酒店线上预订系统

OBSH(Online Booking System of Hotel)

详细设计描述文档

V2.1草稿

2016-10-29

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 胡一凡、毕潇晗 | **2016/10/29** | **初步的描述文档** | **V1.0草稿** |
| 毕潇晗 | **2016/11/6** | **修正了member credit promotion模块的接口规范** | **V2.0草稿** |
| 何阳 | **2016/11/6** | **增加了Hotel的接口规范** | **V2.1草稿** |
| 何阳、胡一凡 | **2016/11/13** | **修改Hotel、Order的顺序图等** | **V3.0正式版** |

16组成员：毕潇晗 顾忻 何阳 胡一凡

目录

[1.引言 2](#_Toc466216707)

[1.1编制目的 2](#_Toc466216708)

[1.2词汇表 2](#_Toc466216709)

[1.3参考资料 2](#_Toc466216710)

[2.产品概述 2](#_Toc466216711)

[3.体系结构设计概述 2](#_Toc466216712)

[4.结构视角 3](#_Toc466216713)

[4.1业务逻辑层的分解 3](#_Toc466216714)

[4.1.1 Userbl模块 3](#_Toc466216715)

[4.1.2 Memberbl模块 8](#_Toc466216716)

[4.1.3 Orderbl模块 11](#_Toc466216717)

[4.1.4 Creditbl模块 17](#_Toc466216718)

[4.1.5Hotelbl模块 2](#_Toc466216719)0

[4.1.6 HotelRoombl模块 2](#_Toc466216720)3

[4.1.7 Promotionbl模块 27](#_Toc466216721)

[5.依赖视角 3](#_Toc466216722)0

# 1.引言

## 1.1编制目的

本报告详细完成对酒店线上预订系统OBSH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。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| OBSH | 酒店线上预订系统 | 无 |

## 1.3参考资料

软件工程开发与计算（卷二）

酒店线上预订系统OBSH软件需求规格说明文档

酒店线上预订系统用例文档

酒店线上预定系统体系结构描述文档

# 2.产品概述

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

# 3.体系结构设计概述

参考酒店线上预订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4.1.2。

### 4.1.1 Userbl模块

**(1) 模块概述**

Userbl模块负责用户的登录、注册、查看修改信息等操作。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

**(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bussinesslogicservice.UserBLService接口。业务逻辑层和数据层之间添加dataservice.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，这样UserController会将对用户的逻辑业务处理委托给User对象。UserPO是作为用户的持久化对象被添加到设计模型中去的。

UserController模块的设计如图4.1.1.1所示

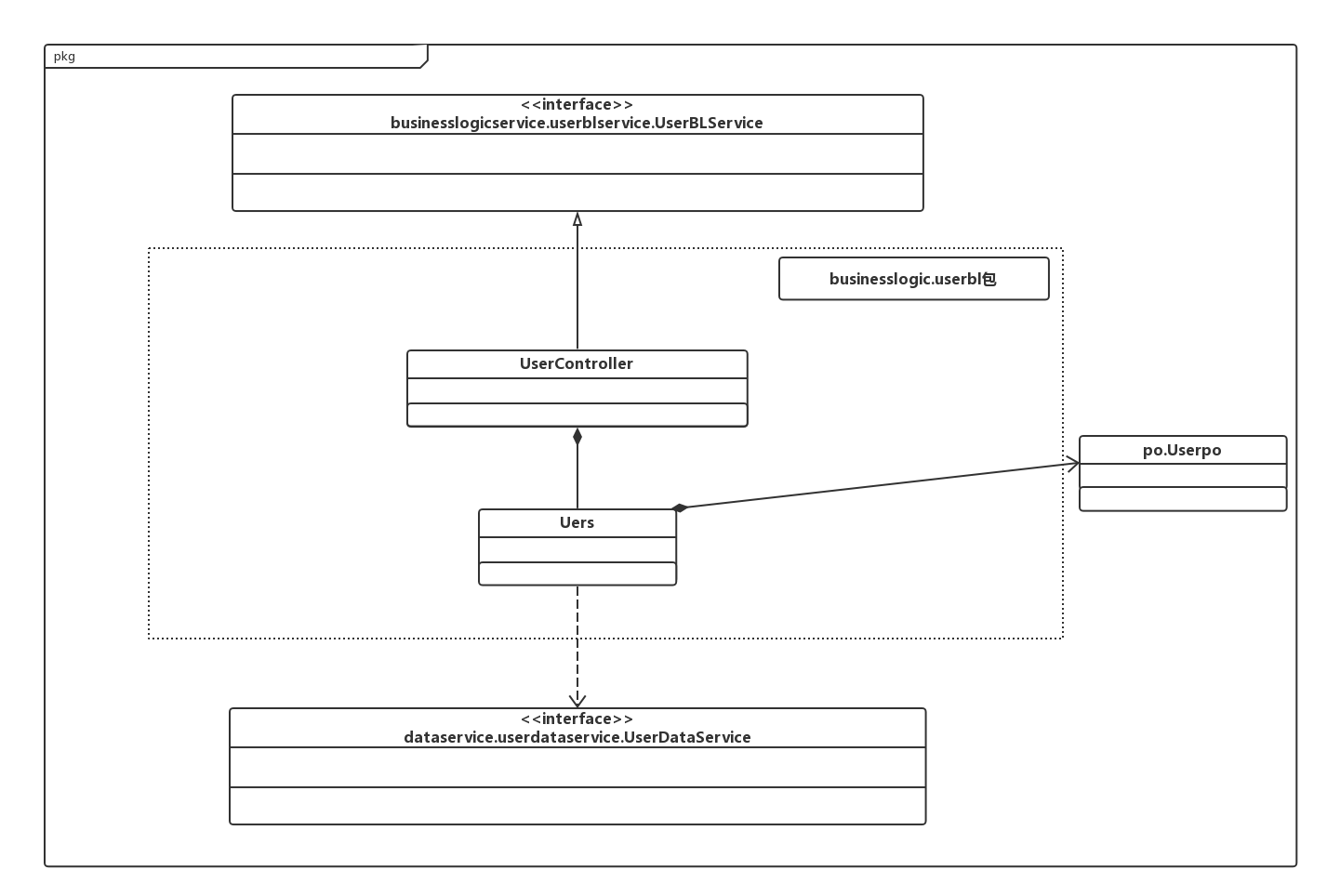


图4.1.1.1

Userbl模块各个类的职责如表4.1.1.1所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| UserController | 负责实现用户界面所需要的服务 |
| Users | 用户的领域模型对象，拥有用户的信息，可以帮助完成用户界面所需要的服务 |

表4.1.1.1

**(3) 模块内部类的接口规范**

UserController和Users的接口规范如表4.1.1.2和表4.1.1.3所示

|  |  |  |
| --- | --- | --- |
| **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，password符合输入规则 |
| 后置条件 | 调用User领域的Login方法 |
| UserController.Views | 语法 | public ArrayList<Hotel> Views(String address,String commercialDistrict) |
| 前置条件 | 已创建一个User领域对象，address,commercialDistrict符合输入规则 |
| 后置条件 | 调用User领域的Views方法 |
| UserController.GetCredit | 语法 | public ResultMessage GetCredit(UserVo vo) |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的GetCredit方法 |
| UserController.ModifyMessage | 语法 | public void ModifyMessage(UserVo vo) |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的ModifyMessage方法 |
| UserController.ModifyHotelMessage | 语法 | public ResultMessage ModifyHotelMessage(HotelVo vo) |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的ModifyHotelMessage方法 |
| UserController.Administration | 语法 | public void Administration(long id) |
| 前置条件 | 已创建一个User领域对象，输入的id符合输入规范 |
| 后置条件 | 调用User领域的Administration方法 |
| UserController.ModifyPassword | 语法 | public ResultMessage ModifyPassword(UserVo vo) |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的ModifyPassword方法 |
| UserController.Cancel | 语法 | public void Cancel(UserVO vo) |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的Cancel方法 |
| UserController.FilterByState | 语法 | public ArrayList<Hotel> filterbystate(OrderState state) |
| 前置条件 | 已创建一个User领域对象，输入的状态符合输入规范 |
| 后置条件 | 调用User领域的Filter方法 |
| UserController.SortByTime | 语法 | public ArrayList<Hotel> SortByTime (ArrayList<Hotel>) |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的SortByTime方法 |
| UserController.ViewDetail | 语法 | public ResultMessage ViewDetail(Hotel hotel) |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的ViewDetail方法 |
| UserController.EndSearch | 语法 | public void EndSearch() |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的EndSearch方法 |
| UserController.AddClient | 语法 | public ResultMessage AddClient(UserVo vo) |
| 前置条件 | 已创建一个User领域对象 |
| 后置条件 | 调用User领域的Add方法 |
| UserController.GetOrderHistory | 语法 | public ResultMessage GetOrderHistory (int id) |
| 前置条件 | 已创建一个User领域对象，输入的id符合输入规范 |
| 后置条件 | 调用User领域的GetOrderHistory方法 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| User.Login | 用户登录 | |
| User.AddClient | 添加一个新的用户 | |
| User.GetCredit | 获得用户的信用值 | |
| User.ModifyMessage | 修改用户的信息 | |
| User.ModifyHotelMessage | 修改酒店的信息 | |
| User.Administration | 添加酒店工作人员和网站营销人员 | |
| User.ModifyPassword | 修改用户密码 | |
| User.Cancel | 用户撤销未执行的订单 | |
| User.FilterByState | 获得根据状态排序的酒店信息列表 | |
| User.SortByTime | 获得根据时间排序的酒店信息列表 | |
| User.ViewDetail | 获得酒店的详细信息 | |
| User.Views | 获得指定地址商圈的酒店列表信息 | |
| User.EndSearch | 结束搜索 | |
| User.GetOrderHistory | 获得历史预订的酒店列表信息 | |

表4.1.1.2 UserController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 | password符合输入规则 |
| 后置条件 | 系统查找是否存在id与相应的password，根据输入返回登录验证的结果 |
| User.Views | 语法 | public ArrayList<Hotel> Views(String address,String commercialDistrict) |
| 前置条件 | address,commercialDistrict符合输入规则 |
| 后置条件 | 系统根据客户输入的地址商圈查找相应范围内的酒店，返回查找的结果 |
| User.GetCredit | 语法 | public ResultMessage GetCredit(UserVo vo) |
| 前置条件 | 无 |
| 后置条件 | 系统返回信用值 |
| User.ModifyMessage | 语法 | public void ModifyMessage(UserVo vo) |
| 前置条件 | 无 |
| 后置条件 | 系统更新客户个人信息 |
| User.ModifyHotelMessage | 语法 | public ResultMessage ModifyHotelMessage(HotelVo vo) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化更新酒店基本信息 |
| User.Administration | 语法 | public void Administration(long id) |
| 前置条件 | 输入的id符合输入规范 |
| 后置条件 | 系统更新用户信息 |
| User.ModifyPassword | 语法 | public ResultMessage ModifyPassword(UserVo vo) |
| 前置条件 | 无 |
| 后置条件 | 系统更新用户密码 |
| User.Cancel | 语法 | public void Cancel(UserVO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化更新用户信息 |
| User.FilterByState | 语法 | public ArrayList<Hotel> filterbystate(OrderState state) |
| 前置条件 | 输入的状态符合输入规范 |
| 后置条件 | 系统根据state返回酒店列表信息 |
| User.SortByTime | 语法 | public ArrayList<Hotel> SortByTime (ArrayList<Hotel>) |
| 前置条件 | 无 |
| 后置条件 | 系统返回按时间排序后的酒店基本信息 |
| User.ViewDetail | 语法 | public ResultMessage ViewDetail(Hotel hotel) |
| 前置条件 | 无 |
| 后置条件 | 系统返回酒店详细信息 |
| User.EndSearch | 语法 | public void EndSearch() |
| 前置条件 | 无 |
| 后置条件 | 系统结束搜索酒店，持久化更新涉及的领域对象的数据 |
| User.AddClient | 语法 | public ResultMessage AddClient(UserVo vo) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化更新用户信息 |
| User.GetOrderHistory | 语法 | public ResultMessage GetOrderHistory (int id) |
| 前置条件 | 输入的id符合输入规范 |
| 后置条件 | 根据输入的id返回酒店预订记录 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| DatabaseFacory.getUserDatabase | 得到User数据库的服务的引用 | |
| UserDataService.find(long id) | 根据用户ID获得单一持久化对象 | |
| UserDataService.finds(String address,String commercialDistrict) | 根据地址、商圈获得多个酒店对象 | |
| UserDataService.findbycondition(UserPO po) | 根据条件获得多个酒店对象 | |
| UserDataService.insert(UserPO po) | 插入单一持久化对象 | |
| UserDataService.delete(UserPO po) | 删除单一持久化对象 | |
| UserDataService.update(UserPO po) | 更新单一持久化对象 | |
| UserDataService.finds(String address,String commercialDistrict) | 根据获得用户对象 | |

