**酒店预订系统**

**详细设计描述文档**

**Cdll小组**

**曹利航 段梦洋刘港欢 刘钦**

**2016-10-28**

**目录**目录有链接、还有导航的哟~

[更新历史](#30j0zll)·······················································································································································3

[1.引言](#1fob9te)····························································································································································4

[1.1编制目的](#3znysh7)················································································································································4

[1.2词汇表](#2et92p0)····················································································································································4

[1.3参考资料](#tyjcwt)················································································································································4

2.产品设计概述················································································································································4

3.体系结构设计概述········································································································································4

4.结构视角······························································································································5

5.依赖视角·····································································································································68

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 段梦洋 | 2016-10-27 | 最初草稿（创建文档模板） | V1.0 |
| 段梦洋 | 2016-10-28 | 添加customer部分 | V1.0.1 |
| 刘钦 | 2016-10-29 | 添加hotel部分 | V1.0.2 |
| 刘港欢 | 2016-10-29 | 添加order部分 | V1.0.3 |
| 曹利航 | 2016-10-29 | 添加SalePromotion和Manager部分 | V1.0.4 |
| 刘钦 | 2016-11-13 | 从hotel中分出room部分 | V1.0.5 |
| 段梦洋 | 2016-11-13 | 从customer中分出credit部分 | V1.0.6 |
| 刘港欢 | 2016-12-24 | 优化order逻辑部分——将不同类型订单的执行整合 | v1.0.7 |
| 刘钦 | 2016-12-29 | promotion逻辑部分重新设计 | v1.0.8 |

**1.引言**

**1.1编制目的**

本报告详细完成对酒店预订系统的概要设计，达到指导详细设计和开放的目的，同时实现和测试人员及用户的沟通

**1.2词汇表**

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

**1.3参考资料**

1. IEEE Std 1016-1998 《IEEE Recommended Practice for Software Design Descriptions》

2.cdll小组，《酒店预订系统需求规格说明书V1.2》

3.丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版社，2012：134—182

1. **产品概述**

参考酒店预订系统用例文档和酒店预订系统软件需求规格说明文档中对产品的概括描述。

**3.体系结构设计描述**

参考酒店预订系统体系结构描述文档中对体系结构设计的描述。

**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。

**4.1.1 customerbl模块**

**（1）模块概述**

customerbl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。

Customerbl模块的指责及接口参见软件体系结构描述文档表 。

**（2）整体结构**

根据体系结构的设计,我们将系统分为展现层、业务逻辑层、数据层。每一层之间为了增加灵活性,我们会添加接口。比如展示层和业务逻辑层之间,我们添加blservice.customerblservice.CustomerBLService接口。业务逻辑层和数据层之间添加 dataservice.customerdataservice.customerDataService接口。为了隔离业务逻辑职责和逻辑控制职责,我们增加了 CustomerController,这样CustomerController会将对审查的业务逻辑处理委托给Customer对象。CustomerPO,CreditPO是作为审查记录的持久化对象被添加到设计模型中去的。而 CalculateLevel封装了根据会员信用和等级制度信息计算会员等级的职责。

Customerbl模块的设计如图1所示。

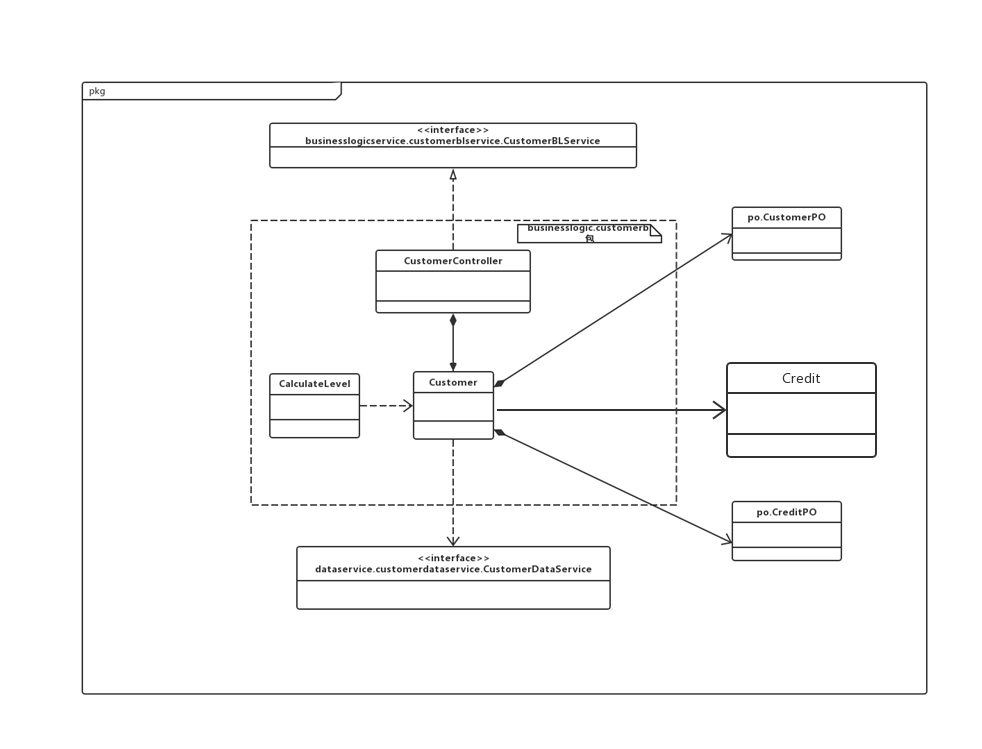


图1 customerbl模块各个类的设计

Customerbl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| CustomerController | 负责实现会员管理界面所需要的服务 |
| Customer | 会员管理的领域模型对象，拥有会员信息、信用值等信息，可以帮助完成会员管理界面所需要的服务 |

**（3）模块内部类的接口规范**

CustomerController和Customer的接口规范如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CustomerController.login | 语法 | public ResultMessage login(String userName, String password); | |
| 前置条件 | 输入符合规则 | |
| 后置条件 | 调用Customer领域对象的find方法 | |
| CustomerController.addCustomer | 语法 | public ResultMessage addCustomer(CustomerVO vo); | |
| 前置条件 | 输入符合规则 | |
| 后置条件 | 调用Customer领域对象的addCustomer方法 | |
| CustomerController.ChangeCustomerInfo | 语法 | public ResultMessage changeCustomerInfo(String ID,CustomerVo vo); | |
| 前置条件 | 已创建一个Customer领域对象，并且输入符合规则 | |
| 后置条件 | 调用Customer领域对象的changeCustomerInfo方法 | |
| CustomerController.getCustomerInfo | 语法 | public CustomerVO getCustomertInfo(String ID); | |
| 前置条件 | 已创建一个Customer领域对象 | |
| 后置条件 | 调用Customer领域对象的getCustomertInfo方法 | |
| CustomerController.successLog | 语法 | public CustomerVO successLog(String userName); | |
| 前置条件 | 用户成功登陆 | |
| 后置条件 | 返回用户信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Customer.login(String userName, String password) | | | 加入一个customer对象 |
| Customer.addCustomer(CustomerVO vo) | | | 插入一个Customer对象 |
| Customer.ChangeCustomerInfo(String userName,CustomerVo vo) | | | 更新Customer对象 |
| Customer.getCustomerInfo(String userName) | | | 根据会员ID显示会员信息 |
| Customer.successLog(String userName) | | | 获取用户VO |
| Customer.getCustomerInfo(String userName) | | | 得到用户信息 |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Customer.login | 语法 | public ResultMessage login(String userName, String password); | |
| 前置条件 | 启动一个会员管理回合 | |
| 后置条件 | 在一个会员管理回合中，加入一个customer对象 | |
| Customer.addCustomer | 语法 | public ResultMessage addCustomer(CustomerVO vo); | |
| 前置条件 | 会员已注册 | |
| 后置条件 | 在一个会员管理回合中，插入一个customer对象 | |
| Customer.ChangeCustomerInfo | 语法 | public ResultMessage changeCustomerInfo(CustomerVO vo); | |
| 前置条件 | 会员已登录 | |
| 后置条件 | 返回此会员管理回合中修改的会员信息 | |
| Customer.getCustomerInfo | 语法 | public CustomerVO getCustomerInfo(String userName); | |
| 前置条件 | 会员已登录 | |
| 后置条件 | 返回此会员的基本信息 | |
| Customer.successLog | 语法 | public CustomerVO successLog(String userName); | |
| 前置条件 | 成功登陆 | |
| 后置条件 | 返回用户信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| CustomerDataService.find(String userName, String password) | | | 根据会员ID和密码进行查找单一持久化对象 |
| CustomerDataService.insert(CustomerPO po) | | | 在数据库中插入CustomerPO对象 |
| CustomerDataService.update(CustomerPO po) | | | 在数据库中更新CustomerPO对象 |
| CustomerDataService.getCustomer(String userName) | | | 根据会员ID得到会员持久化对象 |

**（4）业务逻辑层的动态结构**

图1-1表明了酒店预订系统中，当会员注册之后，顾客业务逻辑处理的相关对象之间的协作。

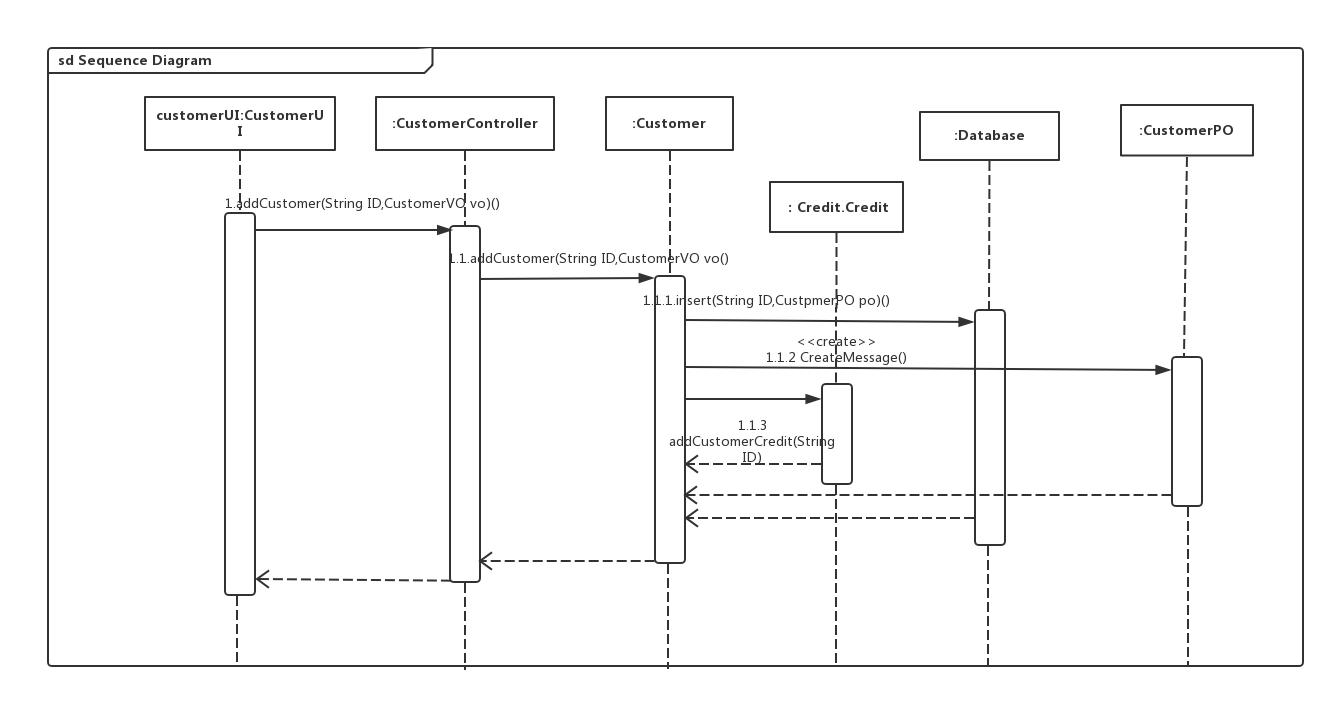


图1-1 注册顺序图

图1-2表明了酒店预订系统中，当会员登录之后，顾客业务逻辑处理的相关对象之间的协作。

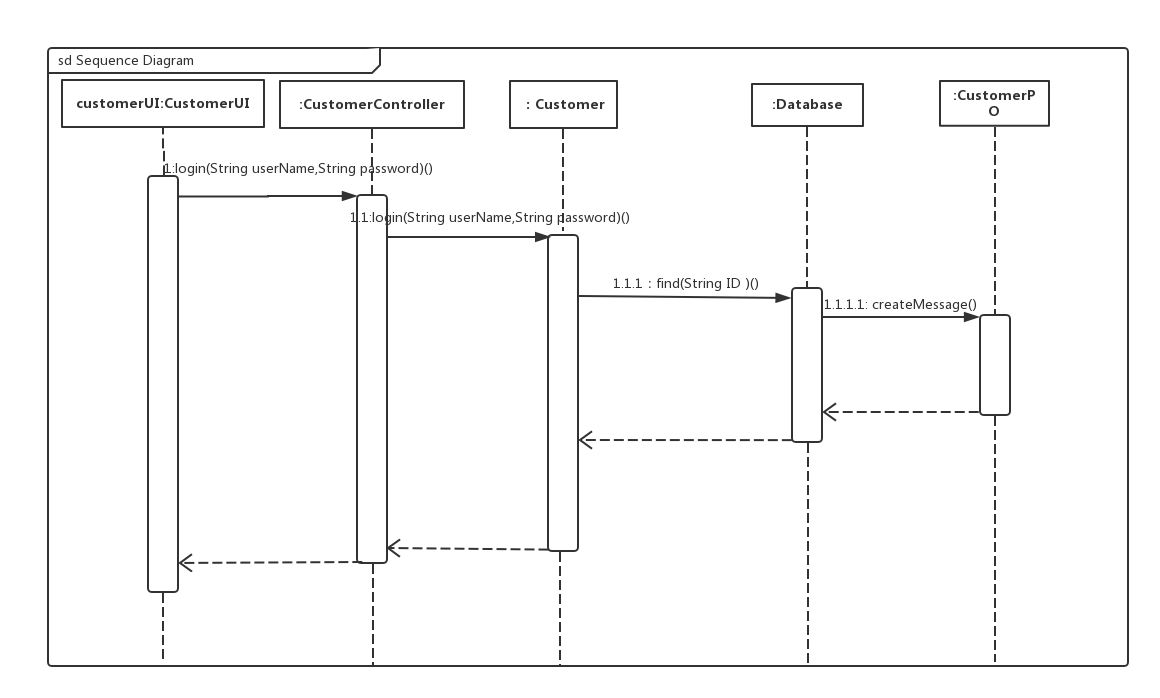


图1-2 登录顺序图

图1-3表明了酒店预订系统中，当会员或网站管理人员查看会员信息之后，顾客业务逻辑处理的相关对象之间的协作。

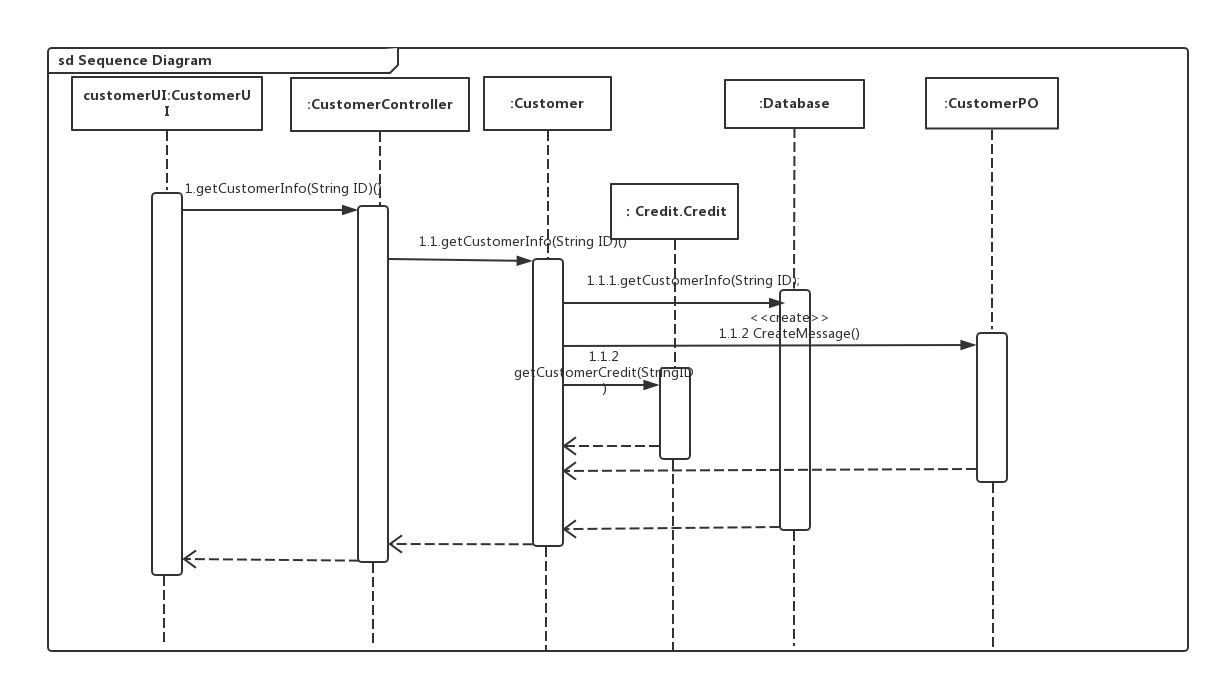


图1-3 查看会员信息的顺序图

图1-4表明了酒店预订系统中，当会员或网站管理人员修改会员信息之后，顾客业务逻辑处理的相关对象之间的协作。

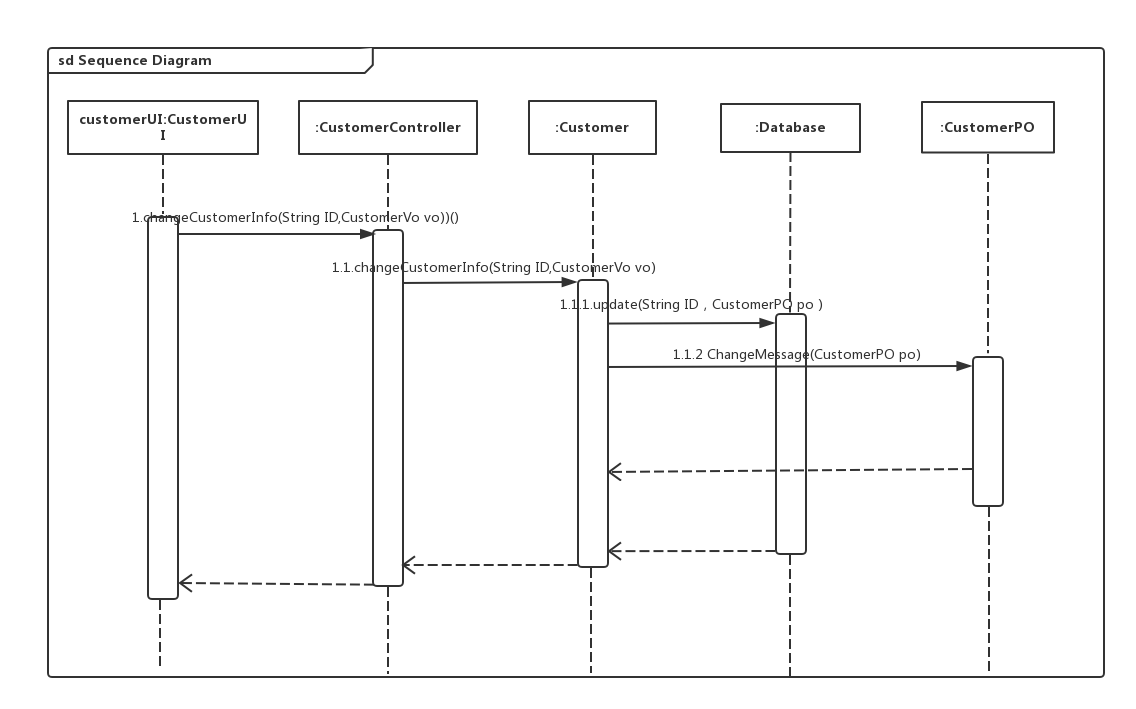


图1-4 修改会员信息的顺序图

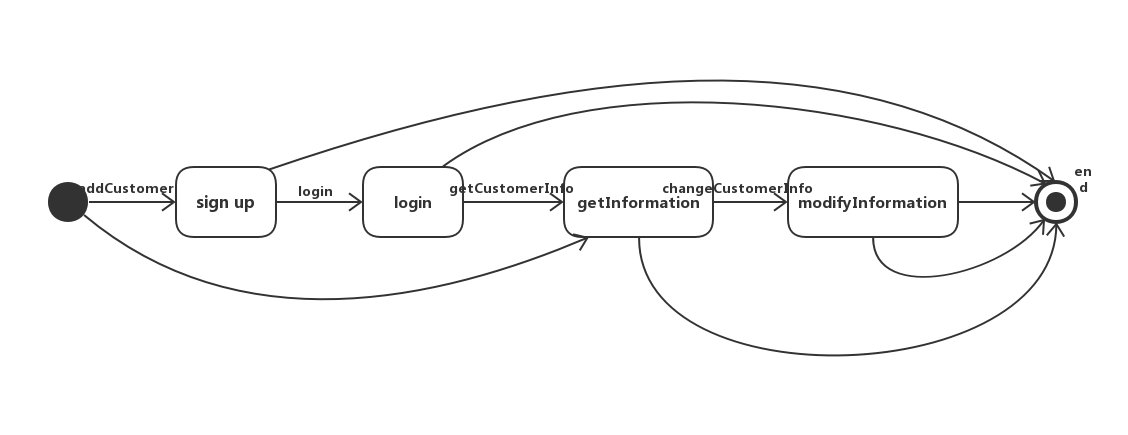
如图1-5所示的状态图描述了Customer对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着addCustomer方法被UI调用，Customer进入Sign up状态；之后通过会员登录进入login状态,通过查看会员信息进入getInformation状态，通过修改会员信息进入modifyInformation状态。UI也可以调用gtCustomerInfo，使Customer进入getInformation状态.

