酒店预订系统

详细设计描述文档 v2.0

南京大学软件学院

45组

刘宇翔 刘伟 刘宗侃 刘铮

2016.10.29

#### 1 / 84

变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 全体成员 | 2016.10.29 | 最初草稿 | V1.0 |
| 刘铮 | 2016.11.6 | 增加逻辑层内部类的接口规范 | V2.0 |
| 刘宇翔 | 2016.12.22 | 修改订单的逻辑层 | V2.1 |

[1 / 84 2](#_Toc465523185)

[1.引言 6](#_Toc465523186)

[1.1.编制目的 6](#_Toc465523187)

[1.2.词汇表 6](#_Toc465523188)

[1.3.参考资料 6](#_Toc465523189)

[2.产品描述 6](#_Toc465523190)

[3.体系结构设计概述 7](#_Toc465523191)

[4.结构视角 7](#_Toc465523192)

[4.1.业务逻辑层的分解 7](#_Toc465523193)

[4.1.1. hotelbl 模块 7](#_Toc465523194)

[4.1.1.1. 模块概述 7](#_Toc465523195)

[4.1.1.2. 整体结构 8](#_Toc465523196)

[4.1.1.3. 模块内部类的接口规范 10](#_Toc465523197)

[4.1.1.4. 业务逻辑层的动态模型 18](#_Toc465523198)

[4.1.2. roombl 模块 28](#_Toc465523199)

[4.1.2.1. 模块概述 28](#_Toc465523200)

[4.1.2.2. 整体结构 28](#_Toc465523201)

[4.1.2.3. 模块内部类的接口规范 30](#_Toc465523202)

[4.1.2.4. 业务逻辑层的动态模型 33](#_Toc465523203)

[4.1.3. orderbl 模块 39](#_Toc465523204)

[4.1.3.1. 模块概述 39](#_Toc465523205)

[4.1.3.2. 整体结构 40](#_Toc465523206)

[4.1.3.3. 模块内部类的接口规范 42](#_Toc465523207)

[4.1.3.4. 业务逻辑层的动态模型 53](#_Toc465523208)

[4.1.4. promotionbl 模块 74](#_Toc465523209)

[4.1.4.1. 模块概述 74](#_Toc465523210)

[4.1.4.2. 整体结构 75](#_Toc465523211)

[4.1.4.3. 模块内部类的接口规范 77](#_Toc465523212)

[4.1.4.4. 业务逻辑层的动态模型 79](#_Toc465523213)

[4.1.5. userbl 模块 85](#_Toc465523214)

[4.1.5.1. 模块概述 85](#_Toc465523215)

[4.1.5.2. 整体结构 86](#_Toc465523216)

[4.1.5.3. 模块内部类的接口规范 90](#_Toc465523217)

[4.1.5.4. 业务逻辑层的动态模型 101](#_Toc465523218)

[5.依赖视角 115](#_Toc465523219)

目录

# 1.引言

##### 1.1.编制目的

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

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

##### 1.2.词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| HRS | 酒店预订系统 | 无 |

##### 1.3.参考资料

1. 酒店预订系统用例文档 v2.0
2. 酒店预订系统需求规格说明文档 v2.0
3. 酒店预订系统软件体系结构设计文档 v3.0

# 2.产品描述

参考酒店预订系统用例文档和酒店预订系统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

参考酒店预订系统软件体系结构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设计文档图 4-1-1。

###### 4.1.1. hotelbl 模块

4.1.1.1. 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系统结构设计文档。

4.1.1.2. 整体结构

根据体系结构的设计，酒店预订系统选择分层体系结构风格，将系统分为 3 层（展示层、业务逻辑层、数据层）,很好地示意了整个高层抽象。展示层包含

GUI 页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。每一层之间为了增加灵活性，添加了接口，比如在展示层和业务逻辑层之间添加 blservice.hotelblservice.HotelblService 接口，在业务逻辑层和数据层之间添加 dataservice.hoteldataservice.HotelDataService 接口。

HotelPO 是作为酒店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 4-1-1-1 所示。

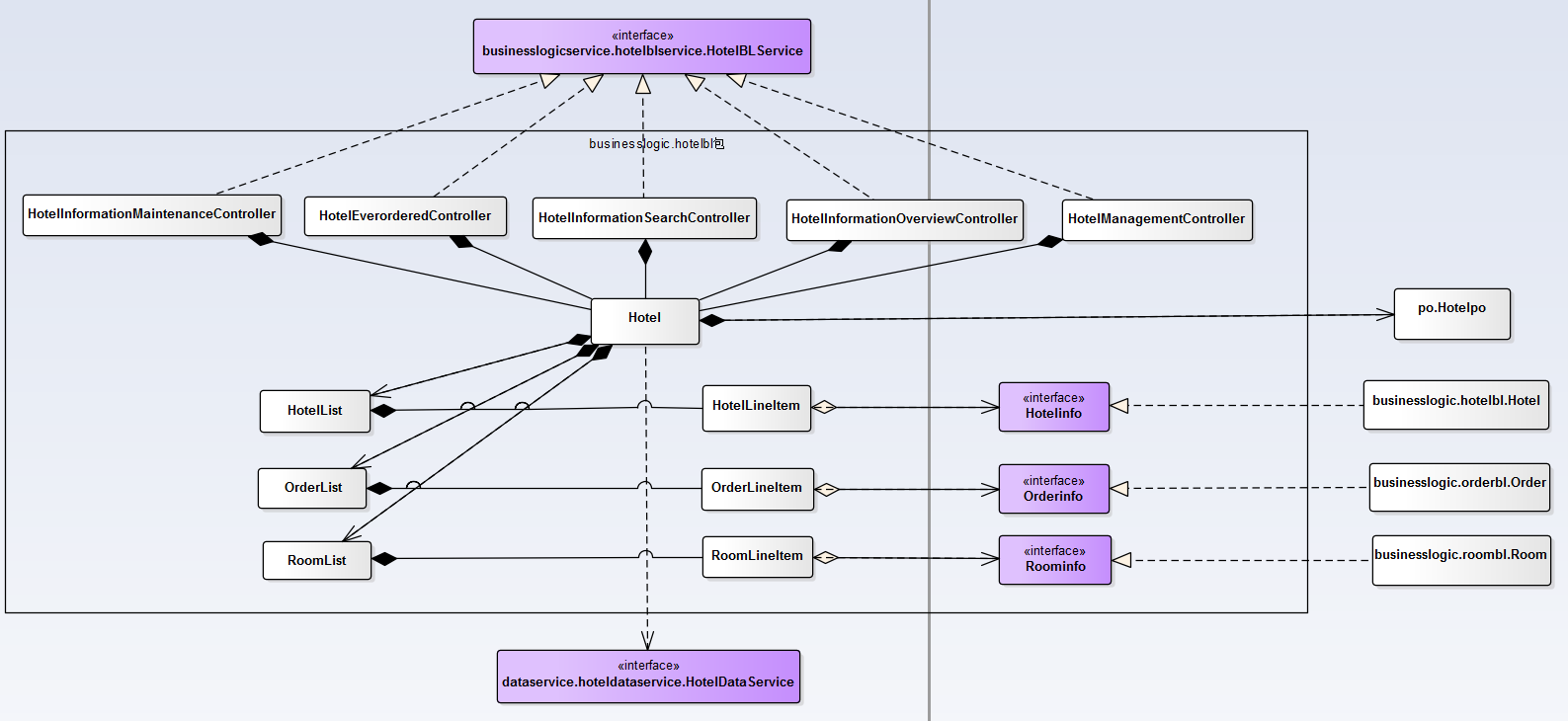


图 4-1-1-1 hotelbl 模块各个类的设计 hotelbl 模块各个类的职责如表 4-1-1-1 所示。

表 4-1-1-1 hotelbl 模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| Hotel | 酒店的领域模型对象，拥有酒店数据的帐号、名称、地址、商圈、星级、评分、简介、设施服务、联系方式，可以帮助完成查找酒店、预定酒店、评价酒店的任务 |
| HotelInformationMaintenanceController | 负责维护酒店的基本信息和更新酒店数据所需要的服务 |
| HotelEverorderedController | 负责查找酒店和用户的历史订单的酒店并返回历史酒店列表的服务 |
| HotelInformationSearchController | 负责查找符合条件的酒店的服务 |
| HotelInformationOverviewController | 负责显示酒店的详细信息的服务 |
| HotelManagementController | 负责酒店帐号管理的服务 |
| HotelList | 获取所有酒店列表 |
| OrderList | 获取所有订单列表 |
| RoomList | 获取所有房间列表 |
| HotelLineItem | 单个酒店的信息 |
| OrderLineItem | 单个订单的信息 |
| RoomLineItem | 单个房间的信息 |
| Hotelinfo | 负责获得酒店的对应信息的服务 |
| Orderinfo | 负责获得订单的对应信息的服务 |
| Roominfo | 负责获得房间的对应信息的服务 |

4.1.1.3. 模块内部类的接口规范

Hotelbl 的接口规范如表 4-1-1-2 所示。

Hotel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Hotel.messagelook | 语法 | public ResultMessage messagelook(HotelVO hvo) |
| 前置条件 | 用户已登录成功 |
| 后置条件 | 查找地址和商圈内的酒店，返回范围内的酒店信息 |
| Hotel.messagesearch | 语法 | public ResultMessage messagesearch(MessageInput in) |
| 前置条件 | 用户已登录成功 |
| 后置条件 | 查找符合条件的酒店并返回酒店列表 |
| Hotel.historylook | 语法 | public ResultMessage historylook(String id) |
| 前置条件 | 用户已登录成功 |
| 后置条件 | 查找用户的历史订单的酒店并返回历史酒店列表 |
| Hotel.messagemaintain | 语法 | public ResultMessage messagemaintain(MessageInput in) |
| 前置条件 | 酒店工作人员已登陆 |
| 后置条件 | 系统修改酒店的基本信息 |
| Hotel.accountadmin | 语法 | public ResultMessage accuntadmin(MessageInput in) |
| 前置条件 | 网站管理人员已登录 |
| 后置条件 | 系统增加酒店帐号和名称以及该酒店工作人员的帐号 |
| Hotel.setscore | 语法 | public boolean setscore(int score) |
| 前置条件 | 用户对酒店作出评分 |
| 后置条件 | 系统增加评分记录，更新评分的均值 |
| Hotel.setcomment | 语法 | public boolean setcomment(String comment) |
| 前置条件 | 用户对酒店作出评价 |
| 后置条件 | 系统增加酒店的评价记录 |
| Hotel.getroominfo | 语法 | public boolean getroominfo(HotelVO vo) |
| 前置条件 | 用户要查看酒店详细信息 |
| 后置条件 | 系统显示剩余房间 |
| Hotel.pricesort | 语法 | public HotelVO pricesort(ArrayList<Hotel> ah) |
| 前置条件 | 客户要求酒店按价格高低排序 |
| 后置条件 | 显示价格由低到高的酒店列表 |
| Hotel.starsort | 语法 | public HotelVO starsort(ArrayList<Hotel> ah) |
| 前置条件 | 客户要求酒店按星级排序 |
| 后置条件 | 显示按照星级排序的酒店列表 |
| Hotel.scoresort | 语法 | public HotelVO scoresort(ArrayList<Hotel> ah) |
| 前置条件 | 客户要求酒店按平均评分排序 |
| 后置条件 | 显示平均评分由高到低的酒店列表 |
| Hotel.gethistoryorder | 语法 | public HotelVO gethistoryorder(Hotel h) |
| 前置条件 | 客户要求查看酒店细节 |
| 后置条件 | 系统显示此客户在该酒店的历史订单 |
| Hotel.gethistoryhotel | 语法 | public HotelVO gethistoryhotel() |
| 前置条件 | 客户要求查看预定过的酒店 |
| 后置条件 | 系统显示此客户的历史预定酒店 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| DatabaseFactroy.getHotelDatabase | 得到Hotel数据库的服务的引用 |
| DatabaseFactroy.getRoomDatabase | 得到Room数据库的服务的引用 |
| HotelDataService.find(String location) | 在数据区寻找符合在区域中的酒店 |
| HotelDataService.search(MessageInput in) | 在数据区寻找符合条件的酒店 |
| DatabaseFactroy.getOrderDatabase | 得到Order数据库的服务的引用 |
| HotelDataService.getHistory(HotelPO po) | 在数据区寻找该酒店的历史订单 |
| 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(MessageInput in) | 更新单一持久化对象 |

Promotion的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Promotion.madebyhotel | 语法 | public ResultMessage madebyhotel (PromotionVO vo) |
| 前置条件 | 酒店工作人员已登录成功 |
| 后置条件 | 系统录入此酒店营销策略，并发布 |
| Promotion.madebyweb | 语法 | public ResultMessage madebyweb (PromotionVO vo ) |
| 前置条件 | 网站营销人员已登录成功 |
| 后置条件 | 系统录入此网站营销策略，并发布 |
| Promotion.memberlevelmade | 语法 | public ResultMessage memberlevelmade(PromotionVO vo) |
| 前置条件 | 网站营销人员已登录成功 |
| 后置条件 | 系统记录会员等级标准 |
| Promotion.cancel | 语法 | public boolean cancel(Promotion promotion) |
| 前置条件 | 客户已登录 |
| 后置条件 | 在系统中取消此订单 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| DatabaseFactroy.getPromotionDatabase | 得到Promotion数据库的服务的引用 |
| DatabaseFactroy.getHotelDatabase | 得到Hotel数据库的服务的引用 |
| DatabaseFactroy.getOrderDatabase | 得到Order数据库的服务的引用 |
| DatabaseFactroy.getUserDatabase | 得到User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.memberlevelinsert(MessageInput in) | 在数据区添加会员等级制度 |
| PromotionDataService.memberlevelupdate(MessageInput in) | 在数据区更新会员等级制度 |