表4.1.1.3 User 的接口规范

**(4) 业务逻辑层的动态模型**

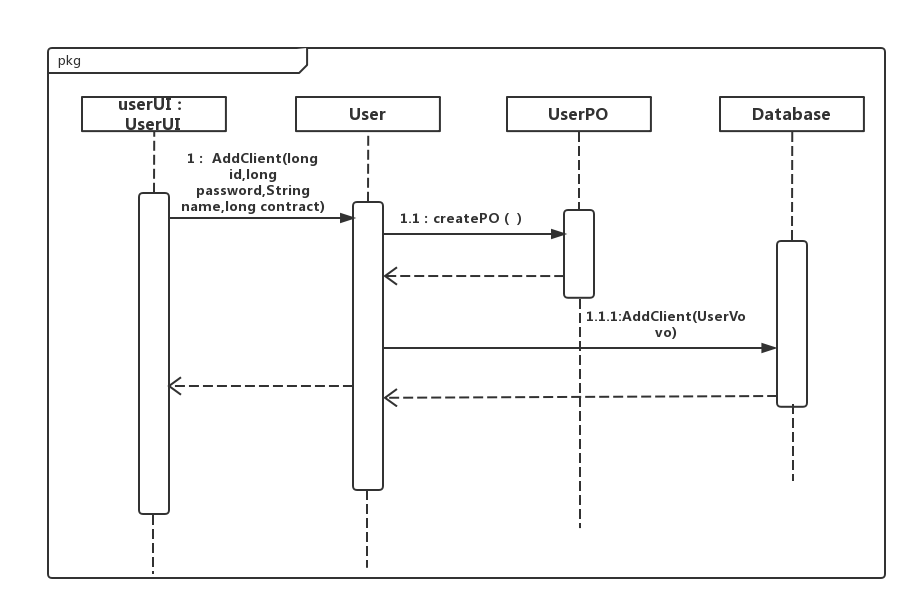


图4.1.1.2 添加客户的顺序图

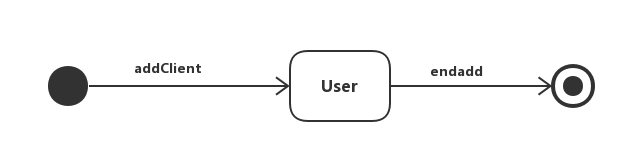


图4.1.1.3 User对象状态图

### 4.1.2 Memberbl模块

**(1) 模块概述**

Memberbl模块负责实现注册会员和计算会员等级。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

**(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。比如展示层和业务逻辑层之间，我们添加bussinesslogicservice.MemberBLService接口。业务逻辑层和数据层之间添加dataservice.MemberDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了MemberController，这样MemberController会将对注册会员返回会员等级委托给Member对象。MemberPO是作为会员信息的持久化对象被添加到设计模型中去的。UserInfo和PromotionInfo都是根据依赖倒置原则，为了消除循环依赖产生的接口。

Memberbl模块的设计如图4.1.2.1所示。

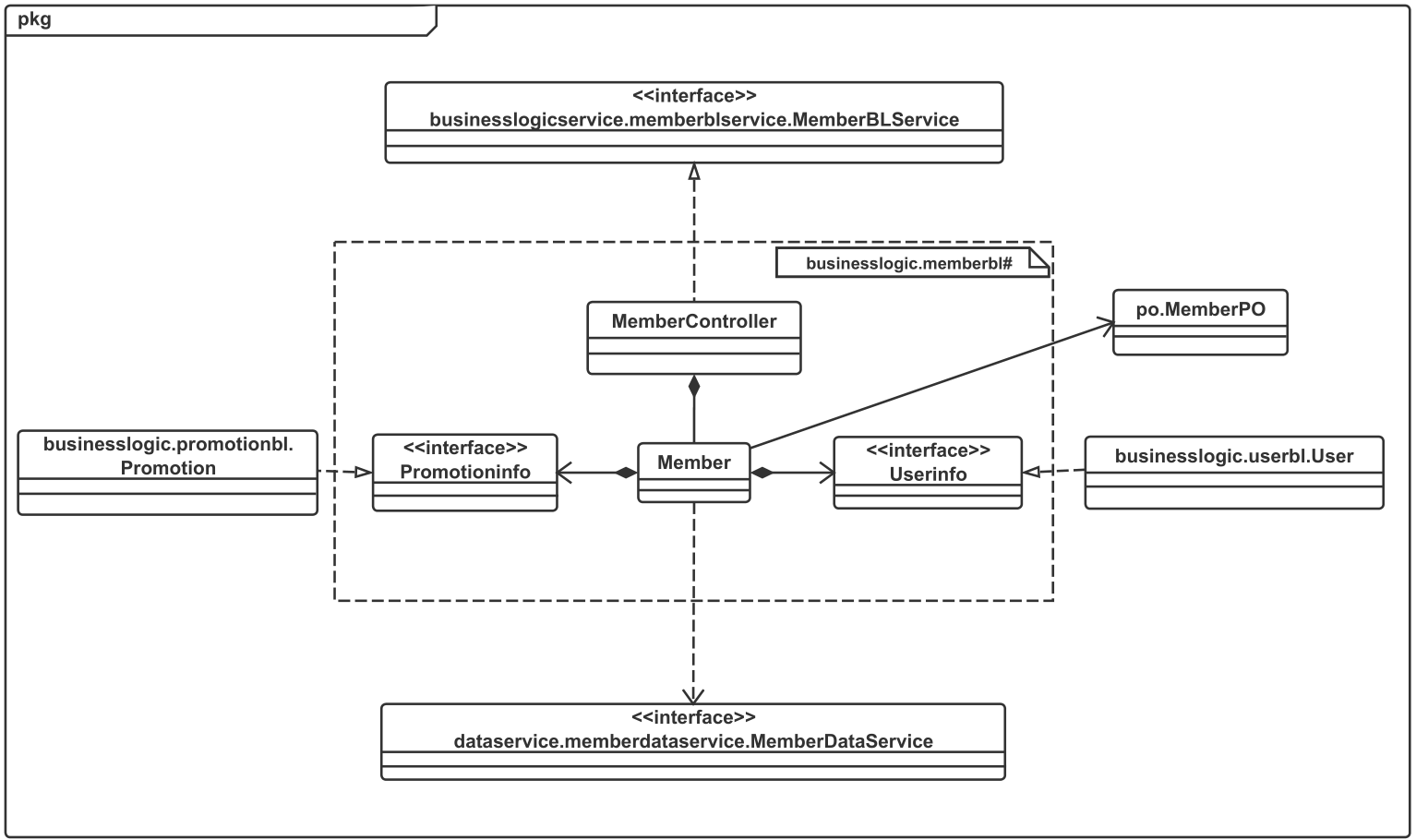


图4.1.2.1

Memberbl模块各个类的职责如表4.1.2.1所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| MemberController | 负责实现注册会员界面所需要的服务 |
| Member | 系统会员的领域模型对象，拥有会员数据的姓名、生日或企业名称，可以帮助完成注册会员及计算会员等级所需要的服务 |

表4.1.2.1

**(3) 模块内部类的接口规范**

MemberController和Member的接口规范如表4.1.2.2和表4.1.2.3所示

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| MemberController.addUser | 语法 | public ResultMessage addUser(int id) |
| 前置条件 | 已创建一个Member领域对象 |
| 后置条件 | 调用Member领域对象的addUser方法 |
| MemberController.createByPersonal | 语法 | public void createByPersonal (int id, string birthday) |
| 前置条件 | 已创建一个Member领域对象，并且已添加客户信息 |
| 后置条件 | 调用Member领域对象的createByPersonal方法 |
| MemberController.createByBusiness | 语法 | public void createByBusiness (int id, string name) |
| 前置条件 | 已创建一个Member领域对象，并且已添加客户信息 |
| 后置条件 | 调用Member领域对象的createByBusinessl方法 |
| MemberController.getRank | 语法 | public int getRank(int rank) |
| 前置条件 | 已创建一个Member领域对象，并且已添加获取会员的信用值，信用值判定方法 |
| 后置条件 | 调用Member领域对象的getRank方法 |
| Member.isMember | 语法 | public boolean createByBusiness (int id) |
| 前置条件 | 已创建一个Member领域对象，并且已添加客户信息 |
| 后置条件 | 调用Member领域对象的isMember方法 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| MemberController.addUser(int id) | 加入一个客户对象 | |

表4.1.2.2 MemberControlle的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Member.addUser | 语法 | | public ResultMessage addUser(int id) |
| 前置条件 | | 启动会员注册 |
| 后置条件 | | 增加客户信息 |
| Member.createByPersonal | 语法 | | public void createByPersonal (int id, string birthday) |
| 前置条件 | | 已添加客户信息 |
| 后置条件 | | 增加该客户个人会员信息 |
| Member.createByBusiness | 语法 | | public void createByBusiness (int id, string name) |
| 前置条件 | | 已添加客户信息 |
| 后置条件 | | 增加该客户企业会员信息 |
| Member.getRank | 语法 | | public int getRank(double credit) |
| 前置条件 | | 已添加会员信息、信用信息；已获得会员等级计算梯度 |
| 后置条件 | | 根据会员信用值多少，计算会员等级 |
| Member.isMember | 语法 | | public boolean createByBusiness (int id) |
| 前置条件 | | 已添加客户信息 |
| 后置条件 | | 无 |
| **需要的服务（需接口）** | | | |
| **服务名** | | **服务** | |
| DatabaseFacory.getMemberDatabase | | 得到member数据库的服务的引用 | |
| MemberDataService.insert(MemberPO po) | | 插入单一持久化对象 | |
| MemberDataService.delete(MemberPO po) | | 删除单一持久化对象 | |
| MemberDataService.update(MemberPO po) | | 更新单一持久化对象 | |
| MemberDataService.find(int id) | | 根据ID进行查找单一持久化对象 | |

