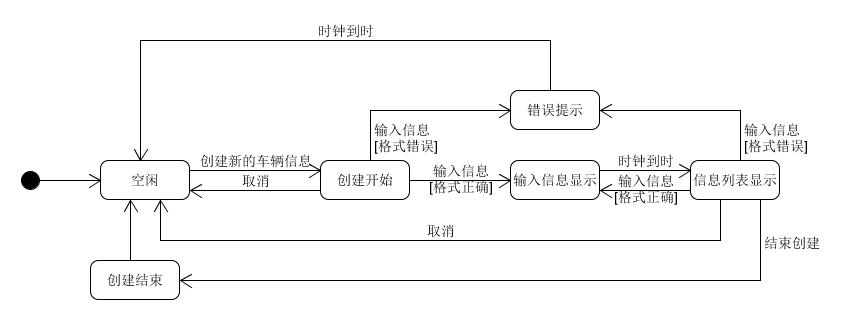
****

**注册顺序图**

**5.Configuration模块**

****

**新增车辆信息顺序图**

****

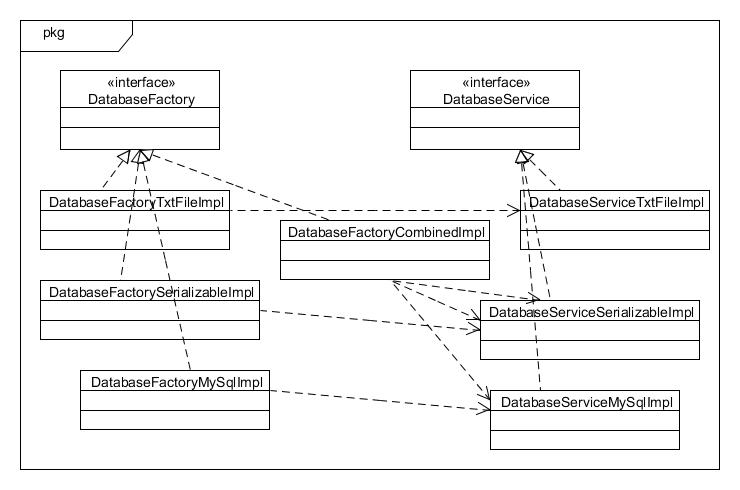
**新增车辆信息状态图**

**4.4.3业务逻辑层的设计原理**

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。

**4.5数据层的分解**

数据层主要给数据映射层服务。对于持久化数据的集体载入、保存、增、删、改、查。这些服务由DataService接口提供。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，而且可能出现混搭的可能，所以，我们在数据层的实现上采用了DataFactory的方式，具体如图所示。



**4.5.1数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| DatabaseFactory | 持久化数据库工厂，提供获得不同数据的工厂方法 |
| DatabaseService | 持久化数据库的接口，提供集体载入、保存、增、删、改、查服务 |
| DatabaseFactoryTxtFileImpl | 基于Txt文件的持久化数据工厂的实现 |
| DatabaseFactorySerializableFileImpl | 基于序列化文件的持久化数据工厂的实现 |
| DatabaseFactoryMySqlImpl | 基于数据库文件的持久化数据工厂的实现 |
| DatabaseFactoryCombinedImpl | 基于混合式的持久化数据工厂的实现 |
| DatabaseServiceTxtFileImpl | 基于Txt文件的持久化数据库的接口，提供集体载入、保存、增、删、改、查服务 |
| DatabaseServiceSerializableFileImpl | 基于序列化文件的持久化数据库的接口，提供集体载入、保存、增、删、改、查服务 |
| DatabaseServiceMySqlImpl | 基于数据库文件的持久化数据库的接口，提供集体载入、保存、增、删、改、查服务 |

