曹利航 段梦洋 刘港欢 刘钦

酒店预订系统

软件详细设计模型

软件详细设计模型

**1.业务逻辑层的分解**

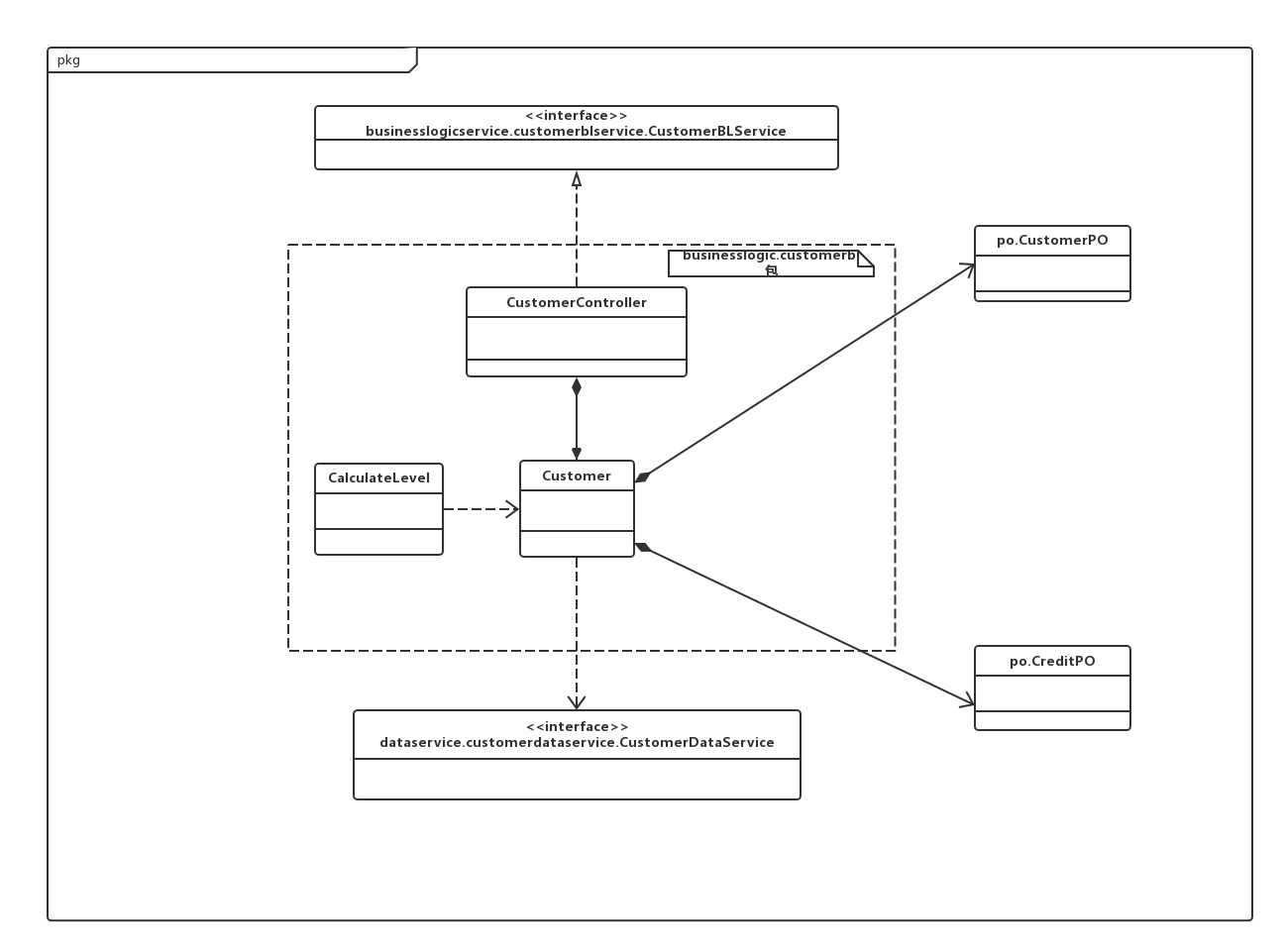
业务逻辑层的开发包图参见软件体系结构文档图3。

**1.1 customerbl模块**

1.1.1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

1.1.2业务逻辑层的动态模型

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。

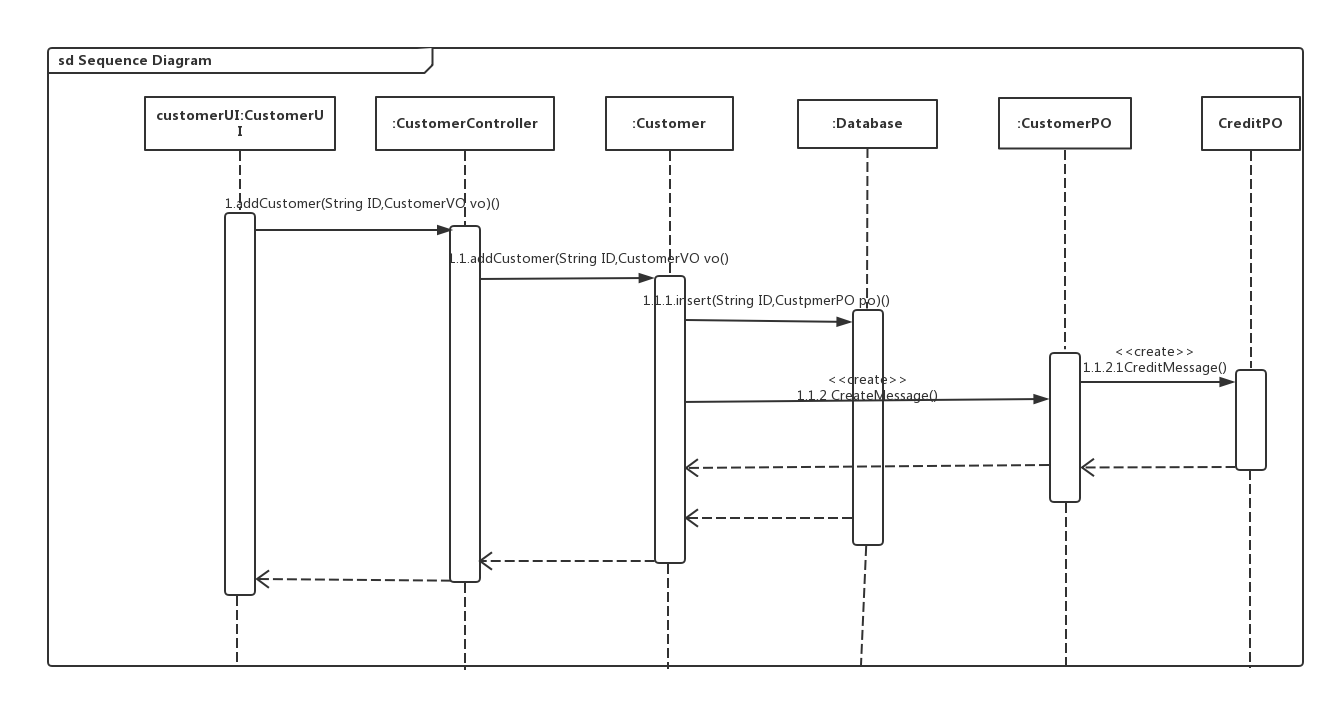
图1-1 注册顺序图

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。

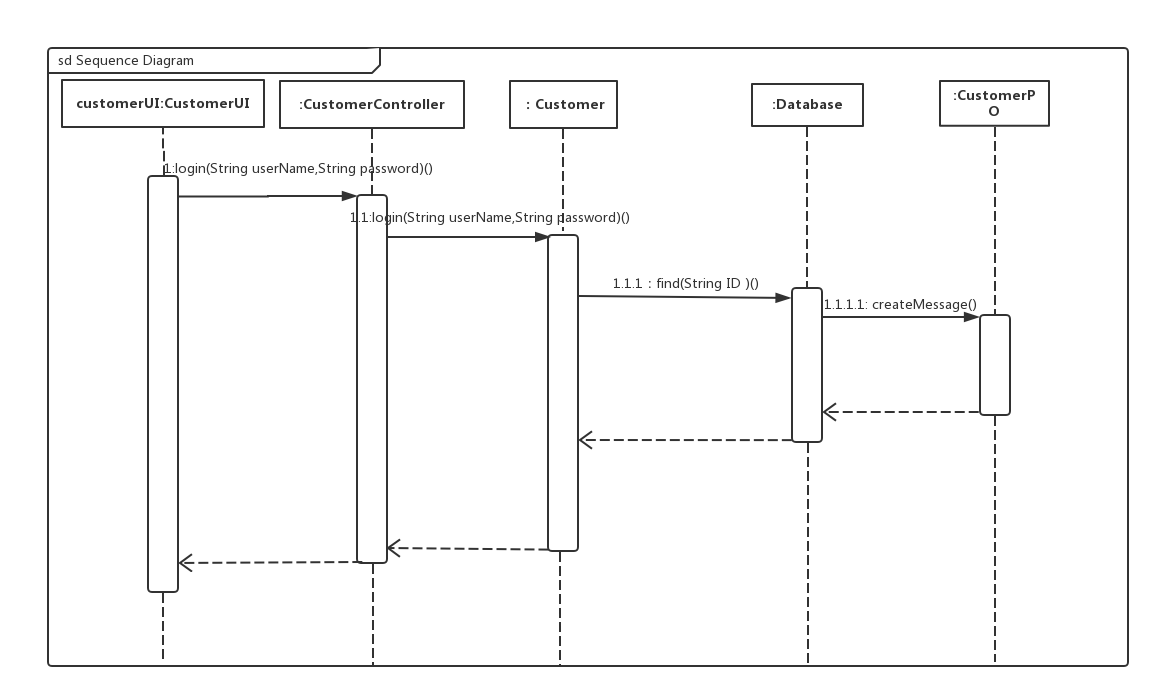
图1-2 登录顺序图

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。

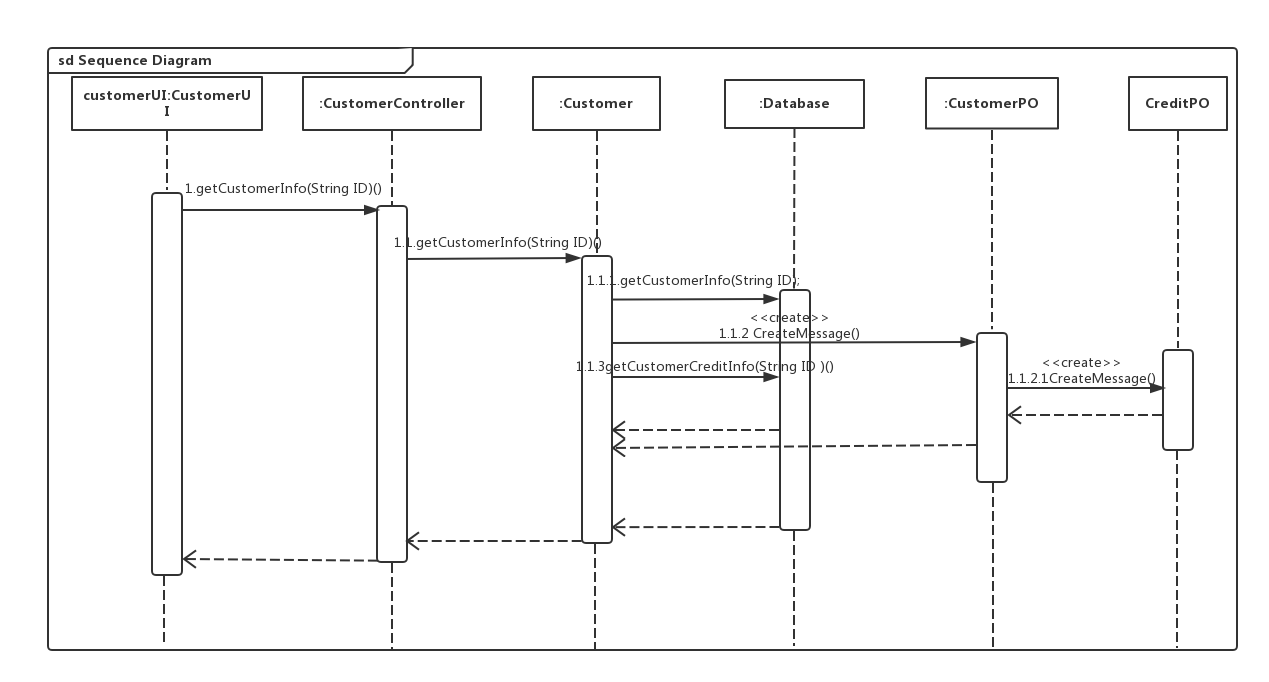


