快递公司管理系统

软件详细设计描述文档

2015/11/1

南京大学软件学院

王新宇 孙康 宋子微 孙超

1.引言

1.1编制目的

本报告详细完成对快递公司管理系统的详细设计，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。

1.2参考资料

·项目实践v4

·快递公司管理系统用例文档v2.0

·快递公司管理系统软件需求规格说明文档

·软件公司管理系统软件体系结构描述文档v1.1

2.产品概述

参考快递公司管理系统用例文档和快递公司管理系统软件需求规格说明文档中对产品的概括描述。

3.体系结构设计概述

参考快递公司管理系统概要设计文档中对体系结构设计的概述。

4.结构视角

4.1业务逻辑层的分解

业务逻辑层的开发包图参见软件体系结构文档图3。

4.1.1Logisticsbl模块

(1)模块概述

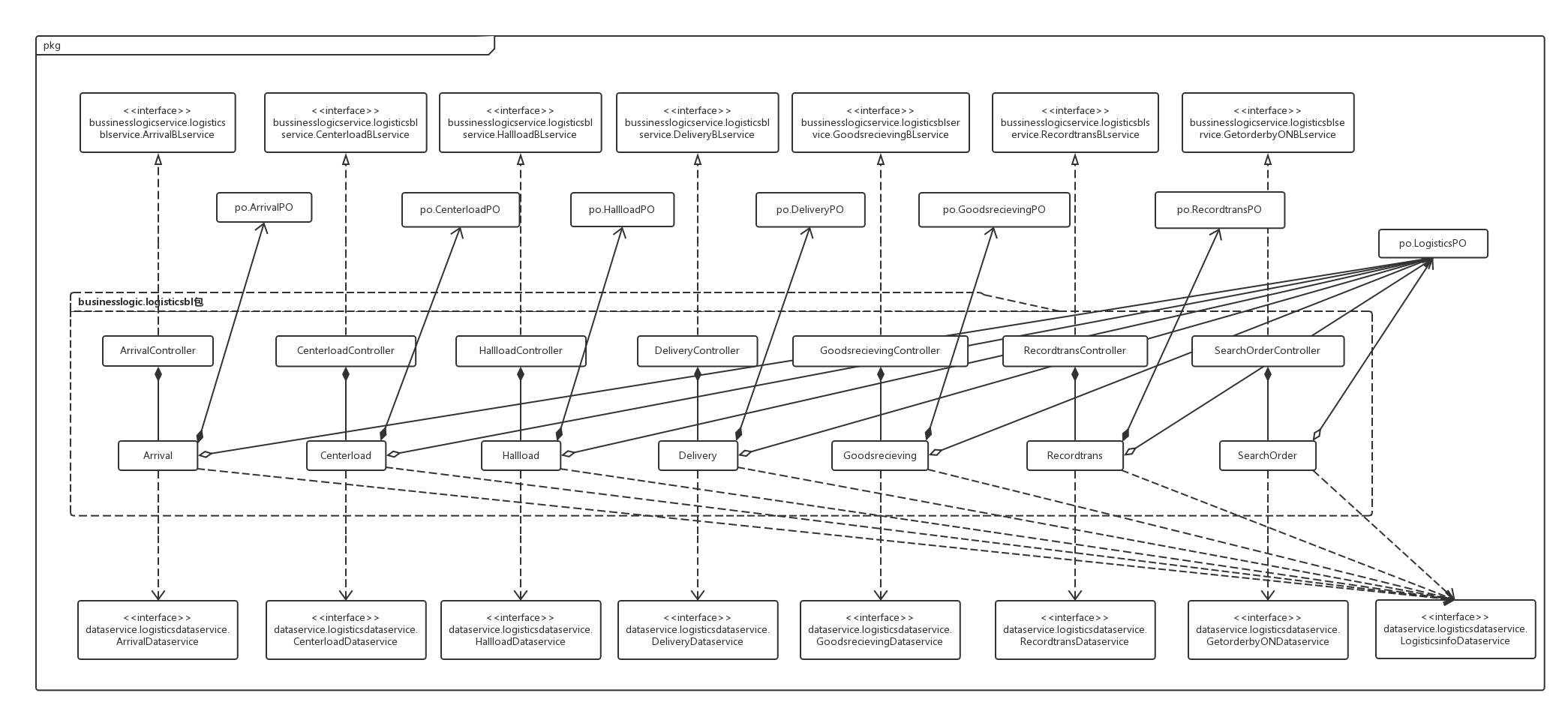
Logisticsbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Logisticsbl模块的职责及接口参见软件系统结构描述文档表？。

(2)整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和逻辑层之间，我们添加businiesslogicservice.logisticsblservice.Arrivalblservice接口。业务逻辑层和数据层之间添加dataservice.logistcsdataservice.Arrivaldataservice接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了ArrivalControler,这样ArrivalController会将对到达任务的业务逻辑职责和逻辑控制职责处理委托给Arrival对象。ArrivalPO是作为到达单记录的持久化对象被添加到设计模型中去的。

Logisticsbl模块设计如图



Logisticsbl模块各类职责如表？所示

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登录界面所需的服务 |
| ArrivalController | 负责实现有关到达单界面所需要的服务 |
| CenterloadController | 负责实现有关中转中心装车单界面所需要的服务 |
| HallloadController | 负责实现有关营业厅装车单界面所需要的服务 |
| DeliveryController | 负责实现有关派件单界面所需要的服务 |
| GoodsrecievingController | 负责实现有关收货单界面所需要的服务 |
| RecordtransController | 负责实现有关中转单界面所需要的服务 |
| SearchOrderController | 负责实现有关营业厅物流信息查询界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的姓名和密码，可以解决登录问题 |
| Arrival | 负责控制到达单的增加，修改 ，查询 |
| AddArrival | 负责到达单的增加 |
| UpdateArrival | 负责到达单的更新 |
| FindArrival | 负责到达单的批量查询 |
| UpdateLogistics | 负责到达单到达时的物流货运状态和轨迹的更新 |
| IdGet | 负责生成到达单的id |
| Centerload | 负责中转中心装车单的增加，更新，查询 |
| AddCenterload | 负责中转中心装车单的增加 |
| UpdateCenterload | 负责中转中心装车单的更新 |
| FindCenterload | 负责中转中心装车单的批量查询 |
| Idget | 负责得到中转中心装车单的id |
| CenterloadUpdateLogistics | 负责中转中心装车单装车时的更新货运状态和物流轨迹 |
| FeeCounter | 负责中转中心装车时的计算运费 |
| Hallload | 负责营业厅装车单的增加，更新，查询的控制 |
| AddLoad | 负责营业厅装车单的增加 |
| UpdateLoad | 负责营业厅装车单的更新 |
| FindLoad | 负责营业厅装车单的批量查找 |
| HallLloadUpdateLogistics | 负责营业厅装车时的更新货运状态和历史轨迹 |
| Idget | 负责营业厅装车单得到id |
| HallFeeCount | 负责营业厅装车时得到运费 |
| Delivery | 负责派件单的增加，查询，更新的控制 |
| AddDelivery | 负责派件单的增加 |
| UpdateDelivery | 负责派件单的更新 |
| FindDelivery | 负责派件单的批量查询 |
| Idget | 负责派件单id的得到 |
| Goodsrecieving | 负责收货单的增加，更新，查询的控制 |
| AddGoods | 负责收货单的增加 |
| FindGoods | 负责收货单的批量查询 |
| UpdateGoods | 负责收货单的更新 |
| GoodsRevUpdateLogistics | 负责收货时的更新货运状态和历史轨迹 |
| Idget | 负责收货时id的得到 |
| Recordtrans | 负责中转单的增加，更新，批量查询的控制 |
| AddTrans | 负责中转单的增加 |
| FindTrans | 负责中转单的批量查询 |
| UpdateTrans | 负责中转单的更新 |
| Idget | 负责中转单id的得到 |
| RecordtransUpdateLogistics | 负责中转时的更新货运状态和历史轨迹 |
| FeeCount | 负责中转时得到运费 |
| SearchOrder | 物流信息查询的领域模型对象，拥有物流信息查询的所有数据，提交单据，修改订单状态 |
| CheckForExistBl | 用于中转单，营业厅装车单，中转中心装车单的检查 |

表11 LogisticsController模块接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | |
| SearchOrderController. getOrderbyON | 语法 | | | public OrderVO getOrderbyON(String orderNumber) |
| 前置条件 | | | 已创建一个SearchOrder领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用SearchOrder领域的getOrderbyON方法 |
| HallloadController. addHallLoad | 语法 | | | **public** ResultMessage update(HallLoadVO vo) |
| 前置条件 | | | 已创建一个HallLoad领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用HallLoad领域的update方法 |
| HallloadController. update | 语法 | | | public ResultMessage addHallLoadForm (HallLoadVO vo) |
| 前置条件 | | | 已创建一个HallLoad领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用HallLoad领域的addHallLoadForm方法 |
| HallloadController. ComputeHLfare | 语法 | | | public double computeHLfare (HLFareVO vo) |
| 前置条件 | | | 已创建一个HallLoad领域对象，并且已输入所有计算运费所需的数据 |
| 后置条件 | | | 调用HallLoad领域的computeHLfare方法 |
| HallloadController. findforms | 语法 | | | Public List<HallLoadVO> findforms(Formstate state,String orgcode); |
| 前置条件 | | | 已创建HallLoad领域对象，并且输入符合规则 |
| 后置条件 | | | 调用HallLoad领域的findforms方法 |
| HallloadController. getid | 语法 | | | public String getid(String orgcode, long date) ; |
| 前置条件 | | | 输入符合规则 |
| 后置条件 | | | 调用HallLoad领域的getid方法 |
| GoodsrecievingController.  GoodsReceving | 语法 | | | public ResultMessage GoodsReceving (GoodsRecevingVO vo) |
| 前置条件 | | | 已创建一个GoodsReceving领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用GoodsReceving领域的GoodsReceving方法 |
| GoodsrecievingController.  findforms | 语法 | | | public List<GoodsReceivingVO> findform(Formstate state,String orgcode) ; |
| 前置条件 | | | 已创建GoodsReceving领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用GoodsReceving领域的findforms方法 |
| GoodsrecievingController.  updateFromHall | 语法 | | | public ResultMessage updateFromHall(GoodsReceivingVO vo,String orgname); |
| 前置条件 | | | 已创建一个GoodsReceving领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用GoodsReceving领域的updatefromhall方法 |
| GoodsrecievingController.  updateFromCenter | 语法 | | | public ResultMessage updateFromCenter(GoodsReceivingVO vo,String orgname); |
| 前置条件 | | | 已创建一个GoodsReceving领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用GoodsReceving领域的updateFromCenter方法 |
| GoodsrecievingController.  getid | 语法 | | | public String getid(String orgcode) ; |
| 前置条件 | | | 输入符合语法规则 |
| 后置条件 | | | 调用GoodsReceving领域的ugetid方法 |
| DeliveryController.  Delivery | 语法 | | | public ResultMessage Delivery(DeliveryVO vo) |
| 前置条件 | | | 已创建一个Delivery领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用Delivery领域的Delivery方法 |
| DeliveryController.  find | 语法 | | | public List<DeliveryVO> find(Formstate state,String orgcode) ; |
| 前置条件 | | | 输入符合输入规则 |
| 后置条件 | | | 调用Delivery领域的find方法 |
| DeliveryController.  findID | 语法 | | | public String findID(String orgcode) ; |
| 前置条件 | | | 输入符合输入规则 |
| 后置条件 | | | 调用Delivery领域的findID方法 |
| DeliveryController.  update | 语法 | | | public String findID(String orgcode) ; |
| 前置条件 | | | 已经创建一个领域对象，输入符合输入规则 |
| 后置条件 | | | 调用Delivery领域的update方法 |
| ArrivalController.  addArrival | 语法 | | | public ResultMessage addArrival(ArrivalVO vo) |
| 前置条件 | | | 已创建一个Arrival领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用Arrival领域的addArrival方法 |
| ArrivalController.  search | 语法 | | | public List<ArrivalVO> search(String orgcode,Formstate state) ; |
| 前置条件 | | | 输入符合输入规则 |
| 后置条件 | | | 调用Arrival领域的search方法 |
| ArrivalController.  updateFromHall | 语法 | | | public ResultMessage updateFromHall(ArrivalVO vo,String orgname); |
| 前置条件 | | | 已创建一个Arrival领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用Arrival领域的updateFromHall方法 |
| ArrivalController.  updateFromCenter | 语法 | | | public ResultMessage updateFromCenter(ArrivalVO vo,String orgname); |
| 前置条件 | | | 已创建一个Arrival领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用Arrival领域的updateFromCenter方法 |
| ArrivalController.  getid | 语法 | | | **public** String getid(String organizationcode) ; |
| 前置条件 | | | 输入符合输入规则 |
| 后置条件 | | | 调用Arrival领域的getid方法 |
| RecordtransController.  Recordtrans | 语法 | | | public ResultMessage Recordtrans (RecordtransVO vo) |
| 前置条件 | | | 已创建一个Recordtrans领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用Recordtrans领域的Recordtrans方法 |
| RecordtransController.  update | 语法 | | | public ResultMessage update (RecordtransVO vo) |
| 前置条件 | | | 已创建一个Recordtrans领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用Recordtrans领域的update方法 |
| RecordtransController.  find | 语法 | | | public List<RecordtransVO> find(Formstate state,String orgcode) ; |
| 前置条件 | | | 已创建Centerload领域对象，并且输入符合输入规则 |
| 后置条件 | | | 调用Centerload领域的find方法 |
| RecordtransController.  getid | 语法 | | | public String getid(String orgcode,long data); |
| 前置条件 | | | 输入符合输入规则 |
| 后置条件 | | | 调用Centerload领域的getid方法 |
| RecordtransController.  getfee | 语法 | | | public double getfee(List<String> allbarcode,String city1,String city2,String transporttype); |
| 前置条件 | | | 输入符合输入规则 |
| 后置条件 | | | 调用Centerload领域的getfee方法 |
| CenterloadController. addCenterLoadForm | 语法 | | | public ResultMessage addCenterLoadForm(CenterloadVO vo); |
| 前置条件 | | | 已经创建了centerload领域对象，并且输入符合规则 |
| 后置条件 | | | 调用Centerload领域的addCenterLoadForm方法 |
| CenterloadController. findCenterLoadForm | 语法 | | | public List<CenterloadVO> findCenterloadForm(Formstate state,String orgcode) ; |
| 前置条件 | | | 输入符合规则 |
| 后置条件 | | | 调用Centerload领域的findCenterLoadForm方法 |
| CenterloadController. update | 语法 | | | public ResultMessage update(CenterloadVO vo); |
| 前置条件 | | | 已经创建了centerload领域对象，并且输入符合规则 |
| 后置条件 | | | 调用Centerload领域的update方法 |
| CenterloadController. getid | 语法 | | | public String getid(String orgcode,long date) ; |
| 前置条件 | | | 输入符合规则 |
| 后置条件 | | | 调用Centerload领域的getid方法 |
| CenterloadController. getfee | | 语法 | | public double getfee(List<String> allbarcode ,String city1,String city2); | |
| 前置条件 | | 输入符合规则 | |
| 后置条件 | | 调用Centerload领域的getfee方法 | |
| 需要的服务(需接口) | | | | |
| Logistics.getOrderbyON | | | 查询订单 | |
| Logistics.HallLoad | | | 录入一个装车单 | |
| Logistics. ComputeHLfare | | | 计算营业厅装车单运费 | |
| Logistics.GoodsReceving | | | 录入一个接货单 | |
| Logistics.Delivery | | | 录入一个派件单 | |
| Logistics.Arrival | | | 录入一个到达单 | |
| Logistics.Recordtrans | | | 录入一个中转单 | |
| Logistics.Centerload | | | 录入一个中转中心装车单 | |
| Logistics. ComputeCLfare | | | 计算中转中心装车单运费 | |

表11 logistics接口规范

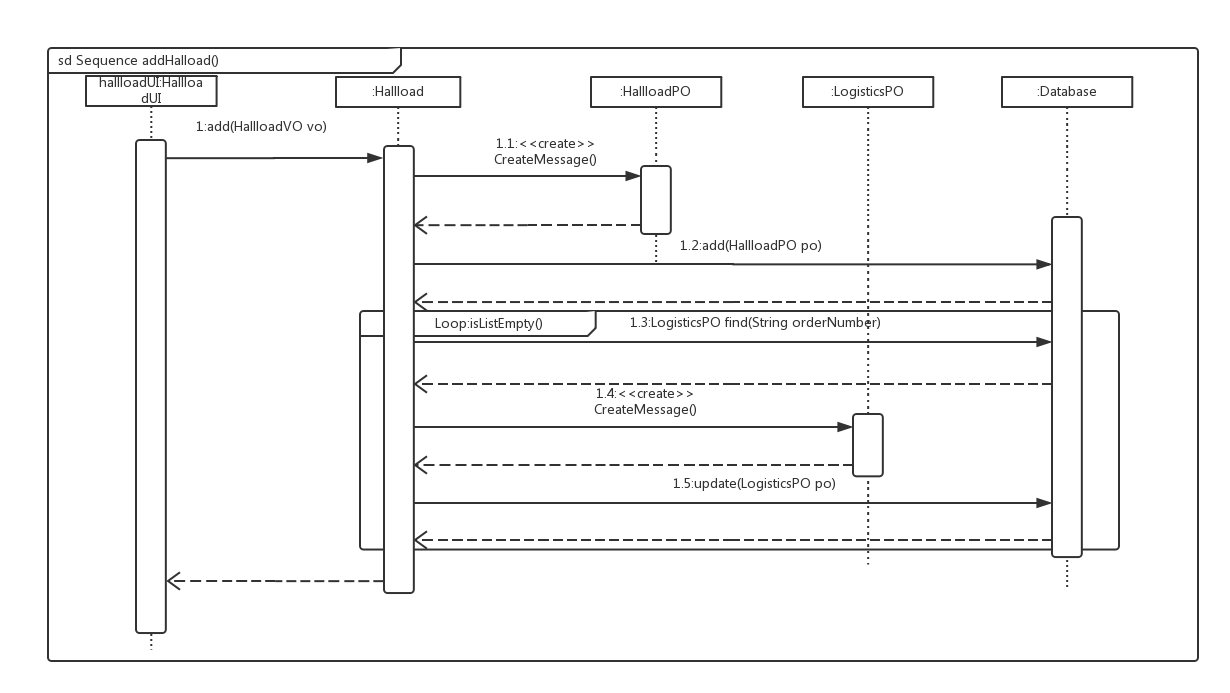
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| SearchOrder. getOrderbyON | 语法 | | public OrderVO getOrderbyON(String orderNumber) |
| 前置条件 | | 数据库中存在响应的order数据 |
| 后置条件 | | 返回调用的ordervo |
| Hallload. update | 语法 | | **public** ResultMessage update(HallLoadVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 数据库中更新入hallload数据 |
| Hallload. addHallLoadForm | 语法 | | public ResultMessage addHallLoadForm (HallLoadVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 数据库中加入新的hallload数据 |
| Hallload. ComputeHLfare | 语法 | | public double computeHLfare (HLFareVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 返回相应的运费信息 |
| Hallload. Findforms | 语法 | | Public List<HallLoadVO> findforms(Formstate state,String orgcode); |
| 前置条件 | | 数据库中有多个hallload数据 |
| 后置条件 | | 返回符合条件的hallload数据 |
| Hallload. Getid | 语法 | | public String getid(String orgcode, long date) ; |
| 前置条件 | | 数据库中有相应的id |
| 后置条件 | | 返回id |
| Goodsrecieving.  GoodsReceving | 语法 | | public ResultMessage GoodsReceving (GoodsRecevingVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 向数据库中加入goosdreceving信息 |
| Goodsrecieving.  findforms | 语法 | | public List<GoodsReceivingVO> findform(Formstate state,String orgcode) ; |
| 前置条件 | | 数据库中已经有goodsreceving数据 |
| 后置条件 | | 返回数据库中符合条件的goodsreceving数据 |
| Goodsrecieving.  updateFromHall | 语法 | | public ResultMessage updateFromHall(GoodsReceivingVO vo,String orgname); |
| 前置条件 | | 数据库中有相对应的goodsreceving数据 |
| 后置条件 | | 向数据库中更新响应的goodsreceving数据 |
| Goodsrecieving.  updateFromCenter | 语法 | | public ResultMessage updateFromCenter(GoodsReceivingVO vo,String orgname); |
| 前置条件 | | 数据库中有相对应的goodsreceving数据 |
| 后置条件 | | 向数据库中更新相应的goodsreceving数据 |
| Goodsrecieving.  getid | 语法 | | public String getid(String orgcode) ; |
| 前置条件 | | 数据库中有 相对应的id存储 |
| 后置条件 | | 返回相对应的id |
| Delivery.  Delivery | 语法 | | public ResultMessage Delivery(DeliveryVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 向数据库中加入delivery数据 |
| Delivery.  find | 语法 | | public List<DeliveryVO> find(Formstate state,String orgcode) ; |
| 前置条件 | | 无 |
| 后置条件 | | 返回数据库中存在的响应的delivery数据 |
| Delivery.  findID | 语法 | | public String findID(String orgcode) ; |
| 前置条件 | | 数据库中有相应的id存储 |
| 后置条件 | | 返回当前delivery的id |
| Delivery.  update | 语法 | | public String findID(String orgcode) ; |
| 前置条件 | | 数据库中有当前的对象数据存储 |
| 后置条件 | | 把新的delivery数据更新入数据库 |
| Arrival.  addArrival | 语法 | | public ResultMessage addArrival(ArrivalVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 将arrival数据加入数据库 |
| Arrival.  search | 语法 | | public List<ArrivalVO> search(String orgcode,Formstate state) ; |
| 前置条件 | | 无 |
| 后置条件 | | 返回数据库中相应的arrival数据 |
| Arrival.  updateFromHall | 语法 | | public ResultMessage updateFromHall(ArrivalVO vo,String orgname); |
| 前置条件 | | 数据库中有相应的arrival对象 |
| 后置条件 | | 向数据库中加入响应的arrival数据 |
| Arrival.  updateFromCenter | 语法 | | public ResultMessage updateFromCenter(ArrivalVO vo,String orgname); |
| 前置条件 | | 数据库中有相应的arrival对象 |
| 后置条件 | | 向数据库中加入相应的arrival数据 |
| Arrival.  getid | 语法 | | **public** String getid(String organizationcode) ; |
| 前置条件 | | 数据库中有arrival的id存储 |
| 后置条件 | | 返回arrival当前的id |
| Recordtrans.  Recordtrans | 语法 | | public ResultMessage Recordtrans (RecordtransVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 向数据库中加入recordtrans数九 |
| Recordtrans.  update | 语法 | | public ResultMessage update (RecordtransVO vo) |
| 前置条件 | | 数据库中有相应的一个recordtrans数据 |
| 后置条件 | | 向数据库中更新响应的recordtrans数据 |
| Recordtrans.  find | 语法 | | public List<RecordtransVO> find(Formstate state,String orgcode) ; |
| 前置条件 | | 无 |
| 后置条件 | | 返回数据库中有的相应的recordtrans数据 |
| Recordtrans.  getid | 语法 | | public String getid(String orgcode,long data); |
| 前置条件 | | 数据库中存有recordtrans的id数据 |
| 后置条件 | | 返回数据库中recordtrans的当前id |
| Recordtrans.  getfee | 语法 | | public double getfee(List<String> allbarcode,String city1,String city2,String transporttype); |
| 前置条件 | | 无 |
| 后置条件 | | 返回运费 |
| Centerload. addCenterLoadForm | 语法 | | public ResultMessage addCenterLoadForm(CenterloadVO vo); |
| 前置条件 | | 无 |
| 后置条件 | | 向数据库 中加入新的centerlaod数据 |
| Centerload. findCenterLoadForm | 语法 | | public List<CenterloadVO> findCenterloadForm(Formstate state,String orgcode) ; |
| 前置条件 | | 无 |
| 后置条件 | | 返回数据中相应的centerload数九 |
| Centerload. Update | 语法 | | public ResultMessage update(CenterloadVO vo); |
| 前置条件 | | 数据库中存在相应的centerlaod数据 |
| 后置条件 | | 将新的centerlaod数据加入数据库中 |
| Centerload. Getid | 语法 | | public String getid(String orgcode,long date) ; |
| 前置条件 | | 数据库中存在centerload的相应id |
| 后置条件 | | 返回centerload 的id |
| Centerload. Getfee | | 语法 | public double getfee(List<String> allbarcode ,String city1,String city2); | |
| 前置条件 | 无 | |
| 后置条件 | 返回运费 | |

|  |
| --- |
| 需要的服务(需接口) |

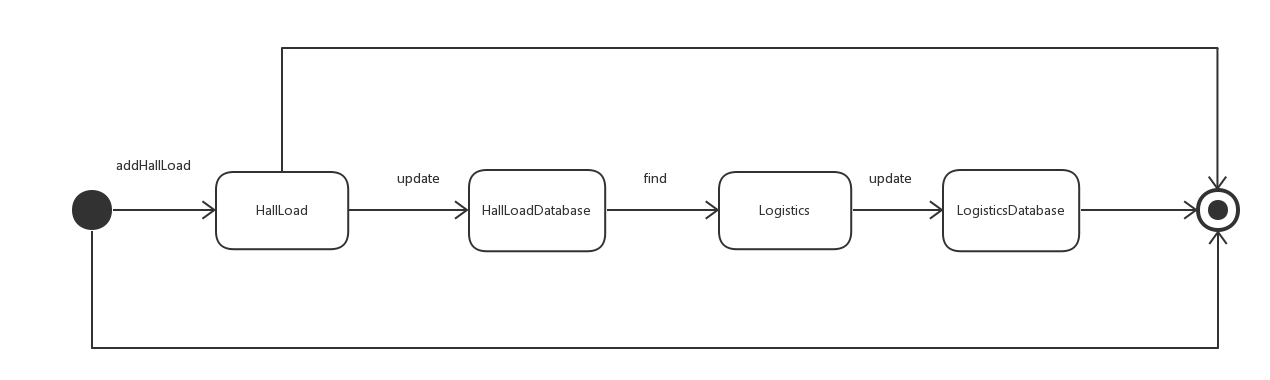
|  |  |
| --- | --- |
| DatabaseFactory. getArrivalform | 得到中转中心到达单数据库的引用 |
| DatabaseFactory. getCenterload | 得到中转中心装车单数据库的引用 |
| DatabaseFactory. getDelivery | 得到派件单数据库的引用 |
| DatabaseFactory. getGoodsreceving | 得到营业厅到达单数据库的引用 |
| DatabaseFactory. getHallload | 得到营业厅装车单数据库的引用 |
| DatabaseFactory. getLogisticsinfo | 得到物流信息数据库的引用 |
| DatabaseFactory. getRecordtrans | 得到中转单数据库的引用 |
| ArrivalFormDataService. getArrivalForm (Formstate state,String orgcode) | 根据单据状态和机构编号获取中转中心到达单的list |
| ArrivalFormDataService. updateArrivalForm (ArrivalPO po) | 更新数据库中的ArrivalPO对象 |
| ArrivalFormDataService. addArrivalForm (ArrivalPO po) | 向数据库中增加ArrivalPO对象 |
| ArrivalFormDataService. findLastID (String orgcode) | 根据机构编号获取ArrivalPO的最新编号 |
| ArrivalFormDataService. getHallLoadBycode(String hallloadcode) | 根据营业厅装车单编号获取营业厅装车单 |
| ArrivalFormDataService. getRecordtransBycode(String recordtranscode) | 根据中转单编号获取中转单 |
| CenterLoadFormDataService. getCenterLoadForm (Formstate state,String orgcode) | 根据单据状态和机构编号获取中转中心装车单单的list |
| CenterLoadFormDataService. updateCenterLoadForm (CenterloadPO po) | 更新数据库中的CenterloadPO对象 |
| CenterLoadFormDataService. addCenterLoadForm (CenterloadPO po) | 向数据库中增加CenterloadPO对象 |
| CenterLoadFormDataService. findLastmotorcode(String orgcode,long date) | 根据机构编号和日期获取CenterloadPO的最新编号 |
| CenterLoadFormDataService. checkCenterLoad(String motorcode) | 检查编号为motorcode的CenterloadPO是否存在 |
| CenterLoadFormDataService. getunitprice(String transporttype) | 根据运输方式获取单位距离的运费 |
| CenterLoadFormDataService. getdistance(String city1,String city2) | 根据城市名获取两城市之间的距离 |
| CenterLoadFormDataService. getweight(String ordernum) | 根据订单号获取该订单中货物的重量 |
| DeliveryFormDataService. getDeliveryForm (Formstate state,String orgcode) | 从数据库中得到RecordtransPO对象 |
| DeliveryFormDataService. updateDeliveryForm (DeliveryPO po) | 根据单据状态和机构编号获取派件单单的list |
| DeliveryFormDataService. addDeliveryForm (DeliveryPO po) | 向数据库中增加DeliveryPO对象 |
| DeliveryFormDataService. findLastID(String orgcode) | 根据机构编号获取DeliveryPO的最新编号 |
| GoodsReceivingFormDataService. getGoodsReceivingForm (Formstate state,String orgcode) | 根据单据状态和机构编号获取营业厅到达单的list |
| GoodsReceivingFormDataService. updateGoodsReceivingForm (GoodsReceivingPO po) | 更新数据库中的GoodsReceivingPO对象 |
| GoodsReceivingFormDataService. addGoodsReceivingForm (GoodsReceivingPO po) | 向数据库中增加GoodsReceivingPO对象 |
| GoodsReceivingFormDataService. findLastID(String orgcode) | 根据机构编号获取GoodsReceivingPO的最新编号 |
| GoodsReceivingFormDataService. getHallLoadBycode(String hallloadcode) | 根据营业厅装车单编号获取营业厅装车单 |
| GoodsReceivingFormDataService. getCenterLoadBycode(String centerloadcode) | 根据中转中心装车单编号获取中转中心装车单 |
| HallLoadFormDataService. getLoadForm (Formstate state,String orgcode) | 根据单据状态和机构编号获取营业厅装车单单的list |
| HallLoadFormDataService. updateLoadForm(HallLoadPO po) | 更新数据库中的HallLoadPO对象 |
| HallLoadFormDataService. addLoadForm (HallLoadPO po) | 向数据库中增加HallLoadPO对象 |
| HallLoadFormDataService. findLastmotorcode(String orgcode,long date) | 根据机构编号和日期获取HallLoadPO的最新编号 |
| HallLoadFormDataService. checkHallLoad (String motorcode) | 检查编号为motorcode的HallLoadPO是否存在 |
| HallLoadFormDataService. getunitprice(String transporttype) | 根据运输方式获取单位距离的运费 |
| HallLoadFormDataService. getdistance(String city1,String city2) | 根据城市名获取两城市之间的距离 |
| HallLoadFormDataService. getweight(String ordernum) | 根据订单号获取该订单中货物的重量 |
| RecordtransFormDataService. getRecordtransForm (Formstate state,String orgcode) | 根据单据状态和机构编号获取中转单的list |
| RecordtransFormDataService. updateRecordtransForm (RecordtransPO po) | 更新数据库中的RecordtransPO对象 |
| RecordtransFormDataService. addRecordtransForm (RecordtransPO po) | 向数据库中增加RecordtransPO对象 |
| RecordtransFormDataService. findLasttranscode(String orgcode,long date) | 根据机构编号和日期获取RecordtransPO的最新编号 |
| RecordtransFormDataService. checkTrans(String transcode) | 检查编号为transcode的RecordtransPO是否存在 |
| RecordtransFormDataService. getunitprice(String transporttype) | 根据运输方式获取单位距离的运费 |
| RecordtransFormDataService. getdistance(String city1,String city2) | 根据城市名获取两城市之间的距离 |
| RecordtransFormDataService. getweight(String ordernum) | 根据订单号获取该订单中货物的重量 |
| LogisticsInfoService. findLogisticsInfo (String ordernum) | 根据订单号获取订单的物流信息 |
| LogisticsInfoService. update(LogisticsPO po) | 更新数据库中的LogisticsPO对象 |

(4)业务逻辑层的动态模型

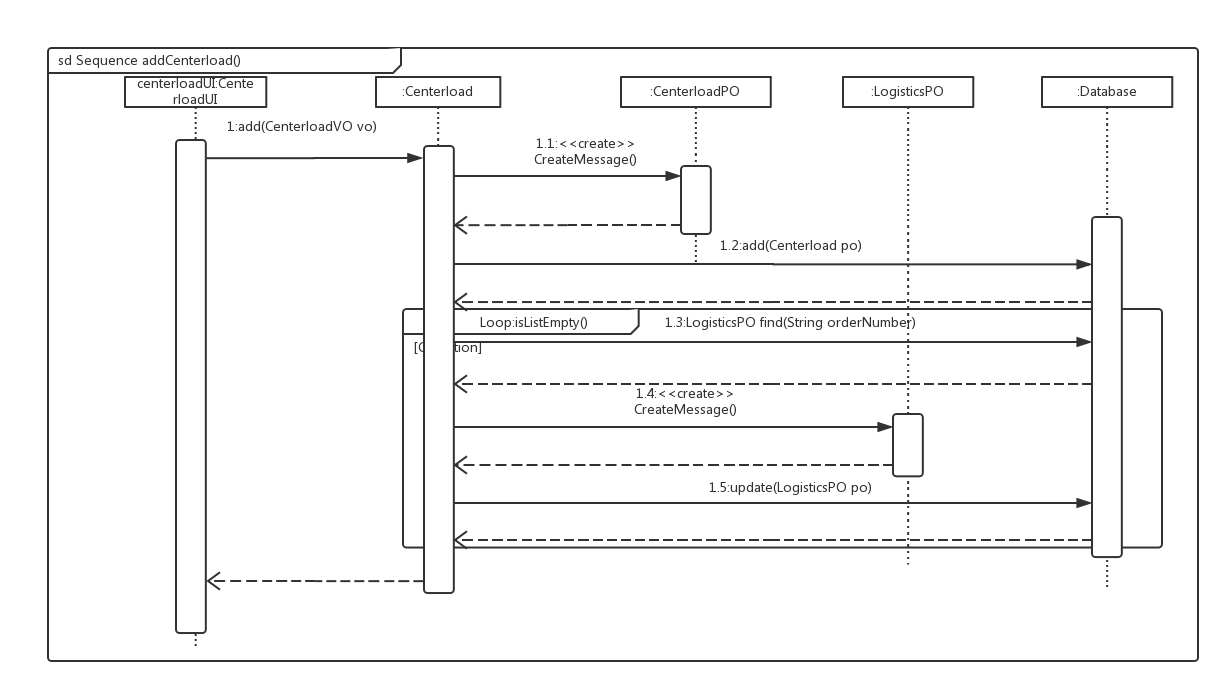
下图表示快递公司管理系统中，当输入营业厅装车单时，相关对象的协作。

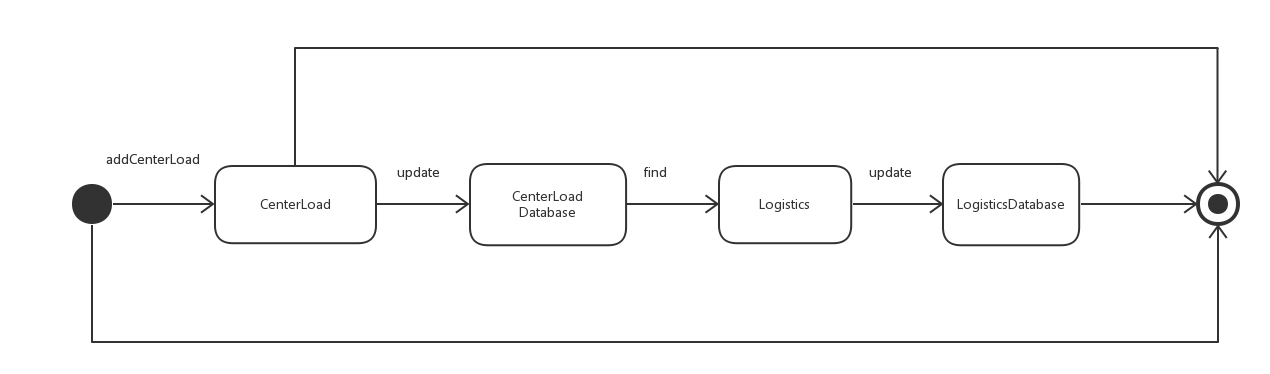


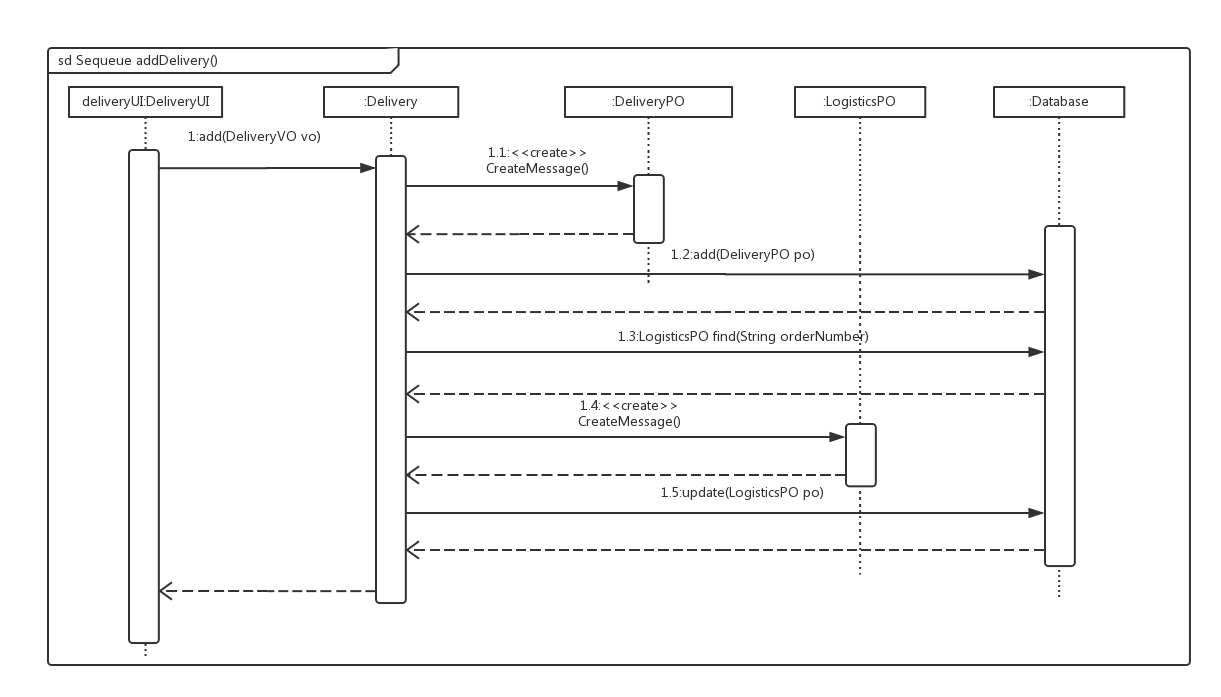
下图描述营业厅Hallload对象的生存期间的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



