**快递物流系统**

**详细设计描述文档**

学 院：南京大学软件学院

成 员：潘凌伟 申彬 谭昕玥 杨雁飞

完成日期：2015年11月5日

目录

[更新历史 6](#_Toc435479058)

[1. 引言 8](#_Toc435479059)

[1.1编制目的 8](#_Toc435479060)

[1.2词汇表 8](#_Toc435479061)

[1.3参考资料 8](#_Toc435479062)

[2. 中层设计 9](#_Toc435479063)

[2.1 expressorderbl模块的静态结构和动态行为 9](#_Toc435479064)

[2.1.1 expressorderbl模块局部模块的职责 9](#_Toc435479065)

[2.1.2 expressorderbl模块局部模块的接口规范 11](#_Toc435479066)

[2.1.3 expressorderbl模块的行为 15](#_Toc435479067)

[2.2 transmanagebl模块的静态结构和动态行为 17](#_Toc435479068)

[2.2.1 transmanagebl模块局部模块的职责 17](#_Toc435479069)

[2.2.2 transmanagebl模块局部模块的接口规范 19](#_Toc435479070)

[2.2.3 transmanagebl模块的行为 24](#_Toc435479071)

[2.3 orderbl模块的静态结构和动态行为 26](#_Toc435479072)

[2.1.1 orderbl模块局部模块的职责 26](#_Toc435479073)

[2.3.2 orderbl模块局部模块的接口规范 28](#_Toc435479074)

[2.3.3 orderbl模块的行为 36](#_Toc435479075)

[2.4 inventorybl模块的静态结构和动态行为 39](#_Toc435479076)

[2.4.1 inventorybl模块局部模块的职责 39](#_Toc435479077)

[2.4.2 inventorybl模块局部模块的接口规范 41](#_Toc435479078)

[2.4.3 inventorybl模块的行为 49](#_Toc435479079)

[2.5 receiptbl模块的静态结构和动态行为 52](#_Toc435479080)

[2.5.1 receiptbl模块局部模块的职责 52](#_Toc435479081)

[2.5.2 receiptbl模块局部模块的接口规范 53](#_Toc435479082)

[3.5.2 receiptbl模块的行为 59](#_Toc435479083)

[2.6 accountbl模块的静态结构和动态行为 60](#_Toc435479084)

[2.6.1 accountbl模块局部模块的职责 60](#_Toc435479085)

[2.6.2 accountbl模块局部模块的接口规范 61](#_Toc435479086)

[2.6.3 accountbl模块的行为 64](#_Toc435479087)

[2.7 tablebl模块的静态结构和动态行为 65](#_Toc435479088)

[2.7.1 tablebl模块局部模块的职责 65](#_Toc435479089)

[2.7.2 tablebl模块局部模块的接口规范 66](#_Toc435479090)

[2.7.3 tablebl模块的行为 69](#_Toc435479091)

[2.8 initialbl模块的静态结构和动态行为 70](#_Toc435479092)

[2.8.1 initialbl模块局部模块的职责 70](#_Toc435479093)

[2.8.2 initialbl模块局部模块的接口规范 70](#_Toc435479094)

[2.8.3 initialbl模块的行为 74](#_Toc435479095)

[2.9 excelbl模块的静态结构和动态行为 75](#_Toc435479096)

[2.9.1 excelbl模块局部模块的职责 75](#_Toc435479097)

[2.9.2 excelbl模块局部模块的接口规范 76](#_Toc435479098)

[2.9.3 excelbl模块的行为 76](#_Toc435479099)

[2.10 loginbl模块的静态结构和动态行为 77](#_Toc435479100)

[2.10.1 loginbl模块局部模块的职责 77](#_Toc435479101)

[2.10.2 loginbl模块局部模块的接口规范 78](#_Toc435479102)

[2.10.3 loginbl模块的行为 78](#_Toc435479103)

[2.11 managebl模块的静态结构和动态行为 79](#_Toc435479104)

[2.11.1 managebl模块局部模块的职责 79](#_Toc435479105)

[2.11.2 managebl模块局部模块的接口规范 81](#_Toc435479106)

[2.11.3 managebl模块的行为 84](#_Toc435479107)

[2.12 strategybl模块的静态结构和动态行为 85](#_Toc435479108)

[2.12.1 strategybl模块局部模块的职责 85](#_Toc435479109)

[2.12.2 strategybl模块局部模块的接口规范 87](#_Toc435479110)

[2.12.3 strategybl模块的行为 92](#_Toc435479111)

[2.13 approvalbl模块的静态结构和动态行为 96](#_Toc435479112)

[2.13.1 approvalbl模块局部模块的职责 96](#_Toc435479113)

[2.13.2 approvalbl模块局部模块的接口规范 97](#_Toc435479114)

[2.13.3 approvalbl模块的行为 100](#_Toc435479115)

[2.14 userbl模块的静态结构和动态行为 101](#_Toc435479116)

[2.14.1 userbl模块局部模块的职责 101](#_Toc435479117)

[2.14.2 userbl模块局部模块的接口规范 102](#_Toc435479118)

[2.14.3 userbl模块的行为 105](#_Toc435479119)

[2.15 用户界面层的行为 107](#_Toc435479120)

[2.15.1 用户界面层的跳转 107](#_Toc435479121)

[2.15.2用户界面层的跳转顺序图示例 107](#_Toc435479122)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 谭昕玥 | 2015-11-1 | 最初草稿（创建文档模板） | V1.0 |
| 全体组员 | 2015-11-5 | 初步文档 | V1.1 |
| 全体成员 | 2015-11-16 | 最终版 | V2.4 |

# 引言

## 1.1编制目的

本报告详细完成对快递物流系统的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| \_ui | 表示某展示层 |  |
| \_bl | 表示某逻辑层 |  |
| \_data | 表示某数据层 |  |

## 1.3参考资料

1. IEEE std 1471-2000
2. 丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版2012：134—182

# 中层设计

## 2.1 expressorderbl模块的静态结构和动态行为

### 2.1.1 expressorderbl模块局部模块的职责

ExpressOrderbl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。ExpressOrderbl模块的职责及接口参见软件体系结构描述文档。根据体系结构的设计，采用分层风格，将系统分为展示层，业务逻辑层，数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了expressorderblservice, expressorderdataservice 两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了ExpressorderController，这样ExpressorderController 将会将用户管理相关的业务逻辑职责和逻辑控制委托给expressorderbl 对象。ExpressorderPO是做为管理信息的持久化对象被添加到设计模型中的。

expressorderbl模块的设计如图2.1.1-1所示.

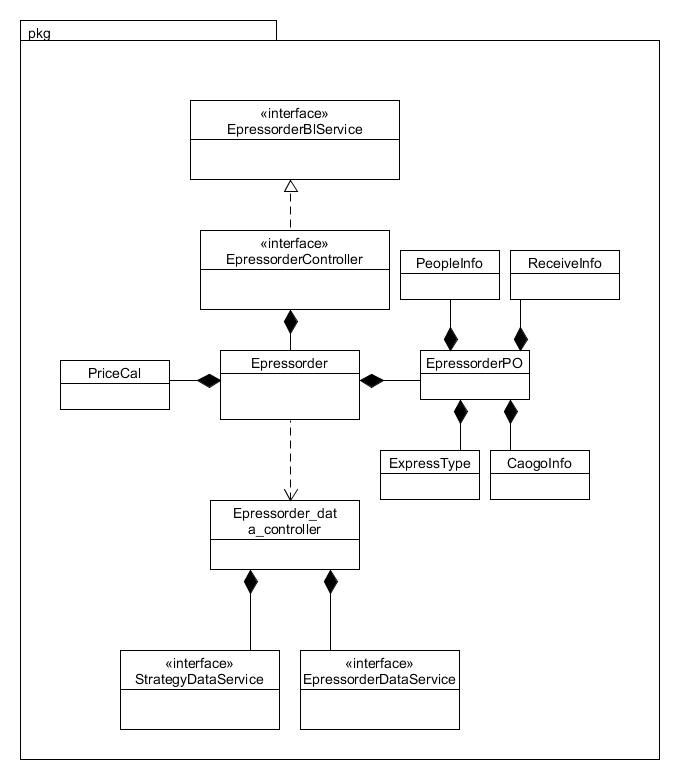


图2.1.1-1 Expressorder模块类的设计

|  |  |
| --- | --- |
| 类 | 职责 |
| ExpressOrderController | 负责实现对应于有关寄件单界面所需的方法 |
| ExpressOrder | 负责处理有关寄件单的功能 |

表2.1.1-2 expressorderbl模块的各个类的职责

### 2.1.2 expressorderbl模块局部模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ExpressorderController.getState | 语法 | public StateVO getState(String order\_id); | |
| 前置条件 | 已知订单号，需要获得订单货运状态 | |
| 后置条件 | 返回订单的货运状态 | |
| ExpressorderController.getArrivingOrders | 语法 | public List<ExpressOrderVO> getArrivingOrders(String institution\_id); | |
| 前置条件 | 已知机构id，需要获得即将送达该机构的订单 | |
| 后置条件 | 返回即将送达该机构的订单 | |
| ExpressorderController.getCurrentOrders | 语法 | public List< ExpressOrderVO > getCurrentOrders(String institution\_id); | |
| 前置条件 | 已知机构id，需要获得当前机构未送出的订单 | |
| 后置条件 | 返回当前机构未送出的订单 | |
| ExpresssorderController.setArrivedOrders | 语法 | public ResultMessage setArrivedOrder(List< ExpressOrderVO> orders,String institution\_id); | |
| 前置条件 | 制定到达单，已知机构id和订单列表 | |
| 后置条件 | 系统更改对应订单的货运状态 | |
| ExpressorderController.setOffOrders | 语法 | public Boolean setOffOrders(List< ExpressOrderVO> orders,String institution\_id); | |
| 前置条件 | 制定装车单，已知机构id和订单列表 | |
| 后置条件 | 系统更改对应订单的货运状态，添加发往地址 | |
| ExpressorderController.createOrder | 语法 | public ExpressOrderVO createOrder(ExpressorderVO order\_info); | |
| 前置条件 | 生成订单，已知完整订单信息 | |
| 后置条件 | 根据订单信息给出报价和时间信息，返回订单PO类 | |
| ExpressorderController.saveOrder | 语法 | Public ResultMessage saveOrder (ExpressOrderVO order); | |
| 前置条件 | 已知订单，需要将订单持久化保存 | |
| 后置条件 | 持久化保存订单信息 | |
| ExpressOrderController.receiveOrder | 语法 | public ResultMessage receiveOrder(String order\_id,ReceiveInfoVO receiveInfo); | |
| 前置条件 | 订单确认收货，已知订单号和收件信息 | |
| 后置条件 | 更改订单货运状态，订单添加收件信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Expressorder.getState | | | 获得物流状态 |
| Expressorder.getArrivingOrders | | | 获得即将到达的订单 |
| Expressorder.getCurrentOrders | | | 获得即将发送的订单 |
| Expresssorder.setArrivedOrders | | | 设置到达状态 |
| Expressorder.setOffOrders | | | 设置发送状态 |
| Expressorder.createOrder | | | 创建订单 |
| Expressorder.saveOrder | | | 保存订单 |
| ExpressOrder.receiveOrder | | | 订单确认收货 |

expressorder\_bl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Expressorder.getState | 语法 | public StateVO getState(String order\_id); | |
| 前置条件 | 已知订单号，需要获得订单货运状态 | |
| 后置条件 | 返回订单的货运状态 | |
| Expressorder.getArrivingOrders | 语法 | public List<ExpressOrderVO> getArrivingOrders(String institution\_id); | |
| 前置条件 | 已知机构id，需要获得即将送达该机构的订单 | |
| 后置条件 | 返回即将送达该机构的订单 | |
| Expressorder.getCurrentOrders | 语法 | public List< ExpressOrderVO > getCurrentOrders(String institution\_id); | |
| 前置条件 | 已知机构id，需要获得当前机构未送出的订单 | |
| 后置条件 | 返回当前机构未送出的订单 | |
| Expresssorder.setArrivedOrders | 语法 | public ResultMessage setArrivedOrder(List< ExpressOrderVO> orders,String institution\_id); | |
| 前置条件 | 制定到达单，已知机构id和订单列表 | |
| 后置条件 | 系统更改对应订单的货运状态 | |
| Expressorder.setOffOrders | 语法 | public Boolean setOffOrders(List< ExpressOrderVO> orders,String institution\_id); | |
| 前置条件 | 制定装车单，已知机构id和订单列表 | |
| 后置条件 | 系统更改对应订单的货运状态，添加发往地址 | |
| Expressorder.createOrder | 语法 | public ExpressOrderVO createOrder(ExpressorderVO order\_info); | |
| 前置条件 | 生成订单，已知完整订单信息 | |
| 后置条件 | 根据订单信息给出报价和时间信息，返回订单PO类 | |
| Expressorder.saveOrder | 语法 | Public ResultMessage saveOrder (ExpressOrderVO order); | |
| 前置条件 | 已知订单，需要将订单持久化保存 | |
| 后置条件 | 持久化保存订单信息 | |
| ExpressOrder.receiveOrder | 语法 | public ResultMessage receiveOrder(String order\_id,ReceiveInfoVO receiveInfo); | |
| 前置条件 | 订单确认收货，已知订单号和收件信息 | |
| 后置条件 | 更改订单货运状态，订单添加收件信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Expressorder\_data.add(ExpressOrderPO order); | | | 新增单一持久化序列 |
| Expressorder\_ data.find(String order\_id); | | | 查找单一持久化序列 |
| Expressorder\_ data.modify(ExpressOrderPO order) | | | 改变单一持久化序列 |
| Expressorder\_data.findArriving(String institution\_id) | | | 查找列表持久化序列 |
| Expressorder\_data.findCurrent(String institution\_id) | | | 查找列表持久化序列 |

### 2.1.3 expressorderbl模块的行为

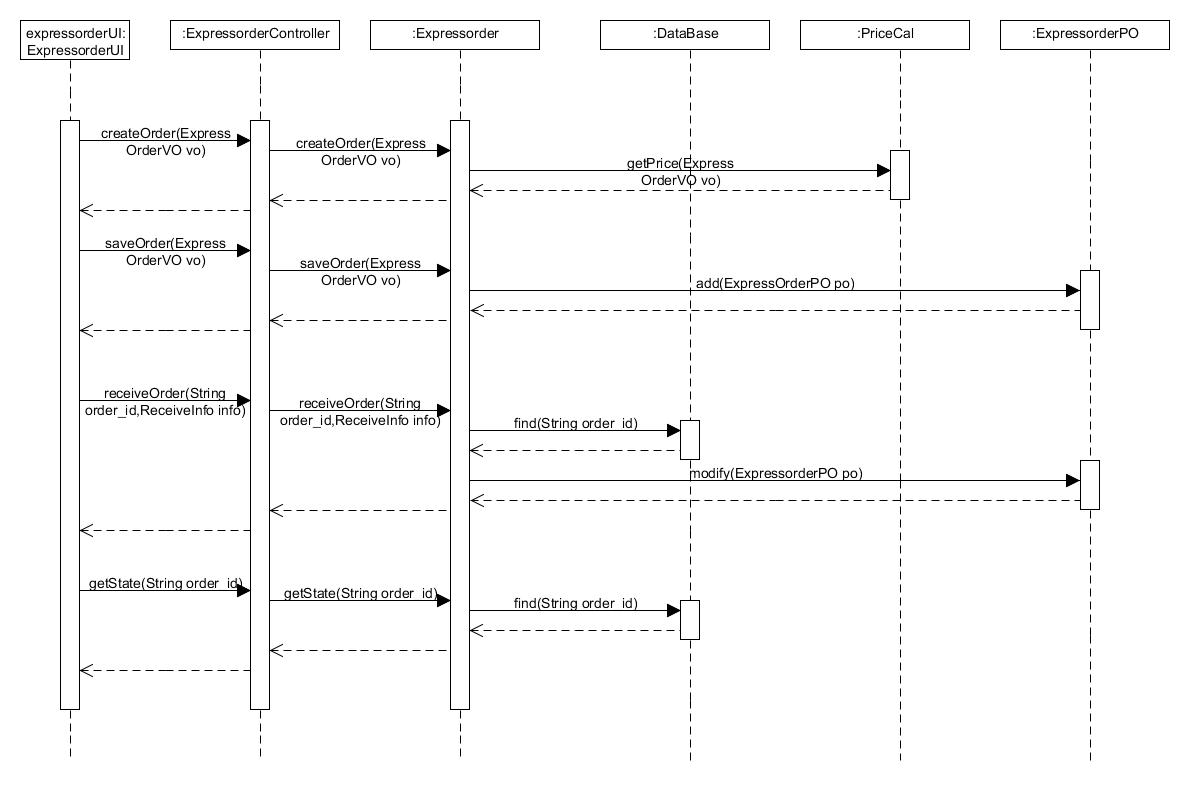


图2.1.3-1 订单创建、物流信息查询、确认收件、保存订单的顺序图

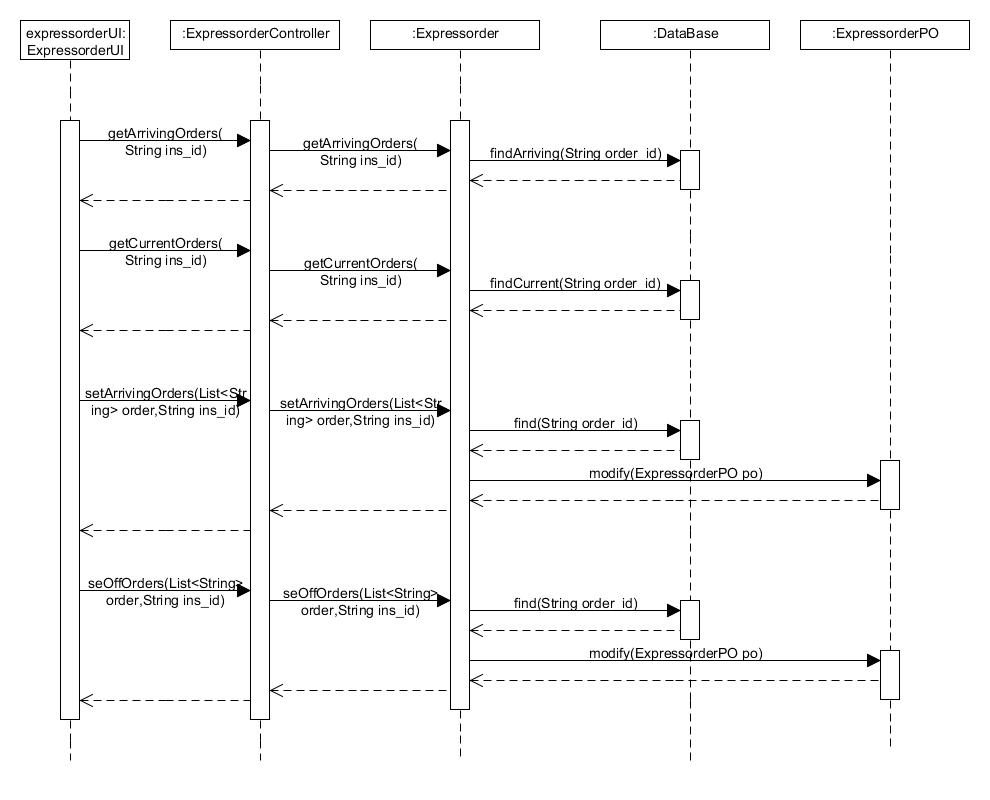


图2.1.3-2 设置和获得订单装车、出发、到达状态的顺序图

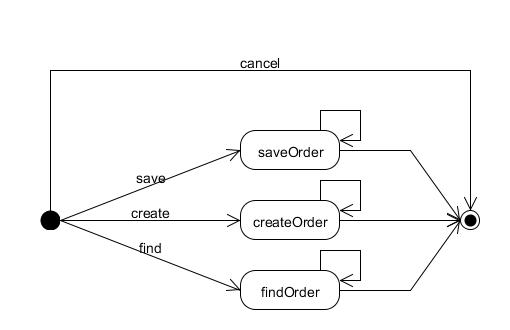


图2.1.3-3 订单创建、物流信息查询、确认收件、保存订单的状态图

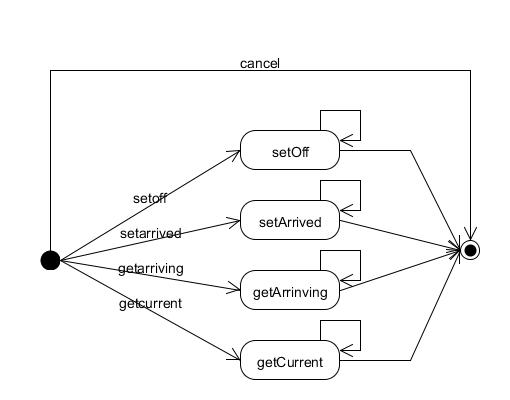


图2.1.3-4 设置和获得订单装车、出发、到达状态的状态图

## 2.2 transmanagebl模块的静态结构和动态行为

### 2.2.1 transmanagebl模块局部模块的职责

transmanagebl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。

transmanagebl模块的职责及接口参见软件体系结构描述文档。根据体系结构的设计，采用分层风格，将系统分为展示层，业务逻辑层，数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了transmanageblservice, transmanagedataservice 两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了TransmanageController，这样TransmanageController 将会将用户管理相关的业务逻辑职责和逻辑控制委托给transmanagebl 对象。CarPO、DriverPO是做为管理信息的持久化对象被添加到设计模型中的。

transmanagebl模块的设计如图2.2.1-1所示.