表4.1.2.3 Member的接口规范

**(4) 业务逻辑层的动态模型**

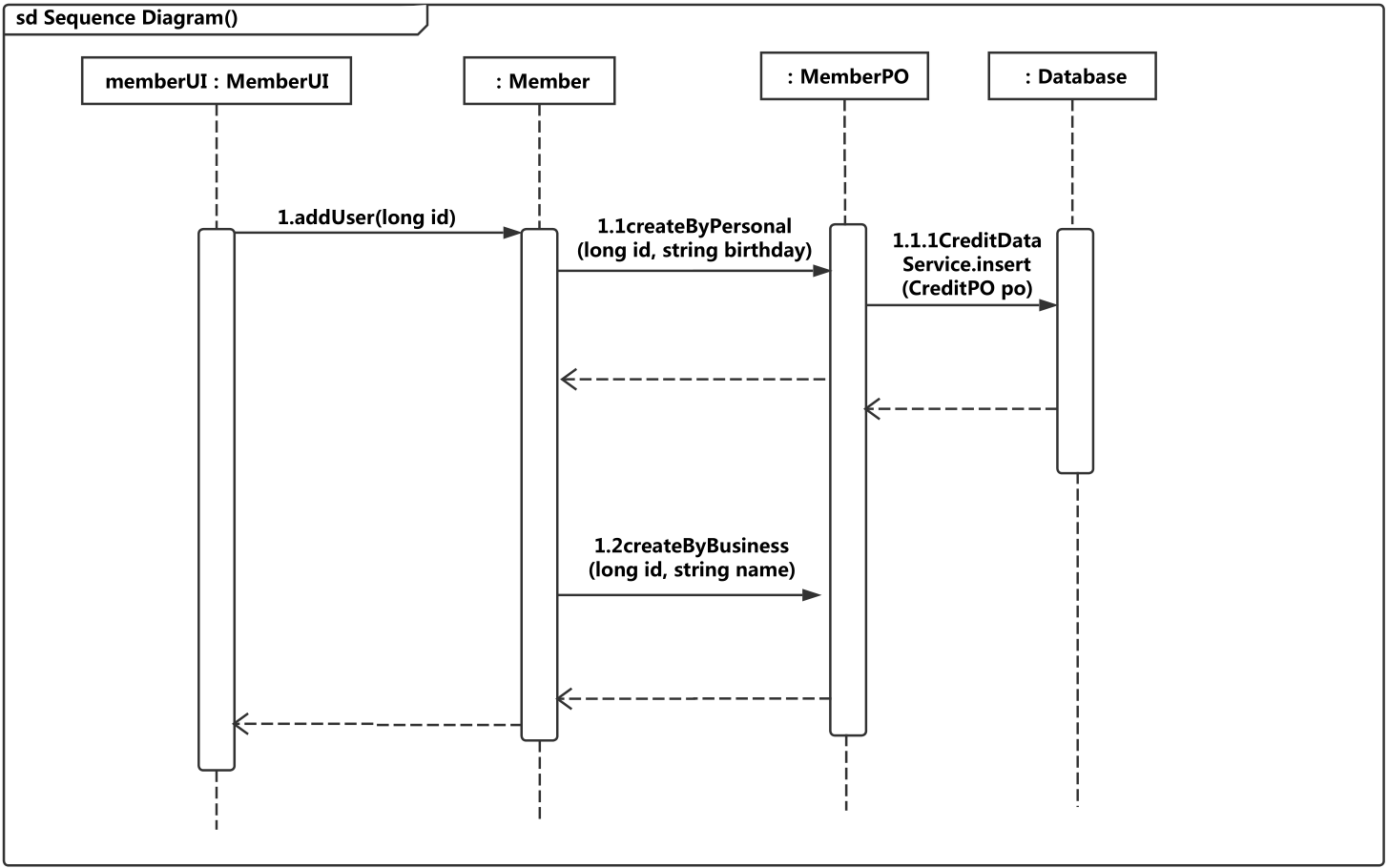


图4.1.2.2 注册会员的顺序图

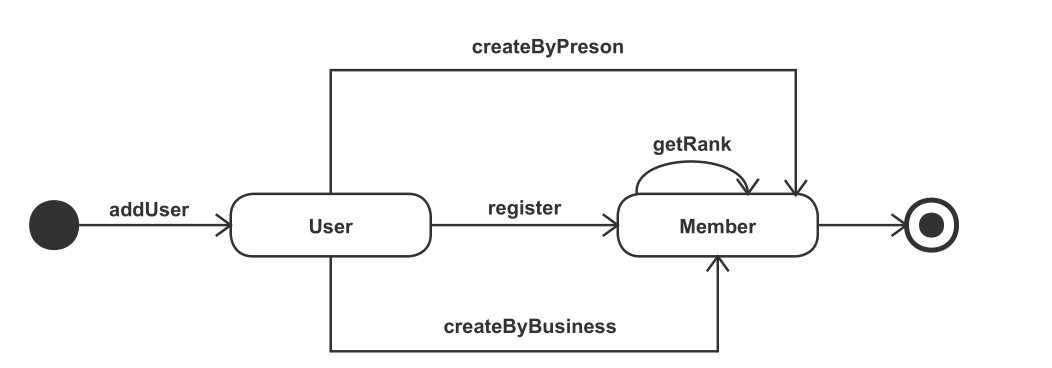


图4.1.2.3 member对象的状态图

### 4.1.3 Orderbl模块

**(1) 模块概述**

Orderbl模块负责处理订单，进行订单的生成、订单的排序修改等操作。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

**(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。比如展示层和业务逻辑层之间，我们添加bussinesslogicservice. OrderBLService接口。业务逻辑层和数据层之间添加dataservice. 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会将对库存的逻辑业务处理委托给Order对象。OrderPO是作为订单记录的持久化对象被添加到设计模型中去的。OrderList的添加是HotelRoomInfo的容器类。

Orderbl模块的设计如图4.1.3.1所示

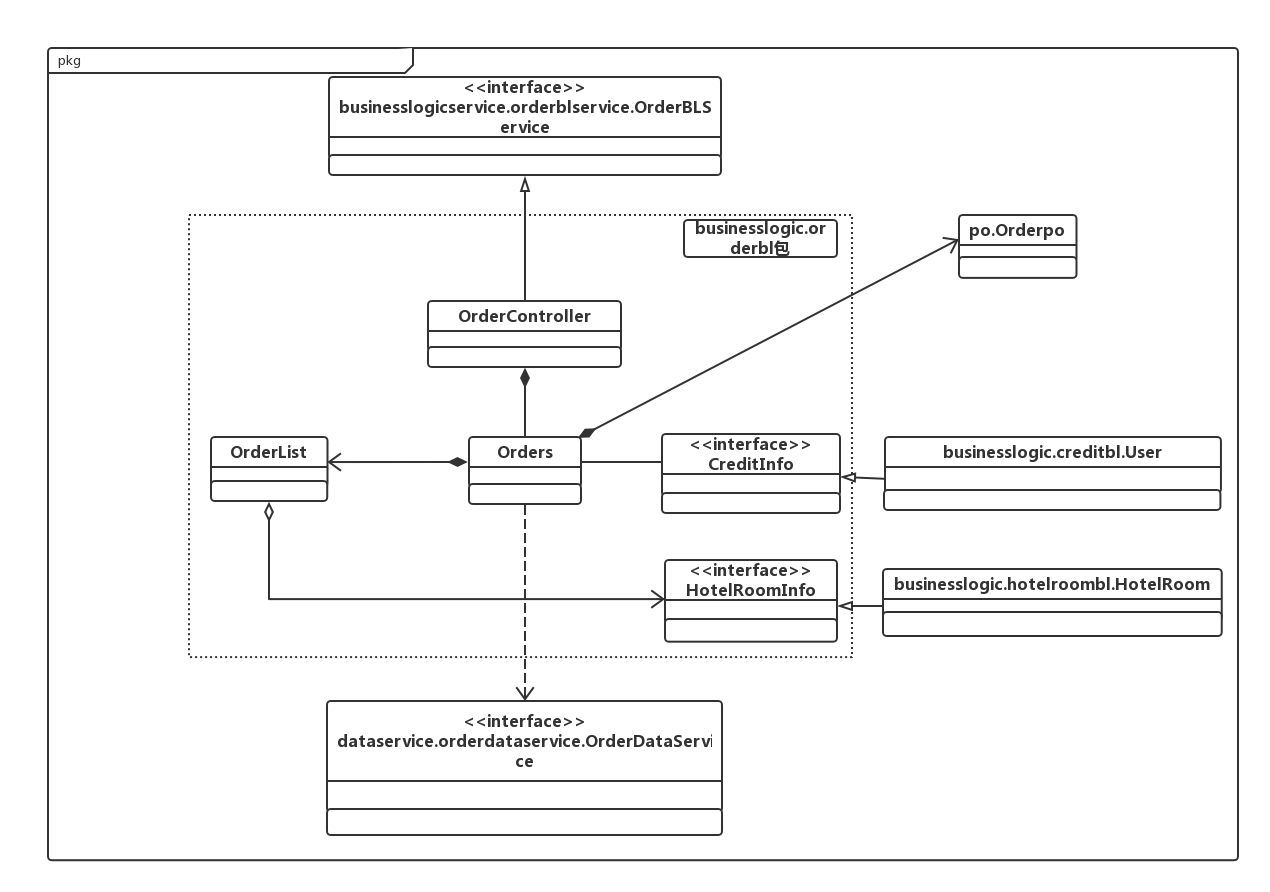


图4.1.3.1

Orderbl模块各个类的职责如表4.1.3.1所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| OrderController | 负责实现订单界面所需要的服务 |
| Orders | 订单的领域模型对象，拥有一次订单所持有的客户、客房、订单记录等信息，可以帮助完成订单界面所需要的服务 |

表4.1.3.1

**(3) 模块内部类的接口规范**

OrderController和Orders的接口规范如表4.1.3.2和表4.1.3.3所示

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| OrderController.Cancellation | 语法 | Public void Cancellation(OrderVo vo) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的Cancellation方法 |
| OrderController.CancelTime | 语法 | Public Time CancelTime(OrderVo vo) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的CancelTime方法 |
| OrderController.CancelKillCredit | 语法 | Public void CancelKillCredit(OrderVo vo) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的CancelKillCredit方法 |
| OrderController.Assess | 语法 | Public ResultMessage Assess(int score,String comment，OrderVo vo) |
| 前置条件 | 已创建一个Order领域对象，并且输入的评分、评语符合输入规则 |
| 后置条件 | 调用Order领域对象的Assess方法 |
| OrderController.Add | 语法 | Public ResultMessage Add(OrderVo vo) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的Add方法 |
| OrderController.CreditCheck | 语法 | Public ResultMessage CreditCheck(OrderVo vo) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的CreditCheck方法 |
| OrderController.CalPromotion | 语法 | Public PromotionVo CalPromotion(int id) |
| 前置条件 | 已创建一个Order领域对象，并且输入的订单号符合输入规则 |
| 后置条件 | 调用Order领域对象的CalPromotion方法 |
| OrderController.CalPrice | 语法 | Public OrderVo CalPrice(PromotionVo vo1,OrderVo vo2) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的CalPrice方法 |
| OrderController.Solve | 语法 | Public ResultMessage Solve(OrderVo vo) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的Solve方法 |
| OrderController.ComplainDeal | 语法 | Public ResultMessage Complain(OrderVo vo) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的ComplainDeal方法 |
| OrderController.ViewByID | 语法 | Public Order ViewByID (int id) |
| 前置条件 | 已创建一个Order领域对象，并且输入的账号符合输入规则 |
| 后置条件 | 调用Order领域对象的ViewByID方法 |
| OrderController.Views | 语法 | Public ArrayList<Order> Views(int id) |
| 前置条件 | 已创建一个Order领域对象，并且输入的账号符合输入规则 |
| 后置条件 | 调用Order领域对象的Views方法 |
| OrderController.ViewByCustom | 语法 | Public ArrayList<Order> ViewByCustom(int id) |
| 前置条件 | 已创建一个Order领域对象，并且输入的账号符合输入规则 |
| 后置条件 | 调用Order领域对象的ViewByCustom方法 |
| OrderController.ViewByDaily | 语法 | Public ArrayList<Order> ViewByDaily(Date date) |
| 前置条件 | 已创建一个Order领域对象，并且输入的日期符合输入规则 |
| 后置条件 | 调用Order领域对象的ViewByDaily方法 |
| OrderController.ViewByState | 语法 | Public ArrayList<Order> ViewByState(OrderState state) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的ViewByState方法 |
| OrderController.TimeSort | 语法 | Public ArrayList<Order> TimeSort ( ArrayList <Order> ) |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的TimeSort方法 |
| OrderController.CreateID | 语法 | Public String CreateID() |
| 前置条件 | 已创建一个Order领域对象 |
| 后置条件 | 调用Order领域对象的CreateID方法 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| Order.Cancellation | 撤销一个订单 | |
| Order.CancelTime | 获得订单开始时间到现在的时间差 | |
| Order.CancelKillCredit | 扣除指定客户的信用值 | |
| Order.Assess | 评价指定酒店 | |
| Order.Add | 添加一个订单对象 | |
| Order.CreditCheck | 检查客户的信用值信息 | |
| Order.CalPromotion | 计算指定订单的适用的最优折扣 | |
| Order.CalPrice | 计算订单的价格 | |
| Order.Solve | 将订单的状态置为已执行 | |
| Order.ComplainDeal | 撤销订单并恢复指定客户的指定数目的信用值 | |
| Order.ViewByID | 获得指定订单号的订单信息 | |
| Order.Views | 获得指定账号需要的订单列表信息 | |
| Order.ViewByCustom | 获得指定客户的订单列表 | |
| Order.ViewByDaily | 把订单列表按日期排序 | |
| Order.ViewByState | 把订单列表按状态排序 | |
| Order.TimeSort | 把订单列表按时间排序 | |
| Order.CreateID | 创建一个新的订单号 | |

表4.1.3.2 OrderController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Order.Cancellation | 语法 | | Public void Cancellation(OrderVo vo) |
| 前置条件 | | 无 |
| 后置条件 | | 系统持久化修改该订单对象的状态 |
| Order.CancelTime | 语法 | | Public Time CancelTime(OrderVo vo) |
| 前置条件 | | 无 |
| 后置条件 | | 系统计算下订单到现在的时间差并返回 |
| Order.CancelKillCredit | 语法 | | Public void CancelKillCredit(OrderVo vo) |
| 前置条件 | | 无 |
| 后置条件 | | 系统持久化修改该订单对应用户的信用值 |
| Order.Assess | 语法 | | Public ResultMessage Assess(int score,String comment，OrderVo vo) |
| 前置条件 | | 已输入评分/评语并提交 |
| 后置条件 | | 系统持久化更新该订单对应酒店的信息 |
| Order.Add | 语法 | | Public ResultMessage Add(OrderVo vo) |
| 前置条件 | | 客户输入订单信息并且格式正确 |
| 后置条件 | | 系统持久化更新该订单的用户信息 |
| Order.CreditCheck | 语法 | | Public ResultMessage CreditCheck(OrderVo vo) |
| 前置条件 | | 已经进入订单生成环节 |
| 后置条件 | | 系统根据用户的信用值返回是否通过 |
| Order.CalPromotion | 语法 | | Public PromotionVo CalPromotion(int id) |
| 前置条件 | | 输入的id符合规范 |
| 后置条件 | | 系统返回该客户满足的最优优惠政策 |
| Order.CalPrice | 语法 | | Public OrderVo CalPrice(PromotionVo vo1,OrderVo vo2) |
| 前置条件 | | 无 |
| 后置条件 | | 系统根据优惠政策计算促销后的价格 |
| Order.Solve | 语法 | | Public ResultMessage Solve(OrderVo vo) |
| 前置条件 | | 无 |
| 后置条件 | | 系统持久化更新该订单的信息 |
| Order.ComplainDeal | 语法 | | Public ResultMessage Complain(OrderVo vo) |
| 前置条件 | | 无 |
| 后置条件 | | 系统持久化更新该订单的用户信息 |
| Order.ViewByID | 语法 | | Public Order ViewByID (int id) |
| 前置条件 | | 输入的订单id符合输入规范 |
| 后置条件 | | 系统返回订单的信息 |
| Order.Views | 语法 | | Public ArrayList<Order> Views(int id) |
| 前置条件 | | 输入的用户id符合输入规范 |
| 后置条件 | | 系统返回订单列表 |
| Order.ViewByCustom | 语法 | | Public ArrayList<Order> ViewByCustom(int id) |
| 前置条件 | | 输入的id符合输入规范 |
| 后置条件 | | 系统返回客户的订单列表 |
| Order.ViewByDaily | 语法 | | Public ArrayList<Order> ViewByDaily(Date date) |
| 前置条件 | | 输入的日期符合输入规范 |
| 后置条件 | | 系统返回对应日期的订单列表 |
| Order.ViewByState | 语法 | | Public ArrayList<Order> ViewByState(OrderState state) |
| 前置条件 | | 输入的订单状态符合输入规范 |
| 后置条件 | | 系统返回对应状态的订单列表 |
| Order.TimeSort | 语法 | | Public ArrayList<Order> TimeSort ( ArrayList <Order> ) |
| 前置条件 | | 无 |
| 后置条件 | | 系统返回按时间排序的订单列表 |
| Order.CreateID | 语法 | | Public String CreateID() |
| 前置条件 | | 无 |
| 后置条件 | | 系统结合当前时间创建一个订单号 |
| **需要的服务（需接口）** | | | |
| **服务名** | | **服务** | |
| UserDataService.find(int id) | | 根据用户ID获得单一持久化对象 | |
| OrderDataService.find(int id) | | 根据订单号ID获得单一持久化对象 | |
| OrderDataService.findbydaily(int id) | | 根据时间得到订单对象 | |
| OrderDataService.findbycustom(Date date) | | 根据用户ID获得订单对象 | |
| OrderDataService.finds(String hotel,UserPO po) | | 根据酒店以及用户得到订单对象 | |
| OrderDataService.findbystate(State state) | | 得到根据状态的订单对象 | |
| OrderDataService.delete(OrderPo po) | | 删除单一持久化对象 | |
| OrderDataService.updata(OrderPo po) | | 更新单一持久化对象 | |
| OrderDataService.insert(OrderPo po) | | 插入单一持久化对象 | |
| DatabaseFacory.getOrderDatabase | | 得到Order数据库的服务的引用 | |
| UserDataService.update(UserPo po) | | 更新单一持久化对象 | |
| HotelRoomDataService.update(HotelRoomPo po) | | 更新单一持久化对象 | |
| CreditDataService.find(CreditPo po) | | 得到单一持久化对象 | |