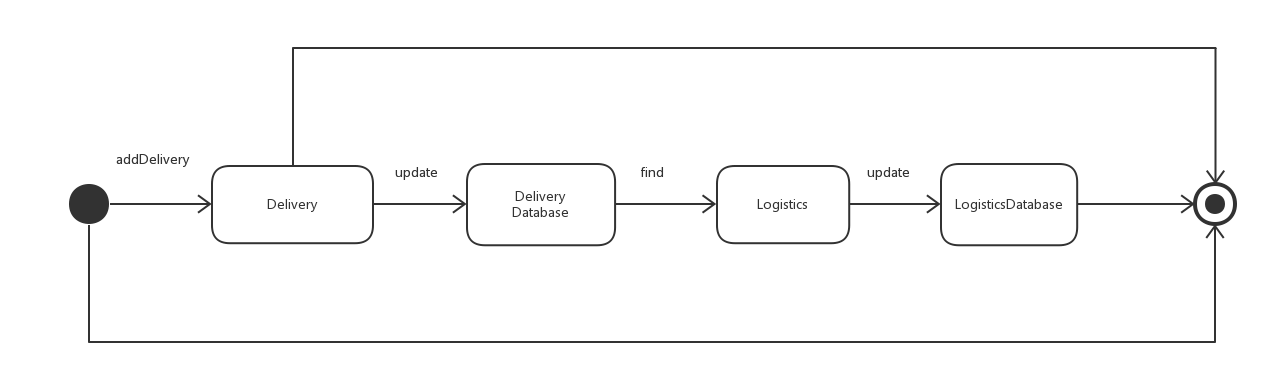
下图表示快递公司管理系统中，当输入中转中心装车单时，相关对象的协作。

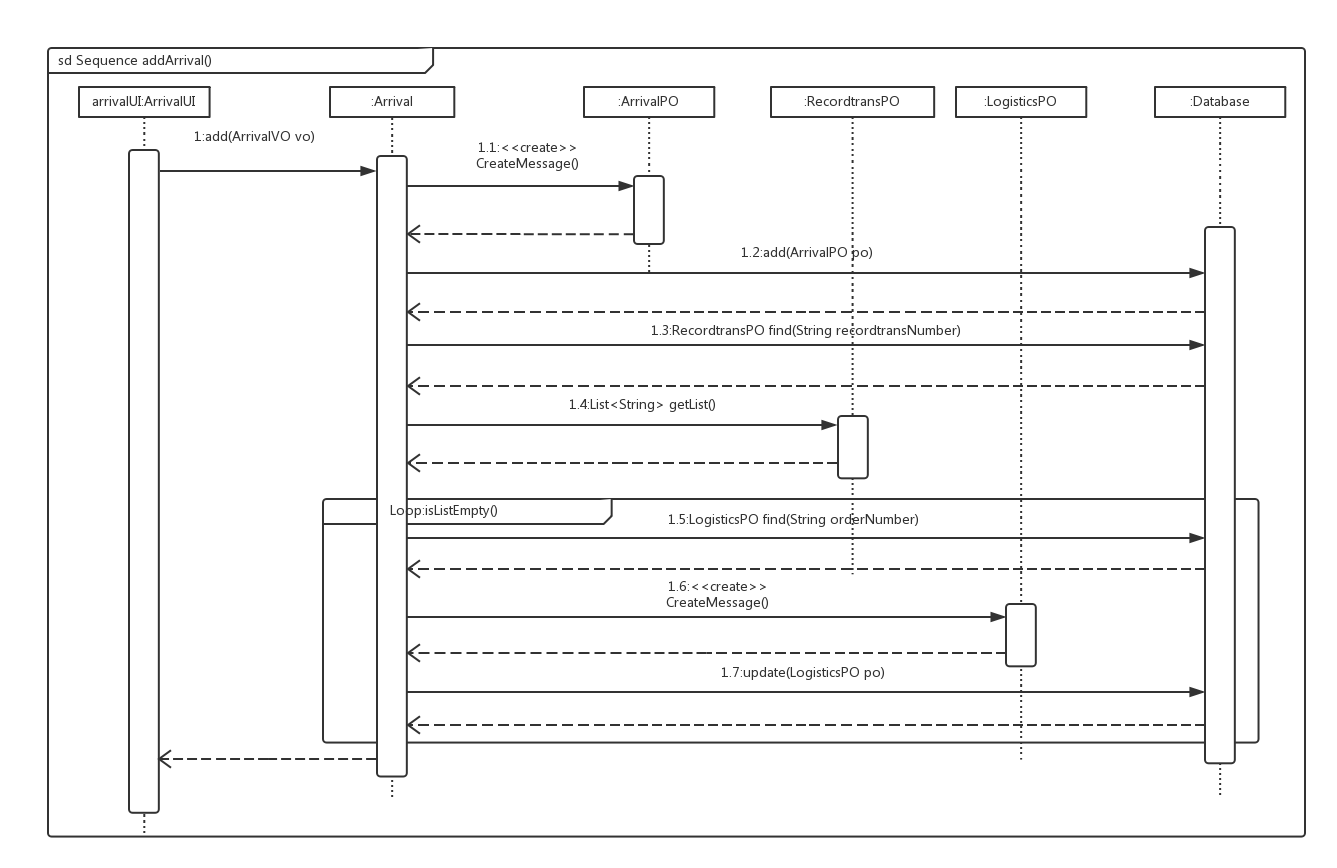


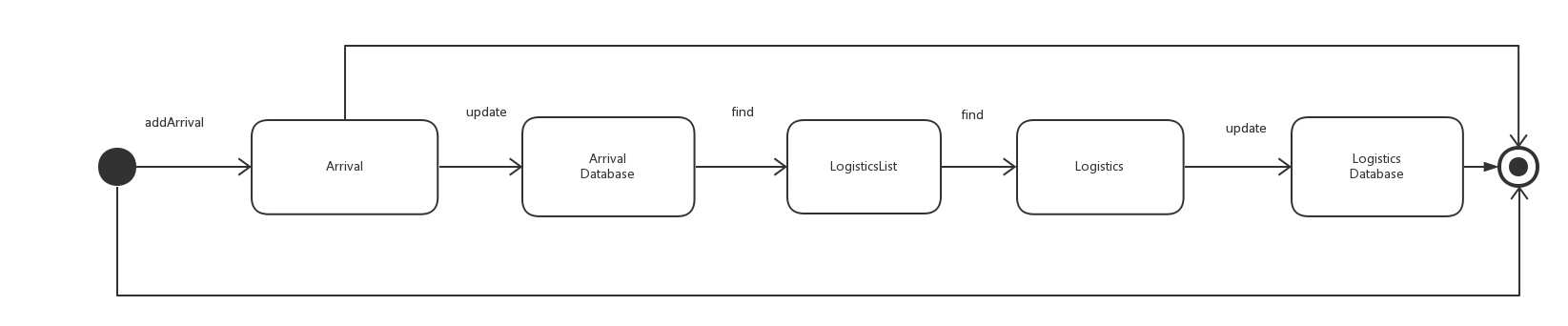
下图描述营业厅Centerload对象的生存期间的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。

下图表示快递公司管理系统中，当输入派件单时，相关对象的协作。 

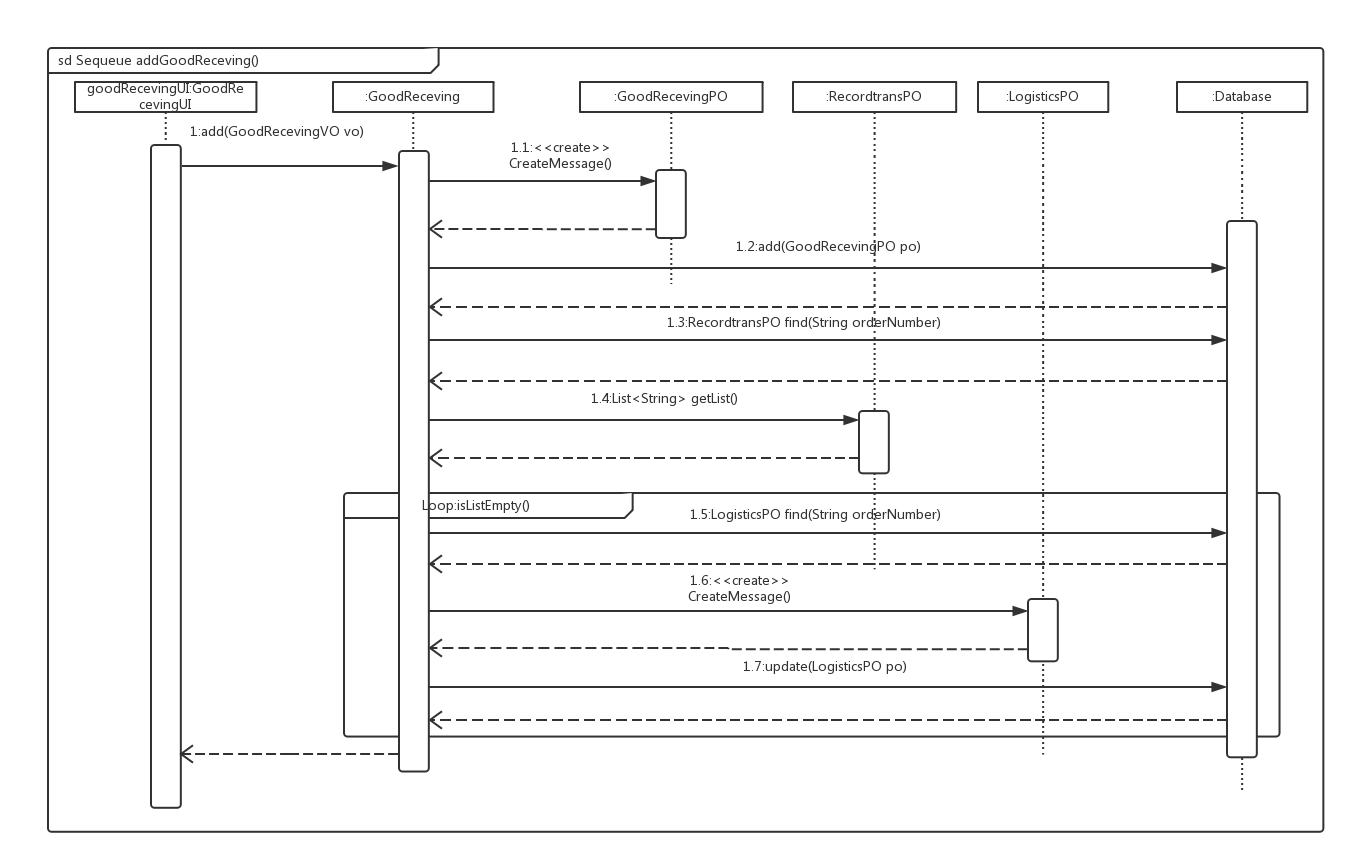
下图描述营业厅Delivery对象的生存期间的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



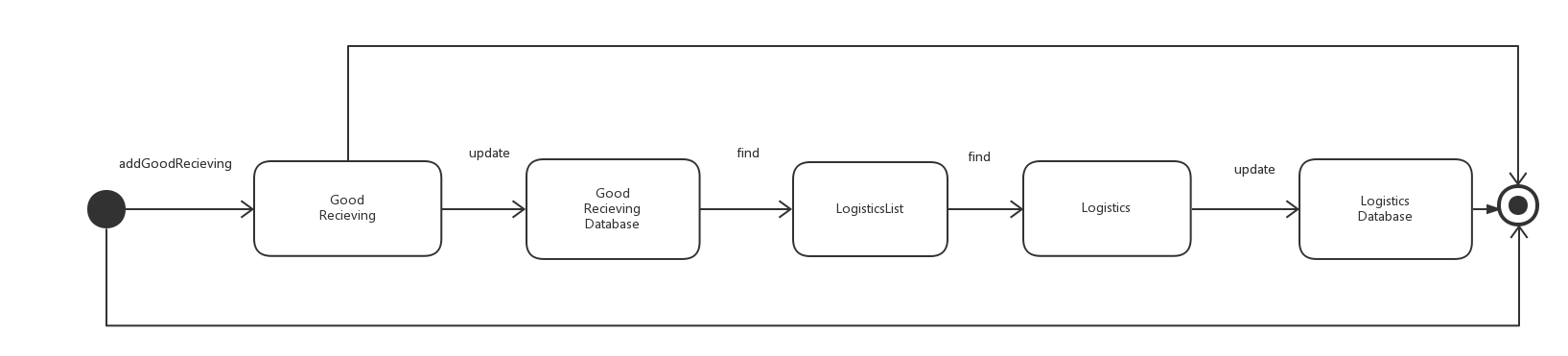
下图表示快递公司管理系统中，当输入到达单时，相关对象的协作。

下图描述营业厅Arrival对象的生存期间的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。

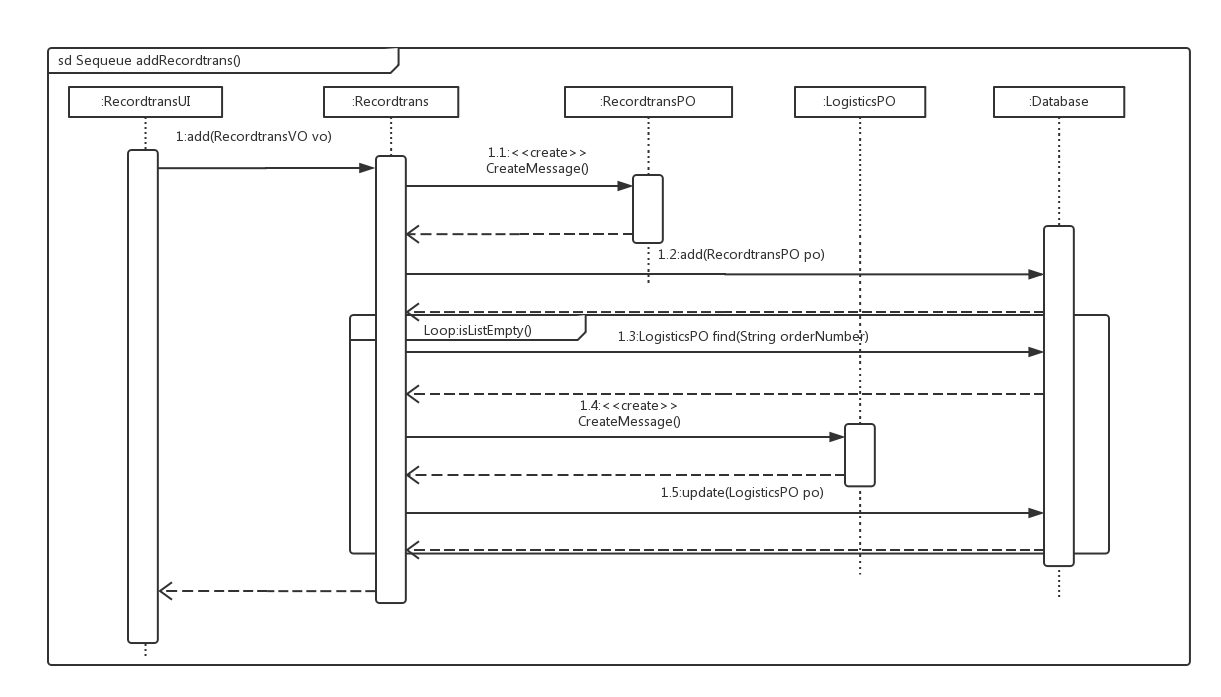
下图表示快递公司管理系统中，当输入收货单时，相关对象的协作。



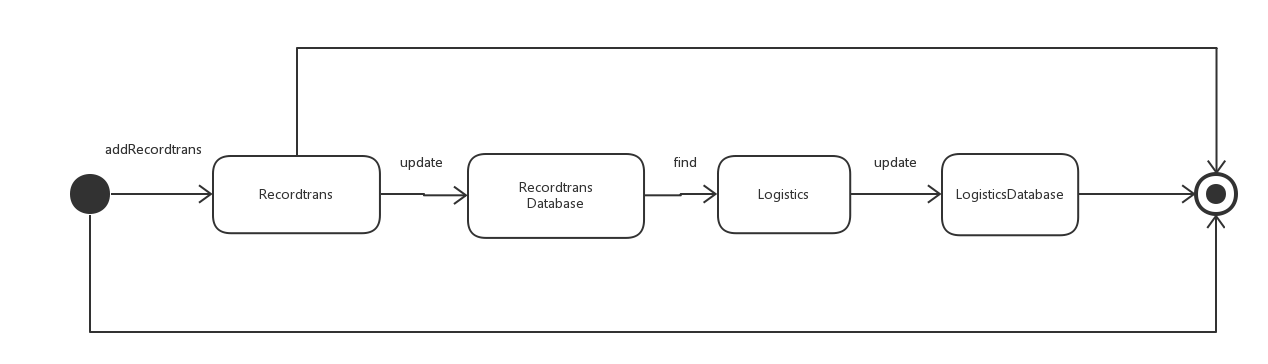
下图描述营业厅GoodReceving对象的生存期间的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



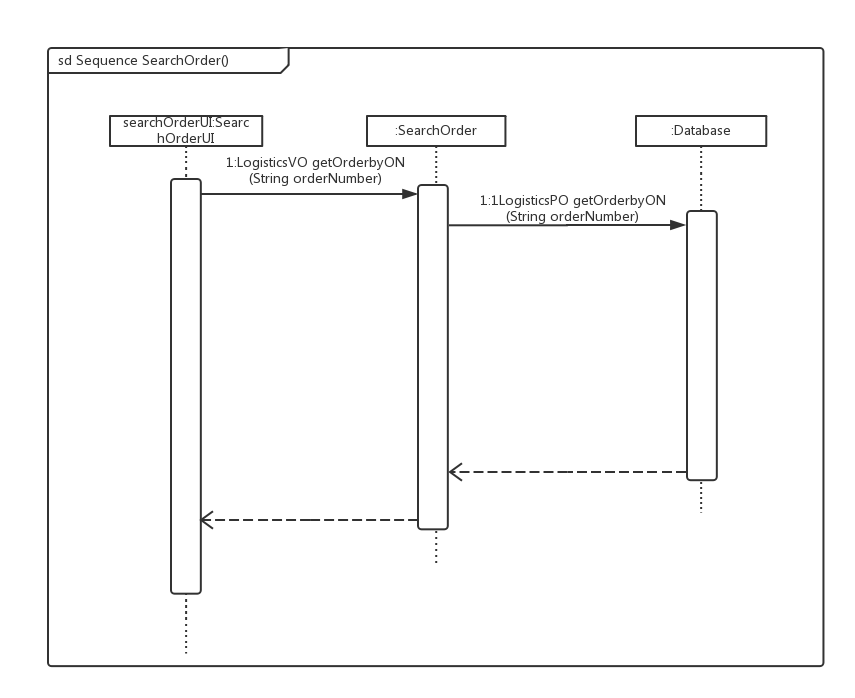
下图表示快递公司管理系统中，当输入中转单时，相关对象的协作。



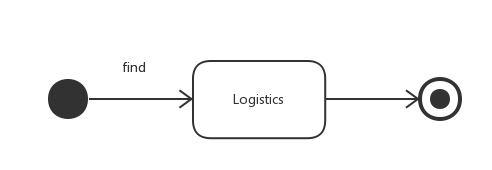
下图描述营业厅Recordtrans对象的生存期间的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



下图表示快递公司管理系统中，当进行物流信息查询时，相关对象的协作。



下图描述营业厅SearchOrder对象的生存期间的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



(5)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层由各自的控制器委托给不同的领域对象。

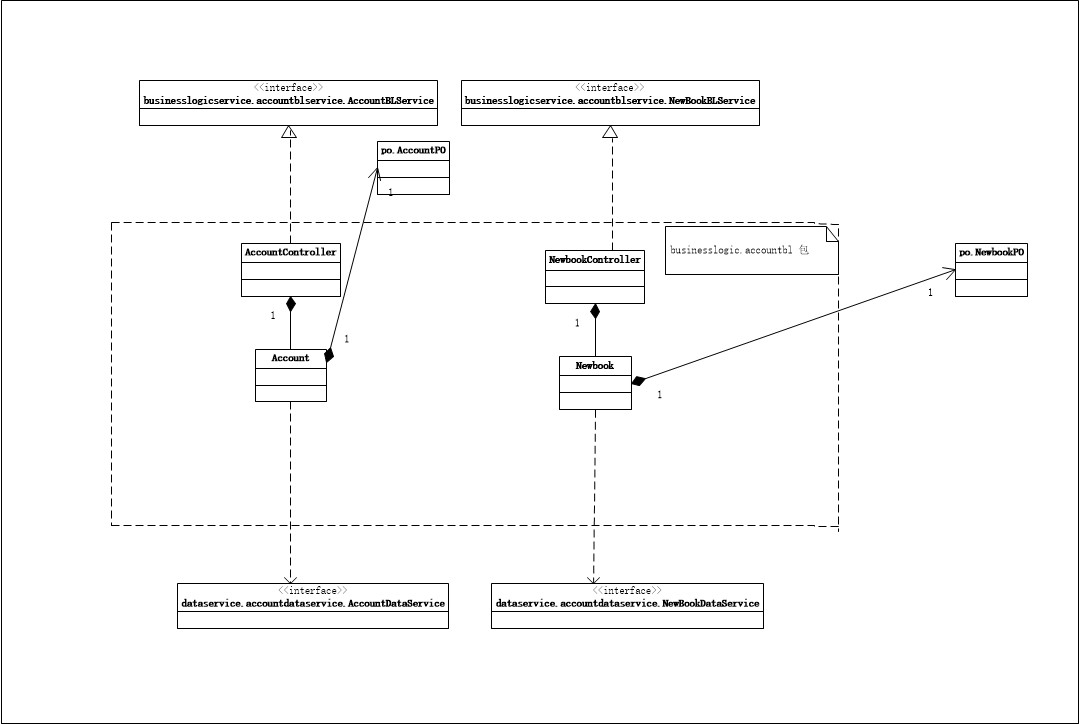
4.1.2 accountbl模块

(1)模块概述

accountbl模块承担的需求参见需求规格说明文档功能需求和相关非功能性需求。 accountbl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

accountbl模块的设计如下图所示。



accountbl模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| AccountController | 负责实现账户管理界面所需要的服务 |
| NewbookController | 负责实现期初建账界面所需要的服务 |
| Account | 账户的领域模型对象，拥有账户的名称和余额，可以帮助完成账户管理界面所需要的服务。 |
| Newbook | 期初建账的领域模型对象，拥有期初建账的人员机构，库存等信息，可以帮助完成期初建账界面所需要的服务。 |

(3)模块内部类的接口规范

AccountController，Account，NewbookController和Newbook接口规范如下

AccountController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| AccountController.addAccount | 语法 | | public ResultMessage addAccount(AccountVO vo) |
| 前置条件 | | 已创建一个Account领域对象，并且输入符合输入规则 |
| 后置条件 | | 调用Account领域对象的addAccount方法 |
| AccountController.delAccount | 语法 | | public ResultMessage delAccount() |
| 前置条件 | | 已创建一个Account领域对象 |
| 后置条件 | | 调用Account领域对象的delAccount方法 |
| AccountController.revAccount | 语法 | | public ResultMessage revAccount(AccountVO vo) |
| 前置条件 | | 已创建一个Account领域对象 |
| 后置条件 | | 调用Account领域对象的revAccount方法 |
| AccountController.getAccountbyName | 语法 | | public AccountVO getAccountbyName(String accountname) |
| 前置条件 | | 已创建一个Account领域对象 |
| 后置条件 | | 调用Account领域对象的getAccountbyName方法 |
| **需要的服务(需接口)** | | | |
| Account.addAccount(AccountVO vo) | | 加入一个账户 | |
| Account.delAccount() | | 删除一个账户 | |
| Account.revAccount(AccountVO vo) | | 更新一个账户 | |
| Account.getAccountbyID(String id, String password) | | 搜索一个账户 | |

Account的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Account.addAccount | 语法 | public ResultMessage addAccount(AccountVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入account信息 |
| Account.delAccount | 语法 | public ResultMessage delAccount() |
| 前置条件 | 数据库中存在相应的account数据 |
| 后置条件 | 在数据库中删除相应的account数据 |
| Account.revAccount | 语法 | public ResultMessage revAccount(AccountVO vo) |
| 前置条件 | 数据库中存在相应的account数据 |
| 后置条件 | 向数据库中更新入新的account数据 |
| Account.getAccountbyName | 语法 | public AccountVO getAccountbyName(String accountname) |
| 前置条件 | 无 |
| 后置条件 | 返回数据库中相应的account信息 |

|  |
| --- |
| 需要的服务(需接口) |

|  |  |
| --- | --- |
| DatabaseFactory.getAccountDa  Tabase | 得到Account数据库的服务和引用 |
| AccountDataService.addAccount(Account PO po) | 向数据库中插入AccoutPO对象 |
| AccountDataService.deleteAccount(Account PO po) | 从数据库中删除AccountPO对象 |
| AccountDataService.updateAccount(Account PO po) | 从数据库中修改更新某一AccountPO对象 |
| AccountDataService.findAccount(String accountname) | 从数据库中查询某一AccountPO对象 |
| AccountDataService. getLastId () | 获取AccountPO的最新ID |

NewbookController的接口规范

|  |  |  |
| --- | --- | --- |
| NewBookController.newBook | 语法 | public ResultMessage NewBook (long id); |
| 前置条件 | 无 |
| 后置条件 | 调用NewBook领域对象的NewBook方法 |
| NewBookController.findNewBook | 语法 | public NewbookVO find(long time); |
| 前置条件 | 已创建一个NewBook领域对象 |
| 后置条件 | 调用NewBook领域对象的find方法 |
| NewBookController.getid | 语法 | public long getid(); |
| 前置条件 | 已创建一个NewBook领域对象 |
| 后置条件 | 调用NewBook领域对象的getid方法 |

|  |  |
| --- | --- |
| **需要的服务(需接口)** | |
| NewBook.NewBook(NewbookVO vo) | 新建一个账 |

NewBook的接口规范

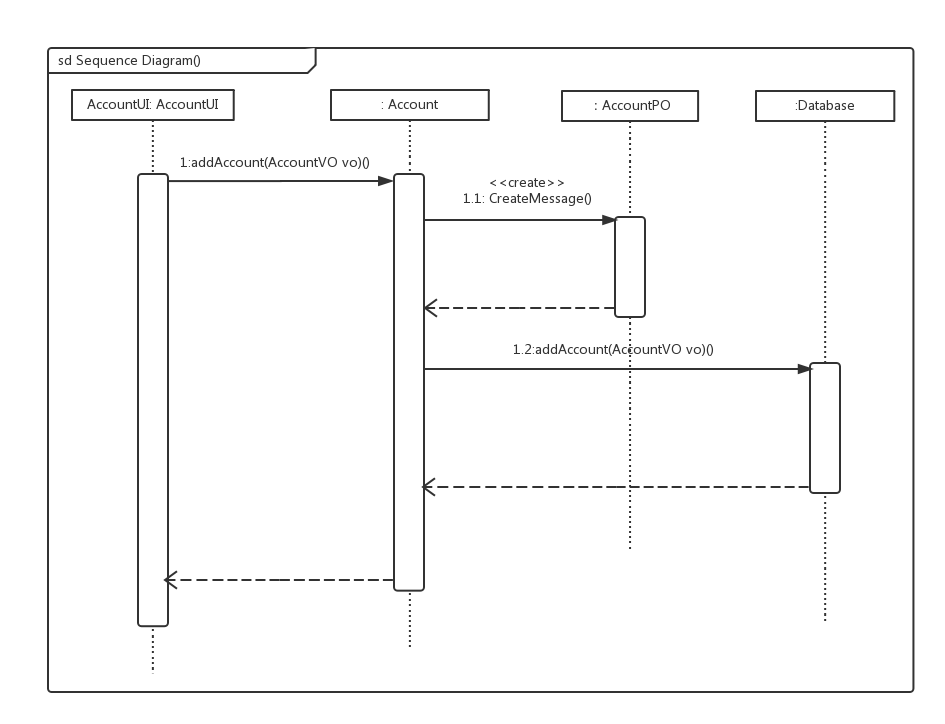
|  |  |  |
| --- | --- | --- |
| NewBook.newBook | 语法 | public ResultMessage NewBook (long id); |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入新的newbook数据 |
| NewBook.findNewBook | 语法 | public NewbookVO find(long time); |
| 前置条件 | 无 |
| 后置条件 | 返回数据库中相应的newbook数据 |
| NewBook.getid | 语法 | public long getid(); |
| 前置条件 | 数据库中有相应的newbook的id |
| 后置条件 | 返回newbook的当前id |

需要的服务(需接口)

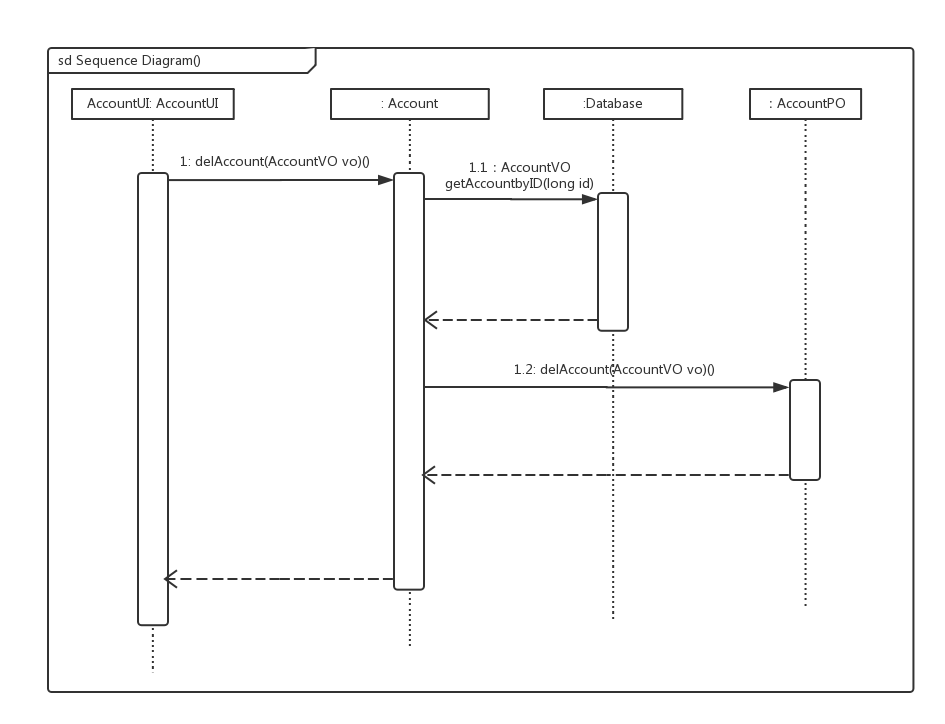
|  |  |  |
| --- | --- | --- |
| DatabaseFactory.getNewBookDa  Ta | 得到NewBook数据库的服务和引用 | |
| NewBookDataService.addNewBook(NewBookPO po) | 向数据库中插入NewBookPO对象 | |
| NewBookDataService.findNewBook(long time) | 从数据库中查询某一NewBookPO对象 | |
| NewBookDataService .getLastBook() | 获取当前的AccountPO、CommodityPO、OrganizationPO、StaffPO、VehiclesPO列表 | |
| NewBookDataService. getLastId () | | 获取NewBookPO的最新ID |

(4) 业务逻辑层的动态模型

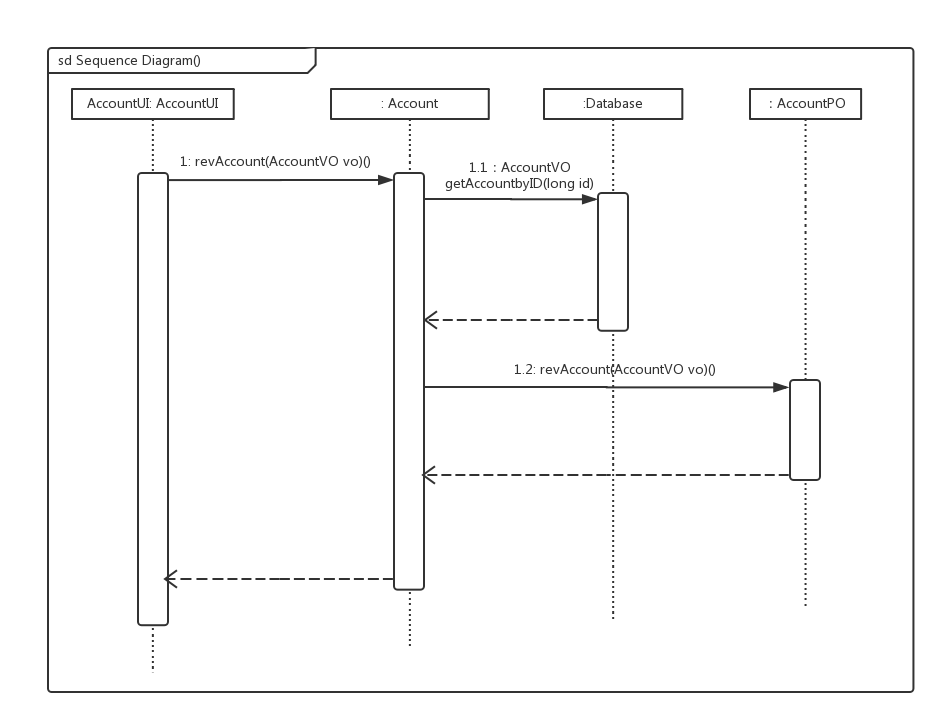
下图表明了当财务人员进行账户增加时，账户管理业务逻辑处理的相关对象之间的协作。



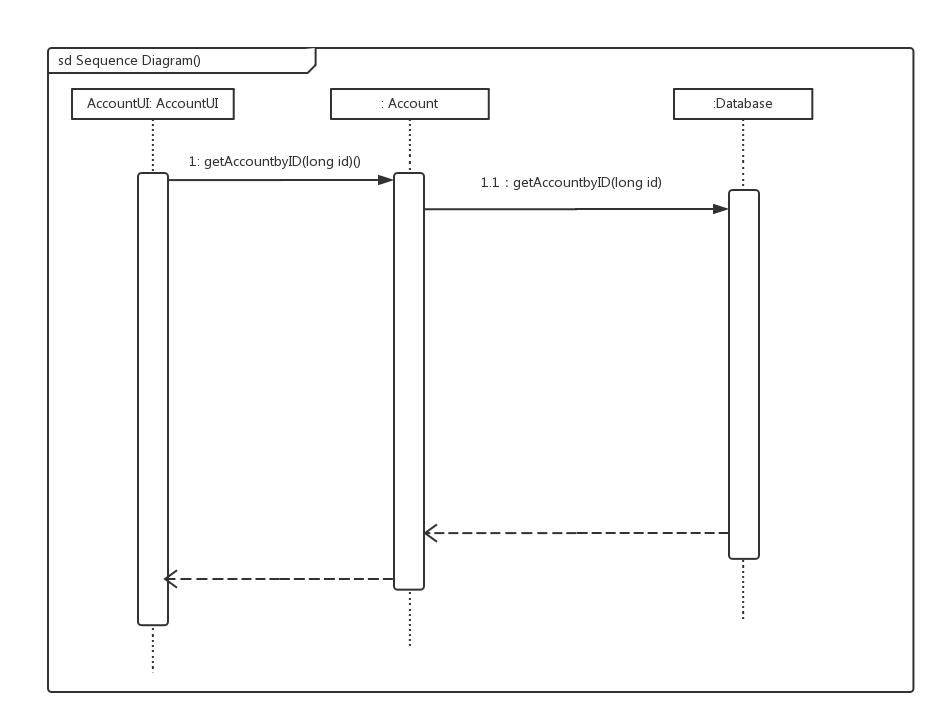
下图表明了当财务人员进行账户删除时，账户管理业务逻辑处理的相关对象之间的协作。



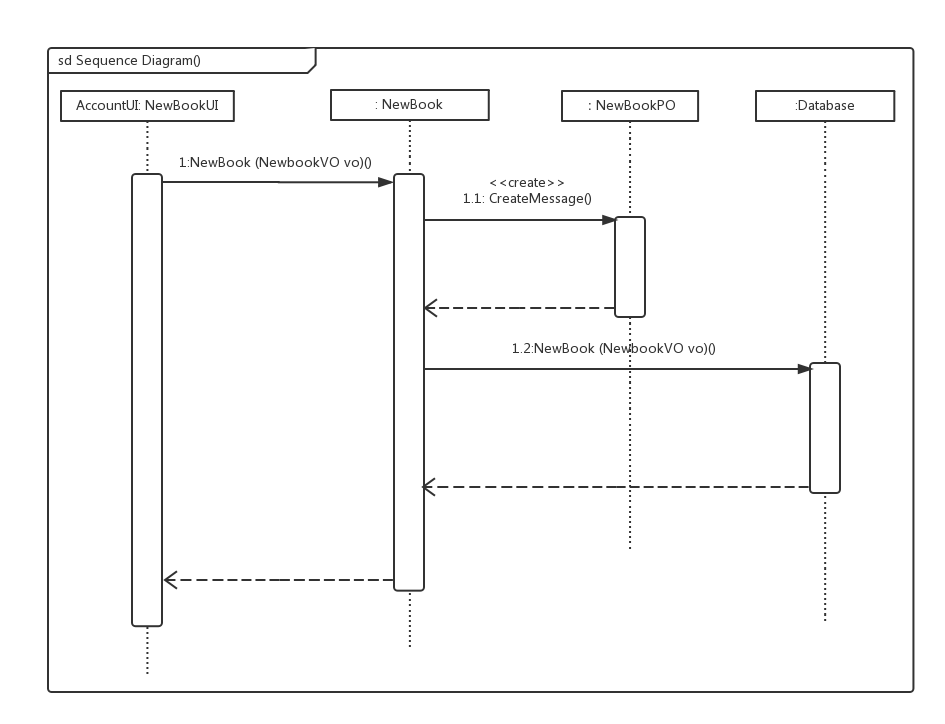
下图表明了当财务人员进行账户修改时，账户管理业务逻辑处理的相关对象之间的协作。



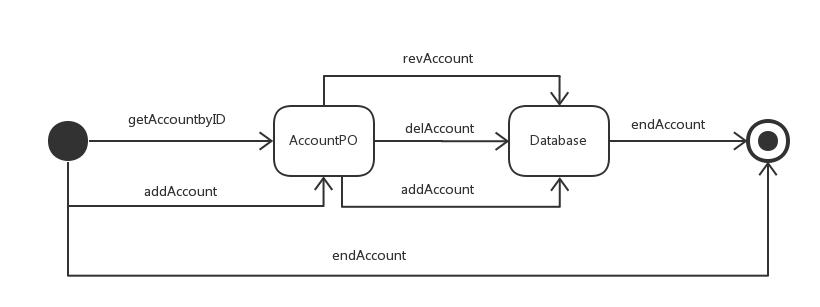
下图表明了当财务人员进行账户查找时，账户管理业务逻辑处理的相关对象之间的协作。



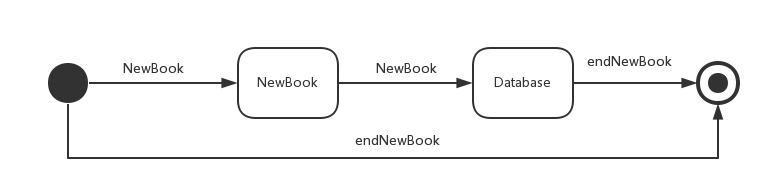
下图表明了当财务人员进行期初建账时，账户管理业务逻辑处理的相关对象之间的协作。



下图为财务人员进行账户的增删改查时的状态图



下图为财务人员进行期初建账时的状态图



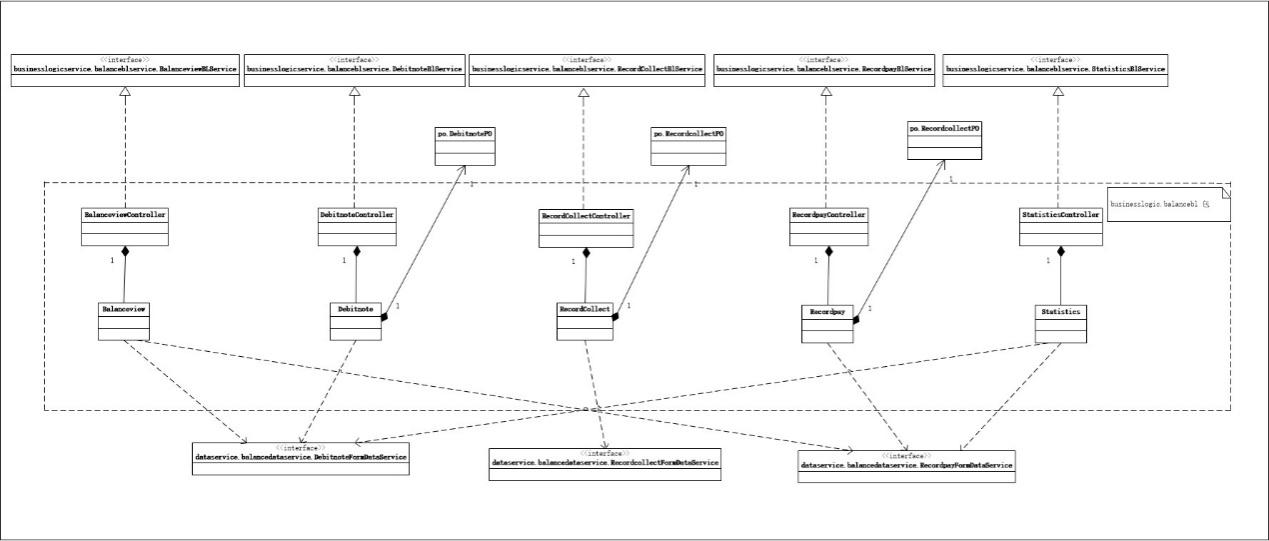
4.1.2 balancebl模块

(1)模块概述

balancebl模块承担的需求参见需求规格说明文档功能需求和相关非功能性需求。 balancebl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

balancebl模块的设计如下图所示。



balancebl模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| BalanceviewController | 负责实现成本收益表界面所需要的服务 |
| DebitnoteController | 负责实现结算管理界面所需要的服务 |
| RecordcollectController | 负责实现记录收款信息界面所需要的服务 |
| RecordpayController | 负责实现记录付款信息界面所需要的服务 |
| StatisticsController | 负责实现经营情况表界面所需要的服务 |
| Balanceview | 成本收益表的领域模型对象，拥有成本收益表的总支出，总收益和总利润，可以帮助完成成本收益表界面所需要的服务。 |
| Debitnote | 结算管理的领域模型对象，拥有结算管理的付款金额，付款日期等信息，可以帮助完成结算管理界面所需要的服务。 |
| Recordcollect | 收款信息的领域模型对象，拥有收款信息的日期，金额，收款方等信息，可以帮助完成记录收款信息界面所需要的服务。 |
| Recordpay | 付款信息的领域模型对象，拥有付款信息的日期，金额，条目等信息，可以帮助完成记录付款信息界面所需要的服务。 |
| Statistics | 经营情况表的领域模型对象，拥有经营情况表的收款信息和付款信息等信息，可以帮助完成经营情况表界面所需要的服务。 |