图1-3 查看会员信息的顺序图

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。

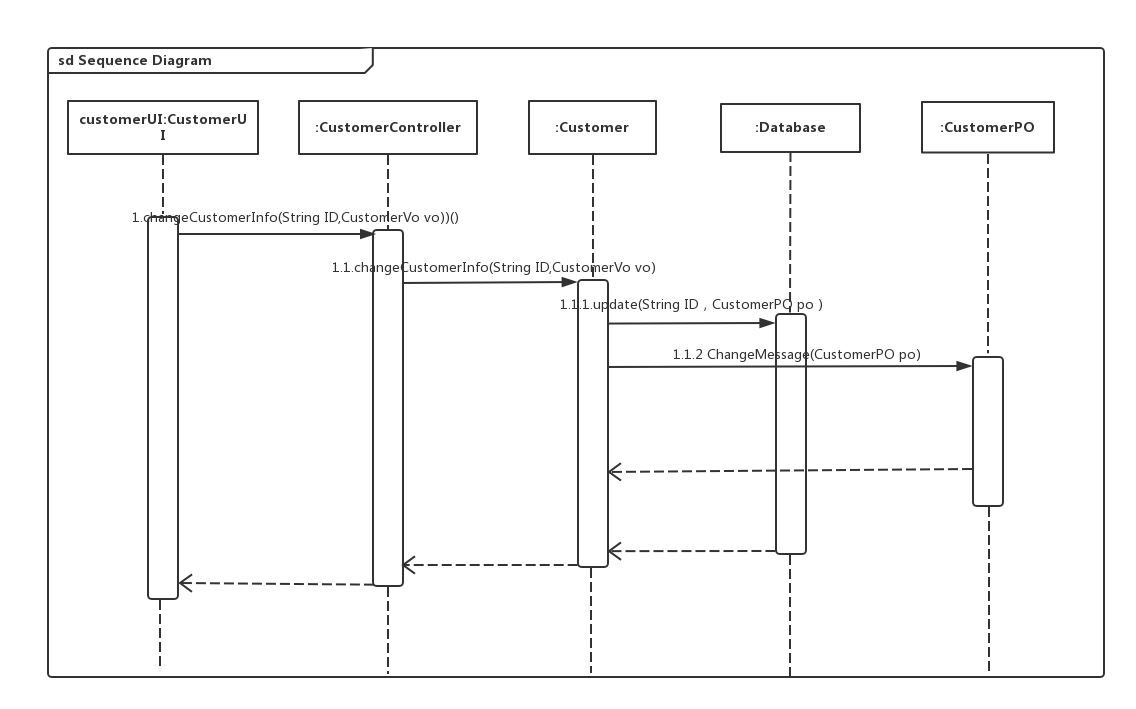


图1-4 修改会员信息的顺序图

图1-5表明了酒店预订系统中，当下订单根据等级确定促销力度之后，顾客业务逻辑处理的相关对象之间的协作。

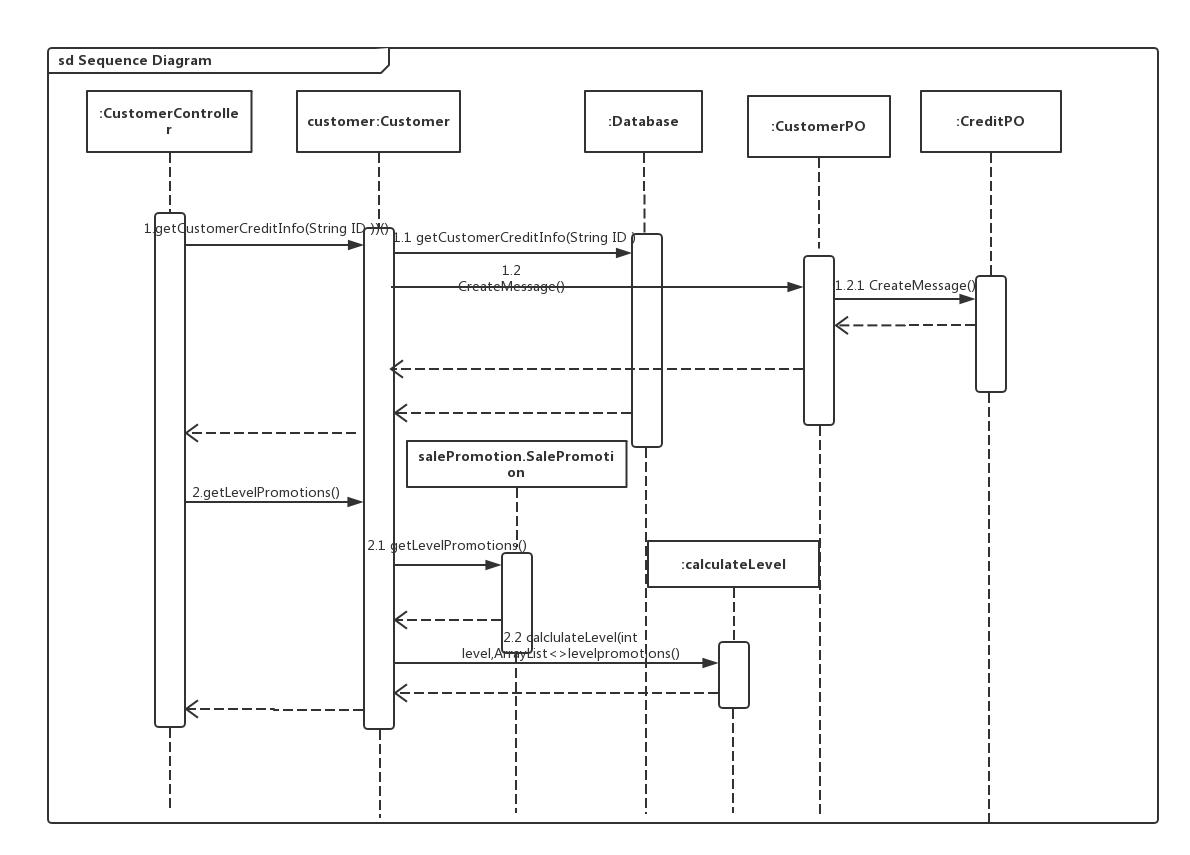
图1-5 查看会员等级的顺序图

图1-6表明了酒店预订系统中，当会员信用随订单情况发生变化之后，顾客业务逻辑处理的相关对象之间的协作。

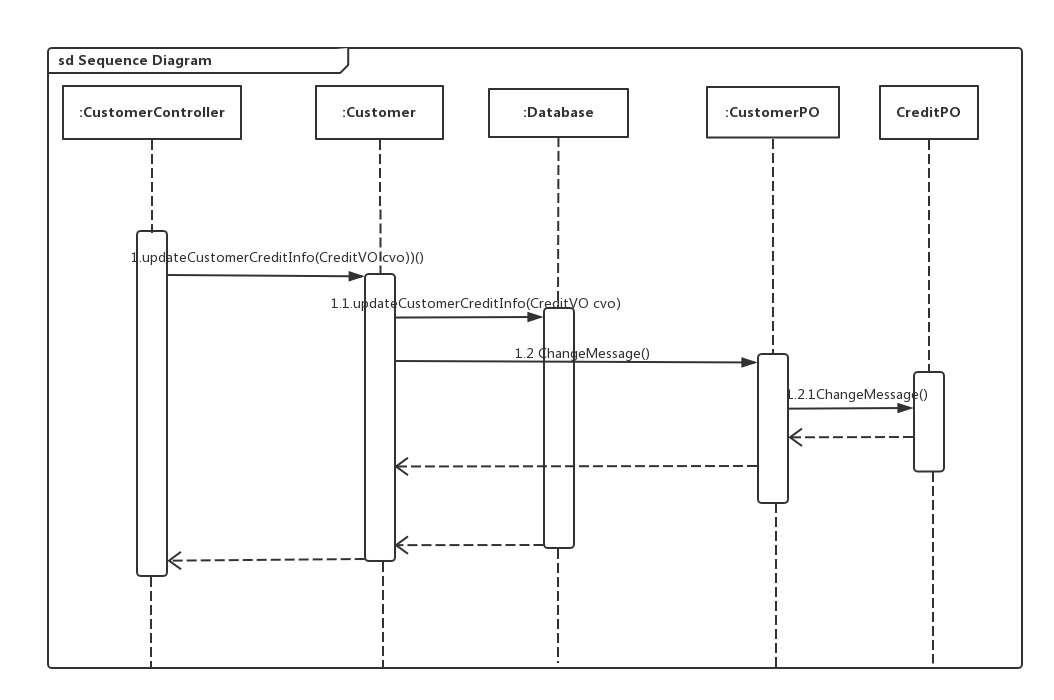


图1-6 修改会员信用的顺序图

如图1-7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,Controller也可以调用chamgeCreditInfo使Customer进入omdifyCredit状态，调用getCustomerLevel使Customer进入getCustomerLevel状态。