表4.1.3.3 Orders的接口规范

**(4) 业务逻辑层的动态模型**

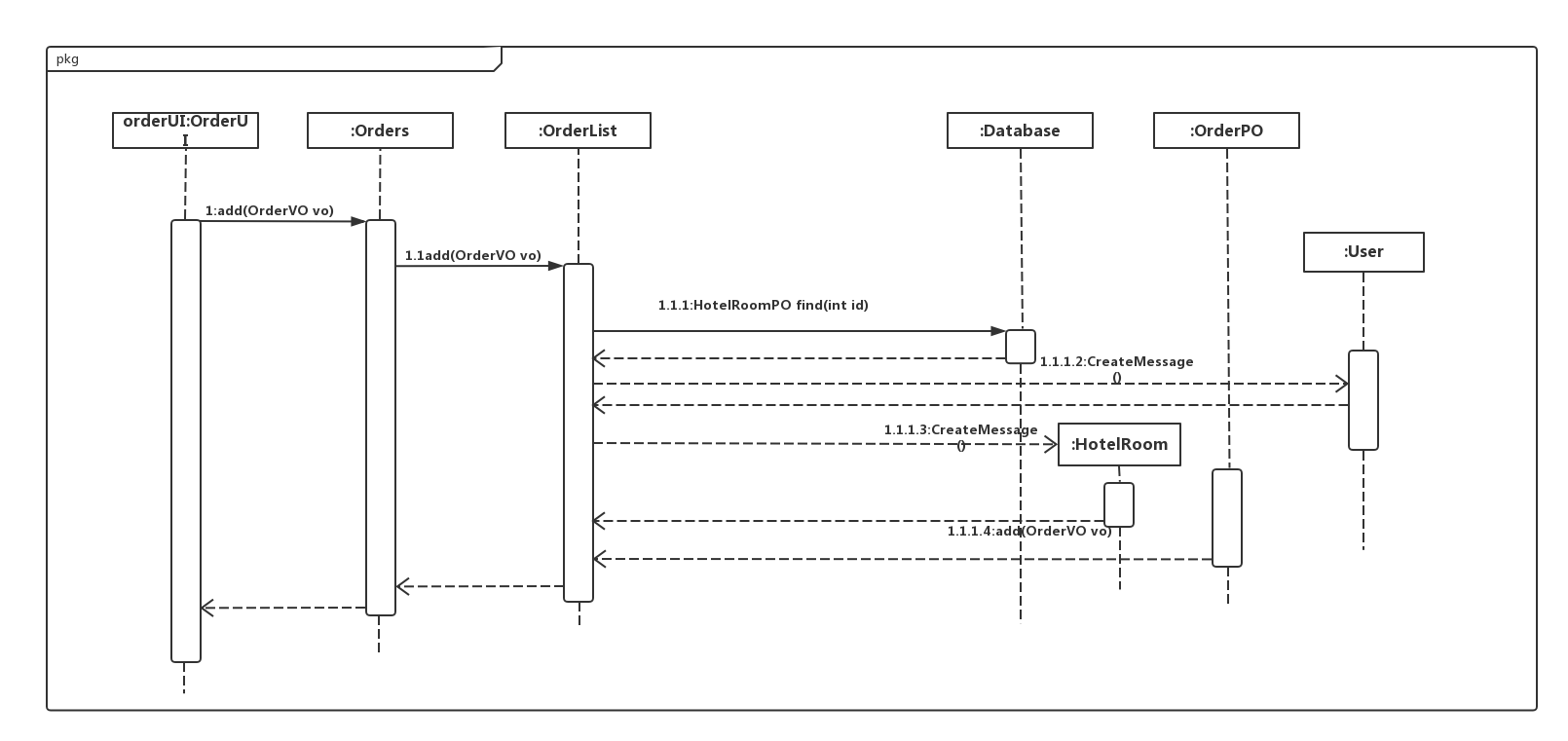


图4.1.3.2 为订单添加客房的顺序图

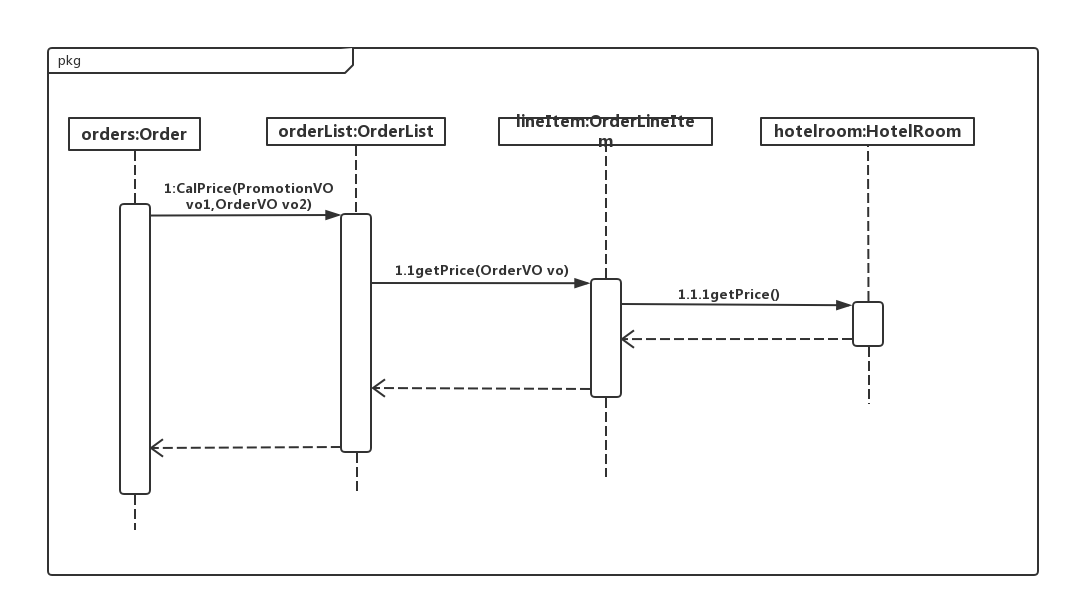


图4.1.3.3 得到订单总价的顺序图

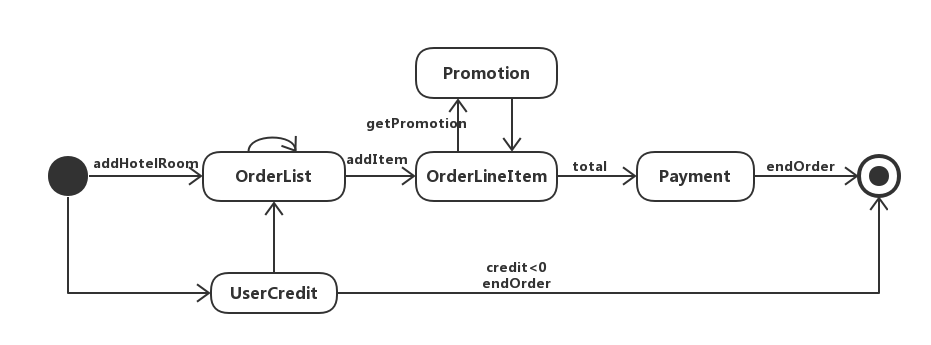


图4.1.3.5 Order对象状态图

### 4.1.4 Creditbl模块

**(1) 模块概述**

Creditbl模块负责实现信用充值和管理客户信用，具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

**(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。比如展示层和业务逻辑层之间，我们添加bussinesslogicserviceCreditBLService接口。业务逻辑层和数据层之间添加dataservice. 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会将对信用值的管理委托给Credit对象。CreditPO是作为信用系统的持久化对象被添加到设计模型中去的。

Orderbl模块的设计如图4.1.4.1所示

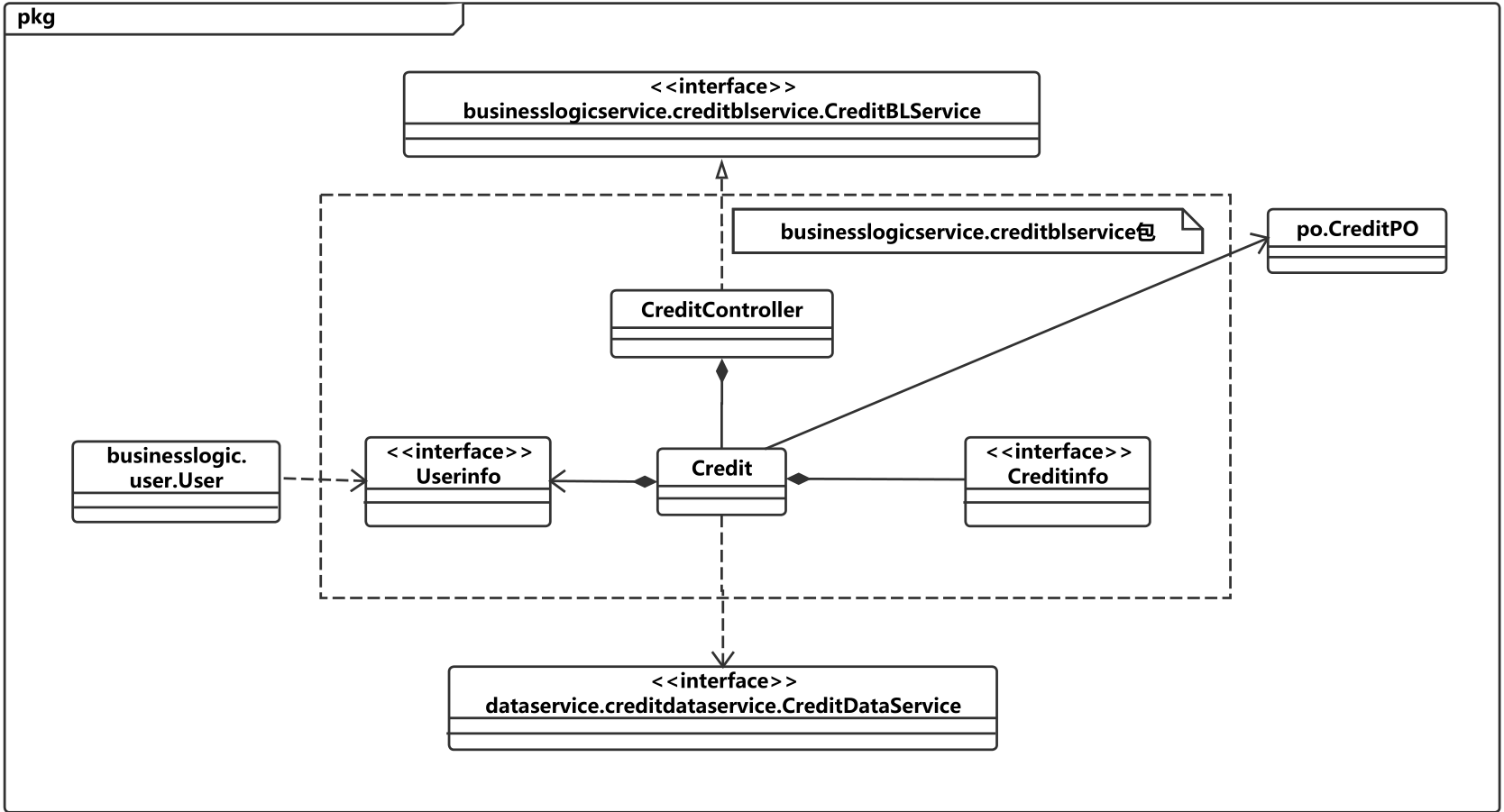


图4.1.4.1

Creditbl模块各个类的职责如表4.1.4.1所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| CreditController | 负责实现各信用充值界面所需要的服务 |
| Credit | 信用系统的领域模型对象，拥有客户的信用值，可以帮助完成信用充值、信用查询需要的服务 |