(3)模块内部类的接口规范

BalanceviewController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| BalanceviewController.Balanceview | 语法 | public BalanceviewVO getBalanceview() |
| 前置条件 | 已创建一个Balanceview领域对象 |
| 后置条件 | 调用Balance领域对象的Balanceview方法 |

需要的服务(需接口)

|  |  |
| --- | --- |
| Balanceview.Balanceview() | 显示成本收益表 |

DebitnoteController的接口规范

|  |  |  |
| --- | --- | --- |
| DebitnoteController.getAllCity | 语法 | **public** List<CityVO> getALLCity(); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的getAllCity（）方法 |
| DebitnoteController.getAllOrganization | 语法 | **public** List<OrganizationVO> getAllOrganization (String city); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的getAllOrganization方法 |
| DebitnoteController.getRecordcollectbyhall | 语法 | **public** DebitnoteVO getRecordcollectbyhall(String orgcode); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的getRecordcollectbyhall方法 |
| DebitnoteController.getRecordcollectbydate | 语法 | **public** DebitnoteVO getRecordcollectbydate(**long** date); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的getRecordcollectbydate方法 |
| DebitnoteController.examcollect | 语法 | **public** DebitnoteExamVO examcollect(DebitnoteVO vo); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的examcollect方法 |

需要的服务(需接口)

|  |  |
| --- | --- |
| Debitnote.Debitnote(DebitnoteVO vo) | 记录一条结算信息 |

RecordcollectController的接口规范

|  |  |  |
| --- | --- | --- |
| RecordcollectController.RecordCollect | 语法 | public ResultMessage Recordcollect (RecordcollectVO vo) |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的Recordcollect方法 |
| RecordcollectController.updateCollect | 语法 | public ResultMessage updatecollect (RecordcollectVO vo) |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的updatecollect方法 |
| RecordcollectController.findform | 语法 | **public** List<RecordcollectVO> findform(Formstate state,String orgcode); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的findform方法 |
| RecordcollectController.getid | 语法 | **public** String getid(String orgcode); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域的getid方法 |

需要的服务(需接口)

|  |  |
| --- | --- |
| RecordcollectController.RecordCollect(RecordcollectVO vo) | 记录一条收款信息 |

RecordpayController的接口规范

|  |  |  |
| --- | --- | --- |
| RecordpayController.AddRecordpay | 语法 | public ResultMessage AddRecordpay(RecordpayVO vo) |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域模型的AddRecordpay方法 |
| RecordpayController.updateRecordpay | 语法 | **public** ResultMessage updateRecordpay(RecordpayVO vo); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域模型的updateRecordpay方法 |
| RecordpayController.findforms | 语法 | **public** List<RecordpayVO> findforms(Formstate formstate); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域模型的findforms方法 |
| RecordpayController.getid | 语法 | **public** String getid(); |
| 前置条件 | 输入信息符合规范 |
| 后置条件 | 调用Balance领域模型的getid方法 |

需要的服务(需接口)

|  |  |
| --- | --- |
| Recordpay.Recordpay(RecordpayVO vo) | 记录一条付款信息 |

StatisticsController的接口规范

|  |  |  |
| --- | --- | --- |
| StatisticsController.Statistics | 语法 | public StatisticsVO Statistics(long startTime,long endTime) |
| 前置条件 | 数据库中存在经营情况信息 |
| 后置条件 | 调用Balance领域模型的Statistics方法 |

需要的服务(需接口)

|  |  |
| --- | --- |
| Statistics.Statistics(long startTime,long endTime) | 显示经营情况表 |

Balanceview的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Balanceview.Balanceview | 语法 | public BalanceviewVO getBalanceview() |
| 前置条件 | 数据库中存在成本收益信息 |
| 后置条件 |  |
| 需要的服务(需接口) | | |

|  |  |
| --- | --- |
| DatabaseFactory.getBalanceDa  Ta | 得到Balance数据库的服务和引用 |
| BalanceDataService.allin() | 获取至今为止的总收入 |
| BalanceDataService.allout() | 获取至今为止的总支出 |

Debitnote的接口规范

|  |  |  |
| --- | --- | --- |
| Debitnote.getAllCity | 语法 | **public** List<CityVO> getALLCity(); |
| 前置条件 | 数据库中存在城市信心 |
| 后置条件 | 无 |
| Debitnote.getAllOrganization | 语法 | **public** List<OrganizationVO> getAllOrganization (String city); |
| 前置条件 | 数据库中存在组织信息 |
| 后置条件 |  |
| Debitnote.getRecordcollectbyhall | 语法 | **public** DebitnoteVO getRecordcollectbyhall(String orgcode); |
| 前置条件 | 数据库中存在recordcollect信息 |
| 后置条件 | 无 |
| Debitnote.getRecordcollectbydate | 语法 | **public** DebitnoteVO getRecordcollectbydate(**long** date); |
| 前置条件 | 数据库中存在recordcollect信息 |
| 后置条件 | 无 |
| Debitnote.examcollect | 语法 | **public** DebitnoteExamVO examcollect(DebitnoteVO vo); |
| 前置条件 | 无 |
| 后置条件 | 无 |

需要的服务(需接口)

|  |  |
| --- | --- |
| DatabaseFactory.getDebitnoteDa  Ta | 得到Debitnote数据库的服务和引用 |
| DebitnoteDataService. getALLCity() | 从数据库中得到所有城市的列表 |
| DebitnoteDataService. getRecordcollectbyhall(String orgcode) | 从数据库中按营业厅编号得到DebitPO对象列表 |
| DebitnoteDataService.getRecordcollectbydate(long date) | 从数据库中按日期得到DebitPO对象列表 |
| DebitnoteDataService. getAllOrganization(String city) | 从数据库中按城市名得到所有机构的列表 |

Recordcollect的接口规范

|  |  |  |
| --- | --- | --- |
| Recordcollect.RecordCollect | 语法 | public ResultMessage Recordcollect (RecordcollectVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入recordcollect信息 |
| Recordcollect.updateCollect | 语法 | public ResultMessage updatecollect (RecordcollectVO vo) |
| 前置条件 | 数据库中存在相应的recordcollect信息 |
| 后置条件 | 向数据库中更新入新的recordcollect信息 |
| Recordcollect.findform | 语法 | **public** List<RecordcollectVO> findform(Formstate state,String orgcode); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Recordcollect.getid | 语法 | **public** String getid(String orgcode); |
| 前置条件 | 数据库中存在相应的recordcollect的id |
| 后置条件 | 无 |

需要的服务(需接口)

|  |  |
| --- | --- |
| DatabaseFactory.getRecordcollectData | 得到Recordcollect数据库的服务和引用 |
| RecordcollectDataService. getRecordcollectForm (Formstate state,String orgcode) | 从数据库中根据单据状态和机构编号获取RecordcollectPO列表 |
| RecordcollectDataService.addRecordcollectForm(RecordcollectPO po) | 向数据库中插入RecordcollectPO对象(收款单) |
| RecordcollectDataService. getlastid (String orgcode) | 获取RecordcollectPO的最新ID |
| RecordcollectDataService.updateRecordcollectForm(RecordcollectPO po) | 向数据库中更新RecordcollectPO对象 |
| RecordcollectDataService. addmoney(String accountname,double fee) | 在账号名为aacountname的账户下增加金额fee |

Recordpay的接口规范

|  |  |  |
| --- | --- | --- |
| Recordpay.AddRecordpay | 语法 | public ResultMessage AddRecordpay(RecordpayVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中更新入新的recordpay数据 |
| Recordpay.updateRecordpay | 语法 | **public** ResultMessage updateRecordpay(RecordpayVO vo); |
| 前置条件 | 数据库中存在相应的recordpay数据 |
| 后置条件 | 向数据库中更新入相应的recordpay数据 |
| Recordpay.findforms | 语法 | **public** List<RecordpayVO> findforms(Formstate formstate); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Recordpay.getid | 语法 | **public** String getid(); |
| 前置条件 | 数据库中存在recordpay的id |
| 后置条件 | 无 |

需要的服务(需接口)

|  |  |
| --- | --- |
| DatabaseFactory.getRecordpayData | 得到Recordpay数据库的服务和引用 |
| RecordpayDataService. getlastid() | 获取RecordpayPO的最新ID |
| RecordpayDataService. getRecordpayForm (Formstate state) | 从数据库中根据单据状态获取RecordpayPO列表 |
| RecordpayDataService.addRecordpayForm(RecordpayPO po) | 向数据库中插入RecordpayPO对象(付款单) |
| RecordpayDataService. removeTheCost(String accountname,double fee) | 在账号名为aacountname的账户下减少金额fee |
| RecordpayDataService.updateRecordpayForm(RecordpayPO po) | 向数据库中更新RecordpayPO对象 |

Statistics的接口规范

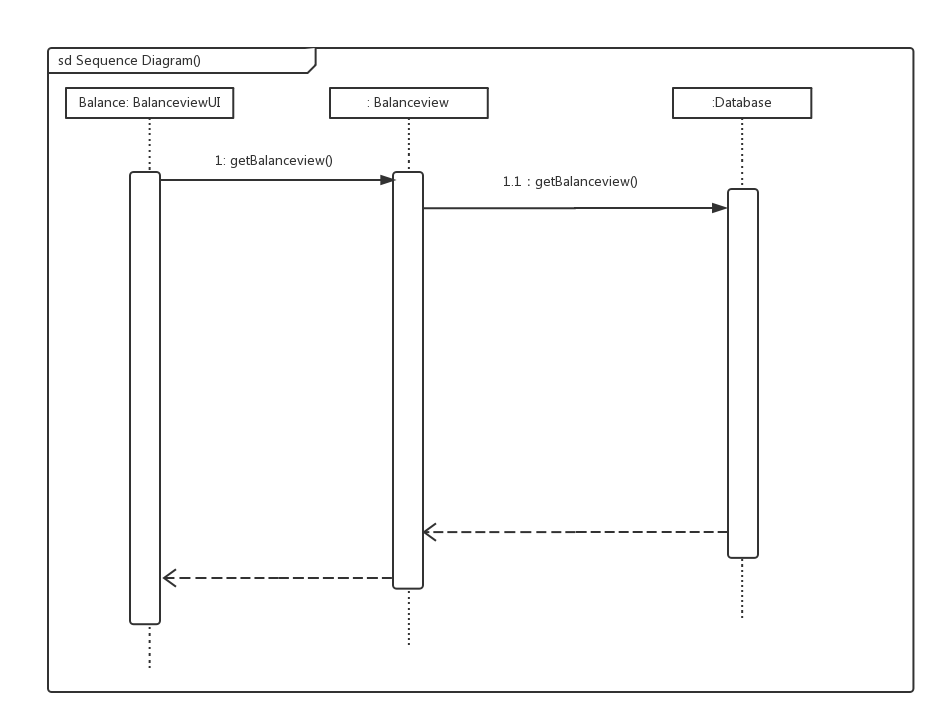
|  |  |  |
| --- | --- | --- |
| Statistics.Statistics | 语法 | public StatisticsVO Statistics(long startTime,long endTime) |
| 前置条件 | 数据库中存在经营情况信息 |
| 后置条件 | 无 |

需要的服务(需接口)

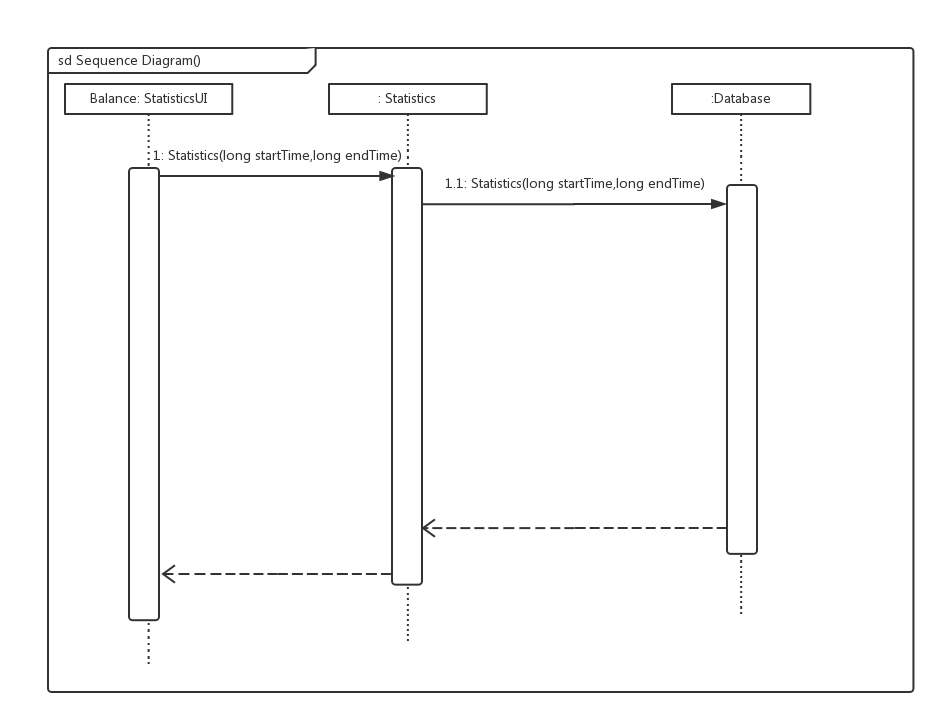
|  |  |
| --- | --- |
| DatabaseFactory.getStatisticsData | 得到Statistics数据库的服务和引用 |
| StatisticsDataService.getallRecordcollectForm(long starttime,long endtime) | 从数据库中根据开始时间和结束时间得到所有RecordcollectPO对象 |
| StatisticsDataService.getallRecordpayForm(long starttime,long endtime) | 从数据库中根据开始时间和结束时间得到所有RecordpayPO对象 |

(4) 业务逻辑层的动态模型

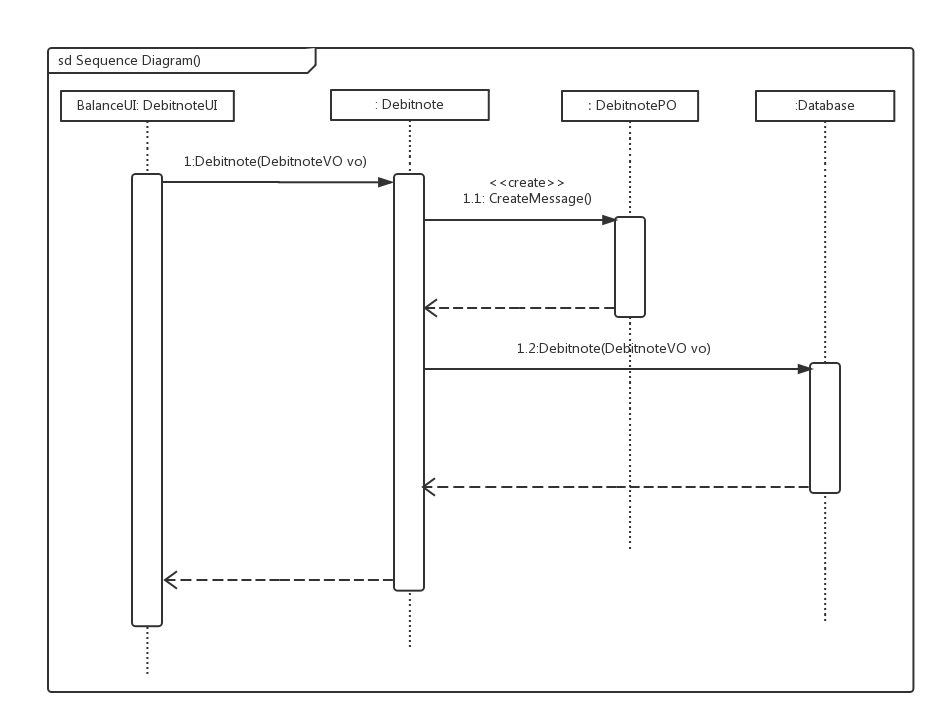
下图表明了当财务人员生成成本收益表时，成本收益业务逻辑处理的相关对象之间的协作。



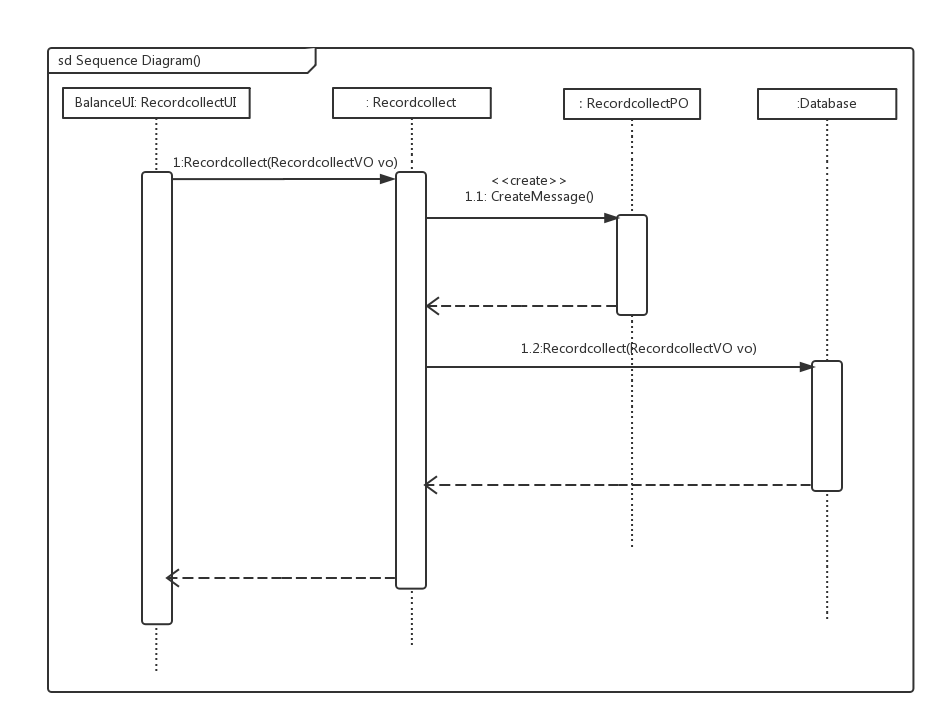
下图表明了当财务人员生成经营情况表时，成本收益业务逻辑处理的相关对象之间的协作。



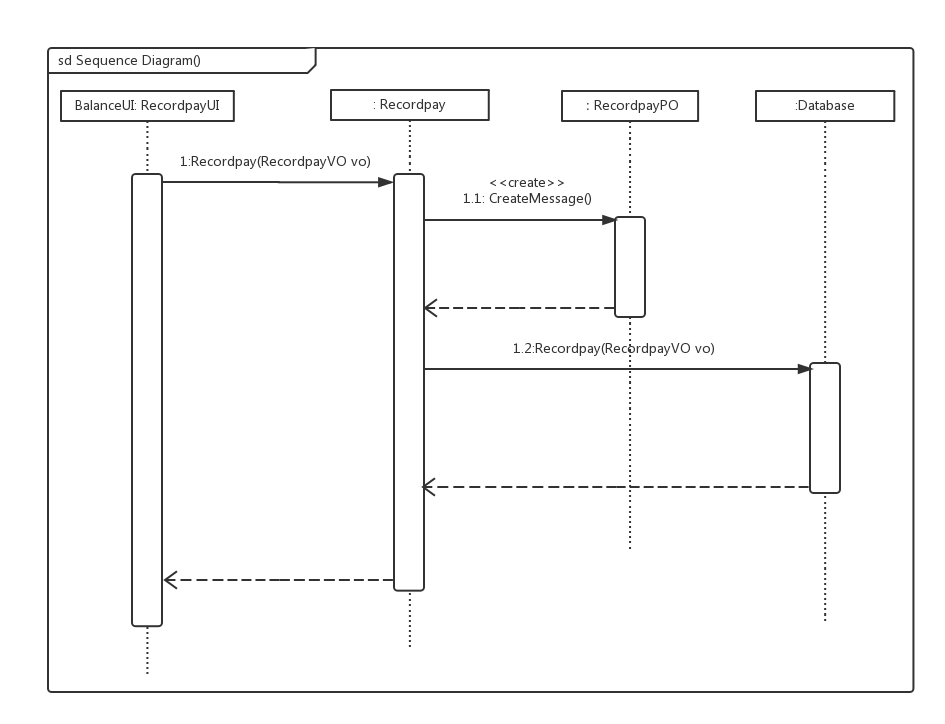
下图表明了当财务人员进行结算管理时，成本收益业务逻辑处理的相关对象之间的协作。



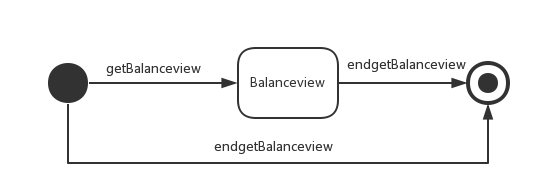
下图表明了当财务人员记录收款信息时，成本收益业务逻辑处理的相关对象之间的协作。



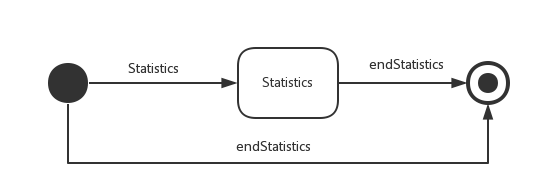
下图表明了当财务人员记录付款信息时，成本收益业务逻辑处理的相关对象之间的协作。



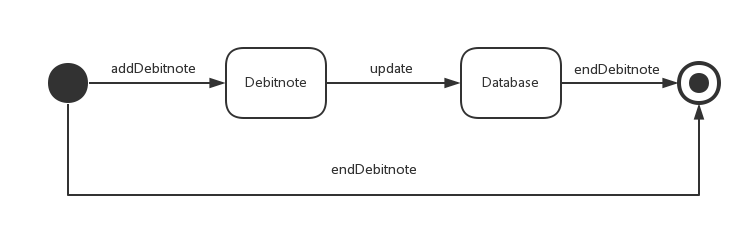
下图为财务人员生成成本收益表时的状态图



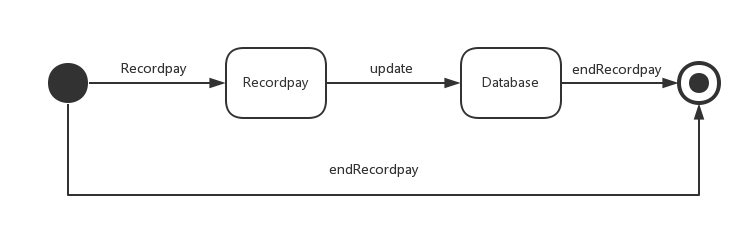
下图为财务人员生成经营情况表时的状态图



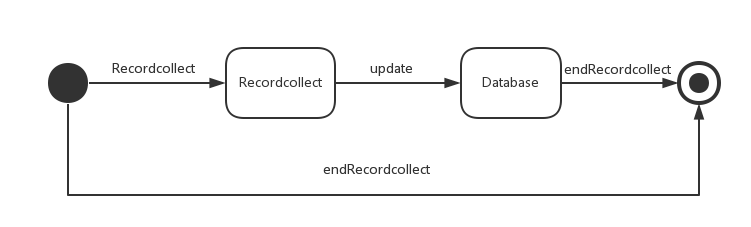
下图为财务人员进行结算管理时的状态图



下图为财务人员记录付款信息时的状态图



下图为财务人员记录收款信息时的状态图



(5)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层由各自的控制器委托给不同的领域对象。

4.1.4 commoditybl模块

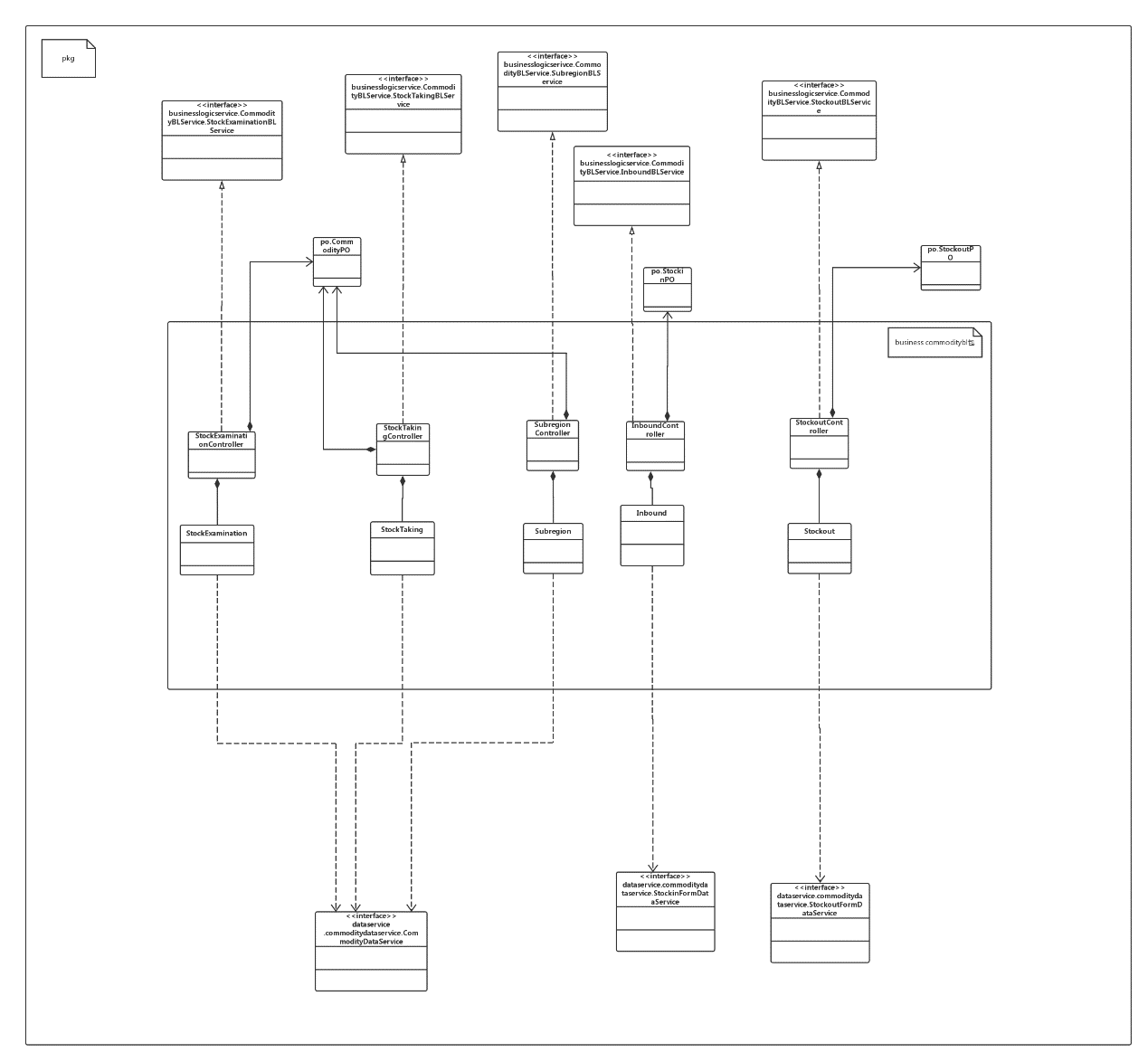
(1)模块概述

commoditybl模块承担的需求参见需求规格说明文档功能需求和相关非功能性需求。 commoditybl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层与业务逻辑层之间，我们添加businesslogicservice.commodityblservice.StockExaminatonBlService接口。业务逻辑层和数据层之间添加dataservice.commoditydataservice.ComodityDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了这样StockExaminationController,StockTakingController,SubreigonController,InboundController,StockoutController会将对StockExamination,StockTaking,Subreigon,Inbound,Stockout的业务逻辑处理委托给StockExamination,StockTaking,Subreigon,Inbound,Stockout对象。CommodityPO,StockinPO,StockoutPO是作为仓库记录的持久化对象被添加到设计模型中去的。

commoditybl模块的设计如下图所示。



Commodity模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| 模块 | 职责 |
| GetRegionInfoController | 负责得到分区信息相关的服务 |
| InboundController | 负责入库界面的服务 |
| StockExaminationController | 负责库存查看界面的服务 |
| StockoutController | 负责出库界面的服务 |
| StocktakingController | 负责库存盘点界面的服务 |
| SubregionController | 负责库存分区调整界面的服务 |
| Commodity | 库存的领域模型对象，拥有一次库存事件处理所持有的库存信息，车辆信息，航班号等信息，可以帮助完成库存界面所需要的服务。 |

(3)模块内部类的接口规范

CommodityController和Commodity的接口规范如表C1和C2所示。

表C1 CommodityController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| CommodityController.getRegionInfo | | 语法 | **public** RegionInfoVO getRegionInfo (String orgcode,**long** blocknum); |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的getRegionInfo方法 |
| CommodityController.wheOverLoad | | 语法 | **public** **boolean** wheOverLoad(StockinVO vo ) ; |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合规范 |
| 后置条件 | 调用Commodity领域对象的wheOverLoad方法 |
| CommodityController.Inbound | | 语法 | public ResultMessage Inbound(InboundVO vo) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的Inbound方法 |
| CommodityController.updateInbound | | 语法 | **public** ResultMessage update(StockinVO vo ); |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的update方法 |
| CommodityController.findInboundForm | | 语法 | **public** List<StockinVO> findInboundForm(Formstate state,String orgcode); |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的findInboundForm方法 |
| CommodityController.Inboundgetid | | 语法 | **public** String getid(String orgcode); |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的getid方法 |
| CommodityController.getEachBlockLimit | | 语法 | **public** List<Long> getEachBlockLimit(**long** blocknum,String orgcode); |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的getEachBlockLimit方法 |
| CommodityController.wheConflict | | 语法 | **public** **boolean** wheConflict(String orgcode,CommodityLocation location); |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的wheConflict方法 |
| CommodityController.Stockout | | 语法 | public ResultMessage Stockout(StockoutVO vo) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的Stockout方法 |
| CommodityController.StockoutUpdate | | 语法 | **public** ResultMessage update(StockoutVO vo) ; |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的update方法 |
| CommodityController.StockoutFindFormss | | 语法 | **public** List<StockoutVO> findforms(Formstate state,String orgcode) ; |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的findforms方法 |
| CommodityController.StockoutGetid | | 语法 | **public** String getid(String orgcode) ; |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的getid方法 |
| CommodityController.Stockexamination | | 语法 | **public** StockExaminationVO StockExamination (String orgcode,**long** startTime, **long** endTime); |
| 前置条件 | 已经创建Commodity领域模型对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的StockExamination方法 |
| CommodityController.Stocktaking | | 语法 | **public** StocktakingVO Stocktaking(String orgcode,**long** date,**long** blocknum); |
| 前置条件 | 已经创建Commodity领域模型对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的Stocktaking方法 |
| CommodityController.StocktakingGetid | | 语法 | **public** **long** getId(String orgcode,**long** date) ; |
| 前置条件 | 已经创建Commodity领域模型对象，输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的getid方法 |
| CommodityController.Subregion | | 语法 | public ResultMessage Subregion(Location beforeLocation, Location afterLocation) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的Subregion t方法 |
| 需要的服务(需接口) | | | |
| Commodity.StockExamination(long startTime, long endTime) | 进行一次库存查看 | | |
| Commodity.Stocktaking() | 进行一次库存盘点 | | |
| Commodity.Inbound(InboundVO vo) | 加入一个入库单 | | |
| Commodity.Stockout(StockoutVO vo) | 加入一个出库单 | | |
| Commodity.Subregion(Location beforeLocation, Location afterLocation) | 进行一次库存分区调整 | | |
| Commodity.getRegionInfo(Region region) | 进行一次库存分区查看 | | |