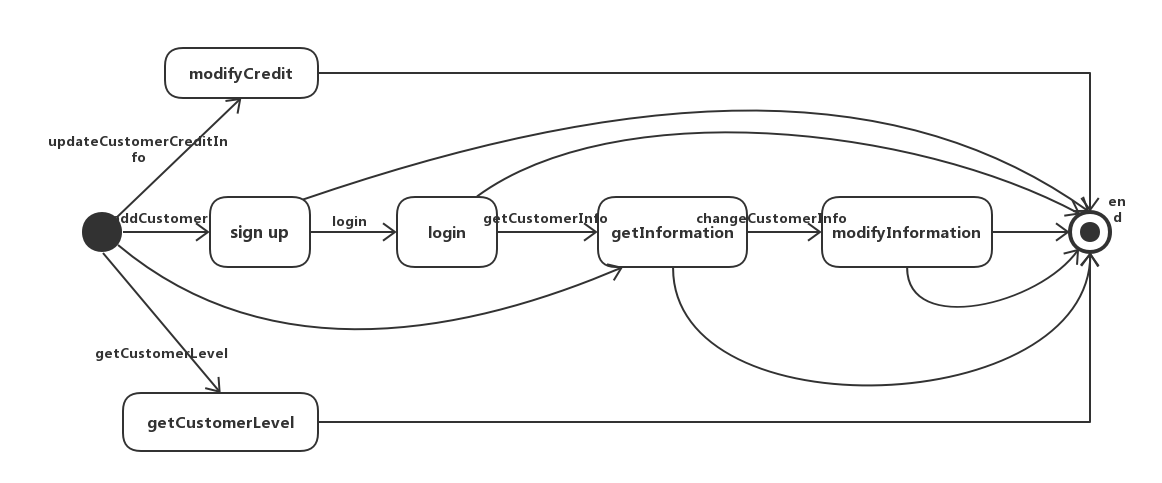


图1-7 Customer对象状态图

**1.2 managerbl模块**

1.2.1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。

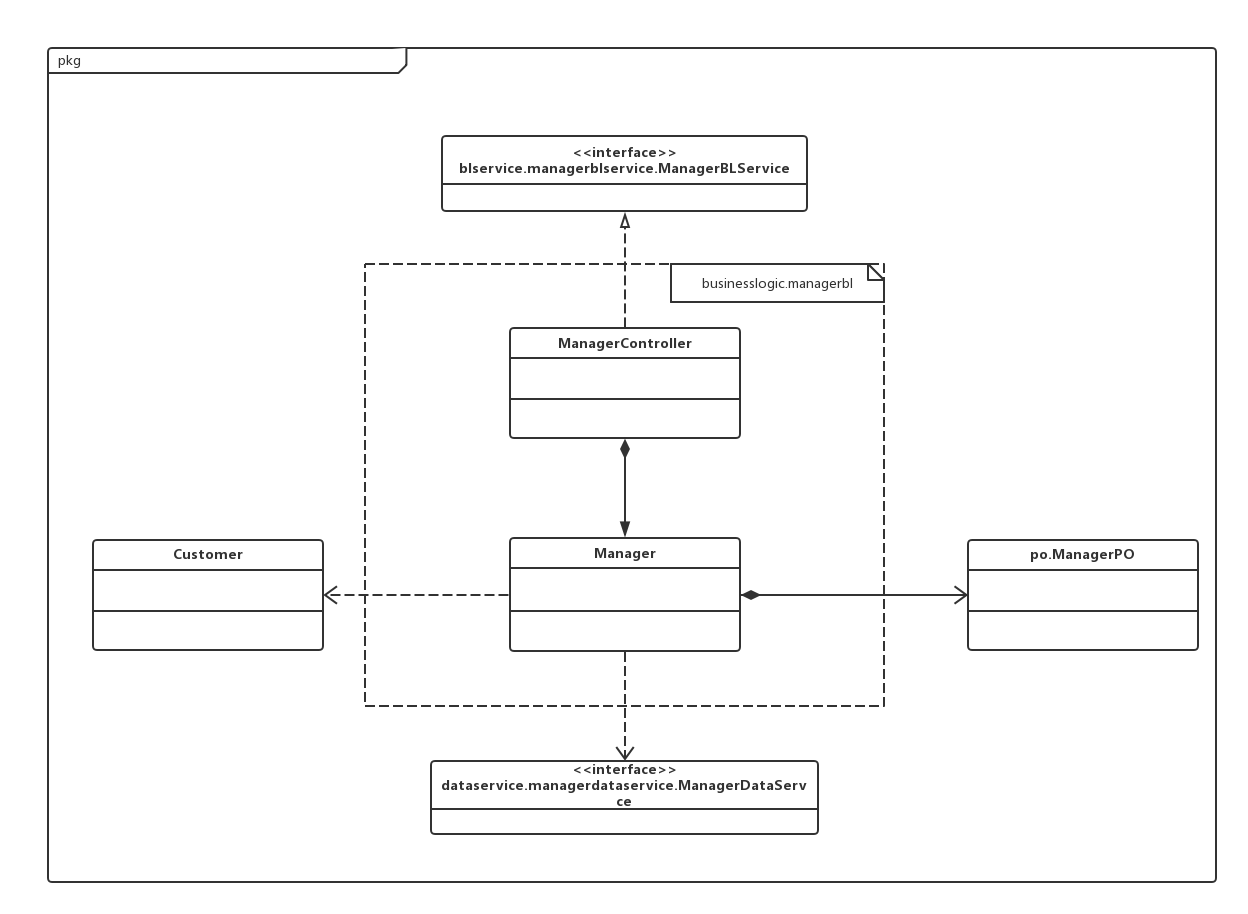


图2 managerbl模块各个类的设计

1.2.2业务逻辑层的动态模型

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。

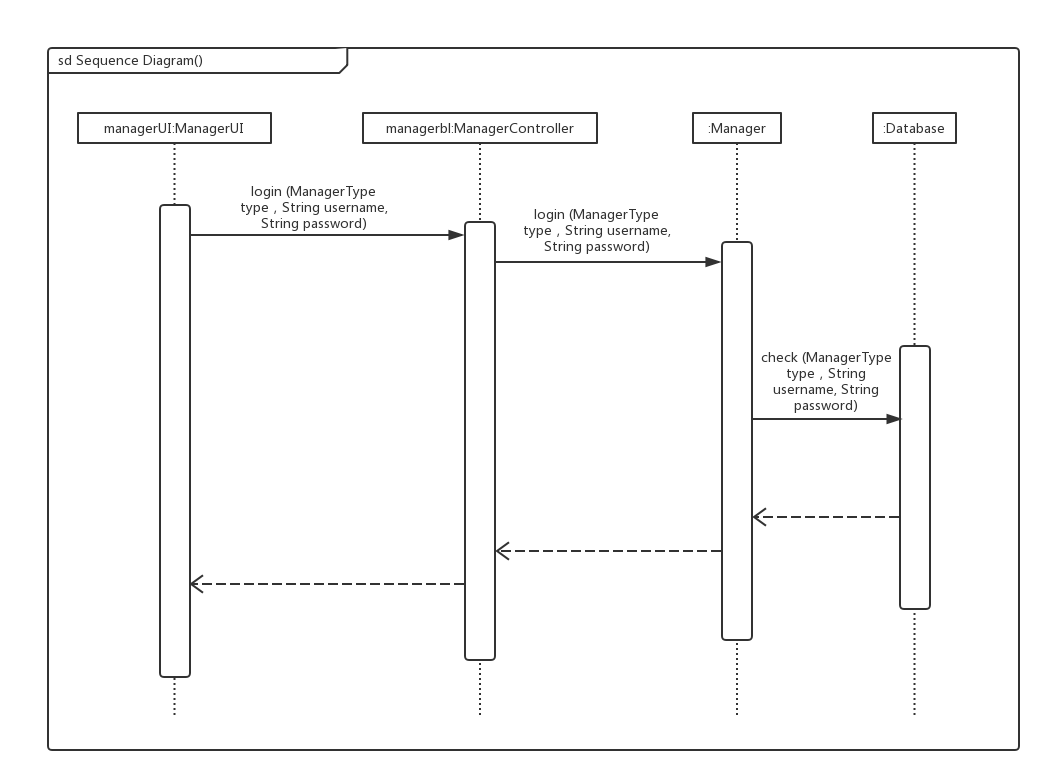


图2 – 1 管理人员登录顺序图

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。

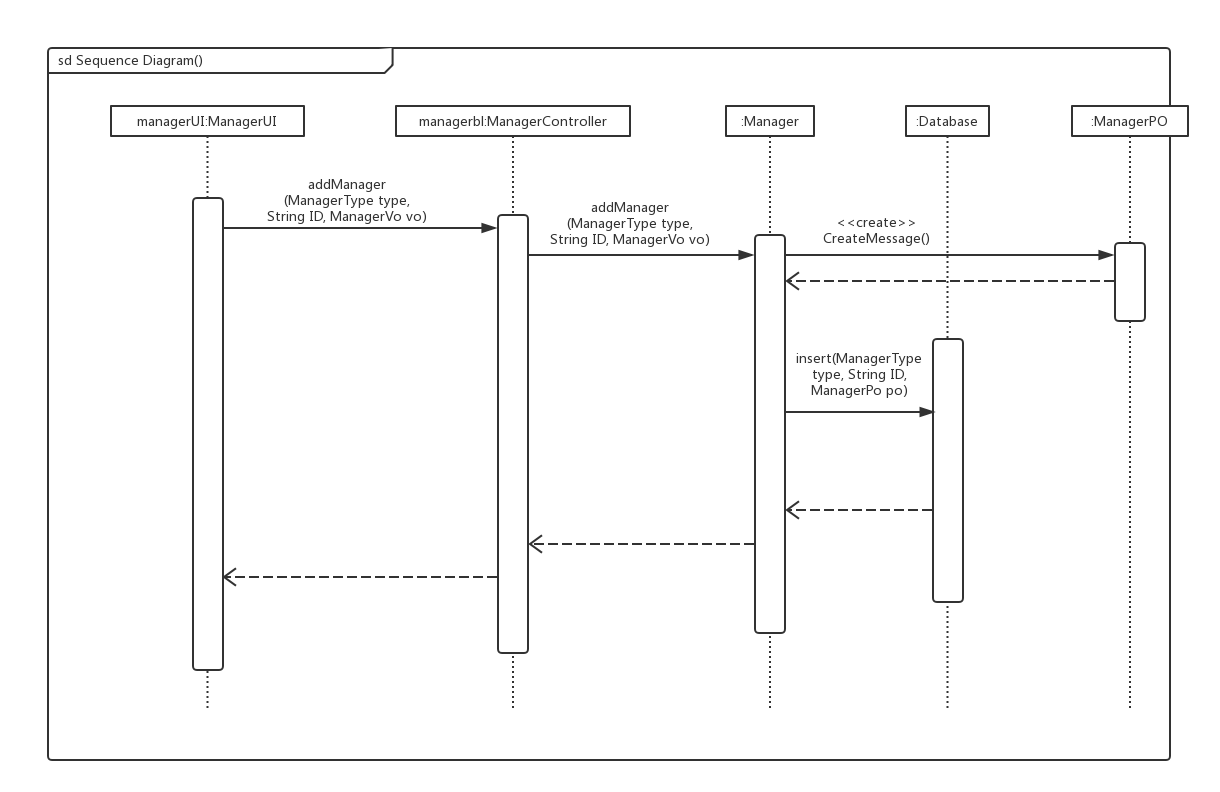


图2 – 2 添加管理人员顺序图

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。

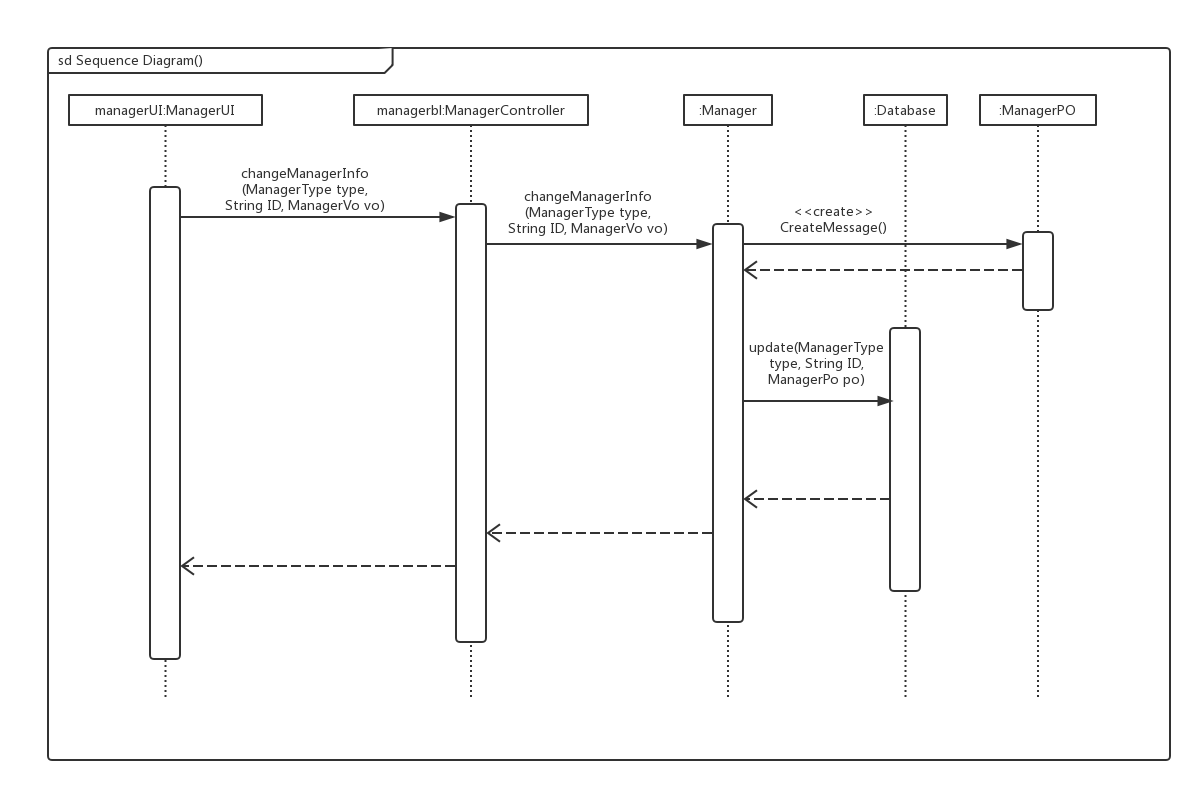


图2 – 3 修改管理人员信息顺序图

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。

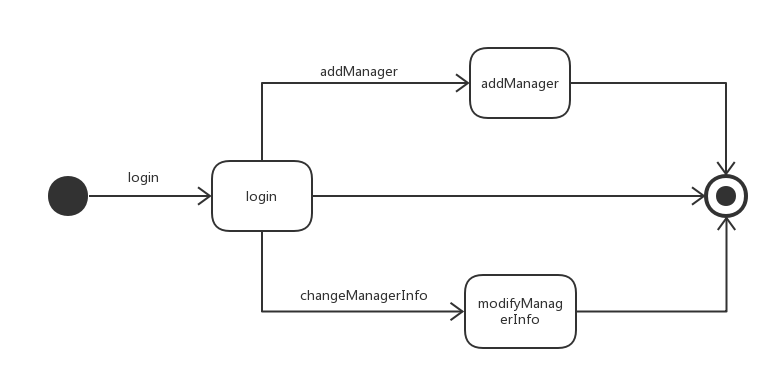


图2-4 Manager对象状态图

**1.3hotelbl模块**

1.3.1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,AvailableRoomPO,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。