表 4-1-1-2 Hotelbl 的接口规范

4.1.1.4. 业务逻辑层的动态模型

图 4-1-1-2 表明了酒店预订系统中，在增加酒店的过程中输入酒店信息

之后，账户管理业务逻辑处理的相关对象之间的协作。

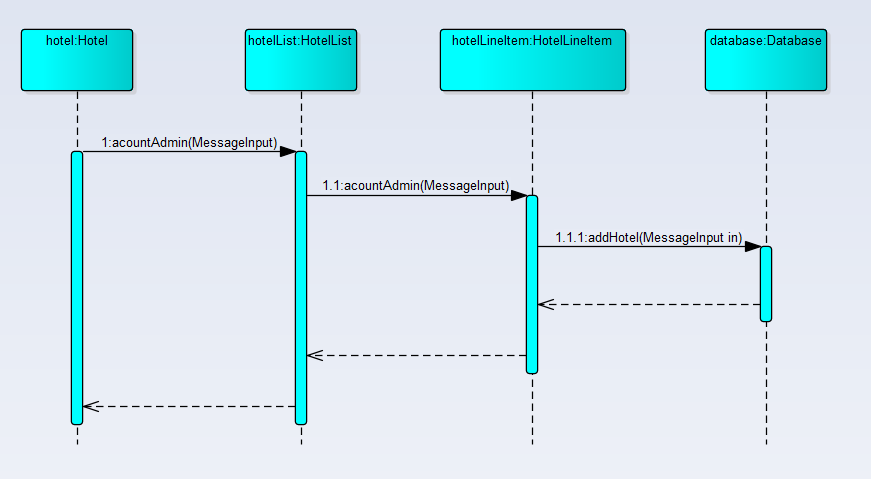
 图 4-1-1-2 增加酒店的顺序图

图 4-1-1-3 表明了酒店预订系统中，在查看历史酒店的过程中输入一个ID

之后，查看历史酒店业务逻辑处理的相关对象之间的协作。

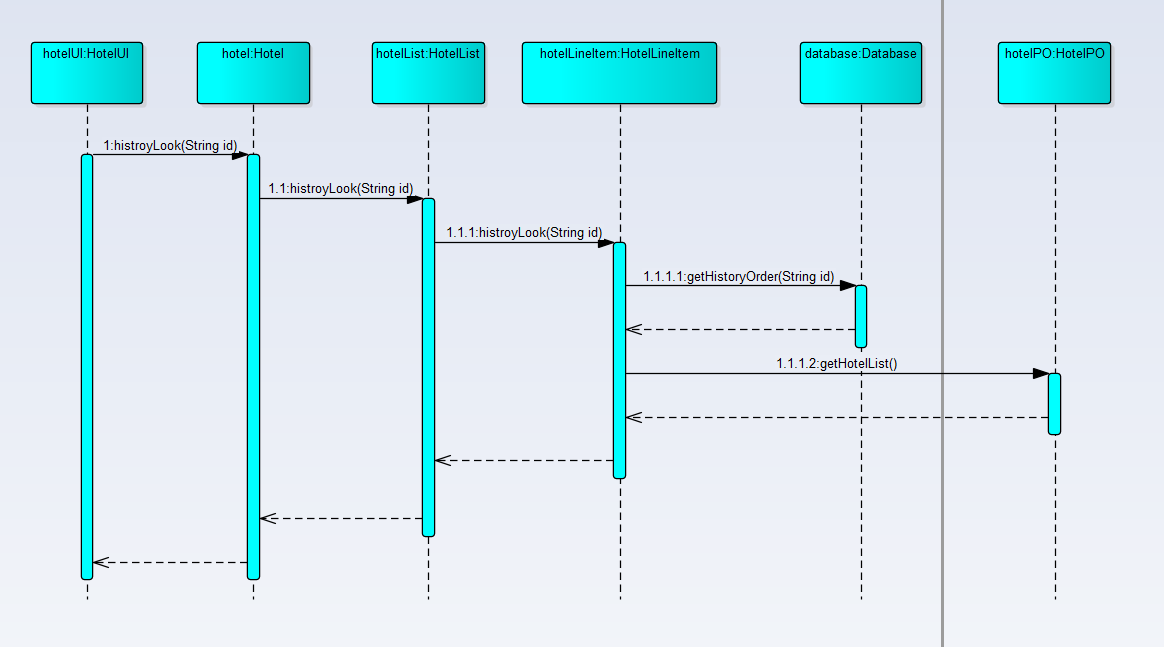
 图 4-1-1-3 查看历史酒店的顺序图

图 4-1-1-4 表明了酒店预订系统中，在查看酒店细节的过程中输入一个酒店VO

之后，查看历史订单业务逻辑处理的相关对象之间的协作。

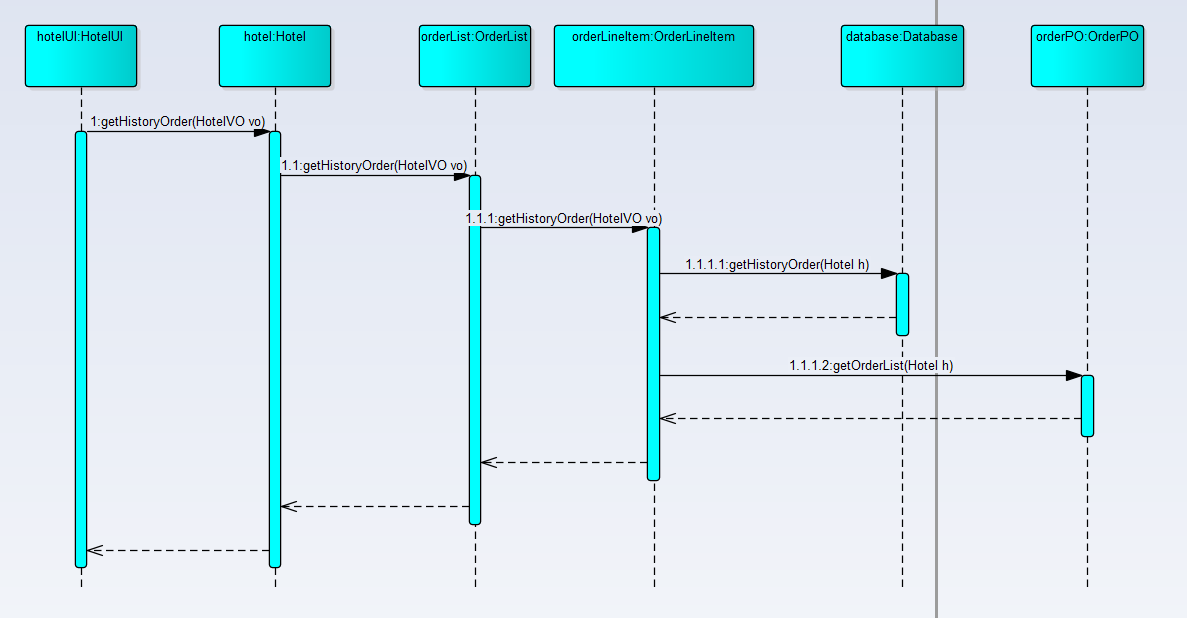
 图 4-1-1-4 查看历史订单的顺序图

图 4-1-1-5表明了酒店预订系统中，在查看房间信息的过程中输入一个酒店 VO

之后，查看房间信息业务逻辑处理的相关对象之间的协作。

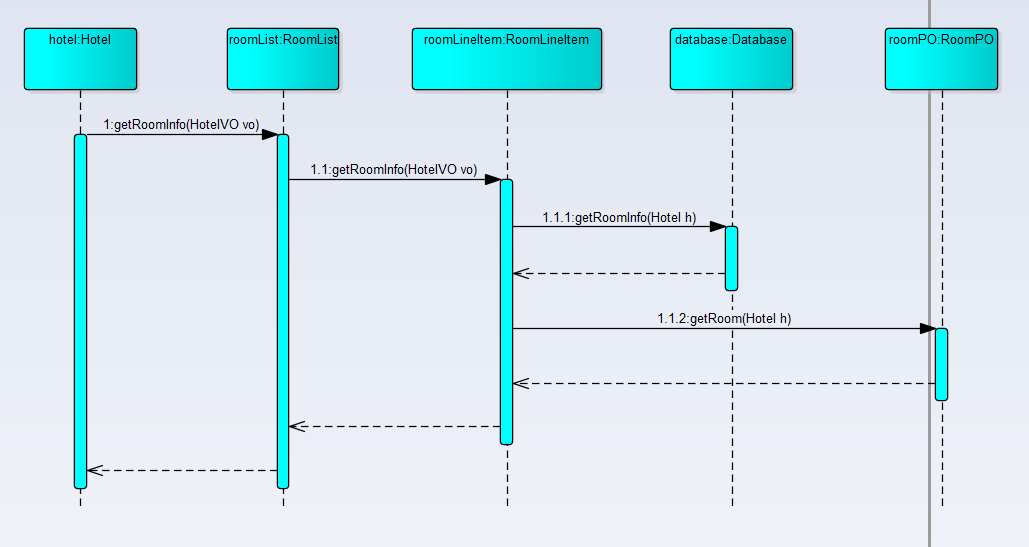


图 4-1-1-5 查看房间信息的顺序图

图4-1-1-6表明了酒店预订系统中，在浏览酒店的过程中输入一个酒店VO

之后，浏览酒店业务逻辑处理的相关对象之间的协作。

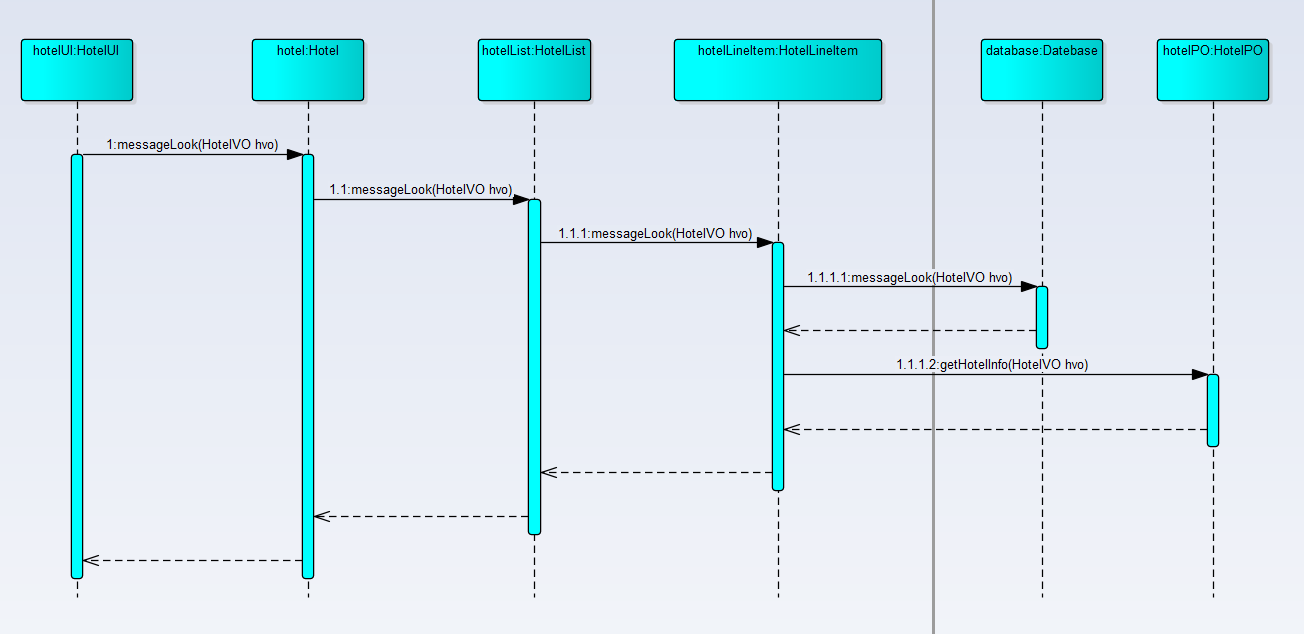


图4-1-1-6浏览酒店的顺序图

图4-1-1-7表明了酒店预订系统中，在维护酒店基本信息的过程中输入修改信息

之后，维护酒店业务逻辑处理的相关对象之间的协作。

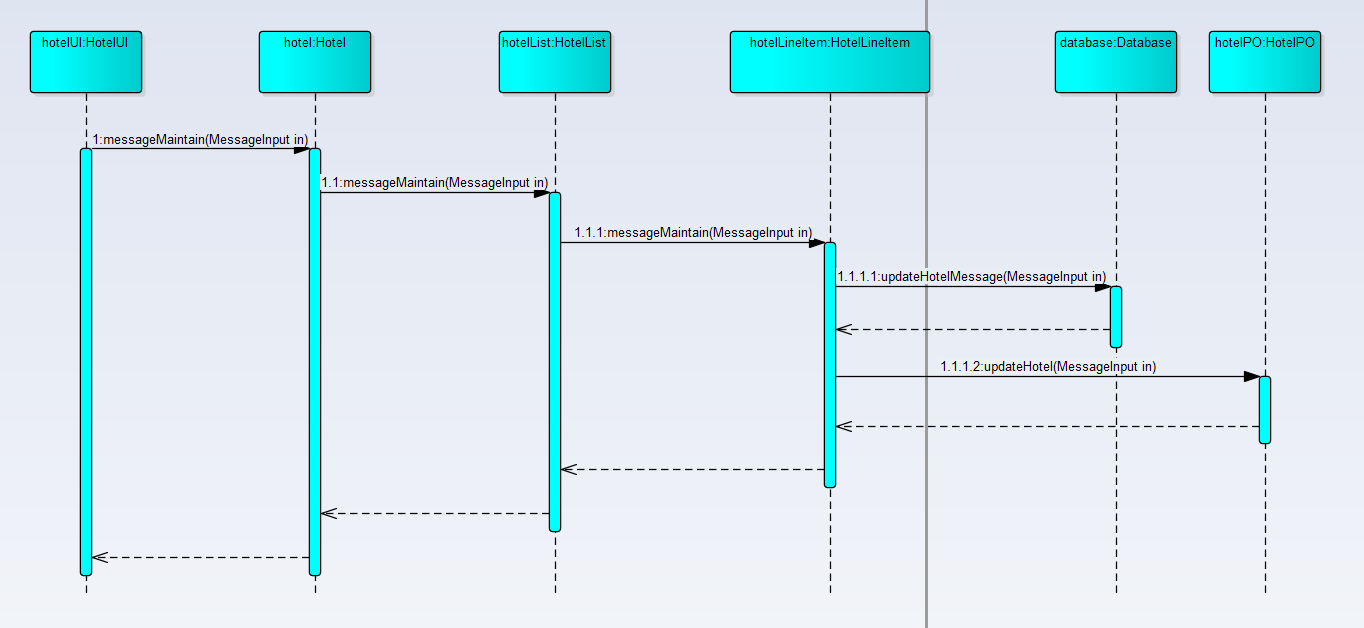


图4-1-1-7维护酒店基本信息的顺序图

图4-1-1-8表明了酒店预订系统中，在浏览酒店的过程中输入一个查找范围

之后，搜索酒店业务逻辑处理的相关对象之间的协作。

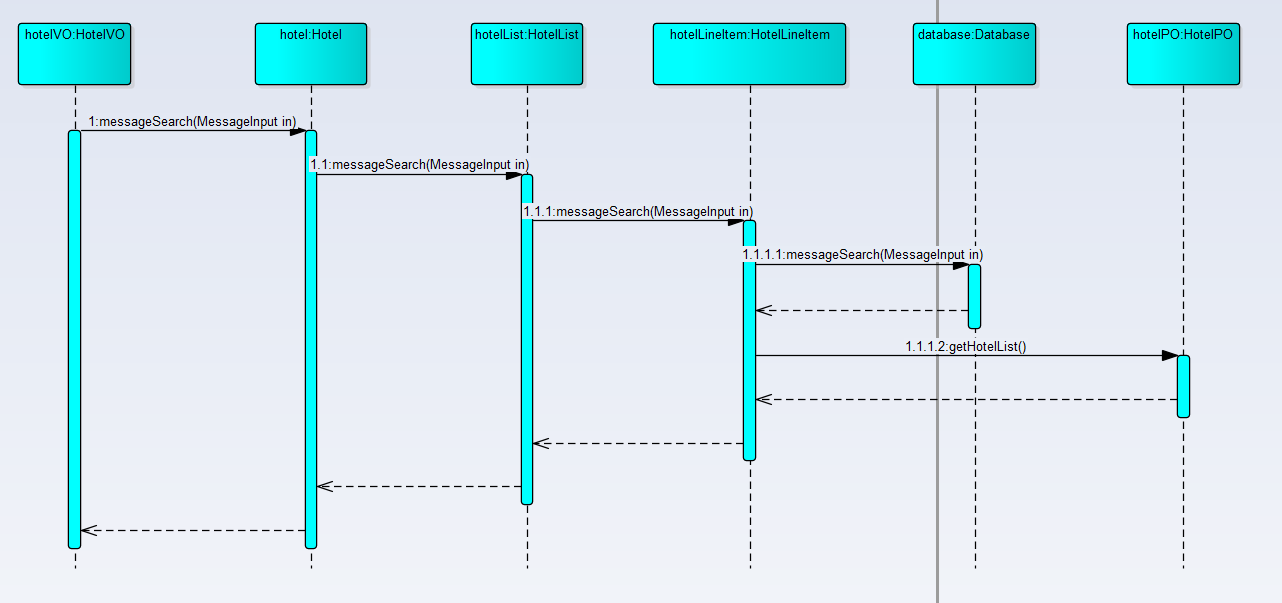


图4-1-1-8搜索酒店的顺序图

图4-1-1-9表明了酒店预订系统中，在按价格由高到低排序的过程中，酒店排序业务逻辑处理的相关对象之间的协作。

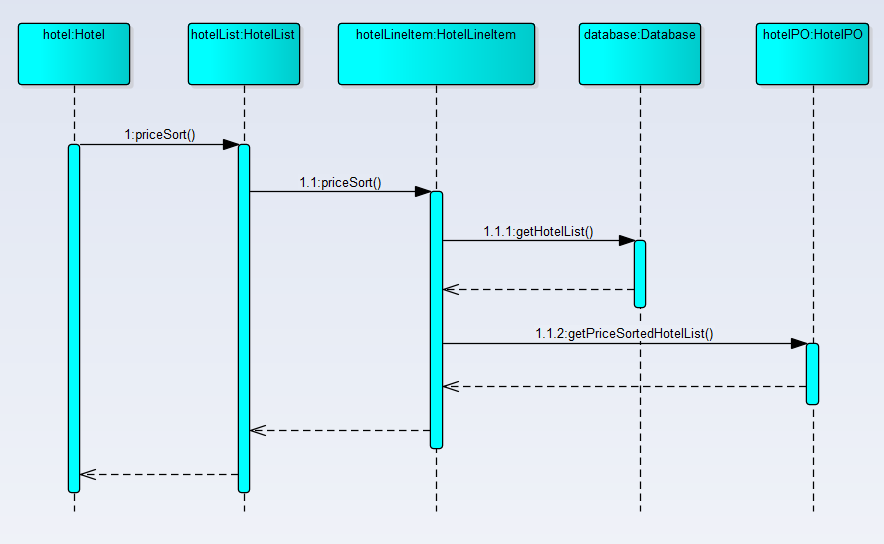


图4-1-1-9酒店价格由高到低排序的顺序图

图4-1-1-10表明了酒店预订系统中，在将酒店按分数由高到低的过程中，酒店排序业务逻辑处理的相关对象之间的协作。

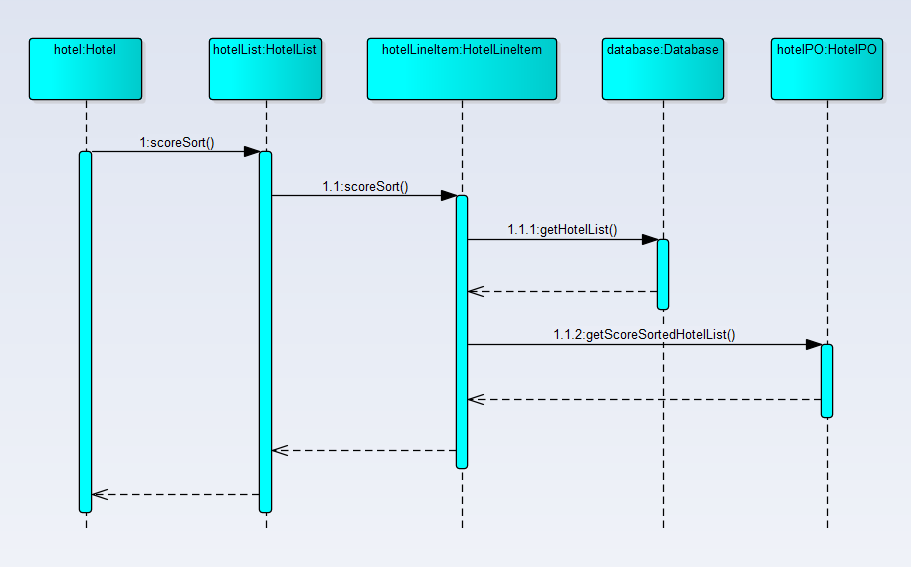


图4-1-1-10酒店评分由高到低排序的顺序图

图4-1-1-11表明了酒店预订系统中，在增加客户评价的过程中输入一个评价

之后，增加客户评价业务逻辑处理的相关对象之间的协作。

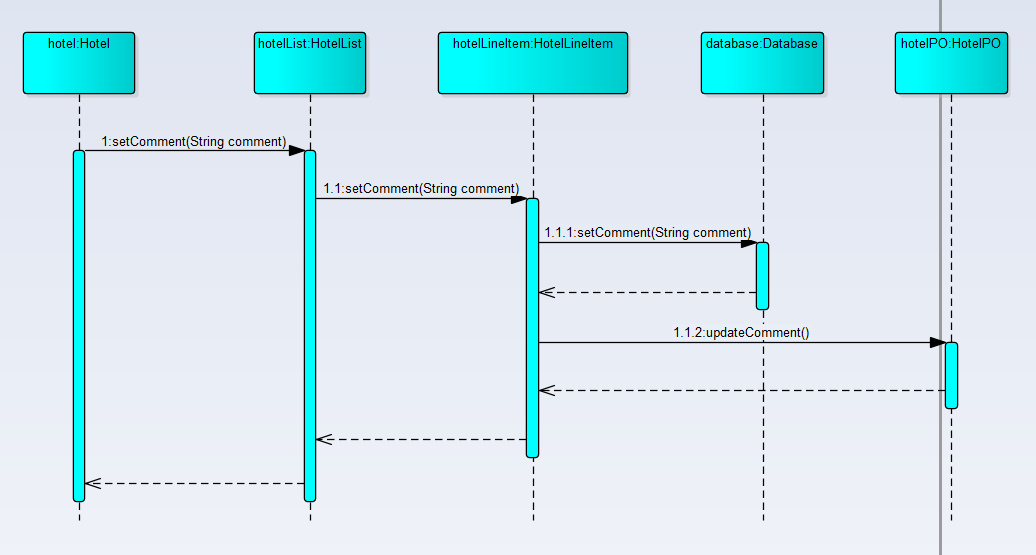


图4-1-1-11增加客户评价的顺序图

图4-1-1-12表明了酒店预订系统中，在增加评分的过程中输入一个分数

之后，增加评分业务逻辑处理的相关对象之间的协作。

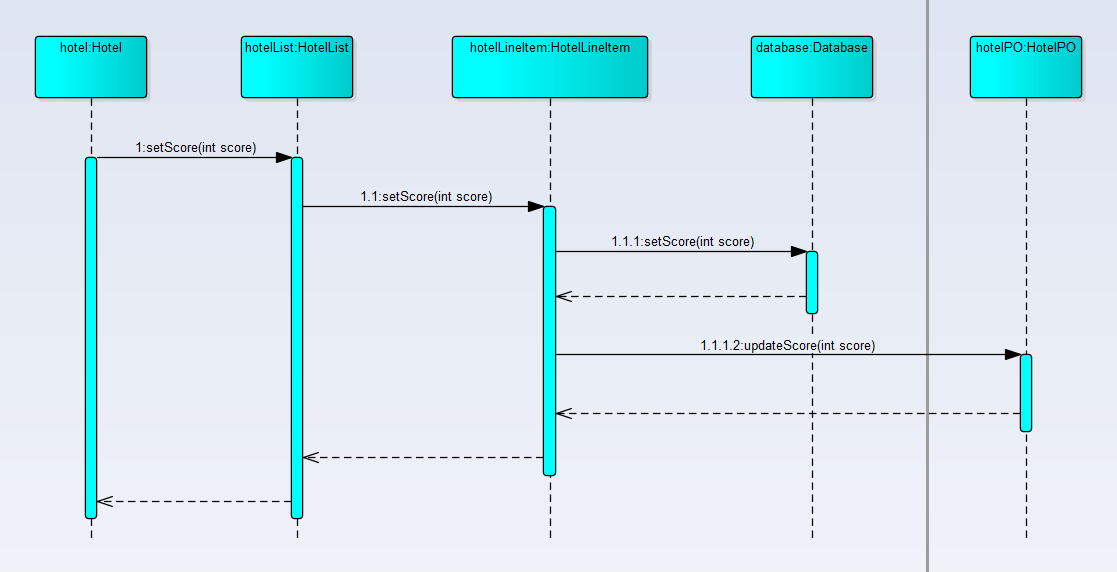


图4-1-1-12增加评分的顺序图

如图 4-1-1-7 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。如果 getEverOrder方法被 UI 调用，

Hotel 进入 EverOrder 状态，之后通过 EndGetEverOrder 进入 结束状态

，如果ManagementInformation方法或者MaintainInformation方法被调用，Hotel进入Hotel状态，之后通过EndUppdateInformation进入结束状态，如果searchInfomation方法被调用，Hotel进入SearchHotelInformation状态，之后通过EndSearch进入结束状态，如果OverviewInformation方法被调用，Hotel进入OverviewInformation状态，之后通过EndOverview进入结束状态。

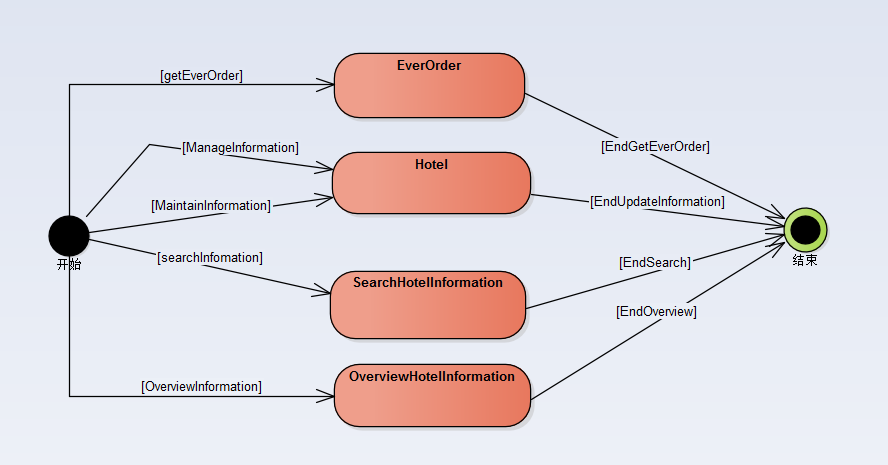


图 4-1-1-6 Hotel 对象状态图

###### 4.1.2. roombl 模块

4.1.2.1. 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系统结构设计文档。

4.1.2.2. 整体结构

根据体系结构的设计，酒店预订系统选择分层体系结构风格，将系统分为 3 层（展示层、业务逻辑层、数据层）,很好地示意了整个高层抽象。展示层包含

GUI 页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。每一层之间为了增加灵活性，添加了接口，比如在展示层和业务逻辑层之间添加 blservice.roomblservice.RoomBLService 接口，在业务逻辑层和数据层之间添加 dataservice.roomdataservice.RoomDataService 接口。RoomPO 是作为人员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 4-1-2-1 所示。

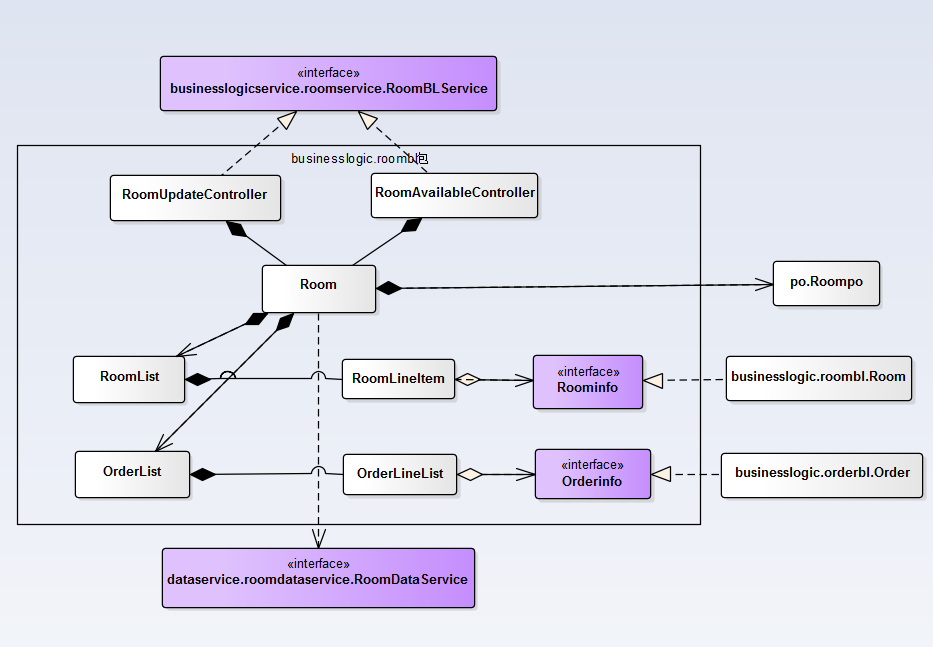


图 4-1-2-1 roombl 模块各个类的设计

roombl 模块各个类的职责如表 4-1-2-1 所示。

表 4-1-2-1 roombl 模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| Room | 房间的领域模型对象，拥有房间数据的房间号、房间状态、房间类型、房间数量、房间价格，可以帮助完成订房、退房的任务 |
| RoomAvailableController | 负责实现可用客房录入所需要的服务 |
| RoomUpdateController | 负责实现房间信息及状态更新所需要的服务 |
| RoomList | 获取所有房间列表 |
| OrderList | 获取所以订单列表 |
| RoomLineItem | 单个房间的信息 |
| OrderLineItem | 单个订单的信息 |
| Roominfo | 负责获得房间的对应信息的服务 |
| Orderinfo | 负责获得房间对应的订单信息的服务 |

4.1.2.3. 模块内部类的接口规范

RoomBL 的接口规范如表 4-1-2-2 所示。

Room的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Room.messageupdate | 语法 | public ResultMessage messageupdate( MessageInput in ) |
| 前置条件 | 酒店工作人员已登录成功 |
| 后置条件 | 系统修改房间信息 |
| Room.messageadd | 语法 | public ResultMessage messageadd(MessageInput in) |
| 前置条件 | 酒店工作人员已登录成功 |
| 后置条件 | 系统修改酒店含有的房间信息 |
| Room.getorderinfo | 语法 | public RoomVO messageadd(Order od) |
| 前置条件 | 客户已登录 |
| 后置条件 | 得到订单上的客户的个人信息和入住退房时间 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| DatabaseFactroy.getRoomDatabase | 得到Room数据库的服务的引用 |
| DatabaseFactroy.getOrderDatabase | 得到Order数据库的服务的引用 |
| RoomDataService.insert( RoomPO po ) | 插入单一持久化对象 |
| RoomDataService.update( RoomPO po ) | 更新单一持久化对象 |
| RoomDataService.delete( RoomPO po ) | 删除单一持久化对象 |
| RoomDataService.changestate( String state) | 在数据库更新房间状态 |