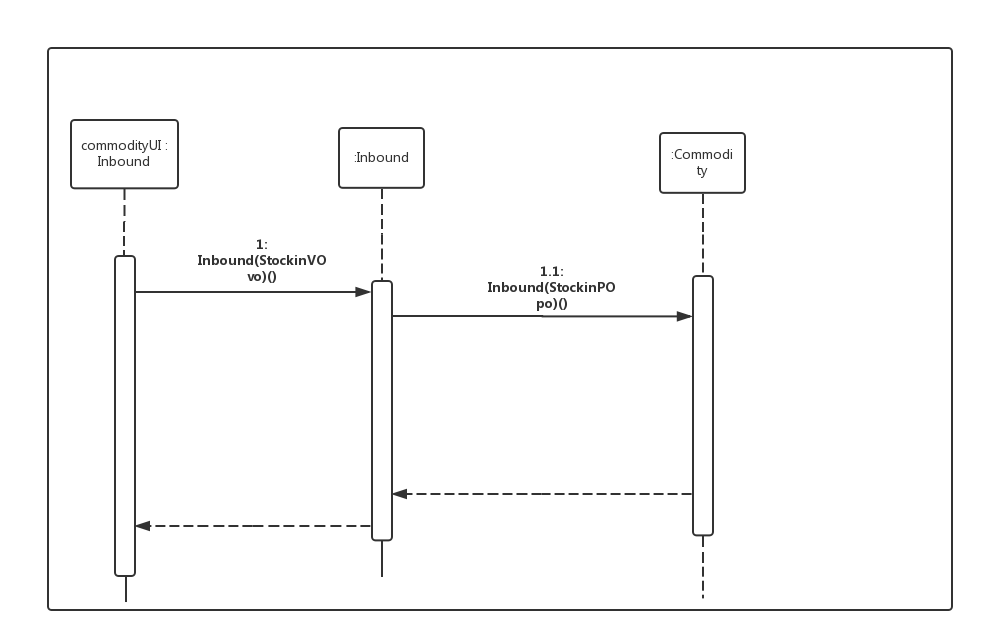
表C2 Commodity模块接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Commodity.getRegionInfo | 语法 | **public** RegionInfoVO getRegionInfo (String orgcode,**long** blocknum); |
| 前置条件 | 数据库中存在分区信息 |
| 后置条件 | 无 |
| Commodity.wheOverLoad | 语法 | **public** **boolean** wheOverLoad(StockinVO vo ) ; |
| 前置条件 | 数据库中存有警戒比例 |
| 后置条件 | 无 |
| Commodity.Inbound | 语法 | public ResultMessage Inbound(InboundVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中存入inbound的数据 |
| Commodity.updateInbound | 语法 | **public** ResultMessage update(StockinVO vo ); |
| 前置条件 | 数据库中存在相应的inbound数据 |
| 后置条件 | 向数据库中更新入新的inbound数据 |
| Commodity.findInboundForm | 语法 | **public** List<StockinVO> findInboundForm(Formstate state,String orgcode); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Commodity.Inboundgetid | 语法 | **public** String getid(String orgcode); |
| 前置条件 | 数据库中存在inbound的id |
| 后置条件 | 无 |
| Commodity.getEachBlockLimit | 语法 | **public** List<Long> getEachBlockLimit(**long** blocknum,String orgcode); |
| 前置条件 | 数据库中存有分区警戒值的信息 |
| 后置条件 | 无 |
| Commodity.wheConflict | 语法 | **public** **boolean** wheConflict(String orgcode,CommodityLocation location); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Commodity.Stockout | 语法 | public ResultMessage Stockout(StockoutVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中存入stockout信息 |
| Commodity.StockoutUpdate | 语法 | **public** ResultMessage update(StockoutVO vo) ; |
| 前置条件 | 数据库中存在相应的stockout数据 |
| 后置条件 | 无 |
| Commodity.StockoutFindFormss | 语法 | **public** List<StockoutVO> findforms(Formstate state,String orgcode) ; |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Commodity.StockoutGetid | 语法 | **public** String getid(String orgcode) ; |
| 前置条件 | 数据库中存在stockout的id |
| 后置条件 | 无 |
| Commodity.Stockexamination | 语法 | **public** StockExaminationVO StockExamination (String orgcode,**long** startTime, **long** endTime); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Commodity.Stocktaking | 语法 | **public** StocktakingVO Stocktaking(String orgcode,**long** date,**long** blocknum); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Commodity.StocktakingGetid | 语法 | **public** **long** getId(String orgcode,**long** date) ; |
| 前置条件 | 数据库中存有stocktaking的id |
| 后置条件 | 无 |
| Commodity.Subregion | 语法 | public ResultMessage Subregion(Location beforeLocation, Location afterLocation) |
| 前置条件 | 数据库中有分区信息 |
| 后置条件 | 无 |

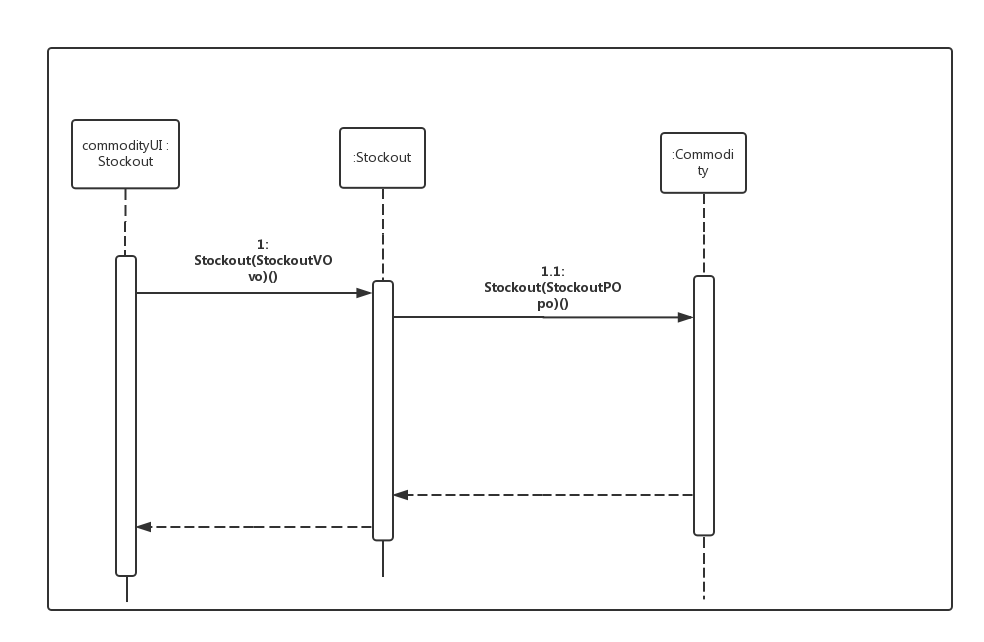
|  |  |
| --- | --- |
| 需要的服务(需接口) | |
| 服务名 | 服务 |
| DatabaseFactory.getCommodityDatabase | 得到Commodity数据库的服务的引用 |
| CommodityDataService.getallCommodity(String orgcode  ) | 根据机构编号得到数据库中所有CommodityPO对象 |
| CommodityDataService.getCommodityBetween(  Int starttime,int endtime) | 得到所有入库时间在starttime和endtime之间的CommodityPO对象 |
| CommodityDataService.addnew(CommodityPO po) | 向数据库中加入一个CommodityPO对象 |
| CommodityDataService.setOuttime(CommodityPO  po,long endtime) | 给某一Commodity对象填写出库时间 |
| CommodityDataService. updateCommodtiy(CommodityPO po) | 更新数据库中某一CommodityPO对象 |
| CommodityDataService.wheOverLoad(String orgcode,long blocknum) | 根据机构编号和区号返回该区是否达到警戒值 |
| CommodityDataService. getCommodityByLocation  (String orgcode,CommodityLocation location) | 根据机构编号和库存位置获取CommodityPO |
| CommodityDataService. getStockNumbyblocknum(long blocknum,String orgcode) | 根据机构编号和库存位置获取该区的排、架、位号的上限 |
| CommodityDataService.isEmpty(String orgcode,CommodityLocation location) | 根据机构编号和库存位置返回该位置是否为空 |
| CommodityDataService.getCommodityByCode(String orgcode,String ordercode) | 根据机构编号和订单号获取CommodityPO |
| CommodityDataService.getStopPoint(String orgcode,long date) | 根据机构编号和日期获取库存盘点的截止点 |
| CommodityDataService.stockTaking(String orgcode,long date,long blocknum) | 根据机构编号、日期和区号获取该区所有的CommodityPO |
| CommodityDataService. getCommodtiyByBlock(String orgcode,long blocknum) | 根据机构编号，区号获取该区所有的CommodityPO |
| CommodityDataService. getAllBlock(String orgcode) | 根据机构编号获取所有区号 |
| CommodityDataService. getCommodityNumber(String orgcode,long begin,long end) | 根据机构编号，开始时间、结束时间得到出库数量、入库数量、总数量 |
| CommodityDataService. getCommodityBetween(String orgcode,long starttime,long endtime) | 根据机构编号，开始时间、结束时间得到所有的CommodityPO |
| DatabaseFactory.getStockinFormDatabase | 得到StockinForm数据库的服务的引用 |
| StockinFormDataService.addStockinForm(StockinPO po) | 向数据库中插入StockinPO对象(付款单) |
| StockinFormDataService.getStockinForm(Formstate state) | 从数据库根据单据状态中得到StockinPO列表 |
| StockinFormDataService.updateStockinForm(StockinPO po) | 向数据库中更新StockinPO对象 |
| StockinFormDataService. getid(String orgcode) | 获取StockinPO的最新ID |
| StockinFormDataService. getStockNumbyblocknum(long blocknum,String orgcode) | 根据机构编号和库存位置获取该区的排、架、位号的上限 |
| StockinFormDataService. isEmpty(String orgcode,CommodityLocation location) | 根据机构编号和库存位置返回该位置是否为空 |
| StockoutFormDataService.addStockoutForm(StockoutPO po) | 向数据库中插入StockoutPO对象(付款单) |
| StockoutFormDataService.getStockoutForm(Formstate state) | 根据单据状态从数据库中得到StockoutPO列表 |
| StockoutFormDataService.updateStockoutForm(StockoutPO po) | 向数据库中更新StockoutPO对象 |
| StockoutFormDataService. getid(String orgcode) | 获取StockoutPO的最新ID |

(4)业务逻辑层的动态模型

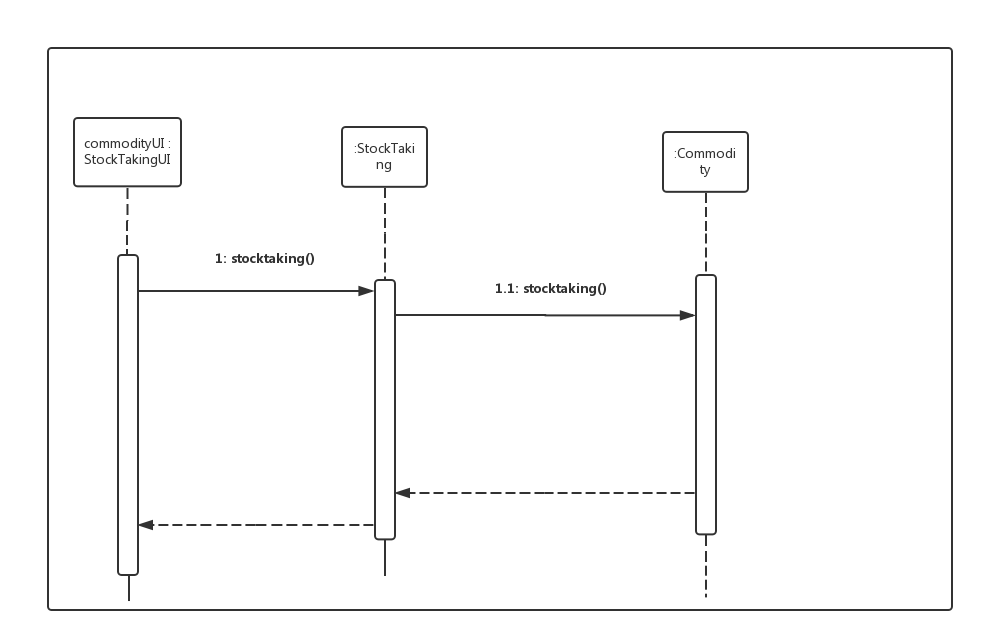
下图表明了快递物流系统中，当库存管理人员填写入库单时，库存方面的业务逻辑处理的相关对象之间的协作。



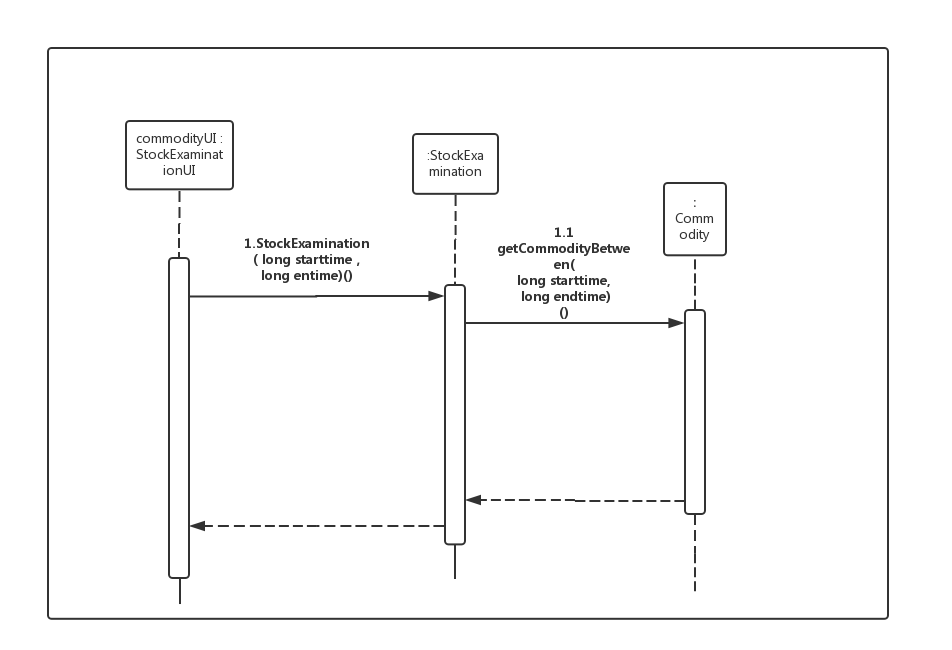
下图表明了快递物流系统中，当库存管理人员填写出库单时，库存方面的业务逻辑处理的相关对象之间的协作。



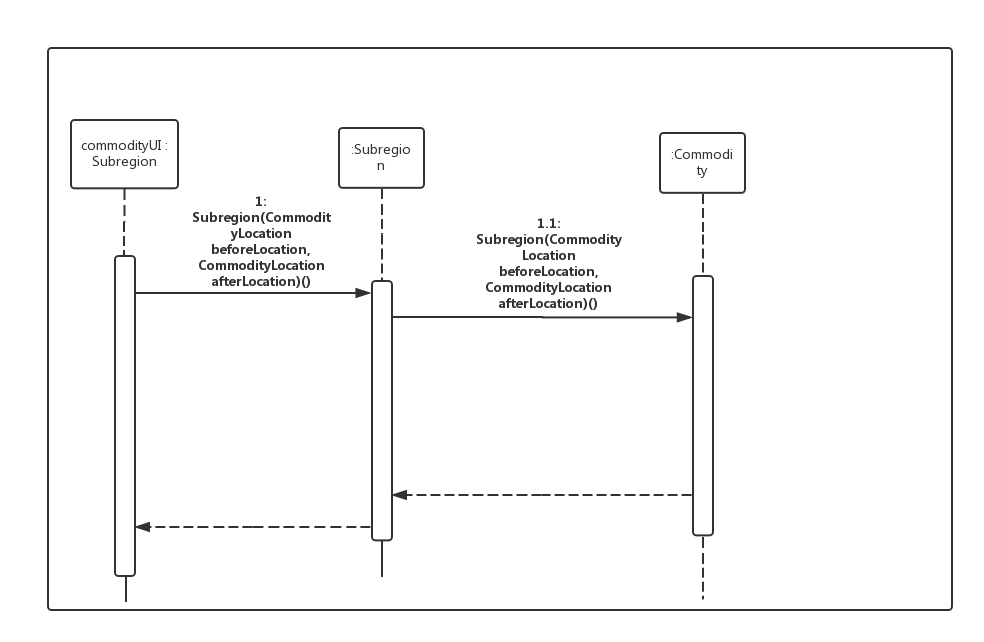
下图表明了快递物流系统中，当库存管理人员请求库存盘点时，库存方面的业务逻辑处理的相关对象之间的协作。



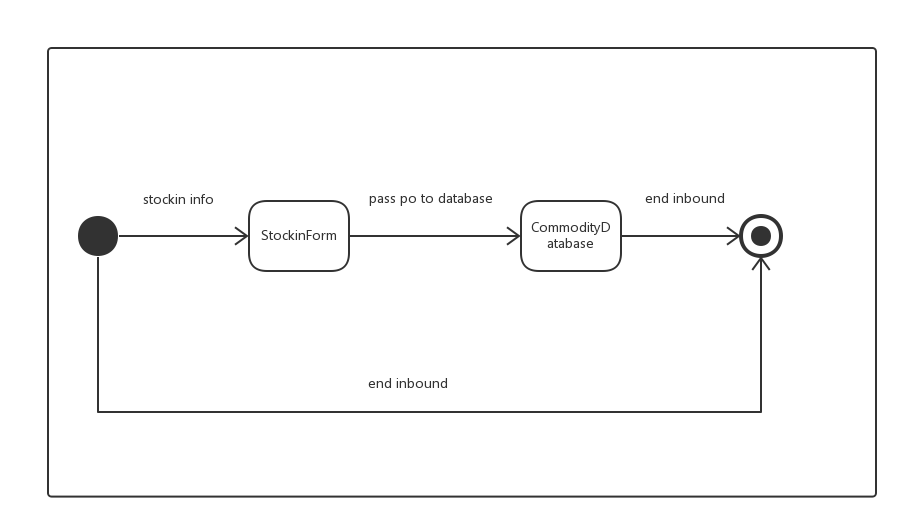
下图表明了快递物流系统中，当库存管理人员请求库存查看时，库存方面的业务逻辑处理的相关对象之间的协作。



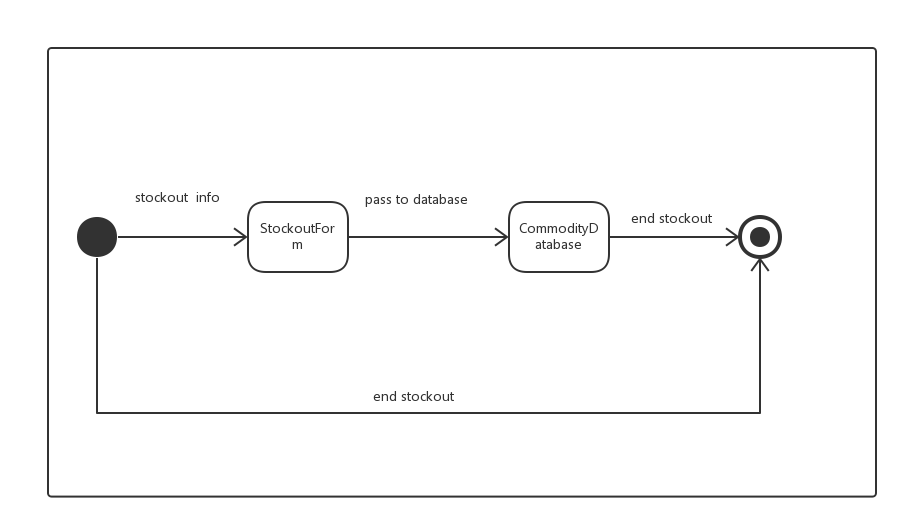
下图表明了快递物流系统中，当库存管理人员请求库存分区调整时，库存方面的业务逻辑处理的相关对象之间的协作。



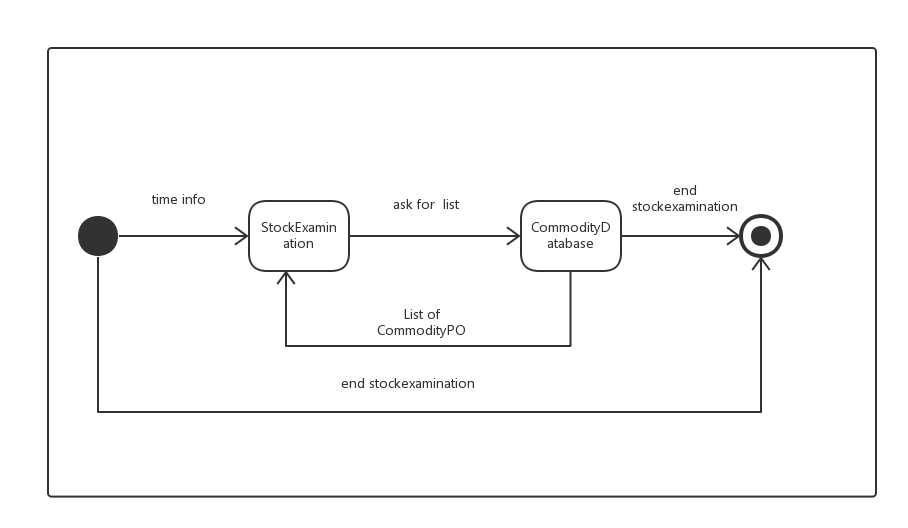
下图表示库存管理人员填写入库单时的状态图。



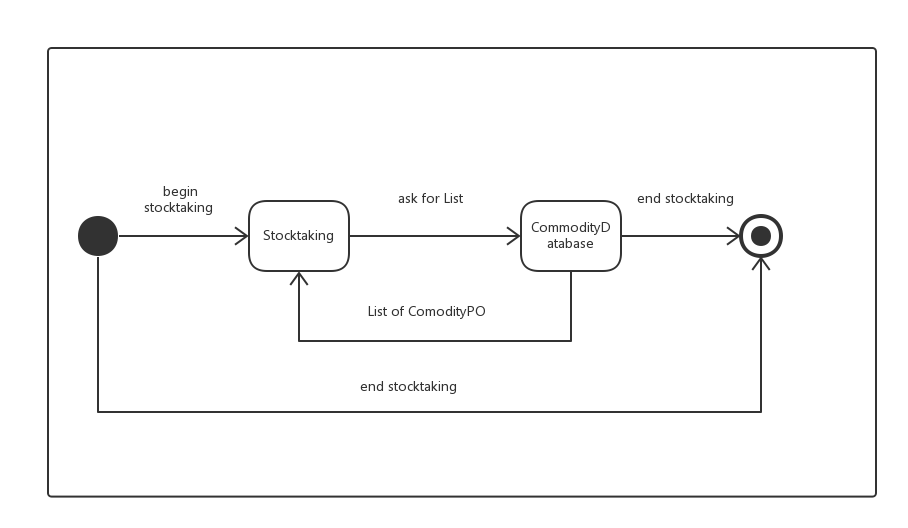
下图表示库存管理人员填写出库单时的状态图。



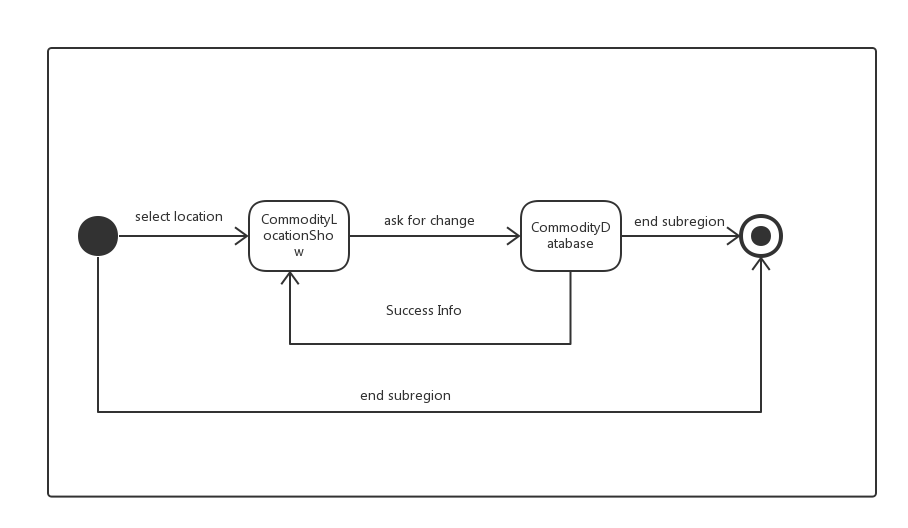
下图表示库存管理人员请求库存查看时的状态图。



下图表示库存管理人员请求库存盘点时的状态图。



下图表示库存管理人员请求库存分区调整时的状态图。



(5)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。

4.1.5 transportbl模块

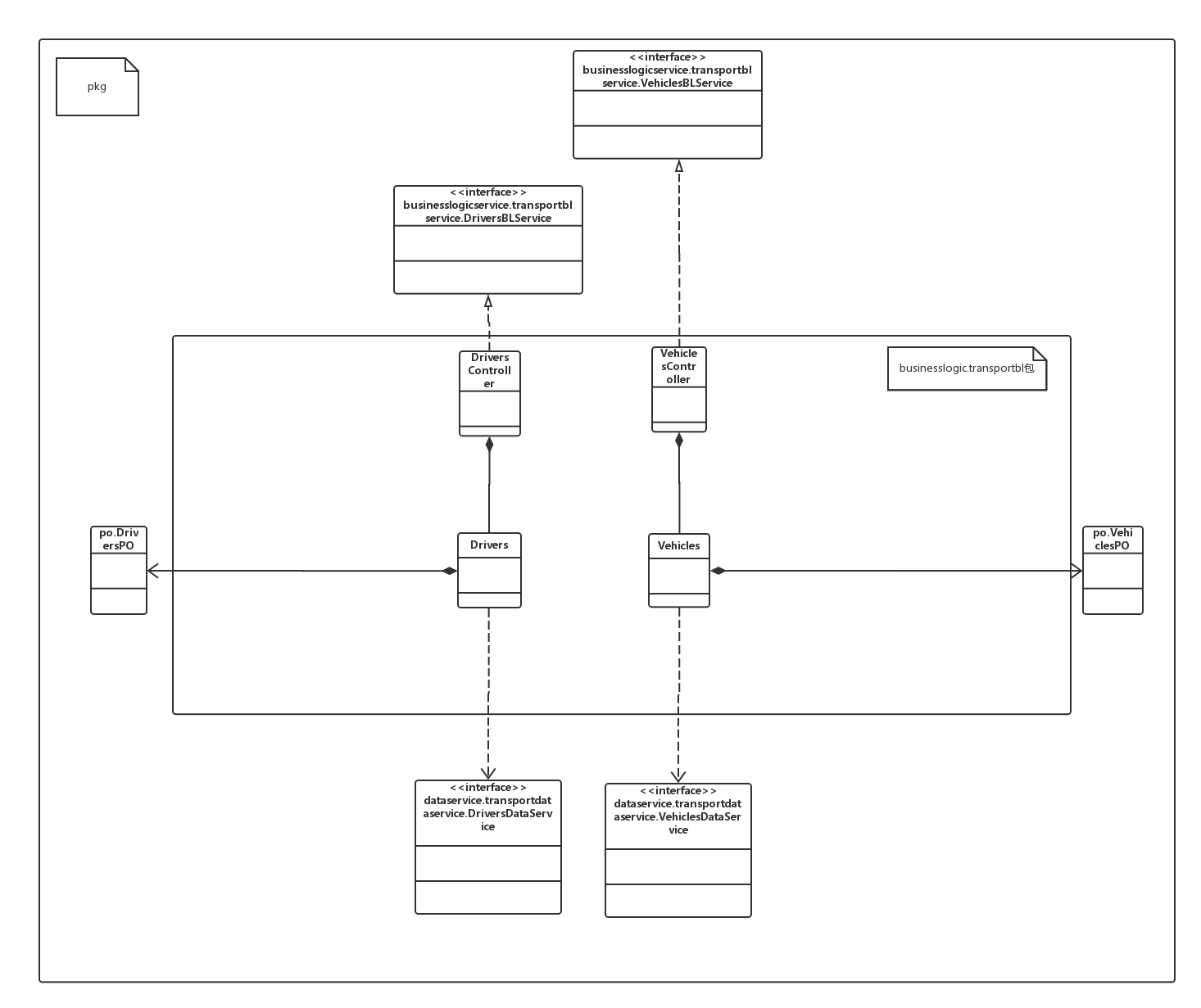
(1)模块概述

transportbl模块承担的需求参见需求规格说明文档功能需求和相关非功能性需求。 transportbl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层与业务逻辑层之间，我们添加businesslogicservice.transportblservice.DriversBLService接口。业务逻辑层和数据层之间添加dataservice.transportdataservice.Drivers DataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了DriversController,VehiclesController，这样DriversController会将对Drivers的业务逻辑处理委托给Drivers对象，VehiclesController会将对Vehicles的业务逻辑处理委托给Vehicles对象。DriversPO和VehiclesPO是作为交通信息管理的持久化对象被添加进设计模型中去的。

transportbl模块的设计如下图所示。



Transport模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| DriverController | 负责实现对应于司机界面需要的服务 |
| VehicleControoler | 负责实现车辆界面需要的服务 |
| Transport | 库存的领域模型对象，拥有一次司机信息界面或者交通信息界面处理所持有的司机信息，车辆信息等信息，可以帮助完成界面所需要的服务。 |

transport模块各个类的职责如表T1，T2所示。

表T1 TransportController模块接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| TransportController.AddDriver | 语法 | public ResultMessage addDriver (DriverVO vo) |
| 前置条件 | 已创建一个Transport领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver领域对象的AddDriver方法 |
| TransportController.DelDriver | 语法 | public ResultMessage delDriver() |
| 前置条件 | 已创建一个Transport领域对象 |
| 后置条件 | 调用Driver领域对象的DelDriver 方法 |
| TransportController.RevDriver | 语法 | public ResultMessage revDriver (DriverVO vo) |
| 前置条件 | 已创建一个Transport领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver领域对象的RevDriver方法 |
| TransportController.getDriverbyDN | 语法 | public DriverVO getDriverby DN(String driverNumber) |
| 前置条件 | 已创建一个Transport领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver领域对象的getDriverbyDN方法 |
| TransportController.getDriverbyName | 语法 | public ResultMessage getDriverbyName (String name) |
| 前置条件 | 已创建一个Transport领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver领域对象的getDriverby Name方法 |
| TransportController.getDriverId | 语法 | **public** String getid(String orgcode) ; |
| 前置条件 | 已创建一个Driver领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Driver领域对象的getid方法 |
| TransportController.AddVehicle | 语法 | public ResultMessage addVehicle (VehicleVo vo) |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Vehicle领域对象的AddVehicle方法 |
| TransportController.DelVehicle | 语法 | public ResultMessage delVehicle () |
| 前置条件 | 已创建一个Vehicle领域对象 |
| 后置条件 | 调用Vehicle领域对象的DelVehicle方法 |
| TransportController.RevVehicle | 语法 | public ResultMessage revVehicle (VehicleVo vo) |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Vehicle领域对象的RevVehicle方法 |
| TransportController.getVehiclebyVN | 语法 | public ResultMessage getVehiclebyVN (String vehicleNumber) |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Vehicle领域对象的getVehiclebyVN方法 |
| TransportController.getVehiclebyPN | 语法 | public ResultMessage getVehiclebyPN (String plateNumber) |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Vehicle领域对象的getVehiclebyPN方法 |
| TransportController.getVehicleId | 语法 | **public** String getid(String orgcode); |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Vehicle领域对象的getid方法 |
| 需要的服务(需接口) | | |
| Transport.AddDriver(DriverVO vo) | | 加入一个司机对象 |
| Transport.DelDriver(DriverVO vo) | | 删除一个司机对象 |
| Transport.RevDriver(DriverVO vo) | | 修改一个司机对象 |
| Transport.getDriverbyDN(long driverNumber) | | 根据司机编号查询一个司机对象 |
| Transport.getDriverbyName(String name) | | 根据司机姓名查询一个司机对象 |
| Transport.AddVehicle(VehicleVo vo) | | 加入一个车辆对象 |
| Transport.DelVehicle(VehicleVO vo) | | 删除一个车辆对象 |
| Transport.RevVehicle(VehicleVO vo) | | 修改一个车辆对象 |
| Transport.getVehiclebyVN(String vehicleNumber) | | 根据车辆编号查询一个车辆信息 |
| Transport.getVehiclebyPN(String plateNumber) | | 根据车牌号查询一个车辆信息 |

表T2 transport模块接口规范

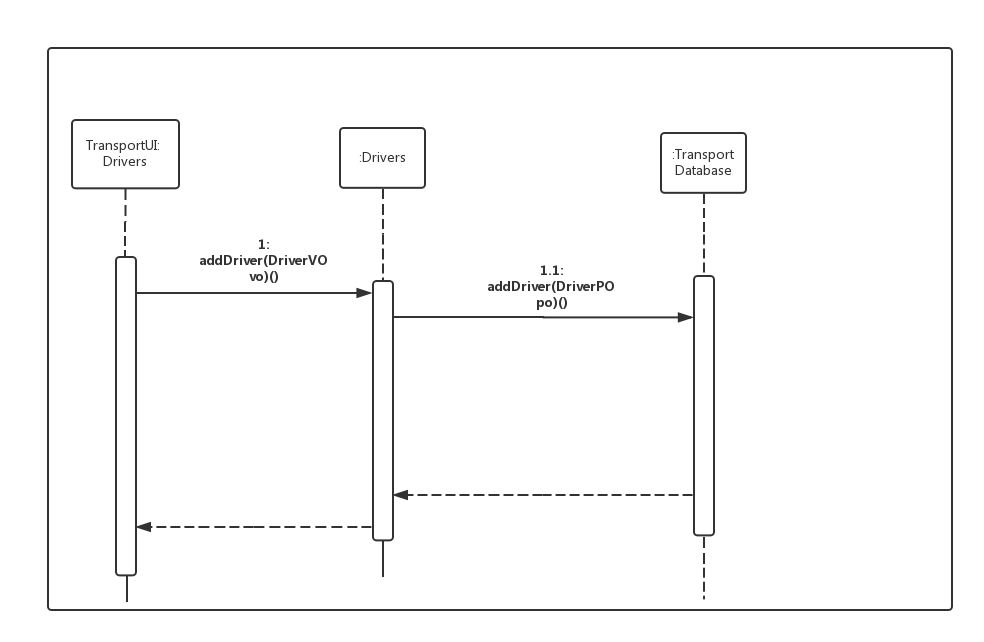
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Transport.AddDriver | 语法 | public ResultMessage addDriver (DriverVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入driver信息 |
| Transport.DelDriver | 语法 | public ResultMessage delDriver() |
| 前置条件 | 数据库中已经存在对应的driver信息 |
| 后置条件 | 在数据库中删除相应的driver信息 |
| Transport.RevDriver | 语法 | public ResultMessage revDriver (DriverVO vo) |
| 前置条件 | 数据库中已经存在对应的driver信息 |
| 后置条件 | 在数据库中修改相应的driver信息 |
| Transport.getDriverbyDN | 语法 | public DriverVO getDriverby DN(String driverNumber) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Transport.getDriverbyName | 语法 | public ResultMessage getDriverbyName (String name) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Transport.getDriverId | 语法 | **public** String getid(String orgcode) ; |
| 前置条件 | 数据库中存有driver的id |
| 后置条件 | 无 |
| Transport.AddVehicle | 语法 | public ResultMessage addVehicle (VehicleVo vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入vehicle信息 |
| Transport.DelVehicle | 语法 | public ResultMessage delVehicle () |
| 前置条件 | 数据库中存在相应的vehicle信息 |
| 后置条件 | 删除数据库中的相应的vehicle数据 |
| Transport.RevVehicle | 语法 | public ResultMessage revVehicle (VehicleVo vo) |
| 前置条件 | 数据库中存在相应的vehicle信息 |
| 后置条件 | 更新数据库中的相应的vehicle数据 |
| Transport.getVehiclebyVN | 语法 | public ResultMessage getVehiclebyVN (String vehicleNumber) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Transport.getVehiclebyPN | 语法 | public ResultMessage getVehiclebyPN (String plateNumber) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Transport.getVehicleId | 语法 | **public** String getid(String orgcode); |
| 前置条件 | 数据库中存在vehicle的id |
| 后置条件 | 无 |

|  |
| --- |
| 需要的服务(需接口) |

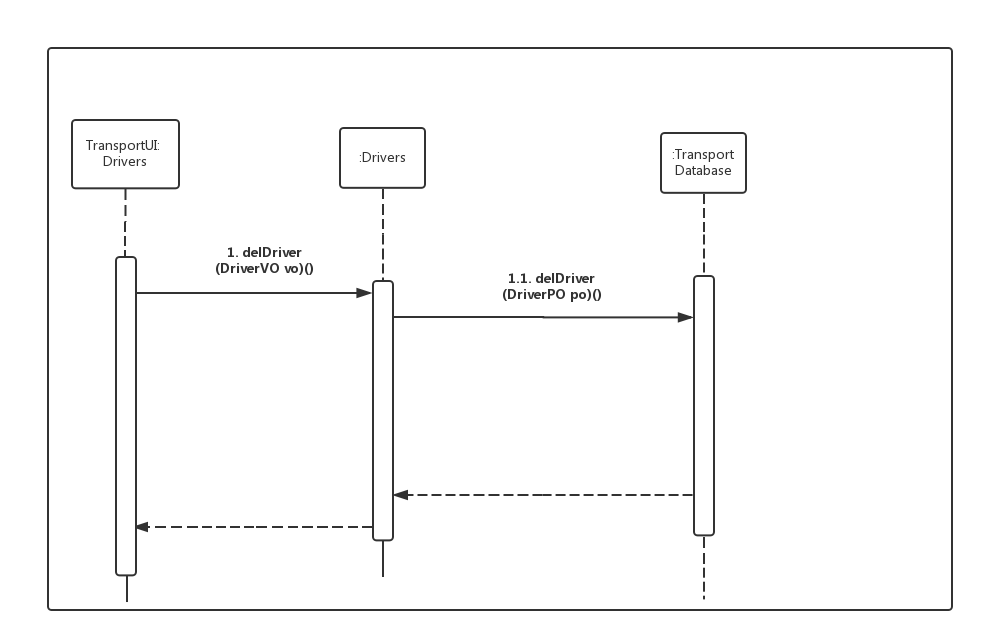
|  |  |
| --- | --- |
| DatabaseFactory. getDriversDatabase | 获取Drivers数据库的引用 |
| DriversDataService.insertDriver(DriversPO po) | 在数据库中插入DriversPO对象 |
| DriversDataService.deleteDriver(DriversPO po) | 在数据库中删除DriversPO对象 |
| DriversDataService.updateDriver(DriversPO po) | 在数据库中修改更新某一DriversPO对象 |
| DriversDataService. findDriverBynum(String identity) | 在数据库中根据司机编号查找某一DriversPO对象 |
| DriversDataService. findDriverByname(String identity) | 在数据库中根据司机姓名查找某一DriversPO对象 |
| DriversDataService. getCurrentID(String orgcode) | 根据机构编号获取DriversPO的最新ID |
| DatabaseFactory. getVehiclesDatabase | 获取Vehicles数据库的引用 |
| VehiclesDataService. insertVehicle(VehiclesPO po) | 在数据库中插入VehiclesPO对象 |
| VehiclesDataService. deleteVehicle(VehiclesPO po) | 在数据库中删除VehiclesPO对象 |
| VehiclesDataService.updateVehicle(VehiclesPO po) | 在数据库中修改更新某一VehiclesPO对象 |
| VehiclesDataService. findVehicleBycode(String identity) | 在数据库中根据车辆编号查找某一VehiclesPO对象 |
| VehiclesDataService. findVehicleByPlateNumber(String plateNumber) | 在数据库中根据车牌号查找某一VehiclesPO对象 |
| VehiclesDataService. getCurrentID(String orgcode) | 根据机构编号获取VehiclesPO的最新ID |