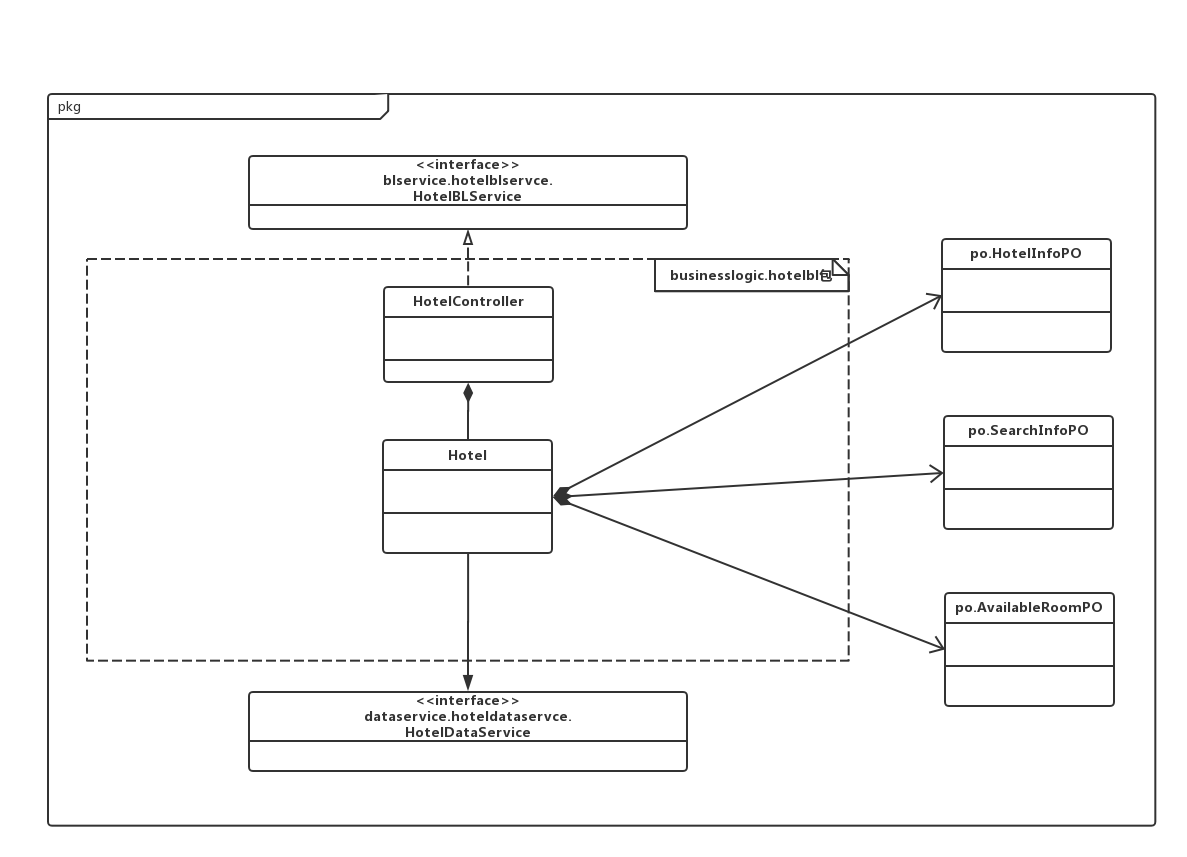


图3 hotelbl模块各个类的设计

1.3.2业务逻辑层的动态模型

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。

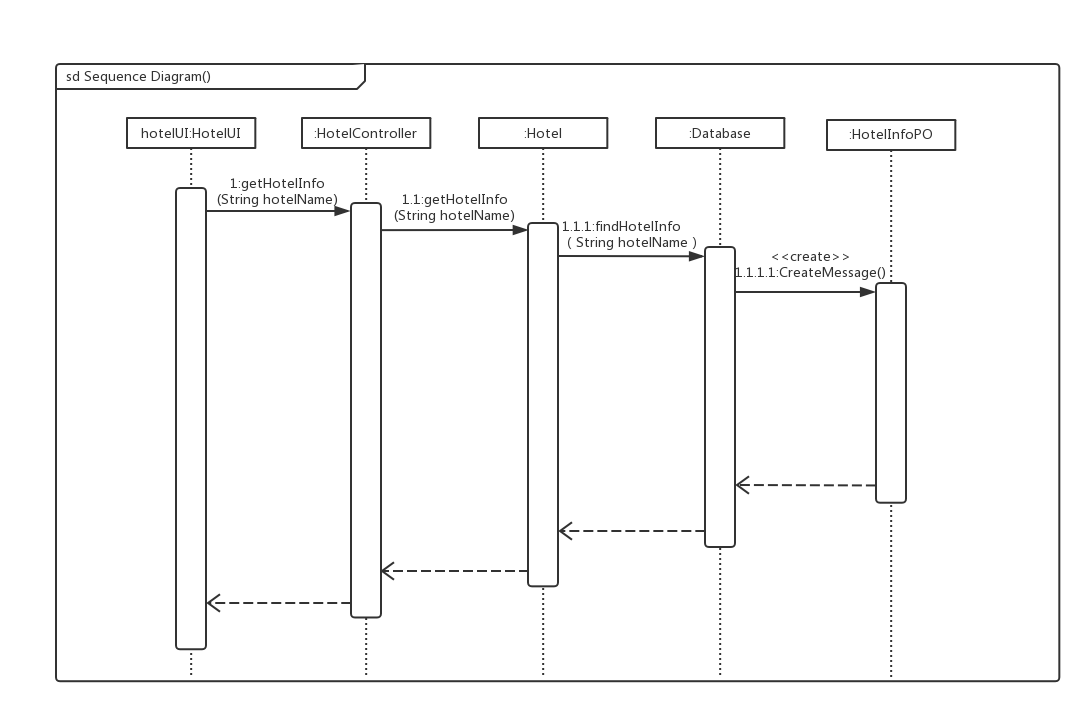


图3-1 查看酒店基本信息顺序图

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。

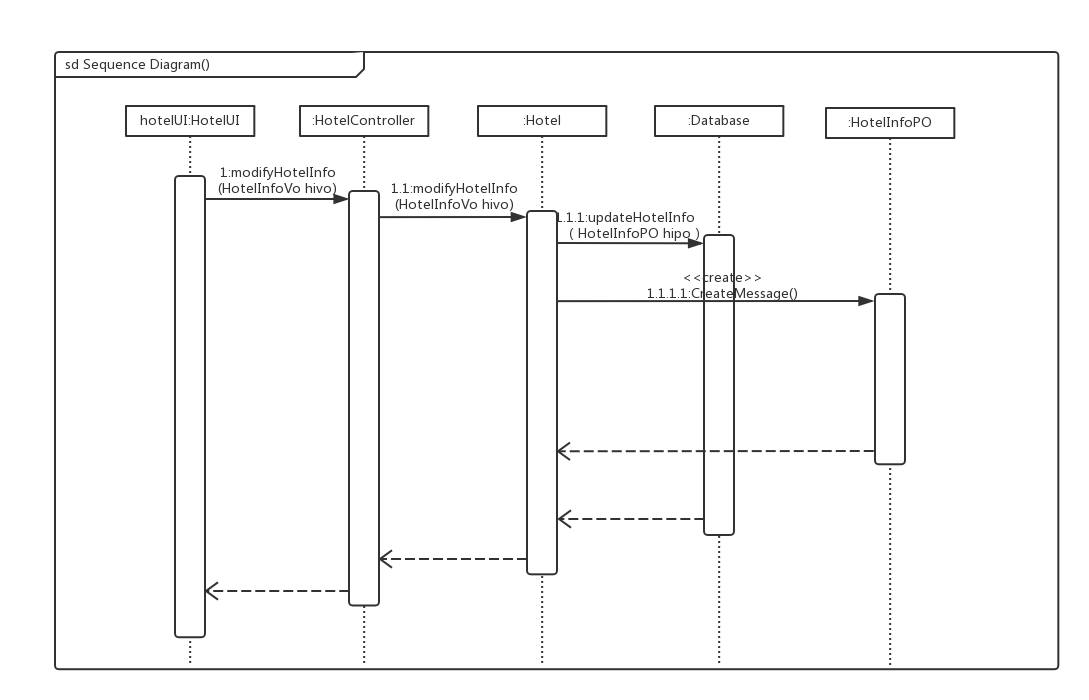


图3-2 修改酒店基本信息顺序图

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

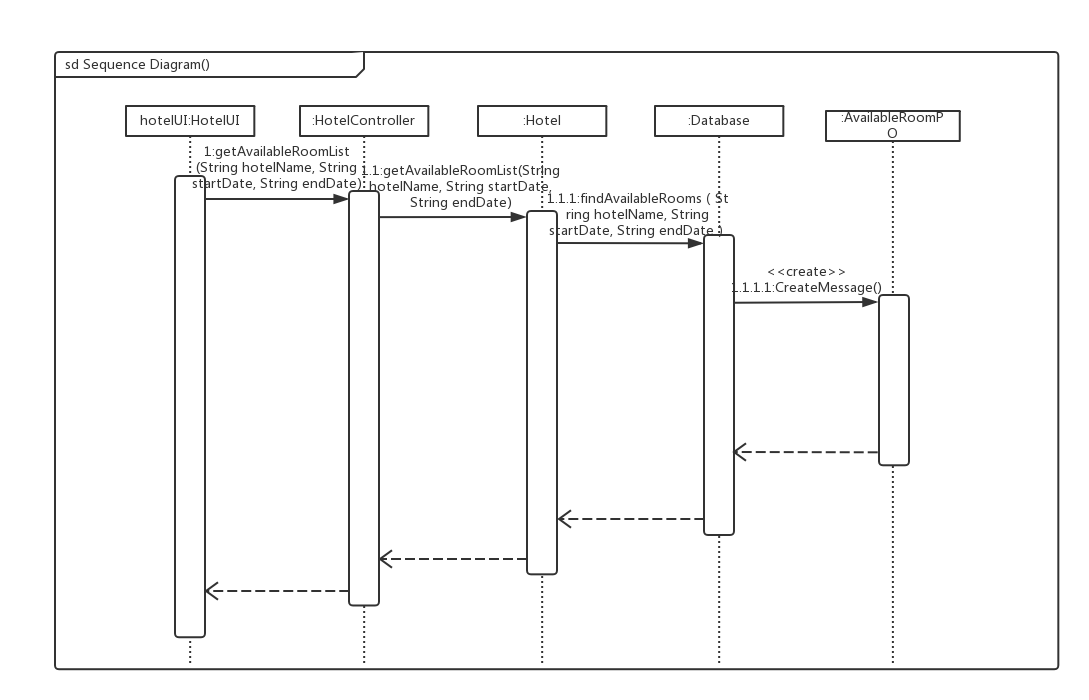


图3-3 查看可用客房的顺序图

图3-4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，酒店业务逻辑处理的相关对象之间的协作。

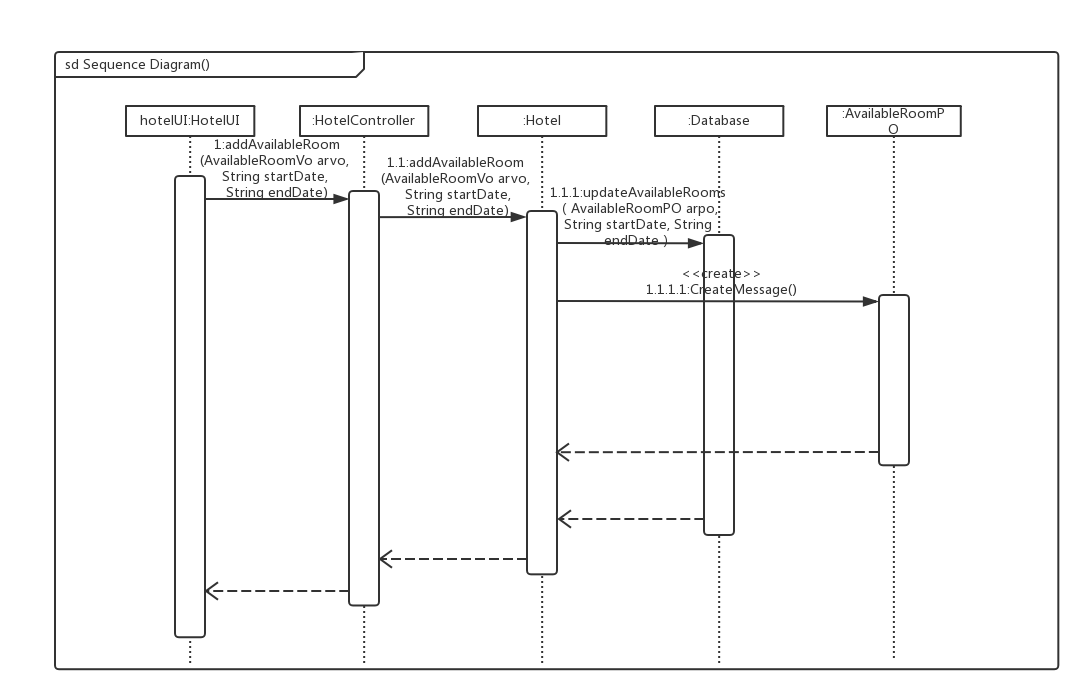


图3-4 添加可用客房的顺序图

图3-5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，酒店业务逻辑处理的相关对象之间的协作。

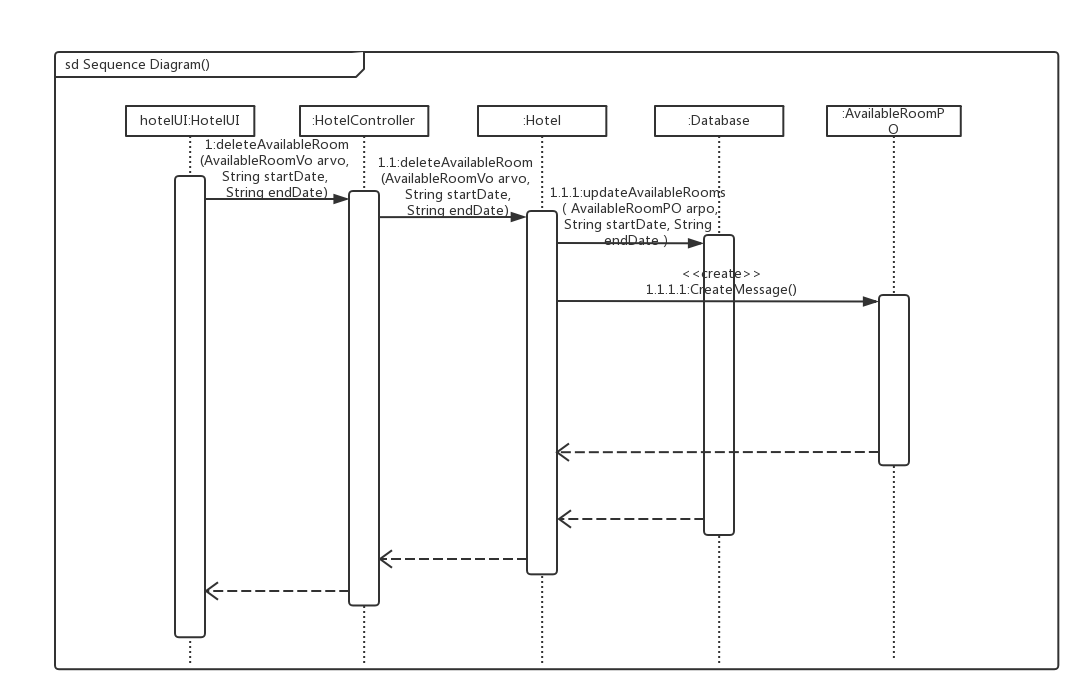


图3-5 删除可用客房的顺序图

图3-6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。

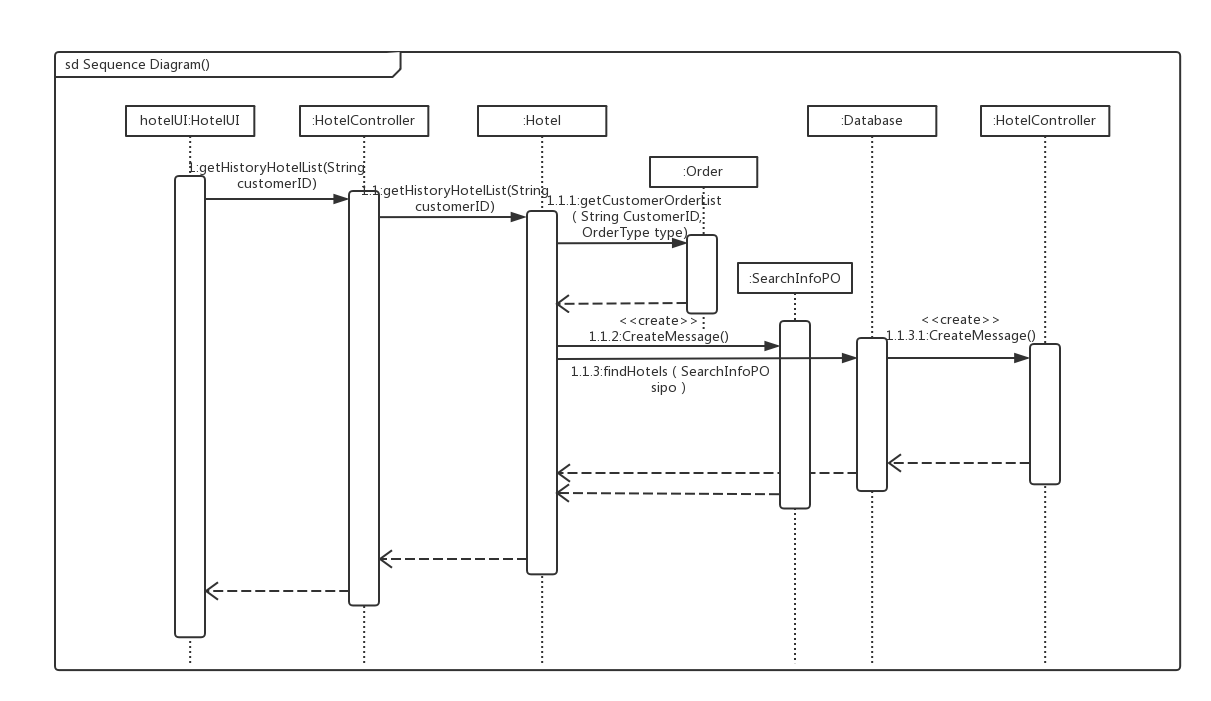