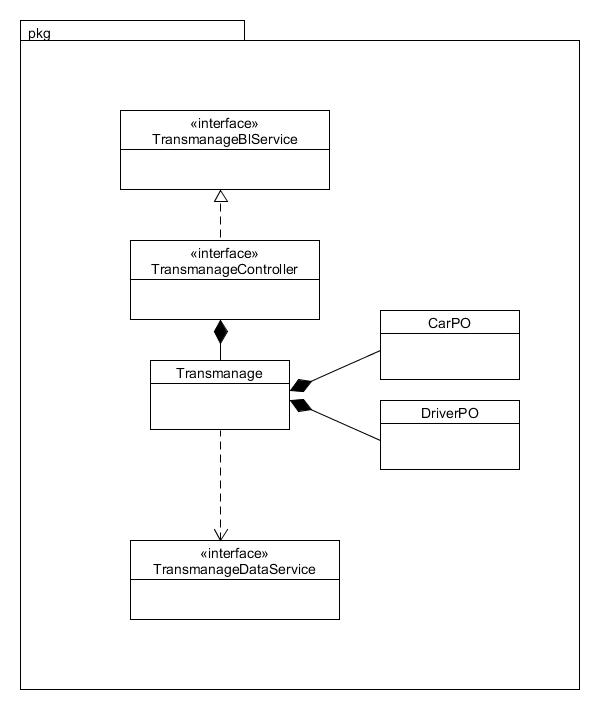


图2.2.1-1 transmanage模块类的设计

### 2.2.2 transmanagebl模块局部模块的接口规范

TransmanageController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ManageController. getAllCar | 语法 | public CarVO[] getAllCar(String id) | |
| 前置条件 | id符合营业厅编号规范 | |
| 后置条件 | 返回该营业厅所有的车辆 | |
| ManageController. getSingleCar | 语法 | public CarVO getSingleCar(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回该id对应的车辆的信息 | |
| ManageController. modifyCar | 语法 | public ResultMessage  modifyCar(CarVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| ManageController. addCar | 语法 | public ResultMessage  addCar(CarVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| ManageController. deleteCar | 语法 | public ResultMessage  deleteCar(String id) | |
| 前置条件 | 对应id的车辆在数据库中 | |
| 后置条件 | 删除对应id的车辆 | |
| ManageController. getAllDriver | 语法 | public DriverVO[] getAllDriver(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回该机构所有的司机 | |
| ManageController. getSingleDriver | 语法 | public DriverVO getSingleDriver(String id) | |
| 前置条件 | id>=0&&id<DriverVO[].length | |
| 后置条件 | 返回该id对应的司机的信息 | |
| ManageController. modifyDriver | 语法 | public ResultMessage  modifyDriver(DriverVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| ManageController. addDriver | 语法 | public ResultMessage  addDriver(DriverVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| ManageController. deleteDriver | 语法 | public ResultMessage  deleteDriver(String id) | |
| 前置条件 | id>=0&&id<DriverVO[].length | |
| 后置条件 | 持久化该操作 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Manage. getAllCar | | | 获得所有车辆信息 |
| Manage. getSingleCar | | | 获得单个车辆信息 |
| Manage. modifyCar | | | 更改车辆信息 |
| Manage.addCar | | | 添加车辆 |
| Manage.deleteCar | | | 删除车辆 |
| Manage. getAllDriver | | | 获得所有司机信息 |
| Manage. getSingleDriver | | | 获得单个司机信息 |
| Manage. modifyDriver | | | 更改司机信息 |
| Manage.addDriver | | | 添加司机 |
| Manage.deleteDriver | | | 删除司机 |

transmanagebl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Manage. getAllCar | 语法 | public CarVO[] getAllCar(String id) | |
| 前置条件 | id符合营业厅编号规范 | |
| 后置条件 | 返回该营业厅所有的车辆 | |
| Manage. getSingleCar | 语法 | public CarVO getSingleCar(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回该id对应的车辆的信息 | |
| Manage. modifyCar | 语法 | public ResultMessage  modifyCar(CarVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. addCar | 语法 | public ResultMessage  addCar(CarVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. deleteCar | 语法 | public ResultMessage  deleteCar(String id) | |
| 前置条件 | 对应id的车辆在数据库中 | |
| 后置条件 | 删除对应id的车辆 | |
| Manage. getAllDriver | 语法 | public DriverVO[] getAllDriver(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回该机构所有的司机 | |
| Manage. getSingleDriver | 语法 | public DriverVO getSingleDriver(String id) | |
| 前置条件 | id>=0&&id<DriverVO[].length | |
| 后置条件 | 返回该id对应的司机的信息 | |
| Manage. modifyDriver | 语法 | public ResultMessage  modifyDriver(DriverVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. addDriver | 语法 | public ResultMessage  addDriver(DriverVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. deleteDriver | 语法 | public ResultMessage  deleteDriver(String id) | |
| 前置条件 | id>=0&&id<DriverVO[].length | |
| 后置条件 | 持久化该操作 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| manage. findSingleCar(String id) | | | 根据ID查找单一持久化对象 |
| manage. modifyCar(CarPO po) | | | 修改单一持久化对象 |
| manage.addCar(CarPO po) | | | 增加单一持久化对象 |
| manage.deleteCar(String id) | | | 删除单一持久化对象 |
| manage. findSingleDriver(String id) | | | 根据ID查找单一持久化对象 |
| manage. modifyDriver(DriverPO po) | | | 修改单一持久化对象 |
| manage.addDriver(DriverPO po) | | | 增加单一持久化对象 |
| manage.deleteDriver(String id) | | | 删除单一持久化对象 |

### 2.2.3 transmanagebl模块的行为

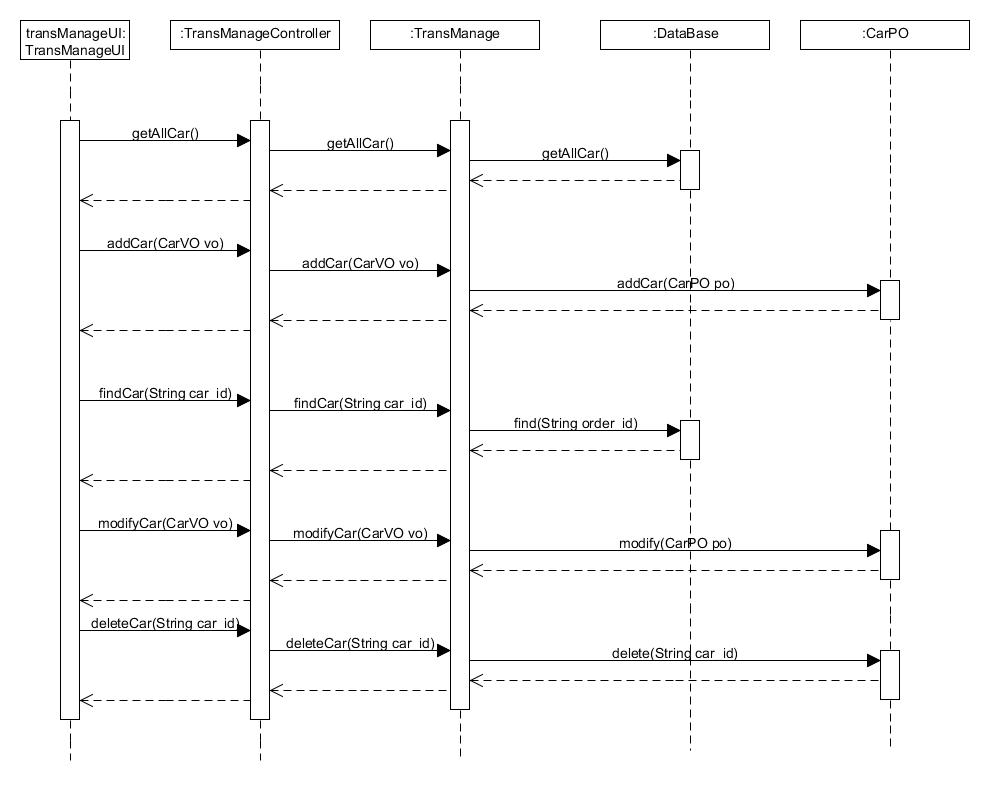


图2.2.3-1 车辆增删改查的顺序图

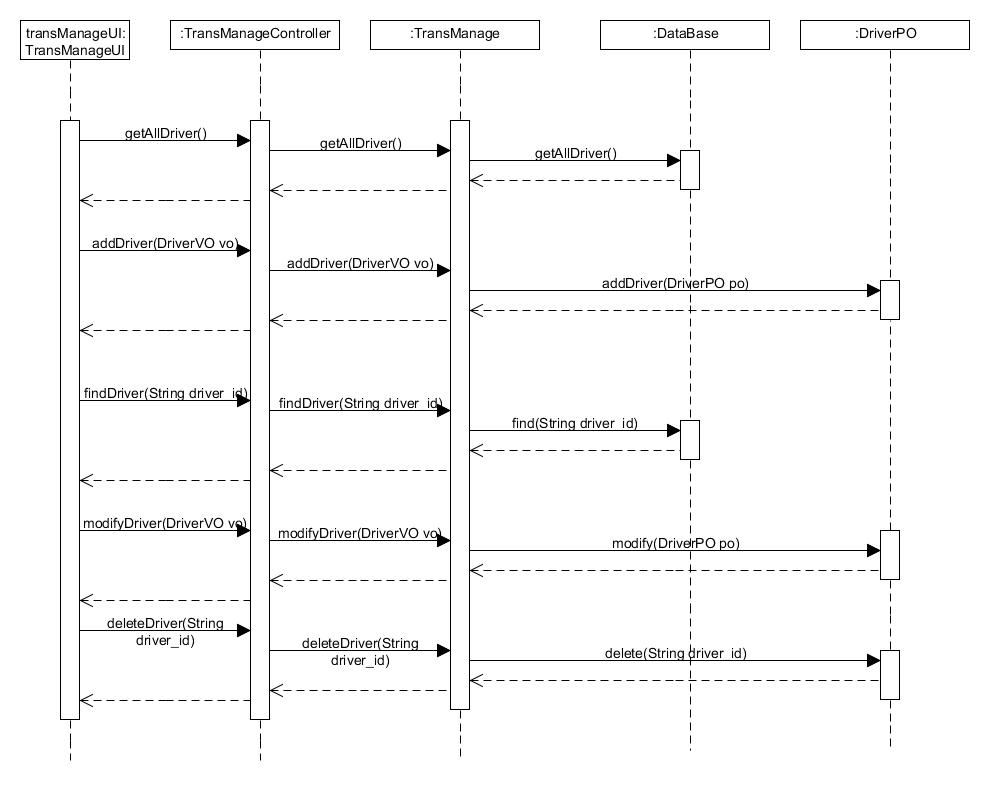


图2.2.3-2 司机增删改查的顺序图

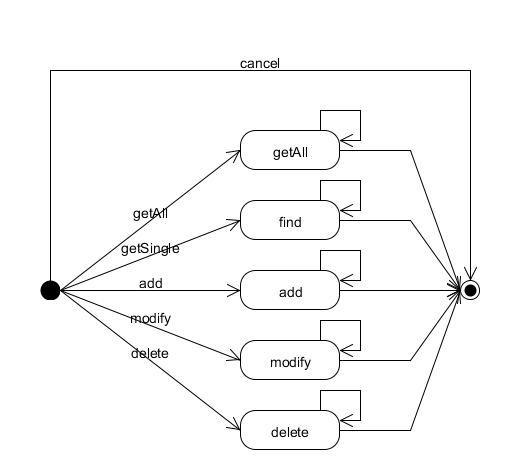


图2.2.3-3 transmanage的状态图

## 2.3 orderbl模块的静态结构和动态行为

### 2.1.1 orderbl模块局部模块的职责

orderbl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。

orderbl模块的职责及接口参见软件体系结构描述文档。根据体系结构的设计，采用分层风格，将系统分为展示层，业务逻辑层，数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了orderblservice, orderdataservice 两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了OrderController，这样OrderController 将会将用户管理相关的业务逻辑职责和逻辑控制委托给orderbl 对象。LoadOrderPO、ArrivedOrderPO、DeliveryOrderPO、StockinOrderPO,StockoutOrderPO是做为管理信息的持久化对象被添加到设计模型中的。

orderbl模块的设计如图2.3.1-1所示.

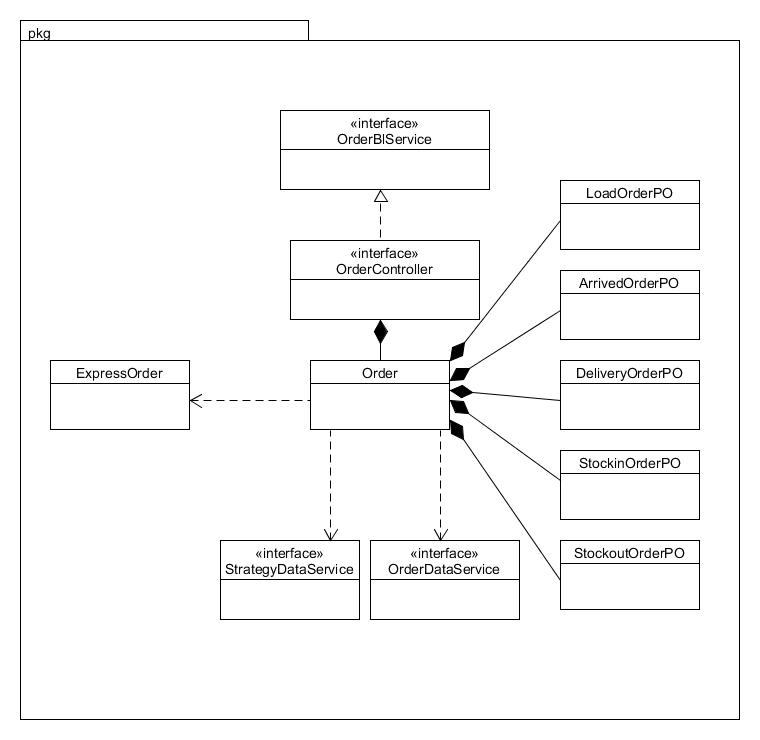


图2.3.1-1 orderbl模块的各个类设计

### 2.3.2 orderbl模块局部模块的接口规范

orderController类的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrderController.  createLoadOrder | 语法 | Public LoadOrderVO createLoadOrders(List<ExpressOrderVO> orders,LoadOrderVO loadOrderInfo) | |
| 前置条件 | 装车单制定，已知订单列表和装车信息 | |
| 后置条件 | 根据订单和装车信息生成装车单，返回装车单 | |
| OrderController.createArrivedOrder | 语法 | Public ArrivedOrderVO createArrivedOrder(List<ExpressOrderVO> orders,ArrivedOrderVO arrivedOrderInfo); | |
| 前置条件 | 到达单制定，已知订单列表和到达信息 | |
| 后置条件 | 根据订单和到达信息生成到达单，返回到达单 | |
| OrderController.createDeliveryOrder | 语法 | public DeliveryOrderVO createDeliveryOrder(List<ExpressOrderVO> orders,DeliveryOrderVO deliveryOrderInfo); | |
| 前置条件 | 派件单制定，已知订单列表和派件信息 | |
| 后置条件 | 根据订单和派件信息生成派件单，返回派件单 | |
| OrderController.createStockInOrder | 语法 | public StockInOrderVO createStockInOrder(List<ExpressOrderVO> orders,StockInOrderVO stockInOrderInfo); | |
| 前置条件 | 入库单制定，已知订单列表和入库信息 | |
| 后置条件 | 根据订单和入库信息生成入库单，返回入库单 | |
| OrderController.createStockOutOrder | 语法 | public StockOutOrderVO createStockOutOrder(List<ExpressOrderVO> orders,StockOutOrderVO stockOutOrderInfo); | |
| 前置条件 | 出库单制定，已知订单列表和出库信息 | |
| 后置条件 | 根据订单和出库信息生成出库单，返回出库单 | |
| OrderController.saveStockOutOrder | 语法 | public ResultMessage saveStockOutOrder(StockOutOrderVO order); | |
| 前置条件 | 已知出库单，需要将出库单持久化保存 | |
| 后置条件 | 持久化保存出库单 | |
| OrderController.saveStockInOrder | 语法 | public ResultMessage saveStockInOrder(StockInOrderVO order); | |
| 前置条件 | 已知入库单，需要将入库单持久化保存 | |
| 后置条件 | 持久化保存入库单 | |
| OrderController.saveDeliveryOrder | 语法 | public ResultMessage saveDeliveryOrder(DeliveryOrderVO order); | |
| 前置条件 | 已知派件单，需要将派件单持久化保存 | |
| 后置条件 | 持久化保存派件单 | |
| OrderController.saveLoadOrder | 语法 | public ResultMessage saveLoadOrder(LoadOrderVO order); | |
| 前置条件 | 已知装车单，需要将装车单持久化保存 | |
| 后置条件 | 持久化保存装车单 | |
| OrderController.saveArrivedOrder | 语法 | public ResultMessage saveArrivedOrders (ArrivedOrderVO order); | |
| 前置条件 | 已知到达单，需要将到达单持久化保存 | |
| 后置条件 | 持久化保存到达单 | |
| OrderController.findLoadOrder | 语法 | public LoadOrderVO findLoadOrder(String id); | |
| 前置条件 | 已知装车单单号 | |
| 后置条件 | 返回装车单 | |
| OrderController.findArrivedOrder | 语法 | public ArrivedOrderVO findArrivedOrder(String id); | |
| 前置条件 | 已知到达单单号 | |
| 后置条件 | 返回到达单 | |
| OrderController.findDeliveryOrder | 语法 | public DeliveryOrderVO findDeliveryOrder(String id); | |
| 前置条件 | 已知派件单单号 | |
| 后置条件 | 返回派件单 | |
| OrderController.findStockInOrder | 语法 | public StockInOrderVO findStockInOrder(String id); | |
| 前置条件 | 已知入库单单号 | |
| 后置条件 | 返回入库单 | |
| OrderController.findStockOutOrder | 语法 | public StockOutOrderVO findStockOutOrder(String id); | |
| 前置条件 | 已知出库单单号 | |
| 后置条件 | 返回出库单 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Order.createLoadOrder | | | 创建装车单 |
| Order.createArrivedOrder | | | 创建到达单 |
| Order.createDeliveryOrder | | | 创建派件单 |
| Order.createStockInOrder | | | 创建入库单 |
| Order.createStockOutOrder | | | 创建出库单 |
| Order.saveStockOutOrder | | | 保存出库单 |
| Order.saveStockInOrder | | | 保存入库单 |
| Order.saveDeliveryOrder | | | 保存派件单 |
| Order.saveLoadOrder | | | 保存装车单 |
| Order.saveArrivedOrder | | | 保存到达单 |
| Order.findLoadOrder | | | 查找装车单 |
| Order.findArrivedOrder | | | 查找到达单 |
| Order.findDeliveryOrder | | | 查找派件单 |
| Order.findStockInOrder | | | 查找入库单 |
| Order.findStockOutOrder | | | 查找出库单 |

order类的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Order.  createLoadOrder | 语法 | Public LoadOrderVO createLoadOrders(List<ExpressOrderVO> orders,LoadOrderVO loadOrderInfo) | |
| 前置条件 | 装车单制定，已知订单列表和装车信息 | |
| 后置条件 | 根据订单和装车信息生成装车单，返回装车单 | |
| Order.createArrivedOrder | 语法 | Public ArrivedOrderVO createArrivedOrder(List<ExpressOrderVO> orders,ArrivedOrderVO arrivedOrderInfo); | |
| 前置条件 | 到达单制定，已知订单列表和到达信息 | |
| 后置条件 | 根据订单和到达信息生成到达单，返回到达单 | |
| Order.createDeliveryOrder | 语法 | public DeliveryOrderVO createDeliveryOrder(List<ExpressOrderVO> orders,DeliveryOrderVO deliveryOrderInfo); | |
| 前置条件 | 派件单制定，已知订单列表和派件信息 | |
| 后置条件 | 根据订单和派件信息生成派件单，返回派件单 | |
| Order.createStockInOrder | 语法 | public StockInOrderVO createStockInOrder(List<ExpressOrderVO> orders,StockInOrderVO stockInOrderInfo); | |
| 前置条件 | 入库单制定，已知订单列表和入库信息 | |
| 后置条件 | 根据订单和入库信息生成入库单，返回入库单 | |
| Order.createStockOutOrder | 语法 | public StockOutOrderVO createStockOutOrder(List<ExpressOrderVO> orders,StockOutOrderVO stockOutOrderInfo); | |
| 前置条件 | 出库单制定，已知订单列表和出库信息 | |
| 后置条件 | 根据订单和出库信息生成出库单，返回出库单 | |
| Order.saveStockOutOrder | 语法 | public ResultMessage saveStockOutOrder(StockOutOrderVO order); | |
| 前置条件 | 已知出库单，需要将出库单持久化保存 | |
| 后置条件 | 持久化保存出库单 | |
| Order.saveStockInOrder | 语法 | public ResultMessage saveStockInOrder(StockInOrderVO order); | |
| 前置条件 | 已知入库单，需要将入库单持久化保存 | |
| 后置条件 | 持久化保存入库单 | |
| Order.saveDeliveryOrder | 语法 | public ResultMessage saveDeliveryOrder(DeliveryOrderVO order); | |
| 前置条件 | 已知派件单，需要将派件单持久化保存 | |
| 后置条件 | 持久化保存派件单 | |
| Order.saveLoadOrder | 语法 | public ResultMessage saveLoadOrder(LoadOrderVO order); | |
| 前置条件 | 已知装车单，需要将装车单持久化保存 | |
| 后置条件 | 持久化保存装车单 | |
| Order.saveArrivedOrder | 语法 | public ResultMessage saveArrivedOrders (ArrivedOrderVO order); | |
| 前置条件 | 已知到达单，需要将到达单持久化保存 | |
| 后置条件 | 持久化保存到达单 | |
| Order.findLoadOrder | 语法 | public LoadOrderVO findLoadOrder(String id); | |
| 前置条件 | 已知装车单单号 | |
| 后置条件 | 返回装车单 | |
| Order.findArrivedOrder | 语法 | public ArrivedOrderVO findArrivedOrder(String id); | |
| 前置条件 | 已知到达单单号 | |
| 后置条件 | 返回到达单 | |
| Order.findDeliveryOrder | 语法 | public DeliveryOrderVO findDeliveryOrder(String id); | |
| 前置条件 | 已知派件单单号 | |
| 后置条件 | 返回派件单 | |
| Order.findStockInOrder | 语法 | public StockInOrderVO findStockInOrder(String id); | |
| 前置条件 | 已知入库单单号 | |
| 后置条件 | 返回入库单 | |
| Order.findStockOutOrder | 语法 | public StockOutOrderVO findStockOutOrder(String id); | |
| 前置条件 | 已知出库单单号 | |
| 后置条件 | 返回出库单 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Order\_data.add(ArrivedOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.add(LoadOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.add(DeliveryOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.add(StockInOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.add(StockOutOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.findLoad(String id); | | | 查找单一持久化序列 |
| Order\_data.findArrived(String id); | | | 查找单一持久化序列 |
| Order\_data.findDelivery(String id); | | | 查找单一持久化序列 |
| Order\_data.findStockIn(String id); | | | 查找单一持久化序列 |
| Order\_data.findStockOut(String id); | | | 查找单一持久化序列 |
| Expressorder.getArrivingOrders | | | 获得即将到达的订单 |
| Expressorder.getCurrentOrders | | | 获得即将发送的订单 |
| Expresssorder.setArrivedOrders | | | 设置到达状态 |
| Expressorder.setOffOrders | | | 设置发送状态 |
| Priceservice.getPrice(String city) | | | 返回距离的计算公式 |
| DistanceService.getDistance(String start,String end) | | | 返回距离 |
| Trans\_Fare\_data.addTransFare(TransFarePO po) | | | 在运费表中增加一条运费持久化对象 |