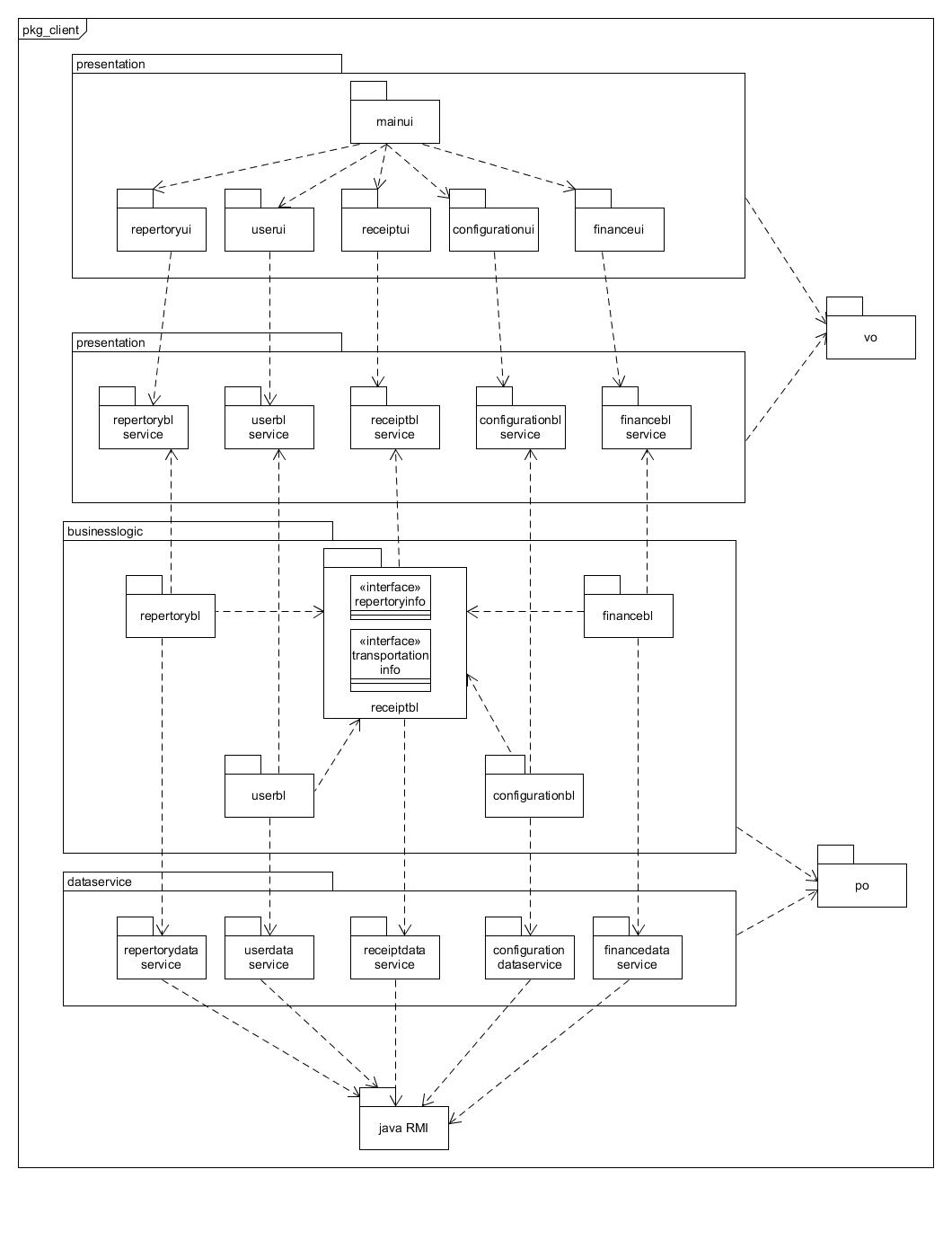
**4.5.2数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DatabaseFactory.getUserDatabase | 语法 | public UserPO getUserDatabase (DataType type)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 得到User数据库的服务的引用 |
| DataService.insert | 语法 | public void insert(PO po) throws RemoteException |
| 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 |
| DataService.delete | 语法 | public void delete(PO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除一个po |
| DataService.update | 语法 | public void update(PO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| DataService.init | 语法 | public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| DataService.finish | 语法 | public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |
| DataService.load | 语法 | public void load()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 整体读取所有po |
| DataService.save | 语法 | public void save() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 整体保存所有po |

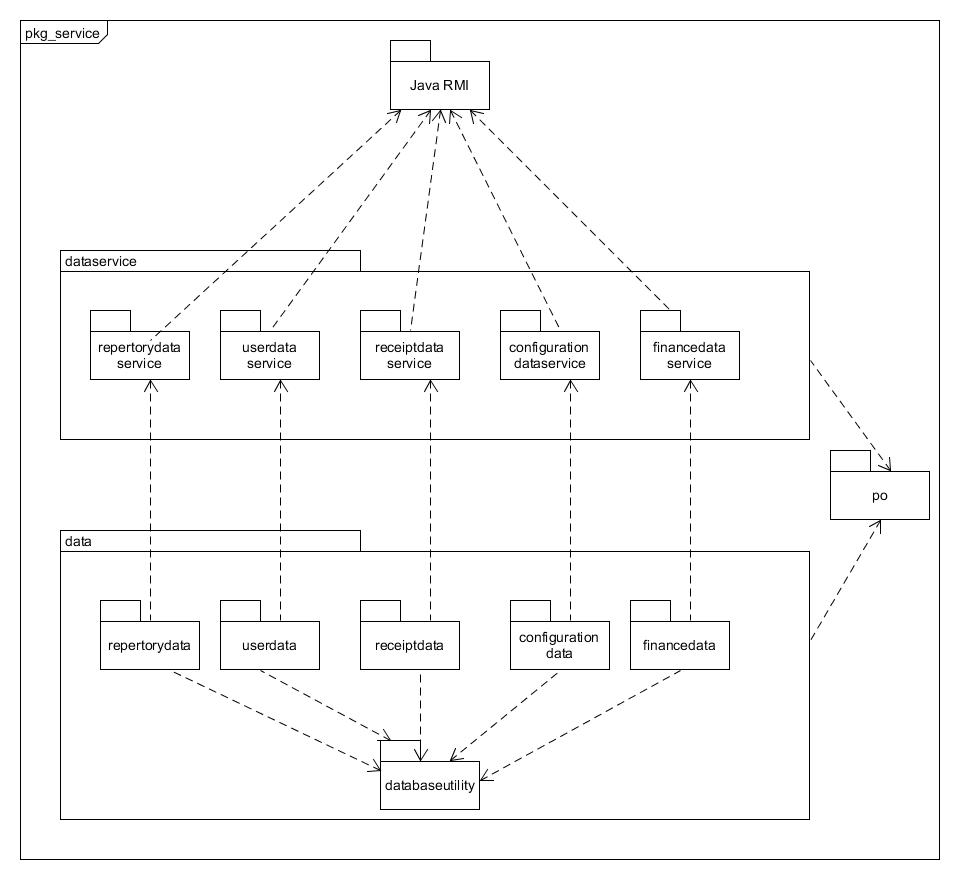
**4.5.** **3 数据层的设计原理**

利用抽象工厂模式对数据库的实现提供很好的可修改性。

**5.依赖视角**



**客户端包图**

****

**服务器包图**