图3-6 查看住过的酒店的顺序图

图3-7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。

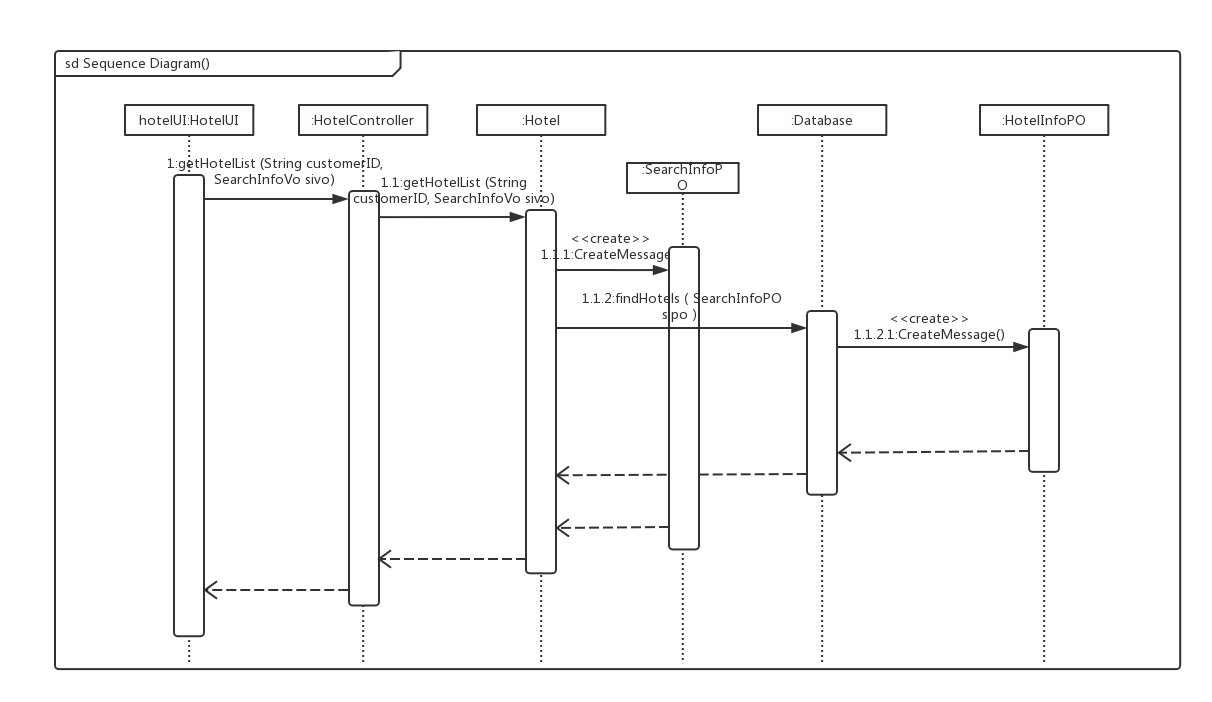


图3-7 搜索酒店的顺序图

图3-8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。

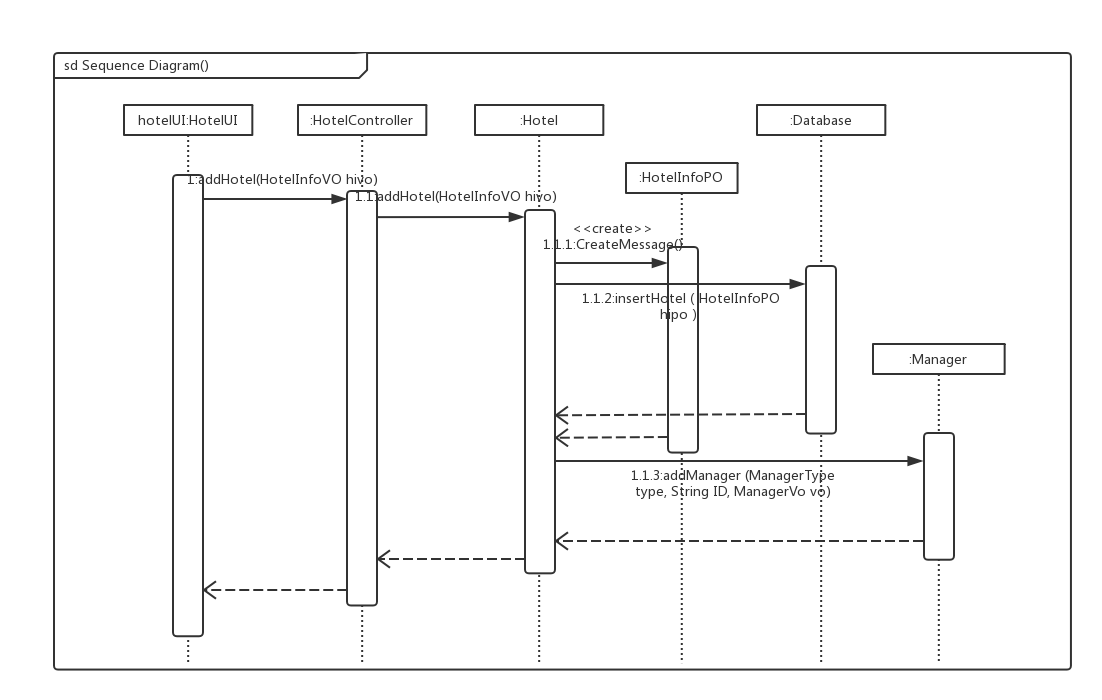


图3-8 添加酒店的顺序图

如图3-9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getAvailableRoomList使Hotel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Hotel进入getHotelList状态。

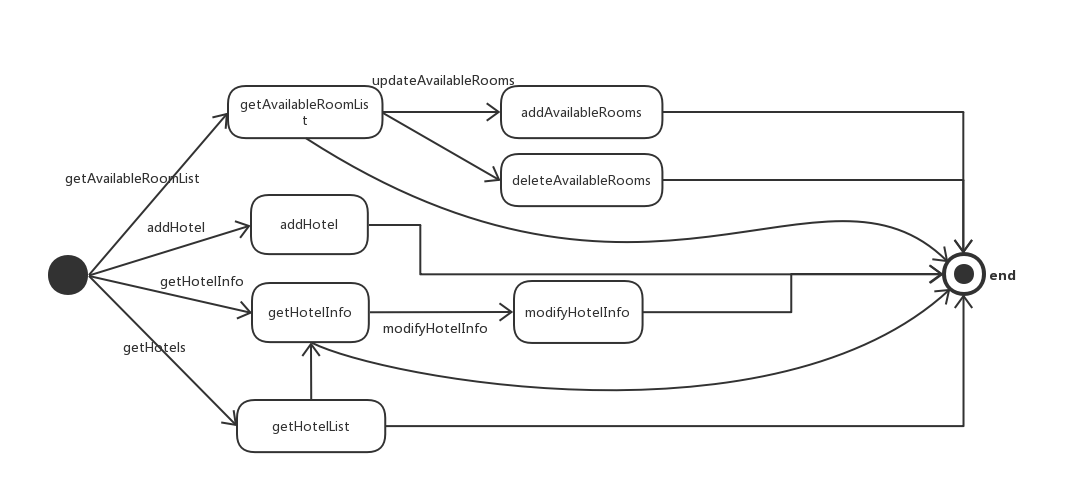


图3-9 Hotel对象状态图

**1.4orderbl模板**

1.4.1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。

reviewbl模块的设计如图7所示。

OrderList 
OrderLineltem 
«interface» 
businesslogicservice.orderblservice.OrderBLService 
po.OrderPO 
OrderControler 
«interface» 
Order 
CustomerCredit 
«interface» 
dataService.orderdataservice.DataService 
businesslogic.customerbl.Custom 