### 2.3.3 orderbl模块的行为

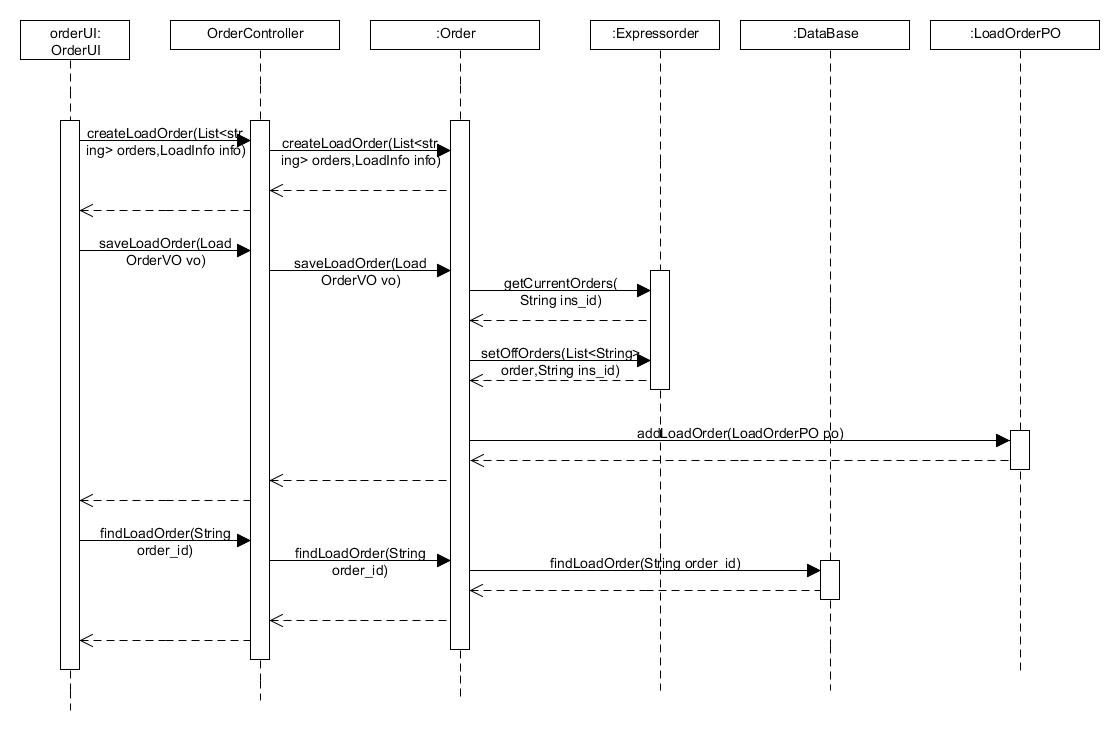


图2.3.3-1装车单生成、保存、查找的顺序图

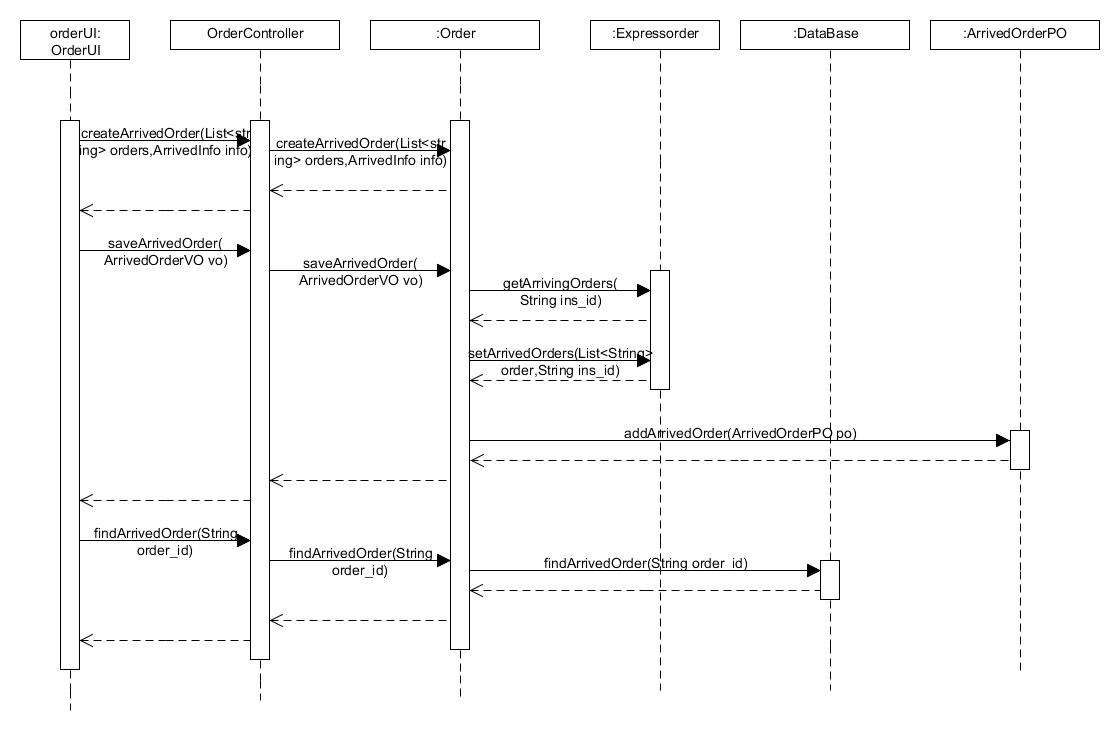


图2.3.3-2到达单生成、保存、查找的顺序图

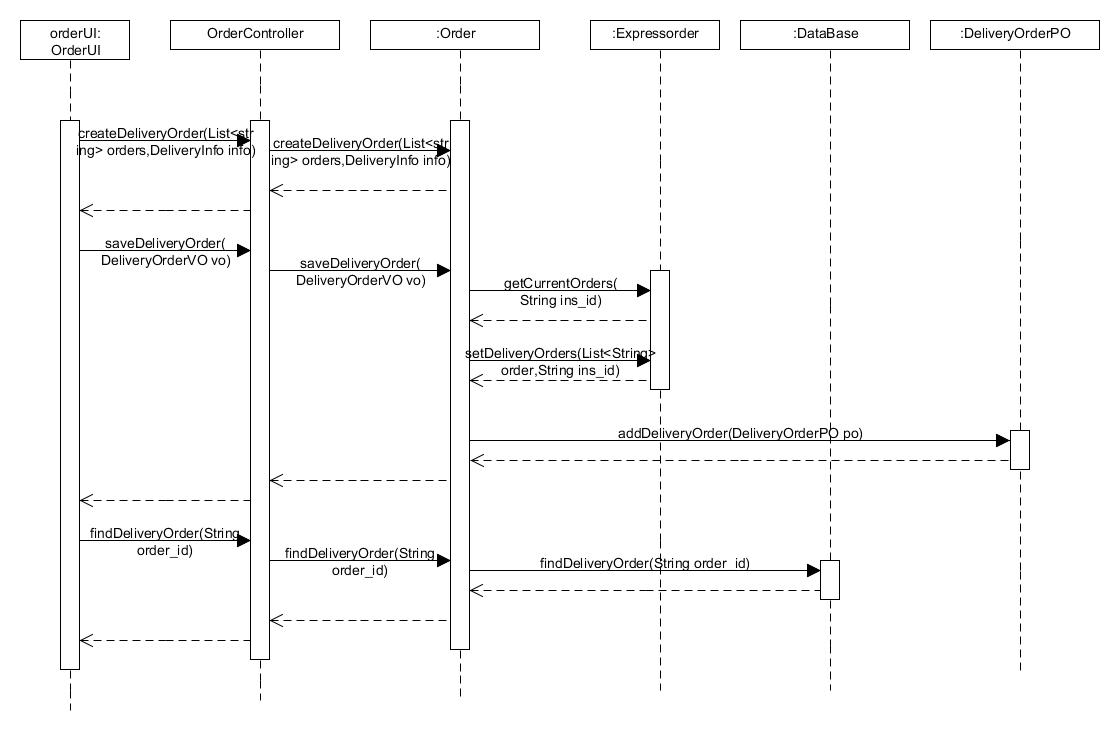


图2.3.3-3派件单生成、保存、查找的顺序图

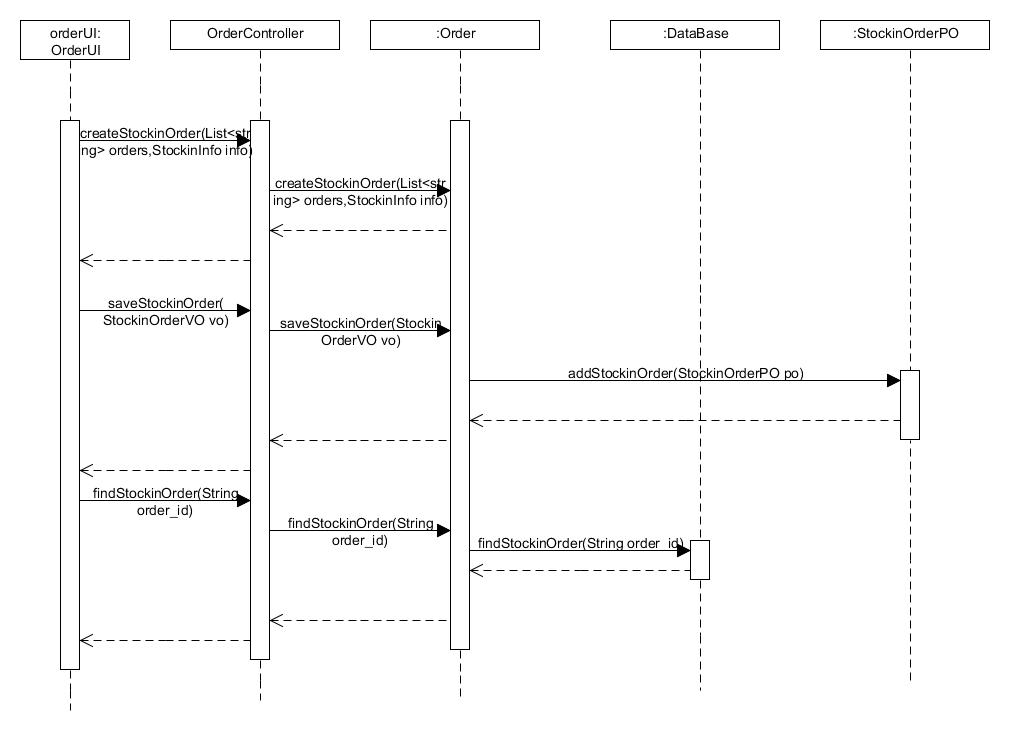


图2.3.3-4入库单生成、保存、查找的顺序图

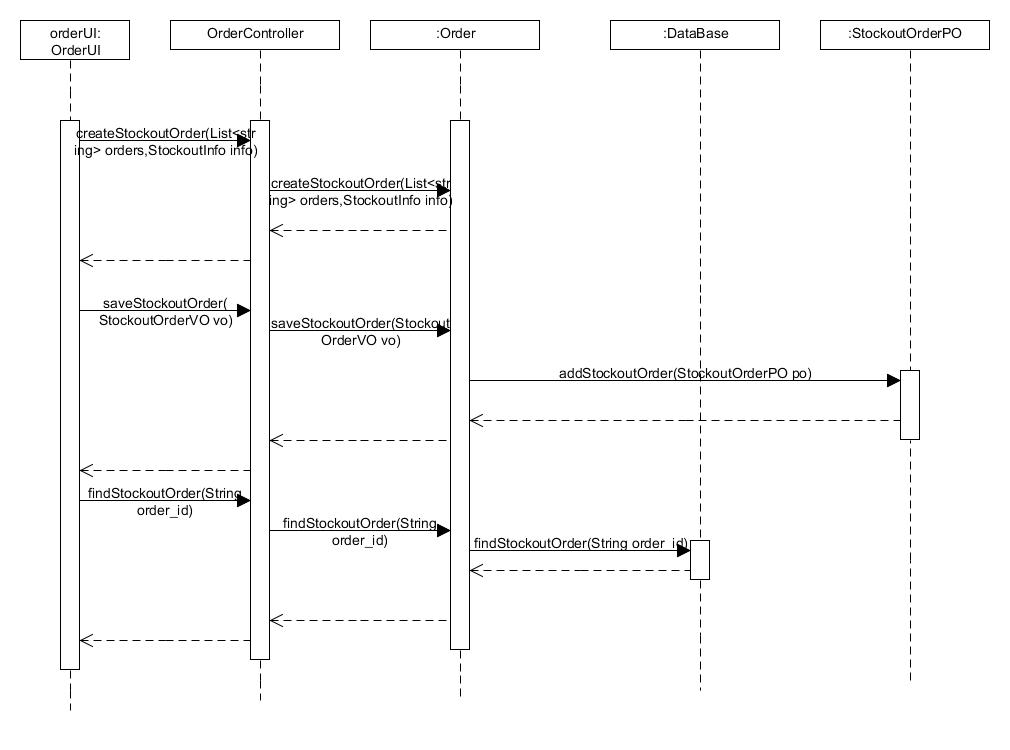


图2.3.3-5出库单生成、保存、查找的顺序图

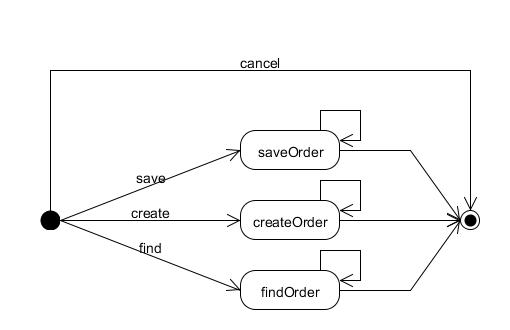


图2.3.3-6出库单生成、保存、查找的顺序图

## 2.4 inventorybl模块的静态结构和动态行为

### 2.4.1 inventorybl模块局部模块的职责

inventorybl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。

inventorybl模块的职责及接口参见软件体系结构描述文档。根据体系结构的设计，采用分层风格，将系统分为展示层，业务逻辑层，数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了inventoryblservice,inventorydataservice 两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了inventoryController，这样inventoryController 将会将用户管理相关的业务逻辑职责和逻辑控制委托给inventorybl 对象。GoodsPO和InventoryPO是作为库存信息的持久化对象被添加到设计模型中的，同时调用StockinOrderPO和StockoutOrderPO。

inventorybl模块的设计如图2.4.1-1所示。

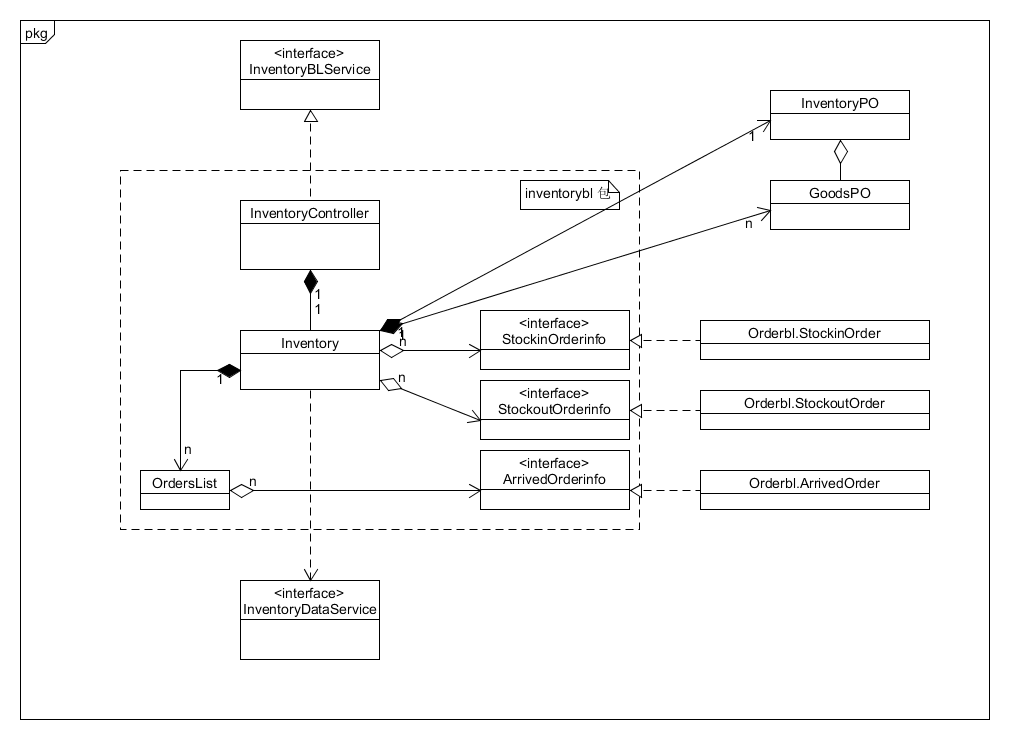


图2.4.1-1inventorybl模块的各个类设计

Inventorybl模块的各个类的职责如表2.4.1-1所示。

|  |  |
| --- | --- |
| 类 | 职责 |
| InventoryController | 负责实现对应于库存管理界面所需的方法 |
| Inventory | 负责管理库存 |

### 2.4.2 inventorybl模块局部模块的接口规范

表2.4.2-1 InventoryController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| InventoryController.observeStock | 语法 | | Public StockObVO observeStock(String begin,String end); |
| 前置条件 | | 已创建一个Inventory领域对象，并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的observeStock方法 |
| InventoryController. countStock | 语法 | | Public StockCheckVO countStock(); |
| 前置条件 | | 已创建一个Inventory领域对象 |
| 后置条件 | | 调用Inventory领域对象的countStock方法 |
| InventoryController. setAlarm | 语法 | | Public ResultMessage setAlarm (double alarmValue); |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的setAlarm方法 |
| InventoryController.distributeSector | 语法 | | Public ResultMessage distributeSector(String beginColumn,String endColumn,String toSector) |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的distributeSector方法 |
| InventoryController.initialadd | 语法 | | Public ResultMessage initialadd (GoodsVO expressorder) |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的initialadd方法 |
| InventoryController.initialdelete | 语法 | | Public ResultMessage initialdelete(String expressorder\_id) |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的initialdelete方法 |
| InventoryController.initialmodify | 语法 | | Public ResultMessage initialmodify （GoosVO expressorder) |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的initialmodify方法 |
| InventoryController.getArrivingOrders | 语法 | | Public List<ExpressOrderVO> getArrivingOrders(String institution\_id); |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的getArrivingOrders方法 |
| InventoryController.stockIn | 语法 | | Public GoodsVO stockIn(ExpressOrderVO expressorder) |
| 前置条件 | | 已创建一个Inventory领域对象 |
| 后置条件 | | 调用Inventory领域对象的stockIn方法 |
| InventoryController.createStockinOrder | 语法 | | Public StockinOrderVO createStockinOrder (List<String> expressorders\_id) |
| 前置条件 | | 已创建一个Inventory领域对象 |
| 后置条件 | | 调用Inventory领域对象的createStockinOrder 方法 |
| InventoryController.stockOut | 语法 | | Public void stockOut(String destination,String transportation,String expressorder\_id) |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的stockOut方法 |
| InventoryController.createStockoutOrder | 语法 | | Public StockoutOrderVO createStockoutOrder (List<ExpressOrderVO> expressorders) |
| 前置条件 | | 已创建一个Inventory领域对象 |
| 后置条件 | | 调用Inventory领域对象的createStockoutOrder 方法 |
| InventoryController.getSingleOrder | 语法 | | Public ExpressOrderVO getSingleExpressOrder(String expressorder\_id) |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的getSingleOrder方法 |
| InventoryController.Alarm | 语法 | | Public String Alarm() |
| 前置条件 | | 已创建一个Inventory领域对象,并且输入符合输入规则 |
| 后置条件 | | 调用Inventory领域对象的Alarm方法 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Inventory.add(ExpressOrderPO expressorder) | | 加入一个快递对象 | |
| Inventory.delete(String expressorder\_id) | | 删除一个快递对象 | |
| Inventory.modify(ExpressOrderPO expressorder) | | 修改一个快递对象 | |
| Inventory.find(String expressorder\_id) | | 根据ID查找一个快递对象 | |
| Inventory.setAlarm(int alarmValue) | | 设置警戒值 | |
| Inventory.getAlarm() | | 获得警戒值 | |
| Order.createStockInOrder | | 生成一个入库单对象 | |
| Order.createStockOutOrder | | 生成一个出库单对象 | |