图1-5 Customer对象状态图

**（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其委托给不同的领域对象。

4.1.2 managerbl模块

（1）模块概述

managerbl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。

managerbl模块的职责及接口参见软件体系结构描述文档表 。

（2）整体结构

根据体系结构的设计,我们将系统分为展现层、业务逻辑层、数据层。每一层之间为了增加灵活性,我们会添加接口。比如展示层和业务逻辑层之间,我们添加blservice.managerblservice.ManagerBLService接口。业务逻辑层和数据层之间添加 dataservice.managerdataservice.ManagerDataService接口。为了隔离业务逻辑职责和逻辑控制职责,我们增加了 ManagerController,这样ManagerController会将对审查的业务逻辑处理委托给Hotel对象。ManagerPO是作为审查记录的持久化对象被添加到设计模型中去的。

managerbl模块的设计如图2所示。

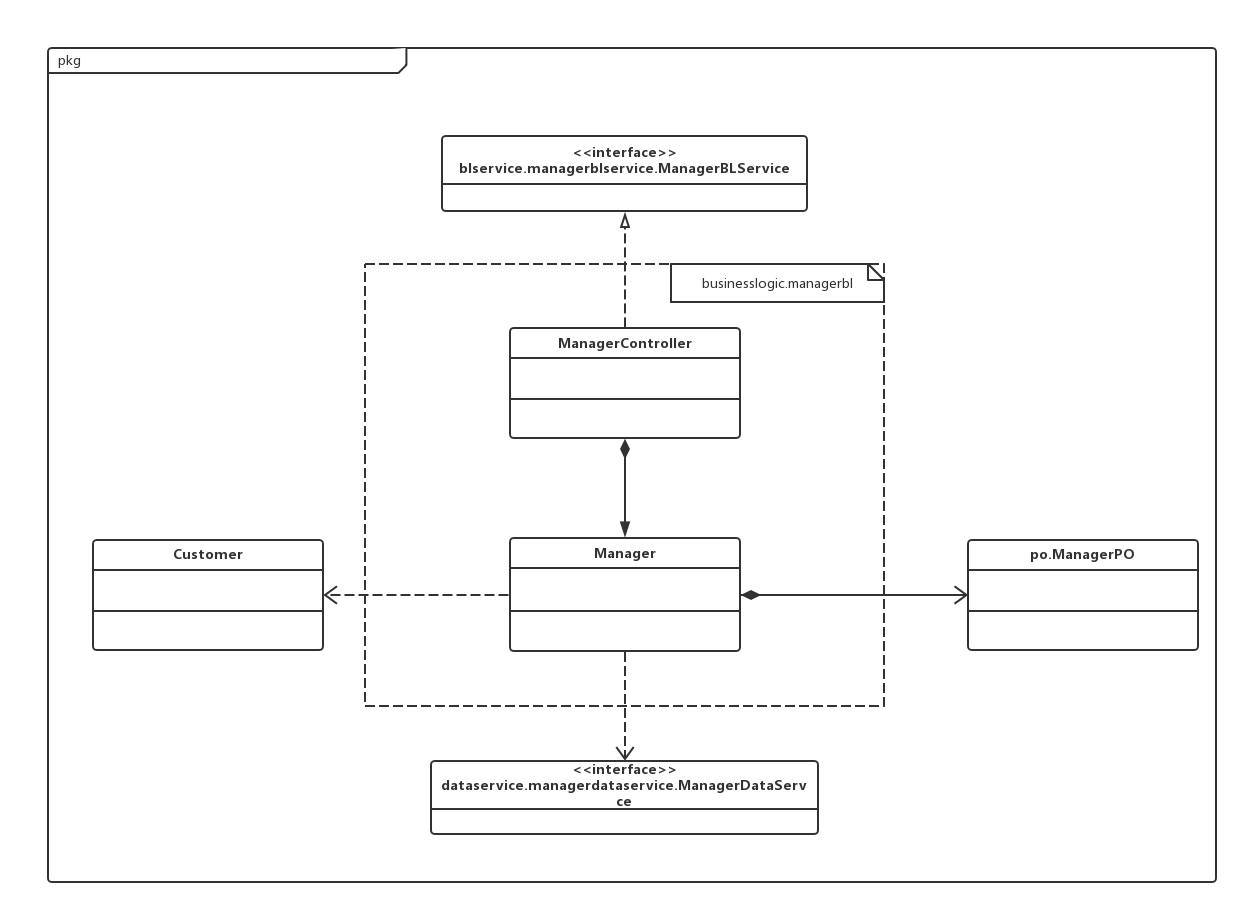


图2 managerbl模块各个类的设计

managerbl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| ManagerController | 负责实现管理人员界面所需要的服务 |
| Manager | 管理人员的领域模型对象，拥有一次管理所需的管理人员的信息，可以帮助完成管理人员界面所需要的服务 |

**（3）模块内部类的接口规范**

ManagerController和Manager的接口规范如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| ManagerController.addManager | 语法 | public ResultMessage addManager (ManagerVo vo) | |
| 前置条件 | 已创建一个Manager领域对象，且管理人员信息合法 | |
| 后置条件 | 调用Manager领域对象的addManager方法 | |
| ManagerController.changeManagerInfo | 语法 | public ResultMessage changeManagerInfo (ManagerVovo) | |
| 前置条件 | 已创建一个Manager领域对象，且管理人员信息合法 | |
| 后置条件 | 调用Manager领域对象的changeManagerInfo方法 | |
| ManagerController.login | 语法 | publicResultMessage login (ManagerType type，String username, String password) | |
| 前置条件 | 已创建一个Manager领域对象，且Password符合输入规则 | |
| 后置条件 | 调用Manager领域对象的login方法 | |
| ManagerController.getManagerInfo | 语法 | public ManagerVO getManagerInfo(ManagerType managerType, String username); | |
| 前置条件 | 无 | |
| 后置条件 | 调用Manager领域对象的getManagrInfo方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Manager.addManager (ManagerVo vo) | | | 添加新的管理人员 |
| Manager.changeManagerInfo ( ManagerVovo) | | | 修改管理人员信息 |
| Manager.login (ManagerType type，String username, String password) | | | 管理人员登录验证 |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Manager.addManager | 语法 | public ResultMessage addManager (ManagerType type, String ID, ManagerVo vo) | |
| 前置条件 | 管理人员信息合法 | |
| 后置条件 | 系统创建相应管理人员，更新相应信息 | |
| Manager.changeManagerInfo | 语法 | public ResultMessage changemanagerInfo (ManagerType type, String ID, ManagerVovo) | |
| 前置条件 | 管理人员信息合法 | |
| 后置条件 | 系统更新其信息 | |
| Manager.login | 语法 | publicResultMessage login (ManagerType type，String username, String password) | |
| 前置条件 | Password符合输入规则 | |
| 后置条件 | 系统查找是否存在对应用户，并根据password返回验证结果 | |
| Manager.getManagerInfo | 语法 | public ManagerVO getManagerInfo(ManagerType managerType, String username); | |
| 前置条件 | 无 | |
| 后置条件 | 根据用户类型和用户名返回ManagerVO | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ManagerDataService.findManagerInfo(ManagerType managerType, String username) | | | 在数据库查找指定的管理人员信息 |
| ManagerDataService.insertManager(ManagerPo po) | | | 在数据库中插入新的管理人员 |
| ManagerDataService.updateManager(ManagerPo po) | | | 在数据库中更新管理人员信息 |
| ManagerDataService.check(ManagerType managerType, String username, String password) throws RemoteException | | | 管理人员登录校验 |

**4）业务逻辑层的动态结构**

图2-1表明了酒店预订系统中，当管理人员进行登录操作时，管理人员业务逻辑处理的相关对象之间的协作。

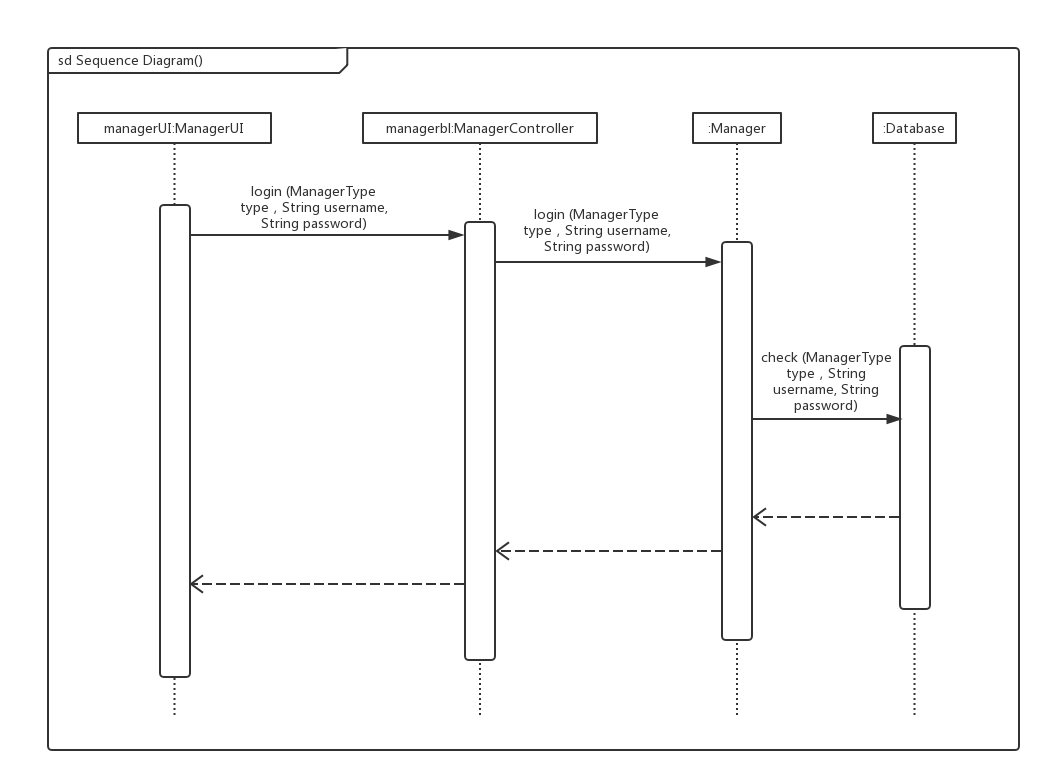


图2 – 1 管理人员登录顺序图

图2-2表明了酒店预订系统中，当添加管理人员时，管理人员业务逻辑处理的相关对象之间的协作。

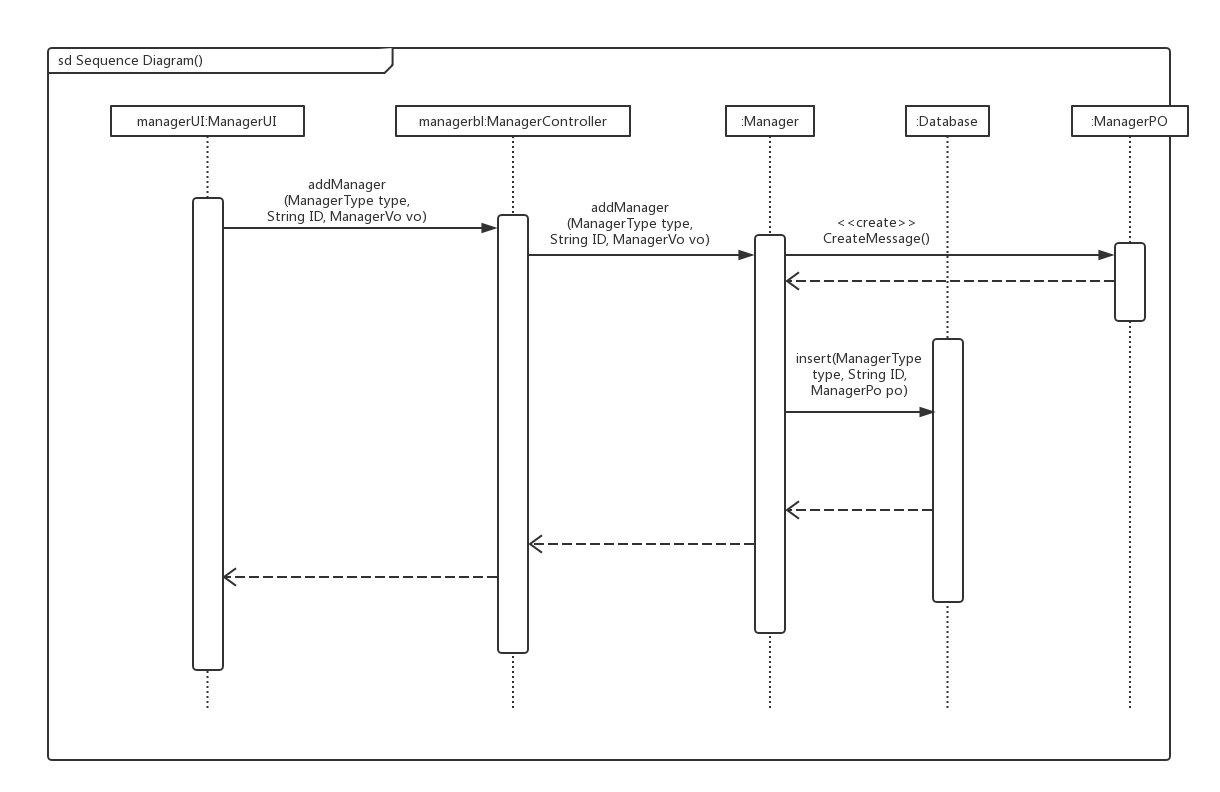


图2 – 2 添加管理人员顺序图

图2-3表明了酒店预订系统中，当修改管理人员信息时，管理人员业务逻辑处理的相关对象之间的协作。

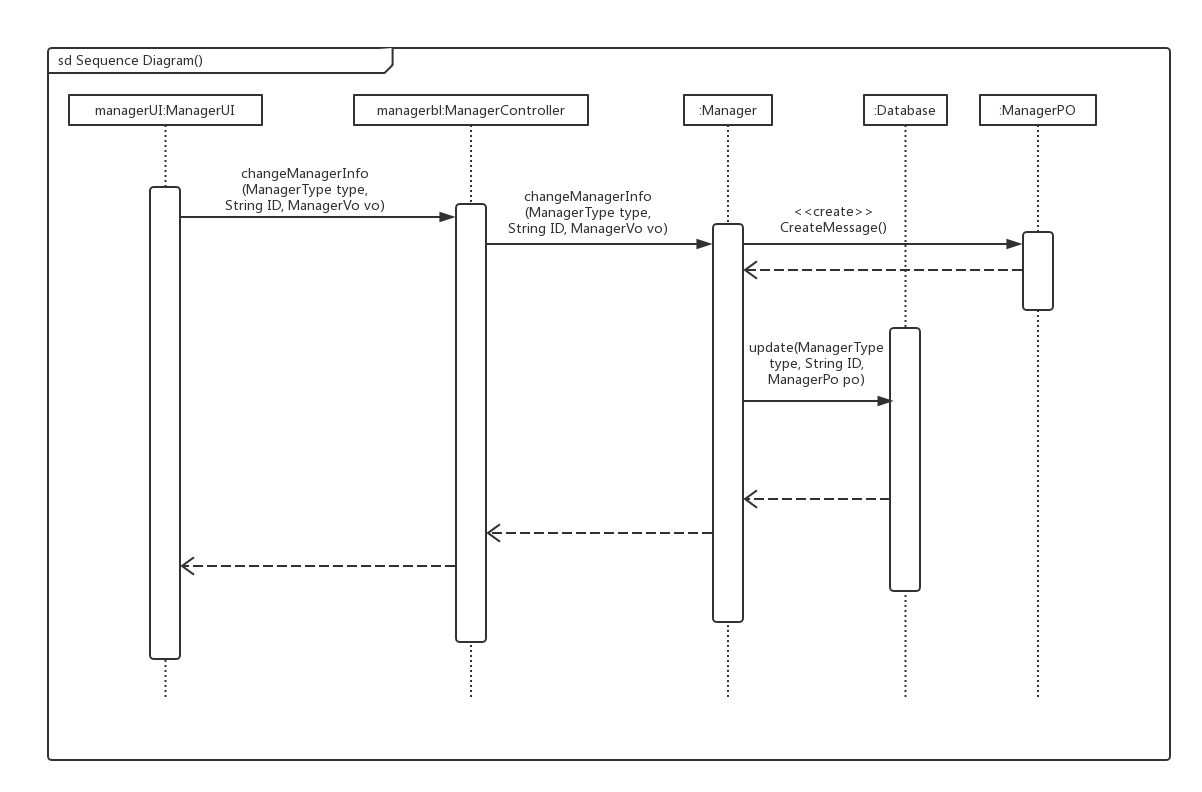


图2 – 3 修改管理人员信息顺序图

如图2-4所示的状态图描述了Manager对象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。Controller可以调用login，使Manager进入login状态，之后通过添加管理人员进入addManager状态，也可以通过修改管理人员信息进入modifyManagerInfo状态。

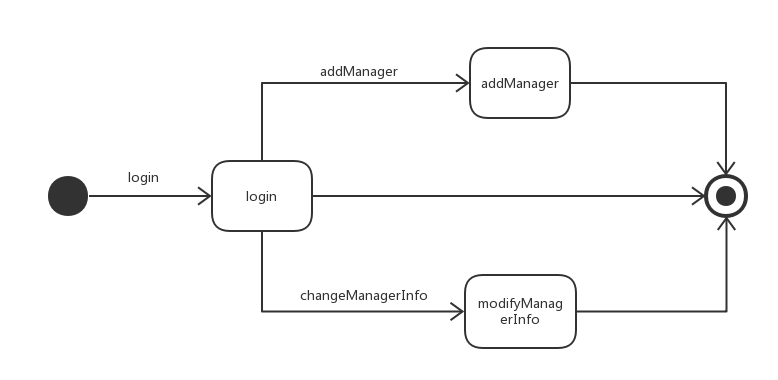


图2-4 Manager对象状态图

**4.1.3hotelbl模块**

**（1）模块概述**

hotelbl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。

hotelbl模块的职责及接口参见软件体系结构描述文档表 。

**（2）整体结构**

根据体系结构的设计,我们将系统分为展现层、业务逻辑层、数据层。每一层之间为了增加灵活性,我们会添加接口。比如展示层和业务逻辑层之间,我们添加blservice.hotelblservice.HotelBLService接口。业务逻辑层和数据层之间添加 dataservice.hoteldataservice.HotelDataService接口。为了隔离业务逻辑职责和逻辑控制职责,我们增加了 HotelController,这样HotelController会将对审查的业务逻辑处理委托给Hotel对象。HotelInfoPO,SearchInfoPO是作为审查记录的持久化对象被添加到设计模型中去的。

hotelbl模块的设计如图3所示。

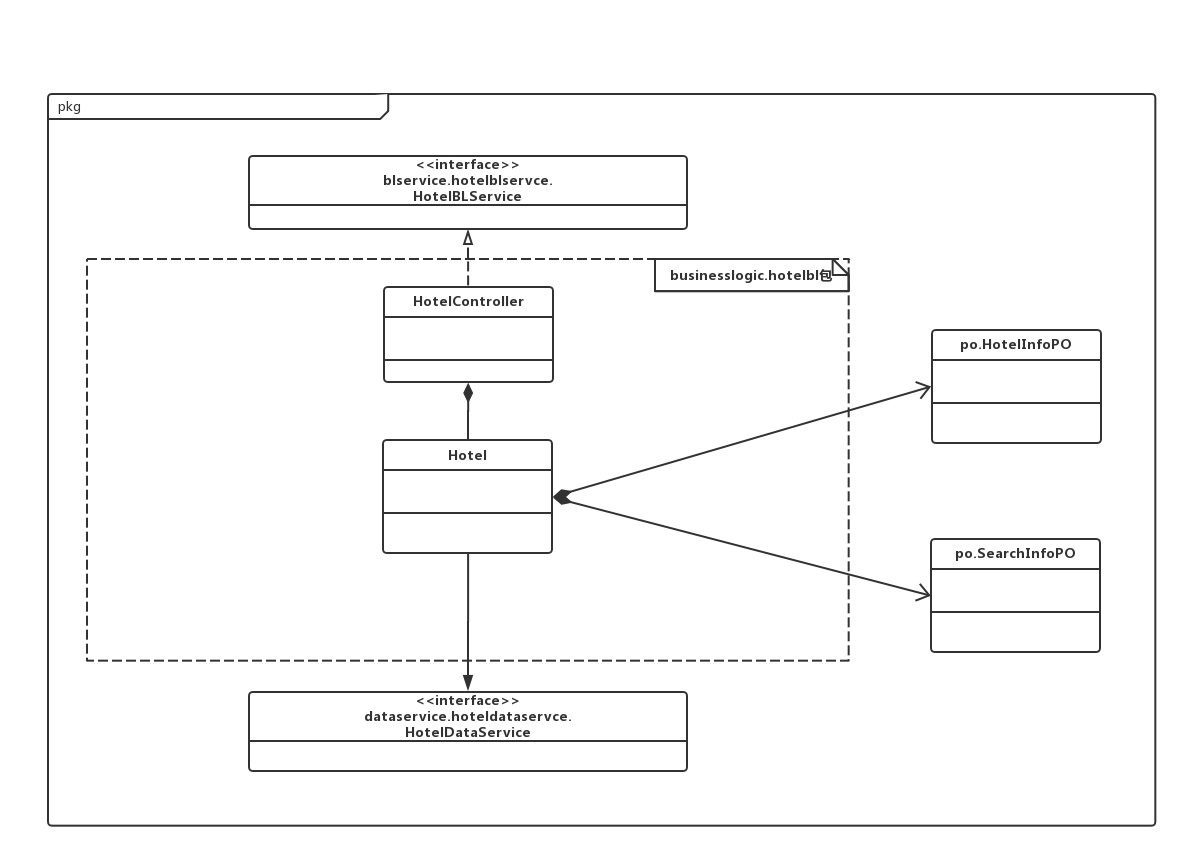


图3 hotelbl模块各个类的设计

hotelbl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| HotelController | 负责实现酒店管理界面所需要的服务 |
| Hotel | 酒店管理的领域模型对象，拥有一次管理所需的酒店等信息，可以帮助完成酒店管理界面所需要的服务 |

**（3）模块内部类的接口规范**

HotelController和Hotel的接口规范如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| HotelController.getHotelInfo | 语法 | public HotelInfoVo getHotelInfo(String hotelName) | |
| 前置条件 | 已创建一个Hotel领域对象 | |
| 后置条件 | 调用Hotel领域对象的getHotelInfo方法 | |
| HotelController.modifyHotelInfo | 语法 | public ResultMessage modifyHotelInfo(HotelInfoVo hivo) | |
| 前置条件 | 已创建一个Hotel领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用Hotel领域对象的modifyHotelInfo方法 | |
| HotelController.getHistoryHotelList | 语法 | public ArrayList<HotelInfoVo> getHistoryList(String customerID) | |
| 前置条件 | 已创建一个Hotel领域对象 | |
| 后置条件 | 调用Hotel领域对象的getHistoryHotelList方法 | |
| HotelController.getHotelList | 语法 | public ArrayList<HotelInfoVo> getHotelList (String customerID, SearchInfoVo sivo) | |
| 前置条件 | 已创建一个Hotel领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用Hotel领域对象的getHotelList方法 | |
| HotelController.addHotel | 语法 | public ResultMessageaddHotel(HotelInfoVO hivo) | |
| 前置条件 | 已创建一个Hotel领域对象 | |
| 后置条件 | 调用Hotel领域对象的addHotel方法 | |
| HotelController.updateHotelScore | 语法 | public ResultMessage updateHotelScore(String hotelName, double score); | |
| 前置条件 | 用户评价订单 | |
| 后置条件 | 调用Hotel领域对象的updateHotelScore方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Hotel. getHotelInfo(String hotelName) | | | 得到酒店基本信息 |
| Hotel. modifyHotelInfo(HotelInfoVo hivo) | | | 修改酒店基本信息 |
| Hotel. getHistoryHotelList(String customerID) | | | 得到历史酒店列表 |
| Hotel. getHotelList (String customerID, SearchInfoVo sivo) | | | 得到酒店列表 |
| Hotel. addHotel(HotelInfoVO hivo) | | | 添加酒店 |
| Hotel. updateHotelScore(String hotelName, double score) | | | 更新酒店评分 |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Hotel.getHotelInfo | 语法 | public HotelInfoVo getHotelInfo(String hotelName) | |
| 前置条件 | 无 | |
| 后置条件 | 返回某一酒店的详细信息 | |
| Hotel.modifyHotelInfo | 语法 | public ResultMessage modifyHotelInfo(HotelInfoVo hivo) | |
| 前置条件 | 输入了正确的酒店修改信息 | |
| 后置条件 | 修改酒店信息，返回修改结果 | |
| Hotel.getHistoryHotelList | 语法 | public ArrayList<HotelInfoVo> getHistoryList(String customerID) | |
| 前置条件 | 无 | |
| 后置条件 | 返回历史酒店列表 | |
| Hotel.getHotelList | 语法 | public ArrayList<HotelInfoVo> getHotelList (String customerID, SearchInfoVo sivo) | |
| 前置条件 | 输入了正确的搜索条件 | |
| 后置条件 | 返回酒店列表 | |
| Hotel.addHotel | 语法 | public ResultMessageaddHotel(HotelInfoVO hivo) | |
| 前置条件 | 无 | |
| 后置条件 | 返回酒店添加结果 | |
| Hotel.updateHotelScore | 语法 | public ResultMessage updateHotelScore(String hotelName, double score); | |
| 前置条件 | 用户评价订单 | |
| 后置条件 | 更新酒店评分 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| HotelDataService.findHotelInfo（String hotelName） | | | 根据酒店名称查找单个酒店详细信息 |
| HotelDataService.updateHotelInfo（HotelInfoPO hipo） | | | 更新酒店详细信息 |
| HotelDataService.findHotels（SearchInfoPO sipo） | | | 根据搜索条件查找多个酒店 |
| HotelDataService.insertHotel（HotelInfoPO hipo） | | | 根据输入的酒店信息添加酒店 |
| HotelDataService.updateHotelScore(String hotelName, double score) | | | 更新酒店评分 |
| Order.getOrderVOList() | | | 得到用户所有订单列表 |