**(3) 模块内部类的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| CreditController.addUser | 语法 | public ResultMessage addUser (int id) |
| 前置条件 | 已创建一个Credit领域对象 |
| 后置条件 | 调用Credit领域对象的addUser方法 |
| CreditController.addCredit | 语法 | public double addCredit (double value) |
| 前置条件 | 已添加客户信息，且收到充值金额数目 |
| 后置条件 | 增加客户相应信用值 |
| CreditController.recoverCredit | 语法 | public double recoverCredit (double price, int tag) |
| 前置条件 | 已创建一个Credit领域对象,并且已添加客户信息 |
| 后置条件 | 调用Credit领域对象的recoverCredit方法 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| CreditController.addUser | 加入一个User对象 | |

表4.1.4.2 CreditController的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Credit.addUser | 语法 | public ResultMessage addUser int id) |
| 前置条件 | 已创建一个Credit领域对象 |
| 后置条件 | 调用Credit领域对象的addUser方法 |
| Credit.addCredit | 语法 | public double addCredit (double value) |
| 前置条件 | 已添加客户信息，且收到充值金额数目 |
| 后置条件 | 增加客户相应信用值 |
| Credit.recoverCredit | 语法 | public void recoverCredit (double price, int tage) |
| 前置条件 | 已添加客户信息 |
| 后置条件 | 选择恢复订单价格一半或全部的信用值 |
| **需要的服务（需接口）** | | |
| **服务名** | | **服务** |
| DatabaseFacory.getCreditDatabase | | 得到Credit数据库的服务的引用 |
| CreditDataService.insert(CreditPO po) | | 插入单一持久化对象 |
| CreditDataService.delete(CreditPO po) | | 删除单一持久化对象 |
| CreditDataService.update(CreditPO po) | | 更新单一持久化对象 |
| CreditDataService.find(int id) | | 根据ID进行查找单一持久化对象 |

表4.1.4.3 Credit的接口规范

**(4) 业务逻辑层的动态模型**

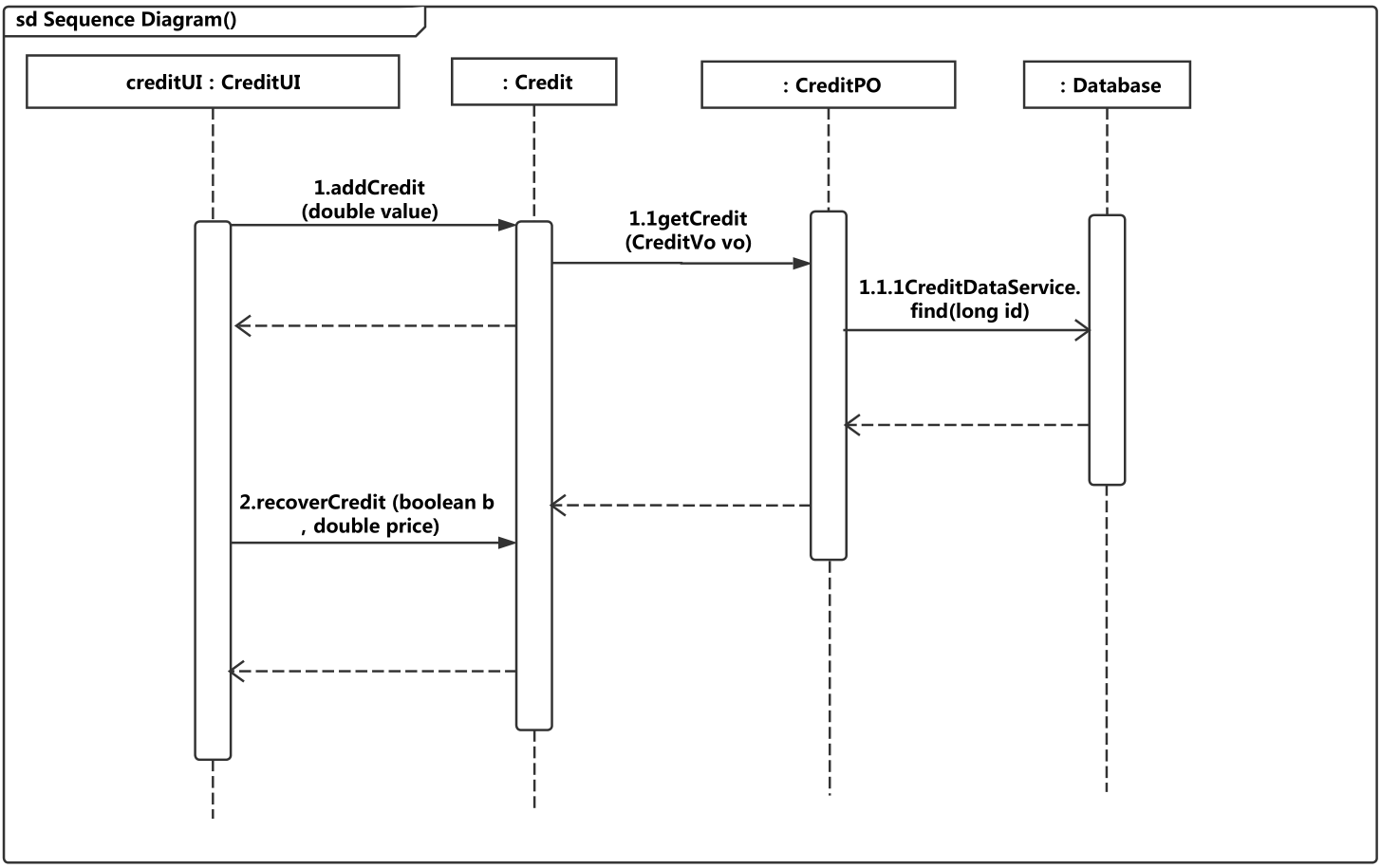


图4.1.4.2信用充值的顺序图



图4.1.4.3 credit对象的状态图

### 4.1.5Hotelbl模块

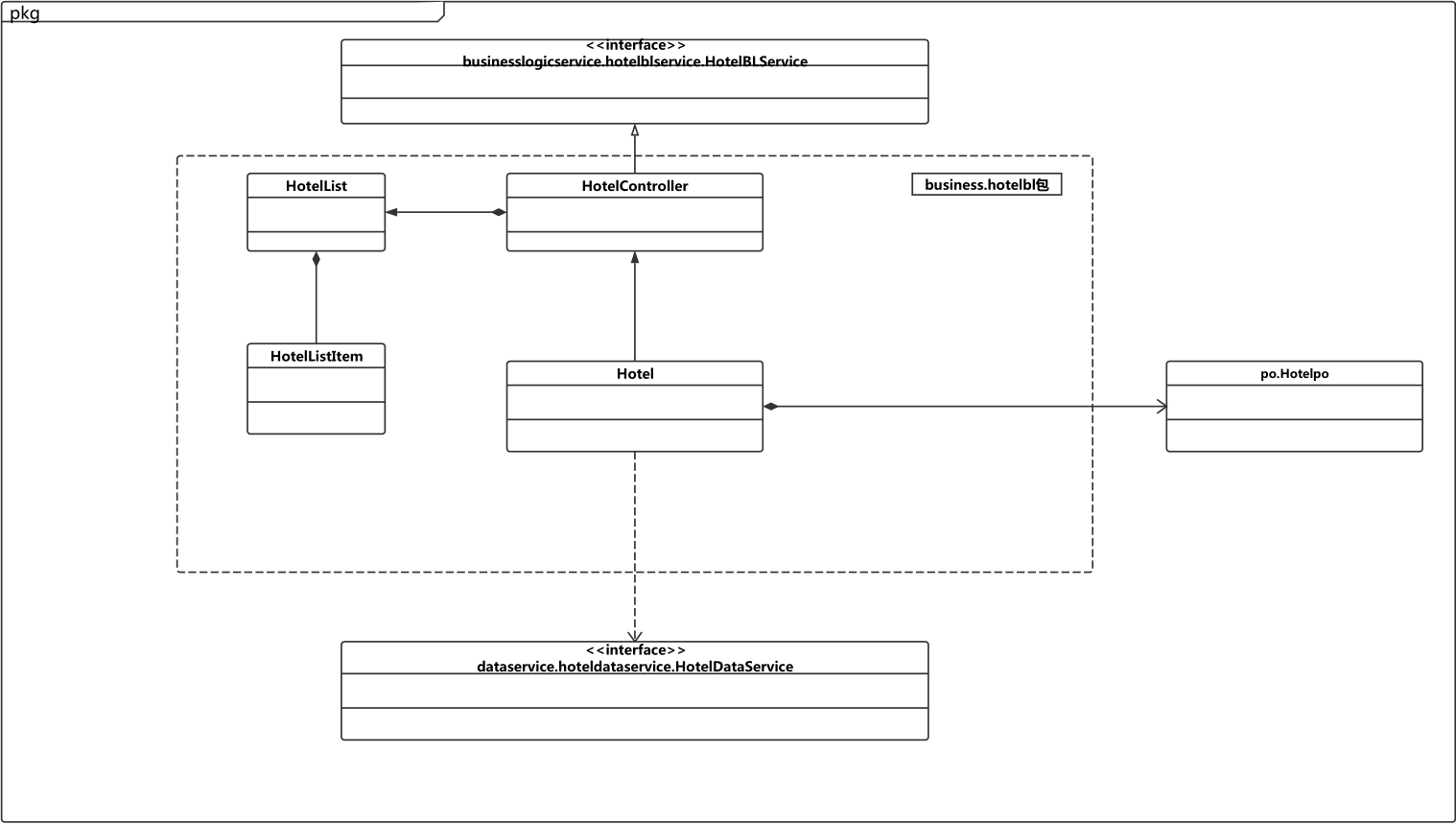
**(1) 模块概述**

Hotelbl模块负责编号，名称，价格，星级，位置评价，设施服务以及客房种类等操作。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

**(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bussinesslogicservice.HotelBLService接口。业务逻辑层和数据层之间添加dataservice.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会将对客房的逻辑业务处理委托给User对象。HotelPO是作为客房的持久化对象被添加到设计模型中去的。

HotelController模块的设计如图4.1.5.1所示



Hotelbl模块各个类的职责如表4.1.5.1所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| HotelController | 负责实现酒店界面所需要的服务 |
| Hotels | 用户的领域模型对象，拥有酒店的信息，可以帮助完成客房界面所需要的服务 |

表4.1.5.1

**(3) 模块内部类的接口规范**

HotelController和Hotels的接口规范如表4.1.5.2和表4.1.5.3所示

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| HotelController.Addhotel | 语法 | public ResultMessage Addhotel (int room) |
| 前置条件 | 已创建一个Hotelhotel领域对象，输入的酒店符合输入规范 |
| 后置条件 | 调用HotelRoom领域的Addroom方法 |
| HotelController.Updatehotel | 语法 | public void Updatehotel () |
| 前置条件 | 已创建一个Hotel领域对象 |
| 后置条件 | 调用Hotel领域的Updatehotel方法 |
| HotelController.Addprice | 语法 | public ResultMessage Addprice (int price) |
| 前置条件 | 已创建一个Hotel领域对象 |
| 后置条件 | 添加房间价格 |
| HotelController.Addstar | 语法 | public ResultMessage Addstar(int star) |
| 前置条件 | 已创建一个Hotel领域对象 |
| 后置条件 | 添加酒店星级 |
| HotelController.Addlocation | 语法 | public ResultMessage Addlocation (String location) |
| 前置条件 | 已创建一个Hotel领域对象 |
| 后置条件 | 添加酒店位置 |
| HotelController.Addsummary | 语法 | public ResultMessage Addsummary (String summary) |
| 前置条件 | 已创建一个Hotel领域对象 |
| 后置条件 | 添加酒店评价 |
| HotelController.Addservices | 语法 | public ResultMessage Addservices (String services) |
| 前置条件 | 已创建一个Hotel领域对象 |
| 后置条件 | 添加酒店设施服务信息 |
| HotelController.Addkind | 语法 | public ResultMessage Addkind (String kind) |
| 前置条件 | 已创建一个Hotel领域对象 |
| 后置条件 | 添加酒店客房类型 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| Hotel.Addhotel | 添加一个酒店 | |