表2.4.2-2 inventory模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Inventory.observeStock | 语法 | | Public StockObVO observeStock(String begin,String end); |
| 前置条件 | | 请求查看库存且输入正确的起始时间 |
| 后置条件 | | 显示仓库在该时间段内出入库快递数量 |
| Inventory. countStock | 语法 | | Public StockCheckVO countStock(); |
| 前置条件 | | 请求库存盘点 |
| 后置条件 | | 根据盘点时间记录截至点，批次批号，显示库存快照 |
| Inventory. setAlarm | 语法 | | Public ResultMessage setAlarm (double alarmValue); |
| 前置条件 | | 请求设置库存警戒值并输入设置的警戒库存占用比例 |
| 后置条件 | | 更新警戒值,系统自动检测 |
| Inventory.distributeSector | 语法 | | Public ResultMessage distributeSector(String beginColumn,String endColumn,String toSector) |
| 前置条件 | | 请求调整库存分区并输入分配的区域和分配给的分区 |
| 后置条件 | | 更新机动区货架状态 |
| Inventory.initialadd | 语法 | | Public ResultMessage initialadd (GoodsVO expressorder) |
| 前置条件 | | 初始化信息时请求增加信息并输入快递信息和存放位置 |
| 后置条件 | | 保存新增信息，更新库存 |
| Inventory.initialdelete | 语法 | | Public ResultMessage initialdelete(String expressorder\_id) |
| 前置条件 | | 初始化信息时请求删除信息并输入快递编号 |
| 后置条件 | | 删除该快递信息，更新库存 |
| Inventory.initialmodify | 语法 | | Public ResultMessage initialmodify （GoosVO expressorder) |
| 前置条件 | | 初始化信息时请求更改信息并输入已有的快递编号和新的存放位置 |
| 后置条件 | | 修改该快递信息，更新库存 |
| Inventory.getArrivingOrders | 语法 | | Public List<ExpressOrderVO> getArrivingOrders(String institution\_id); |
| 前置条件 | | 已知机构id，需要获得即到达该中转中心的订单 |
| 后置条件 | | 返回到达该机构的订单 |
| Inventory.stockIn | 语法 | | Public GoodsVO stockIn(ExpressOrderVO expressorder) |
| 前置条件 | | 请求快递入库并传入入库快递信息 |
| 后置条件 | | 系统为其分配存储位置，设置入库时间，并返回这些信息，更新库存信息 |
| Inventory.createStockinOrder | 语法 | | Public StockinOrderVO createStockinOrder (List<String> expressorders\_id) |
| 前置条件 | | 入库单制定，已知入库快递列表 |
| 后置条件 | | 生成入库单，返回入库单 |
| Inventory.stockOut | 语法 | | Public void stockOut(String destination,String transportation,String expressorder\_id) |
| 前置条件 | | 请求快递出库，并输入出库快递的目的地，货运方式，和快递编号 |
| 后置条件 | | 更新库存信息 |
| Inventory.createStockoutOrder | 语法 | | Public StockoutOrderVO createStockoutOrder (List<ExpressOrderVO> expressorders) |
| 前置条件 | | 出库单制定，已知出库快递列表 |
| 后置条件 | | 生成出库单，返回出库单 |
| Inventory.getSingleOrder | 语法 | | Public ExpressOrderVO getSingleExpressOrder(String expressorder\_id) |
| 前置条件 | | 请求查看单个快递信息并输入快递编号 |
| 后置条件 | | 返回该快递存储位置和快递本身大小，目的地等信息 |
| Inventory.Alarm | 语法 | | Public String Alarm() |
| 前置条件 | | 库存占用比超出警戒值 |
| 后置条件 | | 系统发出警报信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Inventory\_data.add(ExpressOrderPO expressorder) | | 新增单一持久化对象 | |
| Inventory\_data.delete(String expressorder\_id) | | 新增单一持久化对象 | |
| Inventory\_data.modify(ExpressOrderPO expressorder) | | 修改单一持久化对象 | |
| Inventory\_data.find(String expressorder\_id) | | 根据ID查找单一持久化对象 | |
| Inventory\_data.setAlarm(int alarmValue) | | 设置单一持久化信息 | |
| Inventory\_data.getAlarm() | | 获得单一持久化信息 | |
| Order.createStockInOrder | | 生成单一持久化对象 | |
| Order.createStockOutOrder | | 生成单一持久化对象 | |

### 2.4.3 inventorybl模块的行为

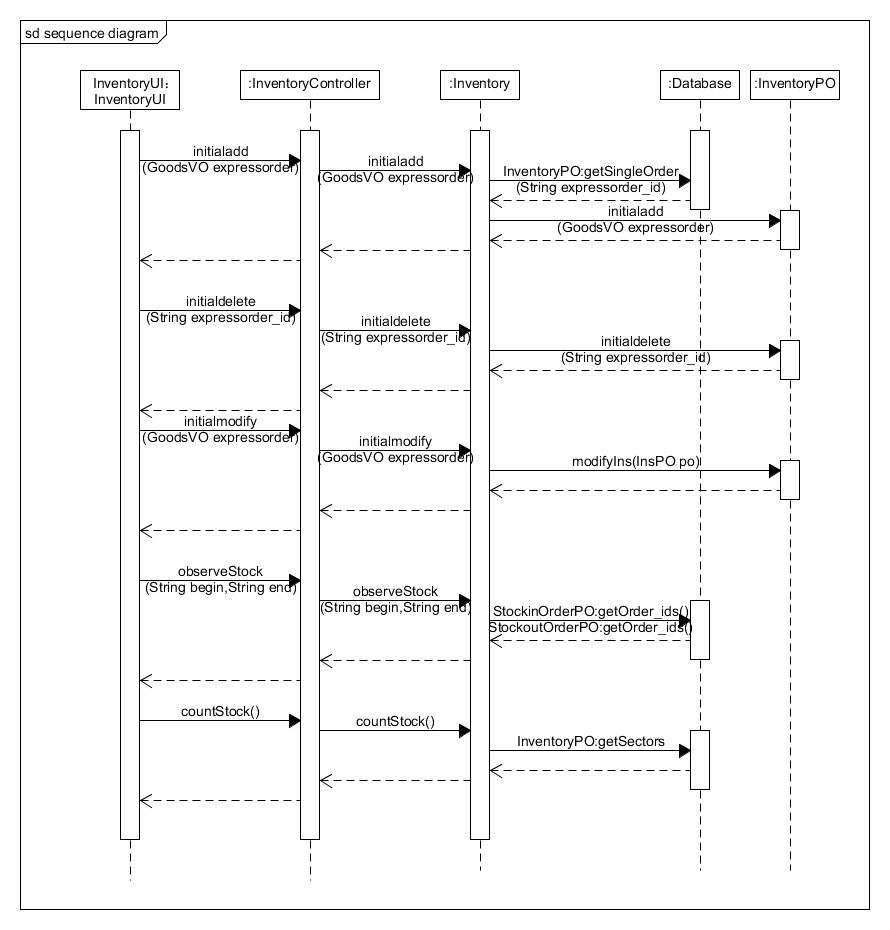


图2.4.3-1-1库存管理的顺序图

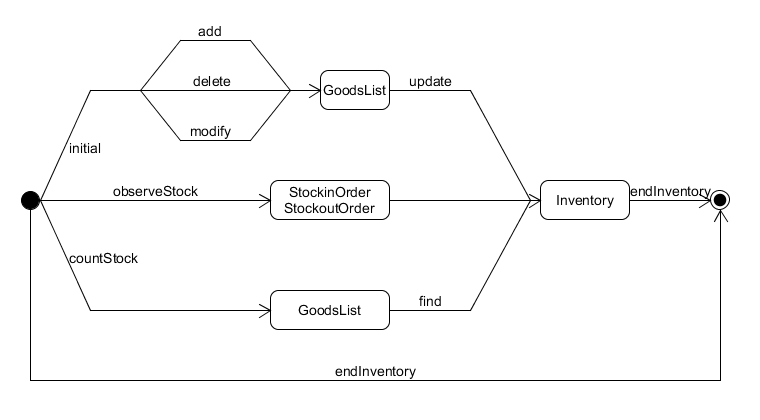


图2.4.3-1-2库存管理的状态图

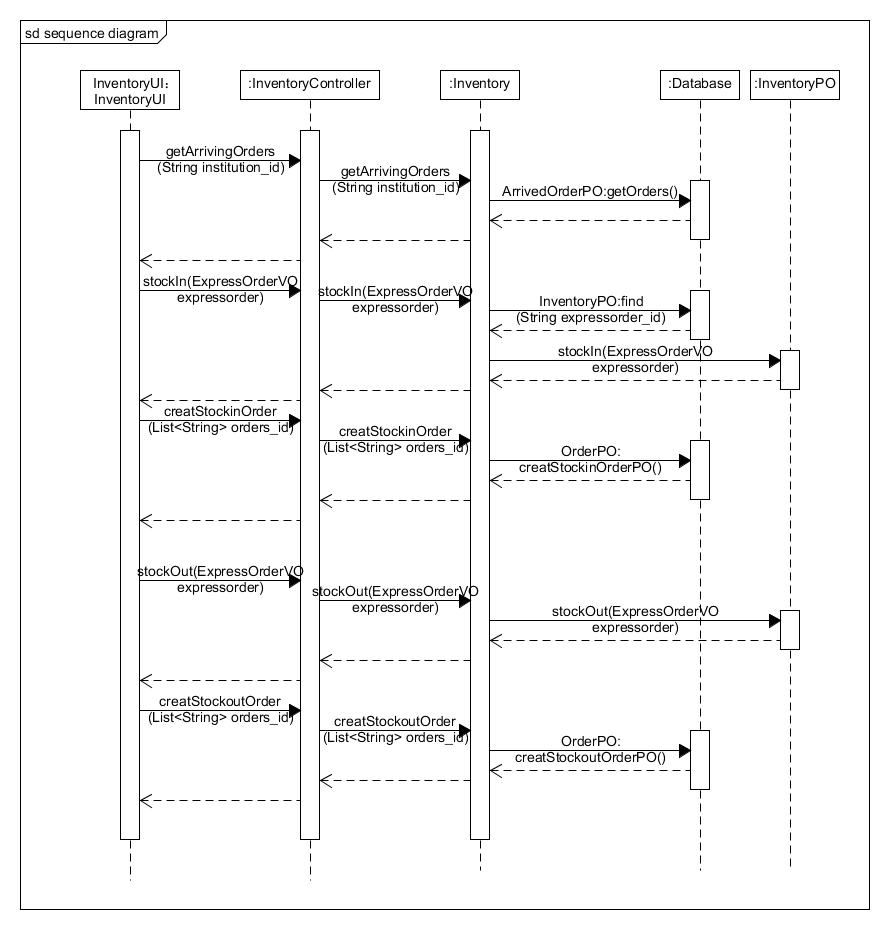


图2.4.3-1-3快递入库出库的顺序图

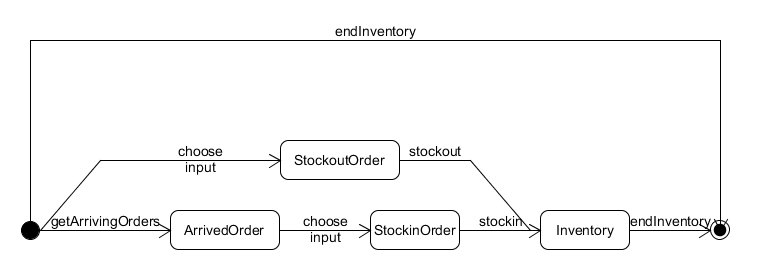


图2.4.3-1-4快递入库出库的顺序图

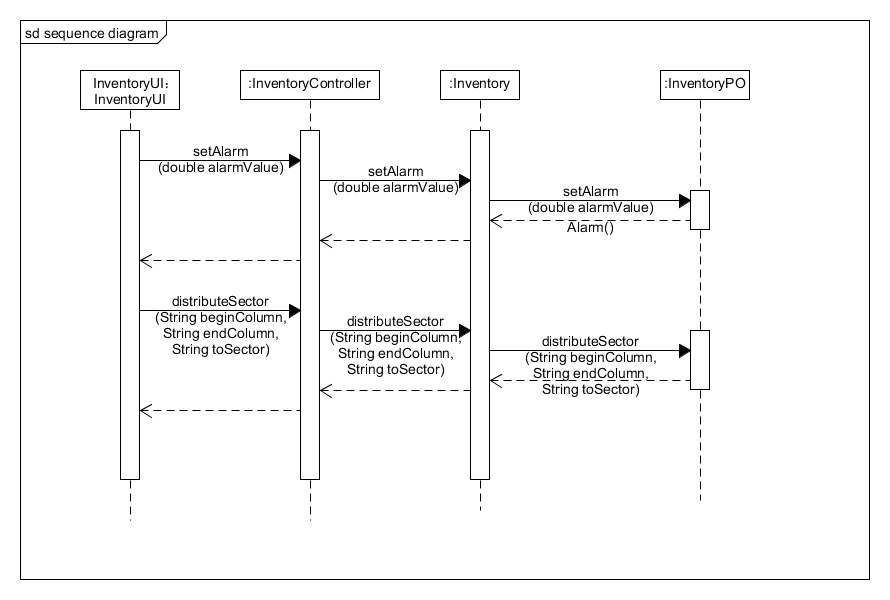


图2.4.3-1-5警戒值设置及分区调整的顺序图

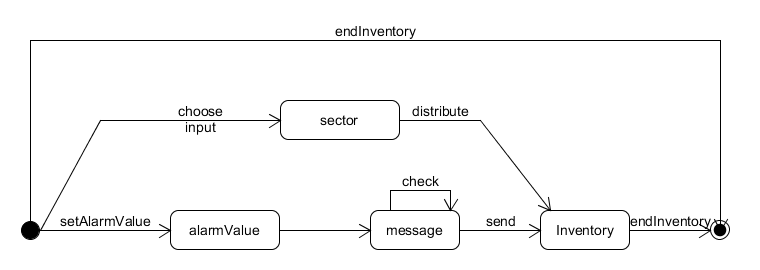


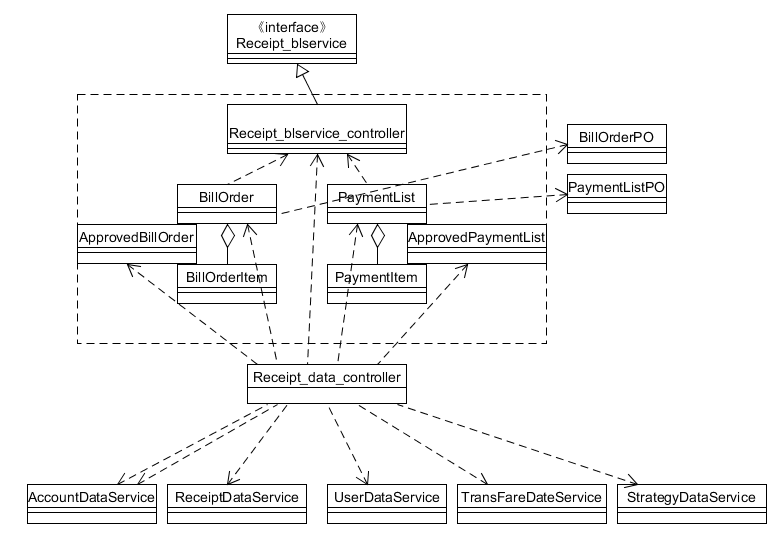
图2.4.3-1-6警戒值设置及分区调整的顺序图

## 2.5 receiptbl模块的静态结构和动态行为

### 2.5.1 receiptbl模块局部模块的职责

Receiptbl模块承担和收款单、付款单制定相关的职责。BusinessClerkUI需要每日提交收款单，financialsUI需要制定付款单，因此都依赖Receiptbl。为降低耦合，该模块向展示层提供receiptbl\_service接口，并在逻辑层用receipt\_bl\_controller实现该接口。Receiptbl所涉及的数据包众多，存储账单要用receipt\_data\_service，收付款单要用account\_data\_service，付款单要用strategy\_data\_service算工资和trans\_fare\_service获得运费，为了降低两层之间耦合，我们使用了receipt\_data\_controller对象，来提供所需接口，屏蔽不必要接口。

图2.5.1-1receiptbl模块的各个类设计



### receiptbl模块局部模块的接口规范

1. 表5.3.2-4 receiptbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ReceiptController.getPaymentList | 语法 | | Public paymentListVO getPaymentList(); |
| 前置条件 | | 启动制定付款单服务 |
| 后置条件 | | 返回收款人的列表 |
| ReceiptController.submitPaymentList | 语法 | | Public ResultMessage submitPaymentList(); |
| 前置条件 | | 提交付款单 |
| 后置条件 | | 系统等待总经理审批，审批通过后，记录付款单信息，更新日志，修改收款人的最后收款时间，修改银行账户数据，修改运费数据条目，清零快递员的本月揽件量 |
| ReceiptController.addPayee | 语法 | | Public ResultMessage addPayee(PayeeVO) |
| 前置条件 | | 用户请求增加收款人 |
| 后置条件 | | 逻辑层增加收款人 |
| ReceiptController.deletePayee | 语法 | | Public ResultMessage deletePayee(PayeeVO) |
| 前置条件 | | 用户请求删除收款人 |
| 后置条件 | | 逻辑层删除收款人 |
| ReceiptController.modifyPayee | 语法 | | Public ResultMessage deletePayee(PayeeVO) |
| 前置条件 | | 用户请求修改收款人 |
| 后置条件 | | 逻辑层修改收款人 |
| ReceiptController.paymetnExcel | 语法 | | Public ResultMessage paymetnExcel (paymentListVO); |
| 前置条件 | | 逻辑层请求导出Excel表 |
| 后置条件 | | 导出excel |
| ReceiptController.getBillOrder | 语法 | | Public billOrderVO getBillOrder |
| 前置条件 | | 制定收款单的服务启动 |
| 后置条件 | | 逻辑层返回收款单PO |
| ReceiptController.addBillItem | 语法 | | Public ResultMessage addBillItem(BillItemVO) |
| 前置条件 | | 用户增加一条收款项 |
| 后置条件 | | 逻辑层增加收款项 |
| ReceiptController.submitBillOrder | 语法 | | Public ResultMessage submitBillOrder() |
| 前置条件 | | 提交收款单的服务启动 |
| 后置条件 | | 系统提交收款单，等待总经理审批 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| PaymentList.addPayee | | 付款单中增加一个付款人 | |
| PaymentList.deletePayee | | 修改营业员、快递员的最后付款时间，清零快递员的本月揽件量 | |
| PaymentList.modifyPayee | | 得到还未付的运费项 | |
| PaymentList.submit | | 提交付款单，状态为未审批 | |
| BillOrder.addBillItem | | 收款单中增加收款项 | |
| BillOrder.submit | | 提交收款单，状态为未审批 | |
| Receirpt\_data.getApprovedPayment(String departmentID) | | 从数据库获得本部门提交的，已经通过审批的单据 | |
| Receirpt\_data.setLocked(String departmentID) | | 将本部门所有已经通过审批的单据上锁（在上边的方法使用之后使用） | |
| Strategy\_data.getSalary | | 从数据库获得快递员提成比、营业员工资 | |
| Strategy\_data.getRent | | 从数据库获得房租的金额和付款日期 | |
| Account.trade(tradePO) | | 从银行账户扣款或收款 | |