图4-orderbl模块各个类的设计

1.4.2业务逻辑层的动态模型

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。

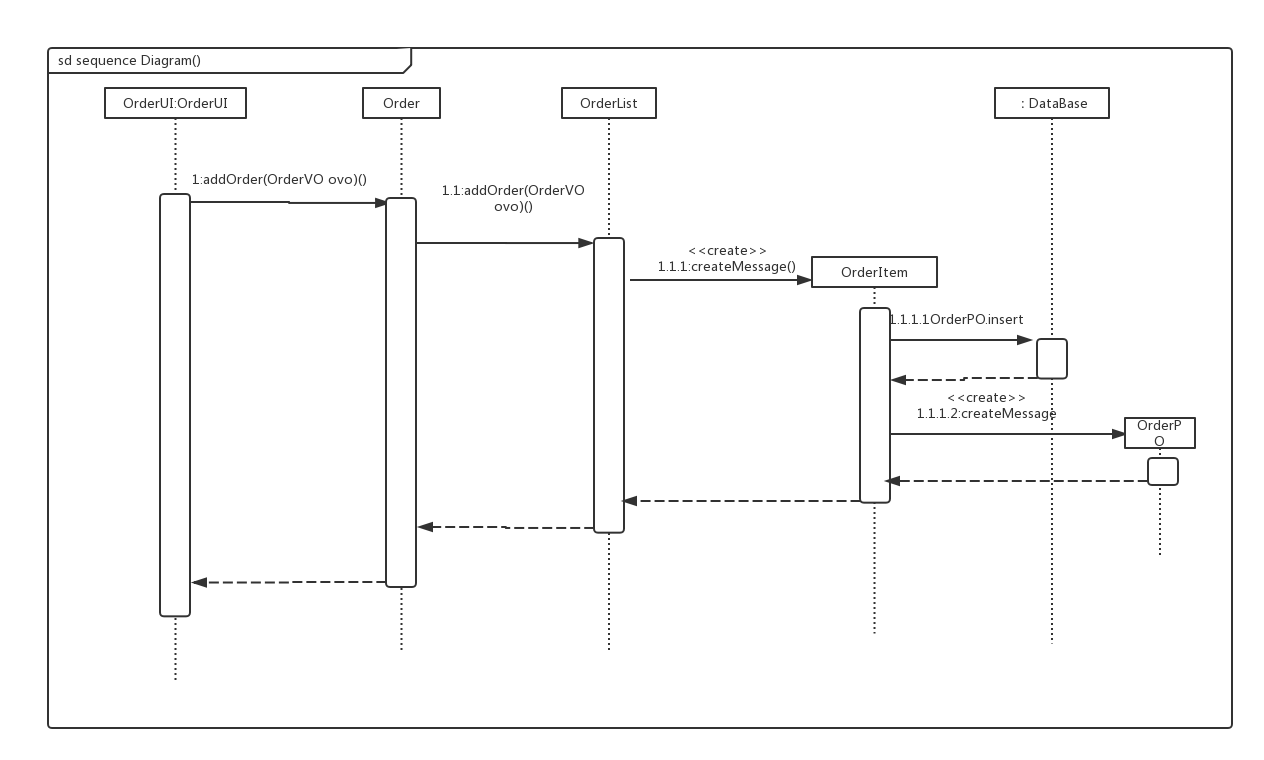


图4-1生成订单的顺序图

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。

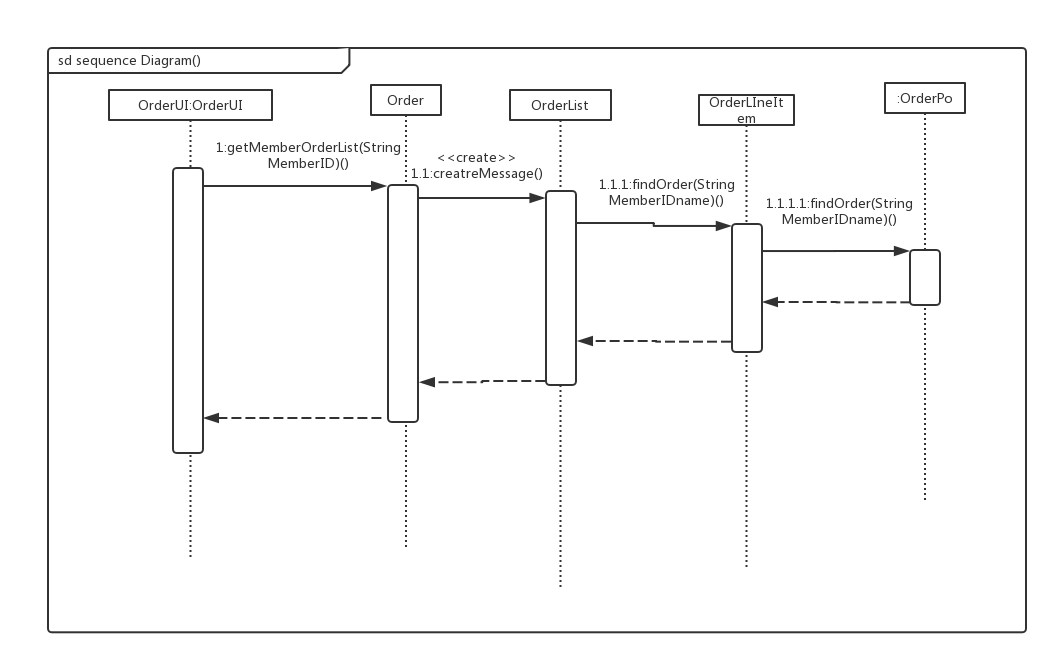


图4-2获取会员订单列表的顺序图

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

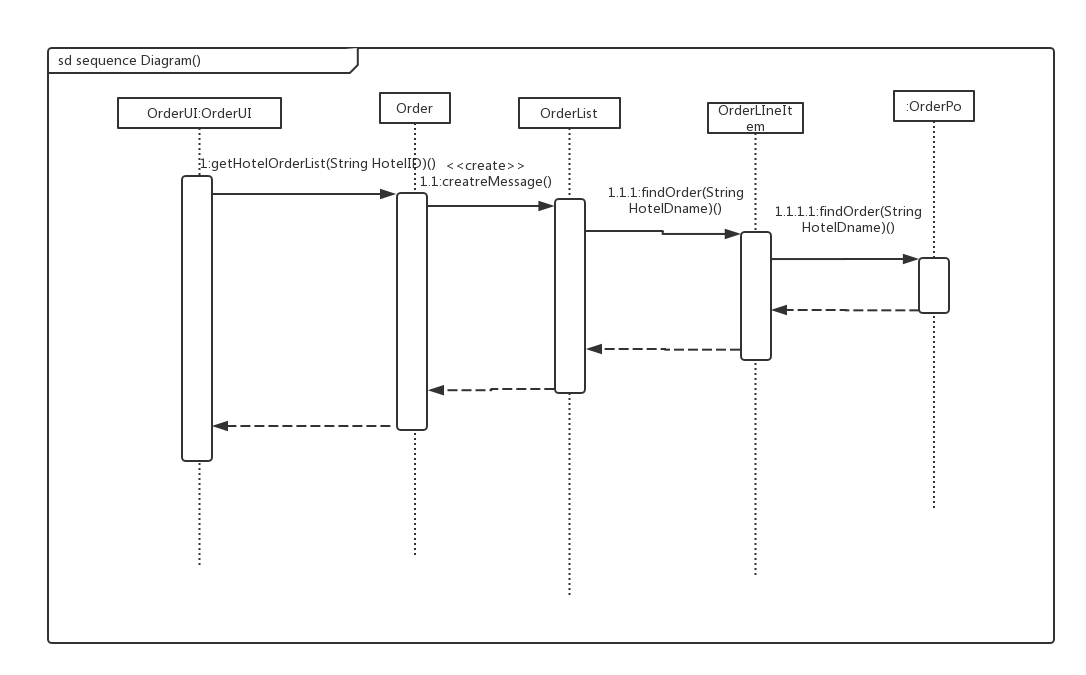


图4-3获取酒店订单列表的相关对象之间的协作

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

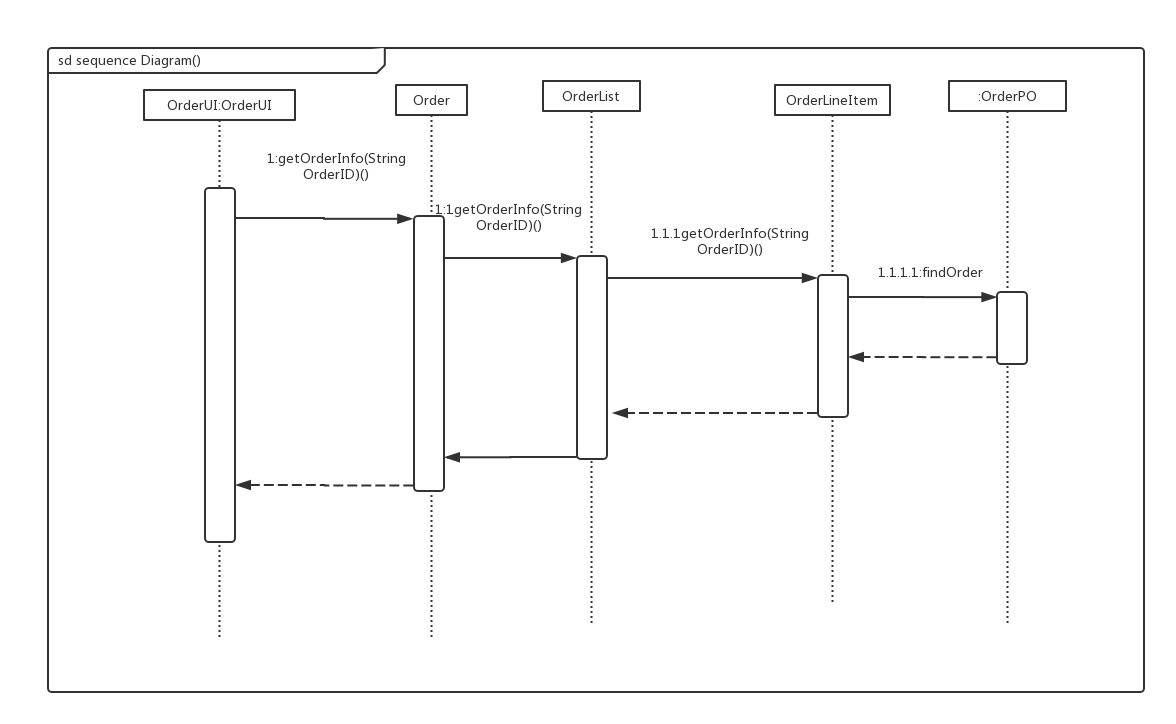


图4-4获取订单信息的顺序图

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

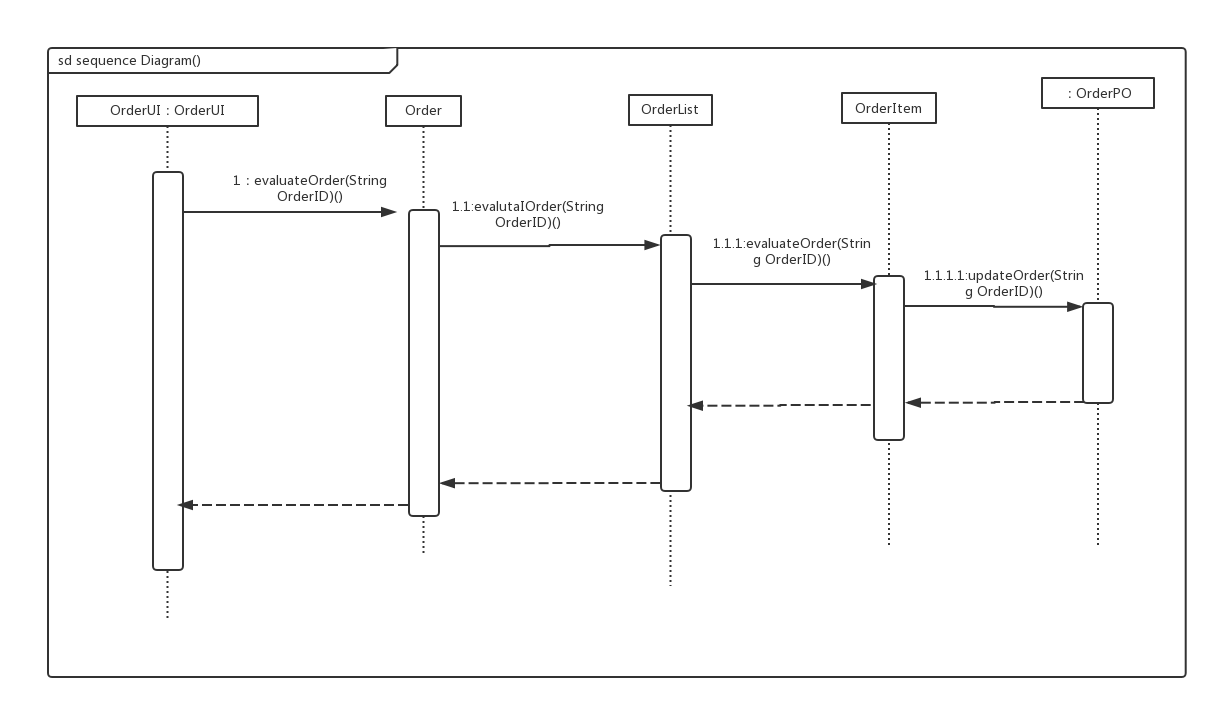


图5-5评价订单的顺序图

如图表明在酒店预订系统中，取消订单中相关对象的协作