**4）业务逻辑层的动态结构**

图3-1表明了酒店预订系统中，当酒店工作人员请求查看酒店基本信息之后，酒店业务逻辑处理的相关对象之间的协作。

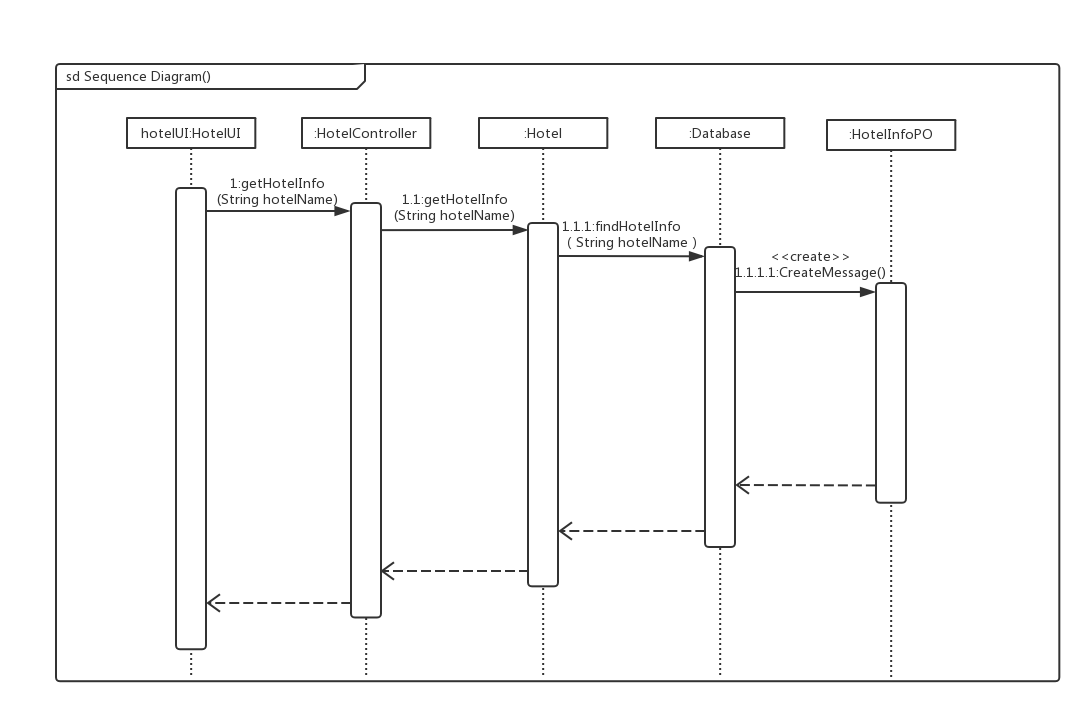


图3-1 查看酒店基本信息顺序图

图3-2表明了酒店预订系统中，当酒店工作人员请求修改酒店基本信息之后，酒店业务逻辑处理的相关对象之间的协作。

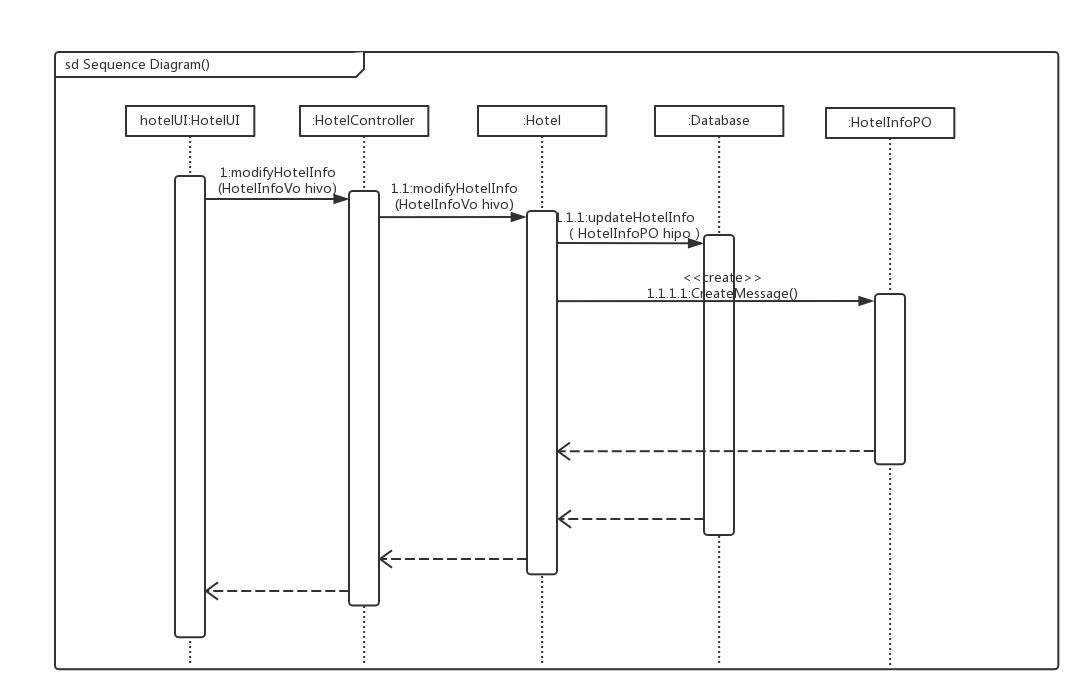


图3-2 修改酒店基本信息顺序图

图3-3表明了酒店预订系统中，当会员请求查看住过的酒店之后，酒店业务逻辑处理的相关对象之间的协作。

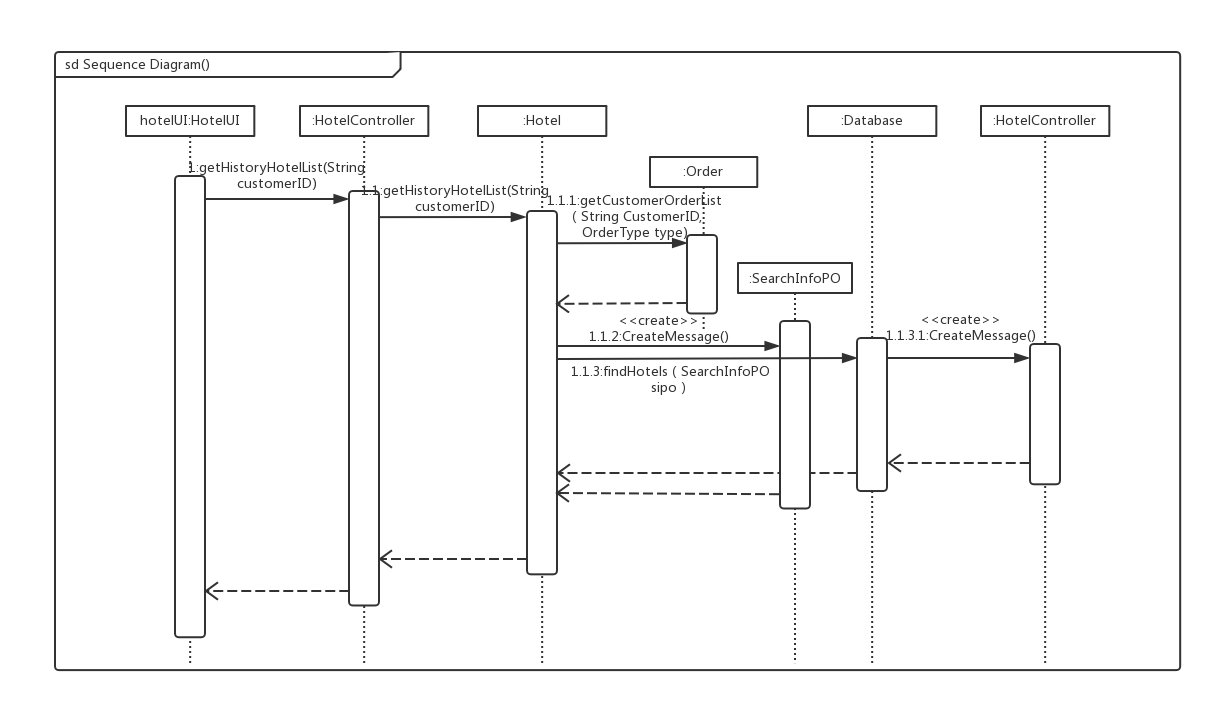


图3-3查看住过的酒店的顺序图

图3-4表明了酒店预订系统中，当会员请求搜索酒店之后，酒店业务逻辑处理的相关对象之间的协作。

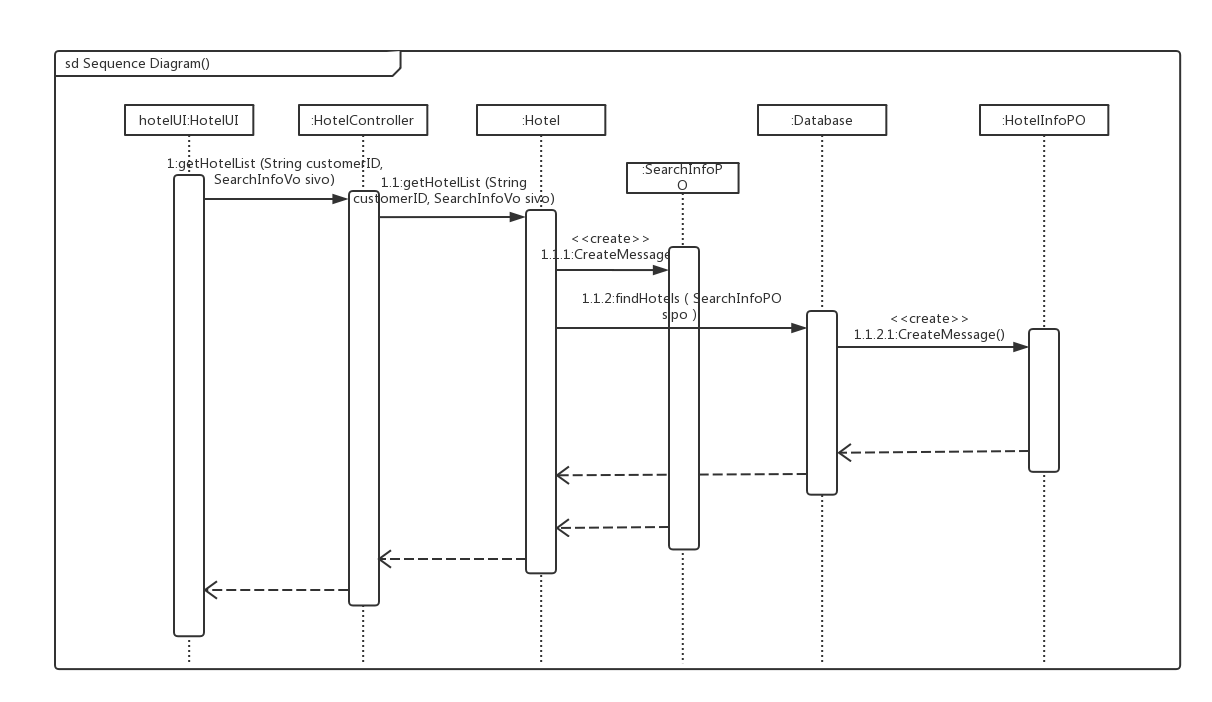


图3-4 搜索酒店的顺序图

图3-5表明了酒店预订系统中，当网站管理人员请求添加酒店之后，酒店业务逻辑处理的相关对象之间的协作。

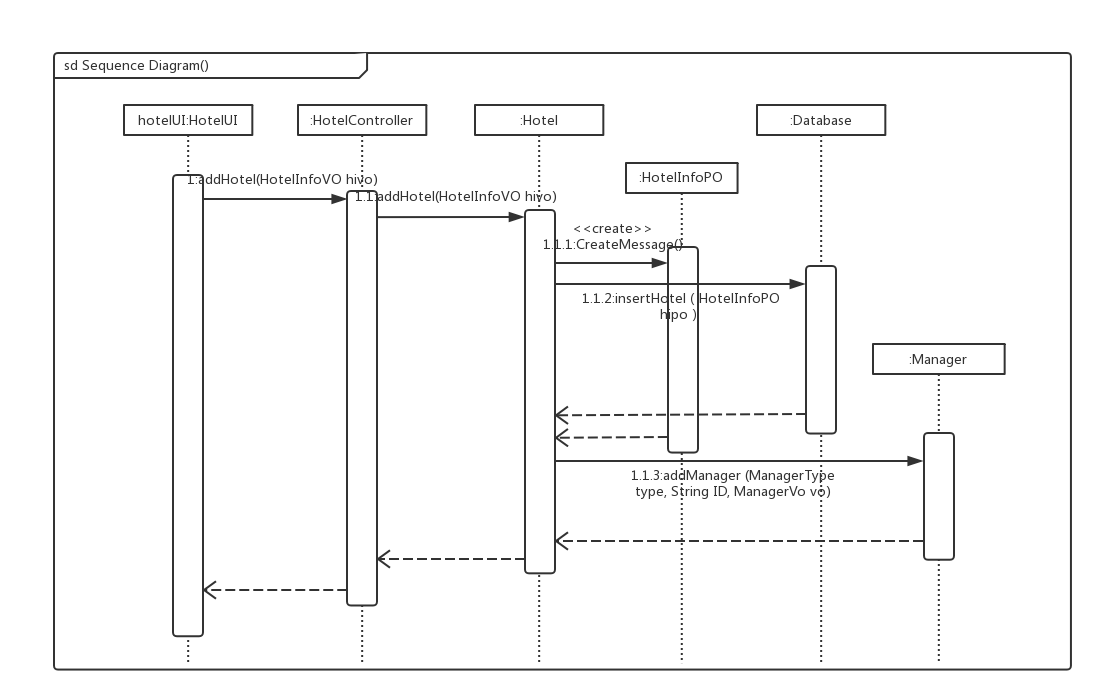


图3-5添加酒店的顺序图

如图3-6所示的状态图描述了Hotel对象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。Controller可以调用getHotelInfo，使Hotel进入getHotelInfo状态，之后通过修改酒店基本信息进入modifyHotelInfo状态。Controller也可以调用addHotel，使Hotel进入addHotel状态,调用getHotelList使Hotel进入getHotelList状态。

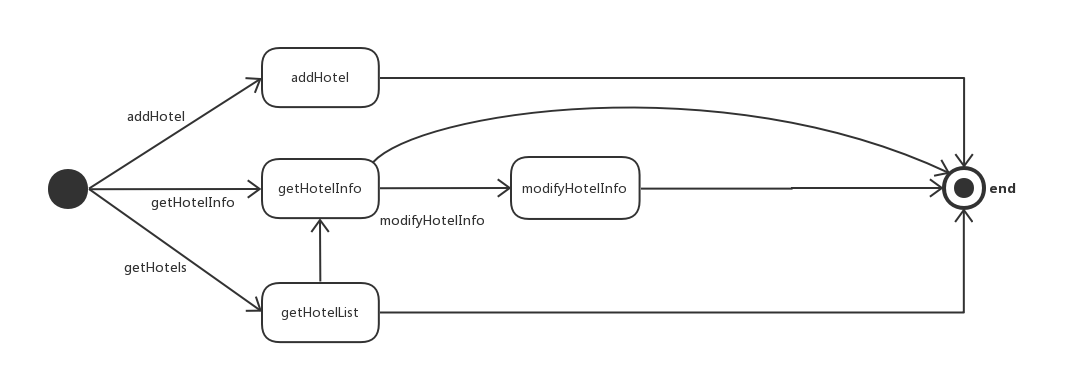


图3-6Hotel对象状态图

**4.1.4orderbl模块**

**（1）模块概述**

orderbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

orderbl模块的职责及接口参见软件体系结构描述文档表 。

**（2）整体结构**

根据体系结构的设计,我们将系统分为展现层、业务逻辑层、数据层。每一层之间为了增加灵活性,我们会添加接口。比如展示层和业务逻辑层之间,我们添加blservice.orderblservice.OrderBLService接口。业务逻辑层和数据层之间添加 dataservice.orderdataservice.OrderDataService接口。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会将对订单的业务逻辑处理委托给Order对象。OrderPO是作为订单的持久化对象被添加到设计模型中去的。而OrderList和orderuLineItem的添加是List的容器类。OrderLineItem保有订单编号、酒店名、会员名和价格等数据,及生成、执行、撤销的职责。而 OrderList封装了关于OrderLineItem的数据集合的数据结构的秘密和显示订单列表的职责。CustomerCrededit是根据依赖倒置原则，为了消除循环依赖而产生的接口。

hotelbl模块的设计如图4所示。

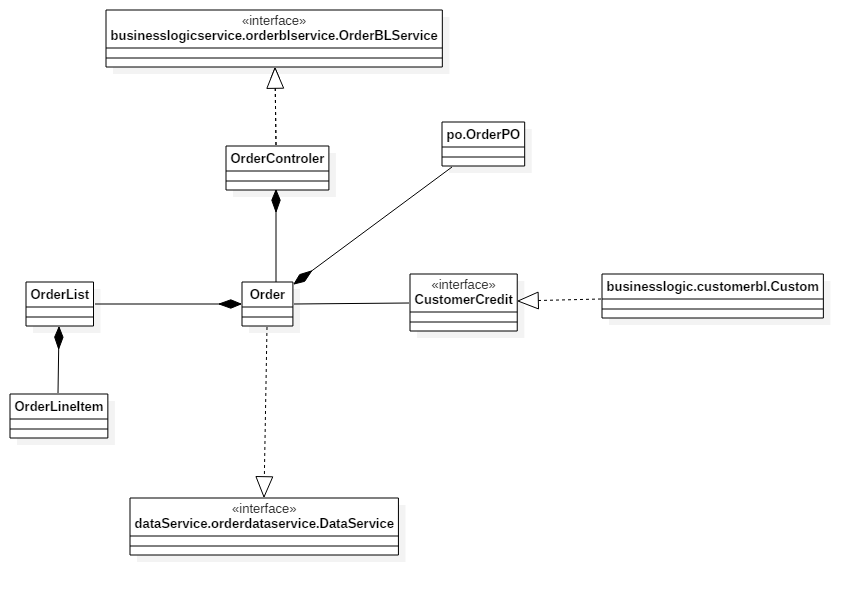


图4 orderbl模块各个类的设计

orderbl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| Ordercontroller | 负责分发订单处理界面所需要的服务 |
| Order | 订单处理的领域模型对象，拥有订单处理所需要的订单信息、酒店信息、会员信息等等，可以帮助完成订单管理界面所需要的服务 |

**（3）模块内部类的接口规范**

orderController和Order的接口规范如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrderController.newOrder | 语法 | public ResultMessage newOrder(OrderVO vo); | |
| 前置条件 | 输入符合规则 | |
| 后置条件 | 调用Order领域对象newOrder方法 | |
| OrderController.getHotelOrderList | 语法 | public ArrayList<OrderVo>getHotelOrderList (String HotelName, OrderType type); | |
| 前置条件 | 输入符合规则 | |
| 后置条件 | 调用Order领域对象的getHotelOrderList方法 | |
| OrderController.getMemberOrderList | 语法 | public ArrayList<OrderVo>getCustomerOrderList (String CustomerID, OrderType type); | |
| 前置条件 | 符合输入规则 | |
| 后置条件 | 调用Order领域对象的getMemberOrderList方法 | |
| OrderController.getOrderInfo | 语法 | public OrderVO getOrderInfo(String OrderID); | |
| 前置条件 | 已经通过调用get\*\*Orderlist方法得到orderList | |
| 后置条件 | 调用Order领域对象的getOrderInfo方法 | |
| OrderController.evaluateOrder | 语法 | public ResultMessage evaluateOrder(String OrderID); | |
| 前置条件 | 已经生成一个Order对象 | |
| 后置条件 | 调用Order领域对象的evaluateOrder方法 | |
| OrderController.executeOrder | 语法 | public ResuMessage excuteOrder(String OrderID) | |
| 前置条件 | 已生成一个Order对象 | |
| 后置条件 | 调用OrderItem对象的excuteOrder() | |
| OrderController.cancelOrder | 语法 | public ResultMessage cancelOrder(String OrderID); | |
| 前置条件 | 已经创建一个Order对象 | |
| 后置条件 | 调用Order的cancelOrder方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Order.newOrder(OrderVO vo) | | | 新增一个OrderItem对象 |
| Order. getMemberOrderList (String memberID, OrderType type) | | | 创建一个MemberOrderList对象 |
| Order. getHotelOrderList (String HotelName, OrderType type) | | | 创建一个HotelOrderList对象 |
| Order.getOrderInfo(String OrderID) | | | 根据订单编号显示订单信息 |
| Order.EvaluateOrder(String OrderID,String evaluation,double value) | | | 评价订单 |
| Order.excuteOrder(String OrderID) | | | 执行订单 |
| Order.cancelOrder(String OrderID) | | | 取消订单 |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Order.newOrder | 语法 | public ResultMessage newOrder(OrderVO vo); | |
| 前置条件 | 启动生成新订单 | |
| 后置条件 | 持久化生成新订单设计的领域对象的信息，并且显示订单价格和订单信息 | |
| Order.getHotelOrderList | 语法 | public ArrayList<OrderVo>getHotelOrderList (String HotelName, OrderType type); | |
| 前置条件 | 启动获取酒店订单列表 | |
| 后置条件 | 获取酒店订单列表 | |
| Order.getMemberOrderList | 语法 | public ArrayList<OrderVo>getCustomerOrderList (String CustomerID, OrderType type); | |
| 前置条件 | 启动获取会员订单列表 | |
| 后置条件 | 显示酒店订单列表 | |
| Order.getOrderInfo | 语法 | public OrderVo getOrderInfo (String OrderID) | |
| 前置条件 | 已经通过调用get\*\*Orderlist方法得到orderList | |
| 后置条件 | 调用Order领域对象的getOrderInfo方法 | |
| OrderController.evaluateOrder | 语法 | Public ResultMessage evaluateOrder (String OrderID,String evaluation,double value) | |
| 前置条件 | 已经生成一个OrderItem对象 | |
| 后置条件 | 调用Orderitem领域对象的evaluateOrder方法 | |
| OrderController.executeOrder | 语法 | Public ResultMessage executeOrder(String orderID); | |
| 前置条件 | 已生成一个OrderItem对象 | |
| 后置条件 | 调用OrderItem对象的excuteOrder() | |
| OrderController.cancelOrder | 语法 | public ResultMessage cancelOrder(String OrderID); | |
| 前置条件 | 已经创建一个OrderItem对象 | |
| 后置条件 | 调用OrderItem的cancelOrder方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| OrderDataService.getCustomerOrderList(String ID) | | | 根据会员ID获取订单列表 |
| OrderDataService.getOrderEvaluation(String orderID) | | | 根据订单ID获取订单评价 |
| OrderDataService.updateOrderEvaluation(String orderID，OrderEvaluationPO oepo) | | | 更新订单评价 |
| OrderDataService.insertOrder(OrderPO opo) | | | 插入单一持久化对象 |
| OrderDataService.deleteOrder(OrderPO opo) | | | 删除单一持久化对象 |
| OrderDataService.updateOrder(OrderPO opo) | | | 在数据库层中更更新单一持久化对象 |
| HotelDataService.insertAvailableRooms(AvailableRoomsPO arpo) | | | 在数据库层中增加可用客房 |
| HotelDataService.deleteAvailableRooms | | | 在数据库层中删除可用客房 |
| OrderDataService.getOrderDatabase | | | 得到order数据库引用 |