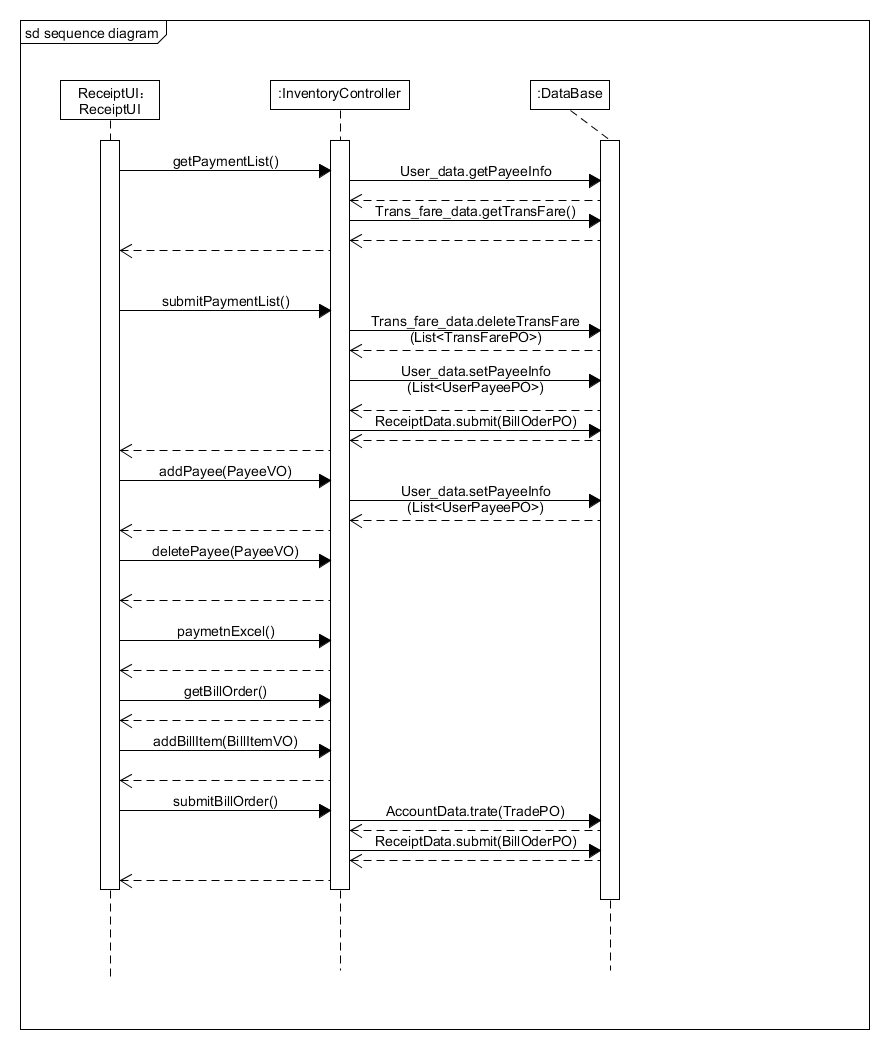
Billorder类接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| BillOrder.addBillItem | 语法 | | Public ResultMessage addBillItem(PayeeVO) |
| 前置条件 | | 用户请求增加收款人 |
| 后置条件 | | 逻辑层增加收款人 |
| BillOrder.submit | 语法 | | Public ResultMessage submit(BillOrder  VO) |
| 前置条件 | | Controller调用提交方法 |
| 后置条件 | | 提交单据至数据库 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Receipt\_data.submitPayment(PaymentListPO) | | 将付款单具体信息提交至数据库，状态为未审批。 | |
| Receipt\_data.submitBillOrder | | 将收款单具体信息提交至数据库，状态为未审批。 | |
| Account.trade(tradePO) | | 从银行账户扣款或收款 | |

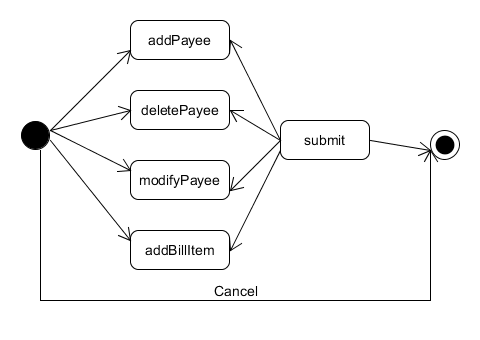
PaymentList类接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| PaymentList.submit | 语法 | | Public ResultMessage submit (PaymentListPO); |
| 前置条件 | | 提交付款单 |
| 后置条件 | | 系统等待总经理审批，审批通过后，记录付款单信息，更新日志，修改收款人的最后收款时间，修改银行账户数据，修改运费数据条目，清零快递员的本月揽件量 |
| PaymentList.addPayee | 语法 | | Public ResultMessage addPayee(PayeeVO) |
| 前置条件 | | 用户请求增加收款人 |
| 后置条件 | | 逻辑层增加收款人 |
| PaymentList.deletePayee | 语法 | | Public ResultMessage deletePayee(PayeeVO) |
| 前置条件 | | 用户请求删除收款人 |
| 后置条件 | | 逻辑层删除收款人 |
| PaymentList.modifyPayee | 语法 | | Public ResultMessage deletePayee(PayeeVO) |
| 前置条件 | | 用户请求修改收款人 |
| 后置条件 | | 逻辑层修改收款人 |
| PaymenList.paymetnExcel | 语法 | | Public ResultMessage paymetnExcel (String Path); |
| 前置条件 | | 逻辑层请求导出Excel表 |
| 后置条件 | | 导出excel |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| PaymentList.addPayee | | 付款单中增加一个付款人 | |
| PaymentList.deletePayee | | 修改营业员、快递员的最后付款时间，清零快递员的本月揽件量 | |
| PaymentList.modifyPayee | | 得到还未付的运费项 | |
| PaymentList.submit | | 提交付款单，状态为未审批 | |
| BillOrder.addBillItem | | 收款单中增加收款项 | |
| BillOrder.submit | | 提交收款单，状态为未审批 | |
| User\_data.getPayeeInfo | | 得到数据库中营业员、快递员的ID、名字、职务和最后收款时间。假如是快递员，就加一项揽件量 | |
| User\_data.setPayeeInfo(List<UserPayeePO>) | | 修改营业员、快递员的最后付款时间，清零快递员的本月揽件量 | |
| Trans\_fare\_data.getTransFare | | 得到还未付的运费项 | |
| Trans\_fare\_data.deleteTransFare(List<TransFarePO>) | | 删除已经付款的运费项 | |
| Strategy\_data.getSalary | | 从数据库获得快递员提成比、营业员工资 | |
| Strategy\_data.getRent | | 从数据库获得房租的金额和付款日期 | |
| Account.trade(tradePO) | | 从银行账户扣款或收款 | |

### receiptbl模块的行为



3.5.2-1 ReceiptBl顺序图



3.5.2-1 ReceiptBl状态图

## 2.6 accountbl模块的静态结构和动态行为

### 2.6.1 accountbl模块局部模块的职责

请注意，AccountBl处理的是和财务人员的库存管理相关的业务，而不是所有和银行账户相关的业务。它负责通过AccountController调用数据层提供的接口，增删银行账户，修改账户名称。同时也调用Log方法，记录用户对银行账户的增删改操作。AccountVO和AccountPO分别作为展示至逻辑，逻辑至数据的值传输对象。AccountController屏蔽掉了一些AccountData提供的不必要接口。AccountBl的模块设计如图2.6.1-1所示。

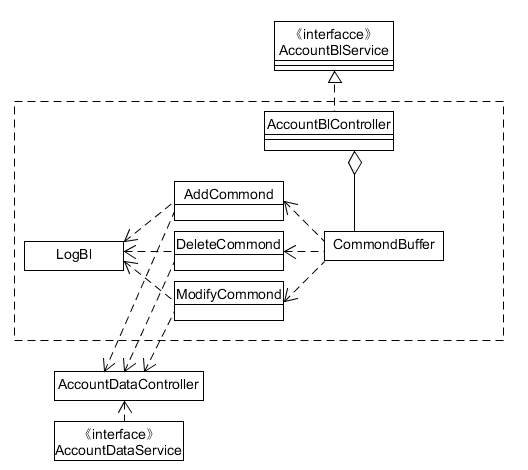


图2.6.1-1

### accountbl模块局部模块的接口规范

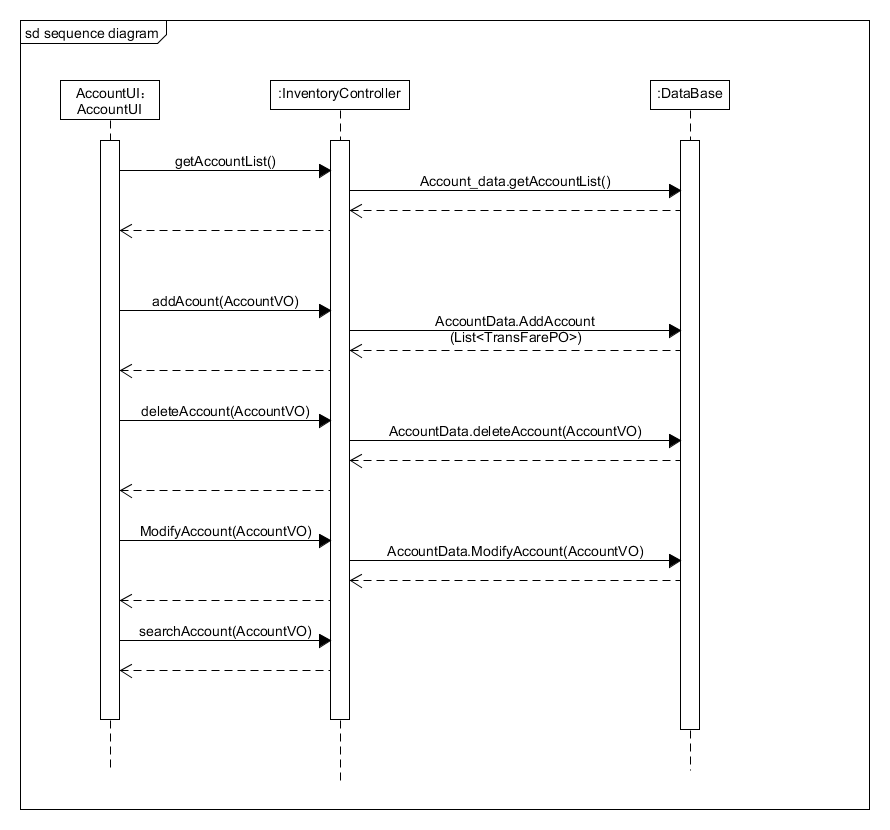
AccountController的接口

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| AccountController.getAccountManageList | 语法 | | Public AccountListVO getAccountManageList() |
| 前置条件 | | 启动账户管理服务 |
| 后置条件 | | 逻辑层返回银行账户的名称和余额列表 |
| AccountController.addAccount | 语法 | | Public ResultMessage addAccount(AccountVO) |
| 前置条件 | | 用户请求增加账户 |
| 后置条件 | | 账户列表增加一个账户，CommondBuffer增加一个增加命令 |
| AccountController.deleteAccount | 语法 | | Public ResusltMessage deleteAccount(AccountVO) |
| 前置条件 | | 用户请求删除账户 |
| 后置条件 | | 账户列表删除一个账户，CommondBuffer增加一个删除命令 |
| AccountController.modifyAcount | 语法 | | Public ResultMessage modifyAccount(AccountVO) |
| 前置条件 | | 用户请求修改账户 |
| 后置条件 | | 账户列表修改一个账户，CommondBuffer增加一个修改命令 |
| AccountController .Confirm | 语法 | | Public ResultMessage confirm() |
| 前置条件 | | 用户确认修改 |
| 后置条件 | | Controller将buffer里的命令依次执行，清空buffer |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| CommondBuffer.addCommond(AccountCommond) | | 增加一条账户信息 | |
| Account\_data. getAccountList | | 获得银行账户列表 | |

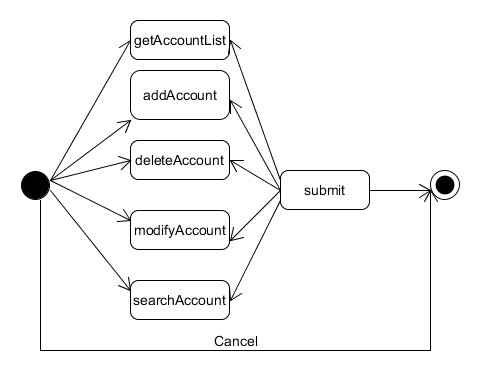
CommondBuffer的接口

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CommondBuffer.addAccount | 语法 | | Public ResultMessage add (AccountVO) |
| 前置条件 | | Controller请求增加账户 |
| 后置条件 | | 账户列表增加一个账户，CommondBuffer增加一个增加命令 |
| CommondBuffer.delete | 语法 | | Public ResusltMessage delete (AccountVO) |
| 前置条件 | | Controller请求删除账户 |
| 后置条件 | | 账户列表删除一个账户，CommondBuffer增加一个删除命令 |
| CommondBuffer.modify | 语法 | | Public ResultMessage modify (AccountVO) |
| 前置条件 | | Controller请求修改账户 |
| 后置条件 | | 账户列表修改一个账户，CommondBuffer增加一个修改命令 |
| Commond.flush | 语法 | | Public ResultMessage flush() |
| 前置条件 | | Controller请求清空缓存 |
| 后置条件 | | Buffer将命令依次执行，清空buffer |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Account\_data.add(AccountManagePO) | | 增加一条账户信息 | |
| Account\_data.delete(String name) | | 删除一条账户信息 | |
| Account\_data.Modify(AccountManagePO) | | 修改一条账户信息 | |

### accountbl模块的行为



2.6.3-1 AccountBl顺序图

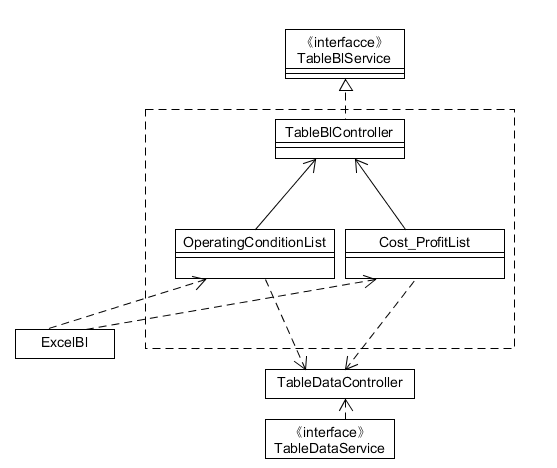


2.6.3-2 AccountBl状态图

## 2.7 tablebl模块的静态结构和动态行为

### 2.7.1 tablebl模块局部模块的职责

TableBl负责财务人员的查看成本收益表和经营状况表的业务逻辑。此处的table指的是成本收益表和经营状况表。它通过调用Excel模块将报表输出为Excel表格。通过Table\_data-Controller获得自己需要的收入支出数据和收付款单记录。Controller是为了屏蔽TableBl不需要的接口。该模块只是读取数据。TableBl的模块设计如图2.6.1-1所示。



### 2.7.2 tablebl模块局部模块的接口规范

TableController模块接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| TableController.getCost\_ProfitList | 语法 | | Public Cost\_ProfitListVO getCost\_ProfitList() |
| 前置条件 | | 查看成本收益表的服务启动 |
| 后置条件 | | 无 |
| TableController.getOperatingConditionList | 语法 | | Public OperatingConditionListVO getOperatingConditionList(Date begin,Date end); |
| 前置条件 | | 查看经营状况表的服务启动 |
| 后置条件 | | 无 |
| TableController. Cost\_ProfitExcel | 语法 | | Public ResultMessage Cost\_ProfitExcel(Cost\_ProfitListVO) |
| 前置条件 | | 导出成本收益表的服务启动 |
| 后置条件 | | 导出表格 |
| TableController. OperatingConditionExcel | 语法 | | Public ResultMessage OperatingConditionExcel(OperatingConditionListVO) |
| 前置条件 | | 导出经营状况表的服务启动 |
| 后置条件 | | 导出表格 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Receipt\_data.getReceipt(Date begin,Date end) | | 获得时间段内所有收付款单的PO表 | |
| Account\_data.getCost\_Profit() | | 获得银行成本收益的PO，包括账户名称、支出、收入 | |

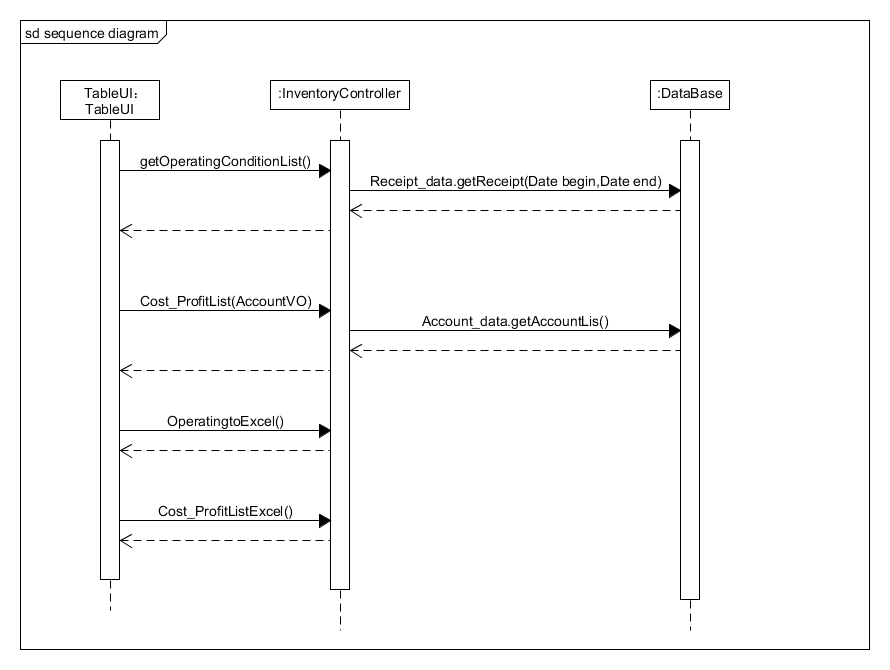
OperatingConditionList模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OperatingConditionList.getVO | 语法 | | Public Cost\_ProfitListVO getVO() |
| 前置条件 | | 查看经营状况表的服务启动 |
| 后置条件 | | 无 |
| OperatingConditionList.toExcel | 语法 | | Public ResultMessage toExcel(String Path); |
| 前置条件 | | Controller请求导出Excel |
| 后置条件 | | 无 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Receipt\_data.getReceipt(Date begin,Date end) | | 获得时间段内所有收付款单的PO表 | |

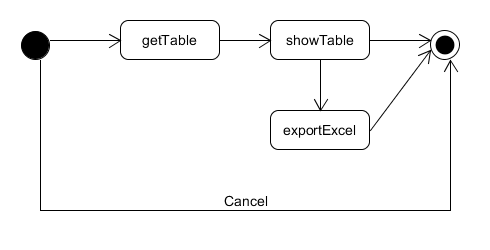
Cost\_ProfitList模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Cost\_ProfitList.getVO | 语法 | | Public Cost\_ProfitListVO getVO() |
| 前置条件 | | 查看成本收益表的服务启动 |
| 后置条件 | | 无 |
| Cost\_ProfitList List.toExcel | 语法 | | Public ResultMessage toExcel(String Path); |
| 前置条件 | | 查看经营状况表的服务启动 |
| 后置条件 | | 无 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Account\_data.getAccountLis() | | 获得银行成本收益的PO，包括账户名称、支出、收入 | |

### 2.7.3 tablebl模块的行为



2.7.3-1 tableBl顺序图



2.7.3-1 tableBl顺序图

## 2.8 initialbl模块的静态结构和动态行为

### 2.8.1 initialbl模块局部模块的职责

InitialBl模块处理和财务人员期初建账相关的业务逻辑。注意：可能我们的系统中期初建账功能和助教或者老师想的不一样。期初建账功能只能修改银行账户信息，其余信息由Initial模块读取，并且存入期初信息中，财务人员不可修改。InitialBl要用的数据service众多，Controller是用来屏蔽接口的。期初建账需要用到AccountBl的CommondBuffer。InitialBl模块设计如图2.8.1-1所示

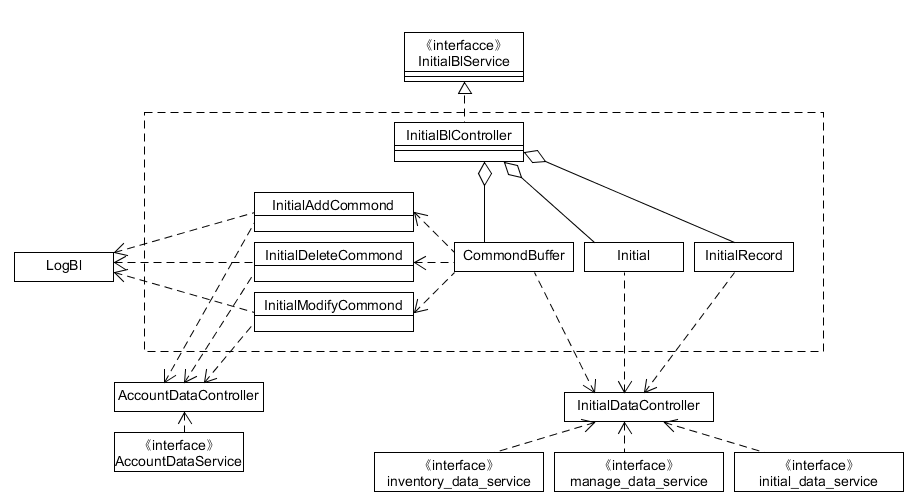


图2.8.1-1 initialBl模块

### 2.8.2 initialbl模块局部模块的接口规范

InitialController模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| InitialBlController.getInitialList | | 语法 | Public InitialVO getInitialList(); |
| 前置条件 | 期初建账功能启动 |
| 后置条件 | UI层进入期初建账界面 |
| InitialBlController.submitInitialList | | 语法 | Public ResultMessage submitInitialList(InitialVO) |
| 前置条件 | 完成期初建账 |
| 后置条件 | 记录期初信息，记录日志，更新数据库的账户信息 |
| InitialController.getInitialHistory | | 语法 | Public InitialHistoryVO getInitialHistory(); |
| 前置条件 | 查看期初信息功能启动 |
| 后置条件 | UI进入查看期初信息界面 |
| InitialBlController.getAccountManageList | 语法 | | Public AccountListVO getAccountManageList() |
| 前置条件 | | 启动账户管理服务 |
| 后置条件 | | 逻辑层返回银行账户的名称和余额列表 |
| InitialBlController.addAccount | 语法 | | Public ResultMessage addAccount(AccountVO) |
| 前置条件 | | 用户请求增加账户 |
| 后置条件 | | 账户列表增加一个账户，CommondBuffer增加一个增加命令 |
| InitialBlController.deleteAccount | 语法 | | Public ResusltMessage deleteAccount(AccountVO) |
| 前置条件 | | 用户请求删除账户 |
| 后置条件 | | 账户列表删除一个账户，CommondBuffer增加一个删除命令 |
| InitialBlController.modifyAcount | 语法 | | Public ResultMessage modifyAccount(AccountVO) |
| 前置条件 | | 用户请求修改账户 |
| 后置条件 | | 账户列表修改一个账户，CommondBuffer增加一个修改命令 |
| InitialBlController .Confirm | 语法 | | Public ResultMessage confirm() |
| 前置条件 | | 用户确认修改 |
| 后置条件 | | Buffer执行命令，清空 |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| AccountBlController.getAccountList | | | 获得银行账户列表 |
| CommondBuffer.add | | | 增加一条银行账户 |
| CommondBuffer.delete | | | 删除一条银行账户 |
| CommondBuffer.modify | | | 修改一条银行账户 |