表 4-1-2-2 RoomBL 的接口规范

4.1.2.4. 业务逻辑层的动态模型

图 4-1-2-2 表明了酒店预订系统中，在获取订单上的客户的个人信息和入住退房时间的过程中输入一个订单VO之后，获取订单信息业务逻辑处理的相关对象之间的协作。

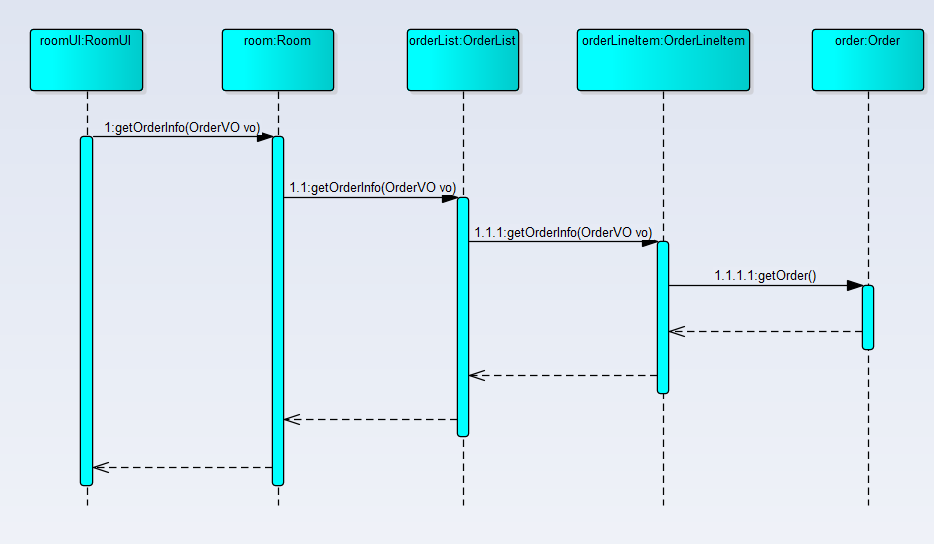


图 4-1-2-2 获取订单信息的顺序图

图 4-1-2-3 表明了酒店预订系统中，在获取某类型房间价格的过程中输入一个房间

之后，获取房间价格业务逻辑处理的相关对象之间的协作。

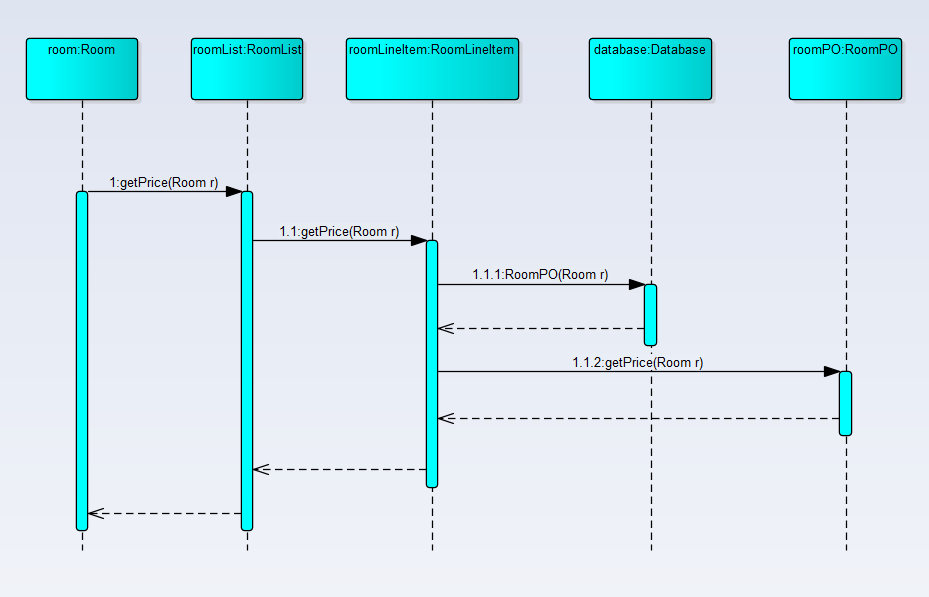
 图 4-1-2-3 获取房间价格的顺序图

图 4-1-2-4 表明了酒店预订系统中，在获取可用房间的过程中，获取可用房间业务逻辑处理的相关对象之间的协作。

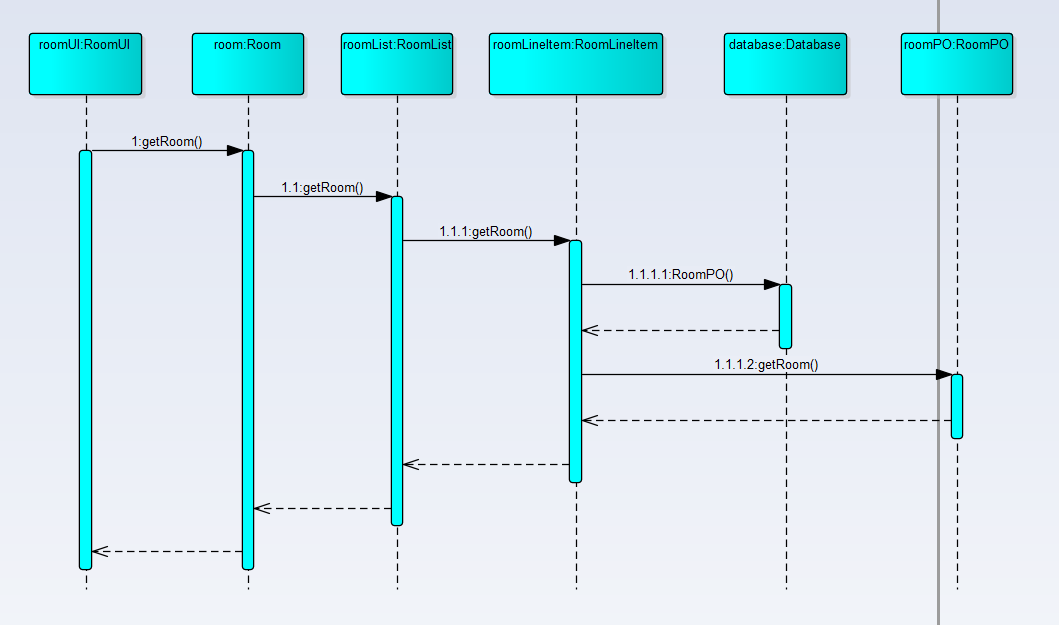
图 4-1-2-4 获取可用房间的顺序图

图 4-1-2-5 表明了酒店预订系统中，在录入可用房间的过程中输入信息

之后，录入可用房间业务逻辑处理的相关对象之间的协作。

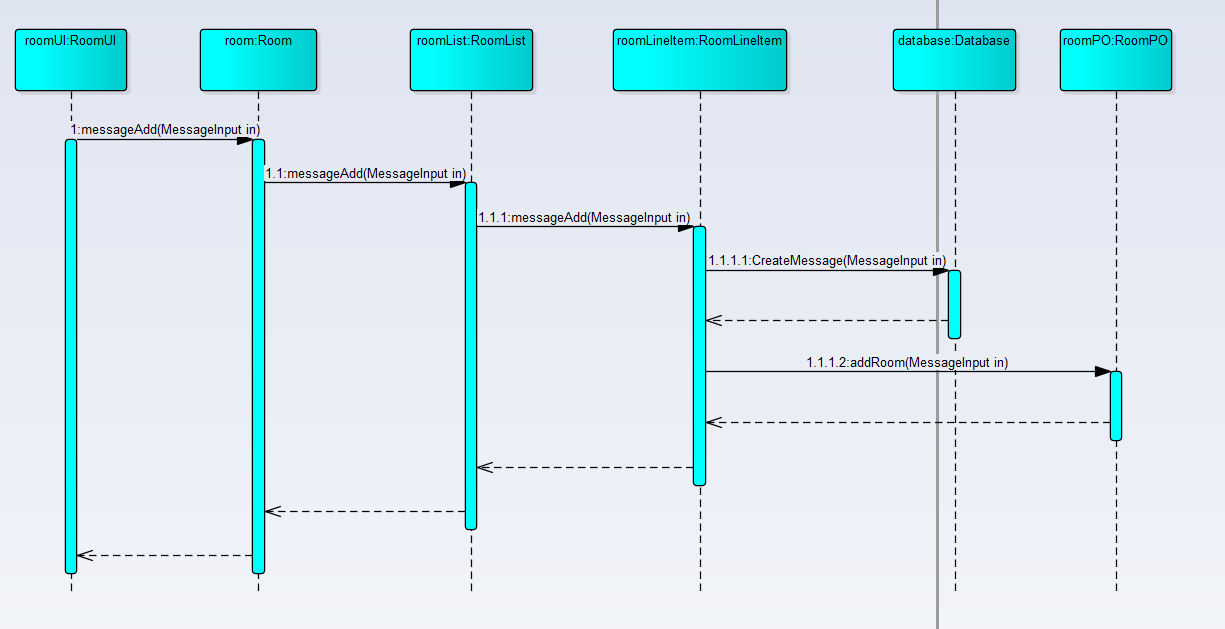


图 4-1-2-5 录入可用房间的顺序图

图 4-1-2-6 表明了酒店预订系统中，在更新房间信息的过程中输入信息

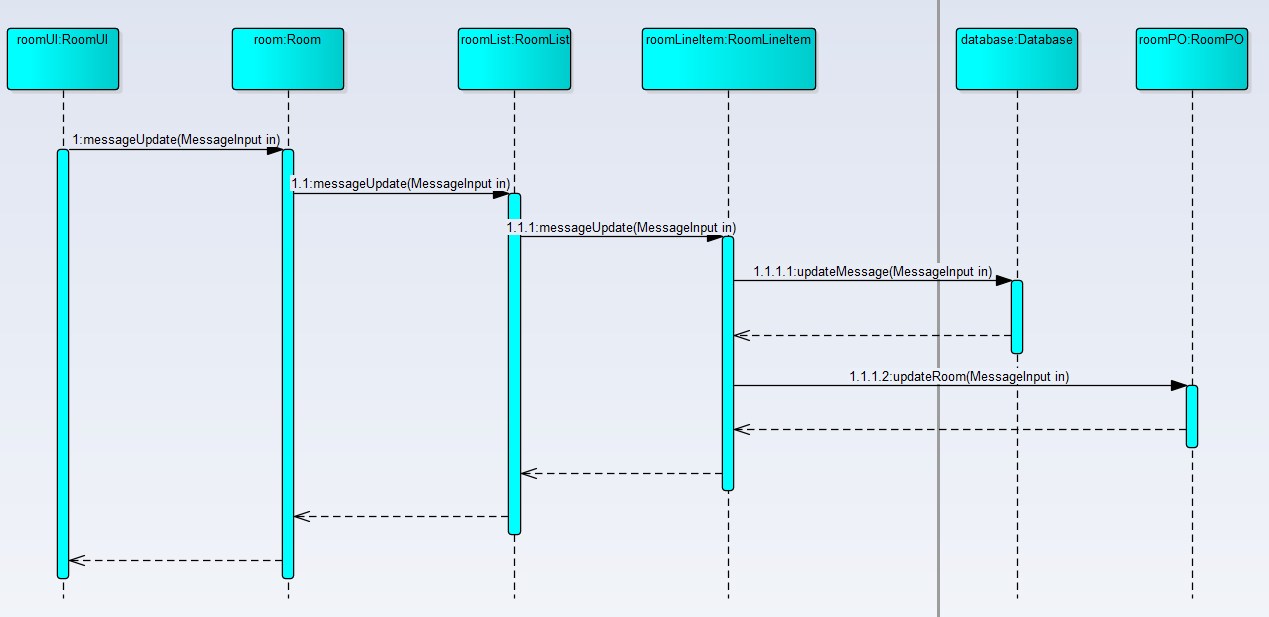
之后，更新房间信息业务逻辑处理的相关对象之间的协作。

图 4-1-2-6更新房间信息的顺序图

如图4-1-2-7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。如果 messageAdd方法被 UI 调

用，Room 进入 Room状态，之后通过 getRoom 或者通过getPrice进入 searchinfo状 态 ， 最 后 通 过endAdd进 入 结 束 状 态，如果RoomAvailable方法被调用，Room进入InputAvailableRoom状态，最后通过endRoomAvailable进入结束状态，如果messageUpdate方法被调用，Room进入updateCheckinout状态，最后通过endUpdate进入结束状态 。

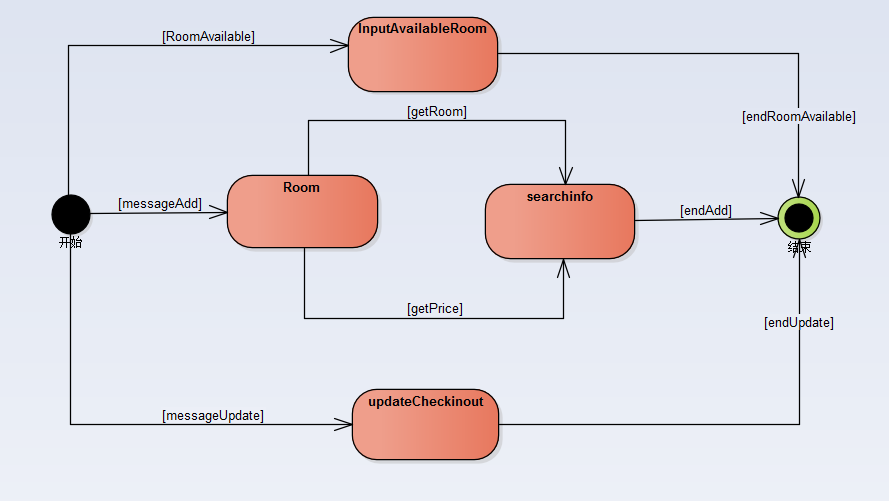


图 4-1-2-7 Room 对象状态图

###### 4.1.3. orderbl 模块

4.1.3.1. 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系统结构设计文档。

4.1.3.2. 整体结构

根据体系结构的设计，酒店预订系统选择分层体系结构风格，将系统分为 3 层（展示层、业务逻辑层、数据层）,很好地示意了整个高层抽象。展示层包含

GUI 页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。每一层之间为了增加灵活性，添加了接口，比如在展示层和业务逻辑层之间添加 blservice.orderblservice.OrderBLService 接口，在业务逻辑层和数据层之间添加 dataservice.orderdataservice.OrderDataService 接口。OrderPO 是作为持久化对象被添加到设计模型中去的。

orderbl 模块的设计如图 4-1-3-1 所示。

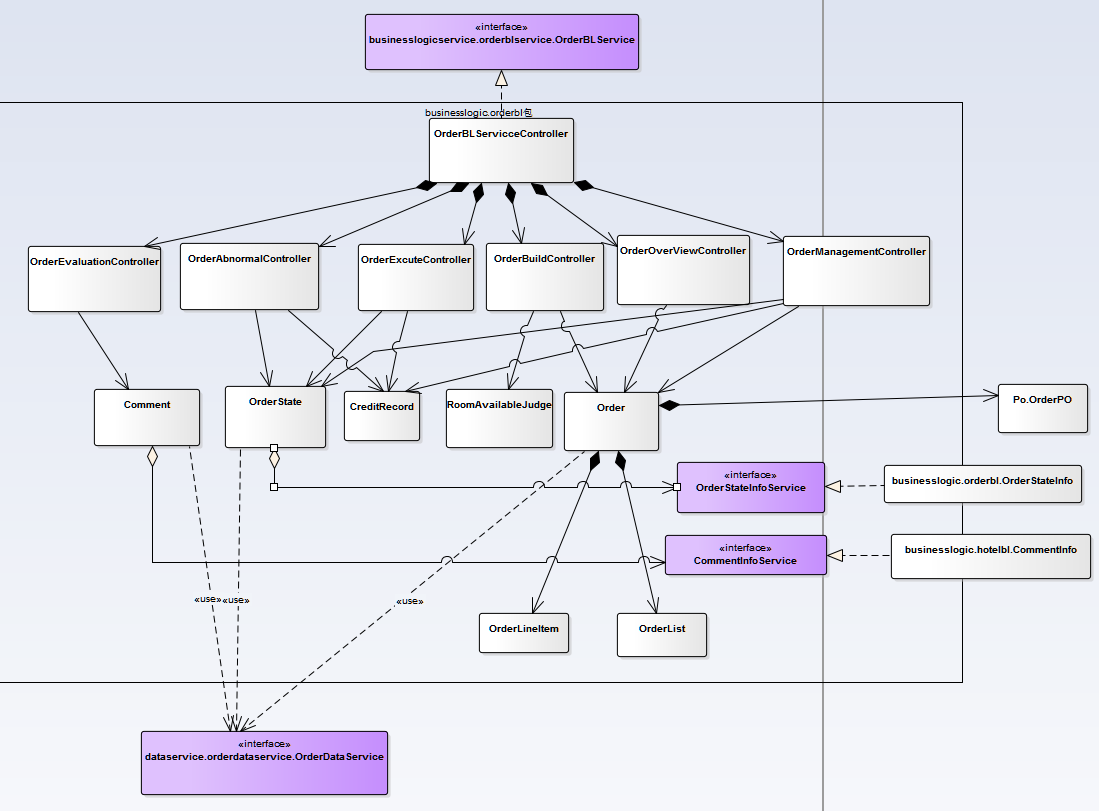


图 4-1-3-1 orderbl 模块各个类的设计

orderbl 模块各个类的职责如表 4-1-3-1 所示。

表 4-1-3-1 orderbl 模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| OrderBuildController | 负责实现对应于订单生成界面所需要的服务 |
| OrderAbnormalController | 负责实现对应于异常订单界面所需要的服务 |
| OrderExecuteController | 负责实现对应于订单执行界面所需要的服务 |
| OrderEvaluationController | 负责实现对应于订单评价界面所需要的服务 |
| OrderManagementController | 负责实现对应于订单管理所需要的服务 |
| OrderOverViewController | 负责实现对应于订单浏览所需要的服务 |
| Order | 订单的领域模型对象，拥有订单号，订单状态，客户名称，客户账号 ，价值，房间类型，入住时间，退房时间，最晚订单执行时间，房间数量，预计入住人数，有无儿童，信用值恢复策略，信用值，信用记录，会员等级，评分，评价等可以帮助完成用户信息维护所需要的服务 |
| OrderState | 封装关于OrderStateInfo的数据集合的数据结构的秘密，可以帮助完成订单管理，订单执行，订单申诉所需要的服务。 |
| Comment | 封装关于CommentInfo的数据集合的数据结构的秘密，可以帮助完成订单评价所需要的服务 |
| RoomAvailableJudge | 封装关于Room的数据集合的数据结构的秘密，可以帮助完成订单生成所需要的服务 |
| CreditRecord | 封装关于Credit的数据集合的数据结构的秘密，可以帮助完成订单申诉，订单管理，订单执行所需要的服务 |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.showAllorder | 语法 | public List<OrderVO> showAllorder(String userid) |
| 前置条件 | 选择所有订单 |
| 后置条件 | 显示该客户的所有订单 |
| Order.showUndoneorder | 语法 | public List<OrderVO> showUndoneorder(String userid); |
| 前置条件 | 选择未执行订单 |
| 后置条件 | 显示该客户的所有未执行订单 |
| Order.showDoneorder | 语法 | public List<OrderVO> showDoneorder(String userid) |
| 前置条件 | 选择已执行订单 |
| 后置条件 | 显示该客户的所有已执行订单 |
| Order.showAbnormalorder | 语法 | public List<OrderVO> showAbnormalorder(String userid) |
| 前置条件 | 选择异常订单 |
| 后置条件 | 显示该客户的所有异常订单 |
| Order.showCancelorder | 语法 | public List<OrderVO> showCancelorder(String userid) |
| 前置条件 | 选择已撤销订单 |
| 后置条件 | 显示该客户的所有已撤销订单 |
| OrderState.cancel | 语法 | public boolean cancel(String userid,String orderID) |
| 前置条件 | 在未执行订单列表，点击撤销按钮 |
| 后置条件 | 撤销所选的订单更新信用值和信用记录 |
| Order.showDetail | 语法 | public OrderVO showDetail(String orderID) |
| 前置条件 | 双击订单 |
| 后置条件 | 显示订单的详细情况 |
| Comment.updatecomment | 语法 | public void updatecomment(String comment,int score,String orderID) |
| 前置条件 | 在未执行订单列表，点击撤销按钮，且该订单未被评价过 |
| 后置条件 | 更新这个订单的评价和评分，同时更新这个订单中酒店的评分 |
| Order.getAllOrders | 语法 | public List<OrderVO> getAllOrders(String hotelId) |
| 前置条件 | 选择所有订单 |
| 后置条件 | 显示该酒店的所有订单 |
| Order.getUnfinishedOrders | 语法 | public List<OrderVO> getUnfinishedOrders(String hotelId) |
| 前置条件 | 选择所有未执行订单 |
| 后置条件 | 显示该酒店的所有未执行订单 |
| Order .getFinishedOrders | 语法 | public List<OrderVO> getFinishedOrders(String hotelId) |
| 前置条件 | 选择所有已执行订单 |
| 后置条件 | 显示该酒店的所有已执行订单 |
| Order.getAbnormalOrders | 语法 | public List<OrderVO> getAbnormalOrders(String hotelId) |
| 前置条件 | 选择所有异常订单 |
| 后置条件 | 显示该酒店的所有异常订单 |
| OrderState.processUnfinishedOrder | 语法 | public boolean processUnfinishedOrder(String orderId) |
| 前置条件 | 在未执行订单列表 |
| 后置条件 | 订单变为已执行同时更新客户的信用值和信用记录 |
| OrderState.processAbnormalOrder | 语法 | public boolean processAbnormalOrder(String orderId,Calendar delayTime) |
| 前置条件 | 在异常订单列表 |
| 后置条件 | 订单变为未执行，恢复客户信用值，同时修改最晚执行时间为输入时间 |
| OrderState.cancelAbnormalOrder | 语法 | public void cancelAbnormalOrder(String orderID) |
| 前置条件 | 客户申诉成功，在异常订单界面 |
| 后置条件 | 更该订单为已撤销，为客户增加信用值和信用记录，更新会员等级 |
| Order.saveOrderInfo | 语法 | public void saveOrderInfo(OrderVO vo) |
| 前置条件 | 客户填写完毕生成订单信息 |
| 后置条件 | 生成一个新订单 |
| Order.updateCredit | 语法 | public void updateCredit(String userID,String orderID,int value,Operate operate) |
| 前置条件 | 对于订单的操作需要修改信用值 |
| 后置条件 | 更新信用值与信用记录 |
| Order.getCanceledOrders | 语法 | public List<OrderVO> getCanceledOrders(String hotelId); |
| 前置条件 | 选择所有已撤销订单 |
| 后置条件 | 显示酒店的所有已撤销订单 |
| Order.whetherMake | 语法 | public boolean whetherMake(int numsOfRoom,String RoomType,String hotelID); |
| 前置条件 | 生成订单时判断是否有房间 |
| 后置条件 | 使生成订单操 作能执行 |
| Order.getUserID | 语法 | public String getUserID(String orderNo) |
| 前置条件 | 提供订单的编号 |
| 后置条件 | 获得该订单中客户的账号 |
| OrderState.updateOrderState | 语法 | public void updateOrderState() |
| 前置条件 | 客户端被打开 |
| 后置条件 | 更新所有超时的订单状态为异常 |
| Order.getNowadaysUnfinishedOrde | 语法 | public List<OrderVO> getNowadaysUnfinishedOrder(String HotelID) |
| 前置条件 | 酒店需要查看当天的未执行订单 |
| 后置条件 | 显示所有当天的未执行订单 |