表4.1.5.2 HotelController 的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Hotel.Addhotel | 语法 | public ResultMessage Addhotel (int room) |
| 前置条件 | 启动一个录入酒店的回合 |
| 后置条件 | 调用HotelRoom领域的Addroom方法 |
| Hotel.Updatehotel | 语法 | public void Updatehotel () |
| 前置条件 | 启动一个录入酒店的回合 |
| 后置条件 | 调用Hotel领域的Updatehotel方法 |
| Hotel.Addprice | 语法 | public ResultMessage Addprice (int price) |
| 前置条件 | 启动一个录入酒店的回合 |
| 后置条件 | 添加房间价格 |
| Hotel.Addstar | 语法 | public ResultMessage Addstar(int star) |
| 前置条件 | 启动一个录入酒店的回合 |
| 后置条件 | 添加酒店星级 |
| Hotel.Addlocation | 语法 | public ResultMessage Addlocation (String location) |
| 前置条件 | 启动一个录入酒店的回合 |
| 后置条件 | 添加酒店位置 |
| Hotel.Addsummary | 语法 | public ResultMessage Addsummary (String summary) |
| 前置条件 | 启动一个录入酒店的回合 |
| 后置条件 | 添加酒店评价 |
| Hotel.Addservices | 语法 | public ResultMessage Addservices (String services) |
| 前置条件 | 启动一个录入酒店的回合 |
| 后置条件 | 添加酒店设施服务信息 |
| Hotel.Addkind | 语法 | public ResultMessage Addkind (String kind) |
| 前置条件 | 启动一个录入酒店的回合 |
| 后置条件 | 添加酒店客房类型 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| HotelDataService.find(int id) | 根据房间号进行查找单一持久化对象 | |
| HotelDataService.insert(HotelPO po) | 插入单一持久化对象 | |
| HotelDataService.delete(HotelPO po) | 删除单一持久化对象 | |
| HotelDataService.update (HotelPO po) | 更新单一持久化对象 | |
| DatabaseFacory.getHotelDatabase | 得到Hotel数据库的服务的引用 | |

表4.1.5.3 Hotel 的接口规范

**(4) 业务逻辑层的动态模型**

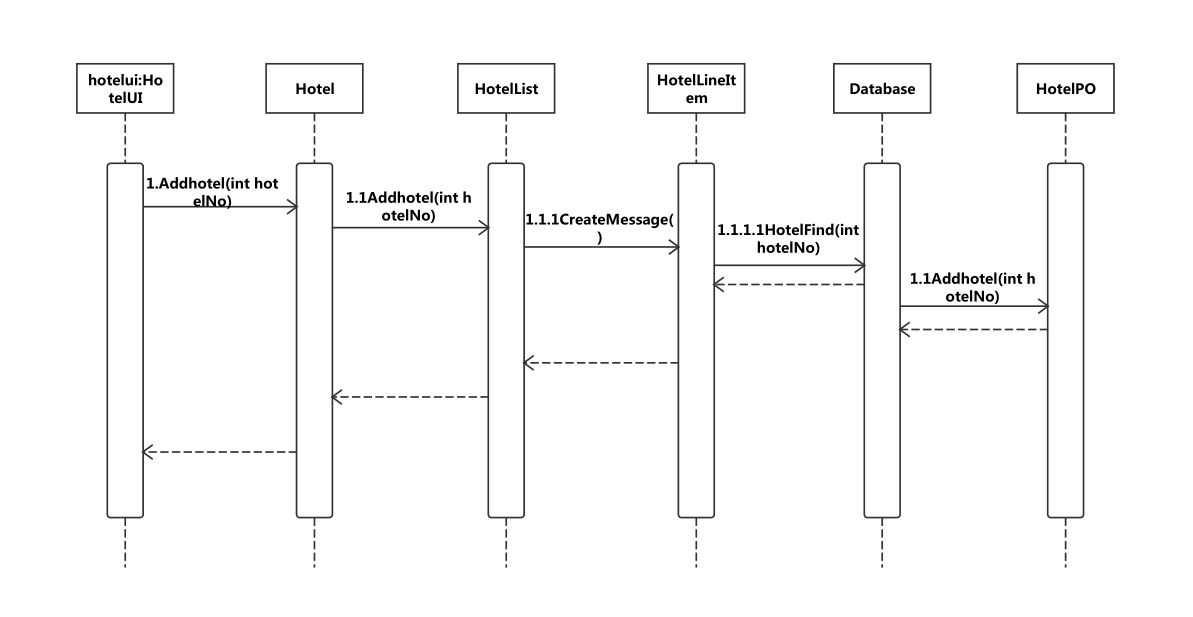


图4.1.5.2添加酒店信息的顺序图

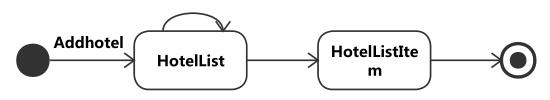


图4.1.5.3 hotel对象的状态图

### 4.1.6 HotelRoombl模块

**(1) 模块概述**

HotelRoombl模块负责客房的录入、退房处理、入住处理等操作。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

**(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bussinesslogicservice.HotelRoomBLService接口。业务逻辑层和数据层之间添加dataservice.HotelRoomDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了HotelRoomController，这样HotelRoomController会将对客房的逻辑业务处理委托给User对象。HotelRoomPO是作为客房的持久化对象被添加到设计模型中去的。

HotelRoomController模块的设计如图4.1.6.1所示

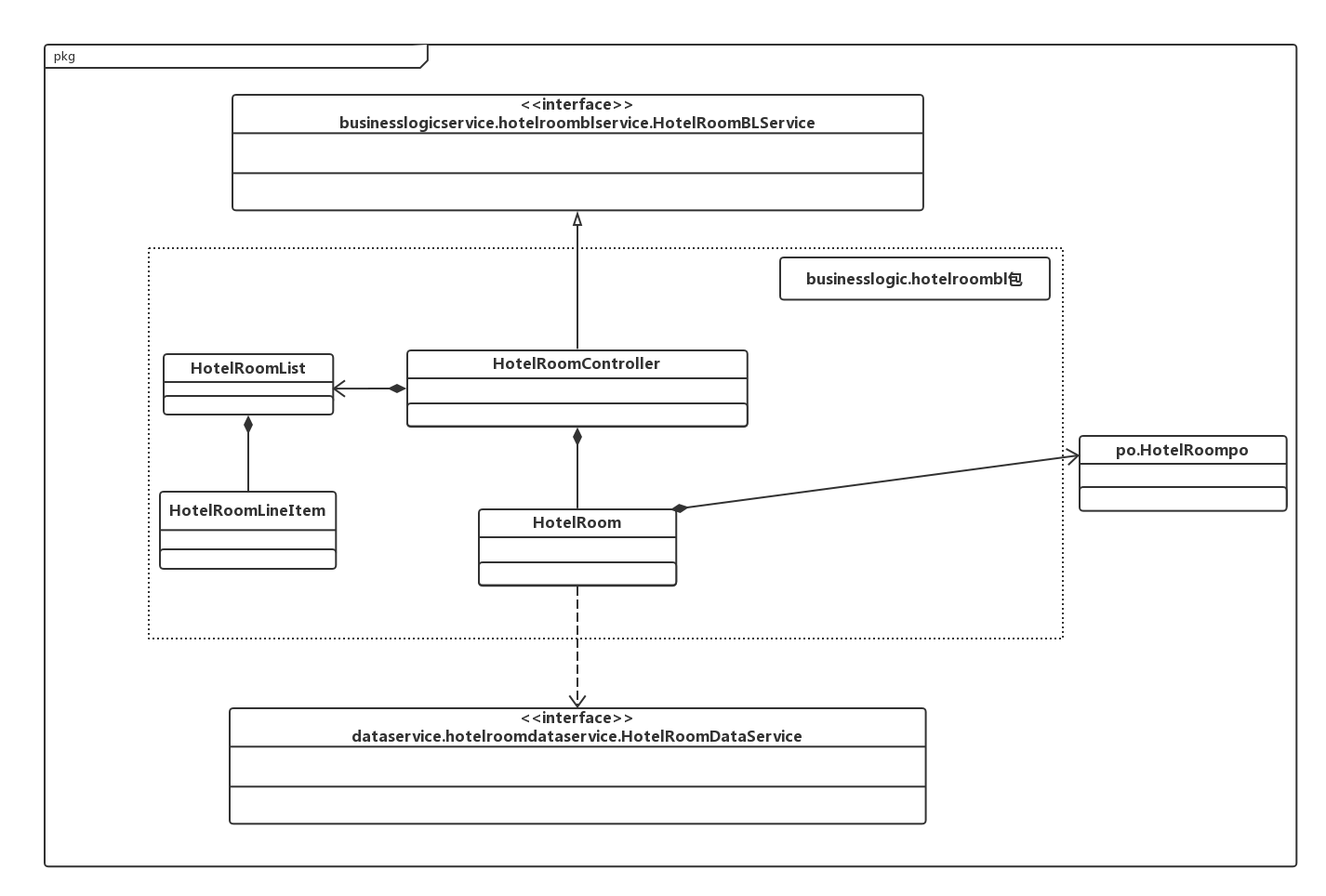


图 4.1.6.1

HotelRoombl模块各个类的职责如表4.1.6.1所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| HotelRoomController | 负责实现客房界面所需要的服务 |
| HotelRooms | 用户的领域模型对象，拥有客房的信息，可以帮助完成客房界面所需要的服务 |

表4.1.6.1

**(3) 模块内部类的接口规范**

HotelRoomController和HotelRooms的接口规范如表4.1.6.2和表4.1.6.3所示

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| HotelRoomController.Addroom | 语法 | public ResultMessage Addroom (int room) |
| 前置条件 | 已创建一个HotelRoom领域对象，输入的房间号符合输入规范 |
| 后置条件 | 调用HotelRoom领域的Addroom方法 |
| HotelRoomController.Updateroom | 语法 | public void Updateroom () |
| 前置条件 | 已创建一个HotelRoom领域对象 |
| 后置条件 | 持久化更新酒店的数据 |
| HotelRoomController.Checkin | 语法 | public ResultMessage Checkin (HotelroomVo vo) |
| 前置条件 | 已创建一个HotelRoom领域对象 |
| 后置条件 | 调用HotelRoom领域的Checkin方法 |
| HotelRoomController.Timein | 语法 | public ResultMessage Timein (Date datein) |
| 前置条件 | 已创建一个HotelRoom领域对象 |
| 后置条件 | 调用HotelRoom领域的Timein方法 |
| HotelRoomController.IDin | 语法 | public ResultMessage IDin (int id) |
| 前置条件 | 已创建一个HotelRoom领域对象，输入的id符合输入规范 |
| 后置条件 | 调用HotelRoom领域的IDin方法 |
| HotelRoomController.TimeOutpro | 语法 | public ResultMessage TimeOutpro (Date dateoutpro) |
| 前置条件 | 已创建一个HotelRoom领域对象 |
| 后置条件 | 调用HotelRoom领域的TimeOutpro方法 |
| HotelRoomController.Timeout | 语法 | public ResultMessage Timeout (Date timeout) |
| 前置条件 | 已创建一个HotelRoom领域对象 |
| 后置条件 | 调用HotelRoom领域的Timeout方法 |
| HotelRoomController.IDout | 语法 | public ResultMessage IDout (int id) |
| 前置条件 | 已创建一个HotelRoom领域对象 |
| 后置条件 | 调用HotelRoom领域的IDout方法 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| HotelRoom.Addroom | 录入客房 | |
| HotelRoom.Updateroom | 更新客房信息 | |
| HotelRoom.Checkin | 录入客房 | |
| HotelRoom.Timein | 更新房间入住的日期 | |
| HotelRoom.IDin | 记录入住房间的房间号 | |
| HotelRoom.TimeOutpro | 记录客房的预计离开时间 | |
| HotelRoom.Timeout | 更新退房日期 | |
| HotelRoom.IDout | 记录退房的房间号 | |

表4.1.6.2 HotelRoomController 的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| HotelRoom.Addroom | 语法 | public ResultMessage Addroom (int room) |
| 前置条件 | 启动一个录入客房的回合 |
| 后置条件 | 在一个录入客房的回合中，增加可用客房 |
| HotelRoom.Updateroom | 语法 | public void Updateroom () |
| 前置条件 | 启动一个录入客房的回合 |
| 后置条件 | 结束此次录入客房，持久化更新涉及的领域对象的数据 |
| HotelRoom.Checkin | 语法 | public ResultMessage Checkin (HotelroomVo vo) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化更新入住信息 |
| HotelRoom.Timein | 语法 | public ResultMessage Timein (Date datein) |
| 前置条件 | 开始更新入住信息 |
| 后置条件 | 系统更新房间入住的日期 |
| HotelRoom.IDin | 语法 | public ResultMessage IDin (int id) |
| 前置条件 | 开始更新入住信息 |
| 后置条件 | 系统记录入住房间的房间号 |
| HotelRoom.TimeOutpro | 语法 | public ResultMessage TimeOutpro (Date dateoutpro) |
| 前置条件 | 开始更新入住信息 |
| 后置条件 | 系统记录该房的预计离开时间 |
| HotelRoom.Timeout | 语法 | public ResultMessage Timeout (Date timeout) |
| 前置条件 | 开始更新退房信息 |
| 后置条件 | 系统更新退房日期 |
| HotelRoom.IDout | 语法 | public ResultMessage IDout (int id) |
| 前置条件 | 开始更新退房信息 |
| 后置条件 | 系统记录退房的房间号 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| HotelroomDataService.find(int id) | 根据房间号进行查找单一持久化对象 | |
| HotelroomDataService.find (State state) | 根据状态进行查找多个持久化对象 | |
| HotelroomDataService.insert(HotelroomPO po) | 插入单一持久化对象 | |
| HotelroomDataSerivce.delete(HotelroomPO po) | 删除单一持久化对象 | |
| HotelroomDataService.update(HotelroomPO po) | 更新单一持久化对象 | |
| DatabaseFacory.getHotelroomDatabase | 得到Hotelroom数据库的服务的引用 | |