(4)业务逻辑层的动态模型

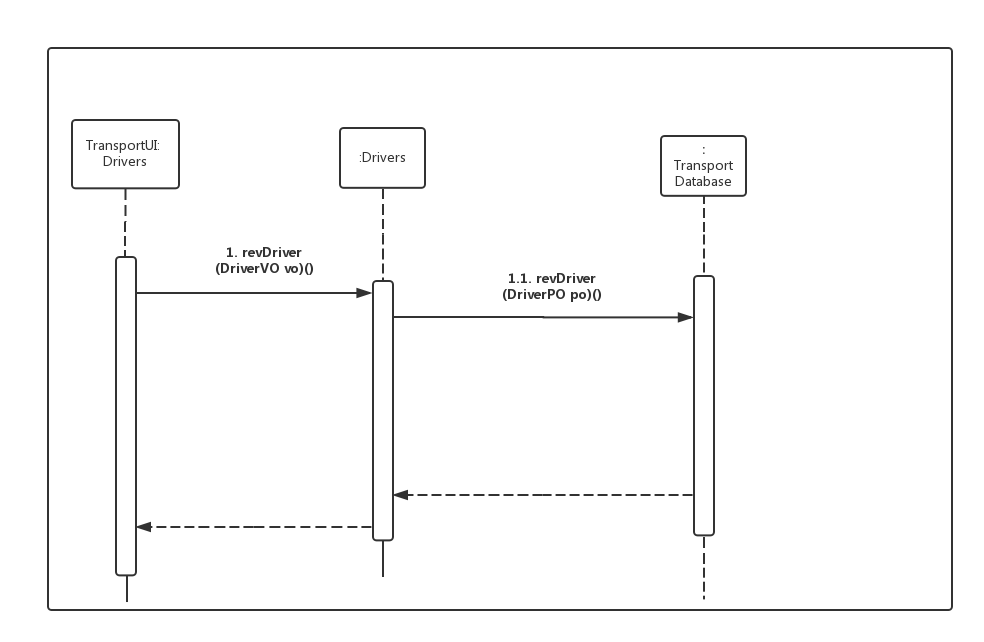
下图表明了当营业厅业务员进行司机信息增加时，司机业务逻辑处理的相关对象之间的协作。



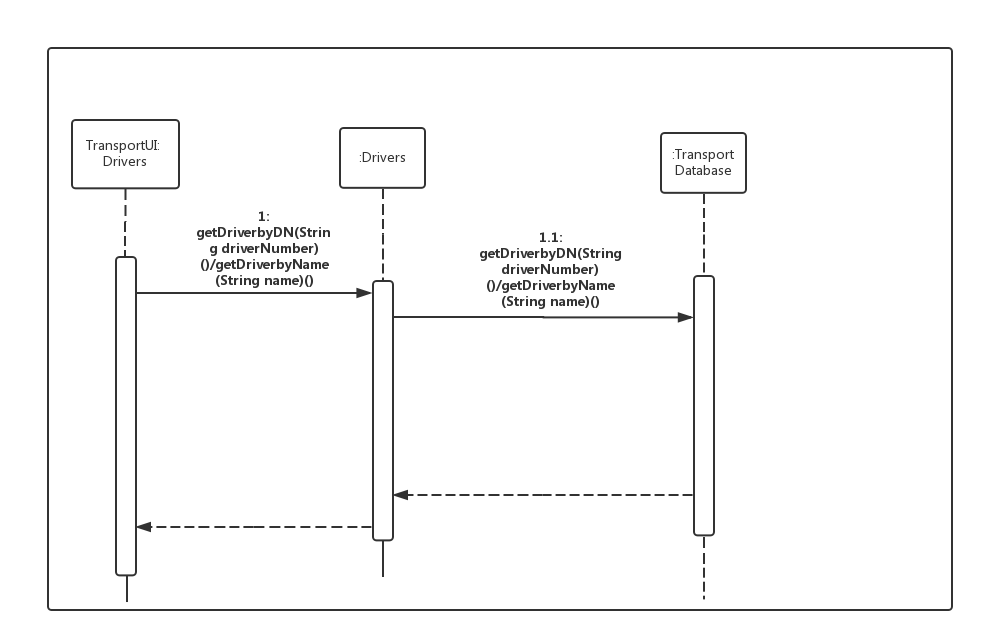
下图表明了当营业厅业务员进行司机信息删除时，司机业务逻辑处理的相关对象之间的协作。



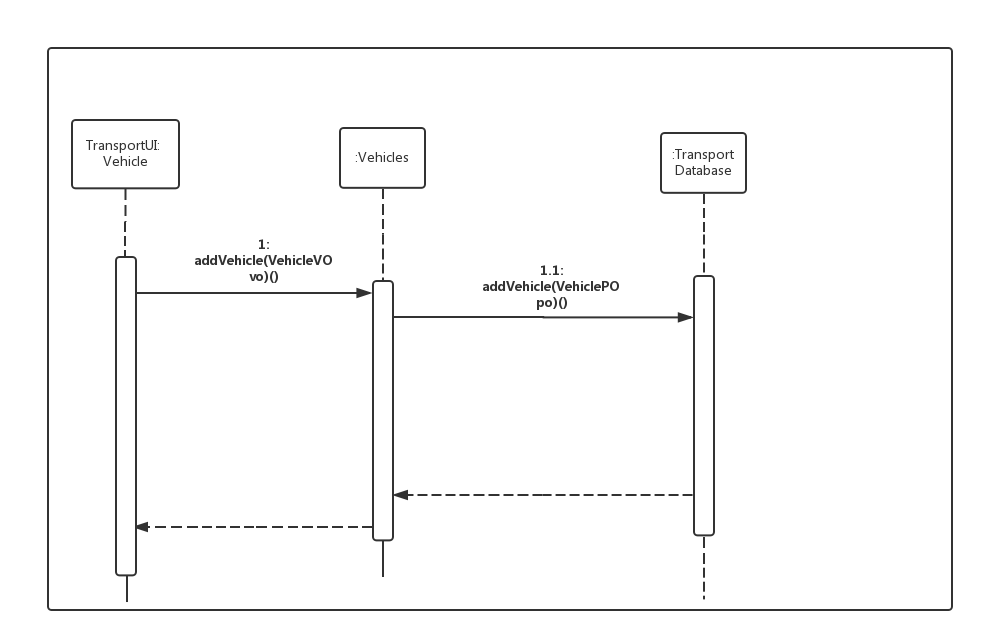
下图表明了当营业厅业务员进行司机信息修改时，司机业务逻辑处理的相关对象之间的协作。



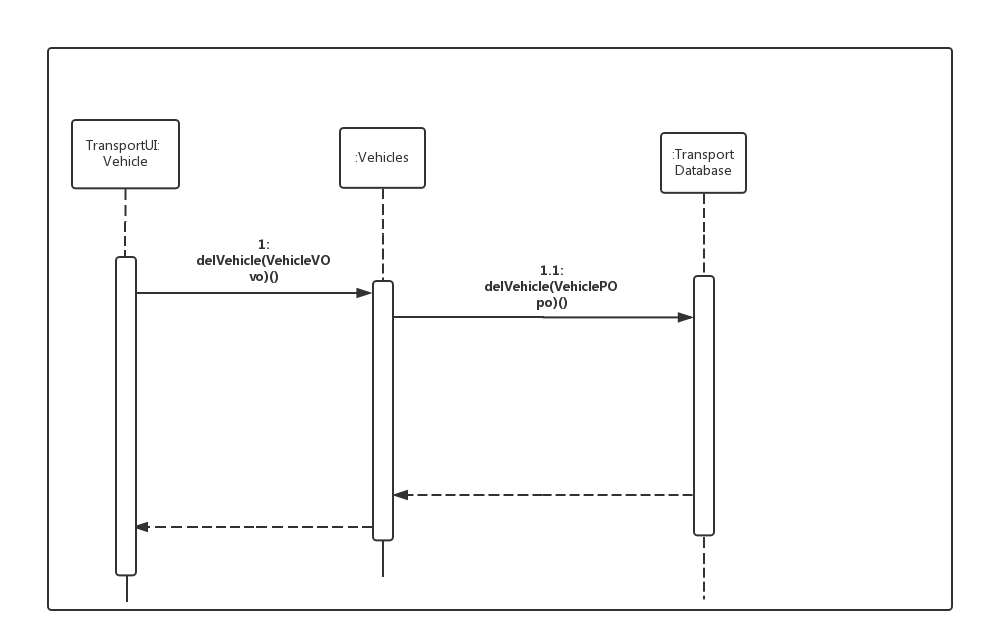
下图表明了当营业厅业务员进行司机信息查询时，司机业务逻辑处理的相关对象之间的协作。



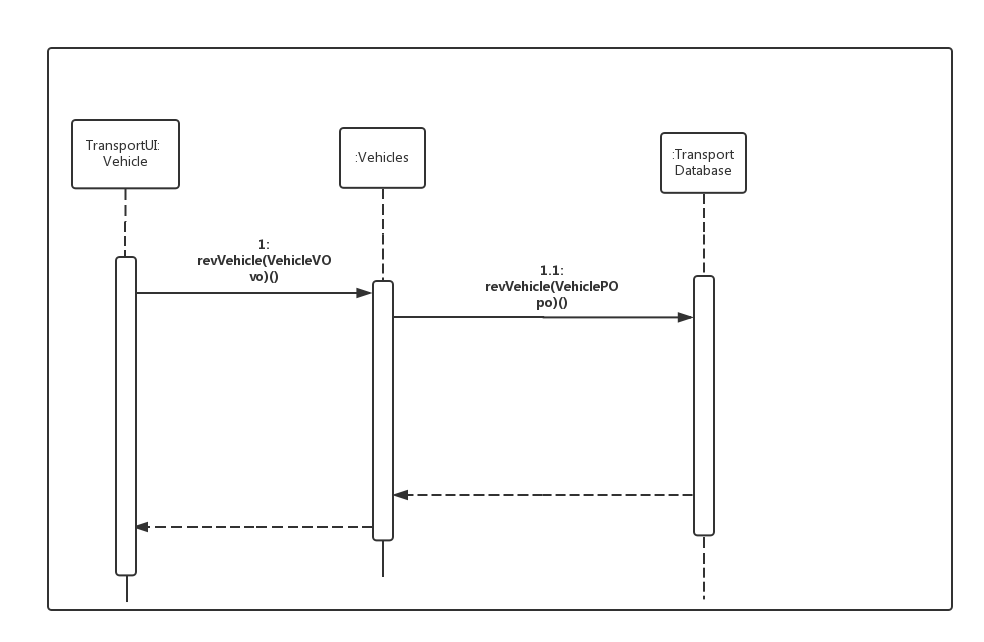
下图表明了当营业厅业务员进行车辆信息增加时，车辆业务逻辑处理的相关对象之间的协作。



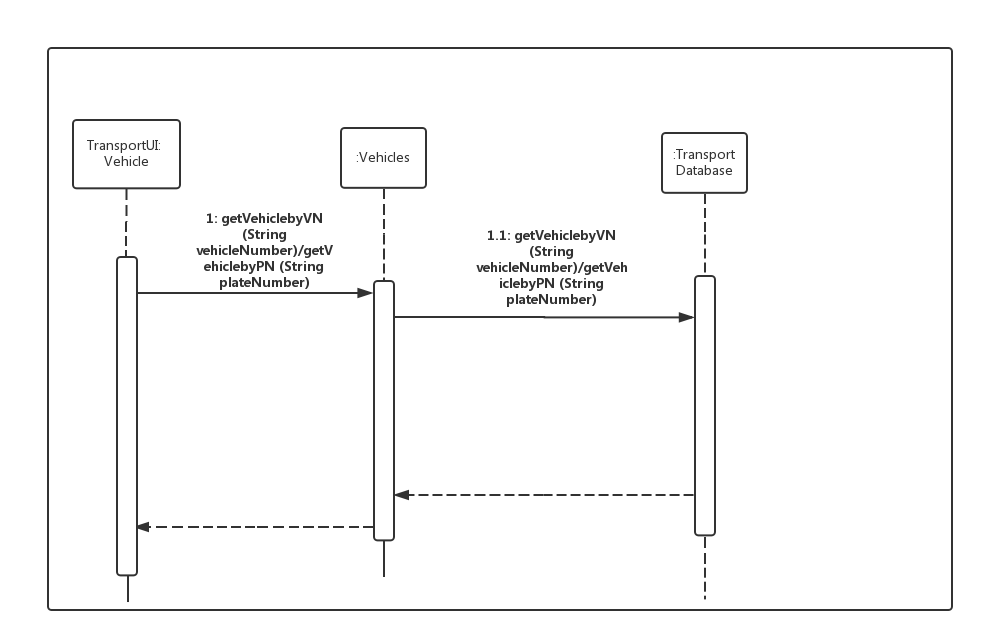
下图表明了当营业厅业务员进行车辆信息删除时，车辆业务逻辑处理的相关对象之间的协作。



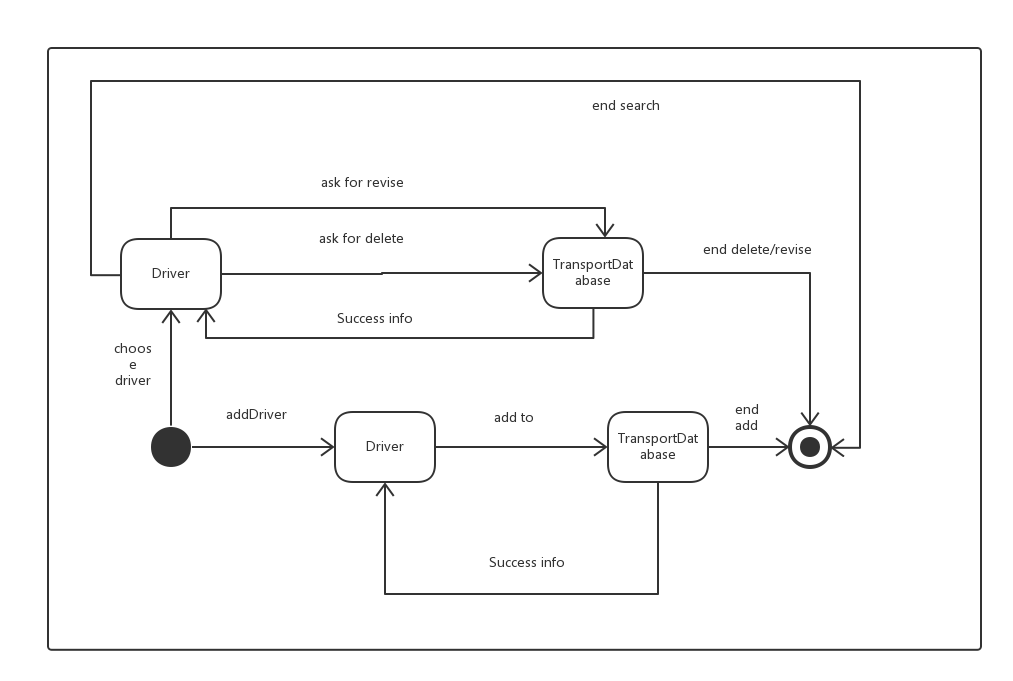
下图表明了当营业厅业务员进行车辆信息修改时，车辆业务逻辑处理的相关对象之间的协作。



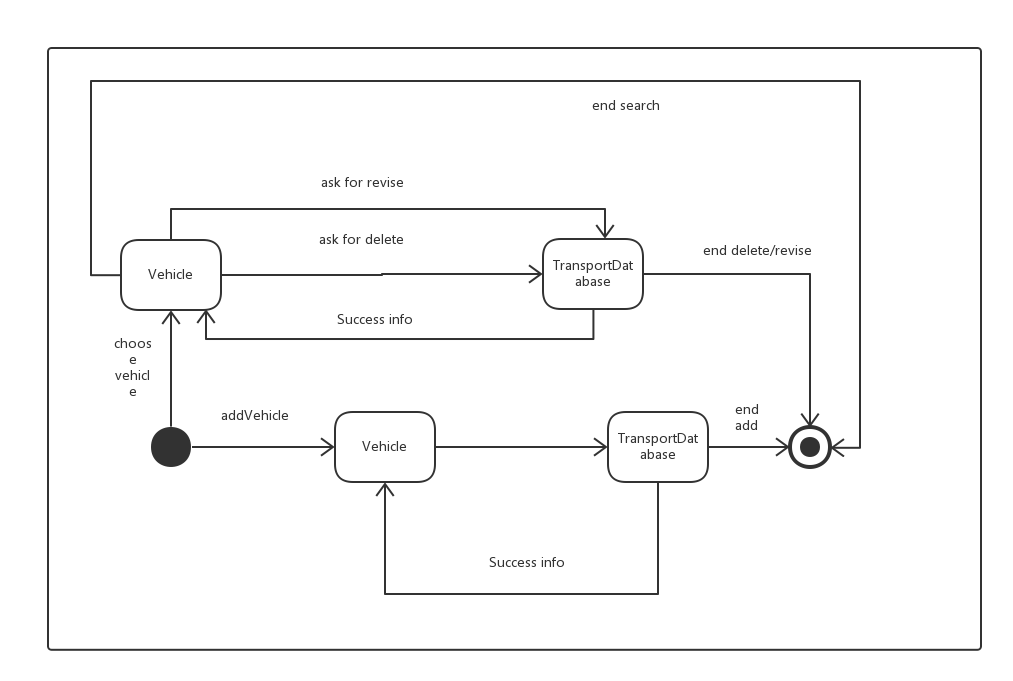
下图表明了当营业厅业务员进行车辆信息查询时，车辆业务逻辑处理的相关对象之间的协作。



下图表明了当营业厅业务员进行司机信息管理时的状态图



下图表明了当营业厅业务员进行车辆信息管理时的状态图



(5)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。

4.1.6 Userbl模块

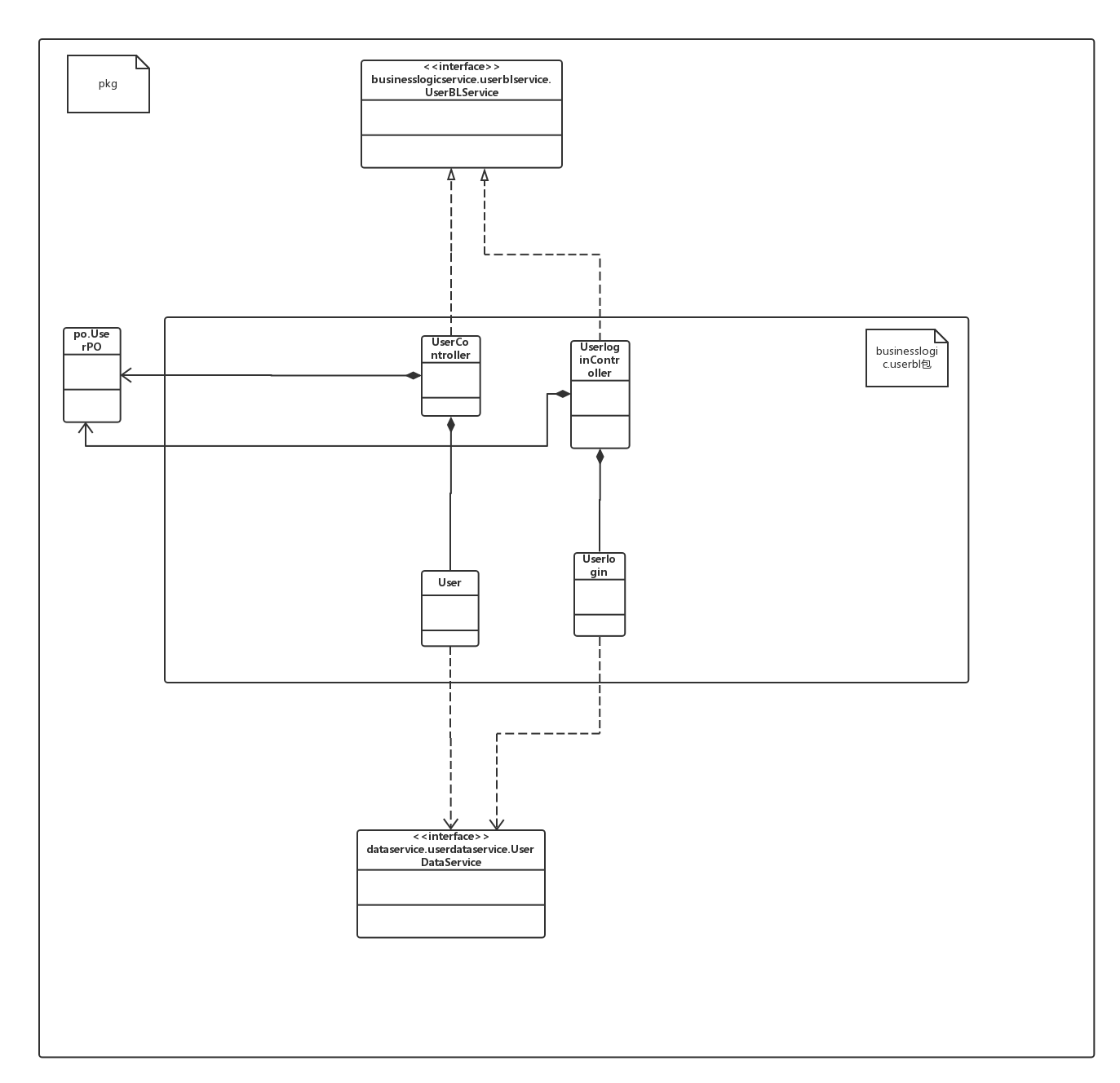
(1)模块概述

Userbl模块承担的需求参见需求规格说明文档功能需求和相关非功能性需求。 Userbl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层与业务逻辑层之间，我们添加businesslogicservice.Userblservice.UserBLService接口。业务逻辑层和数据层之间添加dataservice.Userdataservice.UserDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了UserController,UserloginController这样UserController会将对User的业务逻辑处理委托给User对象，UserloginController会将对Userlogin的业务逻辑处理委托给Userlogin对象。UserPO作为用户管理的持久化对象被添加进设计模型中去的。

userbl模块的设计如下图所示。



user模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登录界面所需要的服务 |
| UserController | 负责实现对应与用户增删改查界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的姓名和密码，可以解决登录问题。拥有一次用户信息界面处理所持有的用户信息等信息，可以帮助完成界面所需要的服务。 |

User模块各个类的职责如表U1，U2所示。

表U1 UserController模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| UserController.Login | 语法 | public ResultMessage login(long id,String password) | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的Login方法 | |
| UserController.AddUser | 语法 | public ResultMessage addUser(UserVO vo) | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的AddUser方法 | |
| UserController.DelUser | 语法 | public ResultMessage delUser(UserVO vo) | |
| 前置条件 | 已创建一个User领域对象 | |
| 后置条件 | 调用User领域对象的DelUser方法 | |
| UserController.RevUser | 语法 | public ResultMessage revUser(UserVO vo) | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的RevUser方法 | |
| UserController.getUser | 语法 | public UserVO getUserbyName(long id) | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的getUserByName方法 | |
| UserController.findOrgByCode | 语法 | **public** OrganizationVO findOrgByCode(String orgcode); | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的findOrgByCode方法 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 |
| User.Login(long id,String password) | | | 登录一个用户对象 |
| User.AddUser(UserVO vo) | | | 加入一个用户对象 |
| User.DelUser(UserVO vo) | | | 删除一个用户对象 |
| User.RevUser(UserVO vo) | | | 修改一个用户对象 |
| User.getUser(long id) | | | 根据id得到一个用户对象 |

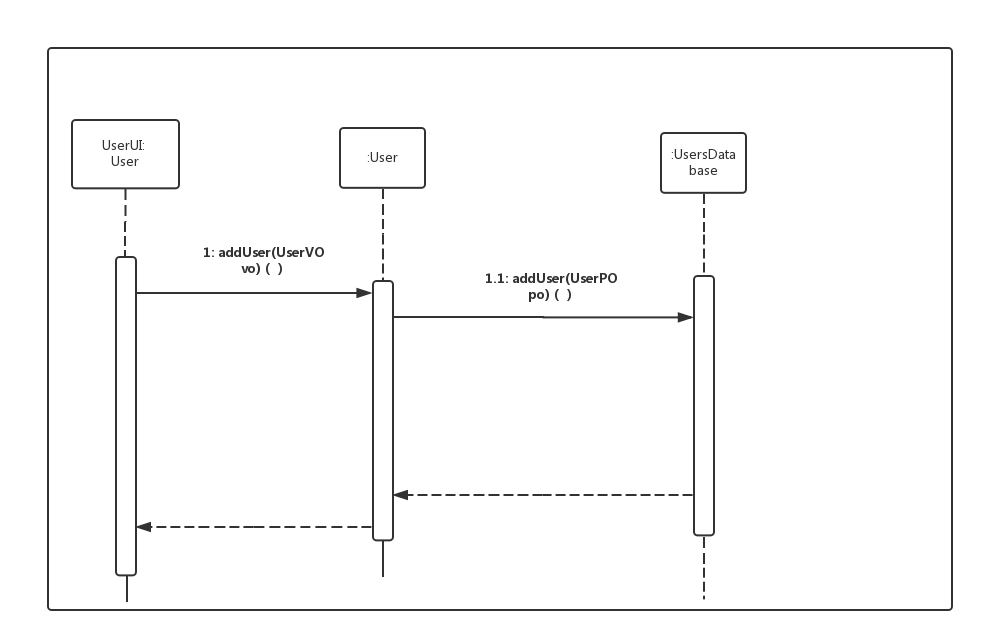
表U2 user模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| User.Login | 语法 | public ResultMessage login(long id,String password) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| User.AddUser | 语法 | public ResultMessage addUser(UserVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入user数据 |
| User.DelUser | 语法 | public ResultMessage delUser(UserVO vo) |
| 前置条件 | 数据库中存在相应的user数据 |
| 后置条件 | 删除数据库中相应的user数据 |
| User.RevUser | 语法 | public ResultMessage revUser(UserVO vo) |
| 前置条件 | 数据库中存在相应的user数据 |
| 后置条件 | 修改数据库中相应的user数据 |
| User.getUser | 语法 | public UserVO getUserbyName(long id) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| User.findOrgByCode | 语法 | **public** OrganizationVO findOrgByCode(String orgcode); |
| 前置条件 | 无 |
| 后置条件 | 无 |

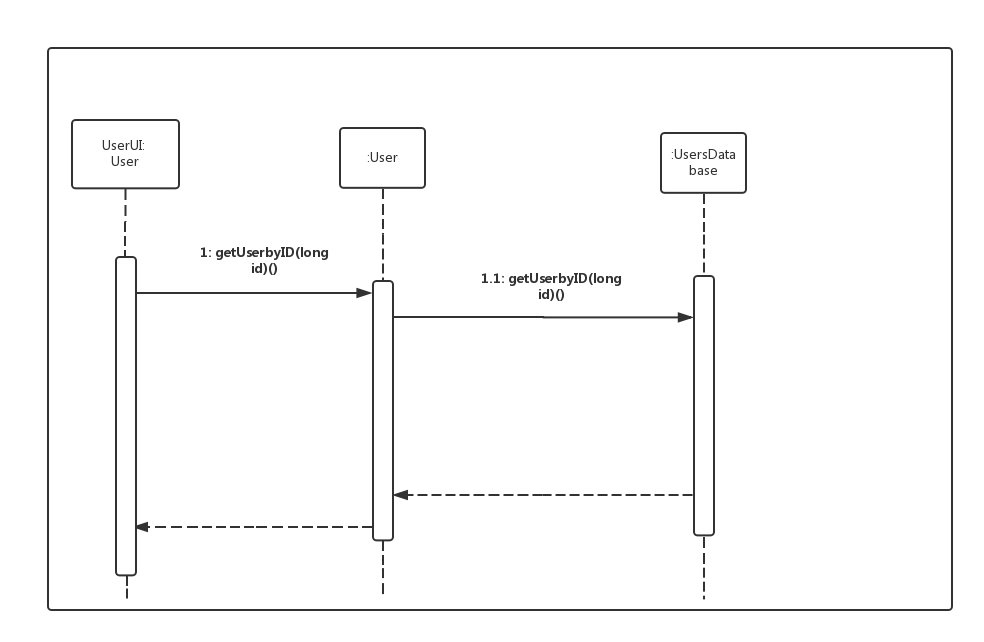
|  |  |
| --- | --- |
| 需要的服务(需接口) | |
| DatabaseFactory.getUserDatabase | 得到User数据库的服务的引用 |
| UserDataService.insert(UserPO po) | 在数据库中插入UserPO对象 |
| UserDataService.delete(UserPO po) | 在数据库中插入UserPO对象 |
| UserDataService.update(UserPO po) | 在数据库中修改更新某一UserPO对象 |
| UserDataService.find(long username) | 在数据库中根据用户名查找某一UserPO对象 |
| UserDataService. getOrganization(String organizationcode) | 在数据库中根据机构编号获取OrganizationPO |

(4)业务逻辑层的动态模型

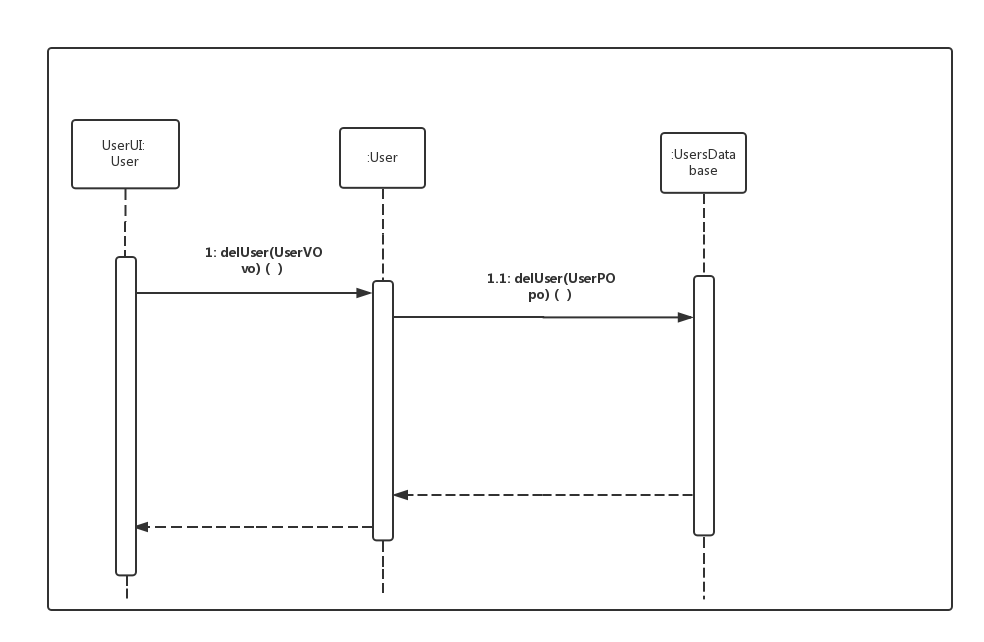
下图表明当管理员进行用户增加时的顺序图



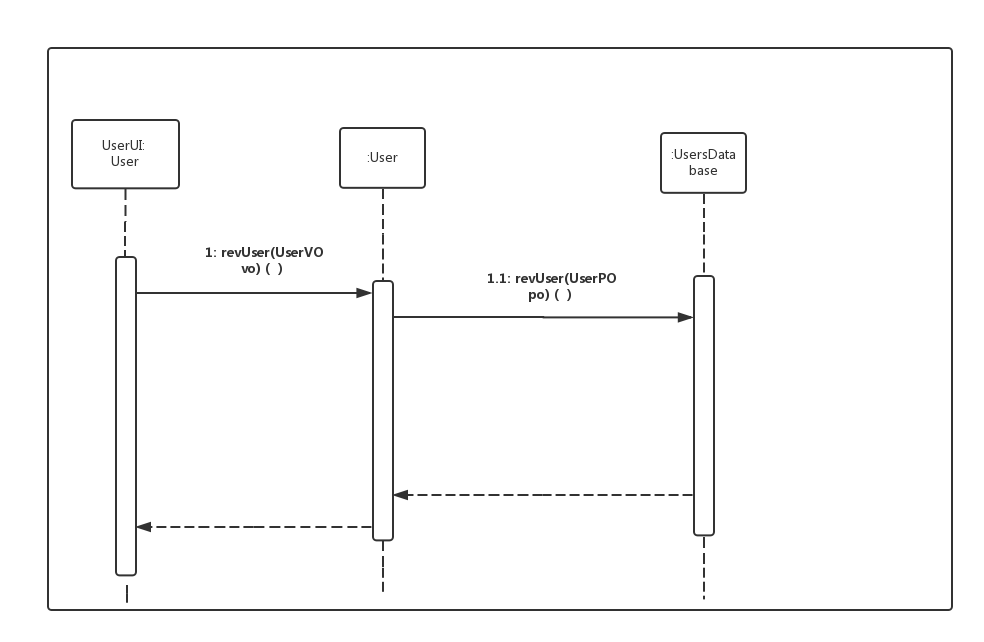
下图表明当管理员进行用户查询时的顺序图



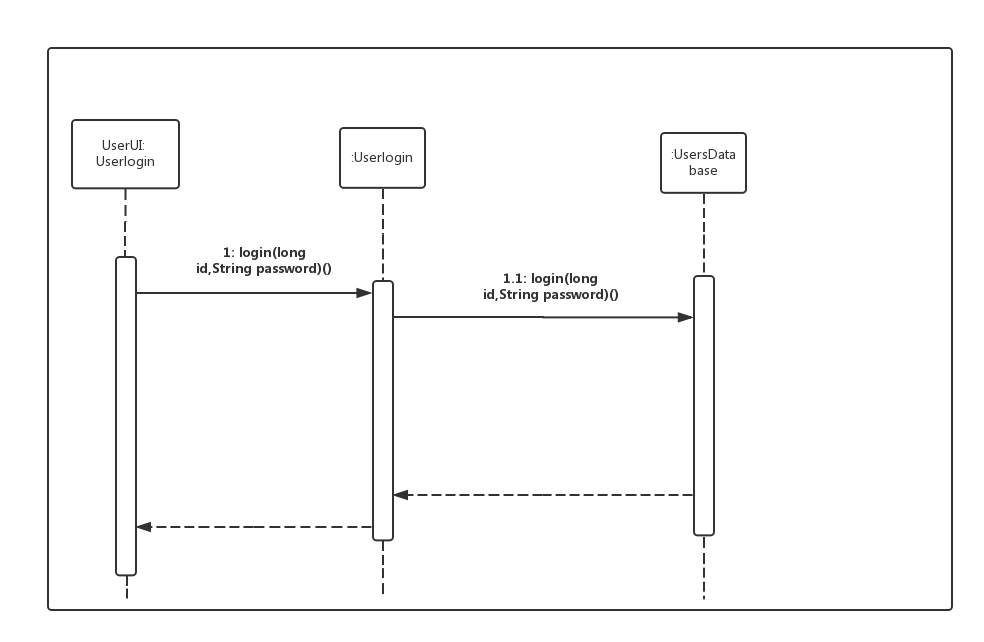
下图表明当管理员进行用户删除时的顺序图



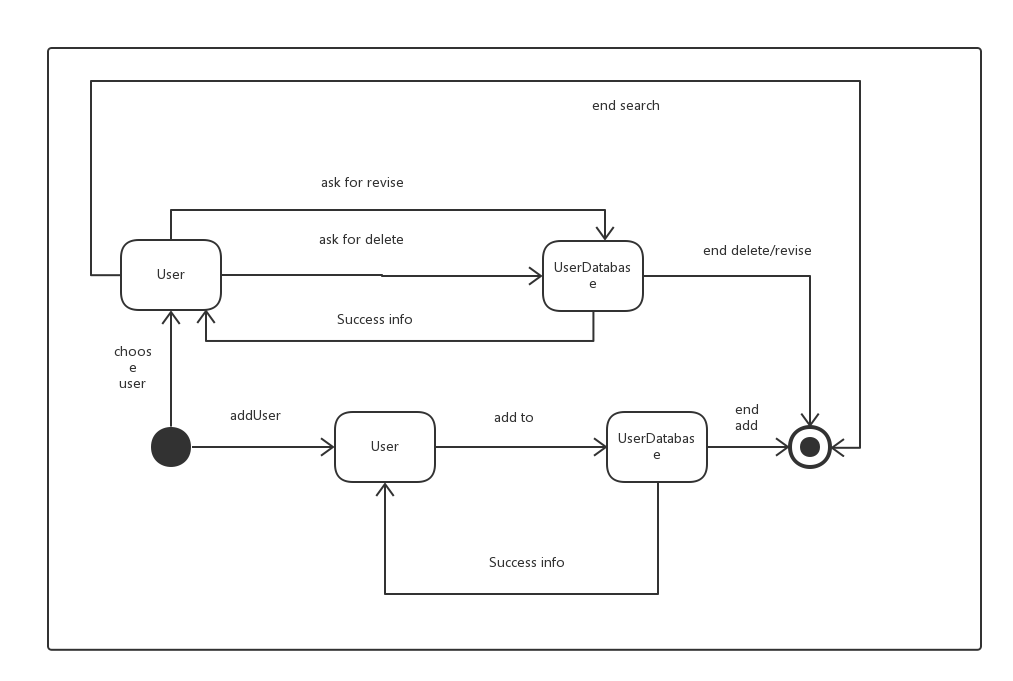
下图表明当管理员进行用户修改时的顺序图



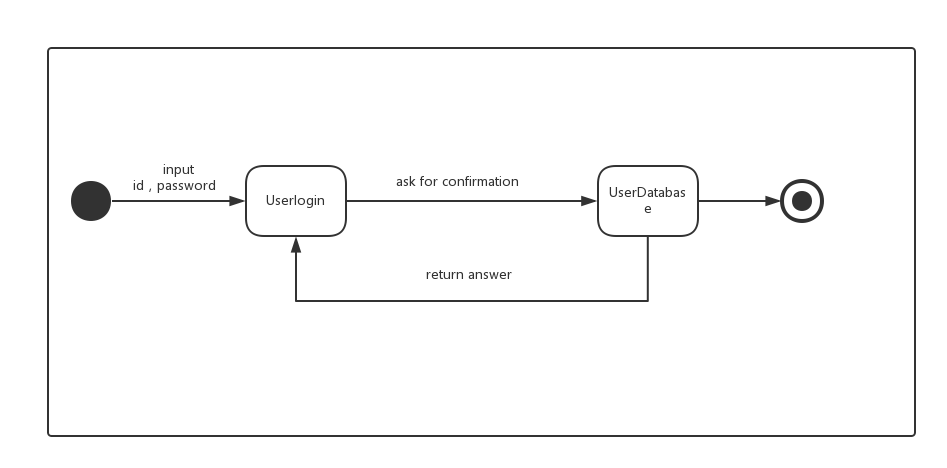
下图表明当管理员进行用户登录管理时的顺序图



下图表明管理员进行用户管理时的状态图



下图表明管理员进行用户登录管理时的状态图



(5)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。

4.1.7orderbl模块

(1)模块概述

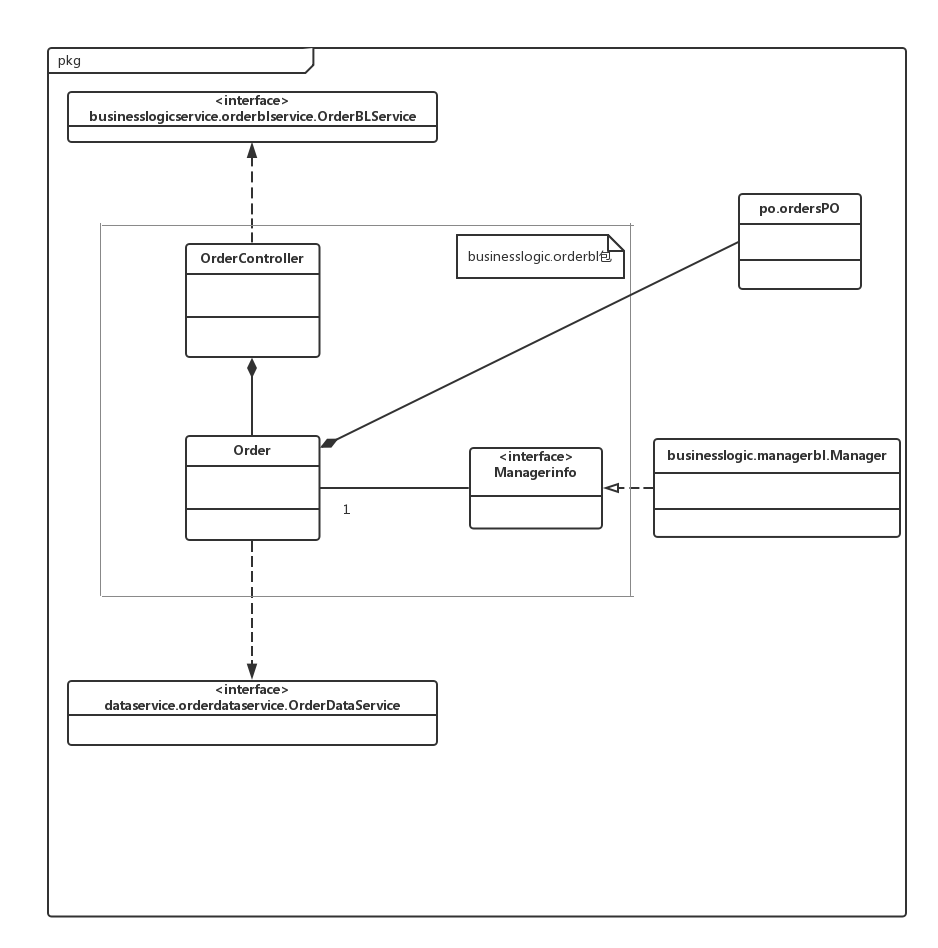
Orderbl模块承担的需求参加需求规格说明文档功能需求及其相关非功能需求。

Orderbl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示曾和业务逻辑层之间，我们添加businesslogicservice.orderblservice.OrderBLService接口。业务逻辑层和数据层之间添加dataservice.orderdataservice.OrderDataService。为了隔离业务逻辑职责和逻辑控制职责，我们增加了OrderController，这样OrderController会将对销售的业务逻辑处理委托给Order对象。ordersPO是作为订单记录的持久化对象被添加到设计模型去的。Managerinfo是根据以来倒置原则，为了消除循环以来而产生的接口。

orderbl模块的设计如下图所示



orderbl模块各个类的设计

orderbl模块各个类的职责如下表所示。

**orderbl模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| ReceptionController | 负责实现对应于收件界面所需要的服务 |
| OrderController | 负责实现订单输入界面所需要的服务 |
| Order | 订单输入和收件的领域模型对象，拥有一次订单输入所持有的寄件人、收件人地址，快件重量，题记等信息，可以帮助完成订单输入界面所需要的服务等信息 |