### 4.1.3.3. 模块内部类的接口规范

OrderBL 的接口规范如表 4-1-3-2 所示。

Order的接口规范

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| DatabaseFactroy.getOrderDatabase | 得到Order数据库的服务的引用 |
| DatabaseFactroy.getUserDatabase | 得到User数据库的服务的引用 |
| DatabaseFactroy.getRoomDatabase | 得到Room数据库的服务的引用 |
| OrderDataService.insert(OrderPO po) | 插入单一持久化对象 |
| OrderDataService.getUserOrderlistinHotel (String userID,String hotelID) | 获得一个用户和酒店相关列表的持久化对象 |
| OrderDataService.update(OrderPO po) | 更新单一持久化对象 |
| OrderDataService.updateOrderState() | 更新所有订单的状态 |
| OrderDataService. findByUserID(String userID,int orderstate) | 获得一个用户相关列表的持久化对象 |
| OrderDataService. findByHotelID(String hotelID,int orderstate) | 获得一个酒店相关列表的持久化对象 |

表 4-1-3-2 OrderBL 的接口规范

### 4.1.3.4. 业务逻辑层的动态模型

图 4-1-3-2表明了酒店预订系统中，按ID查看订单详情时，查看订单业务逻辑处理的相关对象之间的协作。

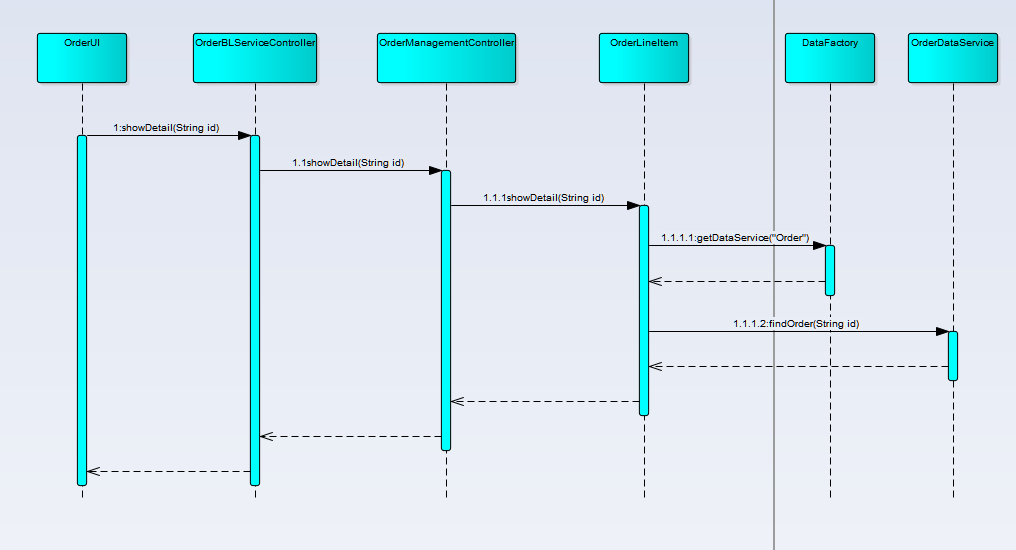
图 4-1-3-2 按ID查看订单详情的顺序图

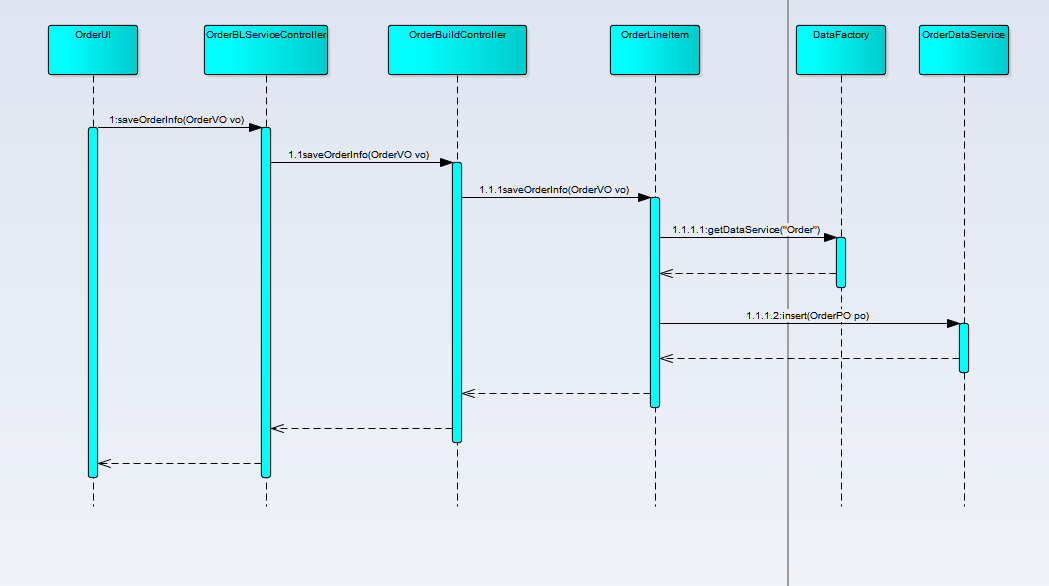
图 4-1-3-3表明了酒店预订系统中，在生成订单时传入一个订单VO后，生成新订单业务逻辑处理的相关对象之间的协作。

图4-1-3-3生成订单的顺序图

图 4-1-3-4表明了酒店预订系统中，在按订单编号获取客户编号时输入一个订单编号以后，查找订单业务逻辑处理的相关对象之间的协作。

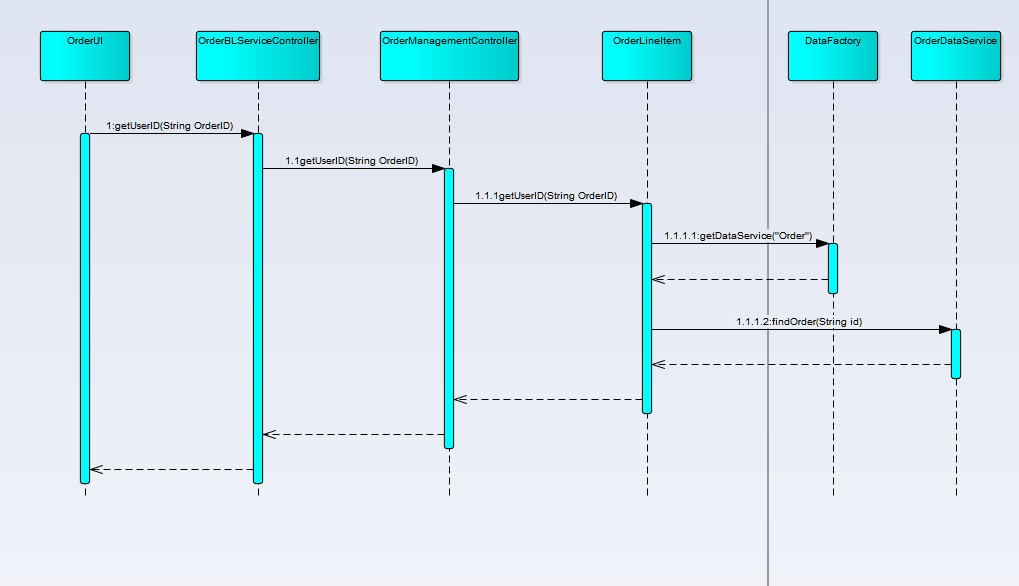


图4-1-3-4按订单编号获取客户编号的顺序图

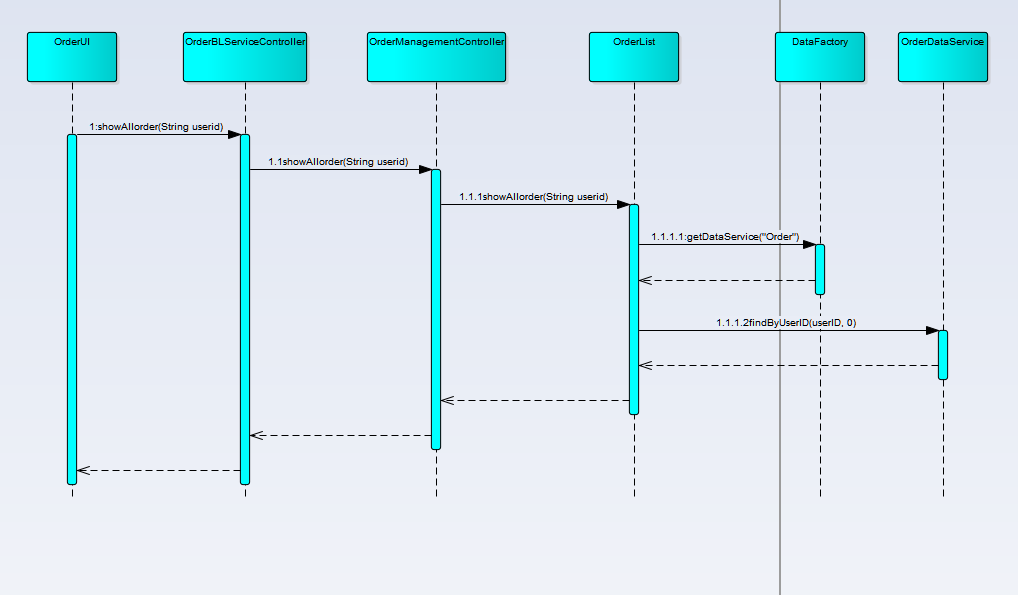
图 4-1-3-5表明了酒店预订系统中，在浏览订单时，查看客户的所有订单业务逻辑处理的相关对象之间的协作。

图4-1-3-5查看客户的所有订单的顺序图

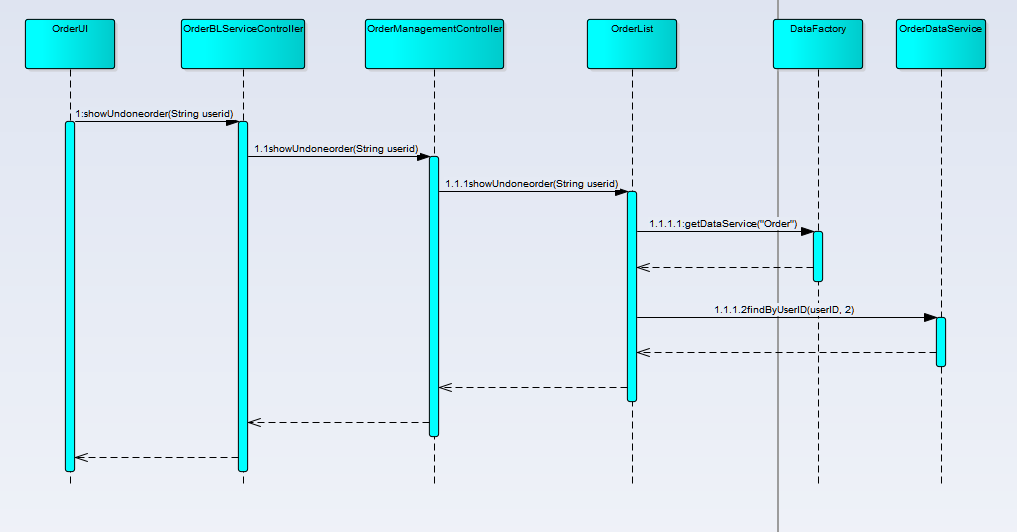
图 4-1-3-6表明了酒店预订系统中，在浏览订单时，查看客户的所有未执行订单业务逻辑处理的相关对象之间的协作。

图4-1-3-6查看客户的所有未执行订单的顺序图

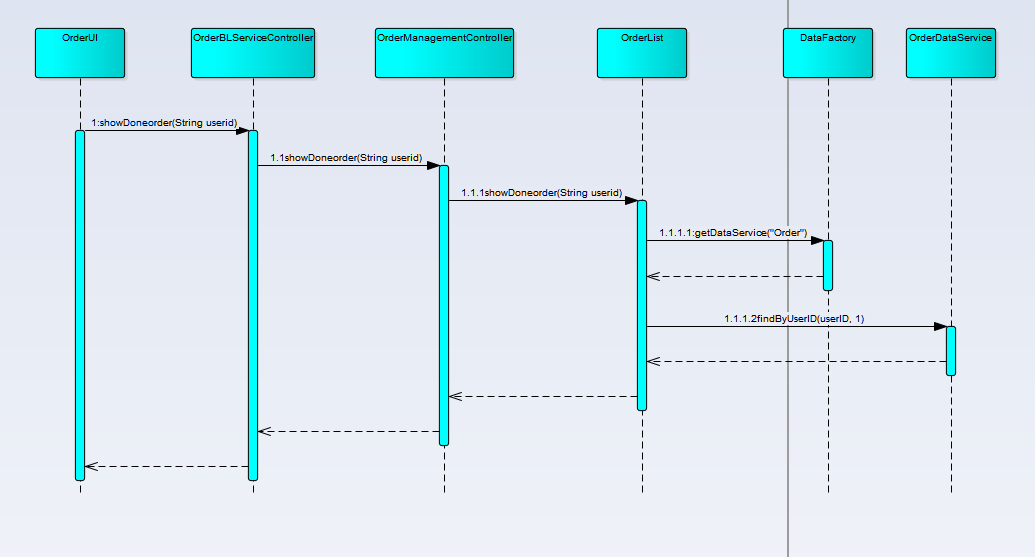
4-1-3-7表明了酒店预订系统中，在浏览订单时，查看客户的所有已执行订单业务逻辑处理的相关对象之间的协作。

图4-1-3-7查看客户的所有已执行订单的顺序图

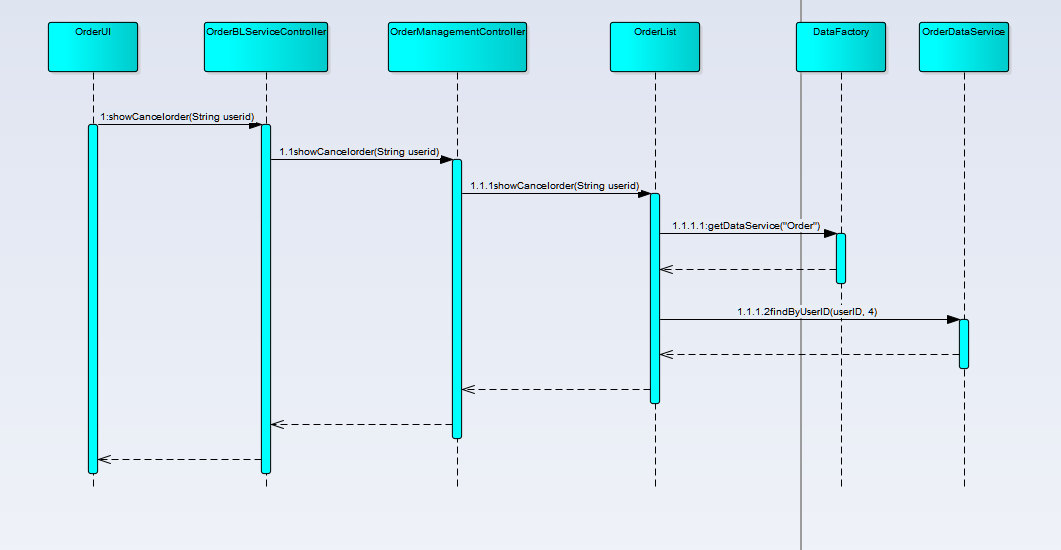
图 4-1-3-8表明了酒店预订系统中，在浏览订单时，查看客户的所有已撤销订单业务逻辑处理的相关对象之间的协作。

图4-1-3-8查看客户的所有已撤销订单的顺序图

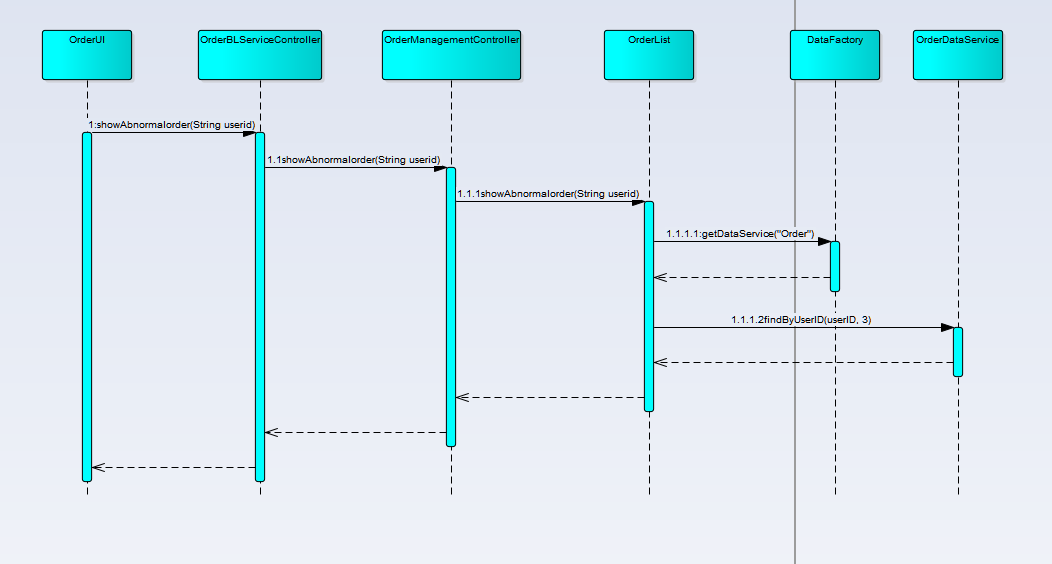
图 4-1-3-9表明了酒店预订系统中，在浏览订单时，查看客户的所有异常订单业务逻辑处理的相关对象之间的协作。

图 4-1-3-9查看客户的所有异常订单的顺序图

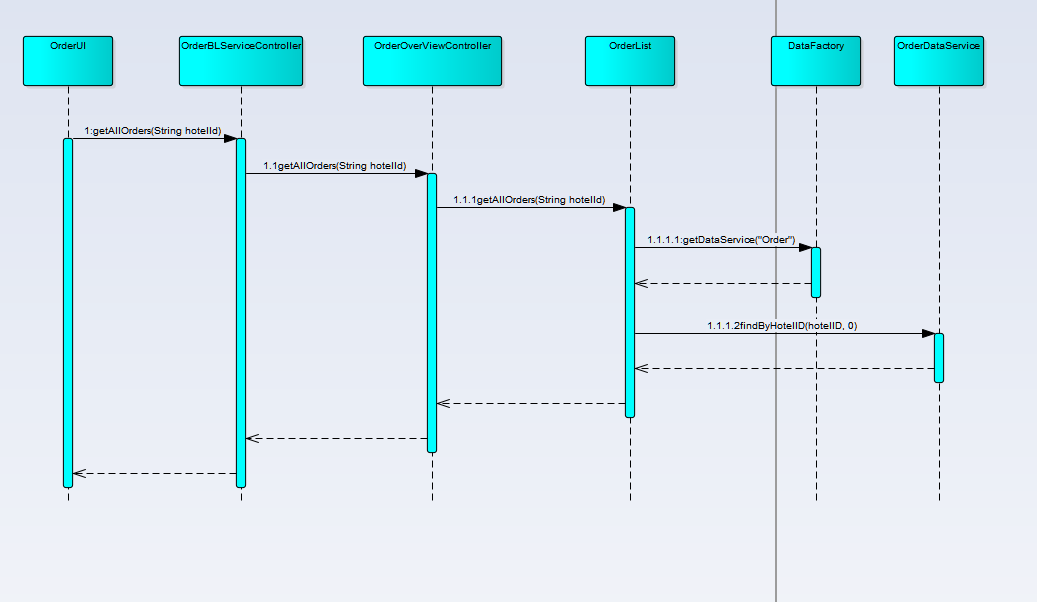
图 4-1-3-10表明了酒店预订系统中，在酒店管理人员浏览订单时，查看酒店的所有订单业务逻辑处理的相关对象之间的协作。

图 4-1-3-10查看酒店的所有订单的顺序图

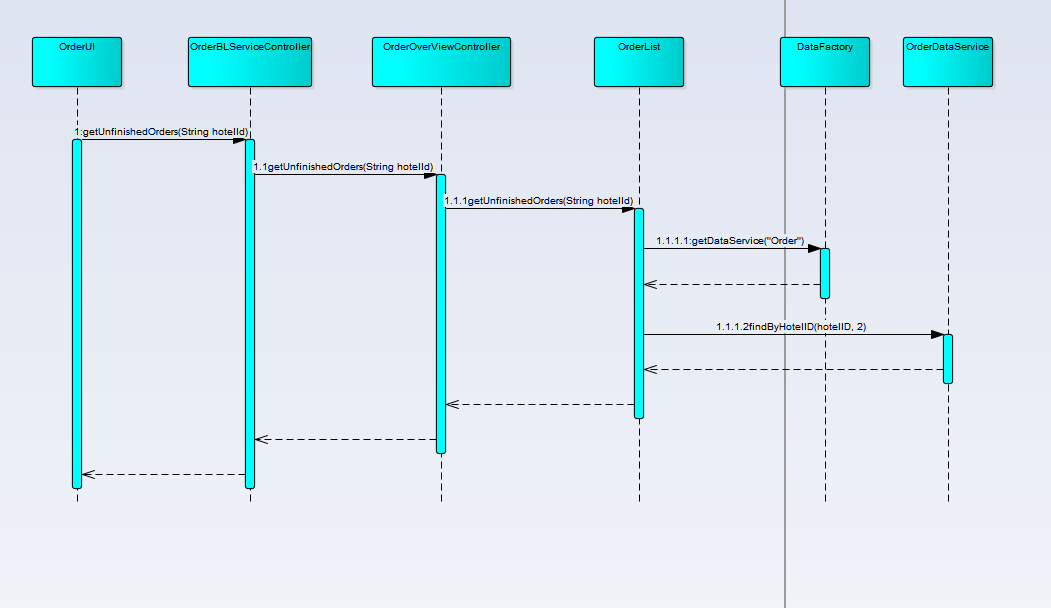
图 4-1-3-11表明了酒店预订系统中，在酒店管理人员浏览订单时，查看酒店的所有未执行订单业务逻辑处理的相关对象之间的协作。