表4.1.6.3 HotelRoom 的接口规范

**(4) 业务逻辑层的动态模型**

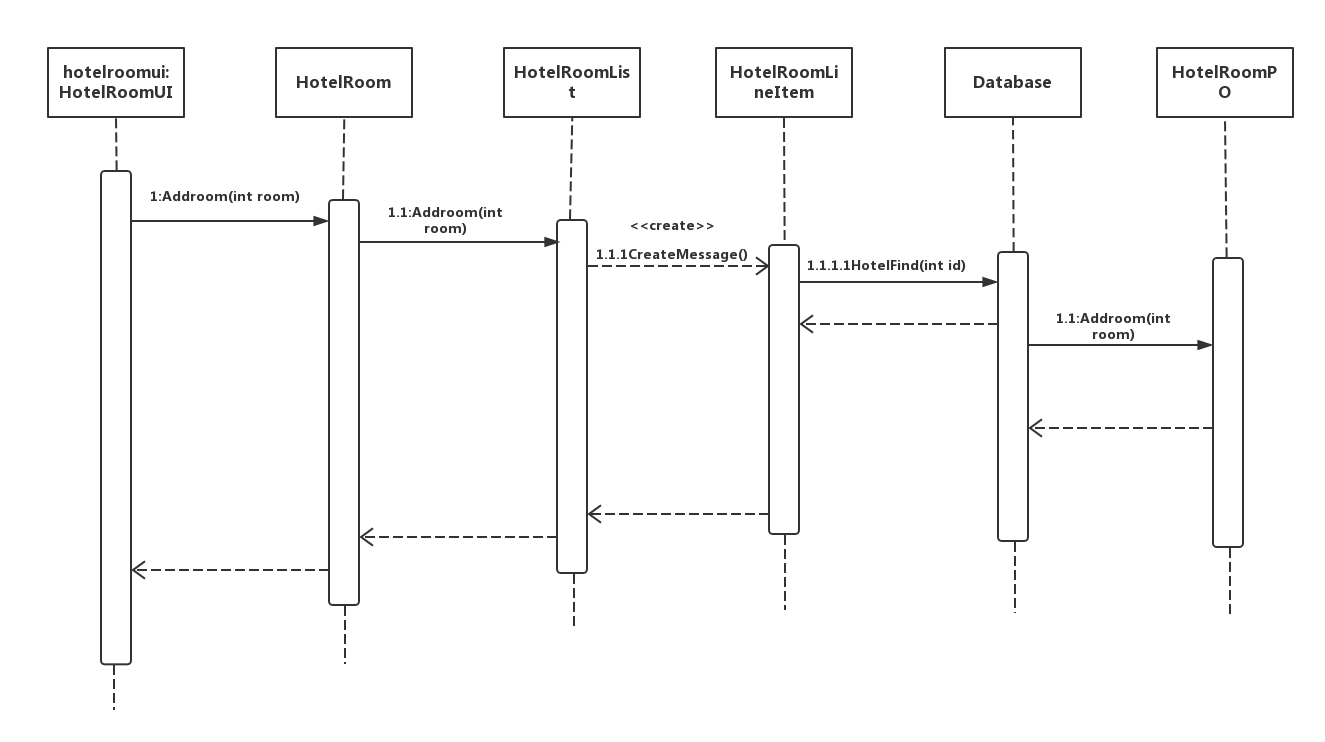
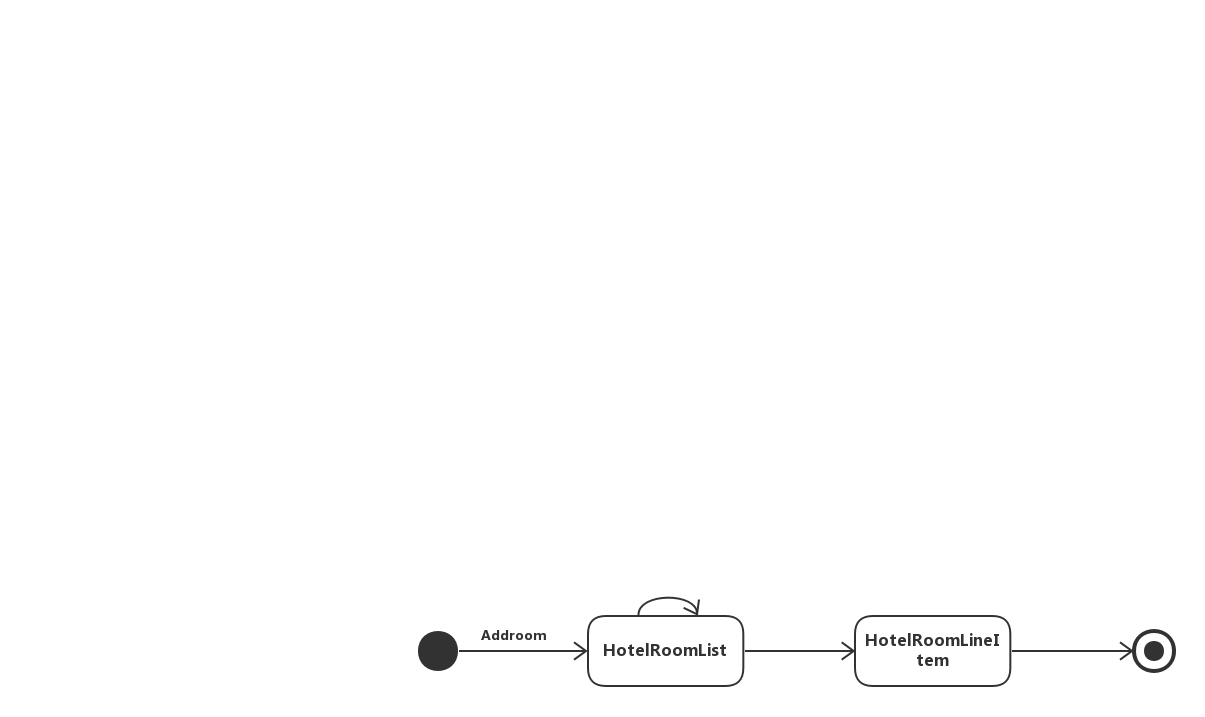
****

图4.1.6.2 添加客房的顺序图

图4.1.6.3 HotelRoom对象的状态图

### 4.1.7 Promotionbl模块

**(1) 模块概述**

Promotionbl模块负责实现对营销策略的管理。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

**(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。比如展示层和业务逻辑层之间，我们添加bussinesslogicservicePromotionBLService接口。业务逻辑层和数据层之间添加dataservice.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.Promotion Controller，这样PromotionController会将对促销的逻辑业务处理委托给Promotion对象。PromotionPO是作为营销策略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4.1.7.1所示

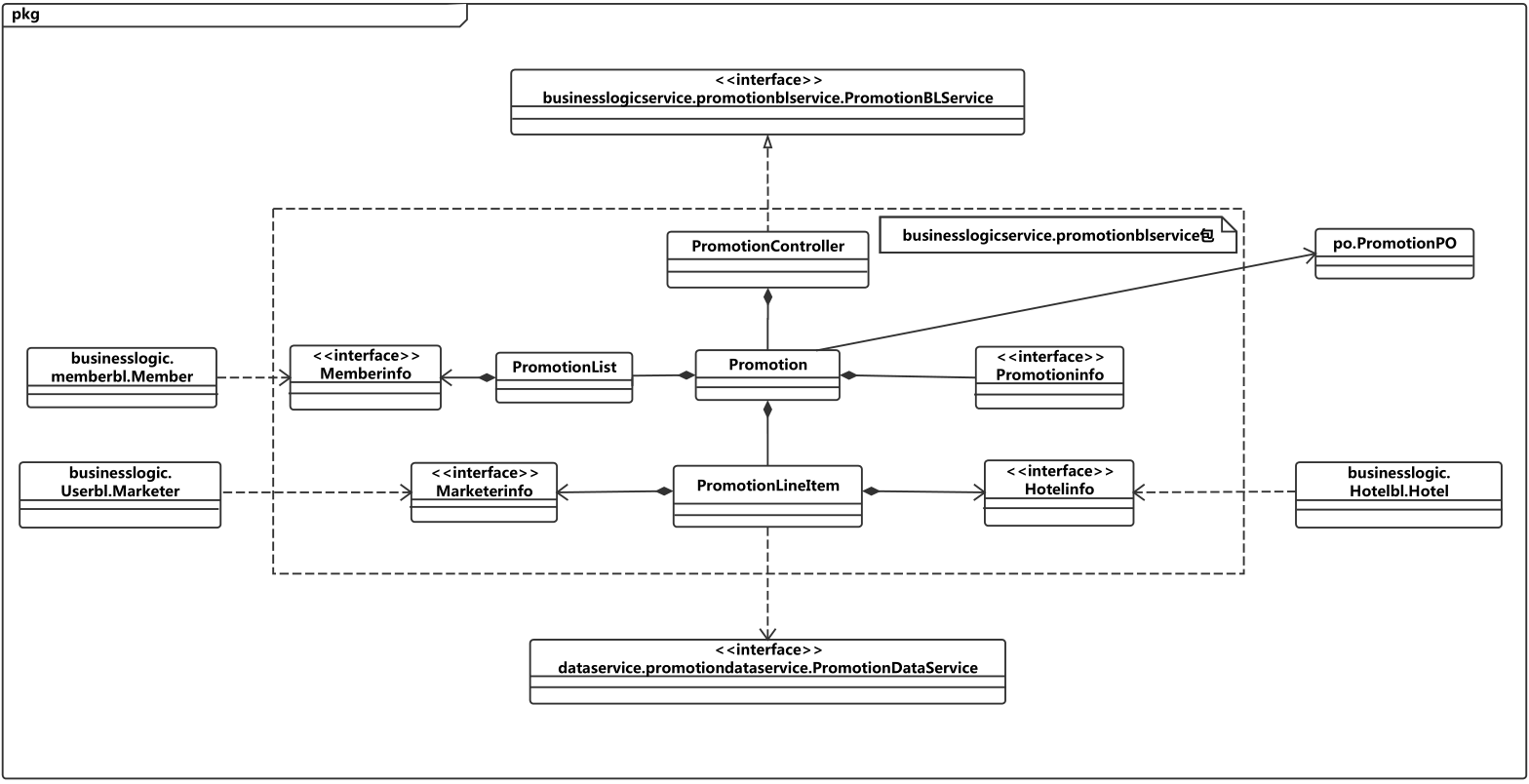


图4.1.7.1

Promotionbl模块各个类的职责如表4.1.7.1所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| PromotionController | 负责实现各营销策略管理界面所需要的服务 |
| Promotion | 营销策略的领域模型对象，拥有营销策略名称内容，可以帮助完成制定营销策略、根据营销策略内容确定价格所需要的服务 |

表4.1.7.1

**(3) 模块内部类的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| PromotionController.judge | 语法 | public int judge (int id) |
| 前置条件 | 已创建一个Promtion领域对象 |
| 后置条件 | 调用Promotion领域对象的judge方法 |
| PromotionController.createNewItem | 语法 | public void createNewItem(promotionVo vo) |
| 前置条件 | 已创建一个Promtion领域对象，并且已确定操作属于酒店工作人员或是网站营销人员 |
| 后置条件 | 调用Promotion领域对象的createNewItem方法  Judge改成一个 |
| Promotion Controller.Del | 语法 | public void Del(promotionVO vo) |
| 前置条件 | 已创建一个Promtion领域对象，并且存在营销策略 |
| 后置条件 | 调用Promotion领域对象的Del方法 |
| PromotionController. Edit | 语法 | public void update (promotionVO vo) |
| 前置条件 | 已创建一个Promtion领域对象，并且存在营销策略 |
| 后置条件 | 调用Promotion领域对象的Edit方法 |
| PromotionController.getLineItem | 语法 | Public list getLineItem（String itemName） |
| 前置条件 | 已创建一个Promtion领域对象，并且存在营销策略 |
| 后置条件 | 调用Promotion领域对象的getListItem方法 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| PromotionController.addHotel | 加入一个Hotel对象 | |
| PromotionController.addMarketer | 加入一个Marketer对象 | |