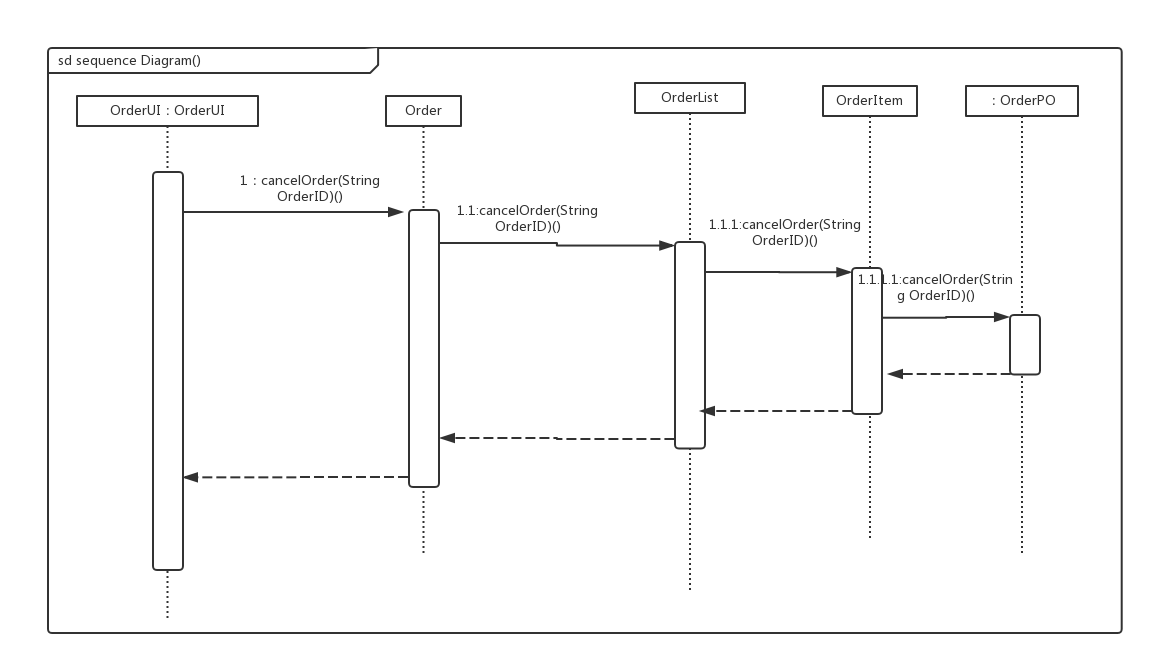


图5-6取消订单的顺序图

如图表明在酒店预订系统中，执行订单各对象之间的协作

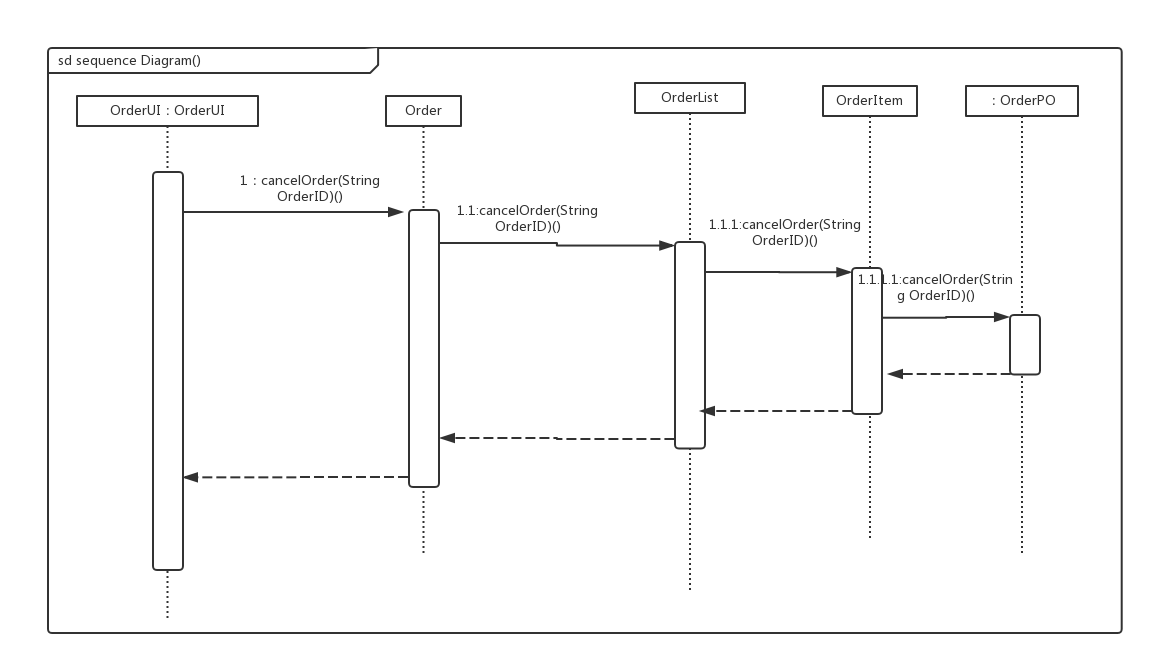
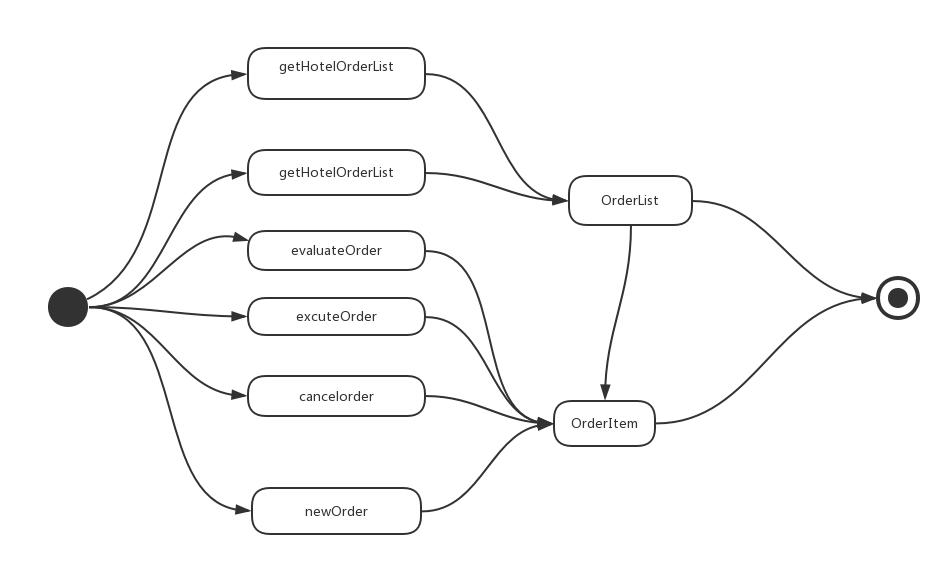


图5-6执行订单的顺序图

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。

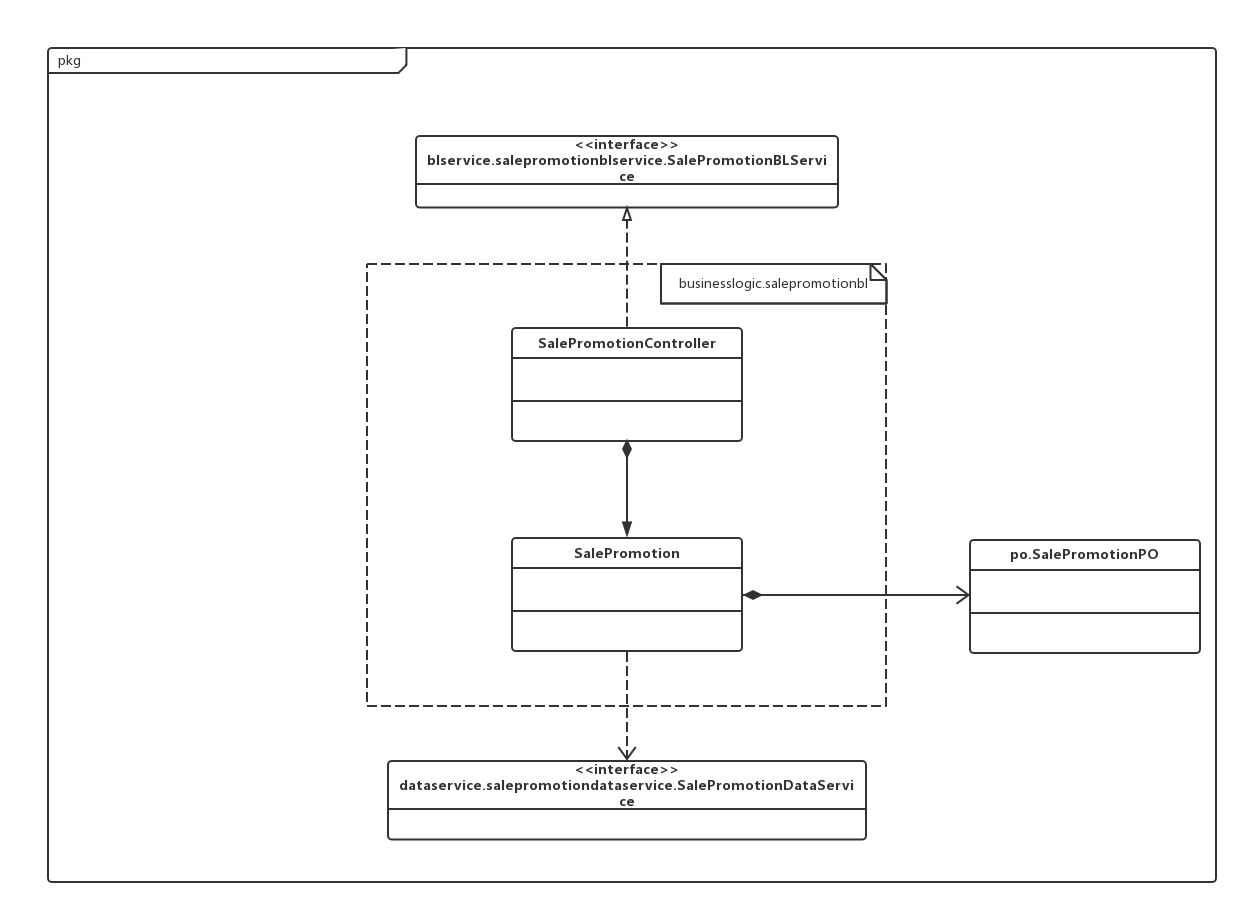


**1.5 salepromotionbl模块**

1.5.1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.salepromotionblservice.SalePromotionBLService接口。业务逻辑层和数据层之间添加 dataservice.salepromotiondataservice.SalePromotionDataService接口。为了隔离业务逻辑职责和逻辑控制职责,我们增加了 SalePromotionController,这样SalePromotionrController会将对审查的业务逻辑处理委托给SalePromotion对象。SalePromotionPO是作为审查记录的持久化对象被添加到设计模型中去的。