图4-1-3-11查看酒店的所有未执行订单的顺序图

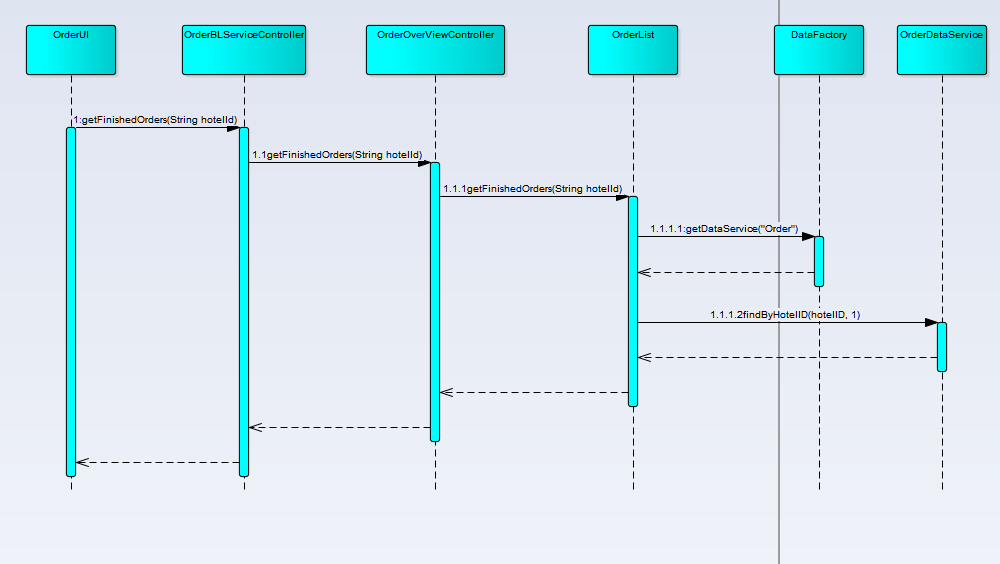
图 4-1-3-12表明了酒店预订系统中，在酒店管理人员浏览订单时，查看酒店的所有已执行订单业务逻辑处理的相关对象之间的协作。

图 4-1-3-12查看酒店的所有已执行订单的顺序图

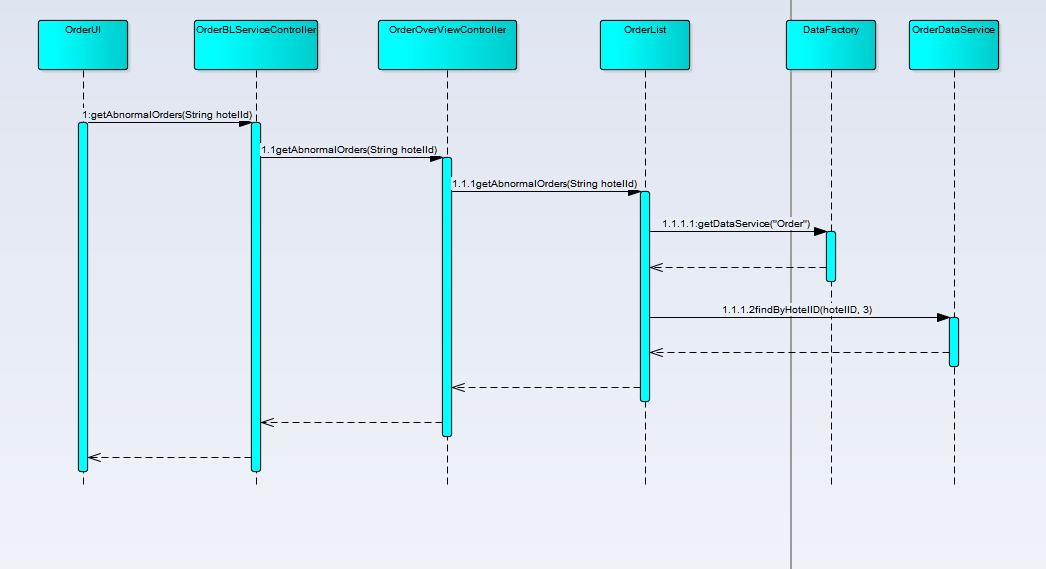
图 4-1-3-13表明了酒店预订系统中，在酒店管理人员浏览订单时，查看酒店的所有异常订单业务逻辑处理的相关对象之间的协作。

图 4-1-3-13查看酒店的所有异常订单的顺序图

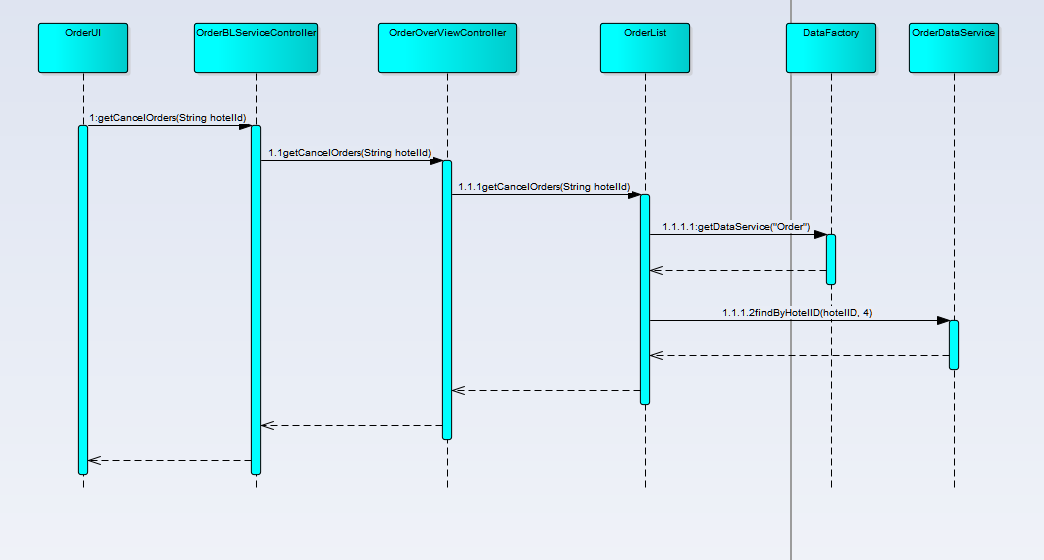
图 4-1-3-14表明了酒店预订系统中，在酒店管理人员浏览订单时，查看酒店的所有撤销订单业务逻辑处理的相关对象之间的协作。

图 4-1-3-14查看酒店的所有撤销订单的顺序图

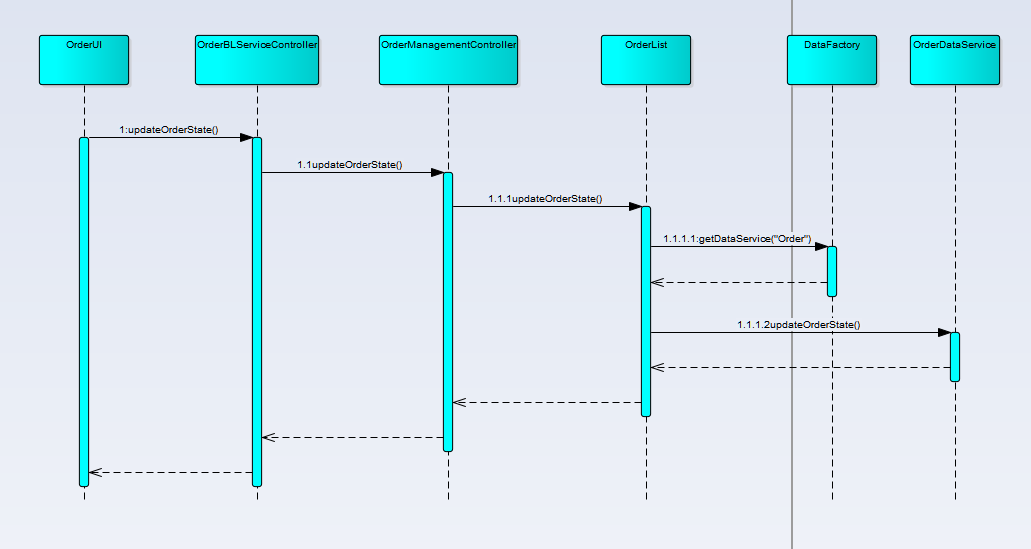
图 4-1-3-15表明了酒店预订系统中，在打开客户端时，将所有超时的订单状态设置为异常的业务逻辑处理的相关对象之间的协作。

图4-1-3-15将所有超时的订单状态设置为异常任务的顺序图

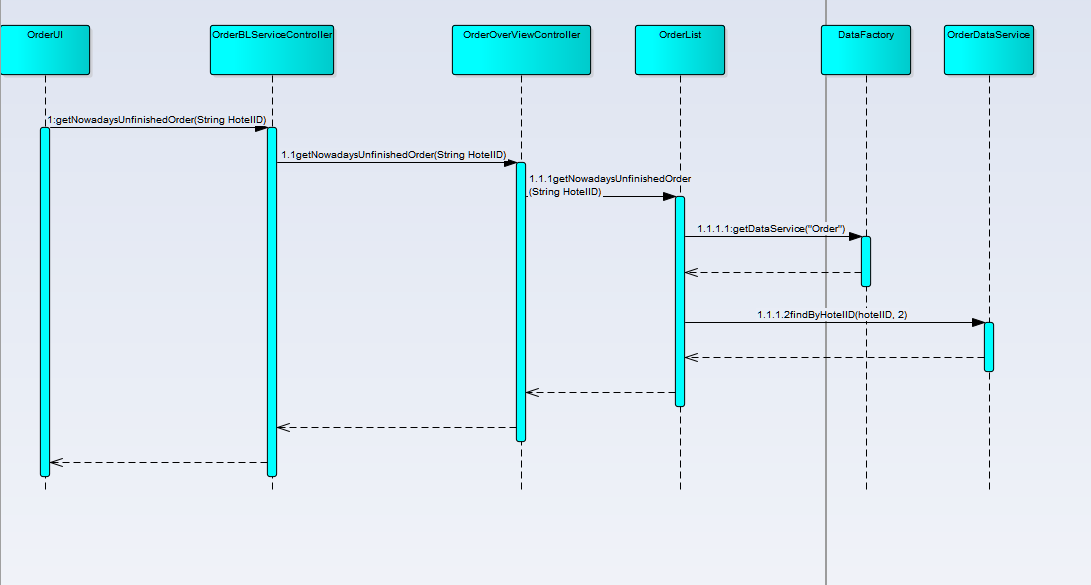
图 4-1-3-16表明了酒店预订系统中，在查看订单时，获得酒店当天所有的未执行订单的业务逻辑处理的相关对象之间的协作。

图4-1-3-1获得酒店当天所有的未执行订单的顺序图

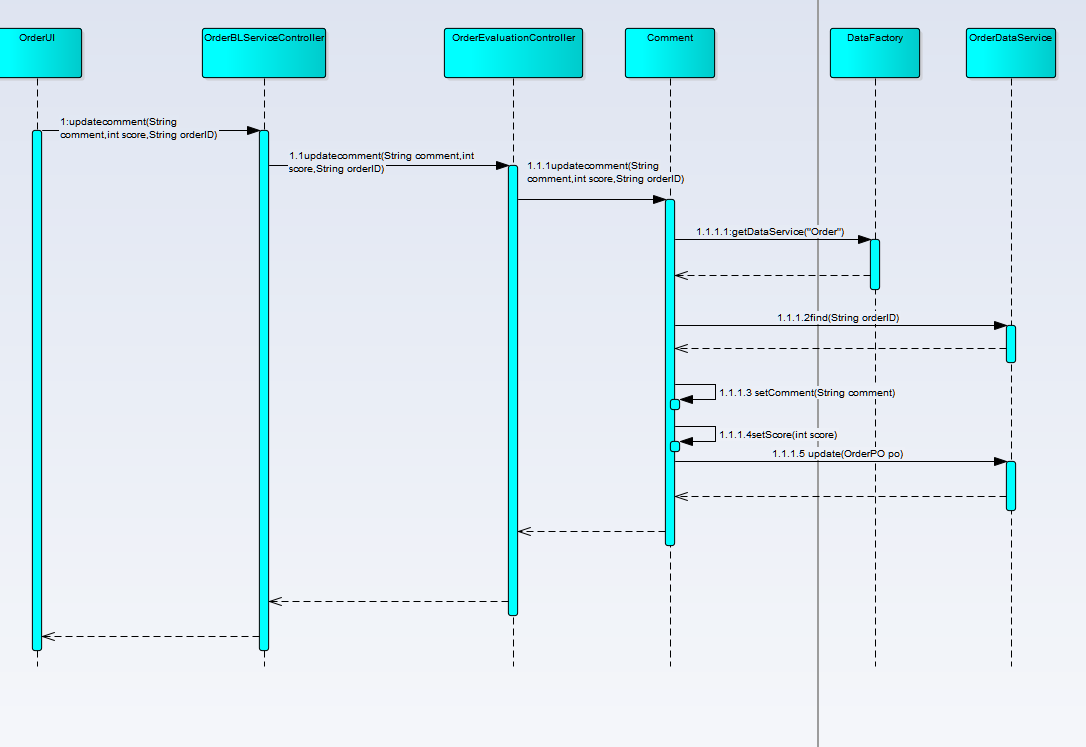
图4-1-3-17表明了酒店预订系统中，在评价订单时，更新订单评价信息业务逻辑处理的相关对象之间的协作。

图4-1-3-18评价订单的顺序图

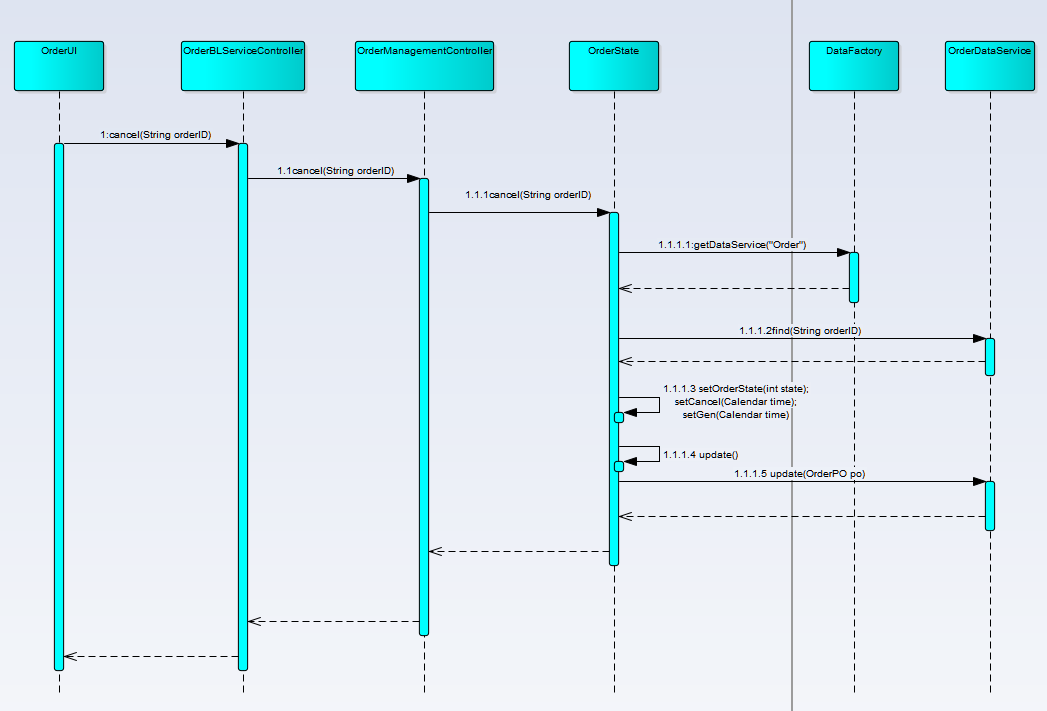
图 4-1-3-19表明了酒店预订系统中，在客户撤销订单时，改变订单状态业务逻辑处理的相关对象之间的协作。

图4-1-3-19撤销订单的顺序图

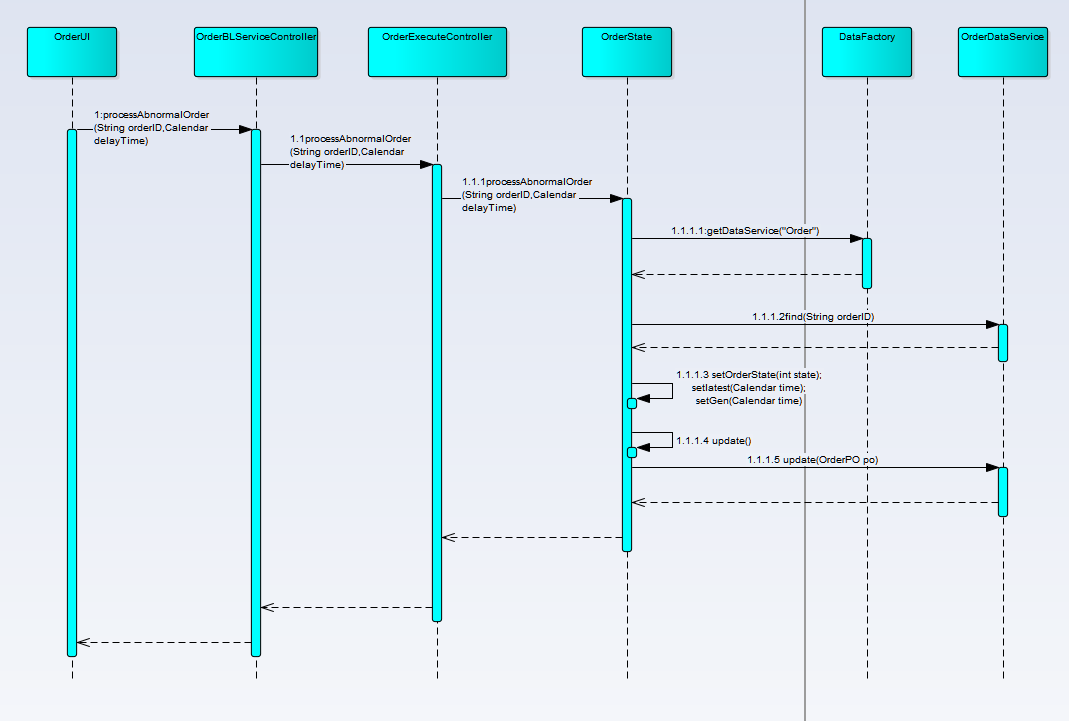
图 4-1-3-20表明了酒店预订系统中，在客户申请延迟入住时，延迟订单最晚执行时间的业务逻辑处理的相关对象之间的协作。

图4-1-3-20处理延迟订单的顺序图

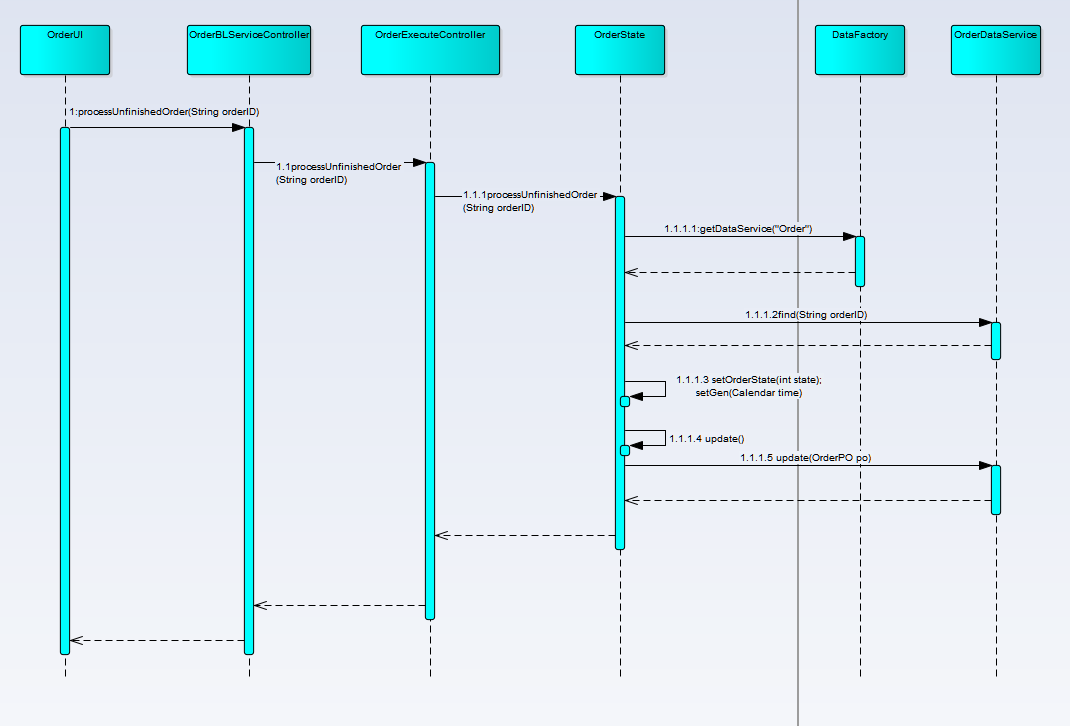
图 4-1-3-21表明了酒店预订系统中，在订单未执行时，将订单状态设置为已执行业务逻辑处理的相关对象之间的协作。