**4）业务逻辑层的动态结构**

如图表明了在酒店预订系统中，当用户输入了订单信息后，生成订单逻辑的相关对象之间的协作。

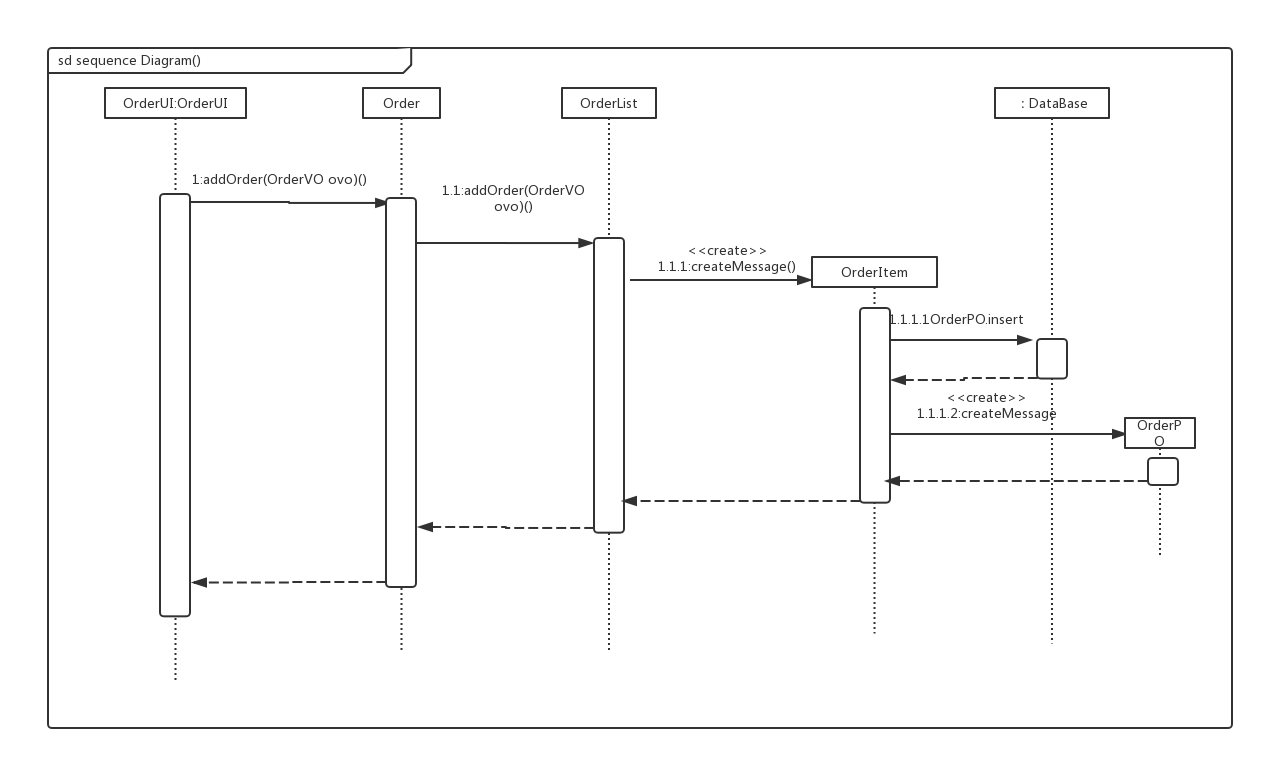


图4-1生成订单的顺序图

如图表明了在酒店预订系统中，获取会员订单列表逻辑的各相关对象之间的协作。

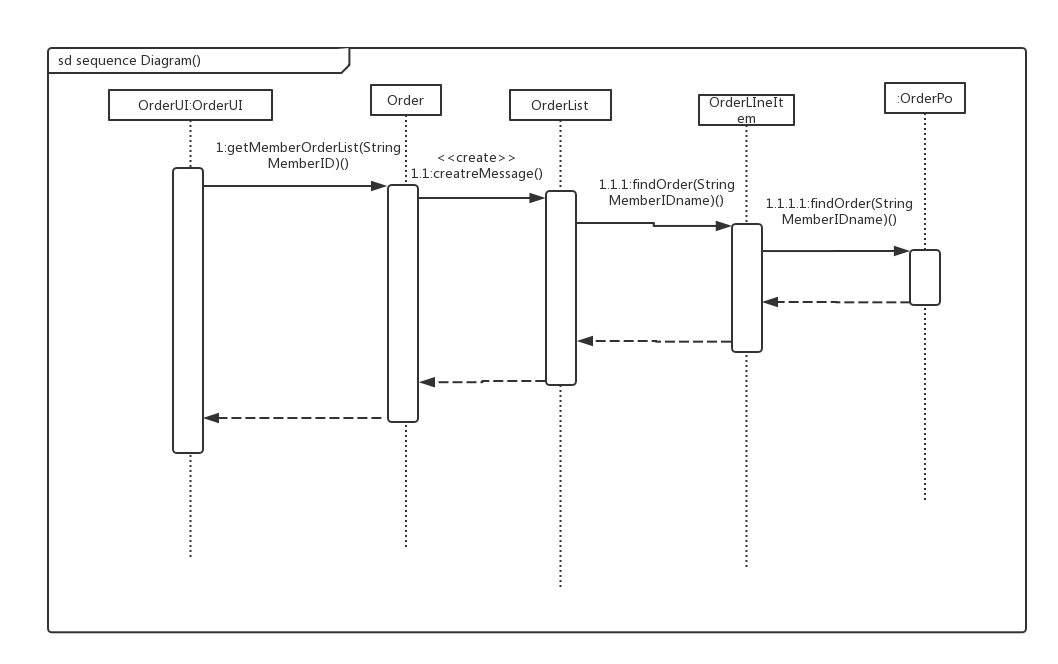


图4-2获取会员订单列表的顺序图

如图表明在酒店预订系统中，获取酒店订单列表相关对象之间的协作

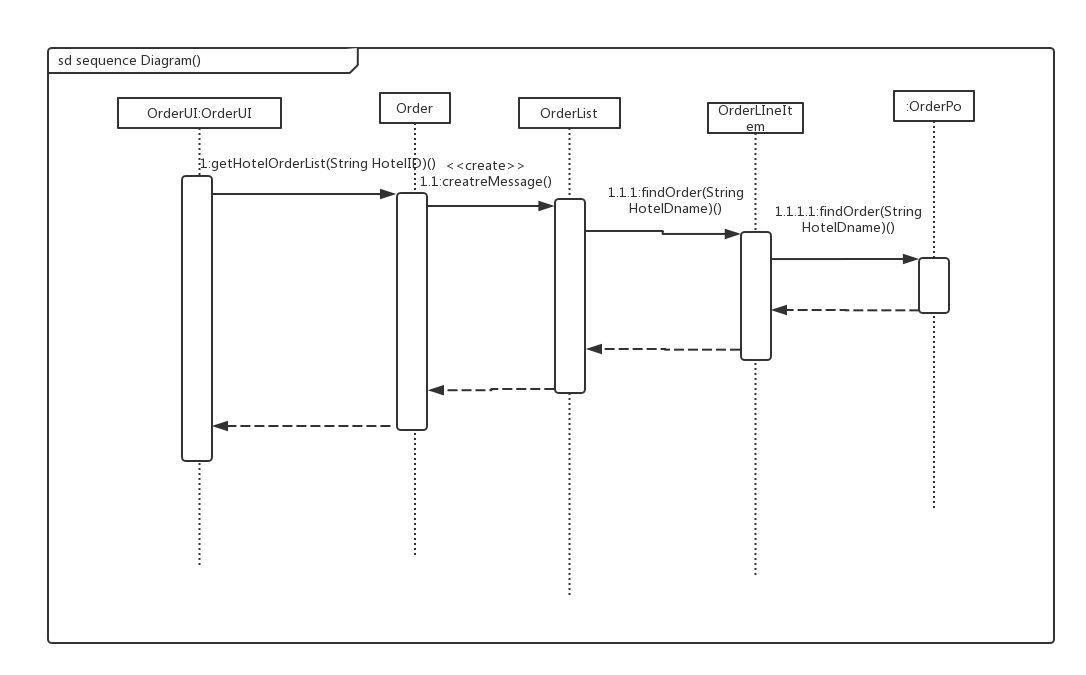


图4-3获取酒店订单列表的相关对象之间的协作

如图表明在酒店预订系统中，获取订单信息相关对象之间的协作

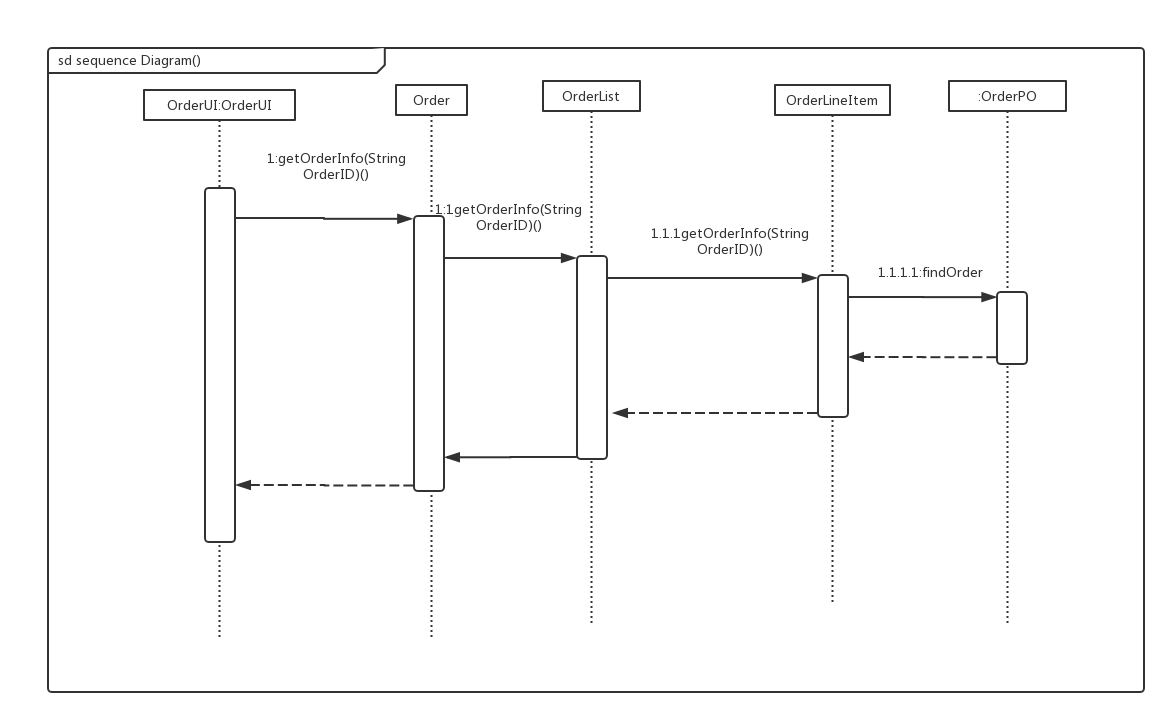


图4-4获取订单信息的顺序图

如图表明在酒店预订系统中，评价订单任务中各相关对象之间的协作

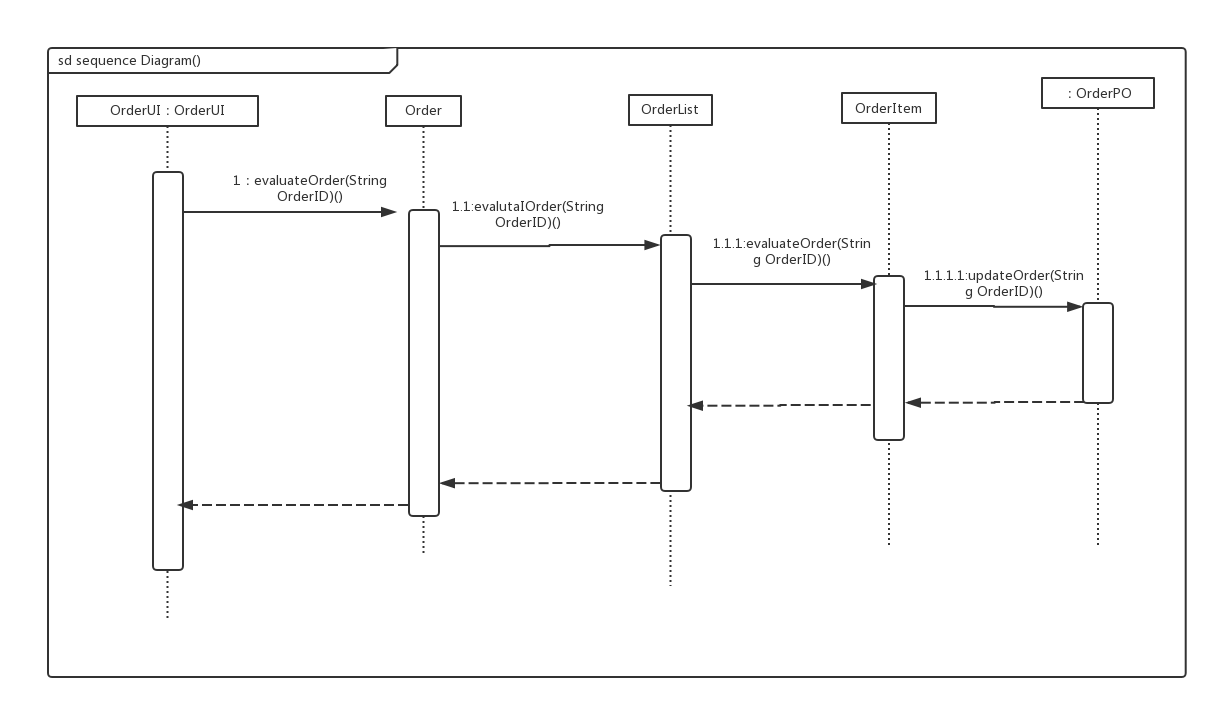


图5-5评价订单的顺序图

如图表明在酒店预订系统中，取消订单中相关对象的协作

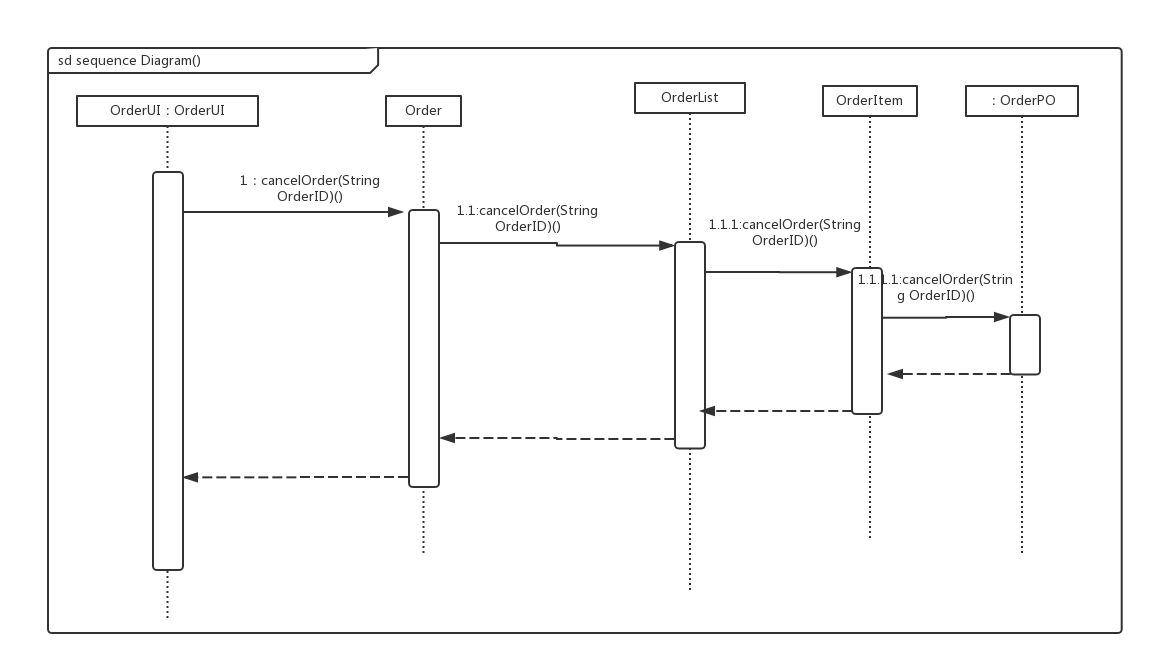


图5-6取消订单的顺序图

如图表明在酒店预订系统中，执行订单各对象之间的协作

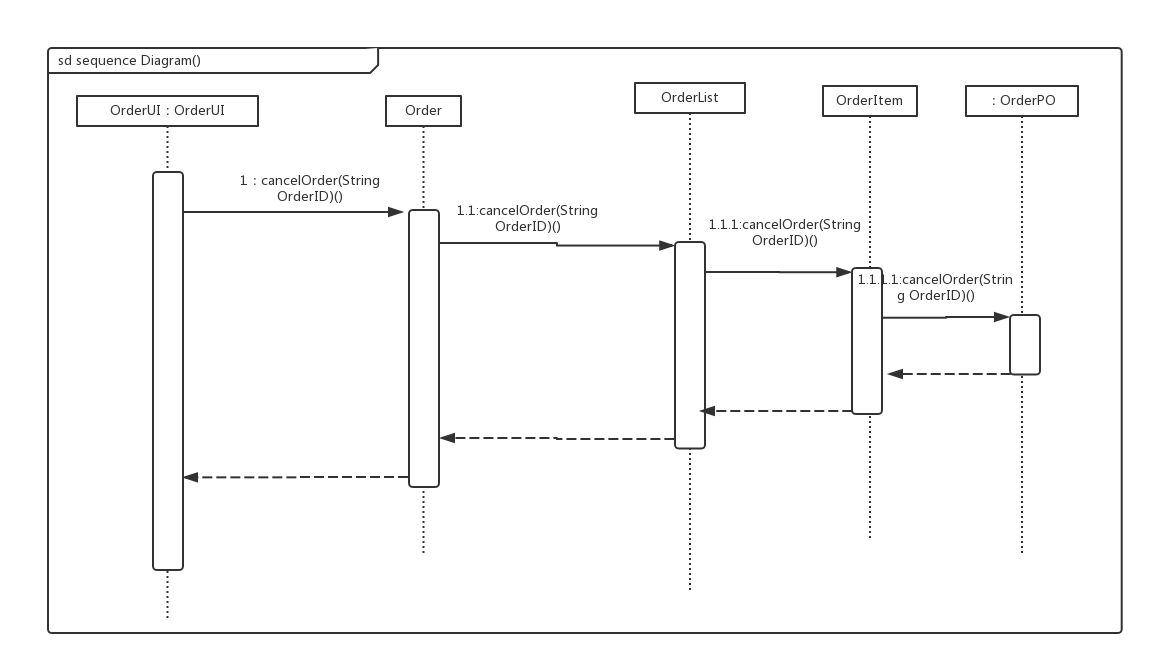
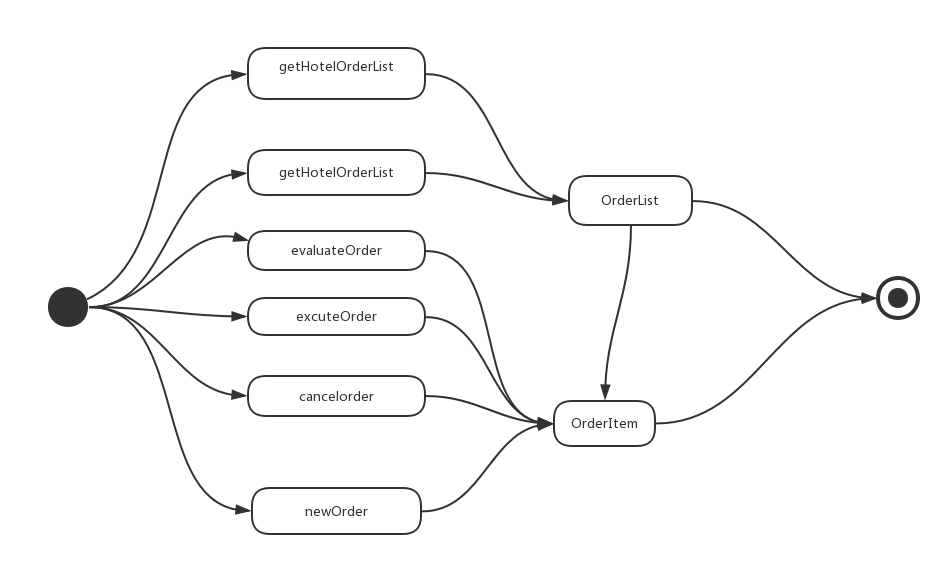


图5-6执行订单的顺序图

如图所示的状态图描述了order对象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由各自的控制其委托给不同的领域对象。

**4.1.5promotionbl模块**

**（1）模块概述**

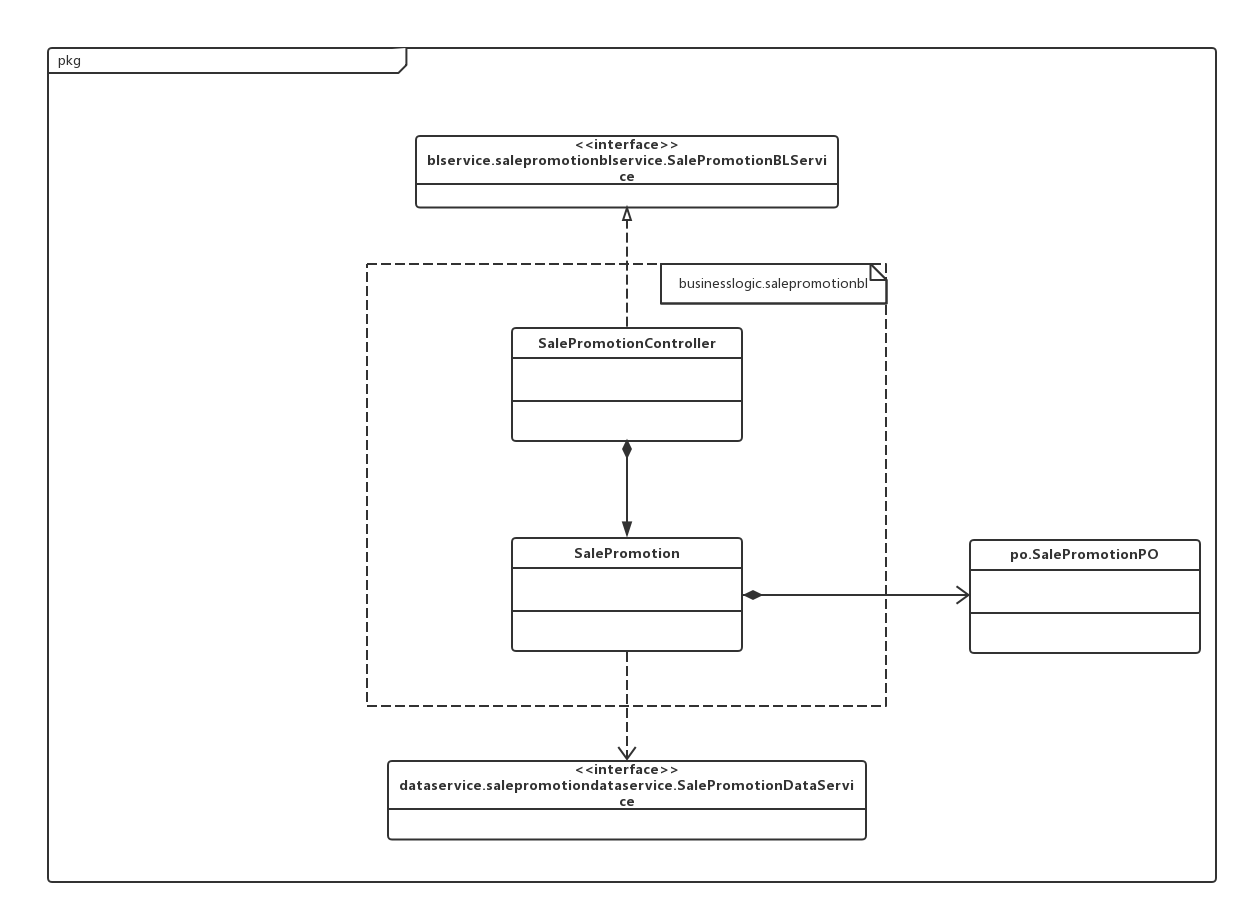
**promotion**bl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**promotion**bl模块的职责及接口参见软件体系结构描述文档表。

（2）整体结构

根据体系结构的设计,我们将系统分为展现层、业务逻辑层、数据层。每一层之间为了增加灵活性,我们会添加接口。比如展示层和业务逻辑层之间,我们添加blservice.promotionblservice.PromotionBLService接口。业务逻辑层和数据层之间添加 dataservice.promotiondataservice.PromotionDataService接口。为了隔离业务逻辑职责和逻辑控制职责,我们增加了 PromotionController,这样PromotionrController会将对审查的业务逻辑处理委托给Promotion对象。PromotionPO是作为审查记录的持久化对象被添加到设计模型中去的。

promotionbl模块的设计如图2所示。



promotionbl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| PromotionController | 负责实现促销策略管理界面所需要的服务 |
| Promotion | 促销策略管理的领域模型对象，拥有一次管理所需的策略类型、策略内容等信息，可以帮助完成促销策略管理界面所需要的服务 |

**（3）模块内部类的接口规范**

PromotionController和Promotion的接口规范如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| PromotionController.getHotelPromotion | 语法 | public HotelPromotionVO getHotelPromotion(HotelPromotionType hotelPromotionType, String hotelName); | |
| 前置条件 | 酒店工作人员查看促销策略，且已创建Promotion领域对象 | |
| 后置条件 | 调用Promotion. getHotelPromotion | |
| PromotionController.getWebPromotion | 语法 | public WebPromotionVO getWebPromotion(WebPromotionType webPromotionType); | |
| 前置条件 | 网站营销人员查看促销策略，且已创建Promotion领域对象 | |
| 后置条件 | 调用Promotion. getWebPromotion | |
| PromotionController.updateHotelPromotiom | 语法 | public ResultMessage updateHotelPromotion(HotelPromotionType hotelPromotionType, HotelPromotionVO hpvo); | |
| 前置条件 | 酒店工作人员更新相应促销策略，且已创建Promotion领域对象 | |
| 后置条件 | 调用Promotion. updateHotelPromotiom | |
| PromotionController.updateWebPromotion | 语法 | public ResultMessage updateWebPromotion(WebPromotionType webPromotionType, WebPromotionVO wpvo); | |
| 前置条件 | 网站营销人员更新相应促销策略，且已创建Promotion领域对象 | |
| 后置条件 | 调用Promotion. updateWebPromotion | |
| PromotionController.calculateLevel | 语法 | public int calculateLevel(double credit); | |
| 前置条件 | 已创建Promotion领域对象 | |
| 后置条件 | 调用Promotion. calculateLevel | |
| PromotionController.calculatePrice | 语法 | public OrderVO calculatePrice(OrderVO orderVO); | |
| 前置条件 | 已创建Promotion领域对象 | |
| 后置条件 | 调用Promotion. calculatePrice | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Promotion.getHotelPromotion | | | 获得一种酒店促销策略 |
| Promotion.getWebPromotion | | | 获得一种网站促销策略 |
| Promotion.updateHotelPromotion | | | 更新一种酒店促销策略 |
| Promotion.updateWebPromotion | | | 更新一种网站促销策略 |
| Promotion.calculateLevel | | | 计算等级 |
| Promotion.calculatePrice | | | 计算价格 |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Promotion.getHotelPromotion | 语法 | public HotelPromotionVO getHotelPromotion(HotelPromotionType hotelPromotionType, String hotelName); | |
| 前置条件 | 无 | |
| 后置条件 | 系统返回一种酒店促销策略 | |
| Promotion.getWebPromotion | 语法 | public WebPromotionVO getWebPromotion(WebPromotionType webPromotionType); | |
| 前置条件 | 无 | |
| 后置条件 | 系统返回一种网站促销策略 | |
| Promotion.updateHotelPromotion | 语法 | public ResultMessage updateHotelPromotion(HotelPromotionType hotelPromotionType, HotelPromotionVO hpvo); | |
| 前置条件 | 无 | |
| 后置条件 | 系统更新一种酒店促销策略 | |
| Promotion.updateWebPromotion | 语法 | public ResultMessage updateWebPromotion(WebPromotionType webPromotionType, WebPromotionVO wpvo); | |
| 前置条件 | 无 | |
| 后置条件 | 系统更新一种网站促销策略 | |
| Promotion.calculateLevel | 语法 | public int calculateLevel(double credit); | |
| 前置条件 | 无 | |
| 后置条件 | 系统返回会员等级 | |
| Promotion.calculatePrice | 语法 | public OrderVO calculatePrice(OrderVO orderVO); | |
| 前置条件 | 填写了完整的订单信息 | |
| 后置条件 | 系统返回计算价格后的OrderVO | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| PromotionDataService.findHotelPromotion(HotelPromotionType hotelPromotionType, String hotelName) | | | 在数据库中得到一种酒店促销策略 |
| PromotionDataService.updateHotelPromotion(HotelPromotionType hotelPromotionType, HotelPromotionPO hppo) | | | 在数据库中更新一种酒店促销策略 |
| PromotionDataService.findWebPromotion(WebPromotionType webPromotionType) | | | 在数据库中得到一种网站促销策略 |
| PromotionDataService.updateWebPromotion(WebPromotionType webPromotionType, WebPromotionPO wppo) | | | 在数据库中更新一种网站促销策略 |
| CustomerBLService.getCustomerInfo(String customerID) | | | 得到会员信息 |
| HotelBLService.getHotelInfo(String hotelName) | | | 得到酒店信息 |