(3)模块内部类的接口规范

OrderController和Order的接口规范如下表所示。

**OrderController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| OrderController.addOrder | 语法 | **public** ResultMessage addOrder(OrderVO vo,String orgcode); | |
| 前置条件 | 已创建一个Order领域对象，输入信息符合规范 | |
| 后置条件 | 调用order领域对象的addOrder方法 | |
| OrderController.orderFare | 语法 | **public** **double** orderFare(OrderFareVO vo); | |
| 前置条件 | 已创建一个Order领域对象，输入信息符合规范 | |
| 后置条件 | 调用order领域对象的orderFare方法 | |
| OrderController.orderFare | 语法 | **public** **double** orderFare(OrderFareVO vo); | |
| 前置条件 | 已创建一个Order领域对象，输入信息符合规范 | |
| 后置条件 | 调用order领域对象的orderFare方法 | |
| OrderController. getOrdercode | 语法 | public String getOrdercode(String orgcode); | |
| 前置条件 | 已创建一个Order领域对象，输入符合规范 | |
| 后置条件 | 调用order领域对象的getOrdercode方法 | |
| OrderController. findOrderByState | 语法 | public List<OrderVO> findOrderByState(Formstate documentstate,String orgcode); | |
| 前置条件 | 已创建一个Order领域对象，输入符合规范 | |
| 后置条件 | 调用order领域对象的findOrderByState方法 | |
| OrderController. updateLogistics | 语法 | public ResultMessage updateLogistics(OrderVO vo); | |
| 前置条件 | 已创建一个Order领域对象，输入符合规范 | |
| 后置条件 | 调用order领域对象的updateLogistics方法 | |
| OrderController.reception | 语法 | public ResultMessage reception (ReceptionVO vo) | |
| 前置条件 | 已创建一个Order领域对象，输入信息符合规范 | |
| 后置条件 | 调用order领域对象的reception方法 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 |
| Order.order(OrderVO vo) | | | 加入一个订单对象 |
| Order.Computefare(OrderFareVO vo) | | | 返回订单的运费 |
| Order.reception(ReceptionVO vo) | | | 更新订单对象的收件信息 |

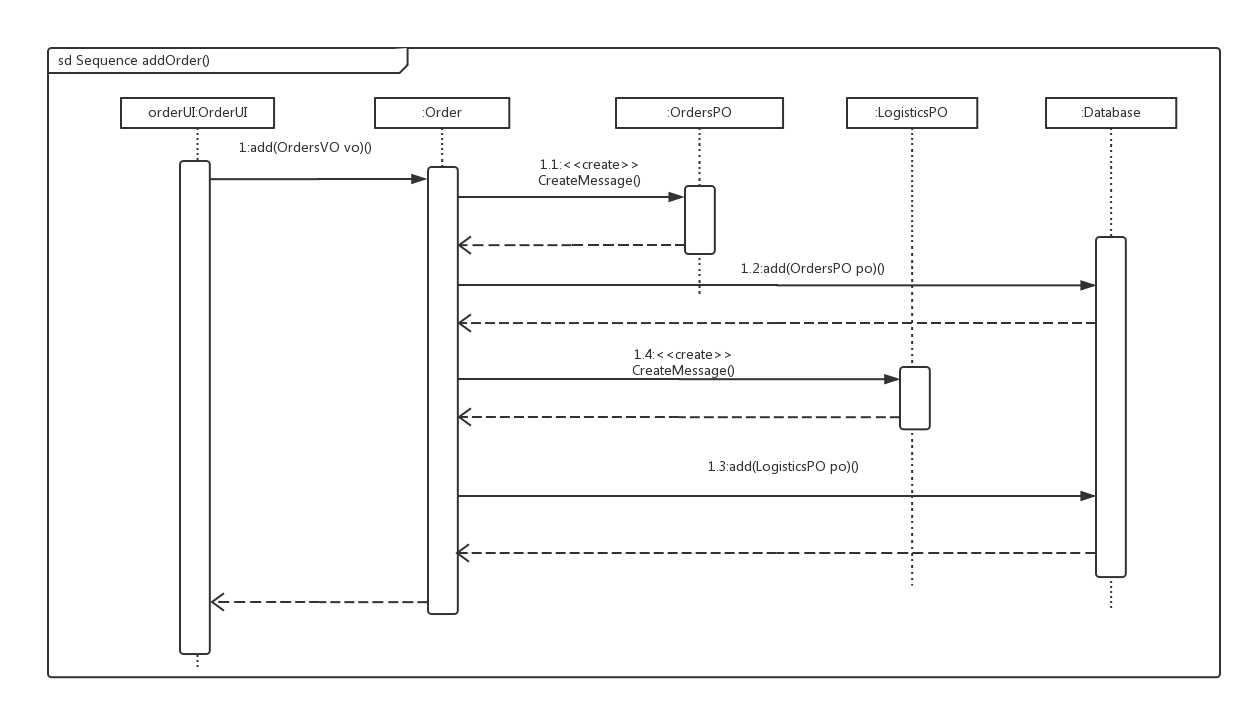
**Order的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Order.addOrder | 语法 | **public** ResultMessage addOrder(OrderVO vo,String orgcode); |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入order信息 |
| Order.orderFare | 语法 | **public** **double** orderFare(OrderFareVO vo); |
| 前置条件 | 数据库中存有价格信息 |
| 后置条件 | 无 |
| Order. getOrdercode | 语法 | public String getOrdercode(String orgcode); |
| 前置条件 | 数据库中存有order的id |
| 后置条件 | 无 |
| Order. findOrderByState | 语法 | public List<OrderVO> findOrderByState(Formstate documentstate,String orgcode); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order. updateLogistics | 语法 | public ResultMessage updateLogistics(OrderVO vo); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.reception | 语法 | public ResultMessage reception (ReceptionVO vo) |
| 前置条件 | 数据库中有相应的order数据对应 |
| 后置条件 | 向数据库中加入reception数据 |

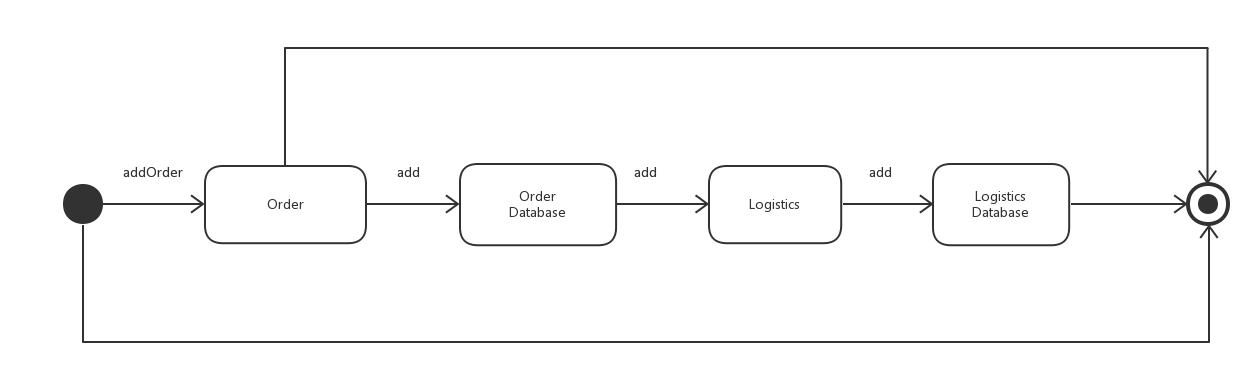
|  |
| --- |
| 需要的服务(需接口) |

|  |  |
| --- | --- |
| DatabaseFactory.getOrdersDa  Tabase | 得到Orders数据库的服务和引用 |
| DatabaseFactory.getLogisticsDa  Tabase | 得到Logistics数据库的服务和引用 |
| OrdersDataService.addOrderForm(OrdersPO Po) | 向数据库中插入OrdersPO对象 |
| OrdersDataService.getOrderForm(Formstate state，,String orgcode) | 从数据库中根据单据状态和机构编号得到OrdersPO对象 |
| OrdersDataService.getOrderForm(String Ordernum) | 从数据库中得到OrdersPO对象 |
| OrdersDataService.updateOrderForm(OrderPO po) | 向数据库中更新OrdersPO对象 |
| OrdersDataService. findLastID(String orgcode) | 获取OrderPO的最新ID |

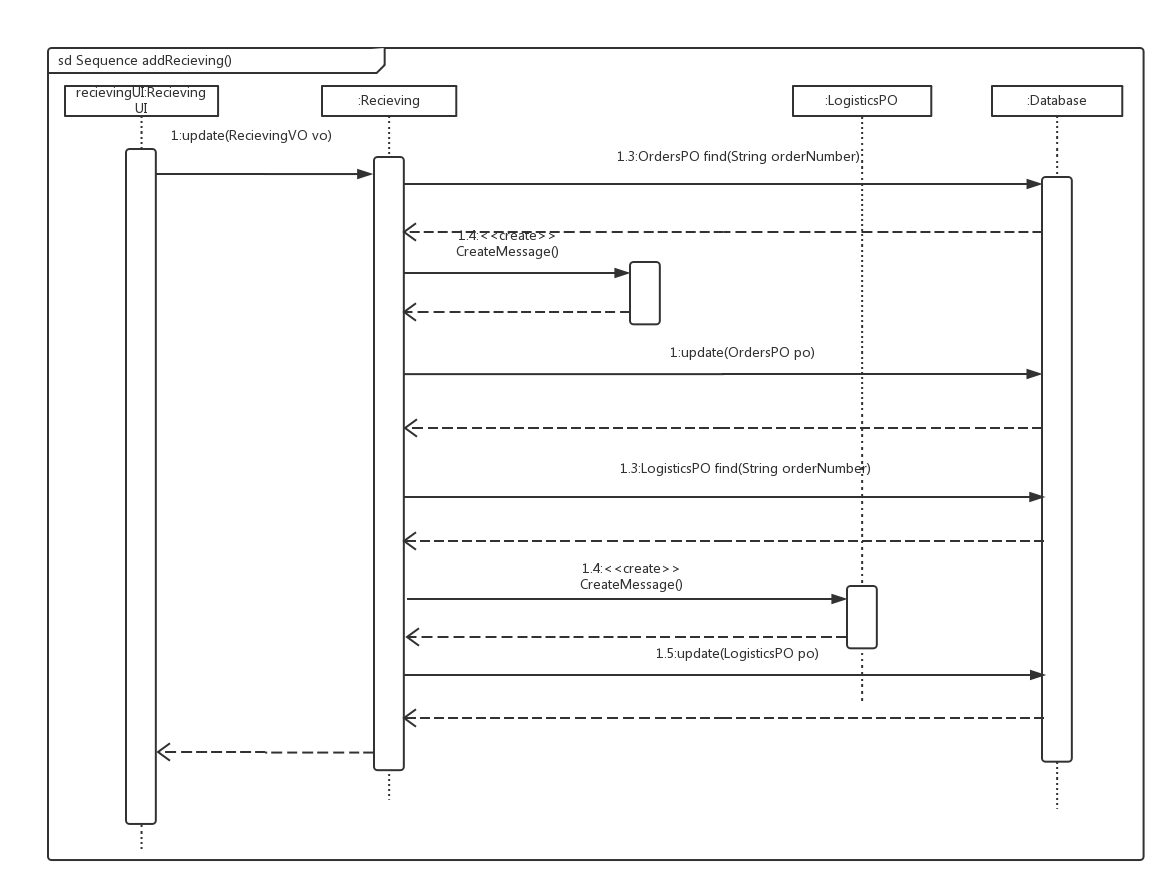
(4)业务逻辑层的动态模型

下图为输入订单时的顺序图

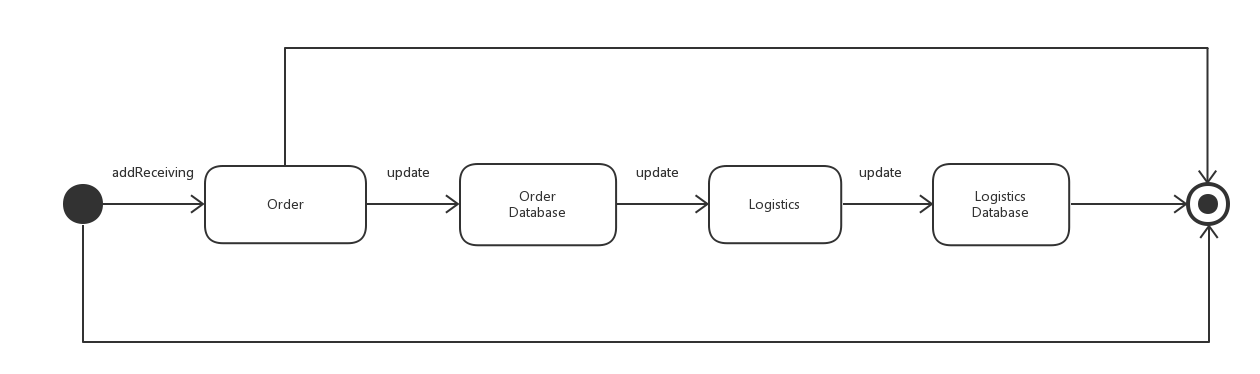
下图为输入订单时的状态图



下图为输入收件信息时的顺序图



下图为输入收件信息时的状态图



(5)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。

4.1.8managerbl模块

(1)模块概述

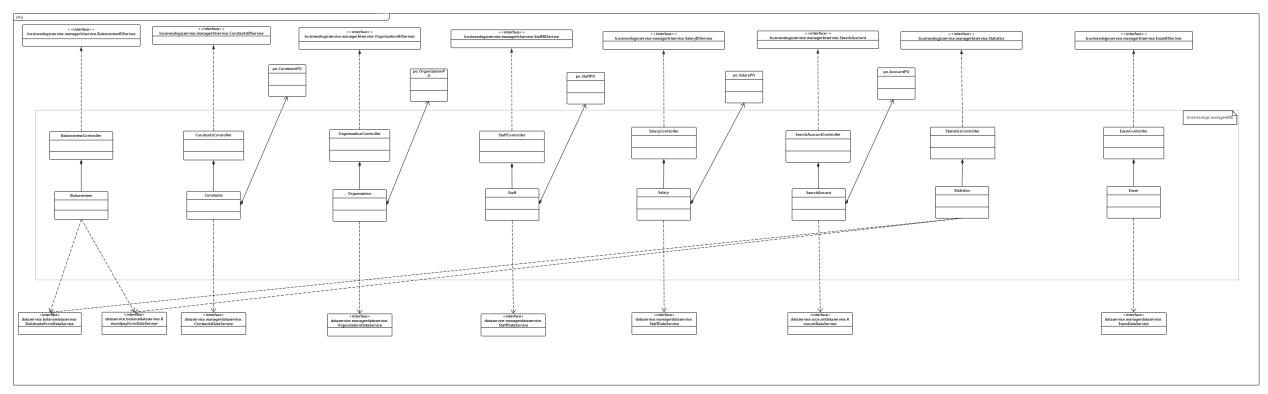
Managerbl模块承担的需求参加需求规格说明文档功能需求及其相关非功能需求。

Managerbl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示曾和业务逻辑层之间，我们添加businesslogicservice.managerblservice.ManagerBLService接口。业务逻辑层和数据层之间添加dataservice.managerdataservice.ManagerDataService。为了隔离业务逻辑职责和逻辑控制职责，我们增加了BalanceviewController,ConstantsController,OrganizationController,StaffController,SalaryController,SearchAccountController,StatisticsController,ExamController，这样BalanceviewController,ConstantsController,OrganizationController,StaffController,SalaryController,SearchAccountController,StatisticsController,ExamController会将对销售的业务逻辑处理委托给Balanceview ,Constants ,Organization ,Staff ,Salary ,SearchAccount ,Statistics ,Exam 对象。ConstantsPO,OrganizationPO,StaffPO,SalaryPO,AccountPO是作为管理记录的持久化对象呗添加到设计模型去的。

Managerbl模块的设计如下图所示。

managerbl模块各个类的设计

Managerbl模块各个类的职责如下表所示。

**Managerbl模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| LoginController | 负责实现对应于登录界所需要的服务 |
| BalanceviewController | 负责实现查看成本收益表界面所需要的服务 |
| ConstantsController | 负责实现常量信息制定界面所需要的服务 |
| OrganizationController | 负责实现机构管理界面所需要的服务 |
| StaffController | 负责实现人员管理界面所需要的服务 |
| SalaryController | 负责实现薪水策略制定界面所需要的服务 |
| SearchAccoutController | 负责实现账户查询界面所需要的服务 |
| StatisticsController | 负责实现查看经营情况界面所需要的服务 |
| ExamController | 负责实现审批单据界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的姓名和密码，可以解决登录问题 |
| Balanceview | 查看成本收益表的领域模型对象，拥有一次查看成本收益表所需要的总收入、总支出、总利润等信息，可以帮助完成查看成本收益表界面所需要的服务 |
| Constants | 常量信息制定的领域模型对象，拥有一次查看常量信息制定所需要的常量名称、数值等信息，可以帮助完成常量信息制定界面所需要的服务 |
| Organization | 机构管理的领域模型对象，拥有一次机构管理所需要的机构名称、类型等信息，可以帮助完成机构管理界面所需要的服务 |
| Staff | 人员管理的领域模型对象，拥有一次人员管理所需要的人员名称、类型等信息，可以帮助完成人员管理界面所需要的服务 |
| Salary | 薪水策略制定的领域模型对象，拥有一次薪水策略制定所需要的薪水支付方式、职业等信息，可以帮助完成薪水策略制定界面所需要的服务 |
| SearchAccount | 账户查询的领域模型对象，拥有一次账户查询所需要的账户名称、密码等信息，可以帮助完成账户查询界面所需要的服务 |
| Statistics | 查看经营情况的领域模型对象，拥有一次经营情况查看所需要的付款单、收款单等信息，可以帮助完成查看经营情况界面所需要的服务 |
| Exam | 审批单据的领域模型对象，拥有一次审批单据所需要的单据状态、单据类型等信息，可以帮助完成审批单据界面所需要的服务 |

(3)模块内部类的接口规范

BalanceviewController和Balanceview的接口规范如下表所示。

**BalanceviewController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| BalanceviewController.getBalanceview | 语法 | public BalanceviewVO getBalanceview() | |
| 前置条件 | 已创建一个Balanceview领域对象 | |
| 后置条件 | 调用Manager领域对象的getBalanceview方法 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 | |
| Balanceview.getBalanceview() | | | 返回一个balanceviewVO对象 | |

**Balanceview的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | |
| Balanceview. getBalanceview | 语法 | public BalanceviewVO getBalanceview() | | |
| 前置条件 | 启动一次成本收益表查看 | | |
| 后置条件 | 返回一个balanceviewVO对象 | | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | 服务 | | |
| DatabaseFactory.getBalanceDa  Ta | | | | 得到Balance数据库的服务和引用 |
| BalanceDataService.allin() | | | | 获取至今为止的总收入 |
| BalanceDataService.allout() | | | | 获取至今为止的总支出 |

ConstantsController和Constants的接口规范如下表所示。

**ConstantsController的接口规范**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | | |
| ConstantsController.addConstants | 语法 | public ResultMessage addConstants (ConstantsVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的addContants方法 | | | |
| ConstantsController.delConstants | 语法 | public ResultMessage delConstants(ConstantsVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的delContants方法 | | | |
| ConstantsController.revConstants | 语法 | public ResultMessage revConstants (ConstantsVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Constants领域对象的revContants方法 | | | |
| ConstantsController.getConstants | 语法 | public ConstantsVO getConstants (string name) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getContants方法 | | | |
| ConstantsController.addCity | 语法 | **public** ResultMessage addCity(CityVO vo); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的addCity方法 | | | |
| ConstantsController.delCity | 语法 | **public** ResultMessage delCity(CityVO vo); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的delCity方法 | | | |
| ConstantsController.revCity | 语法 | **public** ResultMessage revCity(CityVO vo); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的revCity方法 | | | |
| ConstantsController.getAllCity | 语法 | **public** List<CityVO> getAllCity(); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getAllCity方法 | | | |
| ConstantsController.findCity | 语法 | **public** CityVO findCity(String name) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的findCity方法 | | | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | | 服务 | |
| Constants.addConstants | | | | 加入一个常量 | |
| Constants.delConstants | | | | 删除一个常量 | |
| Constants.revConstants | | | | 修改一个常量 | |
| Constants.getConstants | | | | 返回一个常量 | |

**Constants的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Constants.addConstants | 语法 | public ResultMessage addConstants (ConstantsVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入constant数据 |
| Constants.delConstants | 语法 | public ResultMessage delConstants(ConstantsVO vo) |
| 前置条件 | 数据库中存在相应的constant数据 |
| 后置条件 | 从数据库中删除相应的constant数据 |
| Constants.revConstants | 语法 | public ResultMessage revConstants (ConstantsVO vo) |
| 前置条件 | 数据库中存在相应的constant数据 |
| 后置条件 | 从数据库中修改相应的constant数据 |
| Constants.getConstants | 语法 | public ConstantsVO getConstants (string name) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Constants.addCity | 语法 | **public** ResultMessage addCity(CityVO vo); |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入新的city数据 |
| Constants.delCity | 语法 | **public** ResultMessage delCity(CityVO vo); |
| 前置条件 | 数据库中已经存在相应的city数据 |
| 后置条件 | 删除数据库中的相应的city数据 |
| Constants.revCity | 语法 | **public** ResultMessage revCity(CityVO vo); |
| 前置条件 | 数据库中已经存在相应的city数据 |
| 后置条件 | 修改数据库中的相应的city数据 |
| Constants.getAllCity | 语法 | **public** List<CityVO> getAllCity(); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Constants.findCity | 语法 | **public** CityVO findCity(String name) |
| 前置条件 | 无 |
| 后置条件 | 无 |

|  |  |
| --- | --- |
| 需要的服务(需接口) | |
| 服务名 | 服务 | |
| DatabaseFactory.getConstantsDa  Ta | 得到Constants数据库的服务和引用 |
| ConstantsDataService.addConstants(ConstantsPO po) | 向数据库中插入ConstantsPO对象 | |
| ConstantsDataService.deleteConstants(ConstantsPO po) | 从数据库中删除ConstantsPO对象 | |
| ConstantsDataService.updateConstants(ConstantsPO po) | 从数据库中修改更新某个ConstantsPO对象 | |
| ConstantsDataService. findConstants(String name) | 从数据库中根据常量名返回ConstantsPO对象 | |
| ConstantsDataService. fuzzyfindConstants(String name) | 从数据库中根据不完整的常量名返回ConstantsPO对象 | |
| ConstantsDataService. getLastId() | 获取ConstantsPO的最新ID | |

OrganizationController和Organization的接口规范如下表所示。

**OrganizationController的接口规范**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | | |
| OrganizationController.addOrganization | 语法 | public ResultMessage addOrganization (OrganizationVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的addOrganization方法 | | | |
| OrganizationController.delOrganization | 语法 | public ResultMessage delOrganization(OrganizationVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的delOrganization方法 | | | |
| OrganizationController.revOrganization | 语法 | public ResultMessage revOrganization (OrganizationVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的revOrganization方法 | | | |
| OrganizationController.getOrganizationbyCity | 语法 | **public** List<OrganizationVO> getOrganizationbyCity(String city); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getOrganizationbyCity方法 | | | |
| OrganizationController.getOrganizationbyType | 语法 | public List<OrganizationVO> getOrganizationbyType(Organizationtype type); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getOrganizationbyType方法 | | | |
| OrganizationController.getOrganizationbyBoth | 语法 | **public** List<OrganizationVO> getOrganizationbyBoth(String city, Organizationtype type); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getOrganizationbyBoth方法 | | | |
| OrganizationController.getOrganizationAll | 语法 | **public** List<OrganizationVO> getOrganizationAll(); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getOrganizationAllh方法 | | | |
| OrganizationController.getOrganizationCode | 语法 | **public** String getOrganizationCode(String city, Organizationtype type); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getOrganizationCode方法 | | | |
| OrganizationController.findOrgByCode | 语法 | **public** OrganizationVO findOrgByCode(String orgcode); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的findOrgByCode方法 | | | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | | 服务 | |
| Organization.addOrganization | | | | 加入一个机构 | |
| Organization.delOrganization | | | | 删除一个机构 | |
| Organization.revOrganization | | | | 修改一个机构 | |
| Organization.getOrganization | | | | 返回一个机构 | |

**Organization的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Organization.addOrganization | 语法 | public ResultMessage addOrganization (OrganizationVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入新的organization数据 |
| Organization.delOrganization | 语法 | public ResultMessage delOrganization(OrganizationVO vo) |
| 前置条件 | 数据库中有相应的organization数据 |
| 后置条件 | 删除数据库中相应的organization数据 |
| Organization.revOrganization | 语法 | public ResultMessage revOrganization (OrganizationVO vo) |
| 前置条件 | 数据库中有相应的organization数据 |
| 后置条件 | 修改数据库中相应的organization数据 |
| Organization.getOrganizationbyCity | 语法 | **public** List<OrganizationVO> getOrganizationbyCity(String city); |
| 前置条件 | 数据库中存有相应的city信息 |
| 后置条件 | 无 |
| Organization.getOrganizationbyType | 语法 | public List<OrganizationVO> getOrganizationbyType(Organizationtype type); |
| 前置条件 | 数据库中存在相应的组织类型的数据 |
| 后置条件 | 无 |
| Organization.getOrganizationbyBoth | 语法 | **public** List<OrganizationVO> getOrganizationbyBoth(String city, Organizationtype type); |
| 前置条件 | 数据库中存有相应的city信息和organizationtype数据 |
| 后置条件 | 无 |
| Organization.getOrganizationAll | 语法 | **public** List<OrganizationVO> getOrganizationAll(); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Organization.getOrganizationCode | 语法 | **public** String getOrganizationCode(String city, Organizationtype type); |
| 前置条件 | 数据库中存有相应的city信息和organizationtype数据 |
| 后置条件 | 无 |
| Organization.findOrgByCode | 语法 | **public** OrganizationVO findOrgByCode(String orgcode); |
| 前置条件 | 无 |
| 后置条件 | 无 |

|  |  |
| --- | --- |
| 需要的服务(需接口) | |
| 服务名 | 服务 | |
| DatabaseFactory.getOrganizationDa  Ta | 得到Organization数据库的服务和引用 |
| OrganizationDataService.addOrganization(OrganizationPO po) | 向数据库中插入OrganizationPO对象 | |
| OrganizationDataService.deleteOrganization(OrganizationPO po) | 从数据库中删除OrganizationPO对象 | |
| OrganizationDataService.updateOrganization(OrganizationPO po) | 从数据库中修改更新某个OrganizationPO对象 | |
| OrganizationDataService.getAll() | 从数据库中获取所有OrganizationPO | |
| OrganizationDataService. fuzzyfind(String city) | 从数据库中根据城市名获取OrganizationPO列表 | |
| OrganizationDataService. fuzzyfindbytype(Organizationtype type) | 从数据库中根据机构类型获取OrganizationPO列表 | |
| OrganizationDataService. fuzzyfindbyboth(String city,Organizationtype type) | 从数据库中根据城市名和机构类型获取OrganizationPO列表 | |
| OrganizationDataService . getlasthallcode(String city) | 从数据库中根据城市名获取营业厅的最新编号 | |

StaffController和Staff的接口规范如下表所示。

**StaffController的接口规范**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | | |
| StaffController.addStaff | 语法 | public ResultMessage addStaff (StaffVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的addStaff方法 | | | |
| StaffController.delStaff | 语法 | public ResultMessage delStaff(StaffVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的delStaff方法 | | | |
| StaffController.revStaff | 语法 | public ResultMessage revStaff (StaffVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的revStaff方法 | | | |
| StaffController.getStaffAll | 语法 | **public** List<StaffVO> getStaffAll(); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getStaffAll方法 | | | |
| StaffController.getStaffbyOrganization | 语法 | **public** List<StaffVO> getStaffbyOrganization(String organization); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getStaffbyOrganization方法 | | | |
| StaffController.getStaffbyJob | 语法 | **public** List<StaffVO> getStaffbyJob(Job job); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getStaffbyJob方法 | | | |
| StaffController.getStaffbyBoth | 语法 | **public** List<StaffVO> getStaffbyBoth(String organization, Job job); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getStaffbyBoth方法 | | | |
| StaffController.getid | 语法 | **public** **long** getid(); | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getid方法 | | | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | | 服务 | |
| Staff.addStaff | | | | 加入一个人员 | |
| Staff.delStaff | | | | 删除一个人员 | |
| Staff.revStaff | | | | 修改一个人员 | |
| Staff.getStaff | | | | 返回一个人员 | |

**Staff的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Staff.addStaff | 语法 | public ResultMessage addStaff (StaffVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入staff数据 |
| Staff.delStaff | 语法 | public ResultMessage delStaff(StaffVO vo) |
| 前置条件 | 数据库中存在相应的staff数据 |
| 后置条件 | 在数据库中删除相应的staff数据 |
| Staff.revStaff | 语法 | public ResultMessage revStaff (StaffVO vo) |
| 前置条件 | 数据库中存在相应的staff数据 |
| 后置条件 | 在数据库中修改相应的staff数据 |
| Staff.getStaffAll | 语法 | **public** List<StaffVO> getStaffAll(); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Staff.getStaffbyOrganization | 语法 | **public** List<StaffVO> getStaffbyOrganization(String organization); |
| 前置条件 | 数据库中存有organization数据 |
| 后置条件 | 无 |
| Staff.getStaffbyJob | 语法 | **public** List<StaffVO> getStaffbyJob(Job job); |
| 前置条件 | 数据库中存有job数据 |
| 后置条件 | 无 |
| Staff.getStaffbyBoth | 语法 | **public** List<StaffVO> getStaffbyBoth(String organization, Job job); |
| 前置条件 | 数据库中存有organization数据和job数据 |
| 后置条件 | 无 |
| Staff.getid | 语法 | **public** **long** getid(); |
| 前置条件 | 数据库中存有staff对应的id |
| 后置条件 | 无 |

|  |  |
| --- | --- |
| 需要的服务(需接口) | |
| 服务名 | 服务 | |
| DatabaseFactory.getStaffDa  Ta | 得到Staff数据库的服务和引用 |
| StaffDataService.addStaff(StaffPO po) | 向数据库中插入StaffPO对象 | |
| StaffDataService.deleteStaff(StaffPO po) | 从数据库中删除StaffPO对象 | |
| StaffDataService.updateStaff(StaffPO po) | 从数据库中修改更新某个StaffPO对象 | |
| StaffDataService. findStaff(long id) | 从数据库中根据id返回StaffPO对象 | |
| StaffDataService. fuzzyfindbyorganization(String organizationame) | 从数据库中根据机构名称返回StaffPO列表 | |
| StaffDataService. fuzzyfindbyjob(Job job) | 从数据库中根据职业返回StaffPO列表 | |
| StaffDataService. fuzzyfindbyboth(String organizationame,Job job) | 从数据库中根据机构名称和职业返回StaffPO列表 | |
| StaffDataService. getAll() | 从数据库中返回所有StaffPO | |
| StaffDataService. getLastId() | 获取StaffPO的最新ID | |

SalaryController和Salary的接口规范如下表所示。

**SalaryController的接口规范**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | | |
| SalaryController.addSalary | 语法 | public ResultMessage addSalary (SalaryVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的addSalary方法 | | | |
| SalaryController.delSalary | 语法 | public ResultMessage delSalary(SalaryVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的delSalary方法 | | | |
| SalaryController.revSalary | 语法 | public ResultMessage revSalary (SalaryVO vo) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的revSalary方法 | | | |
| SalaryController.getSalary | 语法 | public SalaryVO getSalary (Job job) | | | |
| 前置条件 | 已创建一个Manager领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用Manager领域对象的getSalary方法 | | | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | | 服务 | |
| Salary.addSalary | | | | 加入一个薪水策略 | |
| Salary.delSalary | | | | 删除一个薪水策略 | |
| Salary.revSalary | | | | 修改一个薪水策略 | |
| Salary.getSalary | | | | 返回一个薪水策略 | |

**Salary的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Salary.addSalary | 语法 | public ResultMessage addSalary (SalaryVO vo) |
| 前置条件 | 无 |
| 后置条件 | 向数据库中加入salary数据 |
| Salary.delSalary | 语法 | public ResultMessage delSalary(SalaryVO vo) |
| 前置条件 | 数据库中存在相应的salary数据 |
| 后置条件 | 从数据库中删除相应的salary数据 |
| Salary.revSalary | 语法 | public ResultMessage revSalary (SalaryVO vo) |
| 前置条件 | 数据库中存在相应的salary数据 |
| 后置条件 | 从数据库中修改相应的salary数据 |
| Salary.getSalary | 语法 | public SalaryVO getSalary (Job job) |
| 前置条件 | 数据库中存有job数据 |
| 后置条件 | 无 |

|  |  |
| --- | --- |
| 需要的服务(需接口) | |
| 服务名 | 服务 | |
| DatabaseFactory.getSalaryDa  Ta | 得到Salary数据库的服务和引用 |
| SalaryDataService.addSalary(SalaryPO po) | 向数据库中插入SalaryPO对象 | |
| SalaryDataService.deleteSalary(SalaryPO po) | 从数据库中删除SalaryPO对象 | |
| SalaryDataService.updateSalary(SalaryPO po) | 从数据库中修改更新某个SalaryPO对象 | |
| SalaryDataService.getSalary(SalaryPO po) | 从数据库中返回SalaryPO对象 | |