图4-1-3-21订单入住/订单执行的顺序图

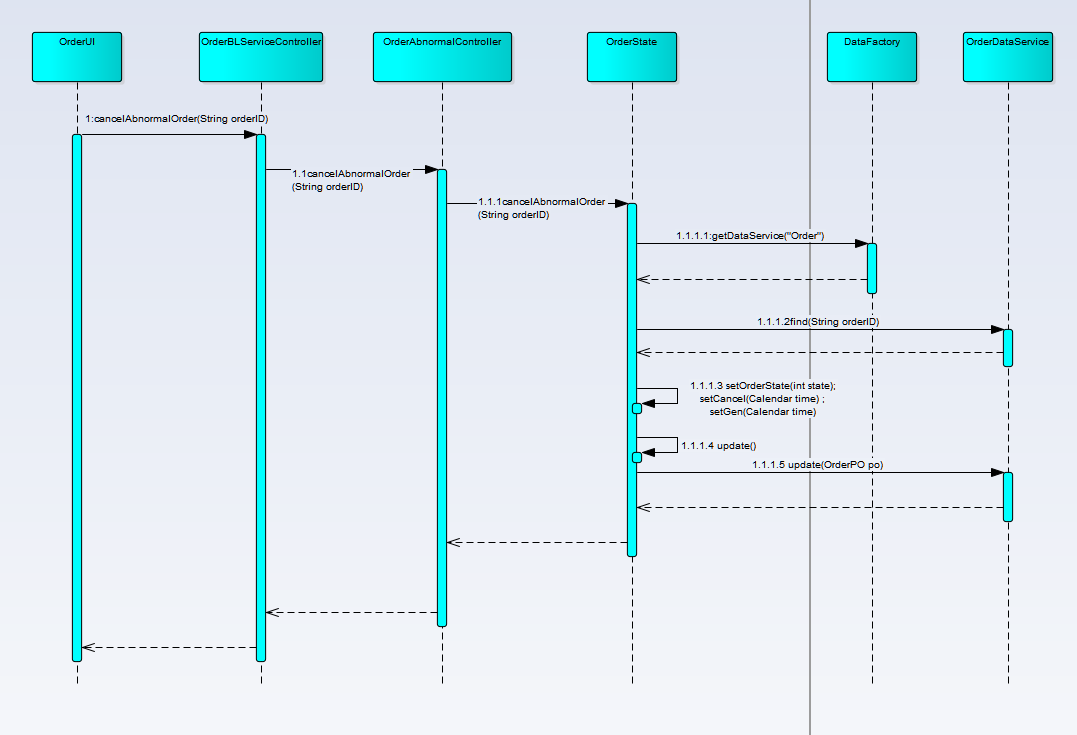
图 4-1-3-22表明了酒店预订系统中，在客户申诉异常订单时，网站营销人员处理异常订单申诉业务逻辑处理的相关对象之间的协作。

图4-1-3-22处理申诉异常订单的顺序图

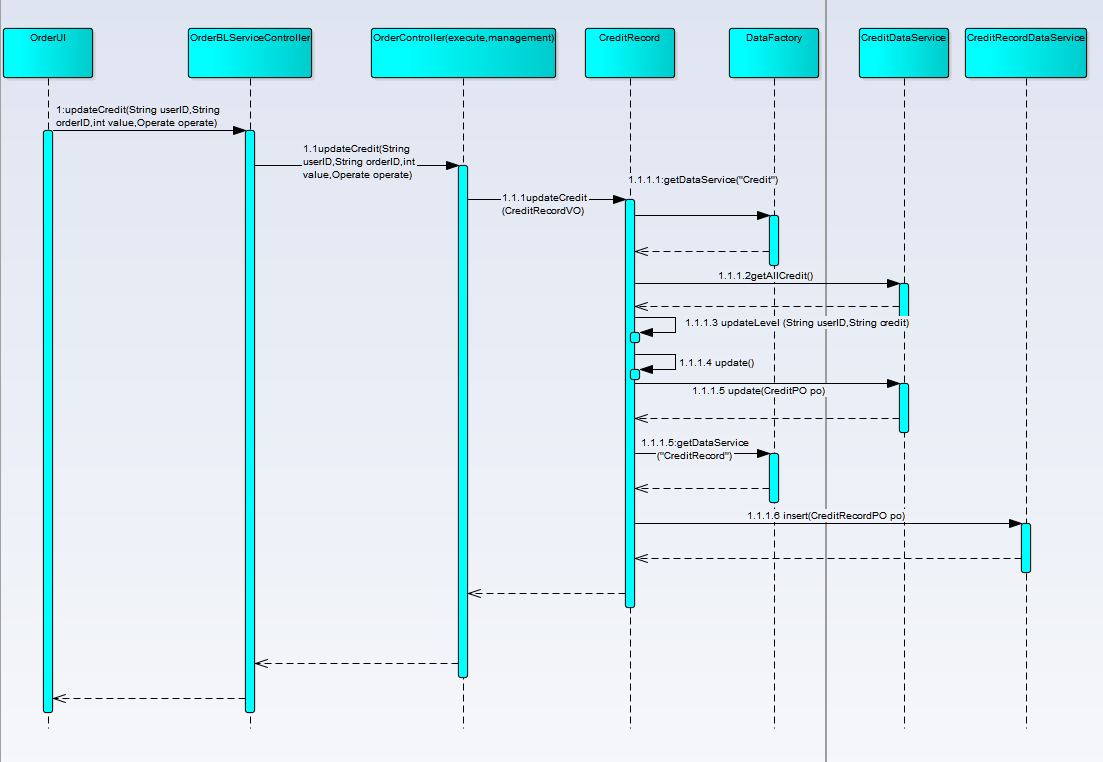
图 4-1-3-23表明了酒店预订系统中，在出现需要更改信用值与信用记录时，处理信用记录变化业务逻辑处理的相关对象之间的协作。

图4-1-3-23信用记录更新的顺序图

图 4-1-3-24表明了酒店预订系统中，在需要预定房间生成订单时，确认是否可以预定的业务逻辑处理的相关对象之间的协作。

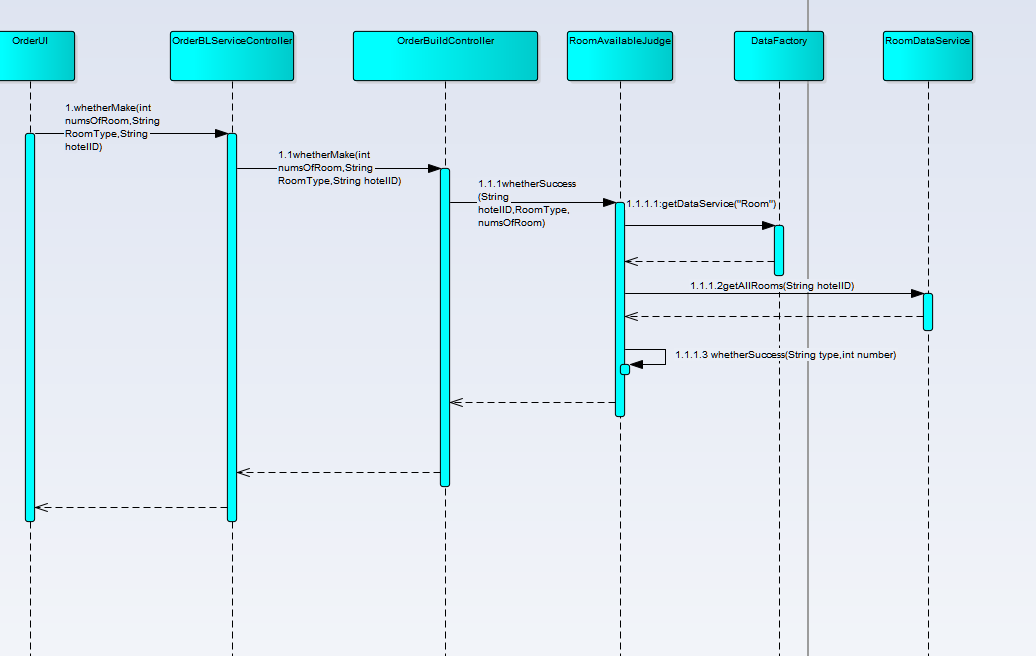
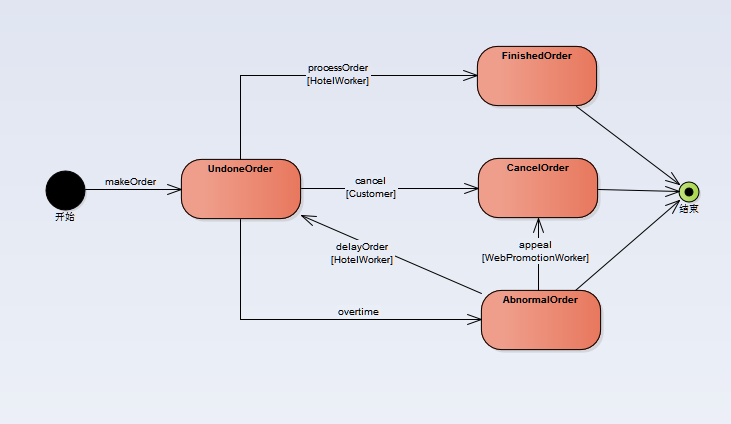


图4-1-3-23确认酒店是否可预定的顺序图

如图 4-1-3-3 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。如果addOrder方法或者browse方法被 UI 调用，进入

Order 状态，之后通过 delete 进入 cancel 状态，最后通过 end 进入结束状态，如果cancel方法被调用，进入cancel状态，之后通过abnormal方法进入abnormalOrder状态，最后通过end进入结束状态，处于Order状态时，可通过delay方法进入delayOrder状态，最后通过end进入结束状态。

图 4-1-3-3 Order 对象状态图

## 4.1.4. promotionbl 模块

### 4.1.4.1. 模块概述

promotionbl 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。

promotionbl 模块的职责及接口参见软件系统结构设计文档。

### 4.1.4.2. 整体结构

根据体系结构的设计，酒店预订系统选择分层体系结构风格，将系统分为 3 层（展示层、业务逻辑层、数据层）,很好地示意了整个高层抽象。展示层包含

GUI 页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。每一层之间为了增加灵活性，添加了接口，比如在展示层和业务逻辑层之间添加 blservice.promotionblservice.PromotionBLService 接口，在业务逻辑层和数据层之间添加 dataservice.promotiondataservice.PromotionDataService 接口。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 4-1-4-1 所示。

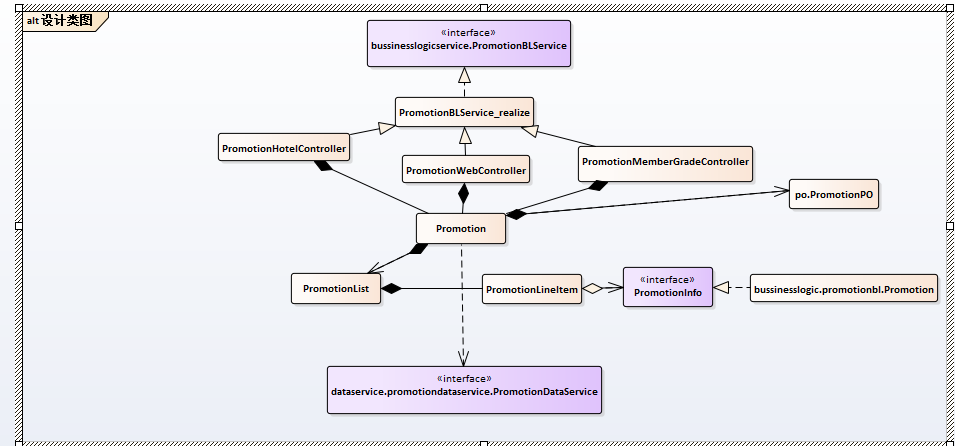


图 4-1-4-1 promotionbl 模块各个类的设计

promotionbl 模块各个类的职责如表 4-1-4-1 所示。

表 4-1-4-1 promotionbl 模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| PromotionBIService\_realize | 负责实现与营销策略有关界面所需要的服务 |
| PromotionHotelController | 负责实现对应于酒店营销策略界面所需要的服务 |
| PromotionWebController | 负责实现对应于网站营销策略界面所需要的服务 |
| PromotionMemberGradeController | 负责实现对应于会员等级折扣界面所需要的服务 |
| Promotion | 营销策略的领域模型对象，拥有策略名称，开始时间，结束时间，会员生日，订单房间数，企业会员，会员等级，商圈，折扣等可以帮助完成营销策略制定所需要的服务 |
| PromotionList | 封装关于Promotion的数据集合的数据结构的秘密，可以帮助完成营销策略界面所需要的服务 |

### 4.1.4.3. 模块内部类的接口规范

PromotionBL 的接口规范如表 4-1-4-2 所示。

Promotion的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Promotion.setmemberGrade | 语法 | public ResultMessage setmemberGrade(int creditvalue) |
| 前置条件 | 用户已登录成功 |
| 后置条件 | 设置会员等级 |
| Promotion.getPromotion | 语法 | public int getPromotion(String userID) |
| 前置条件 | 用户已登录成功 |
| 后置条件 | 查找用户可用的营销策略数 |
| Promotion.add | 语法 | public int add(String promotionnumber,String promotionid,Time begintime,Time endtime,Time birthday,int roomnumber,boolean corporatemember,String hoteldistrict,ResultMessage membergrade,double dcount) |
| 前置条件 | 用户已登录成功 |
| 后置条件 | 添加营销策略 |
| Promotion.delete | 语法 | public int delete(Promotion promotion) |
| 前置条件 | 酒店工作人员已登陆 |
| 后置条件 | 删除营销策略 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| DatabaseFactroy.getPromotionDatabase | 得到Promotion数据库的服务的引用 |
| DatabaseFactroy.getHotelDatabase | 得到Hotel数据库的服务的引用 |
| DatabaseFactroy.getOrderDatabase | 得到Order数据库的服务的引用 |
| DatabaseFactroy.getUserDatabase | 得到User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.memberlevelinsert(MessageInput in) | 在数据区添加会员等级制度 |
| PromotionDataService.memberlevelupdate(MessageInput in) | 在数据区更新会员等级制度 |

表 4-1-4-2 PromotionBL 的接口规范

### 4.1.4.4. 业务逻辑层的动态模型

图 4-1-4-2表明了酒店预订系统中，在查看营销策略时，查看营销策略业务逻辑处理的相关对象之间的协作。

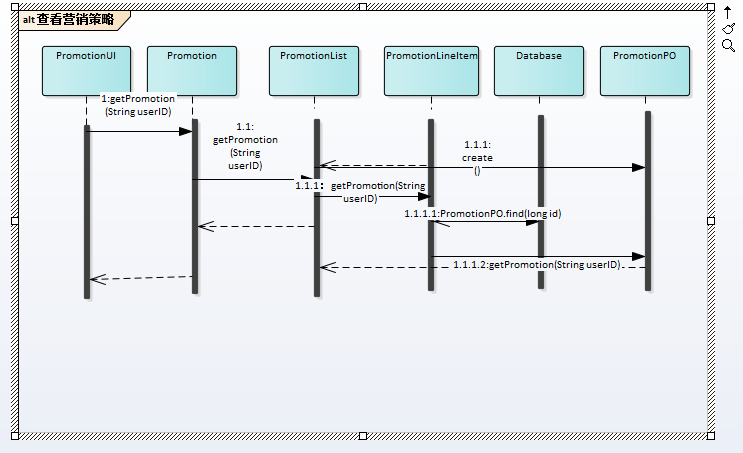


图 4-1-4-2 查看营销策略的顺序图

图 4-1-4-3表明了酒店预订系统中，在制定会员等级折扣时，制定会员等级业务逻辑处理的相关对象之间的协作。

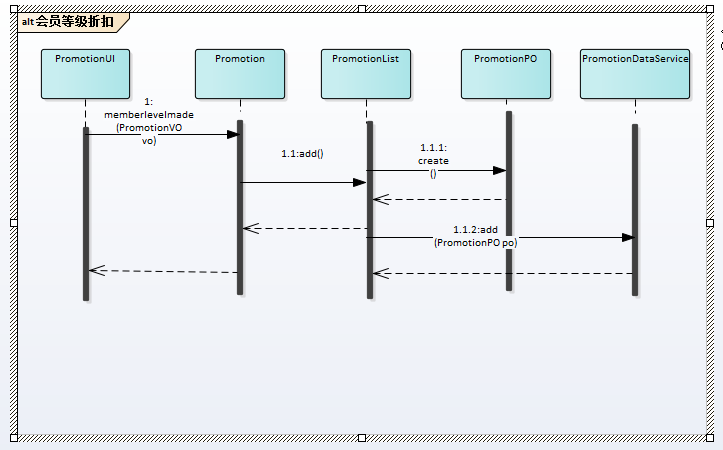


图 4-1-4-3 制定会员等级折扣的顺序图

图 4-1-4-4表明了酒店预订系统中，在制定酒店营销策略时，制定酒店营销策略业务逻辑处理的相关对象之间的协作。

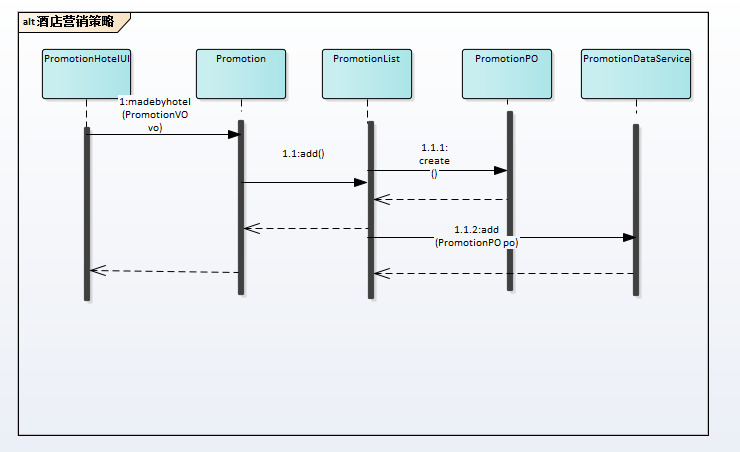


图4-1-4-4制定酒店营销策略的顺序图

图 4-1-4-5表明了酒店预订系统中，在取消制定营销策略时，取消营销策略业务逻辑处理的相关对象之间的协作。

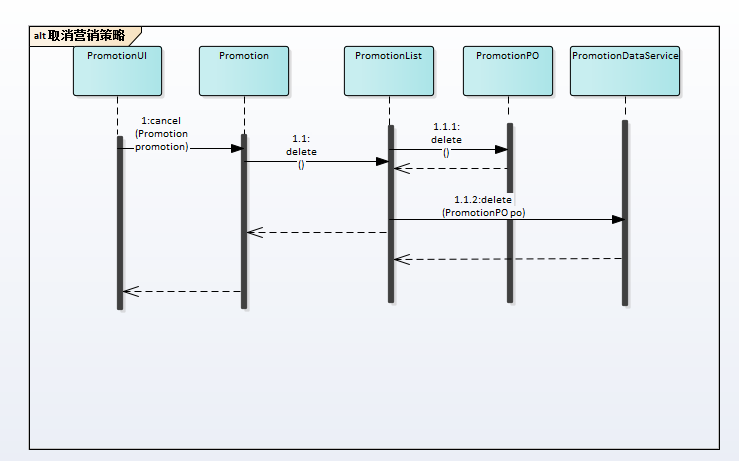


图4-1-4-5取消制定营销策略的顺序图

图 4-1-4-6表明了酒店预订系统中，在制定网站营销策略时，制定网站营销策略业务逻辑处理的相关对象之间的协作。

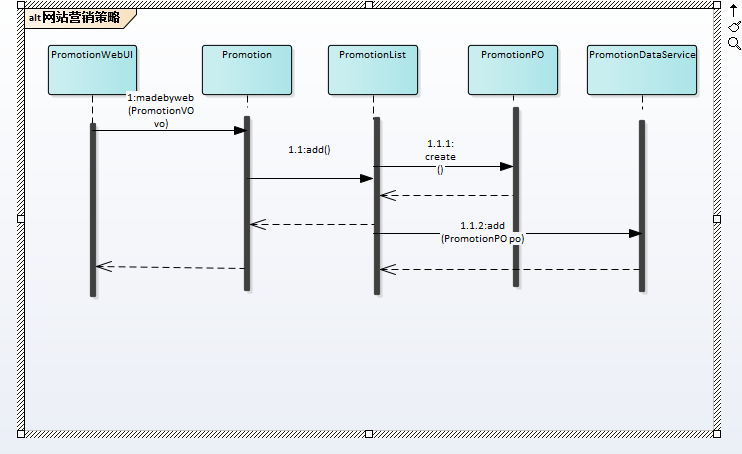


图4-1-4-6制定网站营销策略的顺序图

如图4-1-4-7所示的状态图描述了Promotion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。如果 addPromotion 方法被 UI 调用，

进入 Add 状态，之后通过 delete 进入 Delete 状态，最后通过 end进入结束状态，如果处于开始状态时delete方法被调用，Promotion进入Delete 状态，最后通过 end进入结束状态。

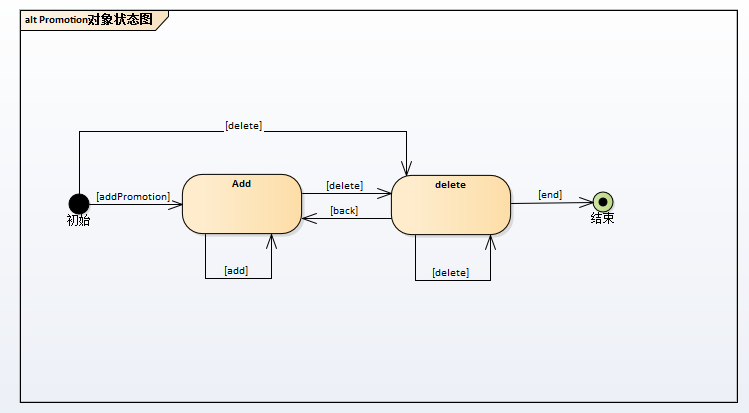


图 4-1-4-7 Promotion 对象状态图

## 4.1.5. userbl 模块

### 4.1.5.1. 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及相关非功能需求。

userbl 模块的职责及接口参见软件系统结构设计文档。

### 4.1.5.2. 整体结构

根据体系结构的设计，酒店预订系统选择分层体系结构风格，将系统分为 3 层（展示层、业务逻辑层、数据层）,很好地示意了整个高层抽象。展示层包含 GUI 页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。每一层之间为了增加灵活性，添加了接口，比如在展示层和业务逻辑

层之间添加 blservice.userblservice.UserBLService 接口，在业务逻辑层和数据层之间添加 dataservice.userdataservice.UserDataService 接口。