**（4）业务逻辑层的动态结构**

图5-1表明了酒店预订系统中，当需要查看现有促销策略时，促销策略业务逻辑处理的相关对象之间的协作。

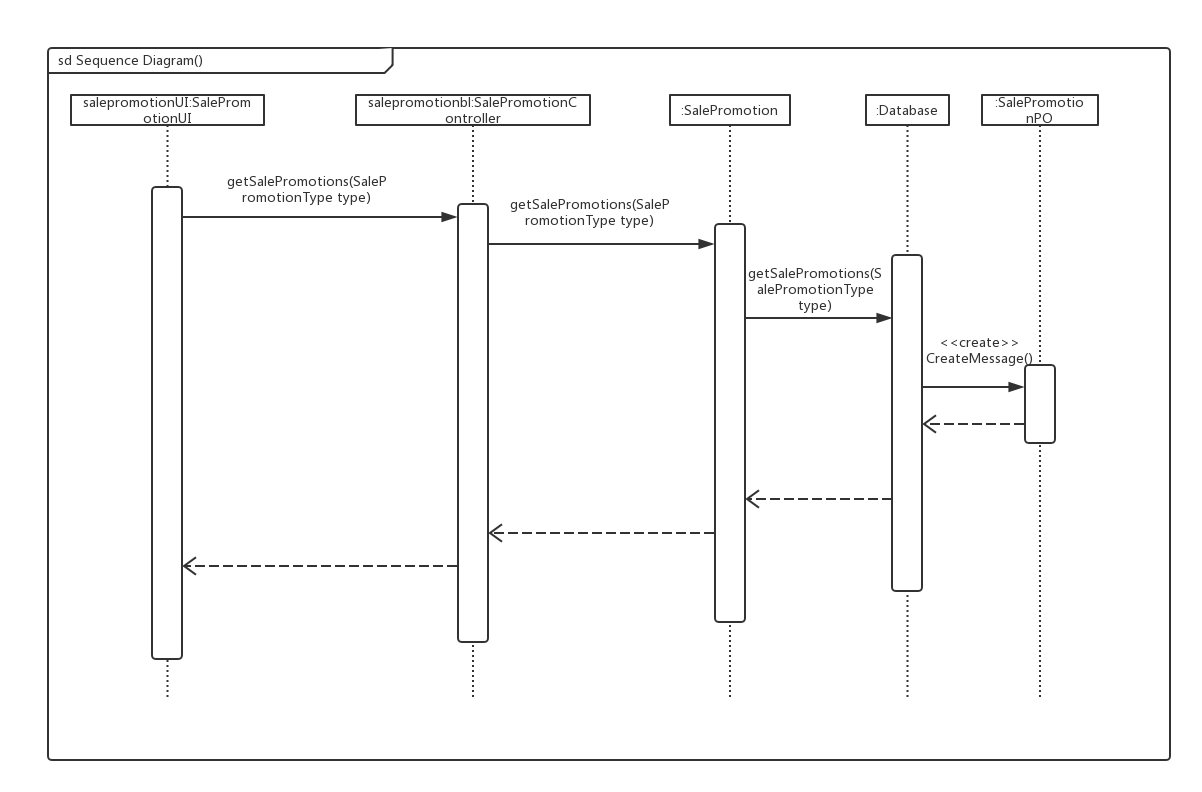


图5-1 查看促销策略

图5-2表明了酒店预订系统中，当需要添加新的促销策略时，促销策略业务逻辑处理的相关对象之间的协作。

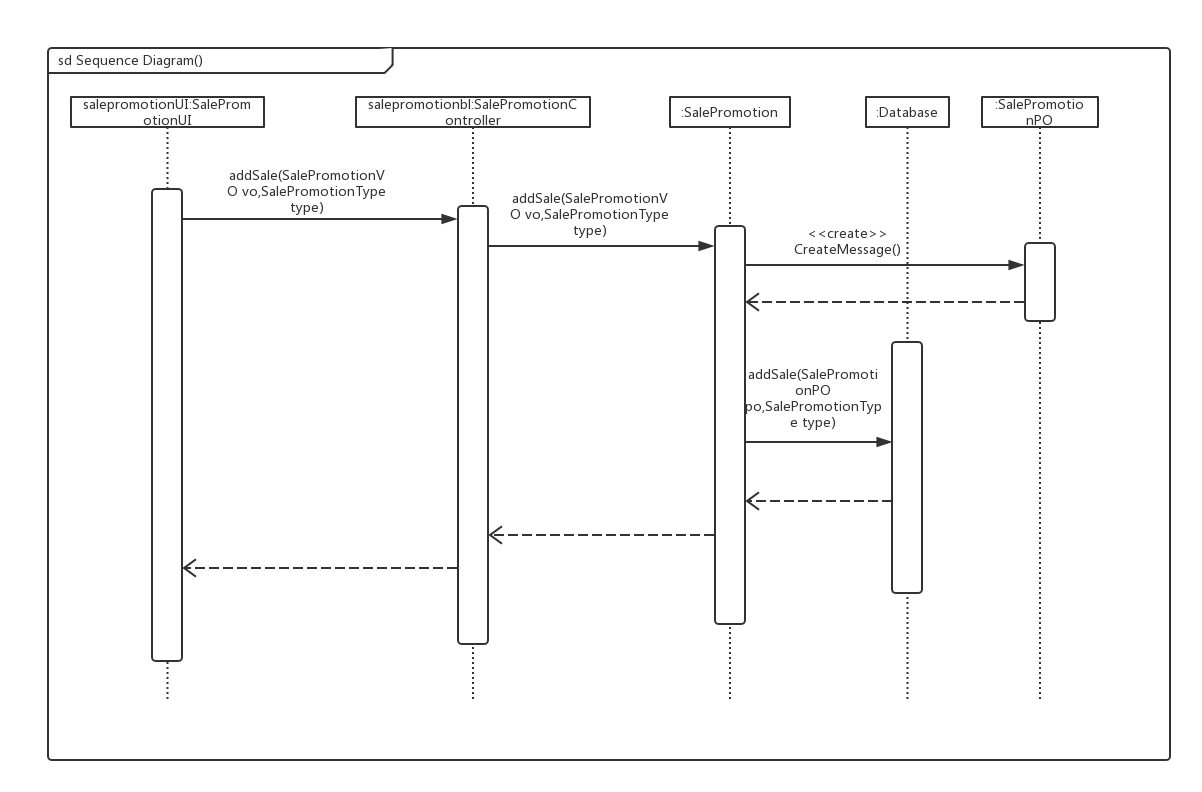


图5-2添加促销策略

图5-3表明了酒店预订系统中，当需要修改现有促销策略时，促销策略业务逻辑处理的相关对象之间的协作。

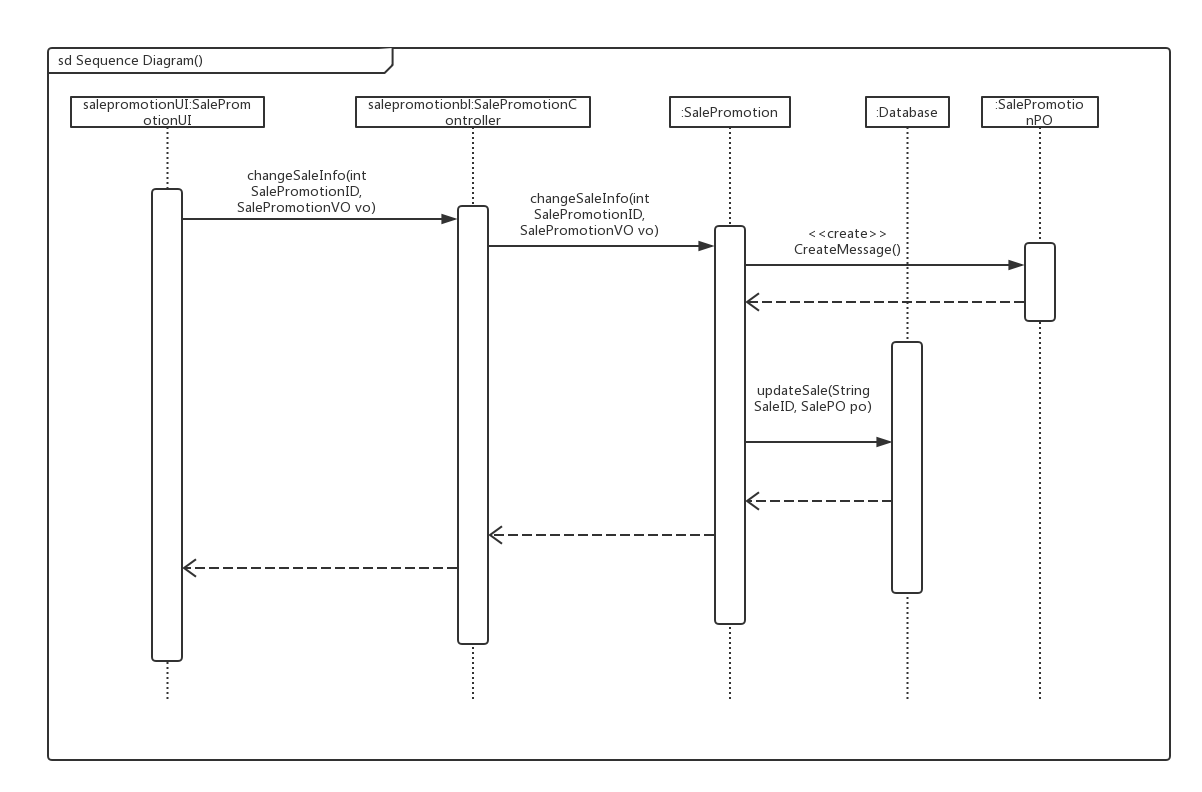


图5-3修改促销策略

图5-4表明了酒店预订系统中，当需要删除无效促销策略时，促销策略业务逻辑处理的相关对象之间的协作。

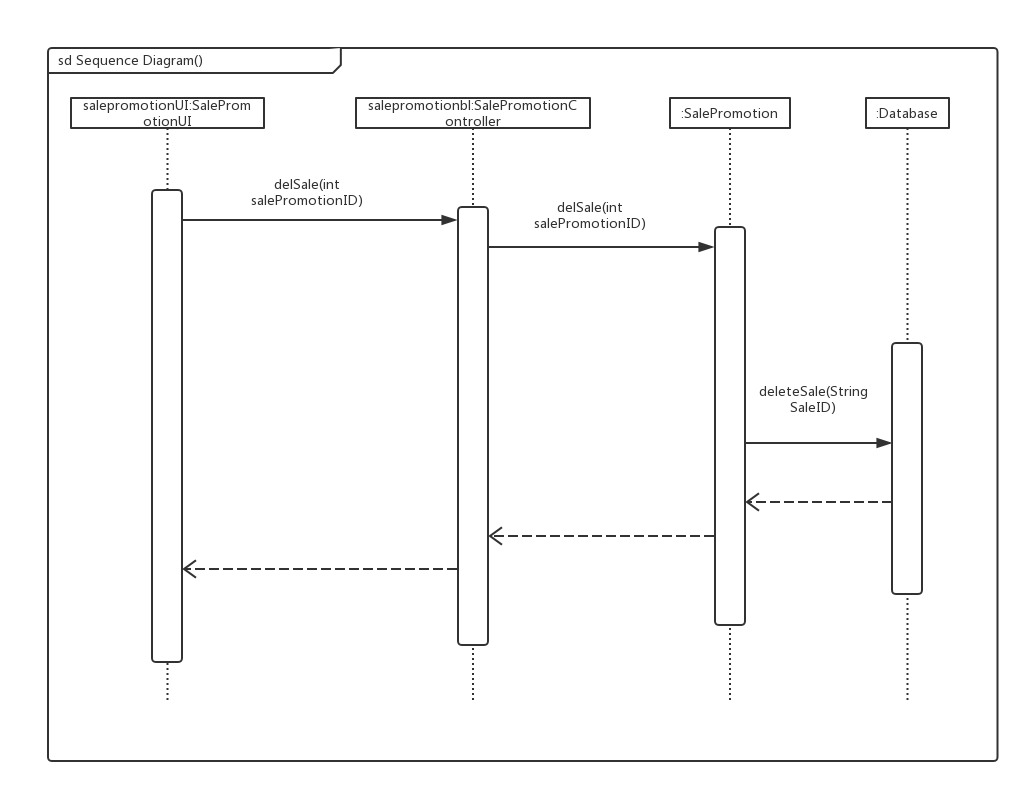


图5-4 删除促销策略

图5-5表明了酒店预订系统中，当需要查看现有等级制度时，促销策略业务逻辑处理的相关对象之间的协作。

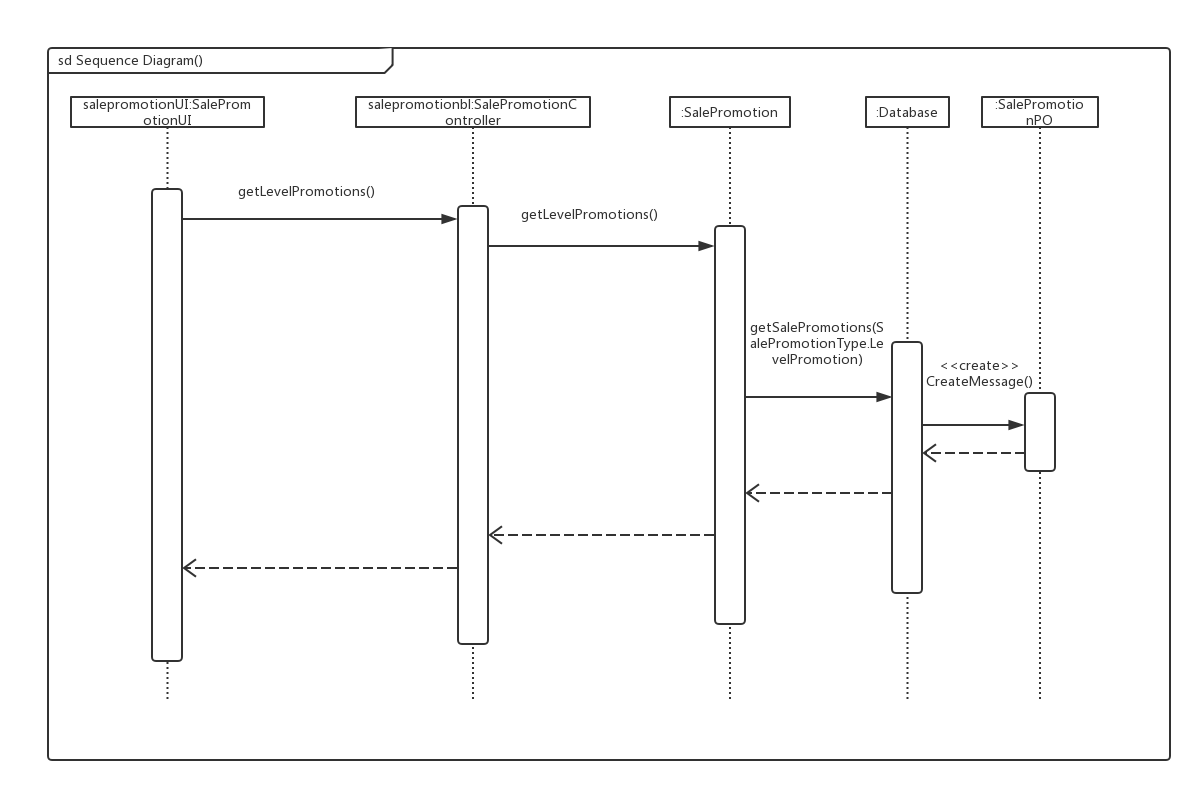


图5-5 查看等级制度

图5-6表明了酒店预订系统中，当需要管理现有等级制度时，促销策略业务逻辑处理的相关对象之间的协作。

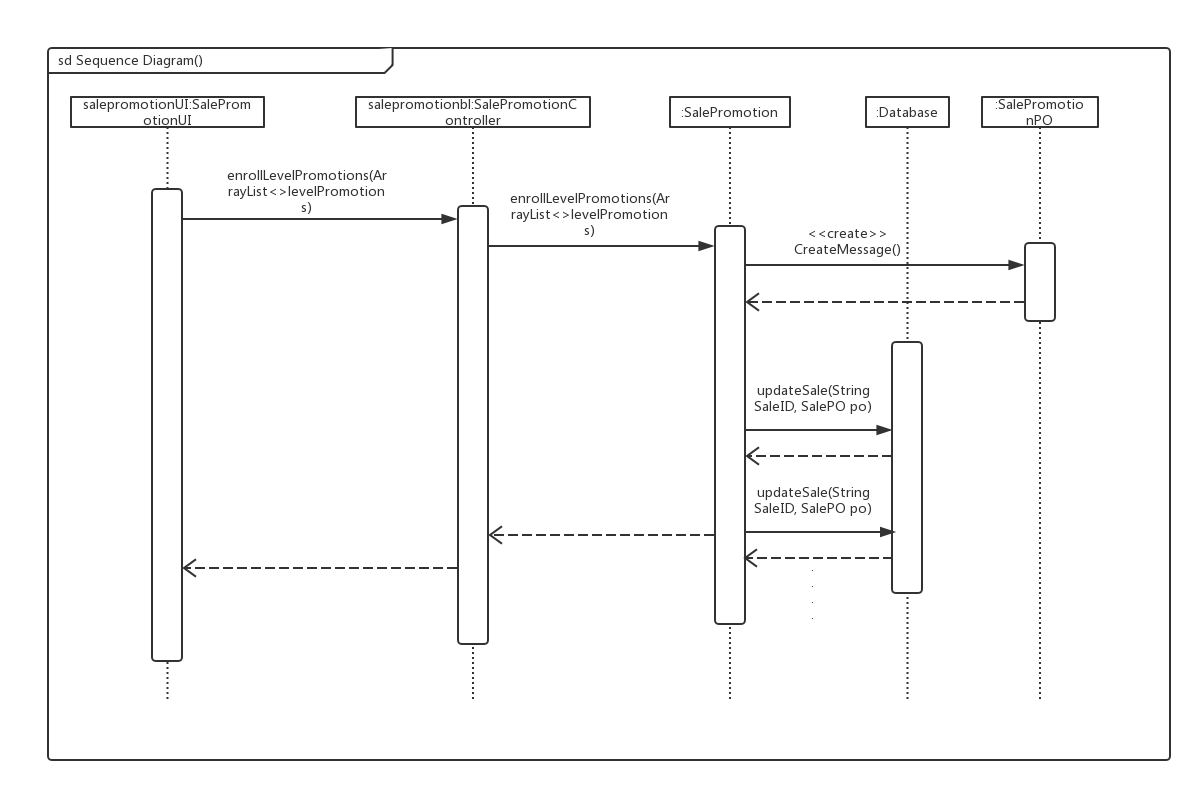
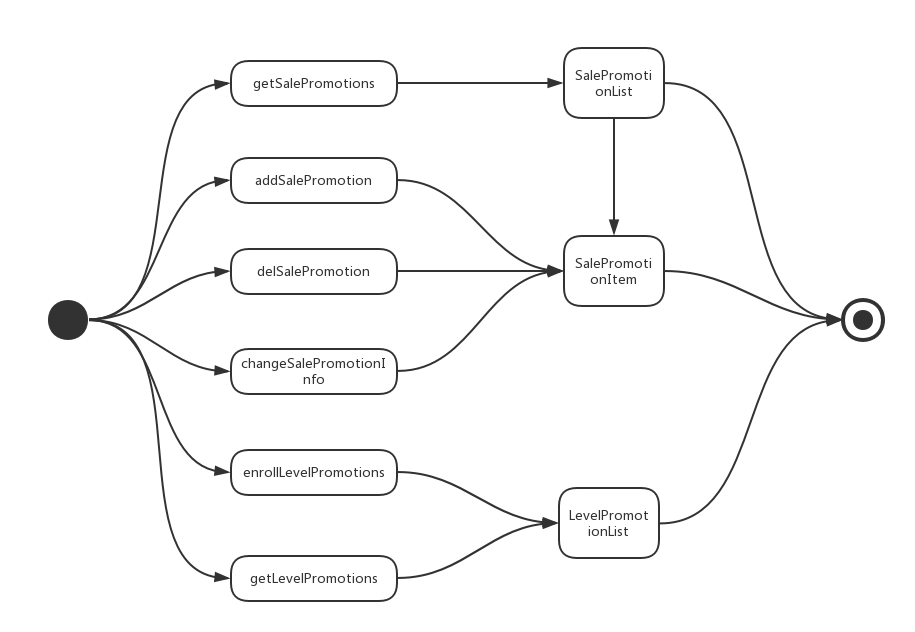


图5-6 管理等级制度

如图5-7所示的状态图描述了SalePromotion对象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由各自的控制其委托给不同的领域对象。

**4.1.6 roombl模块**

**（1）模块概述**

roombl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。

roombl模块的职责及接口参见软件体系结构描述文档表 。

**（2）整体结构**

根据体系结构的设计,我们将系统分为展现层、业务逻辑层、数据层。每一层之间为了增加灵活性,我们会添加接口。比如展示层和业务逻辑层之间,我们添加blservice.roomblservice.RoomBLService接口。业务逻辑层和数据层之间添加 dataservice.roomdataservice.RoomDataService接口。为了隔离业务逻辑职责和逻辑控制职责,我们增加了 RoomController,这样RoomController会将对审查的业务逻辑处理委托给Room对象。AvailableRoomPO是作为审查记录的持久化对象被添加到设计模型中去的。

roombl模块的设计如图6所示。

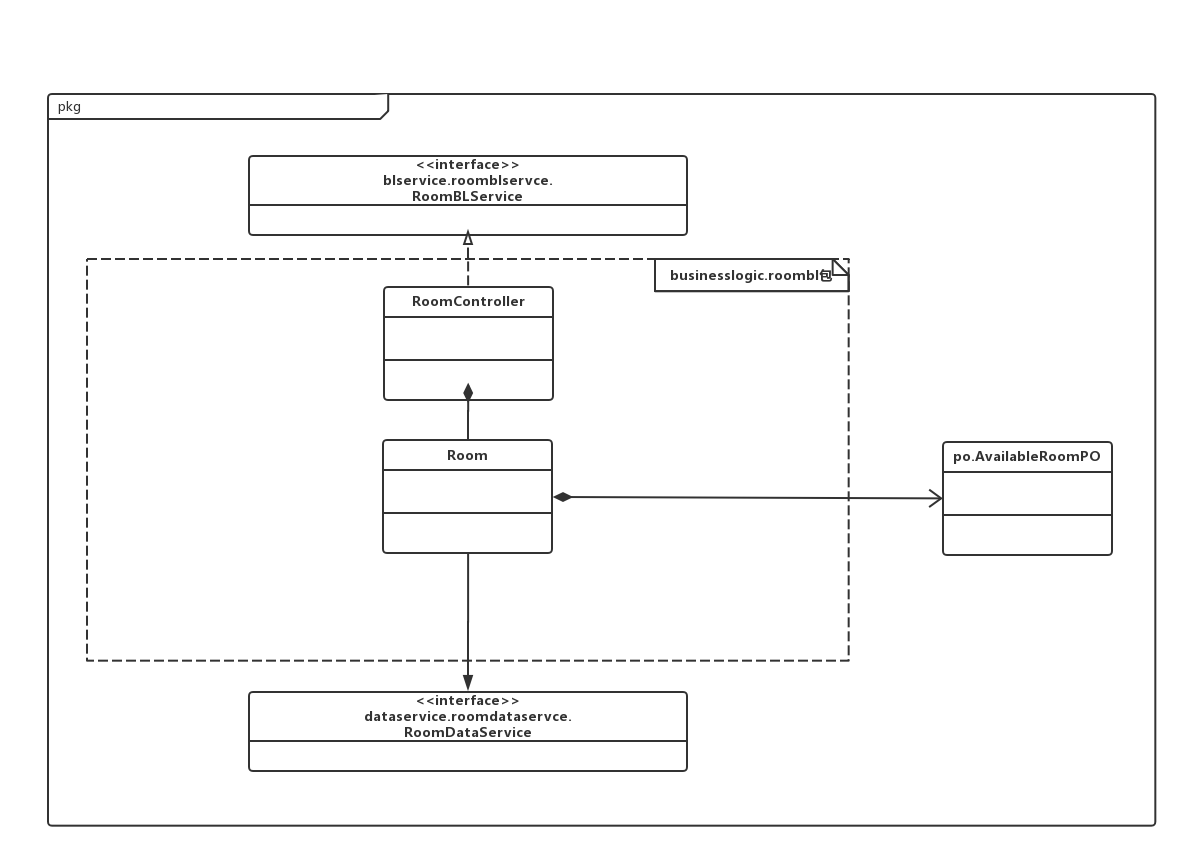


图6roombl模块各个类的设计

roombl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| RoomController | 负责实现客房管理界面所需要的服务 |
| Room | 客房管理的领域模型对象，拥有一次管理所需的客房信息，可以帮助完成客房管理界面所需要的服务 |