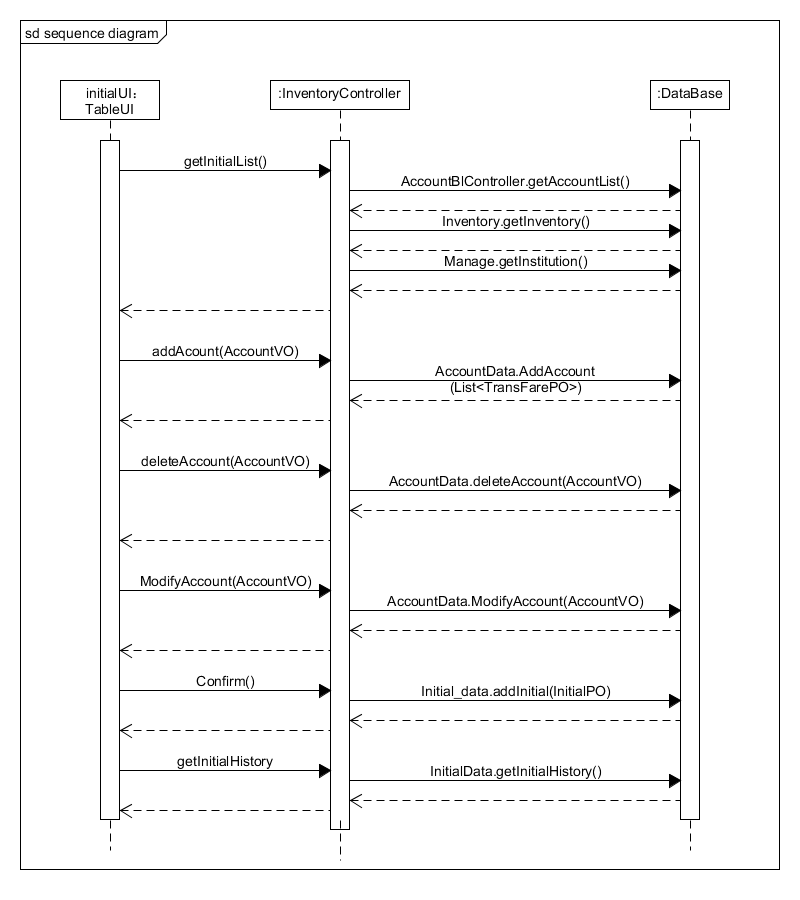
InitialList模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| InitialList.getInitialList | 语法 | | Public InitialVO getInitialList(); |
| 前置条件 | | Controller请求获得期初建账列表 |
| 后置条件 | | 返回期初建账列表 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Inventory.getInventory | | 获得所有仓库的库存信息PO | |
| Manage.getInstitutionPO | | 获得机构、人员、车辆信息 | |
| Initial\_data.addInitial(InitialPO) | | 增加一条期初信息 | |

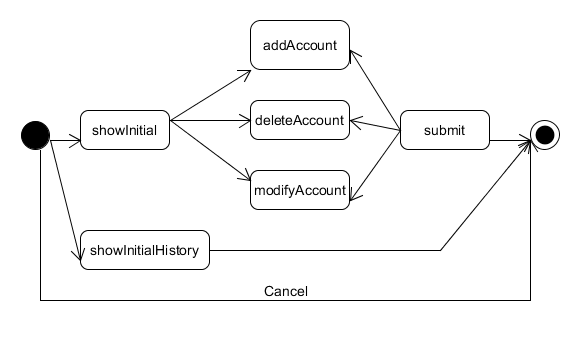
InitialHistoryt模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| InitialList.getInitialHistory | 语法 | | Public InitialHistoryVO getInitialHistory(); |
| 前置条件 | | Controller请求查看期初信息 |
| 后置条件 | | 返回期初历史记录 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Initial\_data.getHistory | | 获得期初信息历史 | |

### 2.8.3 initialbl模块的行为



2.8.3-1 Initial模块顺序图



2.8.3-2 Initial模块状态图

## 2.9 excelbl模块的静态结构和动态行为

### 2.9.1 excelbl模块局部模块的职责

ExcelBl模块不对显示层暴露接口，作为一个工具包而存在，他对外提供Excelable接口，该接口没有定义方法。实现该接口的对象可以传入controller并导出excel。Excelcontroller维护与Excel实际类型相对应的printer。这样设计是考虑到excel对扩展的要求较高，当一种新的对象需要导出excel时，修改范围是，让该对象实现excelable，新建一种导出该对象的printer，并在controller的map中将他们对应起来。也考虑到可以让实现excelable的对象注册类名给controller，然后controller根据类名动态加载printer，这样能省去修改map这一步。不过这样有设计过度之嫌，所以没有采用。ExcelBl模块设计如图2.9.1-1所示。

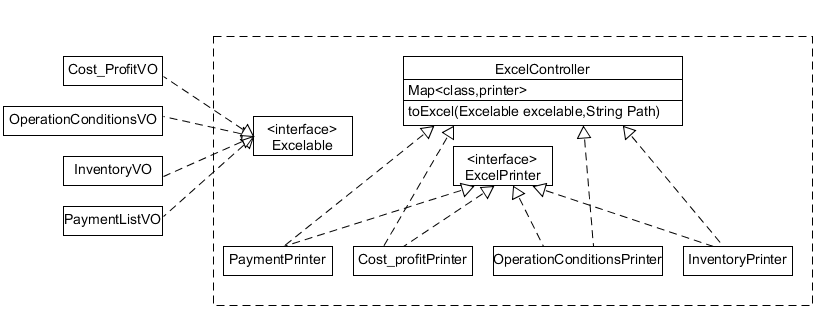


图2.9.1-1ExcelBl模块

### 2.9.2 excelbl模块局部模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ExcelController.toExcel | 语法 | | Public ResultMessage toExcel(Excelable excelable,String Path) |
| 前置条件 | | 业务逻辑层请求导出Excel |
| 后置条件 | | 使用相应printer导出excel |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| ExcelPrinter.toExcel(Excelable excelable,String Path) | | 导出excel | |

### 2.9.3 excelbl模块的行为

以导出经营状况表为例

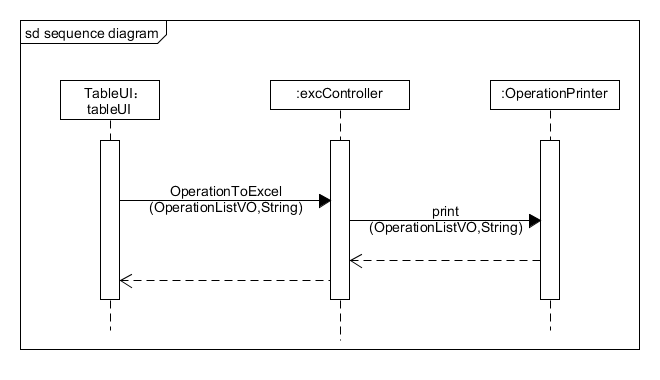
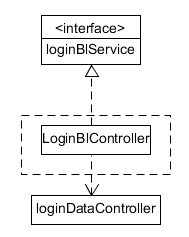


图2.10.3-1 excelBl顺序图

## 2.10 loginbl模块的静态结构和动态行为

### 2.10.1 loginbl模块局部模块的职责

Login处理用户的登录操作，若登录失败则返回失败原因，若成功，则在登陆成功之后加载一些该用户的人员和机构信息作为客户端常量。Login模块内容比较简单，对展示层提供验证UserVO的功能，向下通过LoginController获得用户信息，并验证是否正确。Login模块设计见下图。



2.10.1-1 login模块设计

### 2.10.2 loginbl模块局部模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| loginController.confirm | 语法 | | Public ResultMessage confirm(Login VO) |
| 前置条件 | | 登录功能启动 |
| 后置条件 | | 返回登录结果 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| LoginDataController.confirm | | 调用数据层方法验证用户信息 | |

### 2.10.3 loginbl模块的行为

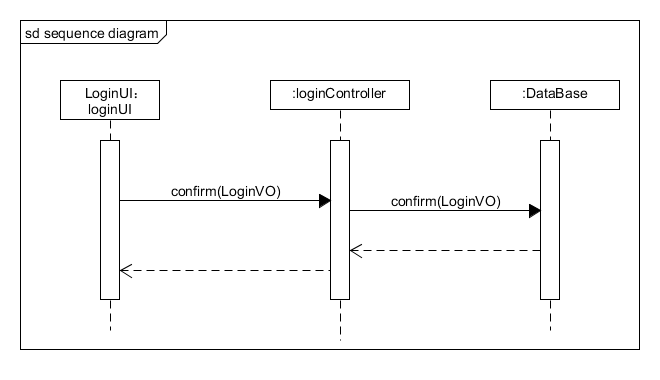
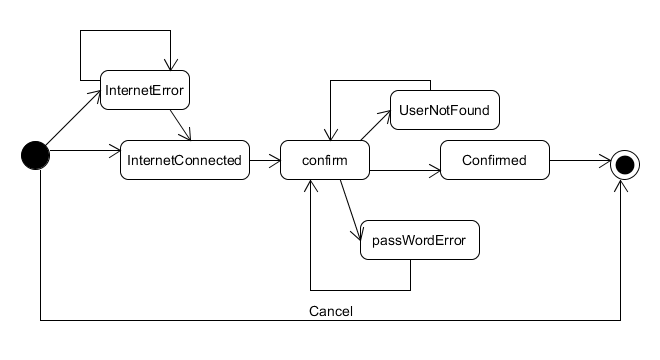


图2.10.3-1 loginBl顺序图



2.10.2-1 login模块设计

## 2.11 managebl模块的静态结构和动态行为

### 2.11.1 managebl模块局部模块的职责

Managebl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。

Managebl模块的职责及接口参见软件体系结构描述文档。根据体系结构的设计，采用分层风格，将系统分为展示层，业务逻辑层，数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了manageblservice,managedataservice 两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了manageController，这样manageController 将会将用户管理相关的业务逻辑职责和逻辑控制委托给managebl 对象。InstitutionPO是做为管理信息的持久化对象被添加到设计模型中的。

managebl模块的设计如图2.11.1-1所示。

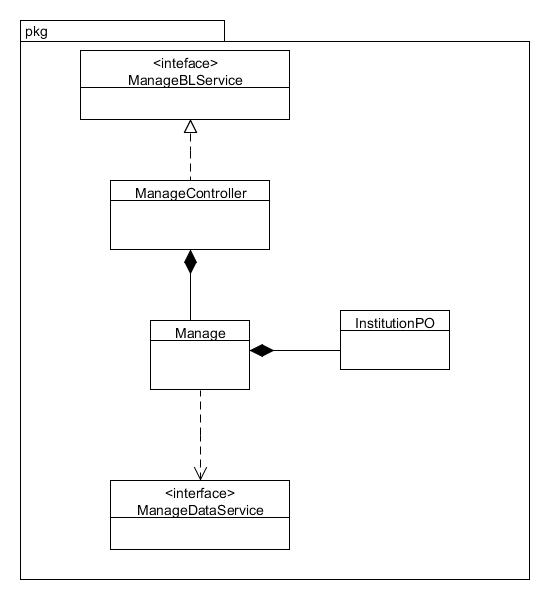


图2.11.1-1 managebl模块的各个类的设计

managebl模块各个类的职责如表2.11.1-1所示。

|  |  |
| --- | --- |
| 类 | 职责 |
| ManageController | 负责实现对应于机构管理界面所需的方法 |
| Manage | 负责管理机构 |

### 2.11.2 managebl模块局部模块的接口规范

表2.11.2-1 ManageController模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ManageController. getAllCity | 语法 | public CityVO[] getAllCity() | |
| 前置条件 | 已创建一个Manage领域对象 | |
| 后置条件 | 调用Manage对象的getAllCity方法 | |
| ManageController. getAllInstitution | 语法 | public InstitutionVO[] getAllInstitution(String id) | |
| 前置条件 | 已创建一个Manage领域对象 | |
| 后置条件 | 调用Manage对象的getAllInstitution方法 | |
| ManageController. getSingleIns | 语法 | public InstitutionVO  getSingleIns(String id) | |
| 前置条件 | 已创建一个Manage领域对象 | |
| 后置条件 | 调用Manage对象的getSingleIns方法 | |
| ManageController. modifyIns | 语法 | public ResultMessage modifyIns(InstitutionVO vo) | |
| 前置条件 | 已创建一个Manage领域对象且VO合法 | |
| 后置条件 | 调用Manage对象的modifyIns方法 | |
| ManageController. addIns | 语法 | public ResultMessage  addIns(InstitutionVO vo) | |
| 前置条件 | 已创建一个Manage领域对象且VO合法 | |
| 后置条件 | 调用Manage对象addIns方法 | |
| ManageController. deleteIns | 语法 | public ResultMessage  deleteIns(String id) | |
| 前置条件 | 已创建一个Manage领域对象且VO合法 | |
| 后置条件 | 调用Manage对象的deleteIns方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Manage. getAllCity | | | 获得持久化对象 |
| Manage. getAllInstitution (String id) | | | 根据ID查找单一持久化对象 |
| Manage. getSingleIns(String id) | | | 根据ID查找单一持久化对象 |
| manage. modifyIns(InstitutionPO po) | | | 修改单一持久化对象 |
| manage.addIns(InstitutionPO po) | | | 增加单一持久化对象 |
| manage.deleteIns(String id) | | | 删除单一持久化对象 |

表2.11.2-2 Manage模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Manage. getAllCity | 语法 | public CityVO[] getAllCity() | |
| 前置条件 |  | |
| 后置条件 | 返回存在机构的所有城市 | |
| Manage. getAllInstitution | 语法 | public InstitutionVO[] getAllInstitution(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回该城市所有的机构 | |
| Manage. getSingleIns | 语法 | public InstitutionVO  getSingleIns(String id) | |
| 前置条件 | id>=0&&id<InstitutionVO[].length | |
| 后置条件 | 返回该id对应的机构的信息 | |
| Manage. modifyIns | 语法 | public ResultMessage modifyIns(InstitutionVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. addIns | 语法 | public ResultMessage  addIns(InstitutionVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. deleteIns | 语法 | public ResultMessage  deleteIns(String id) | |
| 前置条件 | id>=0&&id<InstitutionVO[].length | |
| 后置条件 | 持久化该操作 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DatabaseFactory.getManage | | | 取得Manage数据库的引用 |
| manage. findSingleIns(String id) | | | 根据ID查找单一持久化对象 |
| manage. modifyIns(InstitutionPO po) | | | 修改单一持久化对象 |
| manage.addIns(InstitutionPO po) | | | 增加单一持久化对象 |
| manage.deleteIns(String id) | | | 删除单一持久化对象 |

### 2.11.3 managebl模块的行为

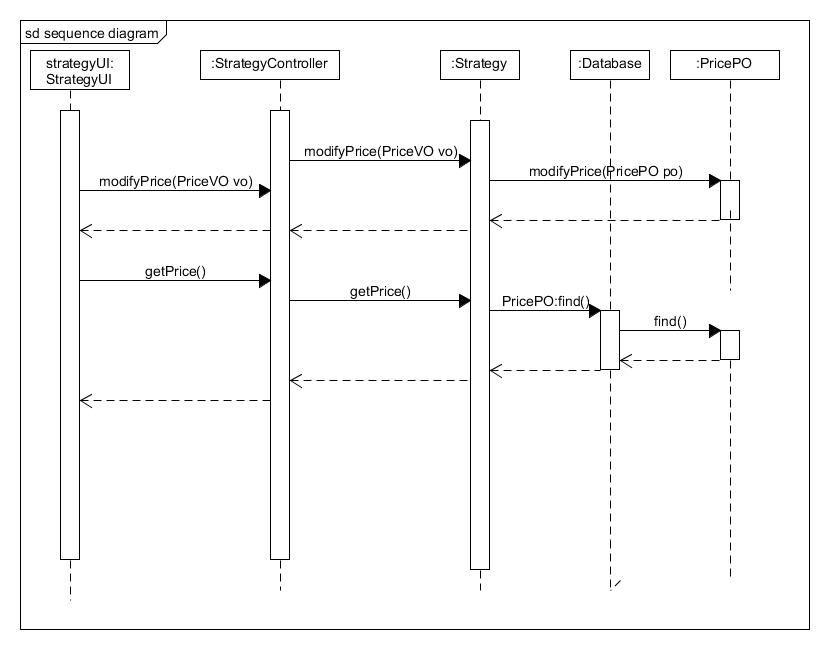


图2.11.3-1 机构管理的顺序图

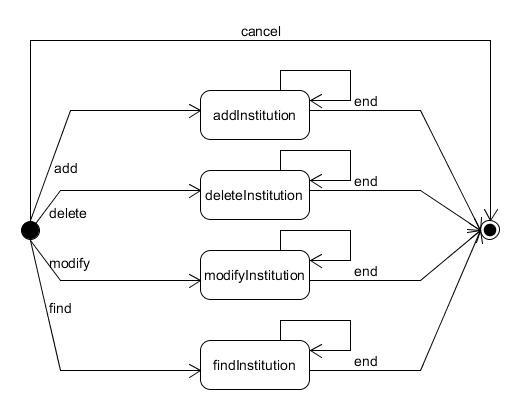


图2.11.3-2 机构管理的状态图

## 2.12 strategybl模块的静态结构和动态行为

### 2.12.1 strategybl模块局部模块的职责

strategybl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。

strategybl模块的职责及接口参见软件体系结构描述文档。根据体系结构的设计，采用分层风格，将系统分为展示层，业务逻辑层，数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了strategyblservice,strategydataservice 两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了strategyController，这样strategyController 将会将用户管理相关的业务逻辑职责和逻辑控制委托给strategybl 对象。pricePO，distancePO，rentPO，salaryPO是做为策略信息的持久化对象被添加到设计模型中的。

strategybl模块的设计如图2.12.1-1所示。

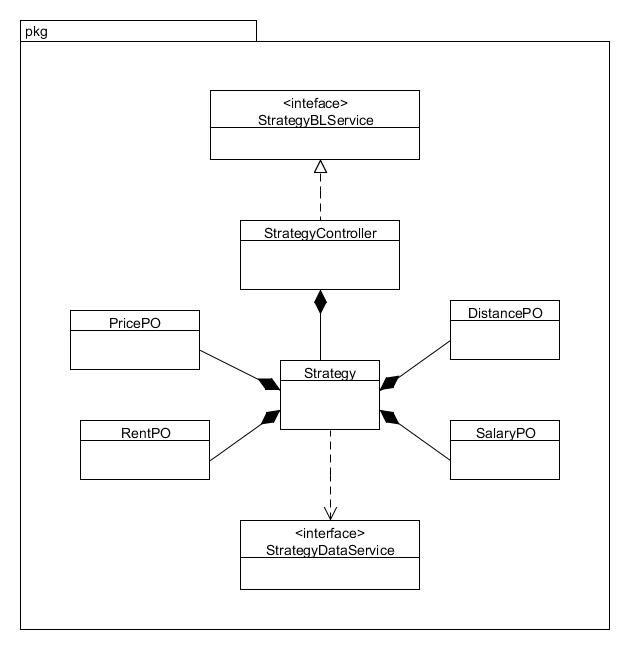


图2.12.1-1 strategybl模块的各个类的设计

strategybl模块各个类的职责如表2.12.1-1所示。

|  |  |
| --- | --- |
| 类 | 职责 |
| StrategyController | 负责实现对应于策略制定界面所需的方法 |
| Strategy | 负责制定策略 |