UserPO 、 CreditRecordPO 、 OrderPO

是作为不同单据的持久化对象被添加到设计模型中去的。PromotionInfo、OrderInfo

都是根据依赖倒置原则，为了消除循环依赖而产生的接口。

userbl 模块的设计如图 4-1-5-1 所示。

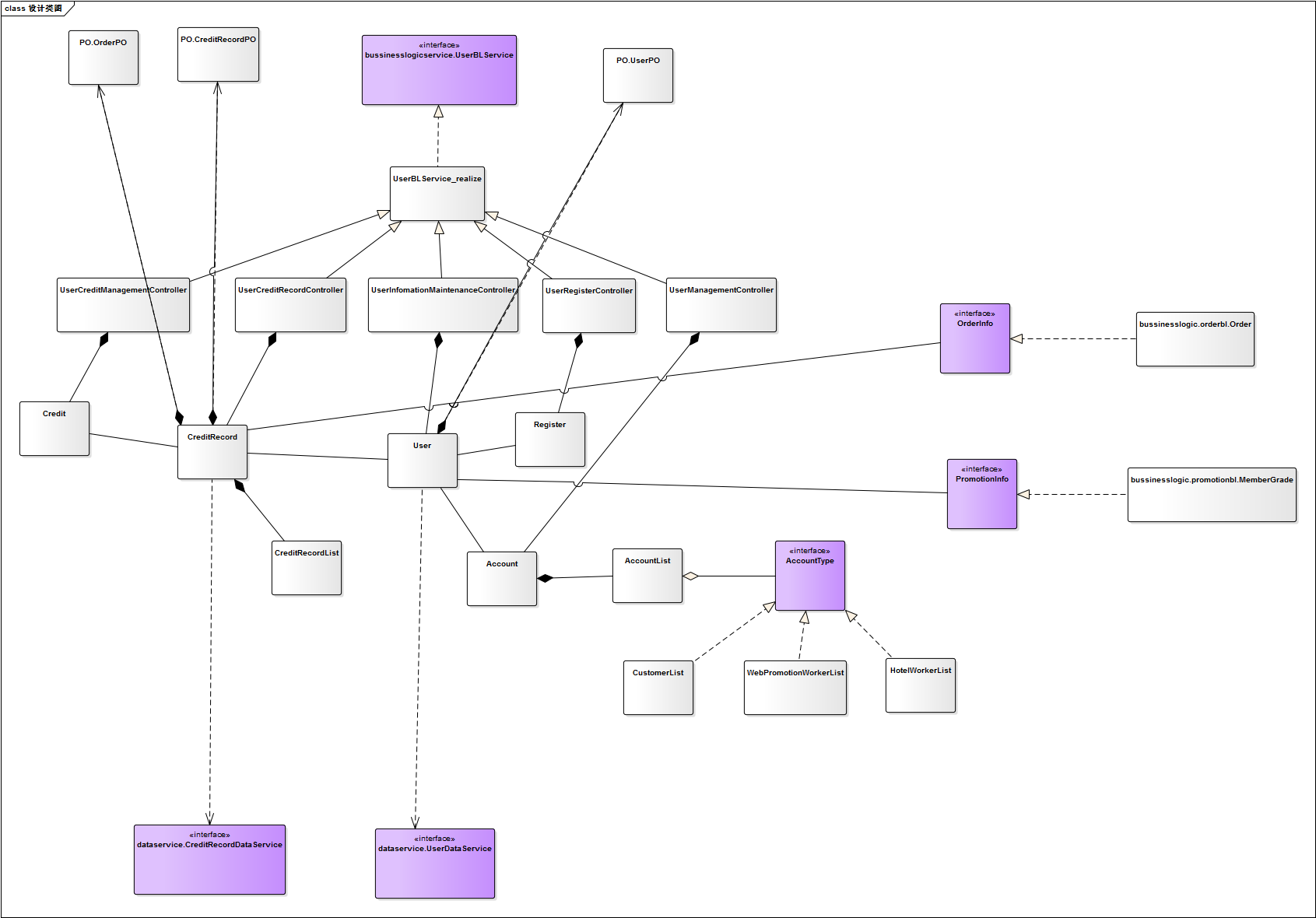


图 4-1-5-1 userbl 模块各个类的设计

userbl 模块各个类的职责如表 4-1-5-1 所示。

表 4-1-5-1 userbl 模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| UserBIService\_realize | 负责实现与用户有关界面所需要的服务 |
| UserCreditManagementController | 负责实现对应于客户信用值管理界面所需要的服务 |
| UserCreditRecordController | 负责实现对应于信用记录界面所需要的服务 |
| UserInformationMaintenanceController | 负责实现对应于用户信息维护界面所需要的服务 |
| UserRegisterController | 负责实现对应于注册界面所需要的服务 |
| UserManagementController | 负责实现对应于账户管理所需要的服务 |
| Credit | 信用管理的领域模型对象，拥有用户账号，充入金额，当前信用值等数据，可以帮助完成信用管理界面所需要的服务 |
| CreditRecord | 信用记录的领域模型对象，拥有记录时间，动作（订单执 行，订单异常，订单撤销，充值），信用度变化， 当前信用值，可以帮助完成信用记录界面所需要的服务 |
| Register | 注册的领域模型对象，拥有注册用户的用户名、账户、密码，可以解决注册问题 |
| User | 系统用户的领域模型对象，拥有客户名称，客户账号，客户联系方式， 会员等级，会员种类，生日，企业名称，可以帮助完成用户信息维护所需要的服务 |
| CreditRecordList | 封装关于CreditRecord的数据集合的数据结构的秘密，可以帮助完成信用记录界面所需要的服务 |
| Account | 账户的领域模型对象 |
| AccountList | 封装关于Account的数据集合的数据结构的秘密，可以帮助完成账户管理所需要的服务 |
| CustomerList | 封装关于CustomerAccount的数据集合的数据结构的秘密，可以帮助完成账户管理所需要的服务 |
| WebPromotionWorkerList | 封装关于WebPromotionWorkerAccount的数据集合的数据结构的秘密，可以帮助完成账户管理所需要的服务 |
| HotelWorkerList | 封装关于HotelWorkerAccount的数据集合的数据结构的秘密，可以帮助完成账户管理所需要的服务 |

### 4.1.5.3. 模块内部类的接口规范

UserBL 的接口规范如表 4-1-5-2 所示。

User的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User.findByID | 语法 | | **public** UserVO findByID(String userID) |
| 前置条件 | | 无 |
| 后置条件 | | 返回属于此ID的用户信息 |
| User.updateLevel | 语法 | | **public** **void** updateLevel(UserVO vo) |
| 前置条件 | | 存在此用户 |
| 后置条件 | | 更新此用户会员等级 |
| User.logout | 语法 | | **public** **void** logout(String id) |
| 前置条件 | | 用户已登录 |
| 后置条件 | | 用户为登出状态 |
| User.login | 语法 | | **public** **void** login(String id) |
| 前置条件 | | 用户未登录 |
| 后置条件 | | 用户为登录状态 |
| User.updateUserInfo | 语法 | | **public** **void** updateUserInfo(UserVO vo) |
| 前置条件 | | 存在此用户 |
| 后置条件 | | 更新用户信息 |
| User.create | 语法 | | **public** **void** create(UserVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 创建一个新用户 |
| 需要的服务（需接口） | | | |
| RemoteHelper.*getInstance*().getDataFactoryService() | | 得到数据工厂服务 | |
| MemberGrade.getLevelSystem() | | 得到会员等级系统 | |
| DataFactoryService.getDataService(String type) | | 得到User数据服务 | |
| UserDataService.update(UserPO po) | | 更新单一持久化对象 | |
| UserDataService.insert(UserPO po) | | 插入单一持久化对象 | |

Register的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Register.add | 语法 | | **public** **void** add(UserVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 增加一个新用户 |
| Register.getUser | 语法 | | **public** UserVO getUser(String id) |
| 前置条件 | | 存在此用户 |
| 后置条件 | | 返回此用户信息 |
| 需要的服务（需接口） | | | |
| User.create(UserVO vo) | | 创建一个新用户 | |
| User.finByID(String id) | | 按ID查找用户 | |

CreditRecord的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CreditRecord.showCreditRecord | 语法 | | **public** HashMap<String,CreditRecordVO> showCreditRecord(String id) |
| 前置条件 | | 存在此用户 |
| 后置条件 | | 返回此用户所有信用记录 |
| CreditRecord.updateCreditRecord | 语法 | | **public** **void** updateCreditRecord(String id,CreditRecordVO vo) |
| 前置条件 | | 存在此用户 |
| 后置条件 | | 更新此用户信用记录 |
| CreditRecord.add | 语法 | | **public** **void** add(String userID,CreditRecordVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 增加此用户的信用记录 |
| CreditRecord.getCreditRecord | 语法 | | **public** CreditRecordVO getCreditRecord(String userID,String orderID) |
| 前置条件 | | 存在此信用记录 |
| 后置条件 | | 返回属于此用户和记录编号的信用记录 |
| 需要的服务（需接口） | | | |
| RemoteHelper.*getInstance*().getDataFactoryService() | | 得到数据工厂服务 | |
| DataFactoryService.getDataService(String type) | | 得到CreditRecord数据服务 | |
| CreditRecordDataService.update(CreditRecordPO po) | | 更新单一持久化对象 | |
| CreditRecordDataService.insert(CreditRecordPO po) | | 插入单一持久化对象 | |

Credit的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Credit.showCredit | 语法 | | **public** **long** showCredit(String id) |
| 前置条件 | | 存在此用户 |
| 后置条件 | | 返回此用户信用值 |
| Credit.updateCredit | 语法 | | **public** **void** updateCredit(String userID,CreditRecordVO vo) |
| 前置条件 | | 存在此用户 |
| 后置条件 | | 更新此用户信用值 |
| 需要的服务（需接口） | | | |
| CreditRecord. add(String userID,CreditRecordVO vo) | | 增加此用户的信用记录 | |

Account的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Account.getUser | 语法 | | **public** UserVO getUser(String account) |
| 前置条件 | | 此账号存在 |
| 后置条件 | | 返回此账号信息 |
| Account.update | 语法 | | **public** **void** update(UserVO vo) |
| 前置条件 | | 此账号存在 |
| 后置条件 | | 更新此账号信息 |
| Account.delete | 语法 | | **public** **void** delete(UserVO vo) |
| 前置条件 | | 此账号存在 |
| 后置条件 | | 删除此账号 |
| Account.add | 语法 | | **public** **void** add(UserVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 增加一个账户 |
| 需要的服务（需接口） | | | |
| AccountList.getAccountList(int type) | | 得到对应类型的账号列表 | |
| RemoteHelper.*getInstance*().getDataFactoryService() | | 得到数据工厂服务 | |
| DataFactoryService.getDataService(String type) | | 得到User数据服务 | |
| UserDataService.update(UserPO po) | | 更新单一持久化对象 | |
| 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 | |

表 4-1-5-2 Userbl的接口规范

### 4.1.5.4. 业务逻辑层的动态模型

图 4-1-5-2 表明了酒店预订系统中，在查找客户时输入一个ID后，查找客户业务逻辑处理的相关对象之间的协作。

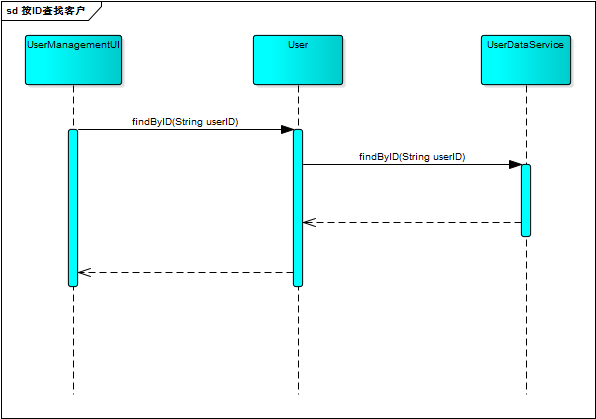


图 4-1-5-2 按ID查找客户的顺序图

图 4-1-5-3 表明了酒店预订系统中，在客户登出时，登出业务逻辑处理的相关对象之间的协作。

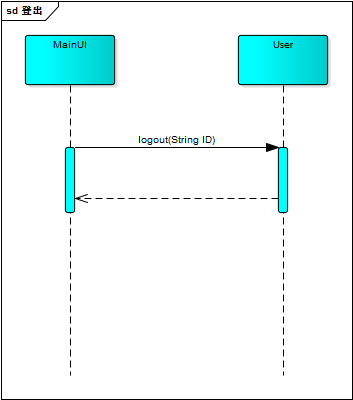


图 4-1-5-3 登出的顺序图

图 4-1-5-4 表明了酒店预订系统中，登录时，登录业务逻辑处理的

相关对象之间的协作。

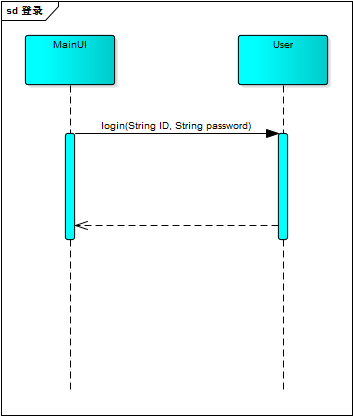


图 4-1-6-4 登录的顺序图

图 4-1-6-5 表明了酒店预订系统中，更新客户会员等级时，更新客户会员等级业务逻辑处理的相关对象之间的协作。

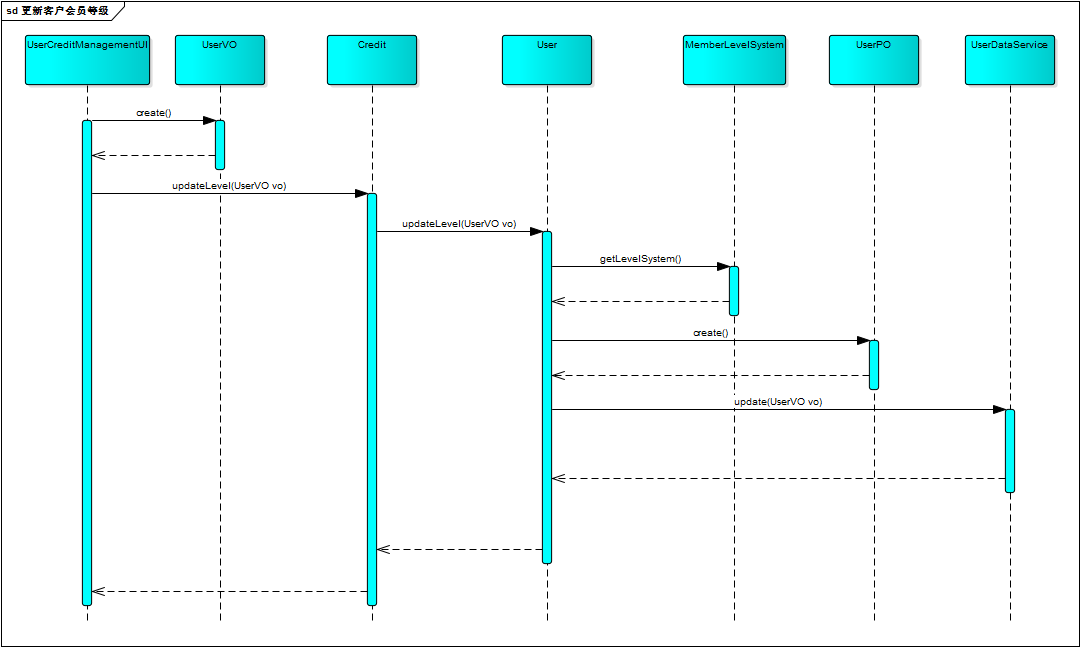
图 4-1-5-5 更新客户会员等级的顺序图

图 4-1-5-6 表明了酒店预订系统中，更新客户信息时，更新客户信息业务逻辑处理的相关对象之间的协作。

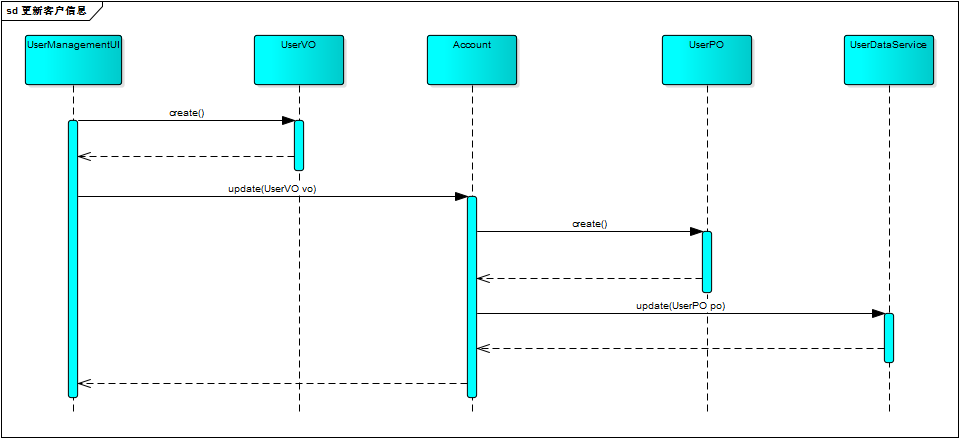


图 4-1-5-6 更新客户信息的顺序图

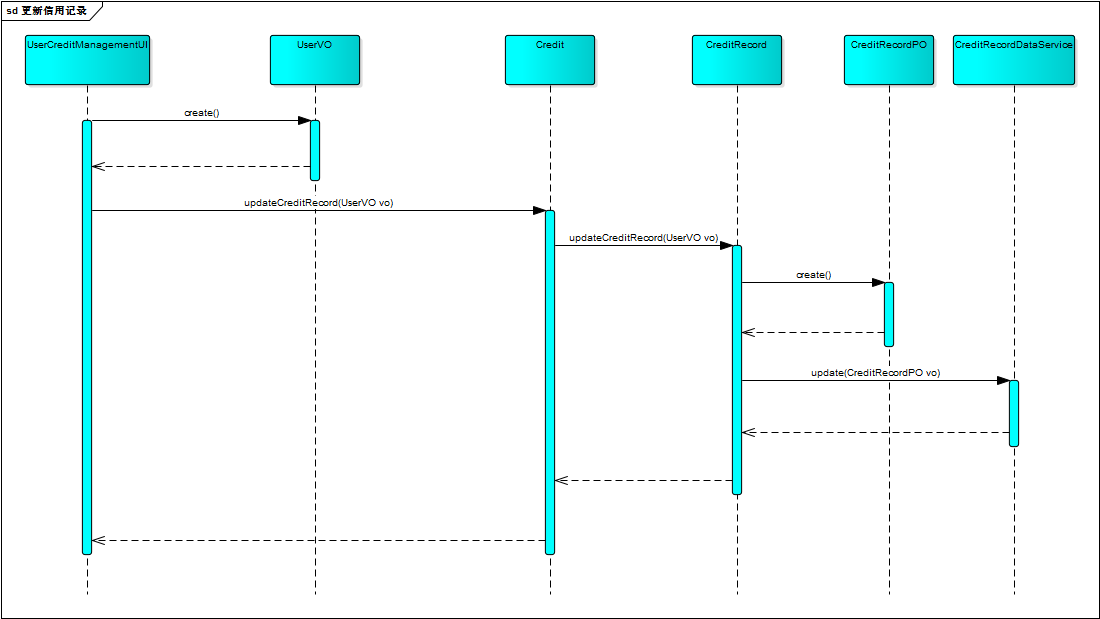
图 4-1-5-7 表明了酒店预订系统中，在更新信用记录时，更新信用业务逻辑处理的相关对象之间的协作。 