**（3）模块内部类的接口规范**

RoomController和Room的接口规范如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| RoomController.getAvailableRoomList | 语法 | public ArrayList<AvailableRoomVo> getAvailableRoomList(String hotelName, String startDate, String endDate) | |
| 前置条件 | 已创建一个Room领域对象 | |
| 后置条件 | 调用Room领域对象的getAvailableRoomList方法 | |
| RoomController.addAvailableRoom | 语法 | public ResultMessage addAvailableRoom(AvailableRoomVo arvo) | |
| 前置条件 | 已创建一个Room领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用Room领域对象的addAvailableRoom方法 | |
| RoomController.deleteAvailableRoom | 语法 | public ResultMessage deleteAvailableRoom(AvailableRoomVo arvo) | |
| 前置条件 | 已创建一个Room领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用Room领域对象的deleteAvailableRoom方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Room. getAvailableRoomList(String hotelName, String startDate, String endDate) | | | 得到可用客房列表 |
| Room. addAvailableRoom(AvailableRoomVo arvo) | | | 添加可用客房 |
| Room. deleteAvailableRoom(AvailableRoomVo arvo) | | | 删除可用客房 |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Room.getAvailableRoomList | 语法 | public ArrayList<AvailableRoomVo> getAvailableRoomList(String hotelName, String startDate, String endDate) | |
| 前置条件 | 输入了正确的开始时间、结束时间 | |
| 后置条件 | 返回可用客房列表 | |
| Room.addAvailableRoom | 语法 | public ResultMessage addAvailableRoom(AvailableRoomVo arvo) | |
| 前置条件 | 输入了正确的客房添加信息 | |
| 后置条件 | 增加可用客房数量，返回录入结果 | |
| Room.deleteAvailableRoom | 语法 | public ResultMessage deleteAvailableRoom(AvailableRoomVo arvo) | |
| 前置条件 | 输入了正确的客房删除信息 | |
| 后置条件 | 减少可用客房数量，返回删除结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| RoomDataService.findAvailableRooms（String hotelName, String startDate, String endDate） | | | 根据酒店名称、开始时间和结束时间查找多间可用客房 |
| RoomDataService.updateAvailableRooms（AvailableRoomPO arpo） | | | 根据输入的房间类型、数量、原始价格更新可用客房 |
| RoomDataService.insertAvailableRooms（AvailableRoomPO arpo） | | | 根据输入的房间类型、数量、原始价格更新可用客房 |

**4）业务逻辑层的动态结构**

图6-1表明了酒店预订系统中，当酒店工作人员请求查看可用客房之后，客房业务逻辑处理的相关对象之间的协作。

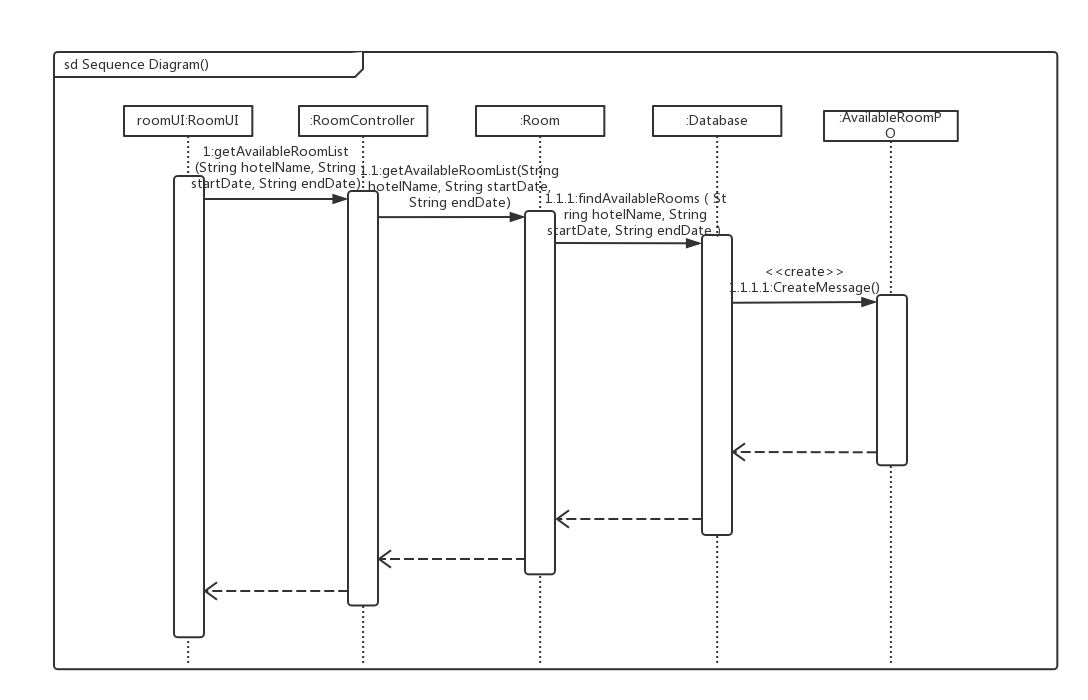


图6-1查看可用客房的顺序图

图6-2表明了酒店预订系统中，当酒店工作人员请求添加可用客房之后，客房业务逻辑处理的相关对象之间的协作。

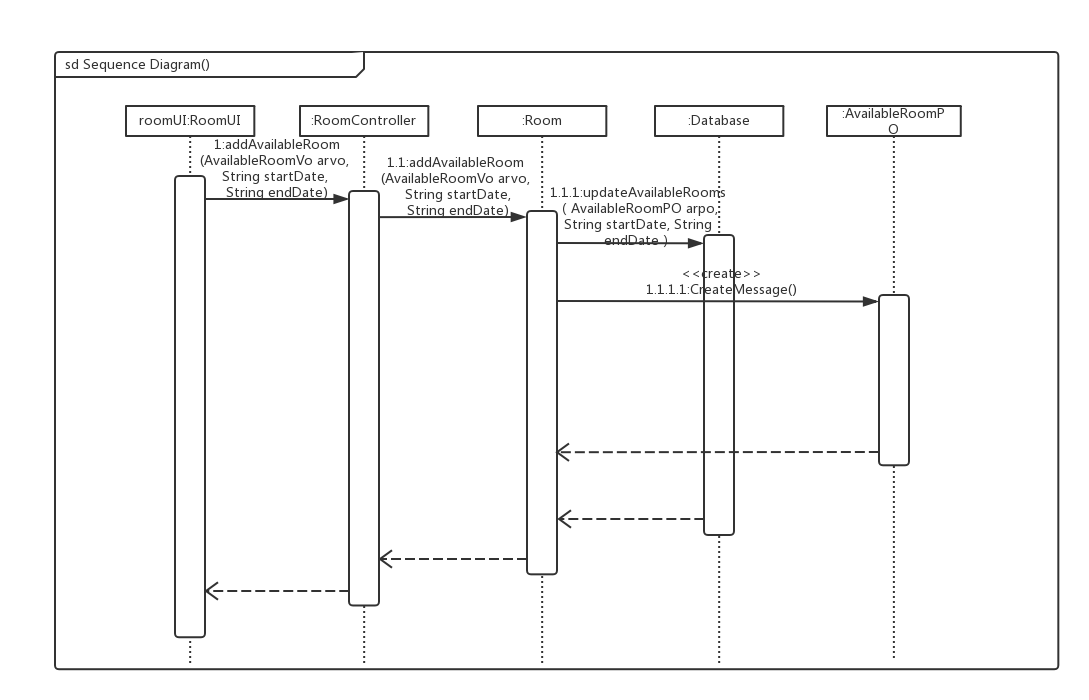


图6-2 添加可用客房的顺序图

图6-3表明了酒店预订系统中，当酒店工作人员请求删除可用客房之后，客房业务逻辑处理的相关对象之间的协作。

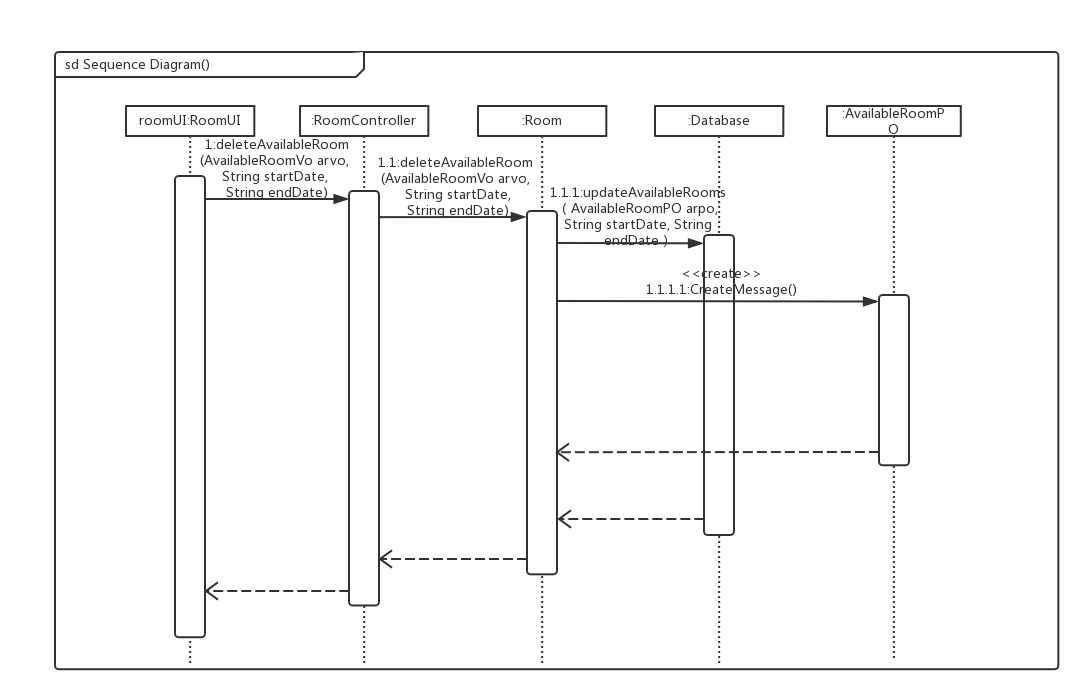


图6-3 删除可用客房的顺序图

如图6-4所示的状态图描述了Room对象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。Controller可以调用getAvailableRoomList使Room进入getAvailableRoomList状态，之后通过增加或删除可用客房进入addAvailableRooms或deleteAvailableRooms状态，调用getHotelList使Room进入getHotelList状态。

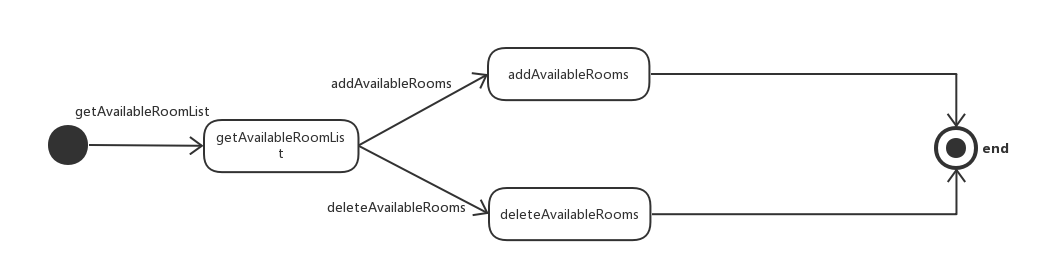


图6-4Room对象状态图

**4.1.7 creditbl模块**

**（1）模块概述**

creditbl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。

Creditbl模块的职责及接口参见软件体系结构描述文档表 。

**（2）整体结构**

根据体系结构的设计,我们将系统分为展现层、业务逻辑层、数据层。每一层之间为了增加灵活性,我们会添加接口。比如展示层和业务逻辑层之间,我们添加blservice.creditblservice.CreditBLService接口。业务逻辑层和数据层之间添加 dataservice.creditdataservice.creditDataService接口。为了隔离业务逻辑职责和逻辑控制职责,我们增加了 CreditController,这样CreditController会将对审查的业务逻辑处理委托给Credit对象。CreditPO是作为审查记录的持久化对象被添加到设计模型中去的。而 CalculateLevel封装了根据会员信用和等级制度信息计算会员等级的职责。

Creditbl模块的设计如图1所示。

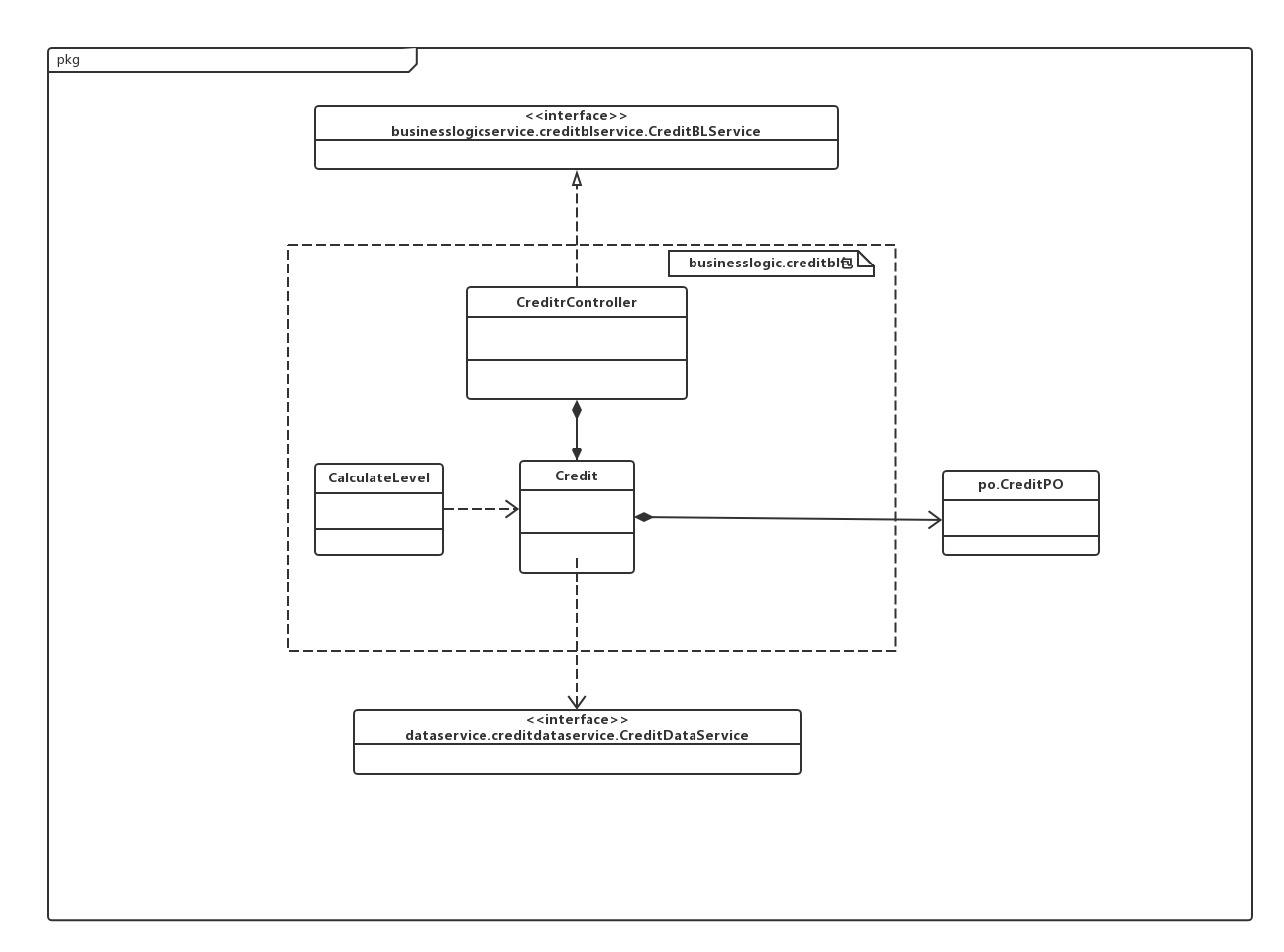


图1 creditbl模块各个类的设计

Creditbl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| CreditController | 负责实现会员信用界面所需要的服务 |
| Credit | 会员信用的领域模型对象，拥有会员信用值、每一次信用变化时间及变化数目，可以帮助完成会员管理界面所需要的服务 |

**（3）模块内部类的接口规范**

CreditController和Credit的接口规范如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CreditController.increaseCredit | 语法 | public void increaseCredit(String userName, int creditChange); | |
| 前置条件 | 信用充值 | |
| 后置条件 | 调用credit的increaseCredit | |
| Credit.getCustomerCreditInfo | 语法 | public List<CreditVO> getCustomerCreditInfo(String userName); | |
| 前置条件 | 已经创建Credit领域对象 | |
| 后置条件 | 调用credit的getCustomerCreditInfo | |
| CreditController.updateCustomerCreditInfo | 语法 | public ResultMessage updateCustomerCreditInfo(String userName, CreditVO creditVO, OrderVO orderVO); | |
| 前置条件 | 已创建一个Customer领域对象，订单状态已改变 | |
| 后置条件 | 调用Customer领域对象的updateCustomerCreditInfo 方法 | |
| Credit.getNumCredit | 语法 | public double getNumCredit(String userName); | |
| 前置条件 | 已经创建Credit领域对象 | |
| 后置条件 | 调用Credit领域对象的getNumCredit方法 | |
| 需要的服务（需接口） | | | |
| Credit.increseCredit(String userName, int creditChange); | | | 增加信用值 |
| Credit.getCustomerCreditInfo(String userName) | | | 获取会员信用值变化记录 |
| Credit.updateCustomerCreditInfo(String userName, CreditVO creditVO, OrderVO orderVO) | | | 更新会员的信用值 |
| Credit. getNumCredit(String userName) | | | 获取会员信用值 |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Credit.getCustomerCreditInfo | 语法 | public List<CreditVO> getCustomerCreditInfo(String userName); | |
| 前置条件 | 启动查看信用值变化 | |
| 后置条件 | 返回信用值变化记录 | |
| Credit.getNumCredit | 语法 | public double getNumCredit(String userName); | |
| 前置条件 | 查看信用值 | |
| 后置条件 | 系统显示信用值 | |
| Credit.increaseCredit | 语法 | public void increaseCredit(String userName, int creditChange); | |
| 前置条件 | 信用充值 | |
| 后置条件 | 系统更新信用值变化 | |
| Credit.updateCustomerCreditInfo | 语法 | public ResultMessage updateCustomerCreditInfo(String userName, CreditVO creditVO, OrderVO orderVO); | |
| 前置条件 | 更新信用值信息（订单执行、订单异常、订单撤销） | |
| 后置条件 | 数据层更新信用值 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| CreditDataService.getCustomerCredits(String userName) | | | 获取用户信用值变化的记录 |
| CreditDataService.insert(CreditInfoPO creditInfoPO) | | | 在数据库中插入新的信用值记录 |