### 2.12.2 strategybl模块局部模块的接口规范

表2.11.2-1 strategyController类接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| StrategyController. getSalary | 语法 | public SalaryVO getSalary() | |
| 前置条件 | 已创建一个Strategy领域对象 | |
| 后置条件 | 调用Strategy对象的getSalary方法 | |
| StrategyController. ModifySalary | 语法 | public ResultMessage modifySalary(SalaryVO vo) | |
| 前置条件 | 已创建一个Strategy领域对象且VO合法 | |
| 后置条件 | 调用Strategy对象的modifySalary方法 | |
| StrategyController. getDistance | 语法 | public DistanceVO getDistance() | |
| 前置条件 | 已创建一个Strategy领域对象 | |
| 后置条件 | 调用Strategy对象的getDistance方法 | |
| StrategyController.  getSingleDistance | 语法 | public DistanceVO getSingleDistance(String ins\_id1,String ins\_id2); | |
| 前置条件 | 已创建一个Strategy领域对象 | |
| 后置条件 | 调用Strategy对象的getSingleDistance方法 | |
| StrategyController. modifyDistance | 语法 | public ResultMessage modifyDistance(DistanceVO vo) | |
| 前置条件 | 已创建一个Strategy领域对象且VO合法 | |
| 后置条件 | 调用Strategy对象的modifyDistance方法 | |
| StrategyController. addDistance | 语法 | public ResultMessage addDistance(DistanceVO vo) | |
| 前置条件 | 已创建一个Strategy领域对象且VO合法 | |
| 后置条件 | 调用Strategy对象的addDistance方法 | |
| StrategyController. getPriceStrategy | 语法 | public PriceVO getPriceStrategy() | |
| 前置条件 | 已创建一个Strategy领域对象 | |
| 后置条件 | 调用Strategy对象的getPriceStrategy方法 | |
| StrategyController. modifyPriceStrategy | 语法 | public ResultMessage modifyPriceStrategy(PriceVO vo) | |
| 前置条件 | 已创建一个Strategy领域对象且VO合法 | |
| 后置条件 | 调用Strategy对象的modifyPriceStrategy方法 | |
| StrategyController.getRent | 语法 | public double getRent(); | |
| 前置条件 | 已创建一个Strategy领域对象 | |
| 后置条件 | 调用Strategy对象的getRent方法 | |
| StrategyController. modifyRent | 语法 | public ResultMessage  modifyRent(double rent) | |
| 前置条件 | 已创建一个Strategy领域对象 | |
| 后置条件 | 调用Strategy对象的modifyRent方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Strategy.getSalary | | | 获得单一持久化对象 |
| Strategy.modifySalary(SalaryPO po) | | | 修改单一持久化对象 |
| Strategy. getDistance() | | | 获得持久化对象 |
| Strategy.  getSingleDistance(getSingleDistance(String ins\_id1,String ins\_id2);) | | | 获得单一持久化对象 |
| Strategy. ModifyDistance(DistanceVO vo) | | | 修改单一持久化对象 |
| Strategy. addDistance(DistanceVO vo) | | | 新增单一持久化对象 |
| Strategy. getPriceStrategy | | | 获得单一持久化对象 |
| Strategy. modifyPriceStrategy(PriceVO vo) | | | 修改单一持久化对象 |
| Strategy.getRent | | | 获得单一持久化对象 |
| Strategy.modifyRent(double rent) | | | 修改单一持久化对象 |

表2.12.2-2 strategy类接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| StrategyController. getSalary | 语法 | public SalaryVO getSalary() | |
| 前置条件 |  | |
| 后置条件 | 返回薪水策略 | |
| StrategyController. ModifySalary | 语法 | public ResultMessage modifySalary(SalaryVO vo) | |
| 前置条件 | vo合法 | |
| 后置条件 | 持久化该操作 | |
| StrategyController. getDistance | 语法 | public DistanceVO getDistance() | |
| 前置条件 |  | |
| 后置条件 | 返回城市距离表 | |
| StrategyController.  getSingleDistance | 语法 | public DistanceVO getSingleDistance(String ins\_id1,String ins\_id2); | |
| 前置条件 |  | |
| 后置条件 | 返回id1与id2对应的机构所在城市距离 | |
| StrategyController. modifyDistance | 语法 | public ResultMessage modifyDistance(DistanceVO vo) | |
| 前置条件 | vo合法 | |
| 后置条件 | 持久化该操作 | |
| StrategyController. addDistance | 语法 | public ResultMessage addDistance(DistanceVO vo) | |
| 前置条件 | vo合法 | |
| 后置条件 | 持久化该操作 | |
| StrategyController. getPriceStrategy | 语法 | public PriceVO getPriceStrategy() | |
| 前置条件 |  | |
| 后置条件 | 返回运费价格策略 | |
| StrategyController. modifyPriceStrategy | 语法 | public ResultMessage modifyPriceStrategy(PriceVO vo) | |
| 前置条件 | vo合法 | |
| 后置条件 | 持久化该操作 | |
| StrategyController.getRent | 语法 | public double getRent(); | |
| 前置条件 |  | |
| 后置条件 | 返回房租价格 | |
| StrategyController. modifyRent | 语法 | public ResultMessage  modifyRent(double rent) | |
| 前置条件 |  | |
| 后置条件 | 持久化该操作 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DatabaseFactory. getStrategy | | | 取得对Strategy数据库的引用 |
| StrategyDataService. findSalaryStrategy() | | | 获得单一持久化对象 |
| StrategyDataService.modifySalaryStrategy  (SalaryPO po) | | | 修改单一持久化对象 |
| StrategyDataService. findDistance() | | | 获得单一持久化对象 |
| StrategyDataService. modifyDistance  (DistancePO po) | | | 修改单一持久化对象 |
| StrategyDataService. findPriceStrategy() | | | 获得单一持久化对象 |
| StrategyDataService. modifyPrice  (PricePO po) | | | 修改单一持久化对象 |
| StrategyDataService. findRent() | | | 获得单一持久化对象 |
| StrategyDataService.  modifyRent(double rent) | | | 修改单一持久化对象 |

### 2.12.3 strategybl模块的行为

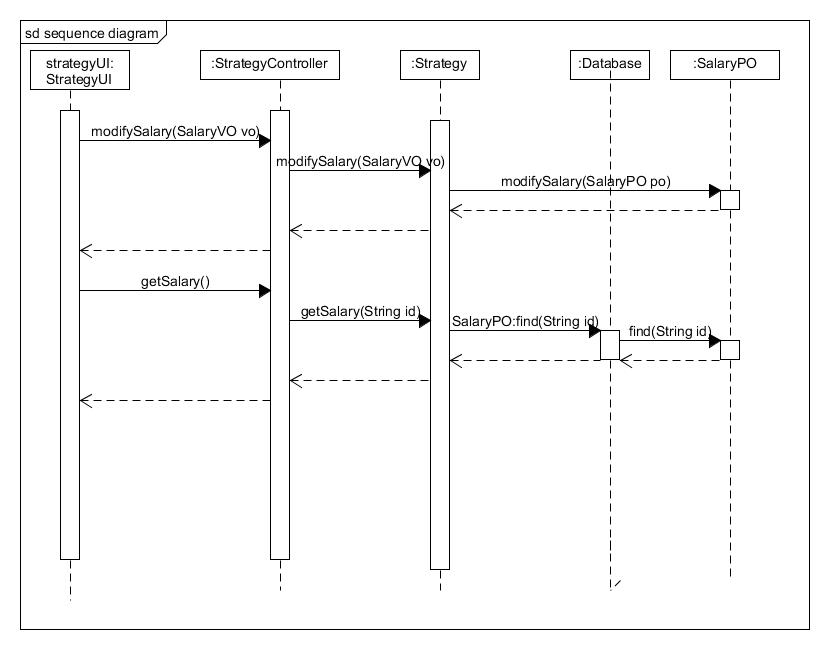


图2.12.3-1 制定薪水策略的顺序图

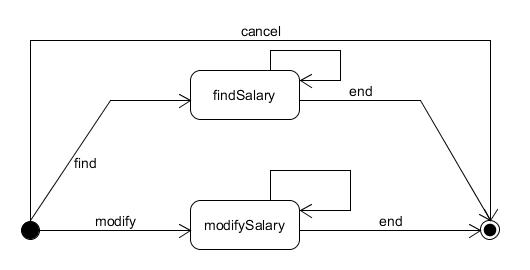


图2.12.3-2 制定薪水策略的状态图

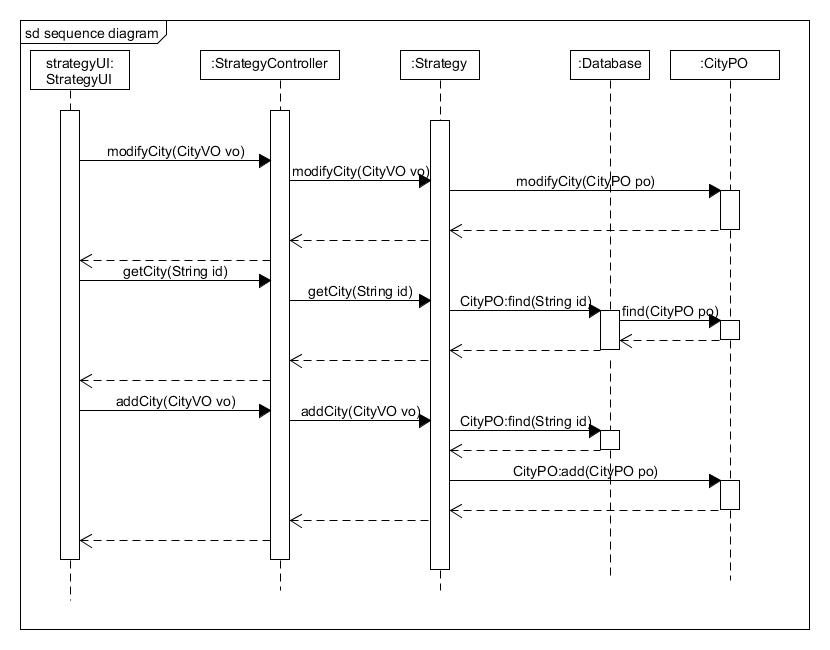


图2.12.3-3 制定城市的顺序图

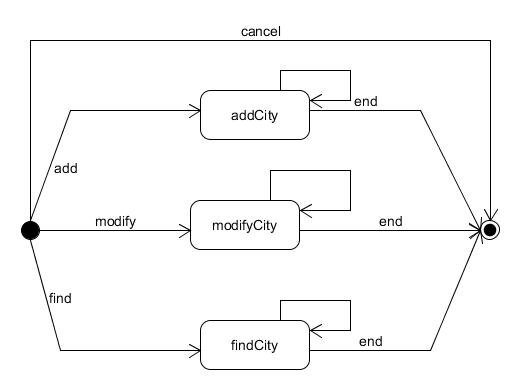


图2.12.3-4 制定城市的状态图

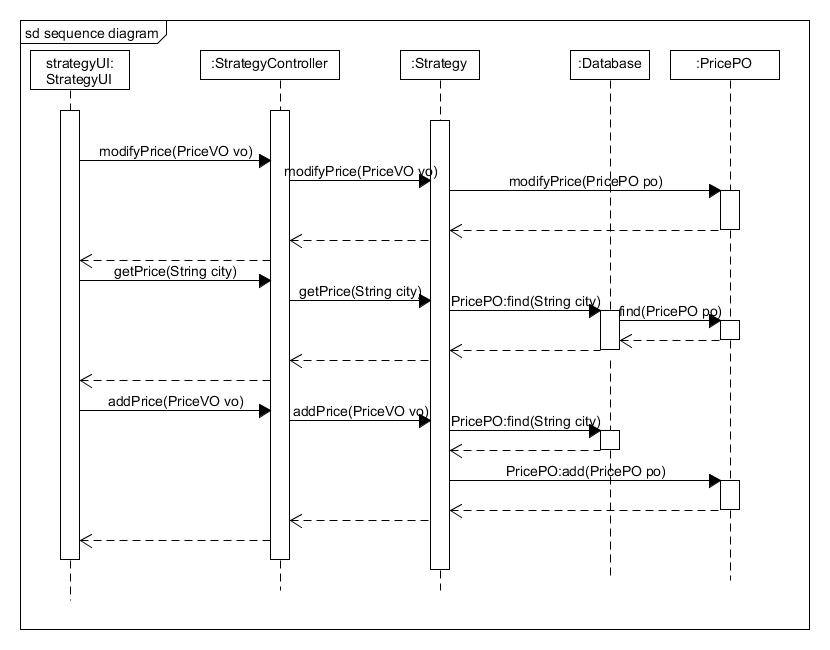


图2.12.3-5 制定价格的顺序图

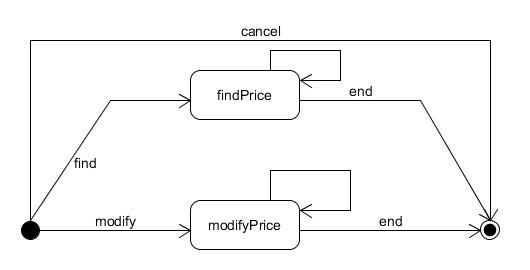


图2.12.3-6 制定价格的状态图

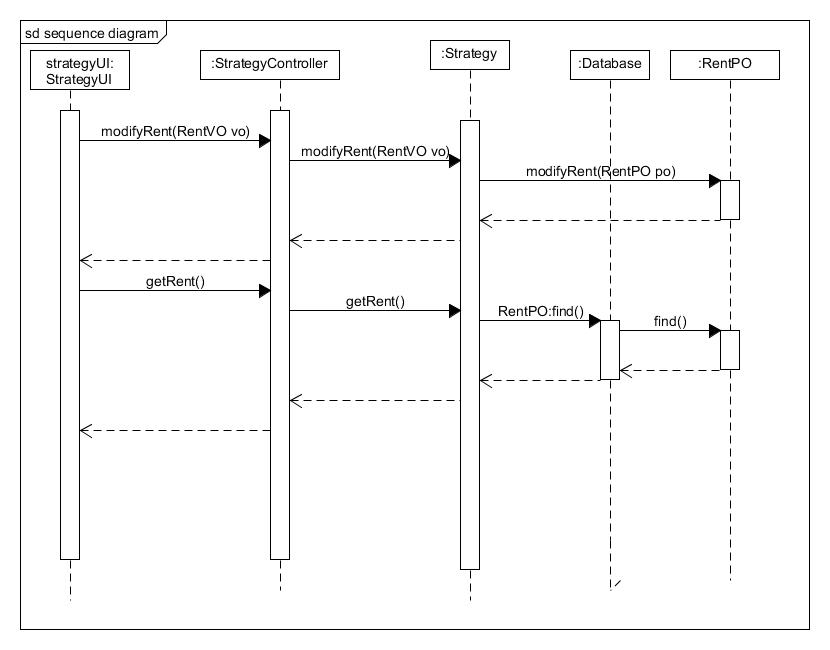


图2.12.3-7 制定租金的顺序图

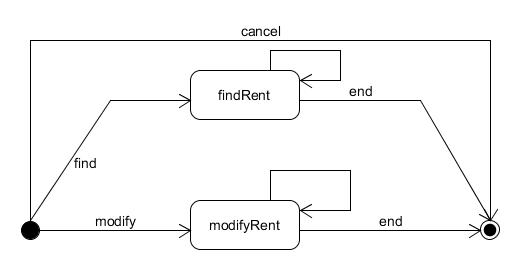


图2.12.3-8 制定租金的状态图

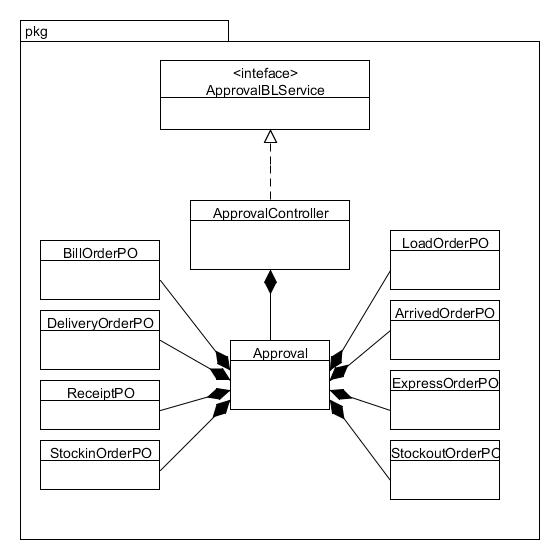
## 2.13 approvalbl模块的静态结构和动态行为

### 2.13.1 approvalbl模块局部模块的职责

approvalbl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。

approvalbl模块的职责及接口参见软件体系结构描述文档。根据体系结构的设计，采用分层风格，将系统分为展示层，业务逻辑层，数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了approvalblservice,approvaldataservice 两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了approvalController， 这样approvalController 将会将审批单据相关的业务逻辑职责和逻辑控制委托给approvalbl 对象。ReceiptPO, BillOrderPO, DeliveryOrderPO, StockinOrderPO, StockoutOrderPO, LoadOrderPO, ArrivedOrderPO, ExpressOrderPO是做为单据信息的持久化对象被添加到设计模型中的。

Approvalbl模块的设计如图2.13.1-1所示。

图2.13.1-1 approvalbl模块的各个类的设计

approvalbl模块各个类的职责如表2.13.1-1所示。

|  |  |
| --- | --- |
| 类 | 职责 |
| ApprovalController | 负责实现对应于审批单据界面所需的方法 |
| approval | 负责单据审批任务 |

### 2.13.2 approvalbl模块局部模块的接口规范

表2.13.2-1 approvalController模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ApprovalController. getSuperOrder | 语法 | | public SuperOrderPO getSuperOrder() |
| 前置条件 | | 已创建一个Approval领域对象 |
| 后置条件 | | 调用approval对象的getSuperOrder方法 |
| ApprovalController. modifySuperOrderState | 语法 | | public ResultMessage  setSuperOrderState(Boolean pass) |
| 前置条件 | | 已创建一个Approval领域对象 |
| 后置条件 | | 调用approval对象的getSuperOrder方法 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Approval.getSuperOrder | | 获得单一持久化对象 | |
| Approval.  modifySuperOrderState | | 修改单一持久化对象 | |

表2.14.2-2 approval模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Approval. getSuperOrder | 语法 | | public SuperOrderPO getSuperOrder() |
| 前置条件 | |  |
| 后置条件 | | 返回单据 |
| Approval. modifySuperOrderState | 语法 | | public ResultMessage  setSuperOrderState(Boolean pass) |
| 前置条件 | |  |
| 后置条件 | | 持久化该操作 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| ApprovalDataService. findOrderPO() | | 获得单一持久化对象 | |
| ApprovalDataService. modifyOrderPO  (OrderPO po) | | 修改单一持久化对象 | |
| Approval. findExpressOrderPO() | | 获得单一持久化对象 | |
| ApprovalDataService. modifyExpressOrderPO  (ExpressOrderPO po) | | 修改单一持久化对象 | |
| ApprovalDataService. findReceiptPO() | | 获得单一持久化对象 | |
| ApprovalDataService. modifyReceiptPO  (ReceiptPO po) | | 修改单一持久化对象 | |

### 2.13.3 approvalbl模块的行为

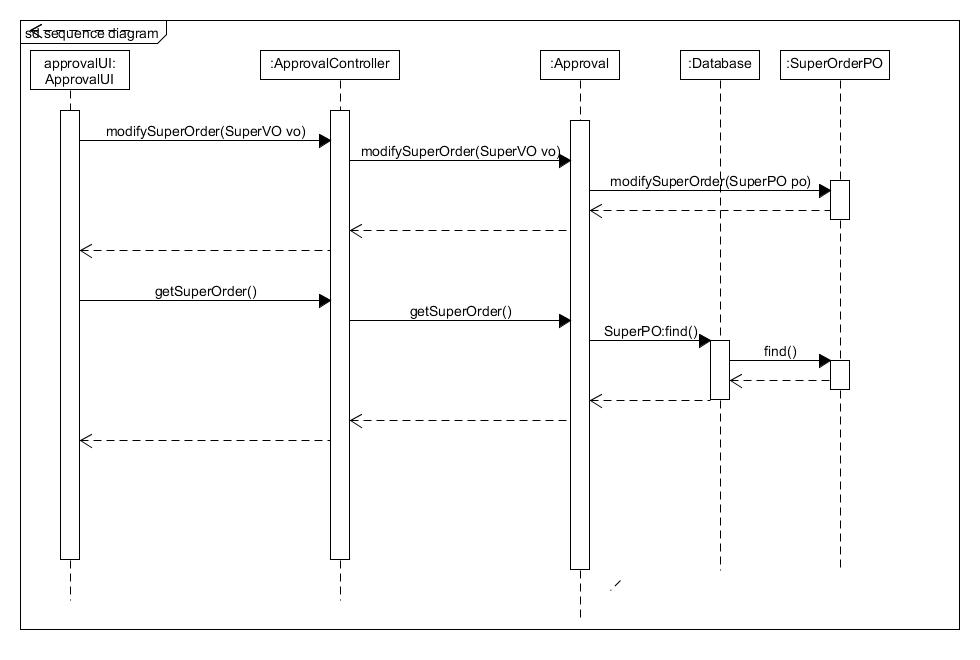


图2.13.3-1 单据审批的顺序图

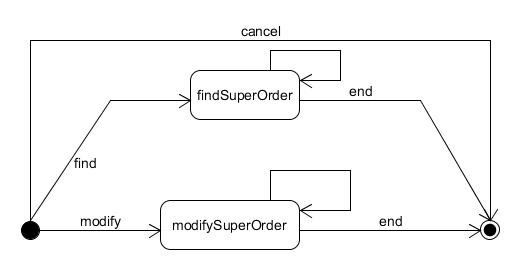


图2.13.3-2 单据审批的状态图

## 2.14 userbl模块的静态结构和动态行为

### 2.14.1 userbl模块局部模块的职责

userbl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。

userbl模块的职责及接口参见软件体系结构描述文档。根据体系结构的设计，采用分层风格，将系统分为展示层，业务逻辑层，数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了userblservice,userdataservice 两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了userController，这样userController 将会将用户管理相关的业务逻辑职责和逻辑控制委托给userbl 对象。userPO是做为用户信息的持久化对象被添加到设计模型中的。

userbl模块的设计如图2.14.1-1所示。

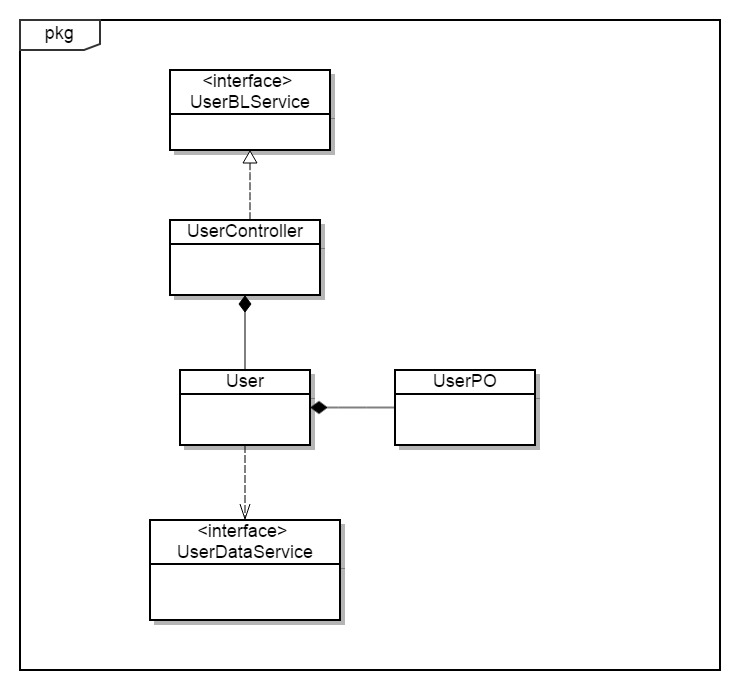


图2.14.1-1 userbl模块的各个类的设计

userbl模块各个类的职责如表2.14.1-1所示。

|  |  |
| --- | --- |
| 类 | 职责 |
| UserController | 负责实现对应于用户管理界面所需的方法 |
| User | 负责管理用户 |