图4-1-5-7更新信用记录的顺序图

图 4-1-5- 8表明了酒店预订系统中，在更新信用值时，更新信用业务逻辑处理的相关对象之间的协作。

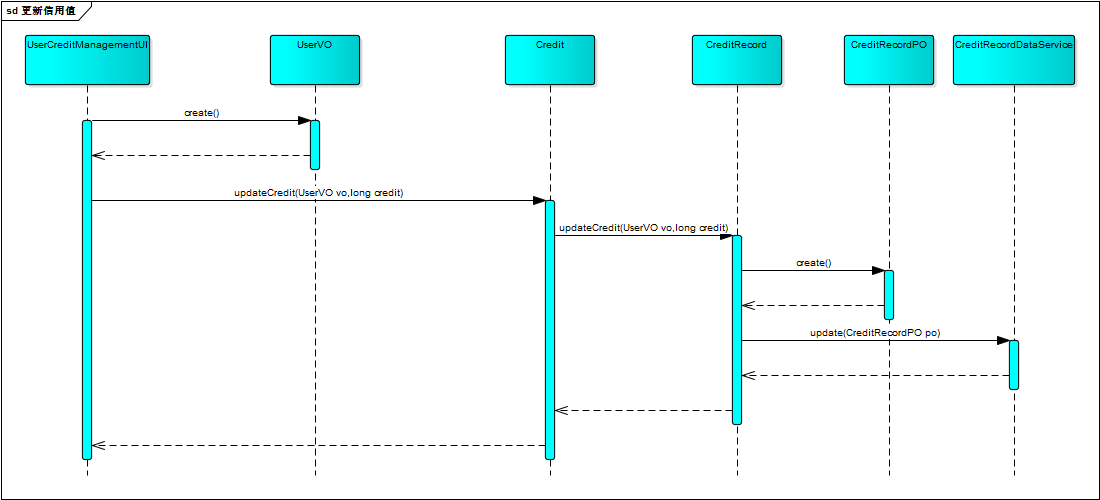


图 4-1-5- 8更新信用值的顺序图

图 4-1-5- 9表明了酒店预订系统中，在删除客户时，删除客户业务逻辑处理的相关对象之间的协作。

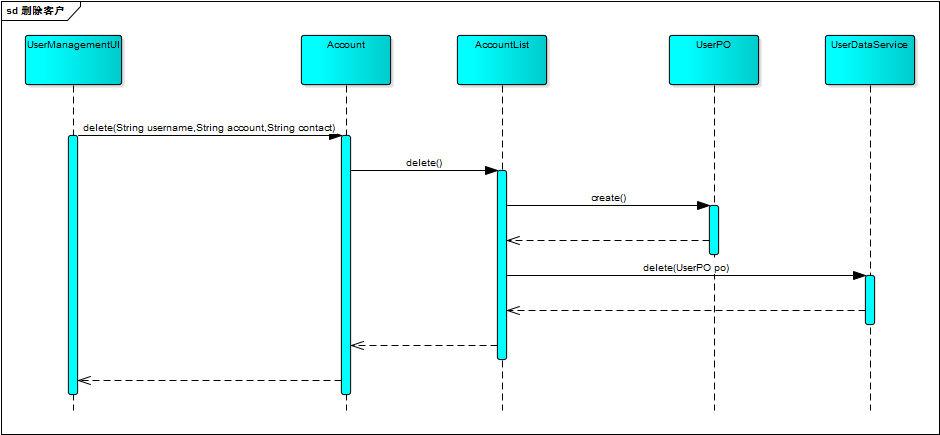


图4-1-5-9删除客户的顺序图

图 4-1-5- 10表明了酒店预订系统中，在添加客户时，添加客户业务逻辑处理的相关对象之间的协作。

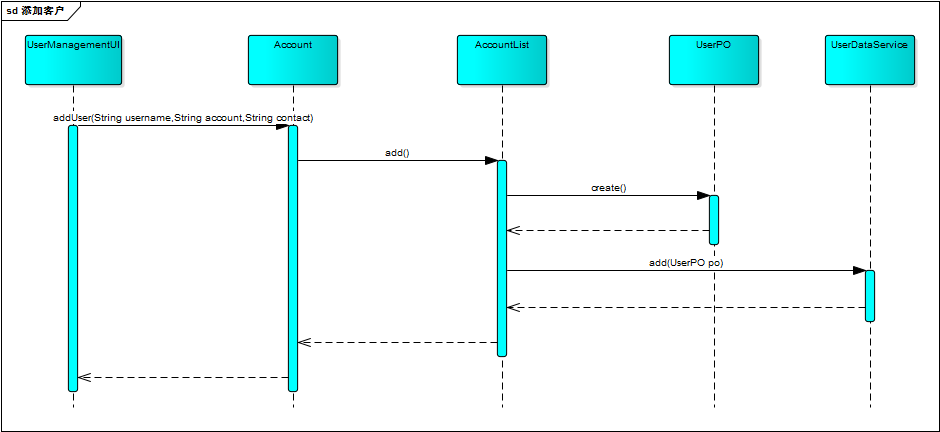


图4-1-5-10添加客户的顺序图

图 4-1-5- 11表明了酒店预订系统中，在维护客户基本信息时，维护客户信息业务逻辑处理的相关对象之间的协作。

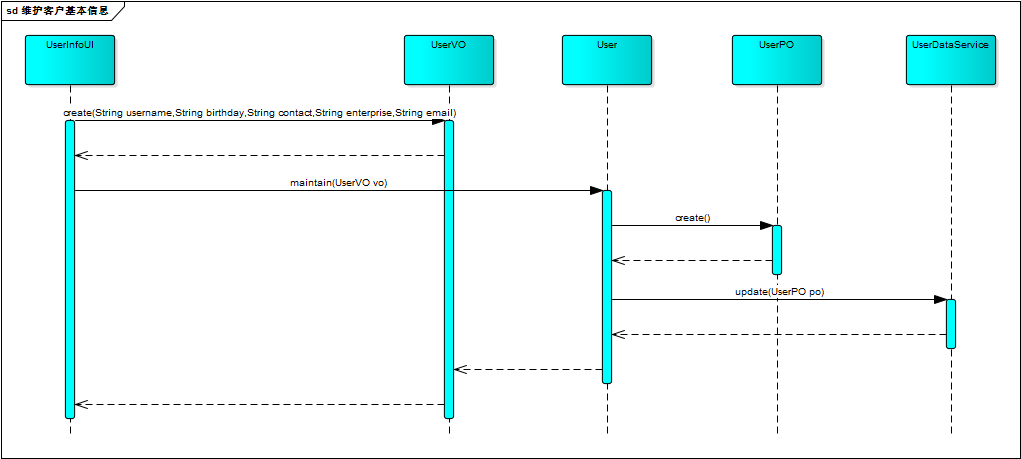


图4-1-5-11维护客户基本信息的顺序图

图 4-1-5- 12表明了酒店预订系统中，在显示信用记录时，显示信用记录业务逻辑处理的相关对象之间的协作。

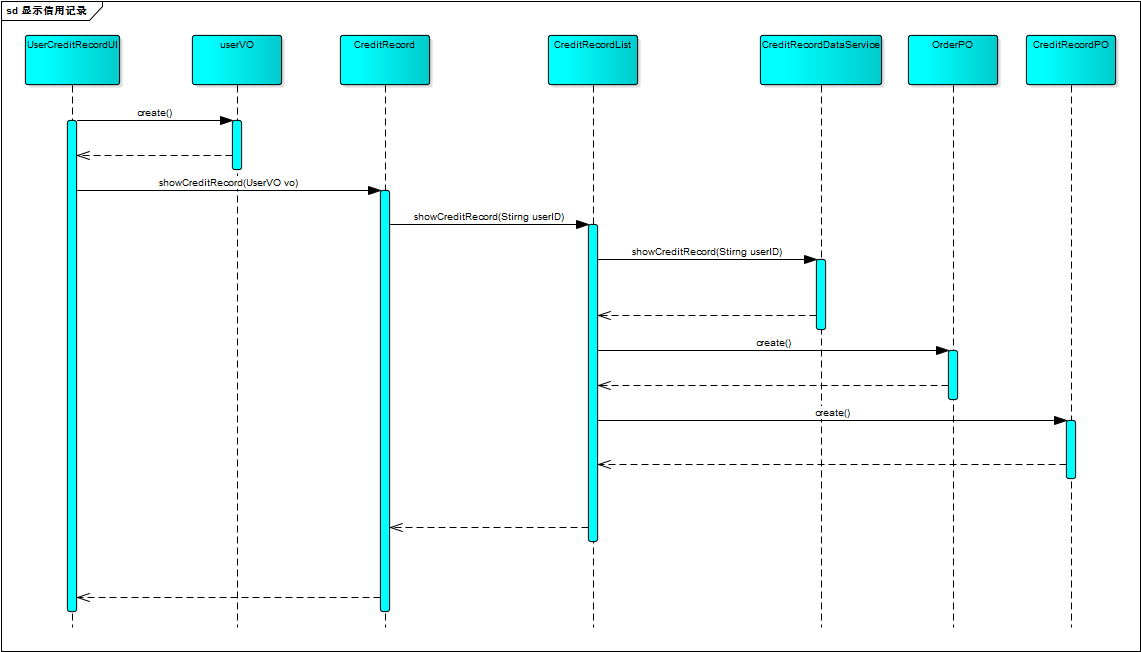


图4-1-5-12显示信用记录的顺序图

图 4-1-5- 13表明了酒店预订系统中，在显示信用值时，显示信用值业务逻辑处理的相关对象之间的协作。

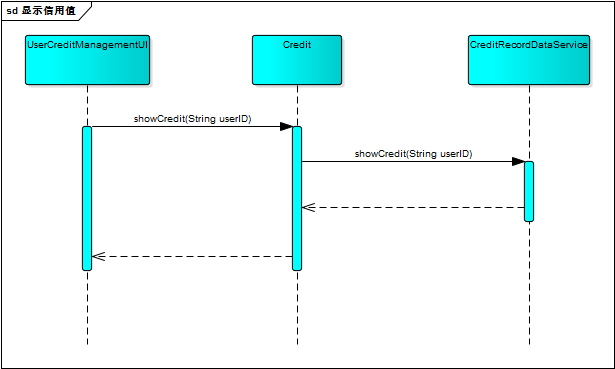


图4-1-5-13显示信用值的顺序图

图 4-1-5- 14表明了酒店预订系统中，在客户注册时，注册业务逻辑处理的相关对象之间的协作。

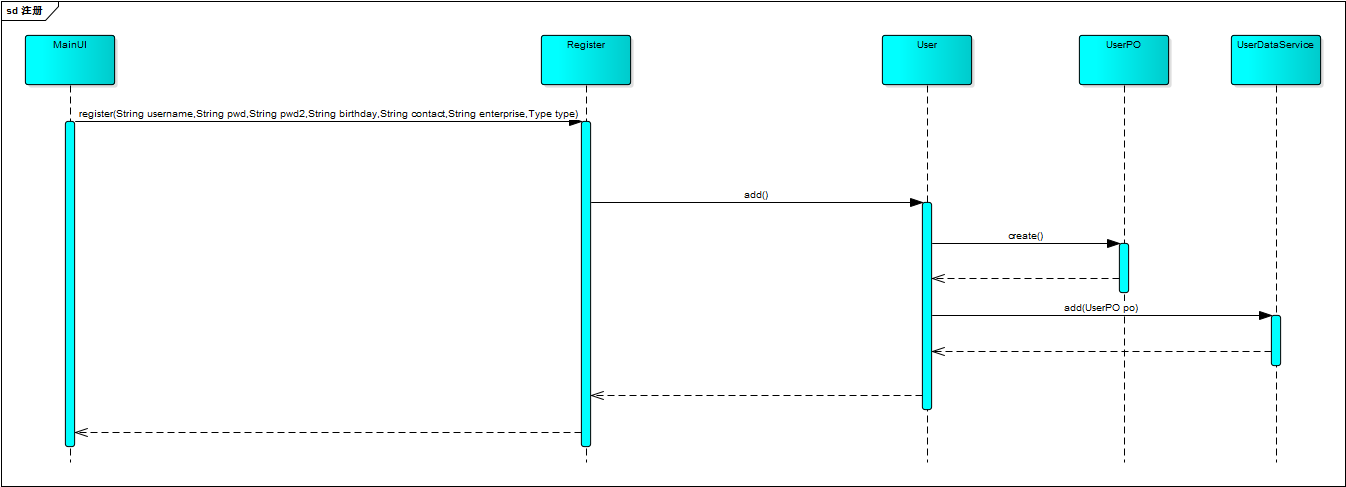
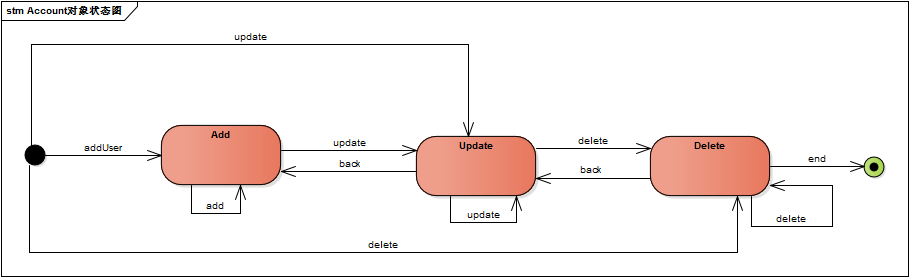


图4-1-5-14注册的顺序图

如图4-1-5-15所示的状态图描述了Account对象的生存期间执行查询单据操作的状态序列，引起转移的事件，以及因状态转移而伴随的动作。如果 addUser 方法被 UI 调用，Account 进入 Add状态，然后如果update方法被调用，则进入 Update状态，然后如果delete方法被调用，则进入 Delete 状态，最后通过end进入结束状态，处于开始状态时，如果调用update方法，则直接进入Update状态，如果调用delete，则直接进入Delete状态。

图 4-1-5-15 Account 对象执行查询操作的状态图

如图 4-1-5-16 所示的状态图描述了 CreditRecord 对象的生存期间执行增加单据操作的状态序列，引起转移的事件，以及因状态转移而伴随的动作。如果

showCreditRecord 方法被 UI 调用，CreditRecord进入 Presentation状态，然后如果调用updateCreditRecord方法，则进入 Updaet 状态， 最后通过end进入结束状态。如果处于开始状态，调用updateRecord方法，则直接进入Update状态。

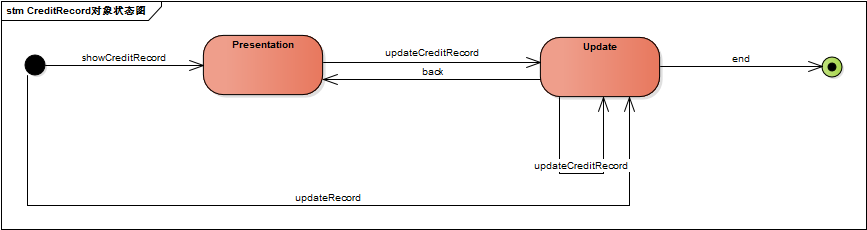
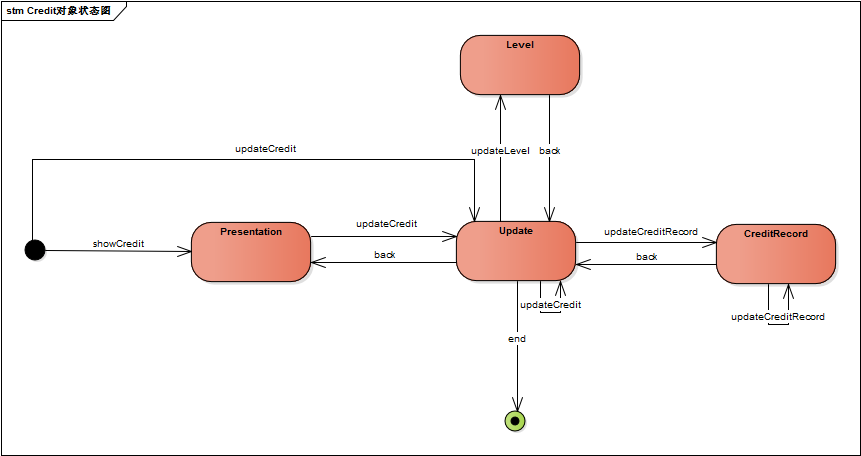
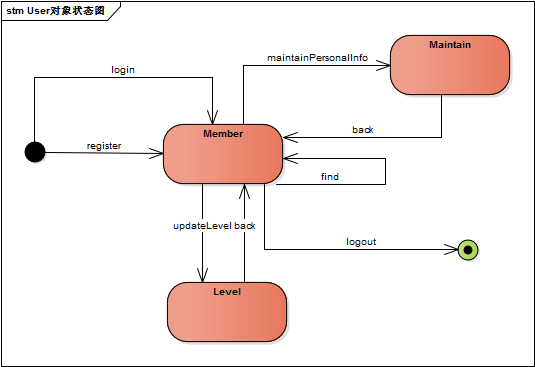


图 4-1-5-16 CreditRecord 对象执行增加操作的状态图

如图 4-1-5-17 所示的状态图描述了 Credit 对象的生存期间执行删除单据信息操作的状态序列，引起转移的事件，以及因状态转移而伴随的动作。如果 show 方法被 UI 调用，Credit 进入Presentation状态，然后 updateCredit方法被 UI 调用，Credit进入 Update 状态，然后可以调用updateLevel方法进入Level状态，也可以调用updateCreditRecord方法进入CreditRecord状态，也可以调用end方法进入结束状态，如果处于开始状态，可以调用updateCredit方法直接进入Update状态。

 图 4-1-5-17 Credit 对象执行删除操作的状态图

如图4-1-5-18所示的状态图描述了User对象的生存期间执行更新单据操作的状态序列，引起转移的事件，以及因状态转移而伴随的动作。如果 register方法或者login方法被 UI 调用，User 进入 Member 状态，如果maintainPersonalInfo方法被调用，则进入Maintain状态，如果updataLevel方法被调用，则进入Level状态，如果logout方法被调用，则进入结束状态。

 图 4-1-5-18 User 对象执行更新操作的状态图

# 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。

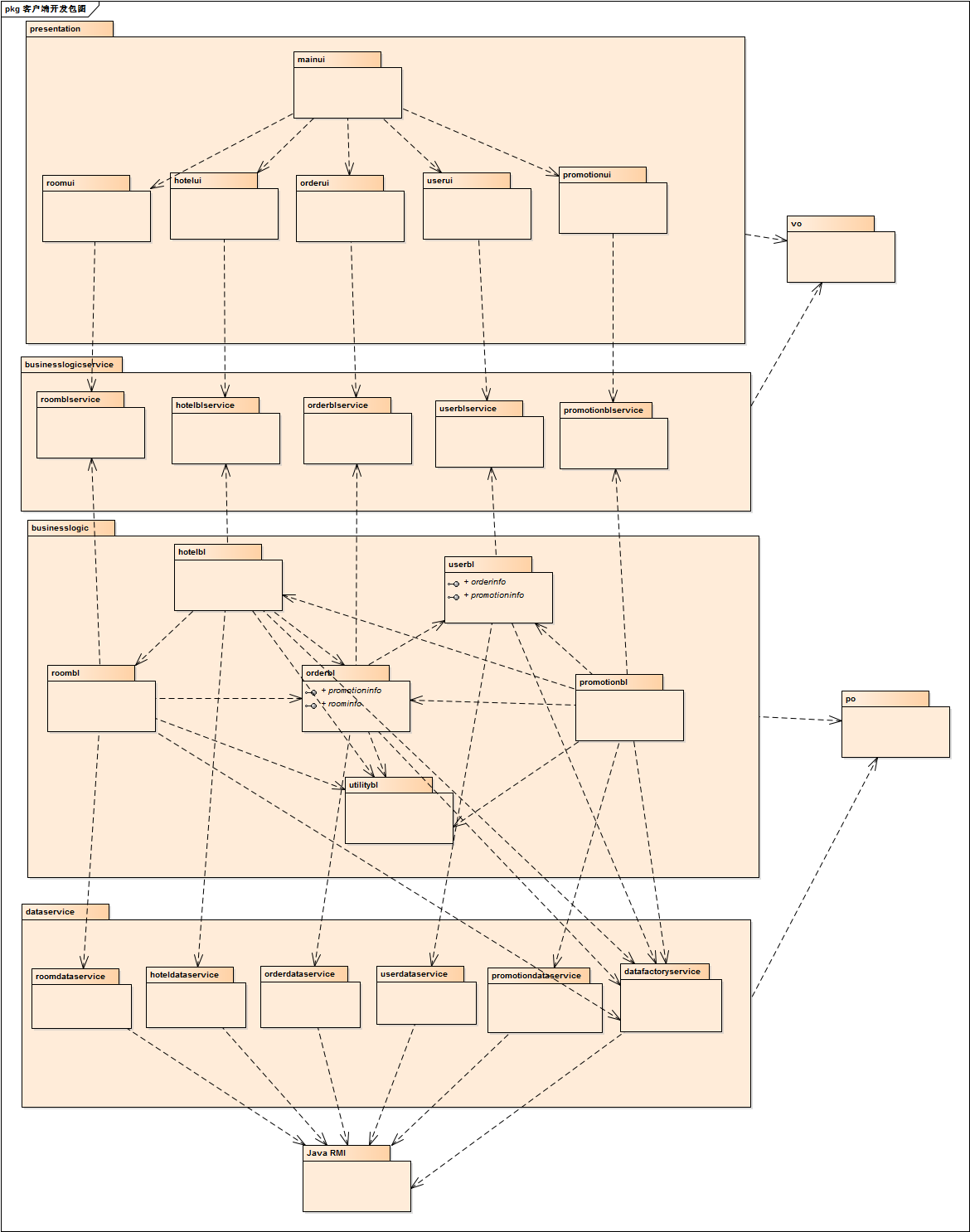
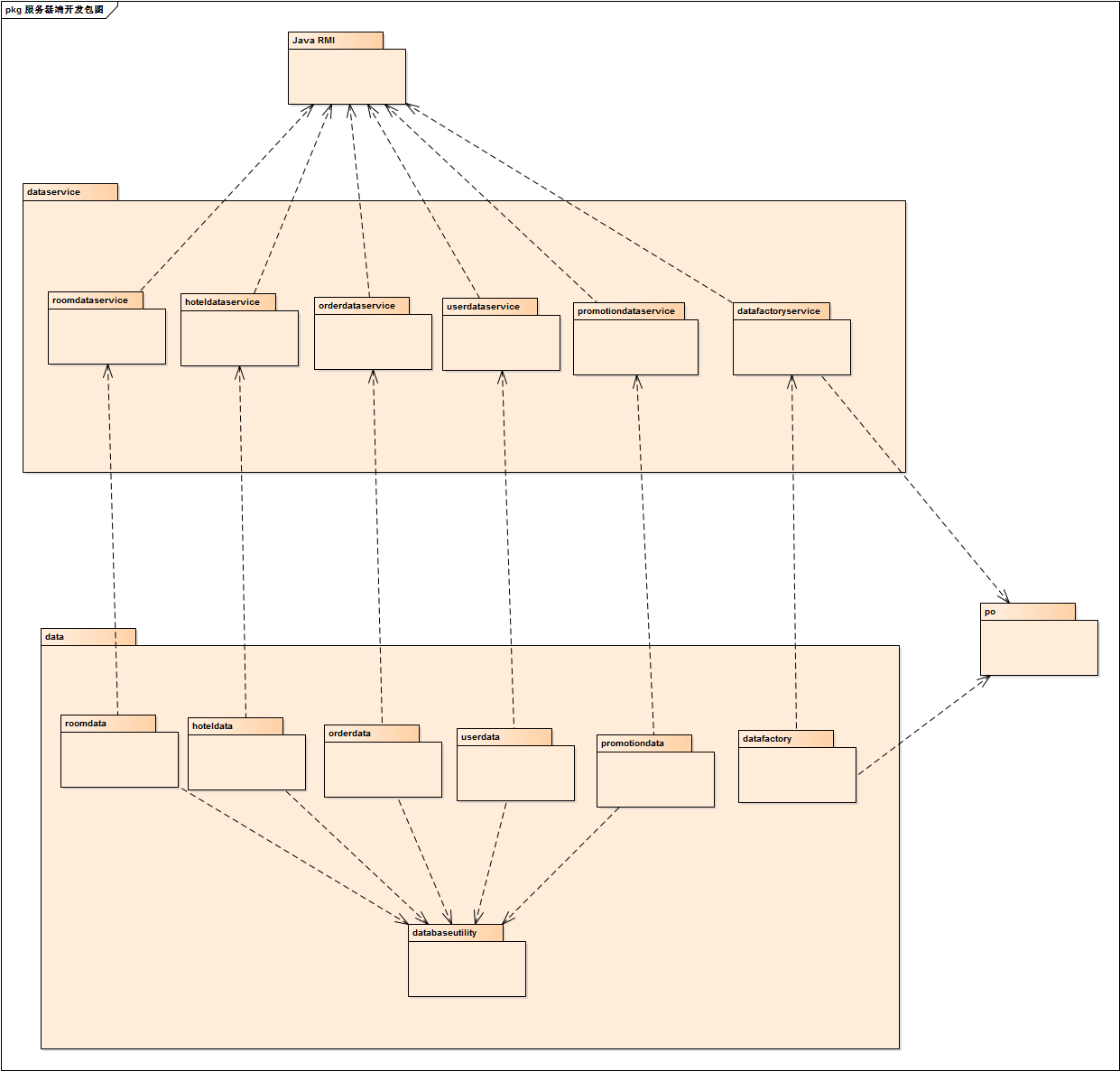


图 5-1 客户端包图

 图 5-2 酒店预订系统服务器端开发包图