**（4）业务逻辑层的动态结构**

图1-1表明了酒店预订系统中，当会员注册之后，会员信用业务逻辑处理的相关对象之间的协作。

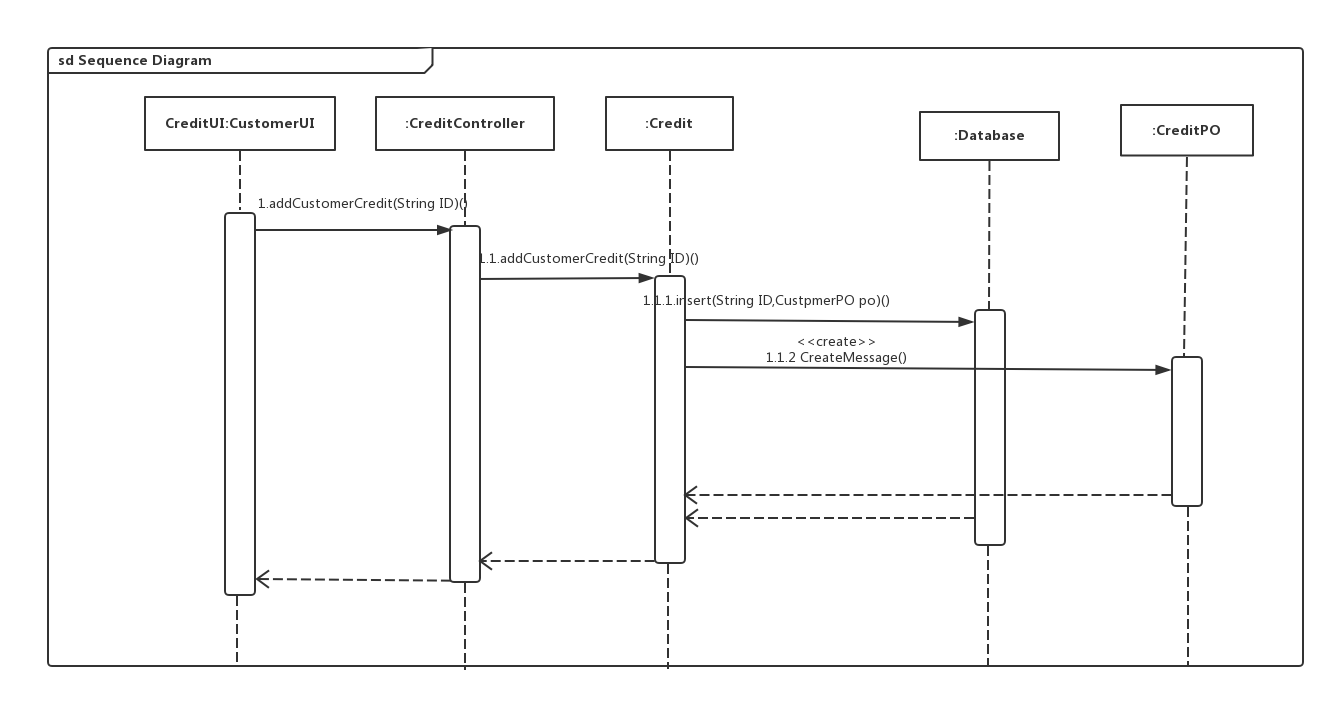


图1-1 增加一个会员信用记录顺序图

图1-2表明了酒店预订系统中，当会员或网站管理人员查看会员信用之后，信用业务逻辑处理的相关对象之间的协作。

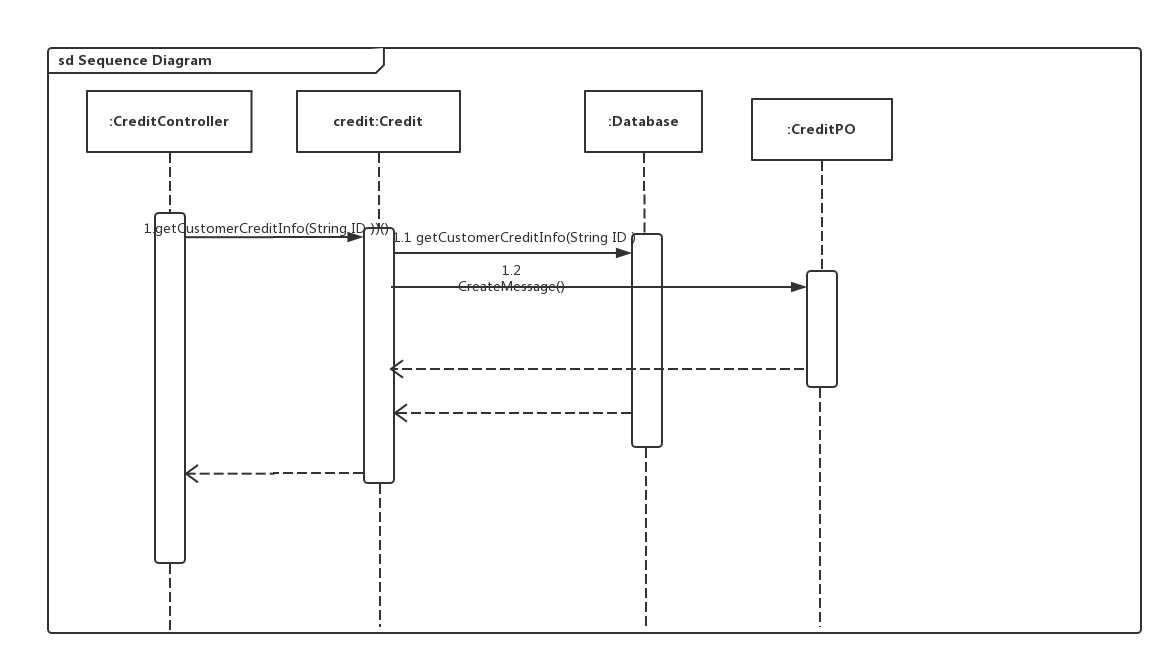


图1-2 查看会员信用的顺序图

图1-3表明了酒店预订系统中，当网站管理人员修改会员信用之后，信用业务逻辑处理的相关对象之间的协作。

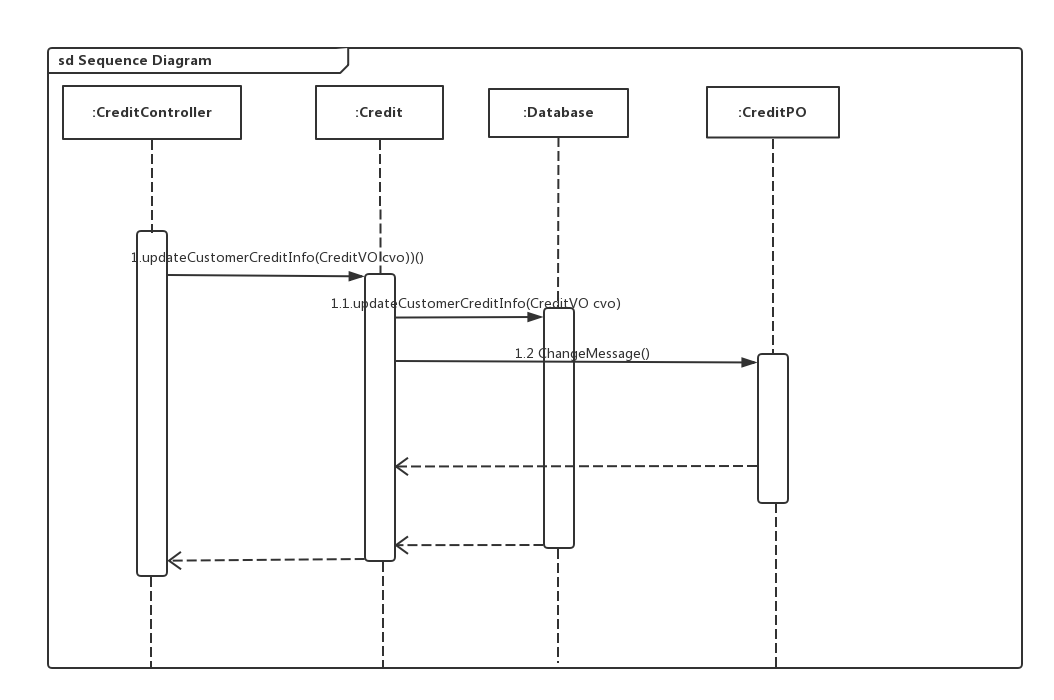


图1-3 修改会员信用的顺序图

图1-4表明了酒店预订系统中，当生成订单查看会员等级时，信用业务逻辑处理的相关对象之间的协作。

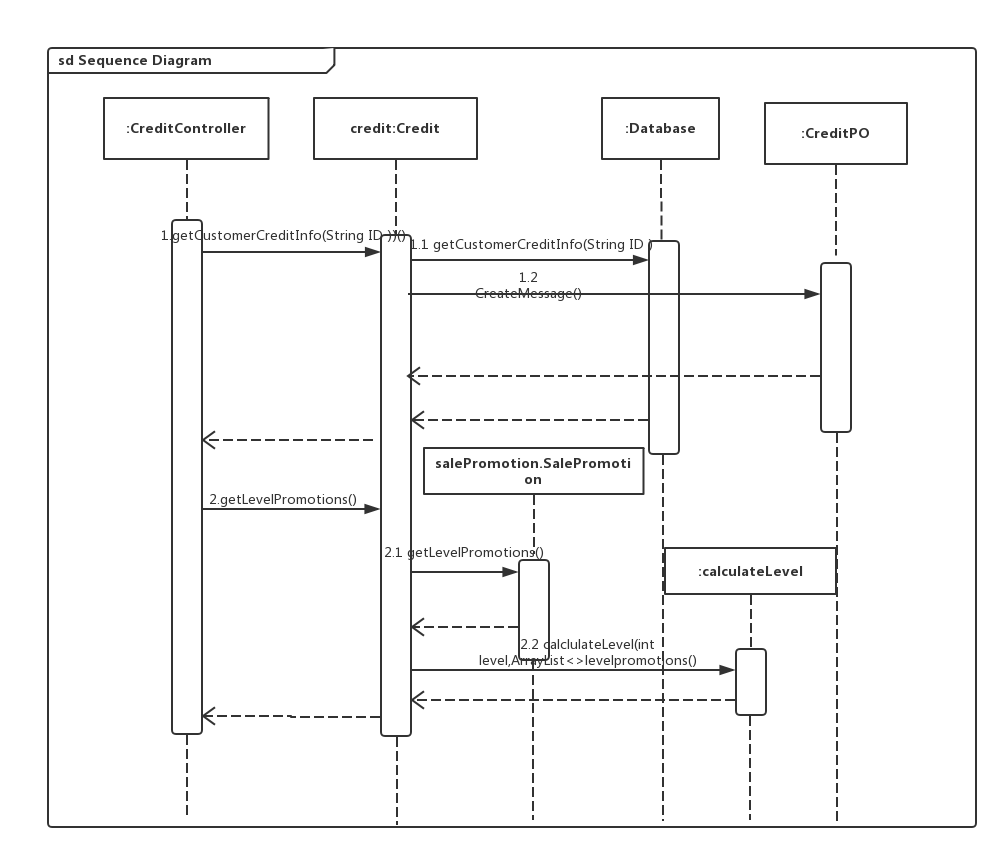


图1-4 查看会员等级的顺序图

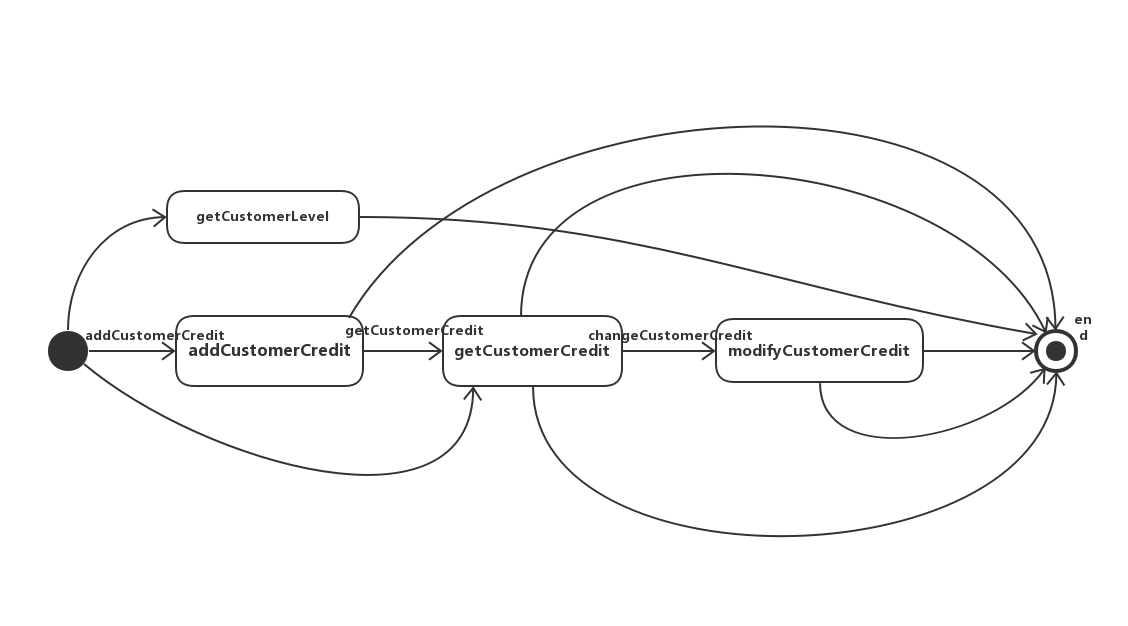
如图1-5所示的状态图描述了Credit对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着addCustomerCredit方法被调用，Credit进入addCustomerCredit状态；之后通过查看会员信用进入getCustomerCredit状态，通过修改会员信息进入modifyCustomerCredit状态。也可以被调用getCustomerIevel，使Credit进入getCustomerLevel状态. 

图1-5 Credit对象状态图

**4.1.8 orderbl模块**

**（1）模块概述**

orderbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

orderbl模块的职责及接口参见软件体系结构描述文档表 。

**（2）整体结构**

根据体系结构的设计,我们将系统分为展现层、业务逻辑层、数据层。每一层之间为了增加灵活性,我们会添加接口。比如展示层和业务逻辑层之间,我们添加blservice.orderblservice.OrderBLService接口。业务逻辑层和数据层之间添加 dataservice.orderdataservice.OrderDataService接口。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会将对订单的业务逻辑处理委托给Order对象。OrderPO是作为订单的持久化对象被添加到设计模型中去的。而OrderList和orderuLineItem的添加是List的容器类。OrderLineItem保有订单编号、酒店名、会员名和价格等数据,及生成、执行、撤销的职责。而 OrderList封装了关于OrderLineItem的数据集合的数据结构的秘密和显示订单列表的职责。CustomerCrededit是根据依赖倒置原则，为了消除循环依赖而产生的接口。

hotelbl模块的设计如图4所示。

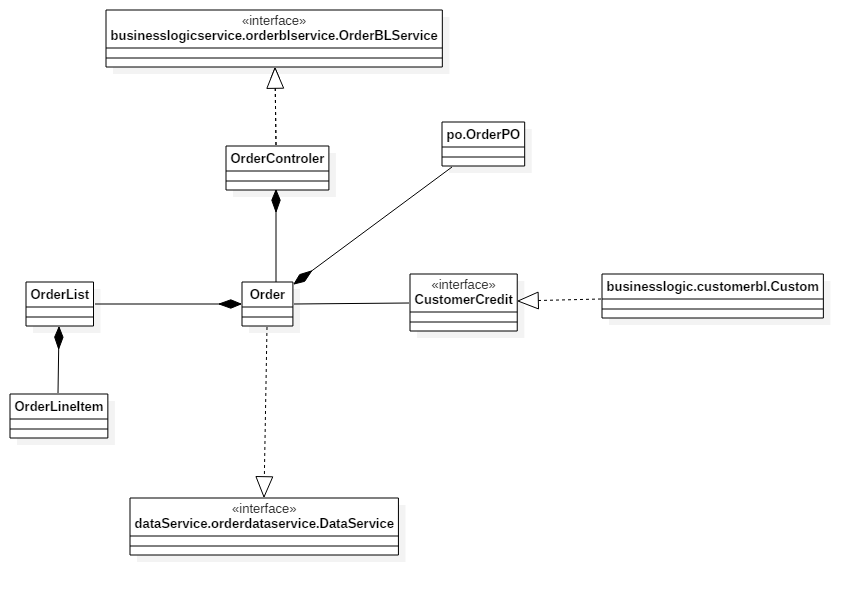


图4 orderbl模块各个类的设计

orderbl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| Ordercontroller | 负责分发订单处理界面所需要的服务 |
| Order | 订单处理的领域模型对象，拥有订单处理所需要的订单信息、酒店信息、会员信息等等，可以帮助完成订单管理界面所需要的服务 |

**（3）模块内部类的接口规范**

orderController和Order的接口规范如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrderController.newOrder | 语法 | public ResultMessage newOrder(OrderVO vo); | |
| 前置条件 | 输入符合规则 | |
| 后置条件 | 调用Order领域对象newOrder方法 | |
| OrderController.getlOrderList | 语法 | public Map<Integer,OrderVO> getOrderVOList(); | |
| 前置条件 | 用户或酒店查看订单列表 | |
| 后置条件 | 调用Order领域对象的getOrderList方法 | |
| OrderController.getOrderEvaluationVOlist | 语法 | public Map<Integer,OrderEvaluationVO> getOrderEvaluationVOList(); | |
| 前置条件 | 用户或者酒店工作人员要查看自己的订单评价列表 | |
| 后置条件 | 调用order领域对象的getOrderEvaluationVOList方法 | |
| OrderController.getOrderInfo | 语法 | public OrderVO getOrderInfo(String OrderID); | |
| 前置条件 | 已经通过调用get\*\*Orderlist方法得到orderList | |
| 后置条件 | 调用Order领域对象的getOrderInfo方法 | |
| OrderController.evaluateOrder | 语法 | public ResultMessage evaluateOrder(OrderEvaluationVO oevo); | |
| 前置条件 | 已经生成一个Order对象 | |
| 后置条件 | 调用Order领域对象的evaluateOrder方法 | |
| OrderController.executeOrder合并执行未执行订单和异常订单。想法，可以利用继承创造出执行订单的多态 | 语法 | public ResuMessage excuteOrder(String OrderID) | |
| 前置条件 | 已生成一个Order对象 | |
| 后置条件 | 调用OrderItem对象的excuteOrder() | |
| OrderController.cancelOrder  同上 | 语法 | public ResultMessage cancelOrder(String OrderID); | |
| 前置条件 | 已经创建一个Order对象 | |
| 后置条件 | 调用Order的cancelOrder方法 | |
| OrderController.getOrderEvaluationByID | 语法 | public OrderEvaluationVO getOrderEvaluationByID(int OrderID); | |
| 前置条件 | 查看订单评价 | |
| 后置条件 | 调用order的getOrderEvaluationByID方法 | |
| Order.getOrderListType | 语法 | public Map<Integer,OrderVO> getOrderListByType(OrderType type); | |
| 前置条件 | 获取特定类型的订单列表 | |
| 后置条件 | 调用order的getOrderListByType | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Order.newOrder(OrderVO ovo) | | | 新增一个Order对象 |
| Order. getOrderVOList(); | | | 获取订单列表 |
| Order.getOrderEvaluationVOList(); | | | 获取订单评价列表 |
| Order.getOrderInfo(String OrderID); | | | 根据订单编号显示订单信息 |
| Order.evaluateOrder(OrderEvaluationVO oevo); | | | 评价订单 |
| Order.excuteOrder(String OrderID) | | | 执行订单 |
| Order.cancelOrder(String OrderID) | | | 取消订单 |
| Order.getOrderEvaluationByID(int OrderID); | | | 根据订单编号获取订单评价 |
| Order.getOrderListByType(OrderType type); | | | 获取特定类型的订单 |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Order.newOrder | 语法 | public ResultMessage newOrder(OrderVO ovo); | |
| 前置条件 | 启动生成新订单 | |
| 后置条件 | 持久化生成新订单设计的领域对象的信息，并且显示订单价格和订单信息 | |
| Order.getOrderEvaluationVOlist | 语法 | public Map<Integer,OrderEvaluationVO> getOrderEvaluationVOList(); | |
| 前置条件 | 用户或者酒店工作人员要查看自己的订单评价列表 | |
| 后置条件 | 系统显示订单评价列表 | |
| Order.getrOrderList | 语法 | public Map<Integer,OrderVO> getOrderVOList(); | |
| 前置条件 | 用户或酒店工作人员获取订单列表 | |
| 后置条件 | 显示订单列表 | |
| Order.getOrderInfo | 语法 | public OrderVO getOrderInfo(int orderID); | |
| 前置条件 | 查看特定订单信息 | |
| 后置条件 | 显示订单信息 | |
| Order.evaluateOrder | 语法 | public ResultMessage evaluateOrder(OrderEvaluationVO oevo); | |
| 前置条件 | 启动订单评价 | |
| 后置条件 | 数据层更新订单评价 | |
| Order.executeOrder | 语法 | public boolean executeOrder(int orderID); | |
| 前置条件 | 执行订单 | |
| 后置条件 | 修改用户的信用值，数据层更新订单信息和用户信用值 | |
| Order.cancelOrder | 语法 | public ResultMessage cancelOrder(int OrderID); | |
| 前置条件 | 已经创建一个OrderItem对象 | |
| 后置条件 | 更新订单信息，改变用户信用值 | |
| Order.getOrderListType | 语法 | public Map<Integer,OrderVO> getOrderListByType(OrderType type); | |
| 前置条件 | 获取特定类型的订单列表 | |
| 后置条件 | 系统显示订单列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| CreditBLService.getNumCredit(String userName) | | | 获取会员信用值 |
| RoomBLservice.getAvailableRoomList(String hotelName, String startDate, String endDate) | | | 获取酒店可用客房列表 |
| CustomerBLService.updateCustomerCreditInfo(String userName, CreditVO creditVO, OrderVO orderVO) | | | 更新用户信用值 |
| OrderDataService.updateOrderEvaluation(OrderEvaluationPO oepo | | | 更新订单评价 |
| OrderDataService.updateOrder(OrderPO opo); | | | 更新订单信息 |
| OrderDataService.deleteOrder(OrderPO opo); | | | 取消订单 |
| CreditBLService.getNumCredit(String userName); | | | 获取用户信用值 |
| CreditBLService.updateCustomerCreditInfo(String userName, CreditVO creditVO, OrderVO orderVO)) | | | 更新用户信用值 |
| OrderDataService.insertOrder(OrderPO opo) | | | 在数据层中增加订单 |