SearchAccountController和SearchAccount的接口规范如下表所示。

**SearchAccountController的接口规范**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | | |
| SearchAccountController. searchAccount | 语法 | public AccountVO searchAccount(String id) | | | |
| 前置条件 | 已创建一个SearchAccount领域对象，输入信息符合规范 | | | |
| 后置条件 | 调用SearchAccount领域对象的getSearchAccount方法 | | | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | | 服务 | |
| SearchAccount.getSearchAccount | | | | 返回一个薪水策略 | |

**SearchAccount的接口规范**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | | |
| SearchAccount. searchAccount | 语法 | public AccountVO searchAccount(String id) | | | |
| 前置条件 | 无 | | | |
| 后置条件 | 无 | | | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | | 服务 | |
| DatabaseFactory.getSalaryDa  Ta | | | | 得到Salary数据库的服务和引用 |
| accountdataservice.findAccount (String accountname) | | | | 从数据库中返回AccountPO对象 | |

StatisticsController和Statistics的接口规范如下表所示。

**StatisticsController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| StatisticsController. Statistics | 语法 | public StatisticsVO Statistics(long startTime,long endTime) | |
| 前置条件 | 已创建一个Statistics领域对象，输入符合规范 | |
| 后置条件 | 调用Statistics领域对象的Statistics方法 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 | |
| Statistics. Statistics(long startTime,long endTime) | | | 返回一个statisticsVO对象 | |

**Statistics的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| Statistics. getStatistics | 语法 | public StatisticsVO Statistics(long startTime,long endTime) | |
| 前置条件 | 启动一次成本收益表查看 | |
| 后置条件 | 返回一个statisticsVO对象 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 | |
| DatabaseFactory.getStatisticsData | | | 得到Statistics数据库的服务和引用 | |
| StatisticsDataService.getallRecordcollectForm(long starttime,long endtime) | | | 从数据库中根据开始时间和结束时间得到所有RecordcollectPO对象 | |
| StatisticsDataService.getallRecordpayForm(long starttime,long endtime) | | | 从数据库中根据开始时间和结束时间得到所有RecordpayPO对象 | |

ExamController和Exam的接口规范如下表所示。

**ExamController的接口规范**

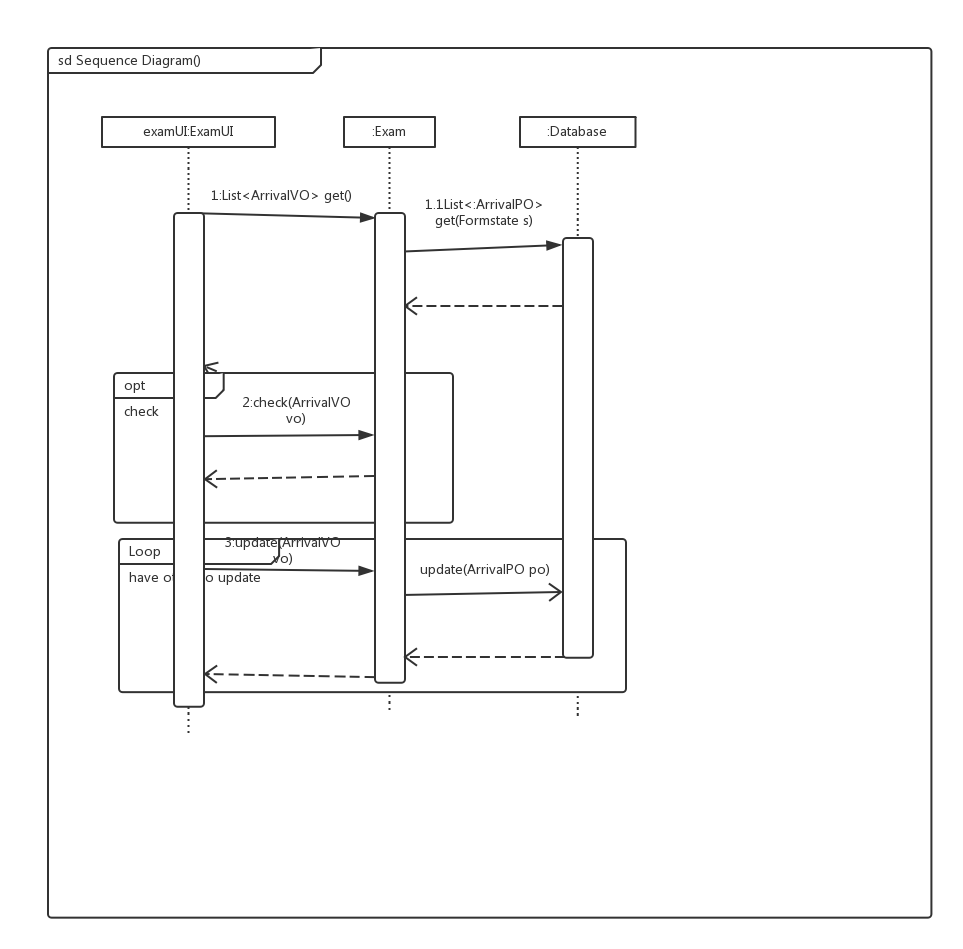
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | |
| Examcontroller.getOrders | 语法 | public List<OrderExamVO> getOrders() | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getOrders方法 | | |
| Examcontroller.OrdersExam | 语法 | public ResultMessage OrdersExam (List<OrderExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入符合规范 | | |
| 后置条件 | 调用Exam领域对象的OrdersExam方法 | | |
| Examcontroller.getHLForms | 语法 | public List<HLFormsExamVO> getHLForms () | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getHLForms方法 | | |
| Examcontroller. HLFormsExam | 语法 | public ResultMessage HLFormsExam (List< HLFormsExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入符合规范 | | |
| 后置条件 | 调用Exam领域对象的HLFormsExam方法 | | |
| Examcontroller.getCLForms | 语法 | public List<CLFormsExamVO> getCLForms () | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getCLForms方法 | | |
| Examcontroller. CLFormsExam | 语法 | public ResultMessage CLFormsExam (List< CLFormsExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的CLFormsExam方法 | | |
| Examcontroller.getGoodsRecevings | 语法 | public List<GoodsRecevingExamVO> getGoodsRecevings() | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getGoodsRecevings方法 | | |
| Examcontroller. GoodsRecevingsExam | 语法 | public ResultMessage GoodsRecevingsExam (List<GoodsRecevingExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的GoodsRecevingsExam方法 | | |
| Examcontroller.getDeliverys | 语法 | public List<DeliveryExamVO> getDeliverys() | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getDeliverys方法 | | |
| Examcontroller. DeliverysExam | 语法 | public ResultMessage DeliverysExam (List<DeliveryExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的DeliverysExam方法 | | |
| Examcontroller.getArrivals | 语法 | public List<ArrivalExamVO> getArrivals() | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getArrivals方法 | | |
| Examcontroller.ArrivalsExam | 语法 | public ResultMessage ArrivalsExam (List<ArrivalExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的ArrivalsExam方法 | | |
| Examcontroller.getRecordtrans | 语法 | public List<RecordtranExamVO> getRecordtrans() | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getRecordtrans方法 | | |
| Examcontroller.RecordtransExam | 语法 | public ResultMessage RecordtransExam (List<RecordtranExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的RecordtransExam方法 | | |
| Examcontroller.getInbounds | 语法 | public List<StockinExamVO> getInbounds() | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getInbounds方法 | | |
| Examcontroller.InboundsExam | 语法 | public ResultMessage InboundsExam (List<StockinExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的InboundsExam方法 | | |
| Examcontroller.getStockouts | 语法 | public List<StockoutExamVO> getStockouts() | | |
| 前置条件 | 已创建一个Exam领域对象， | | |
| 后置条件 | 调用Exam领域对象的getStockouts方法 | | |
| Examcontroller.StockoutsExam | 语法 | public ResultMessage StockoutsExam (List<StockoutExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的StockoutsExam方法 | | |
| Examcontroller.getRecordcollects | 语法 | public List<RecordcollectExamVO> getRecordcollects() | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getRecordcollects方法 | | |
| Examcontroller.RecordcollectsExam | 语法 | public ResultMessage RecordcollectsExam (List<RecordcollectExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的RecordcollectsExam方法 | | |
| Examcontroller.getDebitnotes | 语法 | public List<DebitnoteExamVO> getDebitnotes() | | |
| 前置条件 | 已创建一个Exam领域对象 | | |
| 后置条件 | 调用Exam领域对象的getDebitnotes方法 | | |
| Examcontroller.DebitnotesExam | 语法 | public ResultMessage DebitnotesExam (List<DebitnoteExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的DebitnotesExam方法 | | |
| Examcontroller.getRecordpays | 语法 | public List<RecordpayExamVO> getRecordpays() | | |
| 前置条件 | 数据库中存在收款单信息 | | |
| 后置条件 | 调用Exam领域对象的getRecordpays方法 | | |
| Examcontroller.RecordpaysExam | 语法 | public ResultMessage RecordpaysExam (List<RecordpayExamVO> volist) | | |
| 前置条件 | 已创建一个Exam领域对象，输入信息符合规范 | | |
| 后置条件 | 调用Exam领域对象的RecordpaysExam方法 | | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | | 服务 | |
| Exam.getOrders() | | | | 返回List<OrderExamVO> | |
| Exam.OrdersExam(List<OrderExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getHLForms() | | | | 返回List<HLFormsExamVO> | |
| Exam. HLFormsExam(List< HLFormsExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getCLForms() | | | | 返回ListCHLFormsExamVO> | |
| Exam. CLFormsExam(List< CLFormsExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getGoodsRecevings() | | | | 返回List< GoodsRecevingExamVO > | |
| Exam.GoodsRecevingsExam(List<GoodsRecevingExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getDeliverys() | | | | 返回List< DeliveryExamVO VO> | |
| Exam.DeliverysExam (List<DeliveryExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getArrivals() | | | | 返回List< ArrivalExamVO > | |
| Exam.ArrivalsExam(List<ArrivalExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getRecordtrans() | | | | 返回List< ArrivalExamVO > | |
| Exam.RecordtransExam(List<ArrivalExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getInbounds() | | | | 返回List< StockinExamVO > | |
| Exam.InboundsExam(List<StockinExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getStockouts() | | | | 返回List< StockoutExamVO > | |
| Exam.StockoutsExam(List<StockoutExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getRecordcollects() | | | | 返回List< RecordcollectExamVO > | |
| Exam.RecordcollectsExam(List<RecordcollectExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getDebitnotes() | | | | 返回List< DebitnoteExamVO > | |
| Exam.DebitnotesExam(List<DebitnoteExamVO> volist) | | | | 存储审批后的单据 | |
| Exam.getRecordpays() | | | | 返回List< RecordpayExamVO > | |
| Exam.RecordpaysExam(List<RecordpayExamVO> volist) | | | | 存储审批后的单据 | |

**Exam的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务(供接口) | | | | |
| Exam.getOrders | 语法 | public List<OrderExamVO> getOrders() | |
| 前置条件 | 开始订单查看 | |
| 后置条件 | 返回List<OrderExamVO> | |
| Exam.OrdersExam | 语法 | public ResultMessage OrdersExam (List<OrderExamVO> volist) | |
| 前置条件 | 开始订单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getHLForms | 语法 | public List<HLFormsExamVO> getHLForms () | |
| 前置条件 | 开始营业厅装车单查看 | |
| 后置条件 | 返回List< HLFormsExamVO > | |
| Exam. HLFormsExam | 语法 | public ResultMessage HLFormsExam (List< HLFormsExamVO> volist) | |
| 前置条件 | 开始营业厅装车单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getCLForms | 语法 | public List<CLFormsExamVO> getCLForms () | |
| 前置条件 | 开始中转中心装车单查看 | |
| 后置条件 | 返回List< CLFormsExamVO > | |
| Exam. CLFormsExam | 语法 | public ResultMessage CLFormsExam (List< CLFormsExamVO> volist) | |
| 前置条件 | 开始中转中心装车单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getGoodsRecevings | 语法 | public List<GoodsRecevingExamVO> getGoodsRecevings() | |
| 前置条件 | 开始收货单查看 | |
| 后置条件 | 返回List< GoodsRecevingExamVO > | |
| Exam. GoodsRecevingsExam | 语法 | public ResultMessage GoodsRecevingsExam (List<GoodsRecevingExamVO> volist) | |
| 前置条件 | 开始收货单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getDeliverys | 语法 | public List<DeliveryExamVO> getDeliverys() | |
| 前置条件 | 开始派件单查看 | |
| 后置条件 | 返回List< DeliveryExamVO > | |
| Exam. DeliverysExam | 语法 | public ResultMessage DeliverysExam (List<DeliveryExamVO> volist) | |
| 前置条件 | 开始派件单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getArrivals | 语法 | public List<ArrivalExamVO> getArrivals() | |
| 前置条件 | 开始到达单查看 | |
| 后置条件 | 返回List< ArrivalExamVO > | |
| Exam.ArrivalsExam | 语法 | public ResultMessage ArrivalsExam (List<ArrivalExamVO> volist) | |
| 前置条件 | 开始到达单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getRecordtrans | 语法 | public List<RecordtranExamVO> getRecordtrans() | |
| 前置条件 | 开始中转单查看 | |
| 后置条件 | 返回List< RecordtranExamVO > | |
| Exam.RecordtransExam | 语法 | public ResultMessage RecordtransExam (List<RecordtranExamVO> volist) | |
| 前置条件 | 开始中转单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getInbounds | 语法 | public List<StockinExamVO> getInbounds() | |
| 前置条件 | 开始入库单查看 | |
| 后置条件 | 返回List< StockinExamVO > | |
| Exam.InboundsExam | 语法 | public ResultMessage InboundsExam (List<StockinExamVO> volist) | |
| 前置条件 | 开始入库单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getStockouts | 语法 | public List<StockoutExamVO> getStockouts() | |
| 前置条件 | 开始出库单查看 | |
| 后置条件 | 返回List< StockoutExamVO > | |
| Exam.StockoutsExam | 语法 | public ResultMessage StockoutsExam (List<StockoutExamVO> volist) | |
| 前置条件 | 开始出库单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getRecordcollects | 语法 | public List<RecordcollectExamVO> getRecordcollects() | |
| 前置条件 | 开始收款单查看 | |
| 后置条件 | 返回List< RecordcollectExamVO > | |
| Exam.RecordcollectsExam | 语法 | public ResultMessage RecordcollectsExam (List<RecordcollectExamVO> volist) | |
| 前置条件 | 开始收款单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getDebitnotes | 语法 | public List<DebitnoteExamVO> getDebitnotes() | |
| 前置条件 | 开始收款单查看 | |
| 后置条件 | 返回List< DebitnoteExamVO > | |
| Exam.DebitnotesExam | 语法 | public ResultMessage DebitnotesExam (List<DebitnoteExamVO> volist) | |
| 前置条件 | 开始收款单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Exam.getRecordpays | 语法 | public List<RecordpayExamVO> getRecordpays() | |
| 前置条件 | 开始付款单查看 | |
| 后置条件 | 返回List< RecordpayExamVO > | |
| Exam.RecordpaysExam | 语法 | public ResultMessage RecordpaysExam (List<RecordpayExamVO> volist) | |
| 前置条件 | 开始付款单审批 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| 需要的服务(需接口) | | | | |
| 服务名 | | | | 服务 |

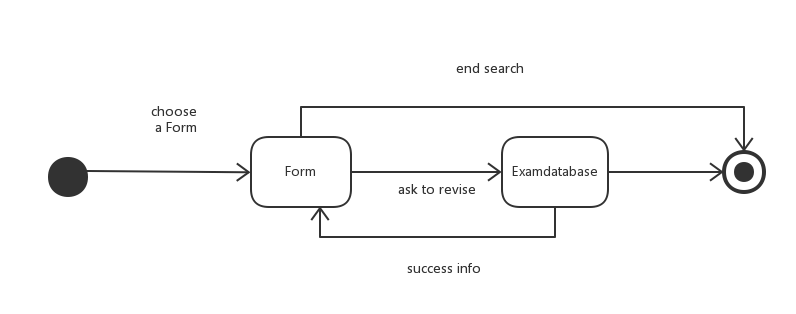
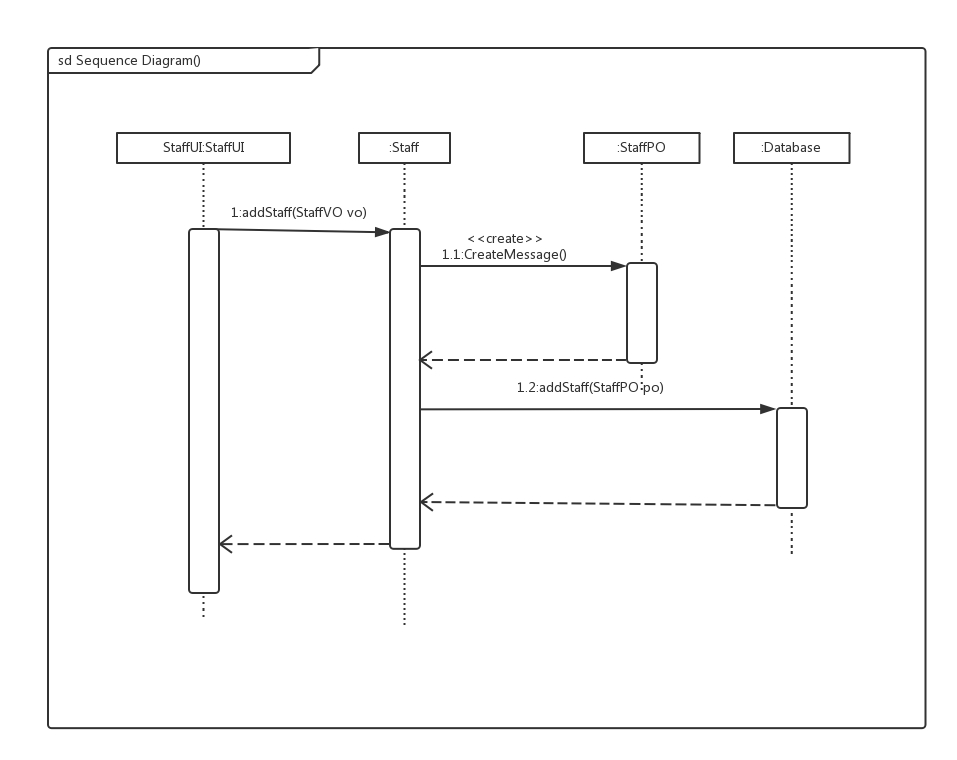
(4)业务逻辑层的动态模型

|  |  |
| --- | --- |
| DatabaseFactory.getExamDa  Ta | 得到Exam数据库的服务和引用 |
| Examdataservice.getOrders(Formstate state) | 返回List<OrderExamPO> | |
| Examdataservice.getHLForms(Formstate state) | 返回List<HLFormsExamPO> | |
| Examdataservice.getCLForms(Formstate state) | 返回ListCHLFormsExamPO> | |
| Examdataservice.getGoodsRecevings(Formstate state) | 返回List< GoodsRecevingExamPO > | |
| Examdataservice.getDeliverys(Formstate state) | 返回List< DeliveryExamPO PO> | |
| Examdataservice.getArrivals(Formstate state) | 返回List< ArrivalExamPO > | |
| Examdataservice.getRecordtrans(Formstate state) | 返回List< ArrivalExamPO > | |
| Examdataservice.getInbounds(Formstate state) | 返回List< StockinExamPO > | |
| Examdataservice.getStockouts(Formstate state) | 返回List< StockoutExamPO > | |
| Examdataservice.getRecordcollects(Formstate state) | 返回List< RecordcollectExamPO > | |
| Examdataservice.getRecordpays(Formstate state) | 返回List< RecordpayExamPO > | |
| Examdataservice. updateOrderForm (OrderPO po) | 存储审批后的OrderPO | |
| Examdataservice. updateArrivalForm (ArrivalPO po) | 存储审批后的ArrivalPO | |
| Examdataservice. updateCenterLoadForm (CenterloadPO po) | 存储审批后的CenterloadPO | |
| Examdataservice. updateDeliveryForm (DeliveryPO po) | 存储审批后的DeliveryPO | |
| Examdataservice. updateGoodsReceivingForm (GoodsReceivingPO po) | 存储审批后的GoodsReceivingPO | |
| Examdataservice. updateLoadForm(HallLoadPO po) | 存储审批后的HallLoadPO | |
| Examdataservice. updateRecordtransForm (RecordtransPO po) | 存储审批后的RecordtransPO | |
| Examdataservice. updateRecordcollectForm (RecordcollectPO po) | 存储审批后的RecordcollectPO | |
| Examdataservice. updateRecordpayForm (RecordpayPO po) | 存储审批后的RecordpayPO | |
| Examdataservice. updateStockinForm (StockinPO po) | 存储审批后的StockinPO | |
| Examdataservice. updateStockoutForm (StockoutPO po) | 存储审批后的StockoutPO | |

下图为总经理审批单据时的顺序图。

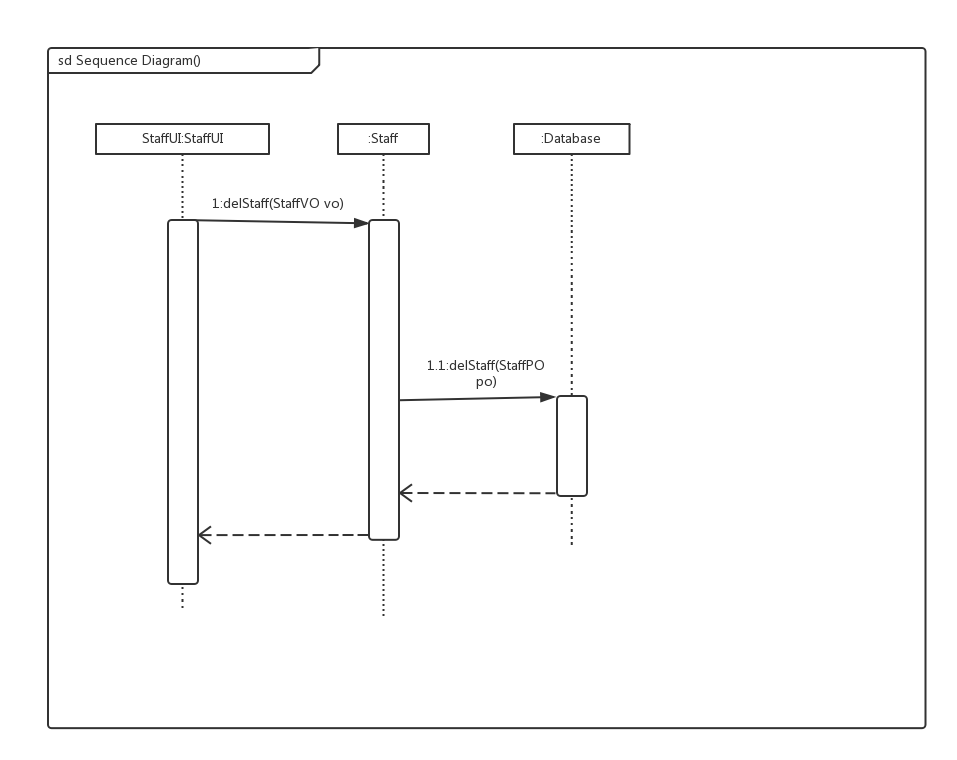
审批单据的顺序图

下图为总经理审批时的状态图



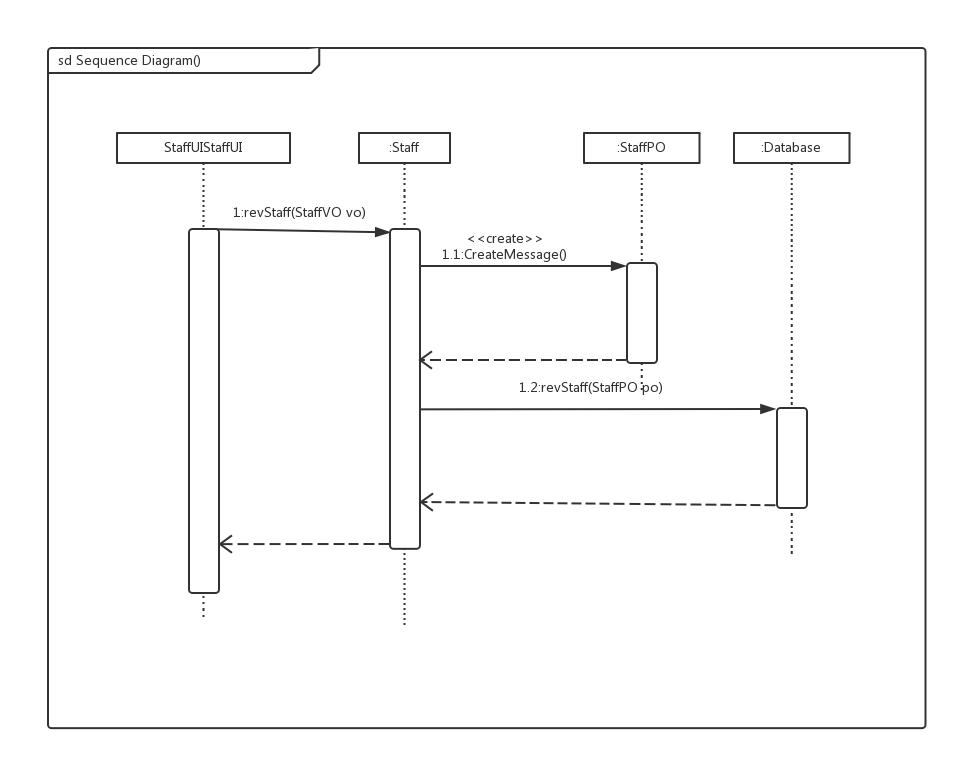
下图为总经理增加人员时的顺序图

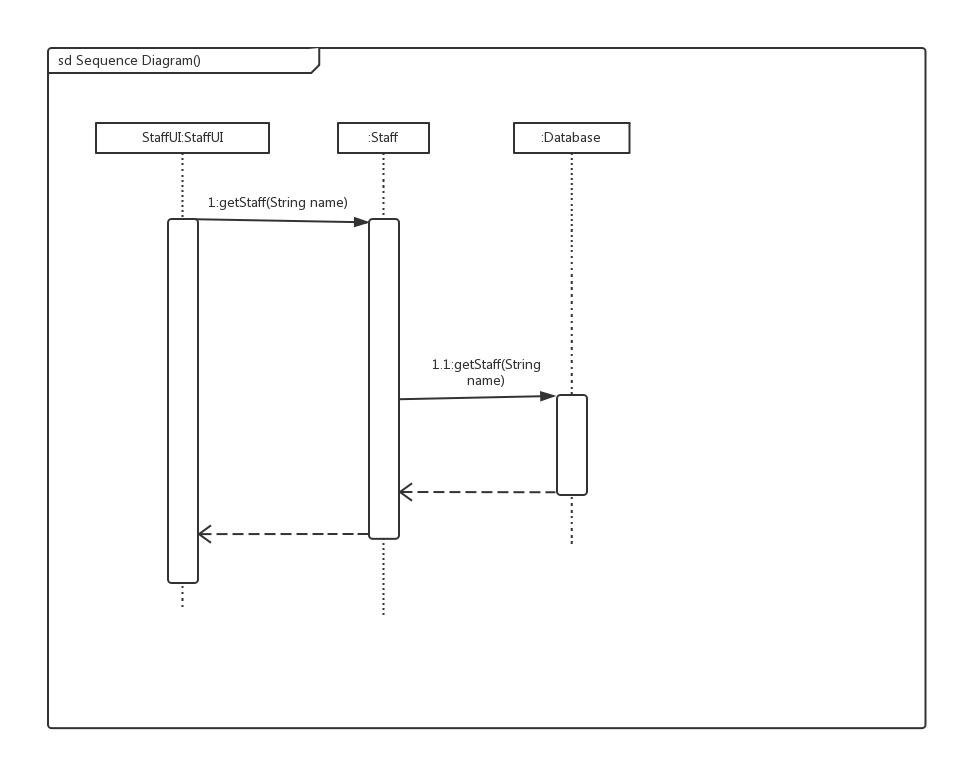
增加人员时的顺序图

下图为总经理删除人员时的顺序图

删除人员的顺序图

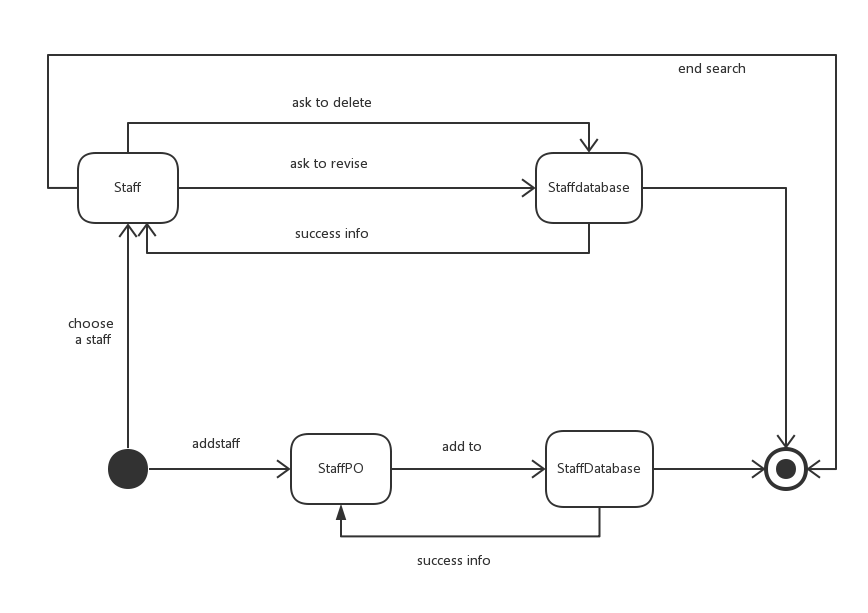
下图为总经理修改人员信息的顺序图

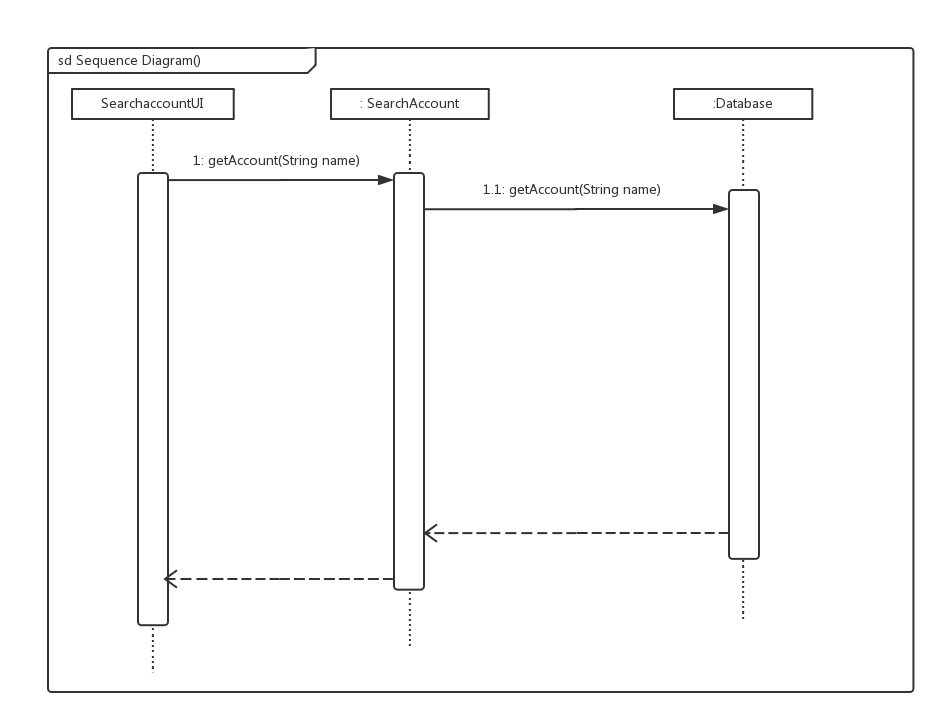
修改人员的顺序图

下图为总经理查询人员信息的顺序图

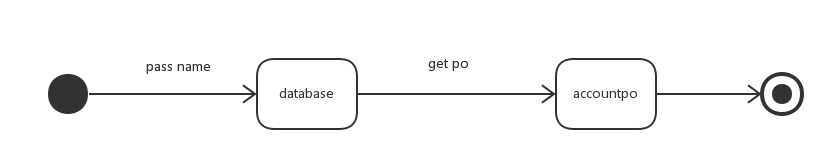
查询人员的顺序图

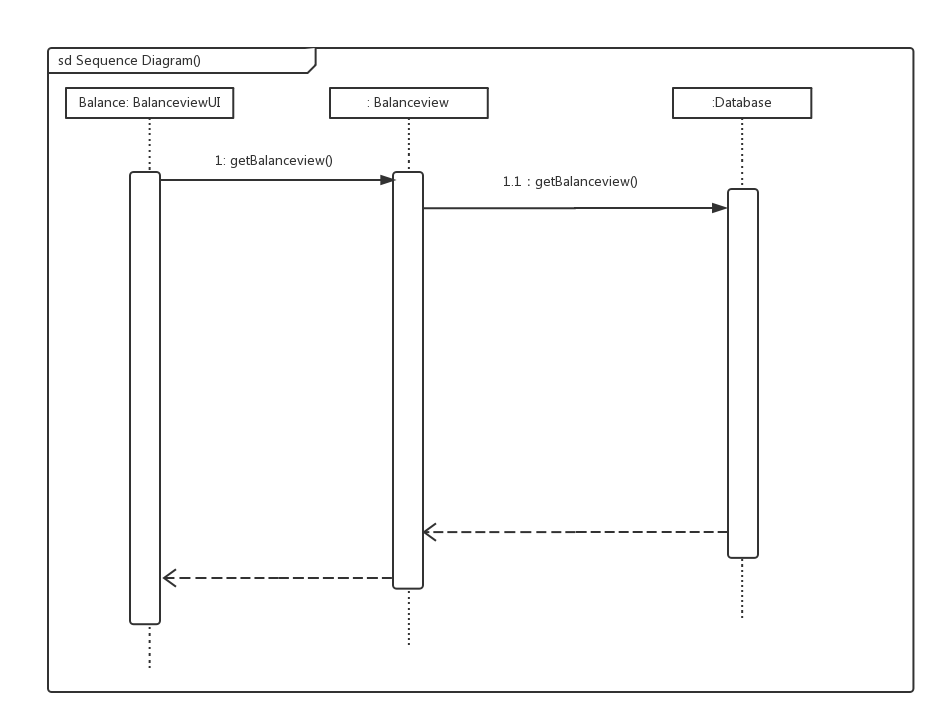
下图为总经理人员管理时的状态图

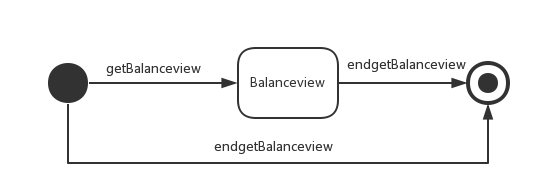
下图为总经理账户查询时的顺序图

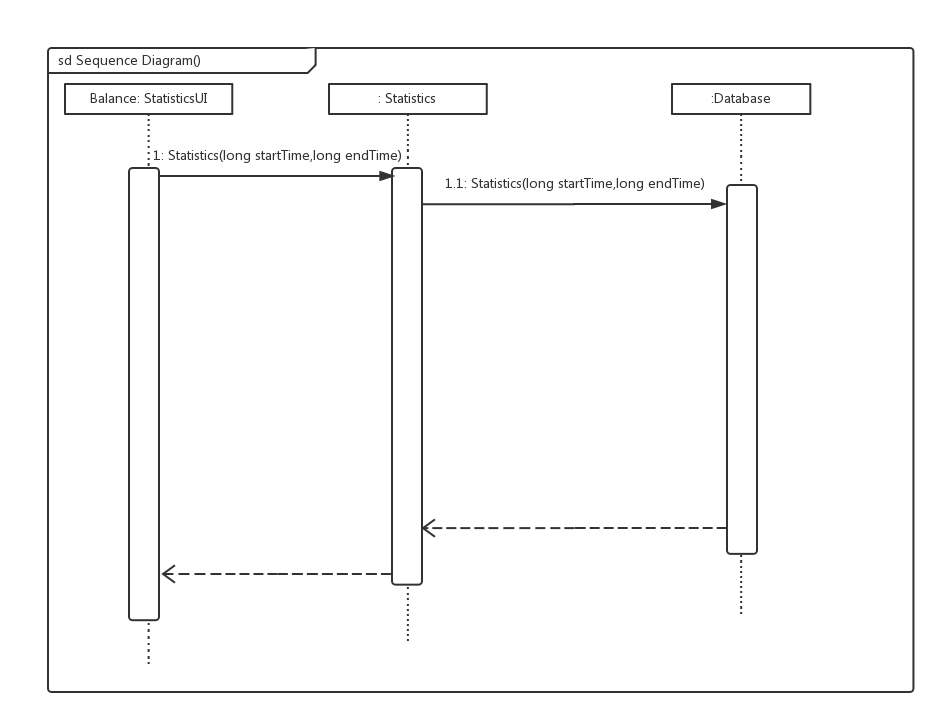


下图为总经理账户查询的状态图

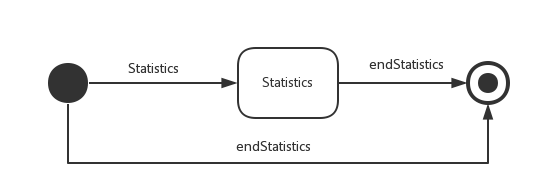
下图为总经理查看经营情况表时的顺序图

下图为总经理查看经营情况表时的状态图

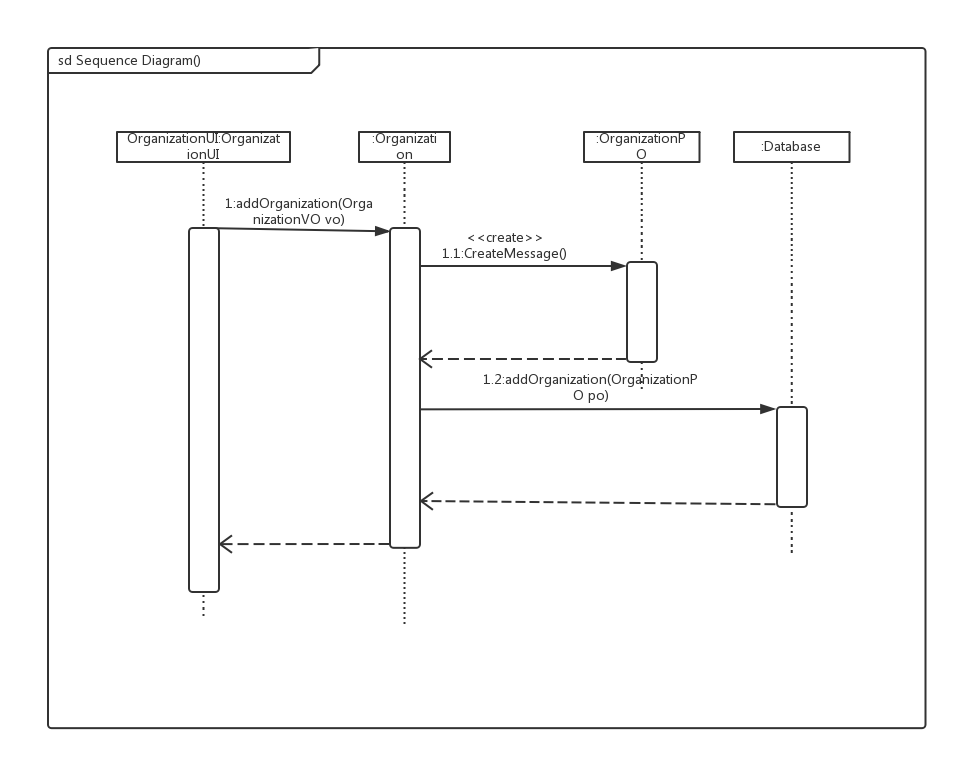
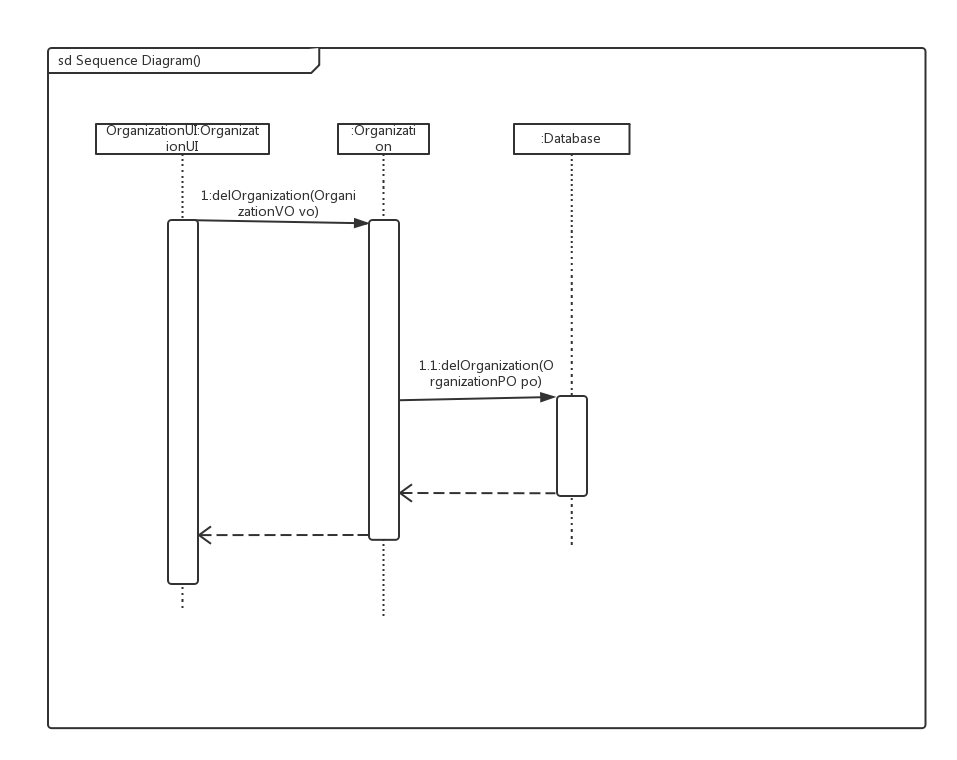
下图为总经理查看统计报表时的顺序图