### 2.14.2 userbl模块局部模块的接口规范

表2.14.2-1 userController模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| userController. getAll | 语法 | | Public List<UserVO> getAll() |
| 前置条件 | | 已创建一个User领域对象 |
| 后置条件 | | 调用user对象的getAll方法 |
| userController. getSingle | 语法 | | Public UserVO getSingle(int id) |
| 前置条件 | | 已创建一个User领域对象 |
| 后置条件 | | 调用user对象的getSingle方法 |
| userController. add | 语法 | | Public ResultMessage add(UserVO newUserVO) |
| 前置条件 | | 已创建一个User领域对象且VO合法 |
| 后置条件 | | 调用user对象的add方法 |
| userController. modify | 语法 | | Public ResultMessage modify(UserVO userVO) |
| 前置条件 | | 已创建一个User领域对象且VO合法 |
| 后置条件 | | 调用user对象的modify方法 |
| userController. delete | 语法 | | Public ResultMessage delete(int id) |
| 前置条件 | | 已创建一个User领域对象 |
| 后置条件 | | 调用user对象的delete方法 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| User. getAll | | 获得持久化对象 | |
| User. getSingle | | 根据id查找单一持久化对象 | |
| User.add | | 新增单一持久化对象 | |
| User.modify | | 修改单一持久化对象 | |
| User.delete | | 删除单一持久化对象 | |

表2.14.2-2 user模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User. getAll | 语法 | | Public List<UserVO> getAll() |
| 前置条件 | | 无 |
| 后置条件 | | 返回所有的用户信息 |
| User. getSingle | 语法 | | Public UserVO getSingle(int id) |
| 前置条件 | | id>=0&&id<userVO.length |
| 后置条件 | | 返回该ID的user信息 |
| User. add | 语法 | | Public ResultMessage add(UserVO newUserVO) |
| 前置条件 | | 输入的ID不存在，password符合规范 |
| 后置条件 | | 新增该持久化对象 |
| User. modify | 语法 | | Public ResultMessage modify(UserVO userVO) |
| 前置条件 | |  |
| 后置条件 | | 持久化该操作 |
| User. delete | 语法 | | Public ResultMessage delete(int id) |
| 前置条件 | | Id>=0&&id<userVO.length |
| 后置条件 | | 持久化该操作 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| user\_data\_service.  add(UserPO po) | | 新增单一持久化对象 | |
| user\_data\_service. modify(UserPO po) | | 修改单一持久化对象 | |
| user\_data\_service.  delete(int id) | | 删除单一持久化对象 | |
| user\_data\_service.  find(int id) | | 根据id查找单一持久化对象 | |

### 2.14.3 userbl模块的行为

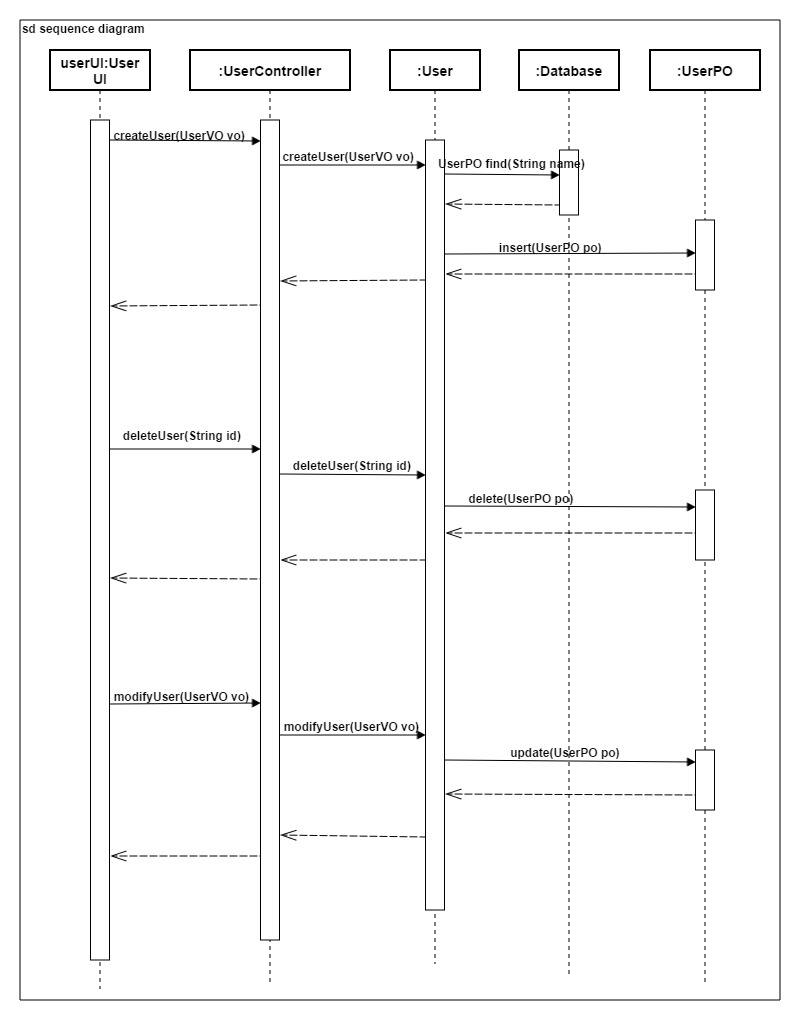


图2.14.3-1 用户管理的顺序图

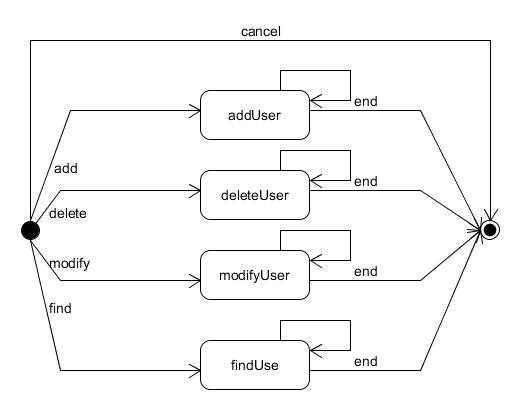
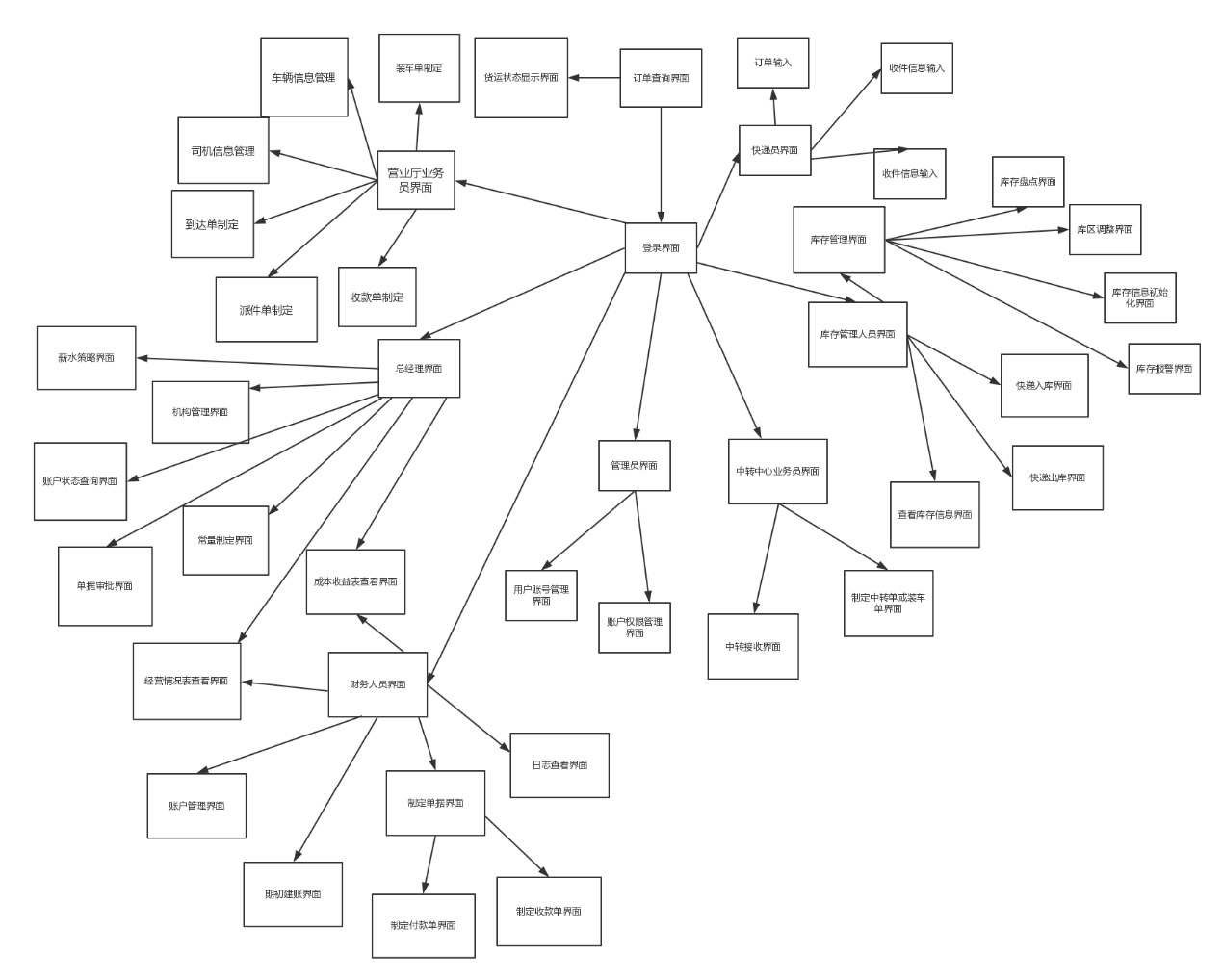


图2.14.3-2 用户管理的状态图

## 2.15 用户界面层的行为

### 2.15.1 用户界面层的跳转

2.15.2 用户界面层的动态行为

### 2.15.2用户界面层的跳转顺序图示例

用户界面的跳转各个人员基本相同，以下给出几个示例，不再赘述

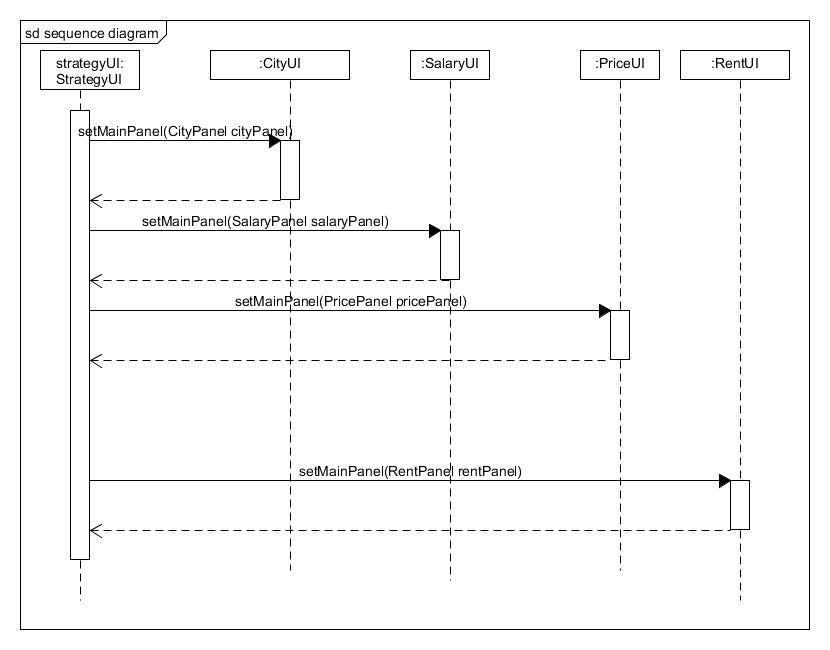


图2.15.2.1 总经理界面跳转顺序图

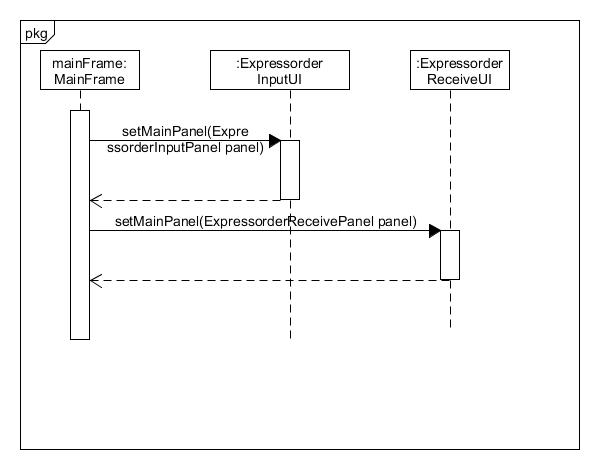


图2.15.2.2 快递员界面跳转顺序图

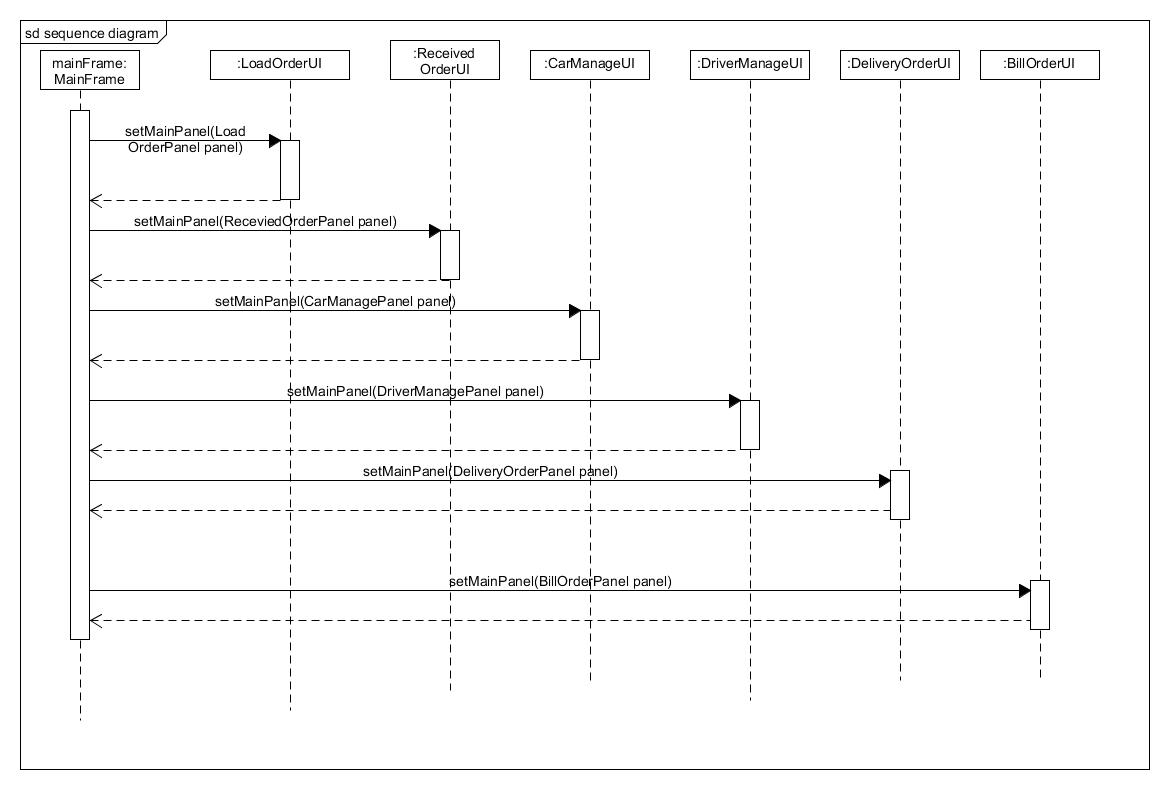


图2.15.2.3 营业厅业务员界面跳转顺序图

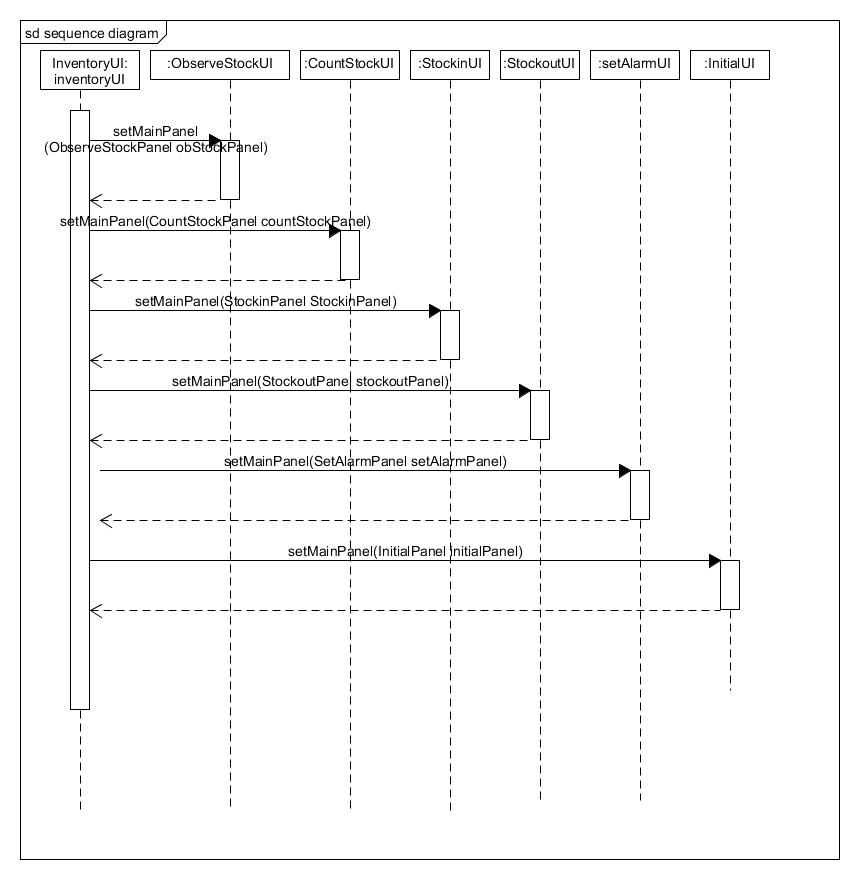


图2.15.2.4 库存管理人员界面跳转顺序图

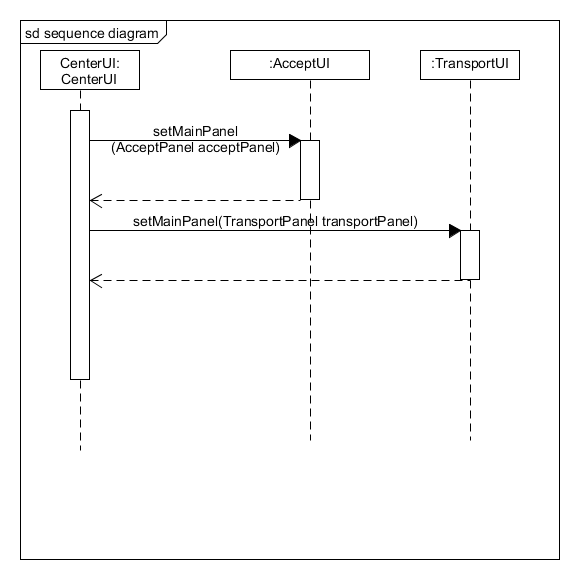


图2.15.2.4中心业务员界面跳转顺序图