**4）业务逻辑层的动态结构**

如图表明了在酒店预订系统中，当用户输入了订单信息后，生成订单逻辑的相关对象之间的协作。

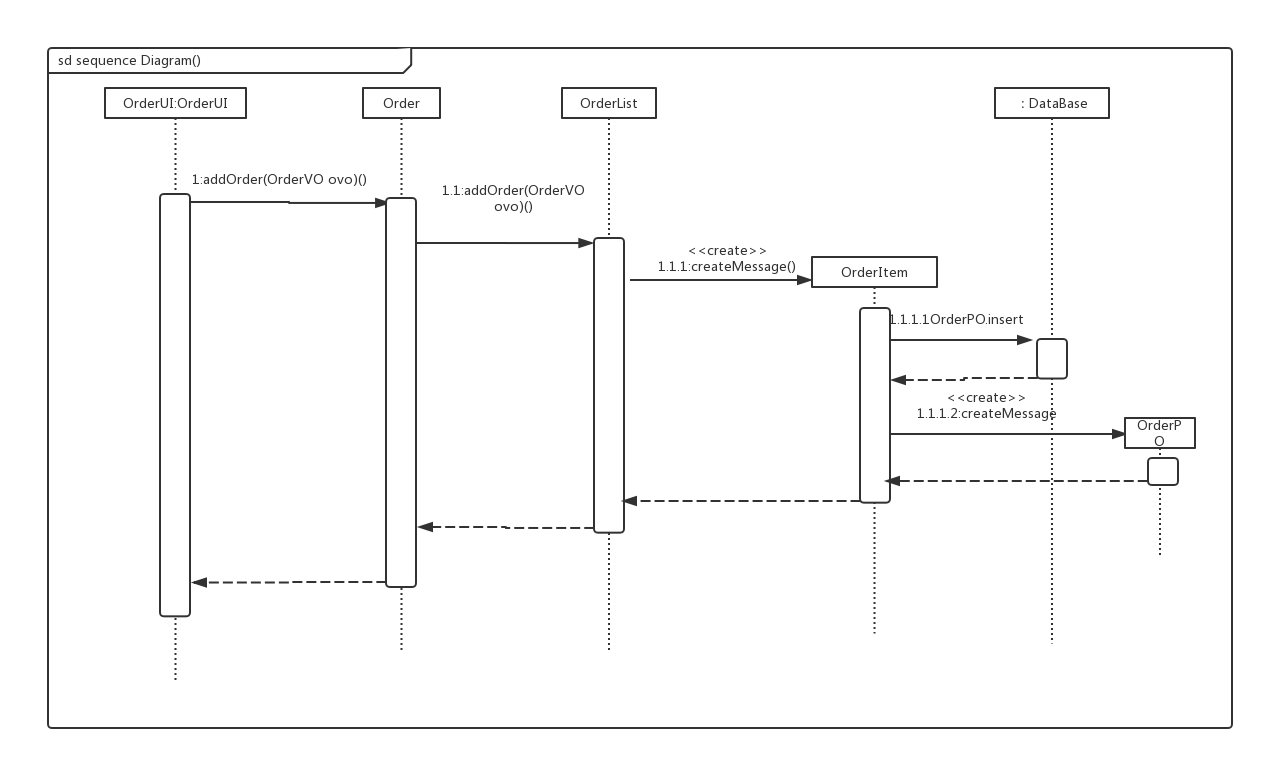


图4-1生成订单的顺序图

如图表明了在酒店预订系统中，获取会员订单列表逻辑的各相关对象之间的协作。

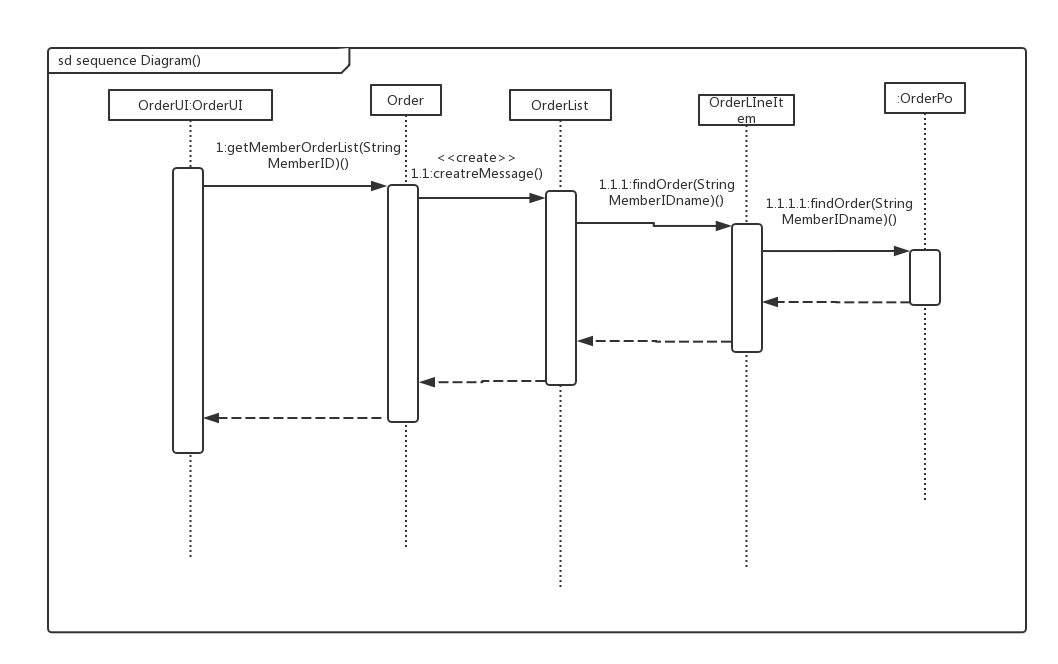


图4-2获取会员订单列表的顺序图

如图表明在酒店预订系统中，获取酒店订单列表相关对象之间的协作

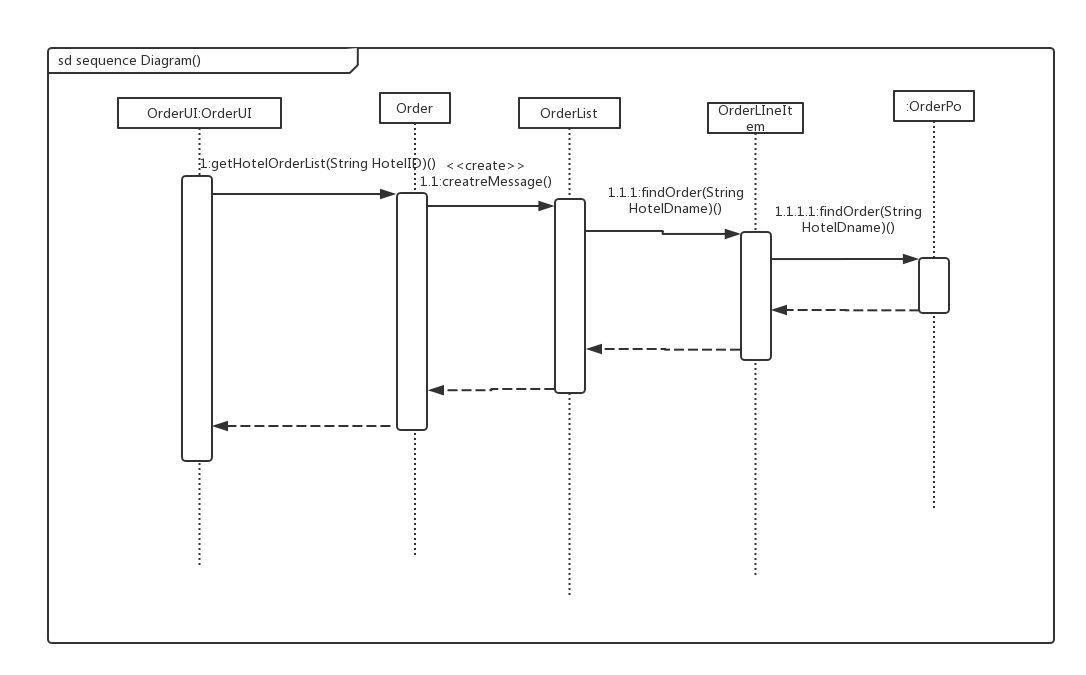


图4-3获取酒店订单列表的相关对象之间的协作

如图表明在酒店预订系统中，获取订单信息相关对象之间的协作

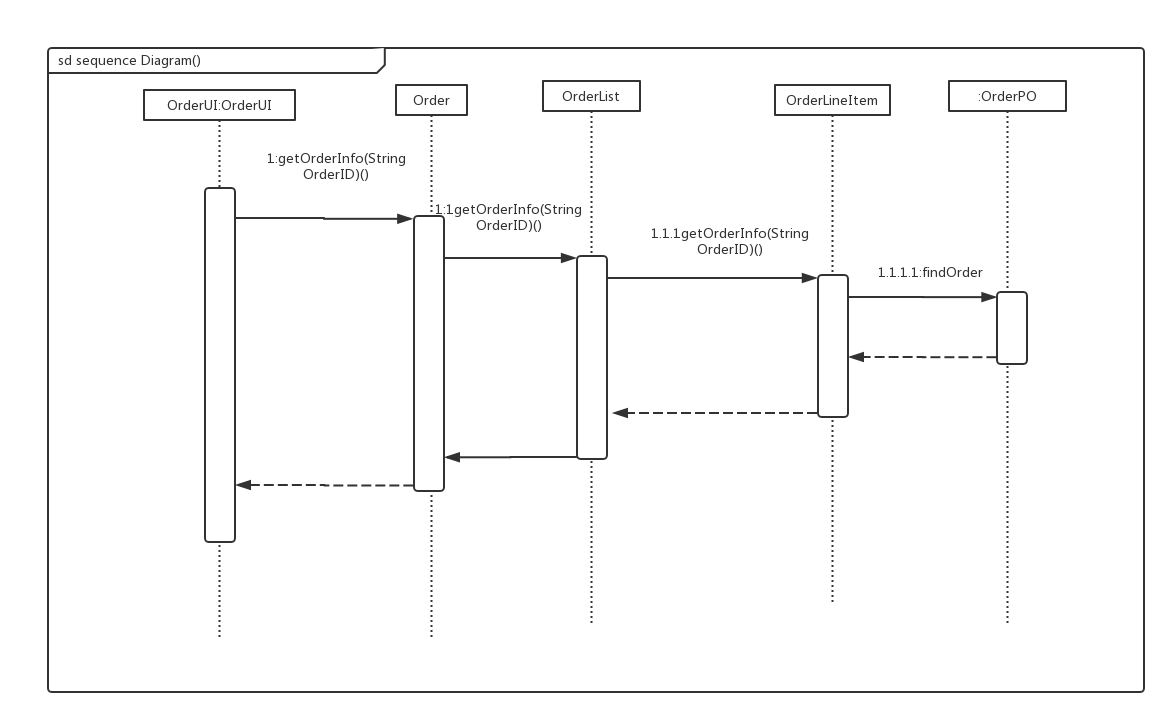


图4-4获取订单信息的顺序图

如图表明在酒店预订系统中，评价订单任务中各相关对象之间的协作

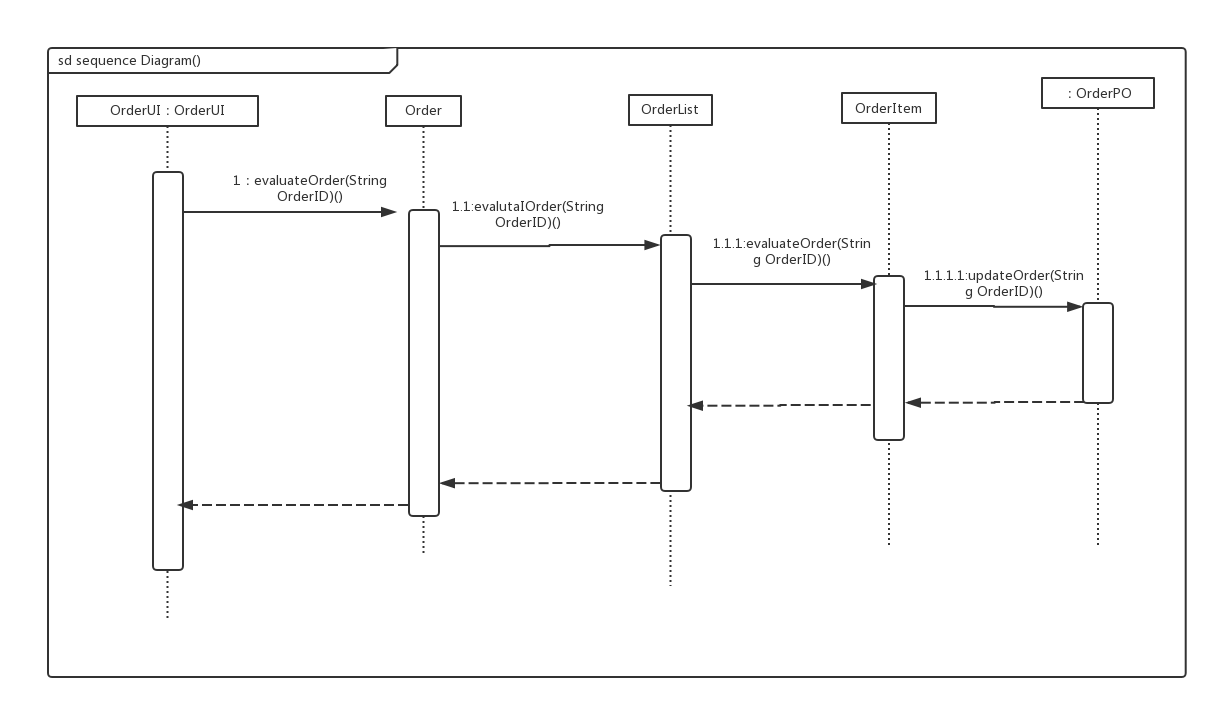


图5-5评价订单的顺序图

如图表明在酒店预订系统中，取消订单中相关对象的协作

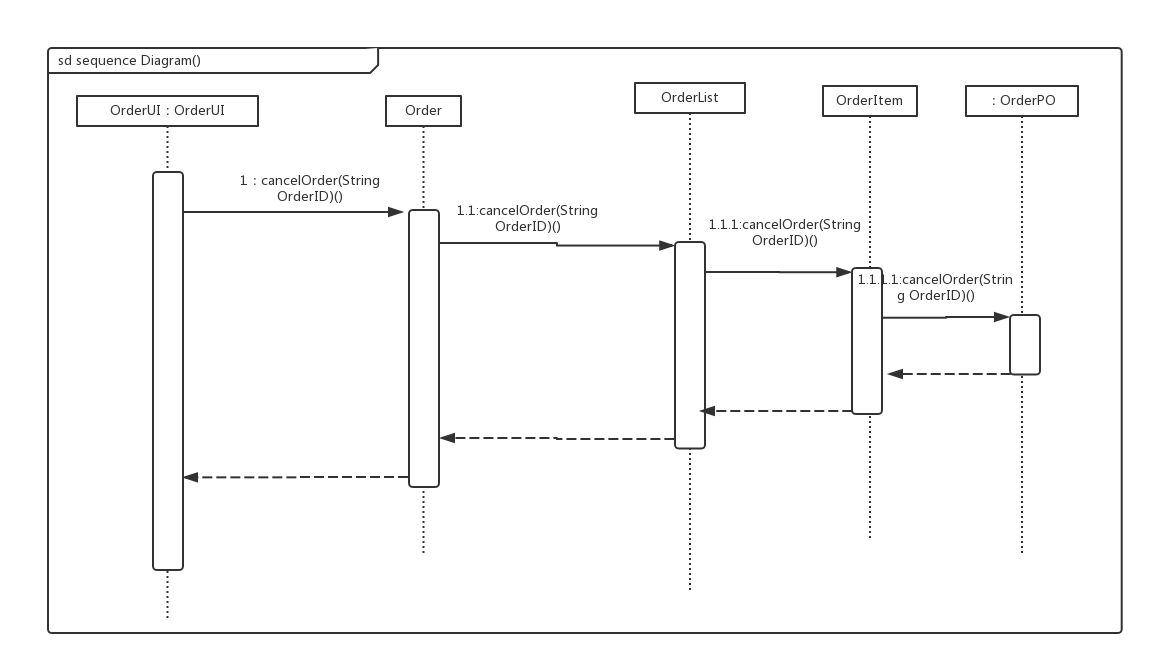


图5-6取消订单的顺序图

如图表明在酒店预订系统中，执行订单各对象之间的协作

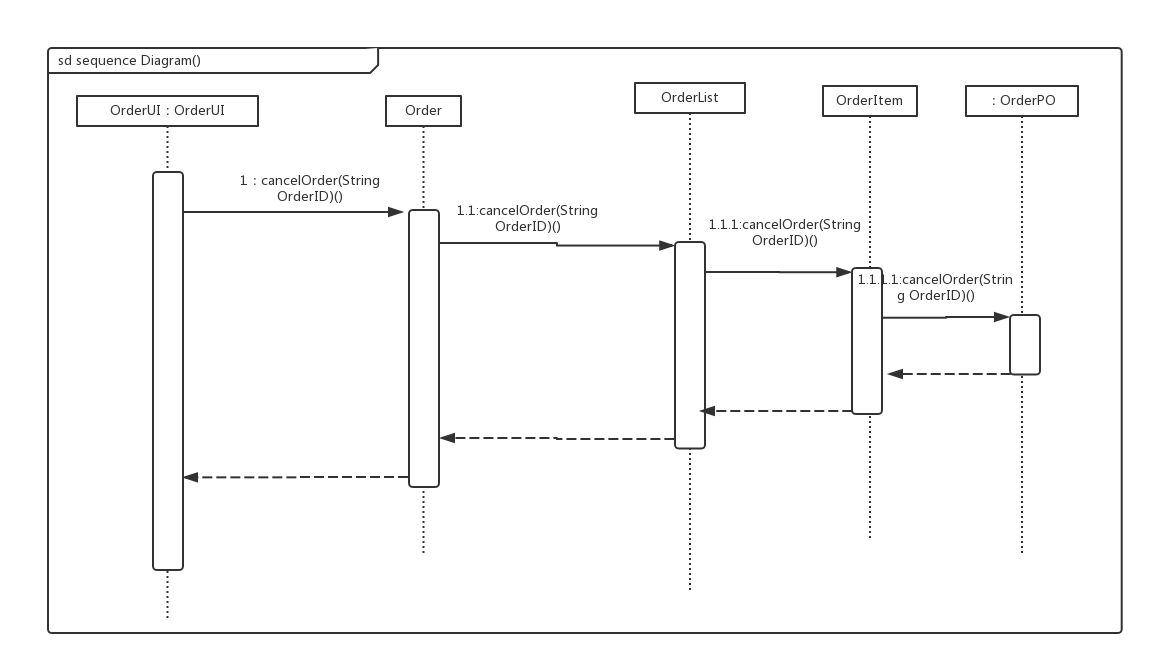
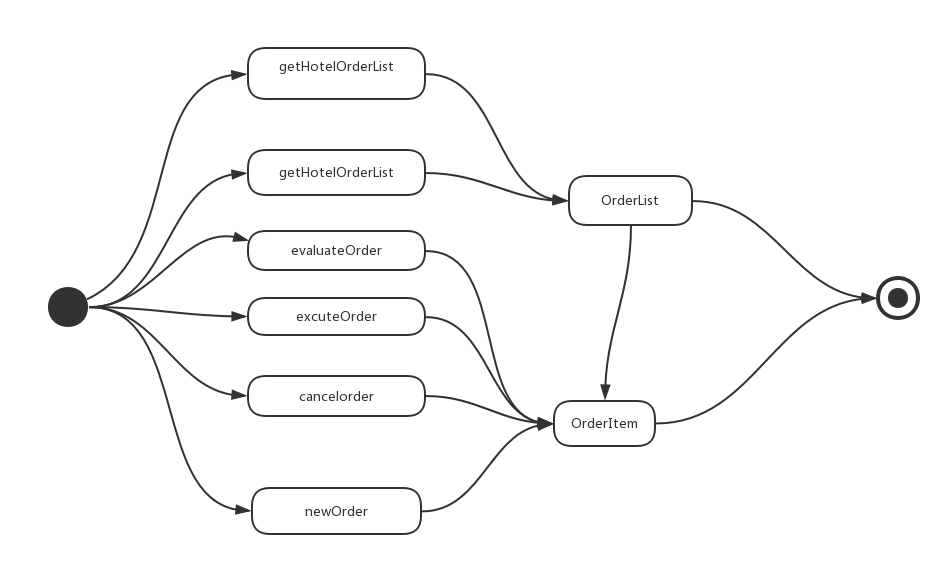


图5-6执行订单的顺序图

如图所示的状态图描述了order对象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由各自的控制其委托给不同的领域对象。

**（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其委托给不同的领域对象。

**4.2数据层的分解**

**4.2.1数据层的分解**

数据层主要给业务逻辑层提供数据防伪服务，包括对于持久化数据的增、删、改、查。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，所示抽象了数据服务。数据层模块的具体描述如图4.2-1所示。

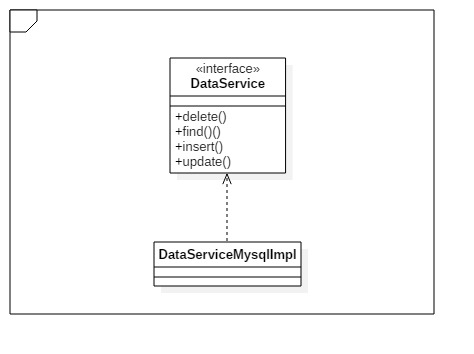


图4.2-1数据层模块的描述

**4.2.2数据层模块的职责**

数据层模块的职责如表4.2-2所示。

表4.2-2 数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| DataService | 持久化数据库的接口，提供集体载入，集体保存、增、删、改、查服务。 |
| DataServiceMysqlImpl | 基于mysql的持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |

**4.2.3数据层模块的接口规范**

表4.2.3-1 数据层Order模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderDataService.getOrderInfo(int orderID) | 语法 | public OrderPO getOrderInfo(int orderID) throws RemoteException; |
| 前置条件 | 启动查看订单详情 |
| 后置条件 | 按ID进行查找返回相应的OrderPO结果 |
| OrderDataService.getCustomerOrderList(String customerID) | 语法 | public ArrayList<CustomerOrderVO> getCustomerOrderList(String customerID) |
| 前置条件 | 无 |
| 后置条件 | 显示用户订单列表 |
| OrderDataService.getHotelOrderList(String hotel) | 语法 | public Map<Integer, OrderPO> getHotelOrderList(String hotel) throws RemoteException; |
| 前置条件 |  |
| 后置条件 | 显示酒店订单列表 |
| OrderDataService.getUnexcutedOrderList | 语法 | public ArrayList<OrderPO> getUnexcutedOrderList() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 获取未执行订单列表 |
| OrderDataService.getAbnormalOrderList() | 语法 | public ArrayList<OrderPO> getAbnormalOrderList() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 获取一场订单列表 |
| OrderDataService.getOrderEvaluation(int orderID) | 语法 | public OrderEvaluationPO getOrderEvaluation(int orderID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 获取订单评价PO |
| OrderDataService.updateOrderEvaluation(OrderEvaluationPO oepo) | 语法 | public Map<Integer, OrderEvaluationPO> updateOrderEvaluation(OrderEvaluationPO oepo)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 更新订单评价 |
| OrderDataService.getHotelEvalvation(String hotel) | 语法 | public Map<Integer, OrderEvaluationPO> getHotelEvalvation(String hotel) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 获取酒店评价列表 |
| OrderDataService.getCustomerEvaluation(String CustommerID) | 语法 | public Map<Integer, OrderEvaluationPO> getCustomerEvaluation(String CustommerID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 获取用户评价 |
| OrderDataService.insertOrder(OrderPO opo) | 语法 | public Map<Integer, OrderPO> insertOrder(OrderPO opo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 增加一个订单PO |
| OrderDataService.deleteOrder(OrderPO opo) | 语法 | public Map<Integer, OrderPO> deleteOrder(OrderPO opo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 删除一个订单 |
| OrderDataService.updateOrder(OrderPO opo) | 语法 | public Map<Integer, OrderPO> updateOrder(OrderPO opo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 更新一个订单 |
|  |  |  |

表4.2.3-2 数据层Customer模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CustomerDataService.find(String userName, String password) | 语法 | public ResultMessage find(String userName, String password) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的CustomerPO结果 |
| CustomerDataService.insert(CustomerPO po) | 语法 | Public void insert(CustomerPO po ) throws RemoteException; |
| 前置条件 | 同样ID的po在CustomerDataService中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| CustomerDataService.update(CustomerPO po) | 语法 | Public void update(CustomerPO po ) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| CustomerDataService.getCustomer(String userName) | 语法 | Public CustomerPO getCustomer(String userName ) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的PO |
| 后置条件 | 按ID进行查找返回PO对象 |

表5.4.2-3 数据层promotion模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionDataService.findHotelPromotion | 语法 | public HotelPromotionPO findHotelPromotion(HotelPromotionType hotelPromotionType, String hotelName) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中查找某一酒店促销策略 |
| PromotionDataService.updateHotelPromotion | 语法 | public boolean updateHotelPromotion(HotelPromotionType hotelPromotionType, HotelPromotionPO hppo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中更新某一酒店促销策略 |
| PromotionDataService.findWebPromotion | 语法 | public WebPromotionPO findWebPromotion(WebPromotionType webPromotionType) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中查找某一网站促销策略 |
| PromotionDataService.updateWebPromotion | 语法 | public boolean updateWebPromotion(WebPromotionType webPromotionType, WebPromotionPO wppo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中更新某一网站促销策略 |

表4.2.3-4 数据层hotel模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelDataService.findHotelInfo | 语法 | public HotelInfoPO findHotelInfo (String hotelName) |
| 前置条件 | 无 |
| 后置条件 | 根据酒店名称查找并返回相应的HotelInfoPO结果 |
| HotelDataService.updateHotelInfo | 语法 | public void updateHotelInfo (HotelInfoPO hipo) |
| 前置条件 | 在数据库中存在同样ID的HotelInfoPO |
| 后置条件 | 更新一个HotelInfoPO |
| HotelDataService.findHotels | 语法 | public ArrayList<HotelInfoPO>findHotels (SearchInfoPO sipo) |
| 前置条件 | 无 |
| 后置条件 | 根据搜索信息查找并返回多个HotelInfoPO结果 |
| HotelDataService.insertHotel | 语法 | public boolean insertHotel(HotelInfoPO hipo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中插入一个酒店 |
| HotelDataService.updateHotelScore | 语法 | public boolean updateHotelScore(String hotelName, double score) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 更新酒店评分 |

表4.2.3-5 数据层manager模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManagerDataService.check | 语法 | public ResultMessage check(ManagerType managerType, String username, String password) |
| 前置条件 | 无 |
| 后置条件 | 按管理人员类型和username进行查找,若找到则检查其password，返回相应的ResultMessage结果 |
| ManagerDataService.insert | 语法 | public ResultMessage insert(ManagerType type, ManagerPo po) |
| 前置条件 | 同样ID、type的ManagerPo对象在数据库中不存在 |
| 后置条件 | 在数据库中添加新的ManagerPo对象 |
| ManagerDataService.update | 语法 | public ResultMessage update(ManagerType type, ManagerPo po); |
| 前置条件 | 需要更新数据库中的某个ManagerPo对象 |
| 后置条件 | 在数据库中更新对应ManagerPo对象 |
| ManagerDataService.findManagerInfo | 语法 | public ManagerPO findManagerInfo(ManagerType managerType, String username) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按managerType和username进行查找返回相应的ManagerPO结果 |

表4.2.3-6 数据层Credit模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreditDataService.getCustomerCredits(String userName) | 语法 | public ArrayList<CreditInfoPO> getCustomerCredits(String userName)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 获取信用值变化记录 |
| CreditDataService.insert(CreditInfoPO creditInfoPO) | 语法 | public ResultMessage insert(CreditInfoPO creditInfoPO)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 插入一条信用值变化记录 |

表4.2.3-7 数据层Room模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomDataService.findAvailableRooms | 语法 | public ArrayList<AvailableRoomPO> findAvailableRooms(String hotelName) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 获取可用客房列表 |
| RoomDataService.updateAvailableRooms | 语法 | public boolean updateAvailableRooms(AvailableRoomPO arpo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 更新可用客房 |
| RoomDataService.insertAvailableRooms | 语法 | public boolean insertAvailableRooms(AvailableRoomPO arpo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 插入可用客房 |

**5.依赖视角**

下图是客户端和服务器端各自的包之间的依赖关系。

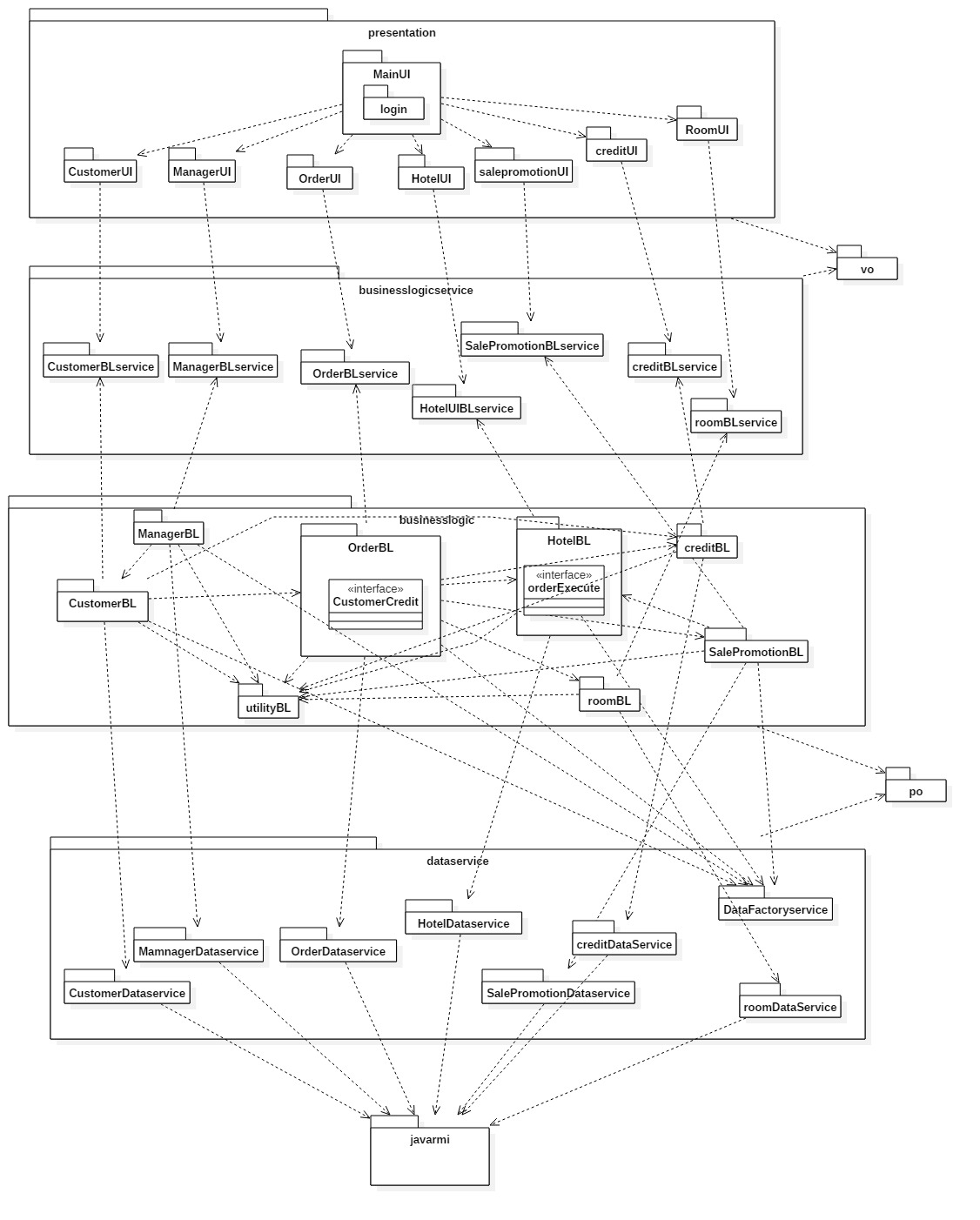


图5.1-1 酒店预订系统客户端开发包图

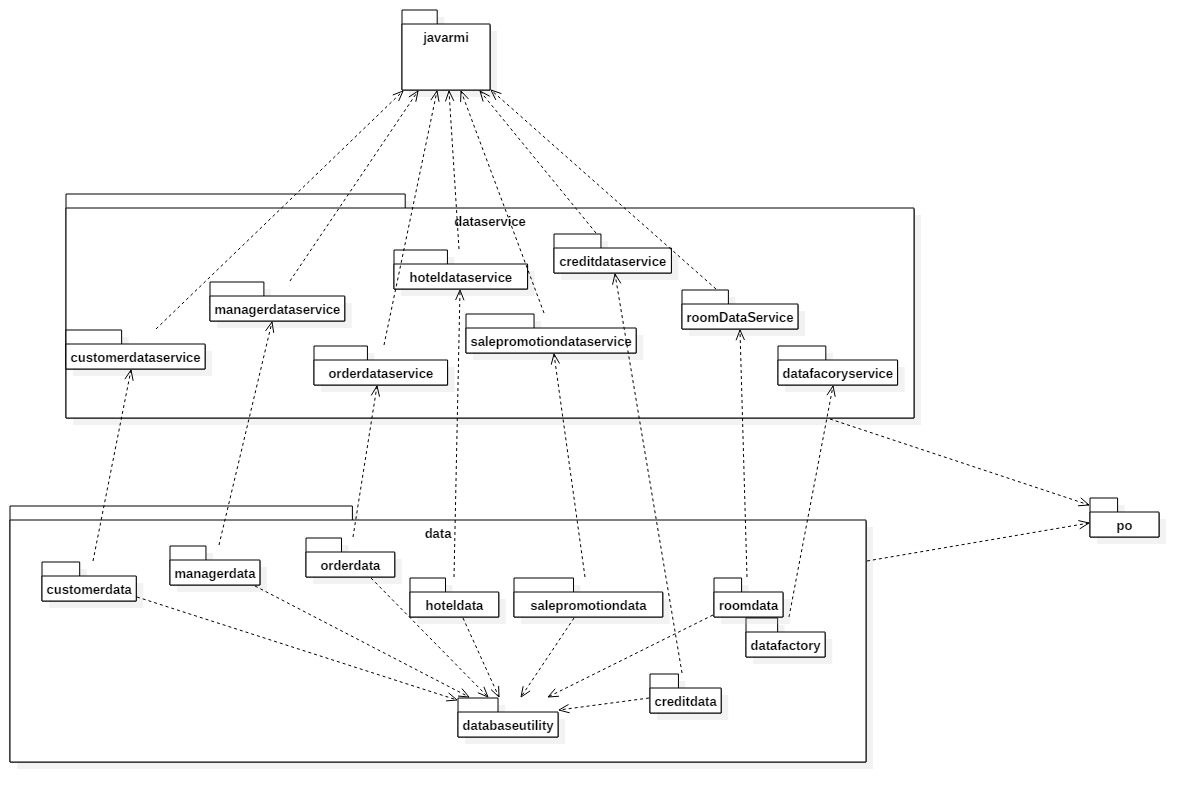


图5.1-2 酒店预订系统服务器端开发包图