下图为总经理查看统计报表时的状态图



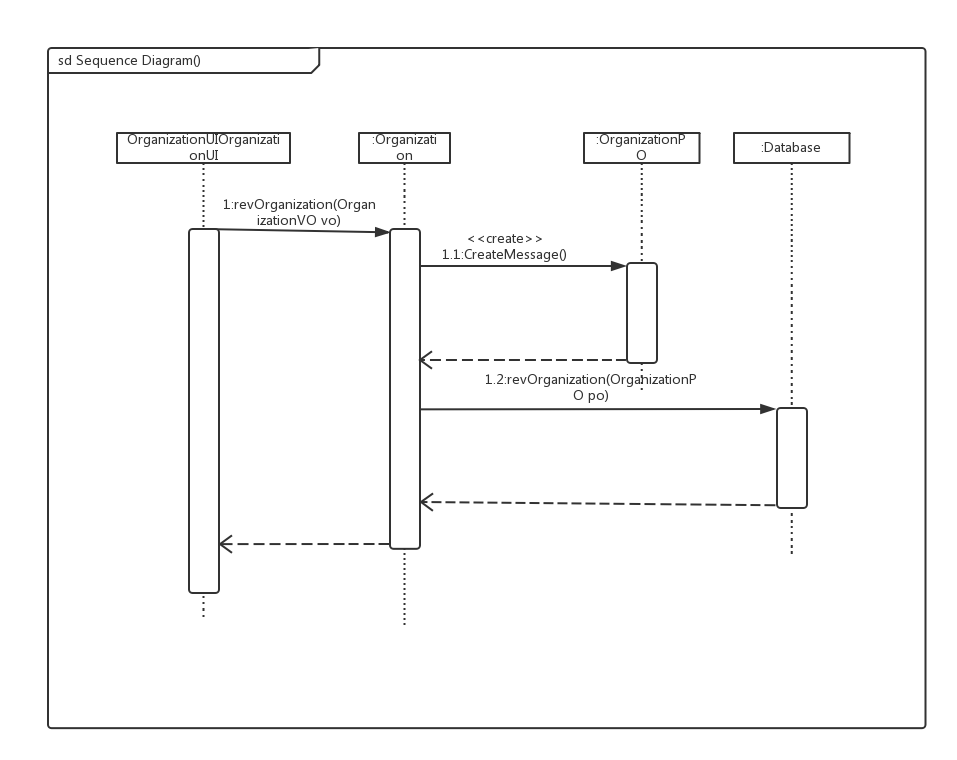
下图为总经理增加机构时的顺序图

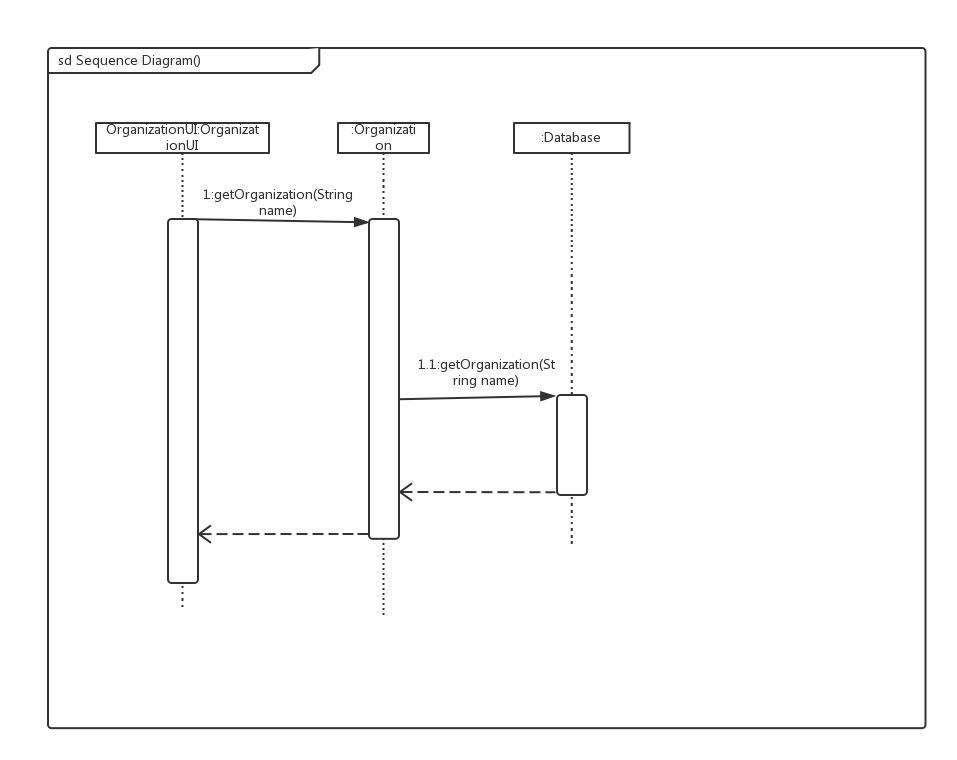
增加机构时的顺序图

下图为总经理删除机构时的顺序图

删除机构的顺序图

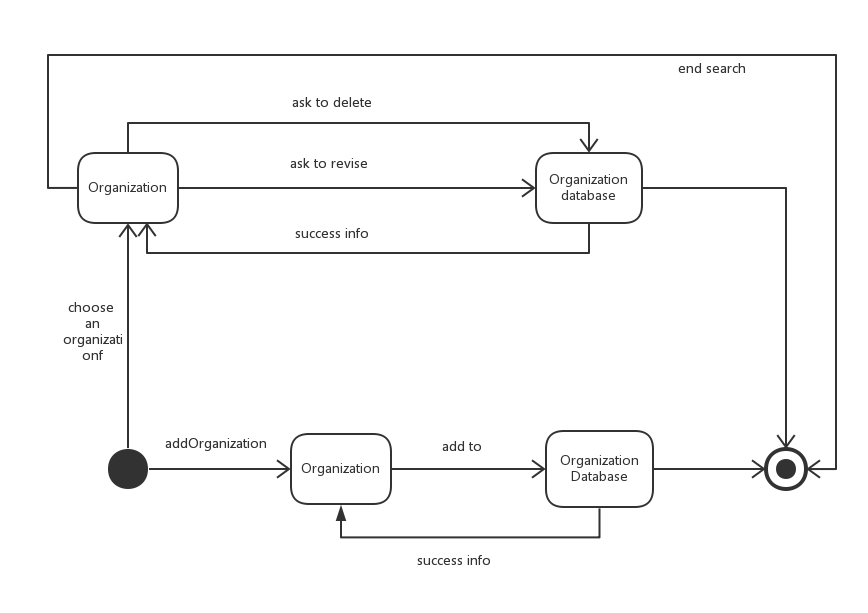
下图为总经理修改机构时的顺序图

修改机构时的顺序图

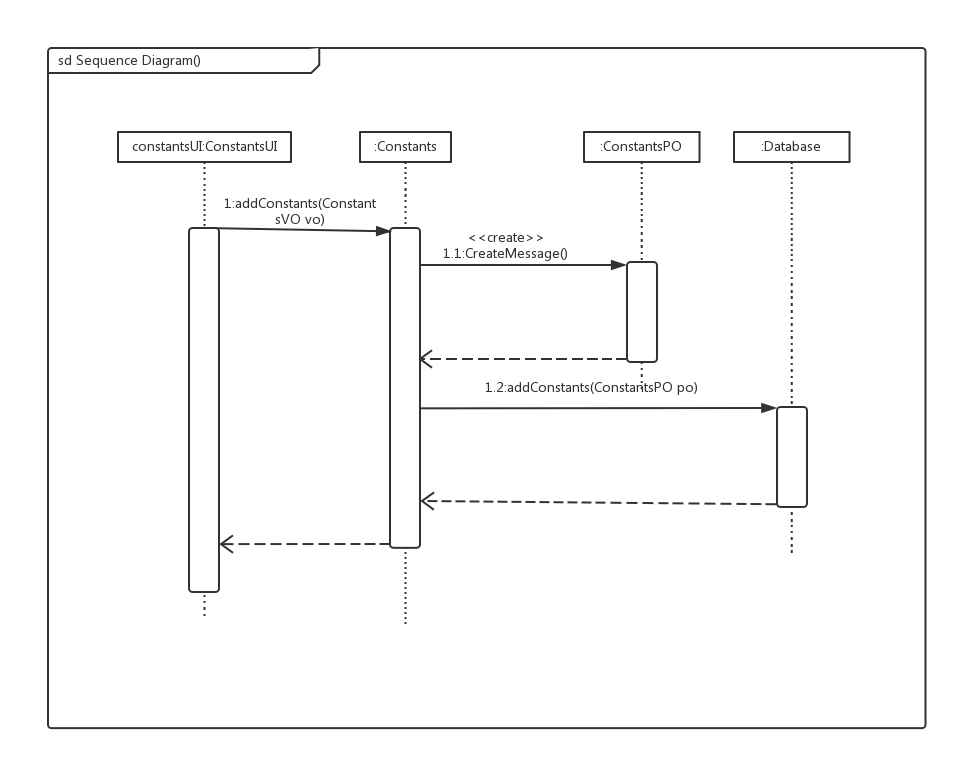
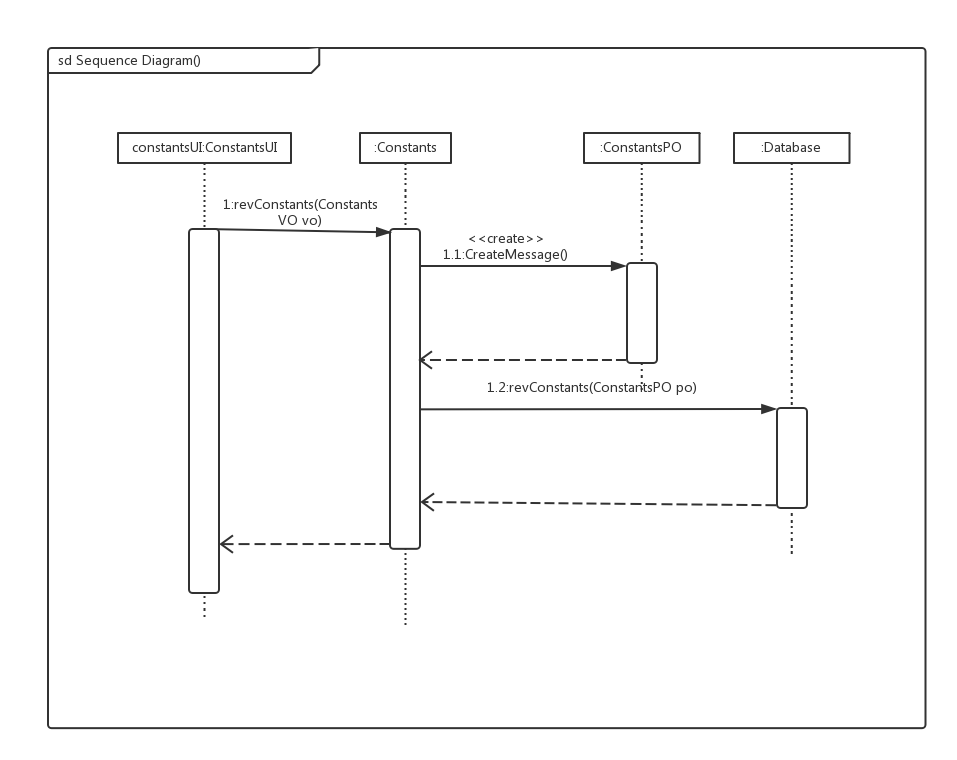
下图为总经理查询机构时的顺序图

查询机构时的顺序图

下图为总经理机构管理时的状态图



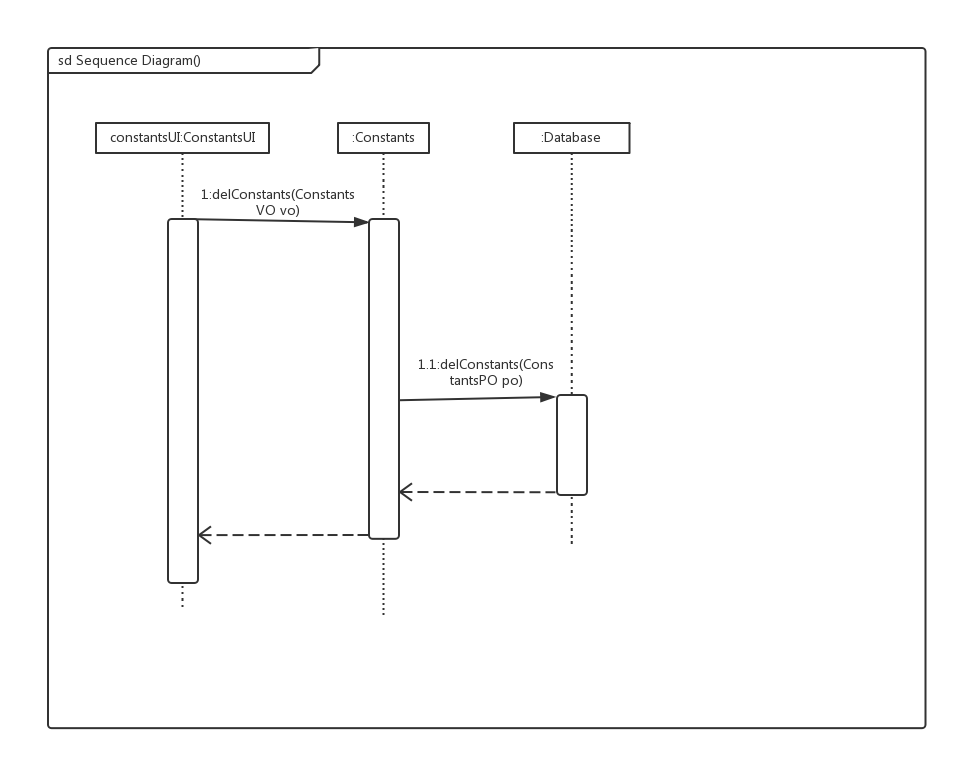
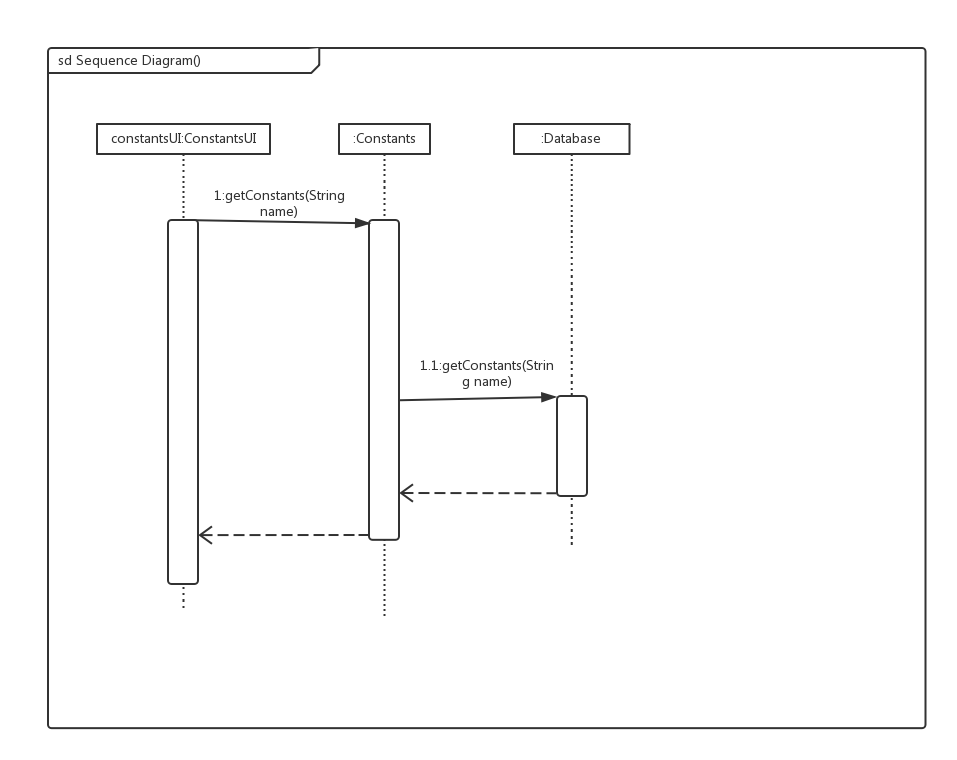
下图为总经理增加常量时的顺序图

下图增加常量的顺序图

下图为总经理修改常量时的顺序图

修改常量时的顺序图

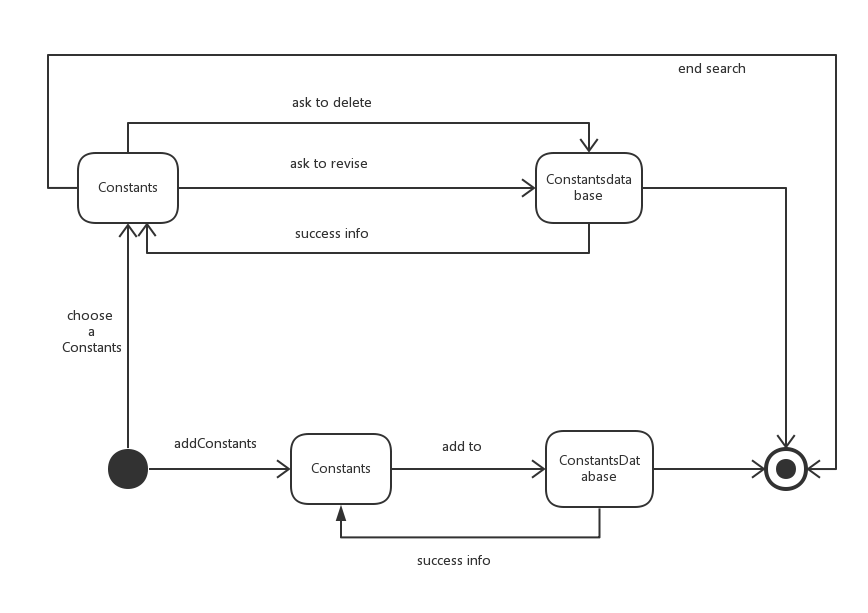
下图为删除常量时的顺序图

删除常量的顺序图

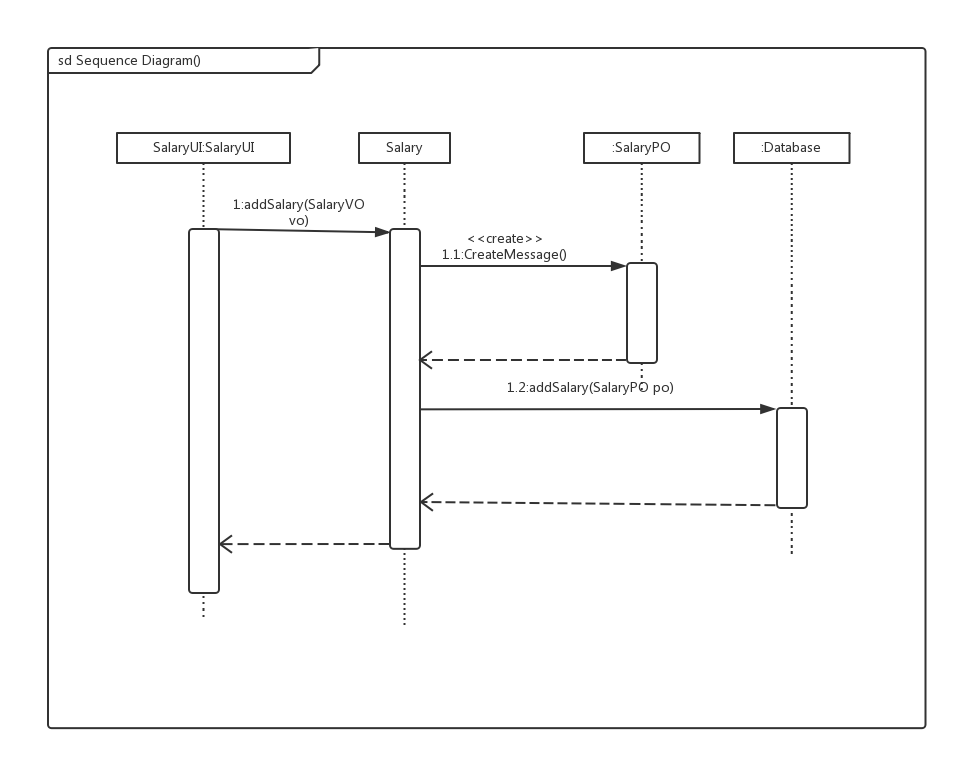
下图为总经理查询常量时的顺序图

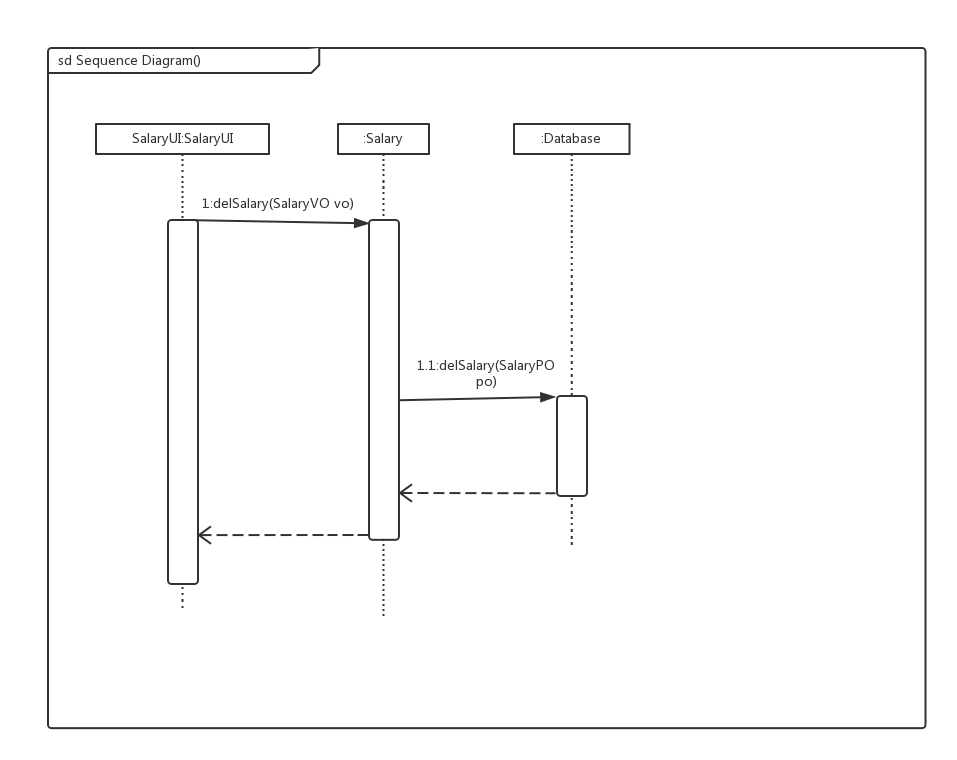
查询常量的顺序图

下图为总经理常量信息制定时的状态图



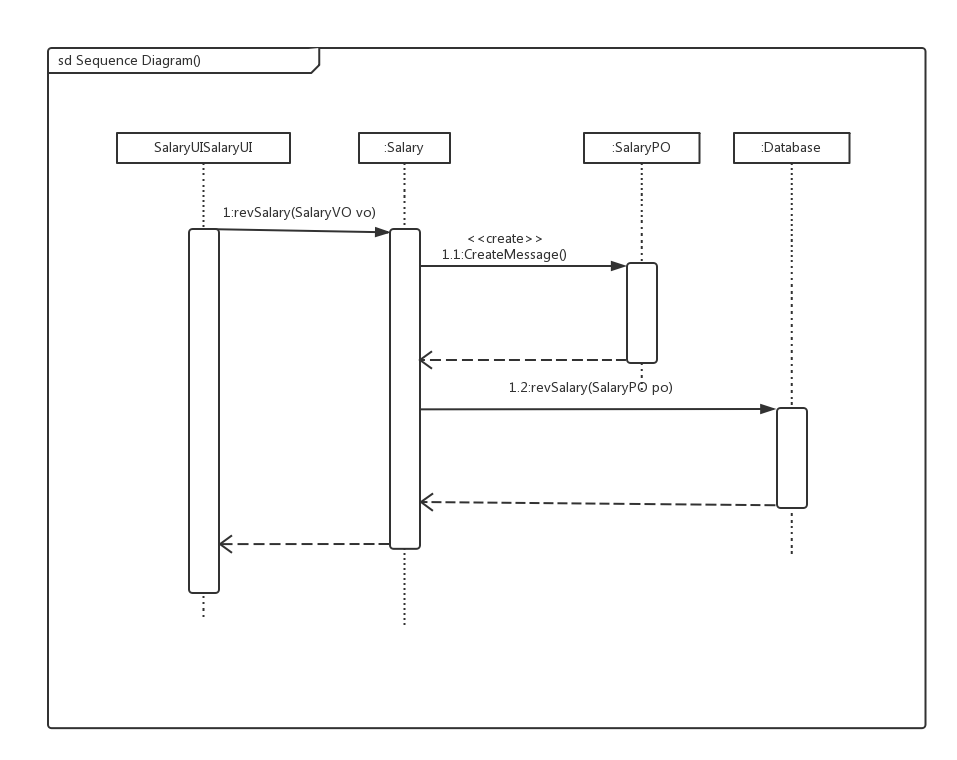
下图为总经理增加薪水策略的顺序图

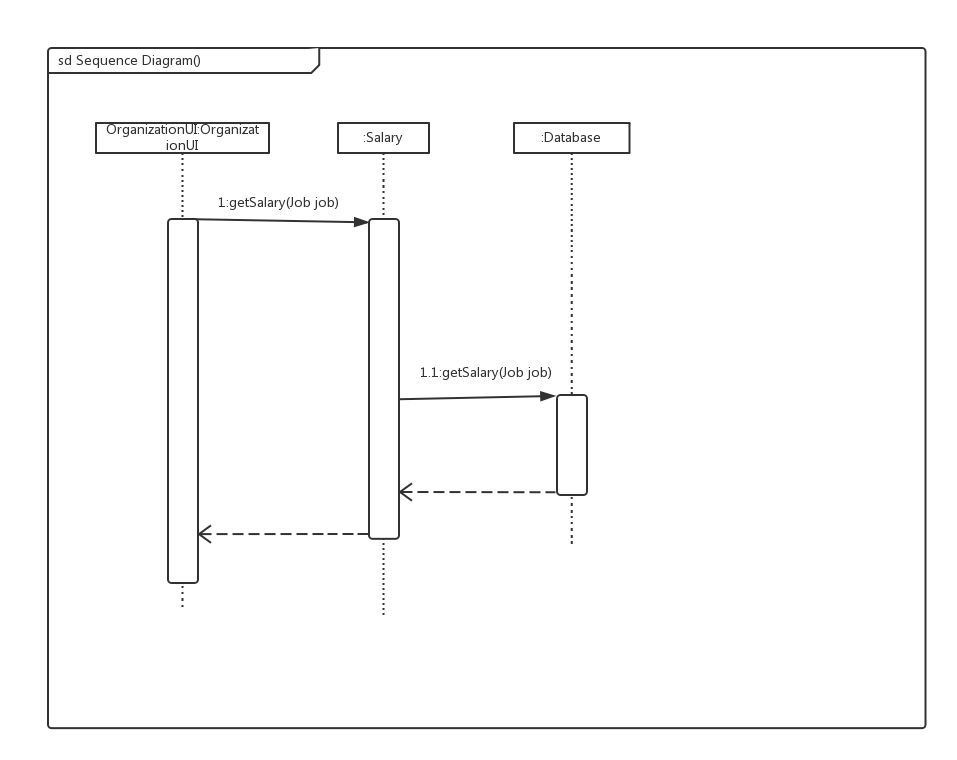
增加薪水策略的顺序图

下图为总经理删除薪水策略时的顺序图

删除薪水策略的顺序图

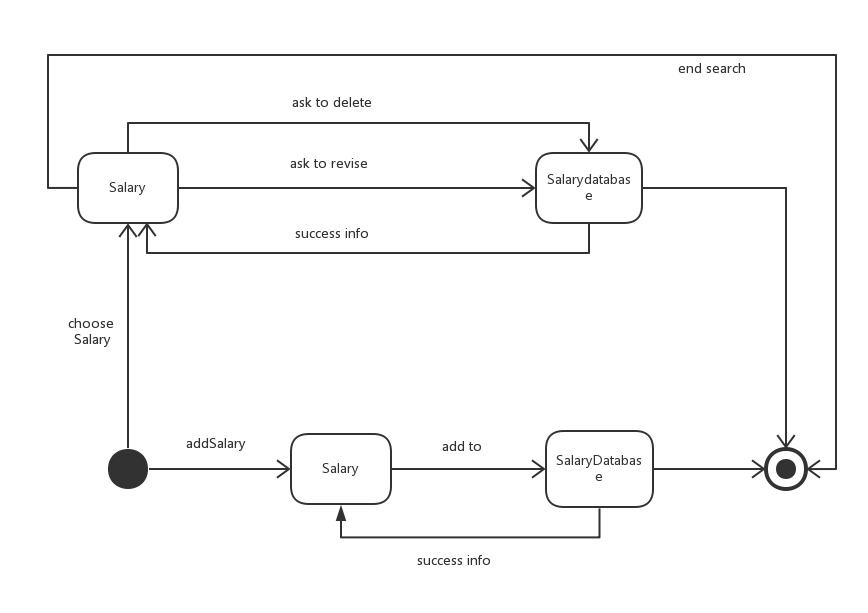
下图为总经理修改薪水策略的顺序图

修改薪水策略的顺序图

下图为总经理查询薪水策略的顺序图

查询薪水策略的顺序图

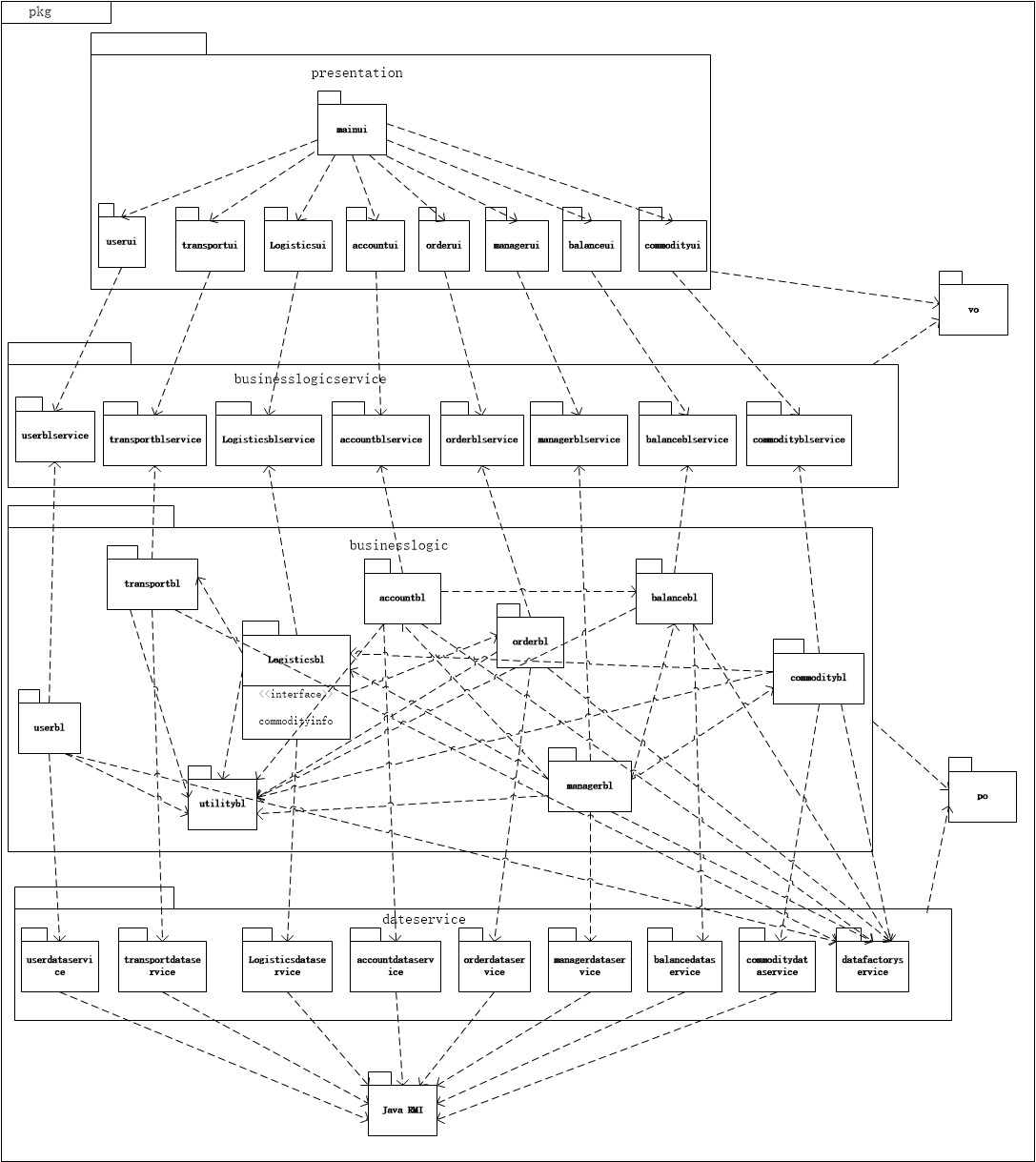
下图为总经理薪水策略制定时的状态图



(5)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。

5.依赖视角

下图是客户端各自包之间的依赖关系。

下图是服务器端各自包之间的依赖关系。