表4.1.7.2 PromotionController的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Promotion. judge | 语法 | public int judge (int id) |
| 前置条件 | 启动对营销策略的管理 |
| 后置条件 | 确定营销策略表所属为酒店还是网站 |
| Promotion.createNewItem | 语法 | public void createNewItem(promotionVo vo) |
| 前置条件 | 已确定操作属于酒店工作人员或是网站营销人员 |
| 后置条件 | 增加营销策略 |
| Promotion.Del | 语法 | public promotionVO Del(promotionVO vo) |
| 前置条件 | 已添加营销策略 |
| 后置条件 | 系统更新营销策略列表 |
| Promotion.Edit | 语法 | public promotionVO update (promotionVO vo) |
| 前置条件 | 已添加营销策略 |
| 后置条件 | 系统更新营销策略列表 |
| Promotion.getLineItem | 语法 | public list getLineItem（String itemName） |
| 前置条件 | 已添加营销策略 |
| 后置条件 | 返回营销策略信息 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| DatabaseFacory.getPromotionDatabase | 得到promotion数据库的服务的引用 | |
| PromotionDataService.insert(PromotionPO po) | 插入单一持久化对象 | |
| PromotionDataService.delete(PromotionPO po) | 删除单一持久化对象 | |
| PromotionDataService.show(PromotionPO po) | 得到所有的单一持久化对象 | |
| PromotionDataService.update(PromotionPO po) | 更新单一持久化对象 | |

表4.1.7.3 Promotion的接口规范

**(4) 业务逻辑层的动态模型**

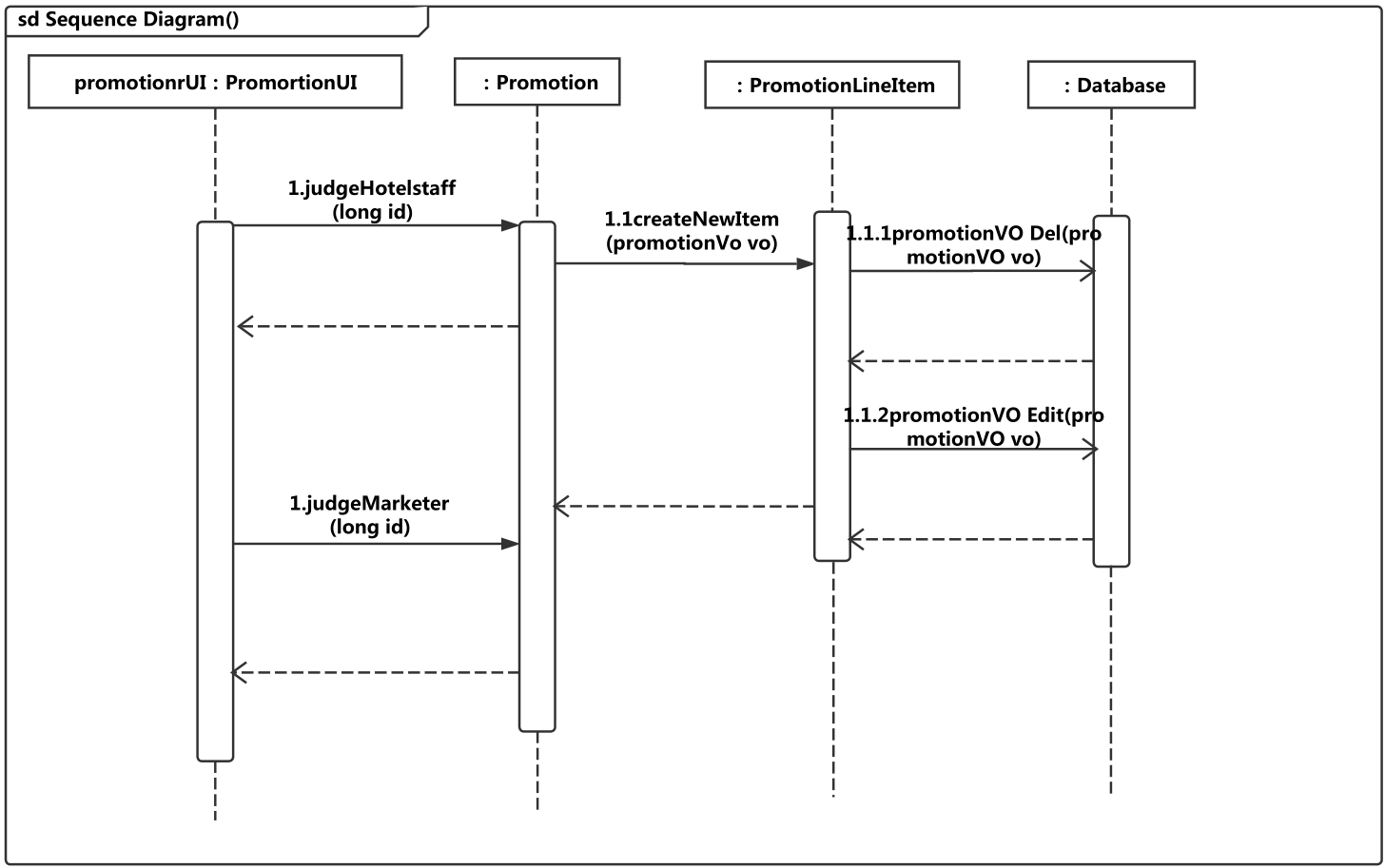


图4.1.7.2 制定营销策略的顺序图

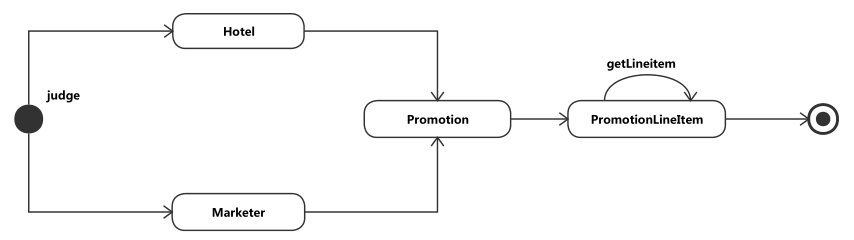


图4.1.7.3 promotion对象的状态图

# 5.依赖视角

图5.1和图5.2是客户端和服务器端各自的包之间的依赖关系。

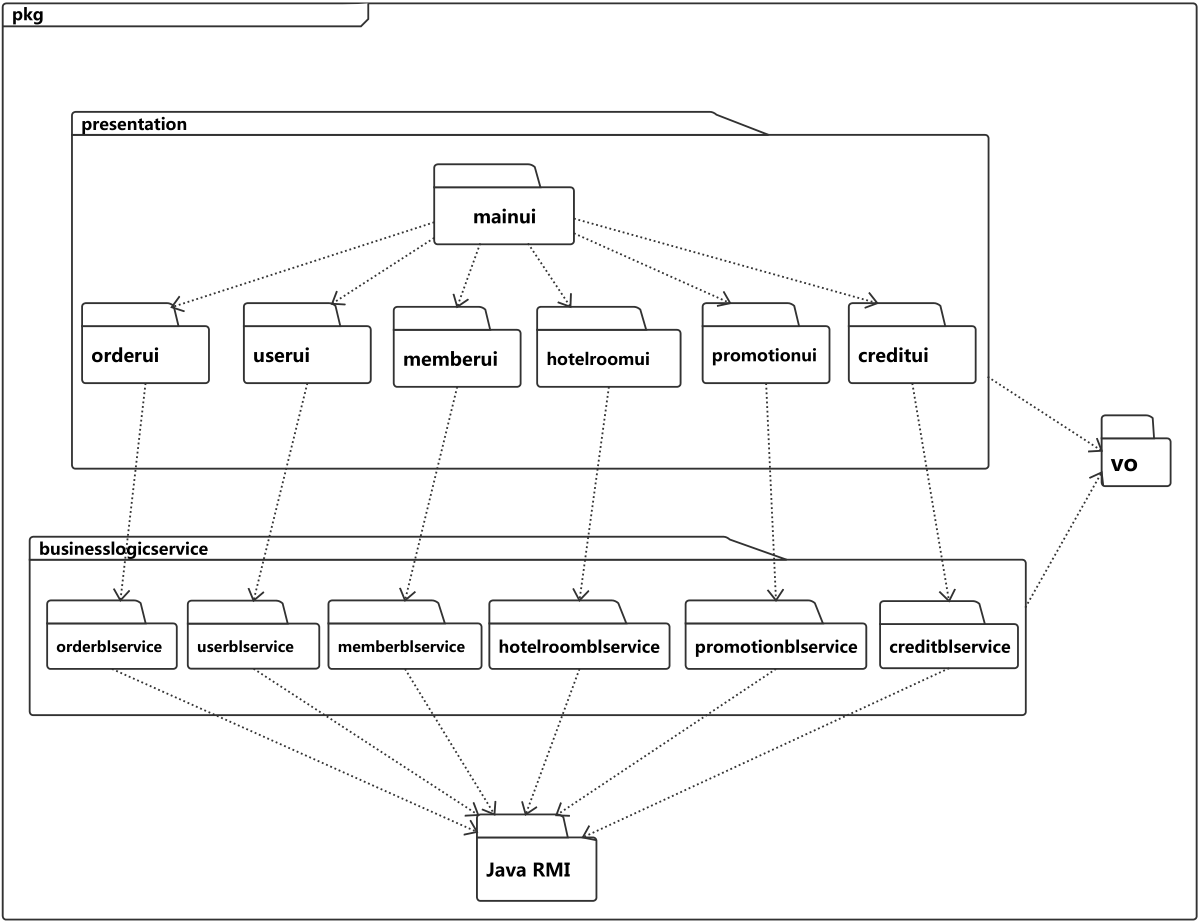


图5.1

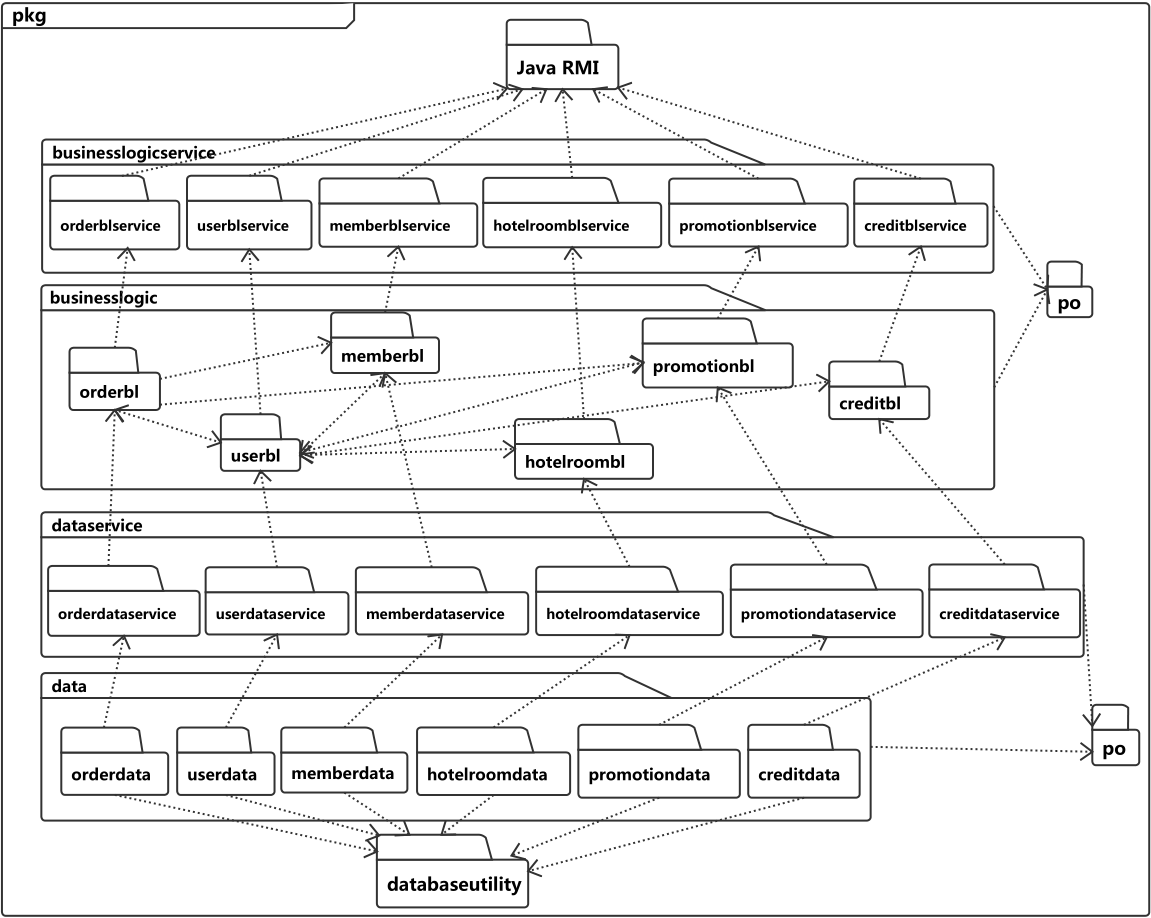


图5.2