salepromotionbl模块的设计如图2所示。



1.5.2业务逻辑层的动态模型

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。

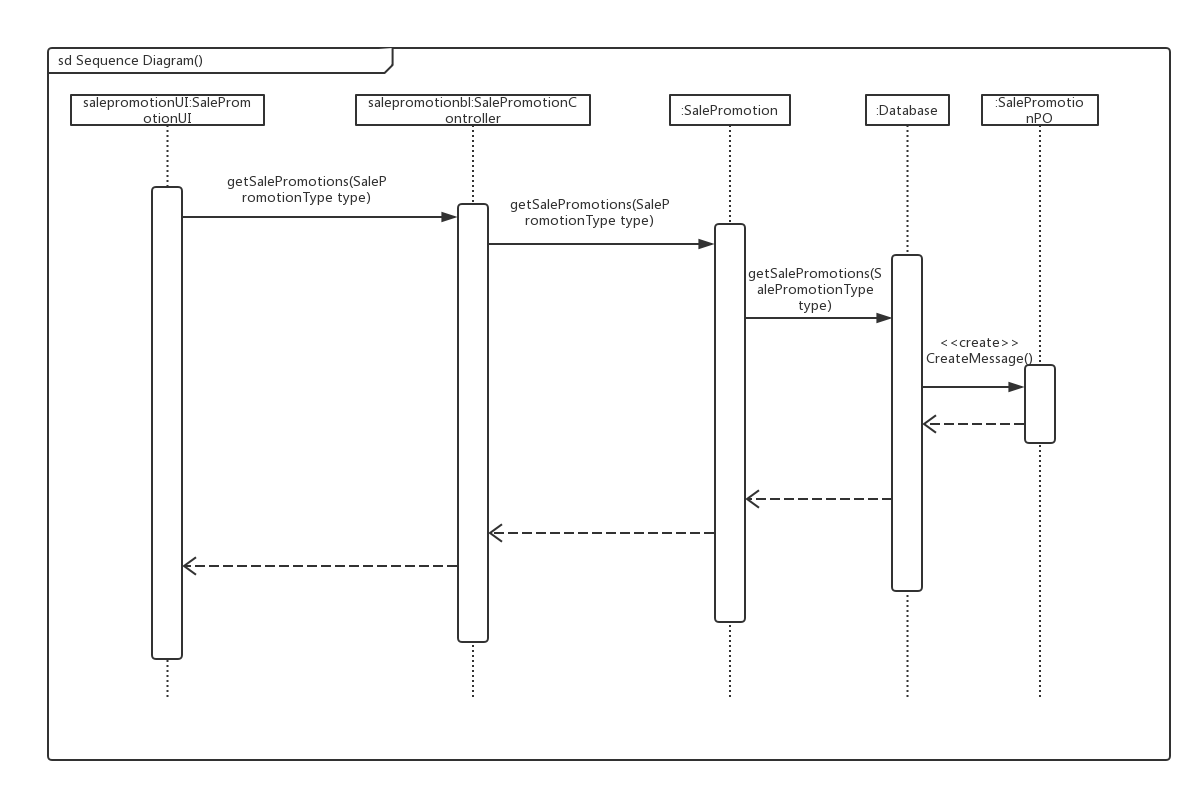


图5-1 查看促销策略

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。

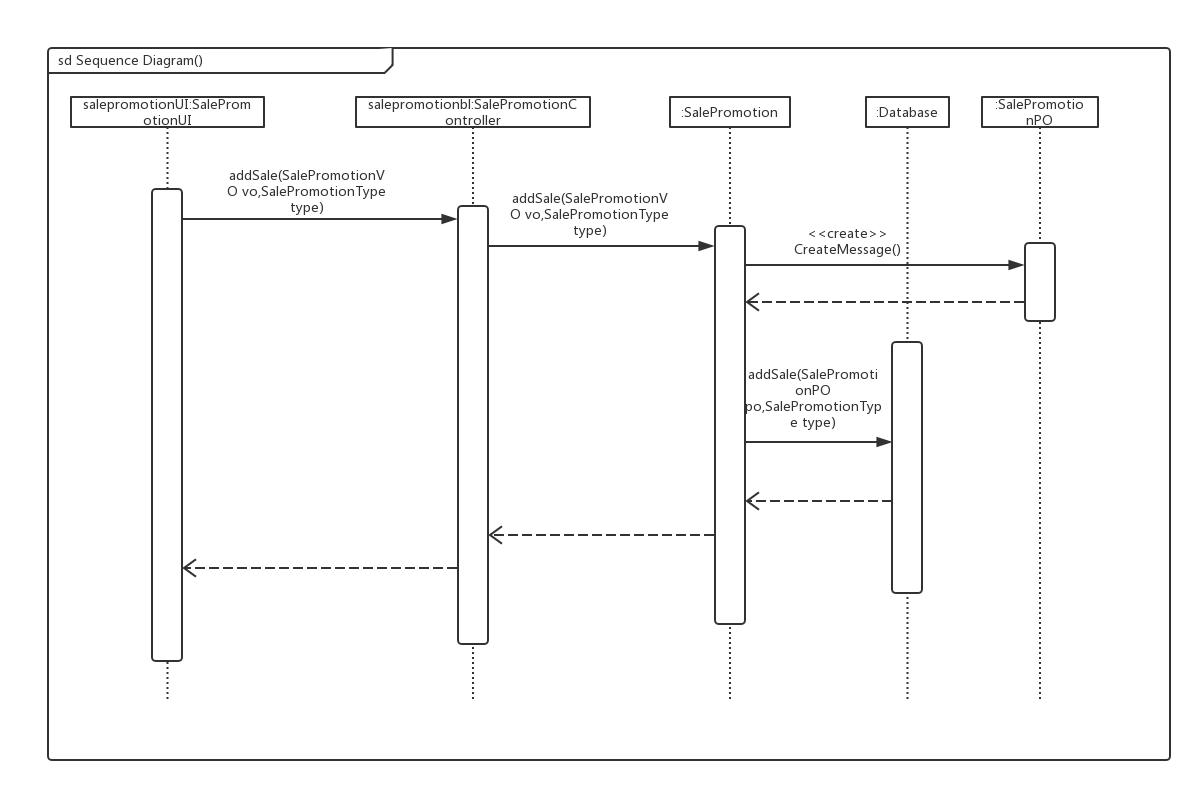


图5-2 添加促销策略

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。

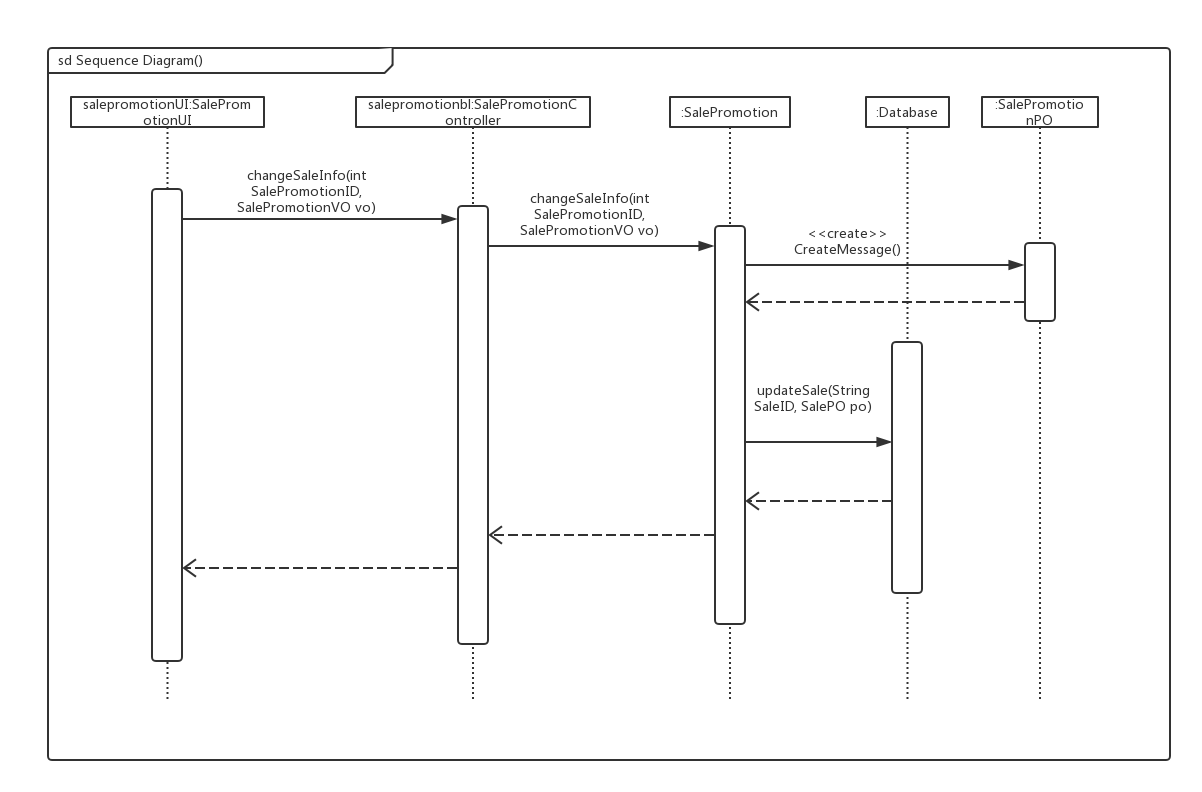


图5-3 修改促销策略

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。

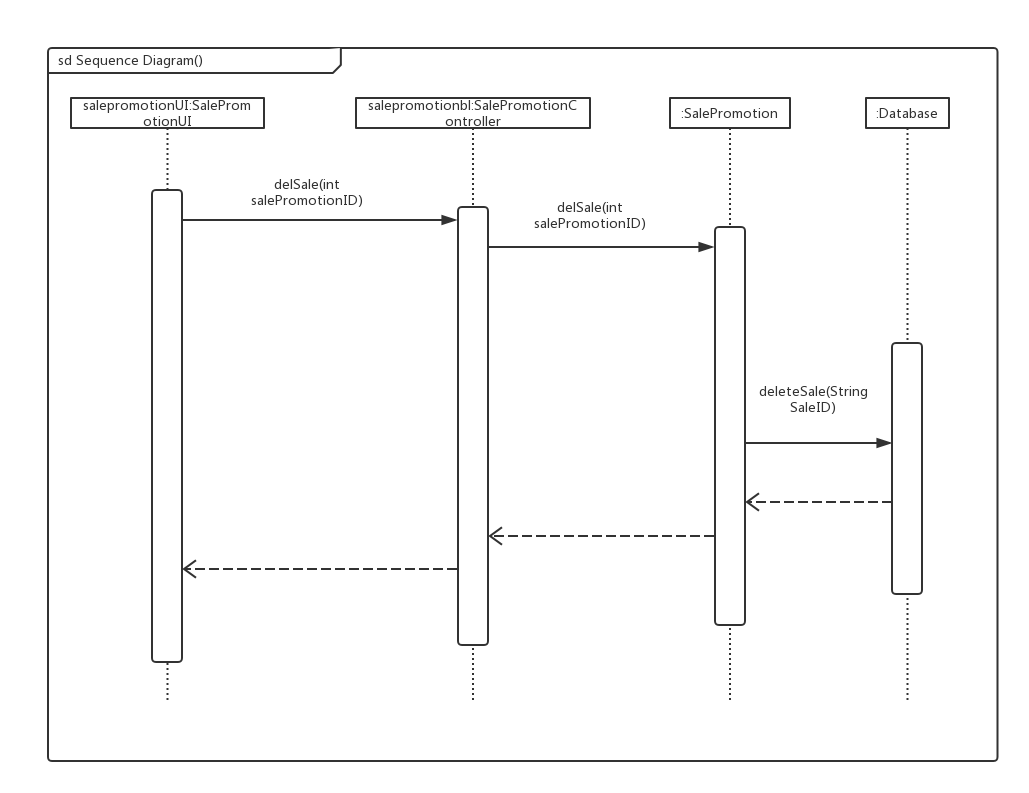


图5-4 删除促销策略

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。

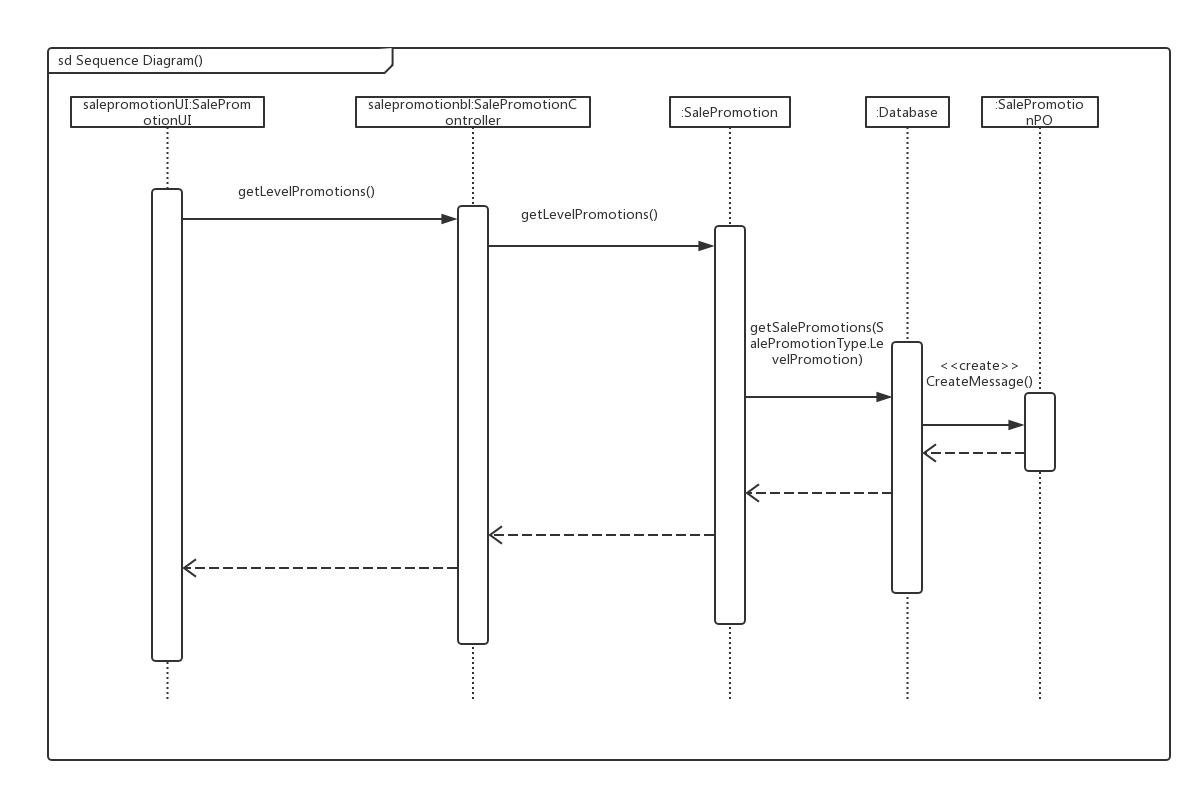


图5-5 查看等级制度

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。

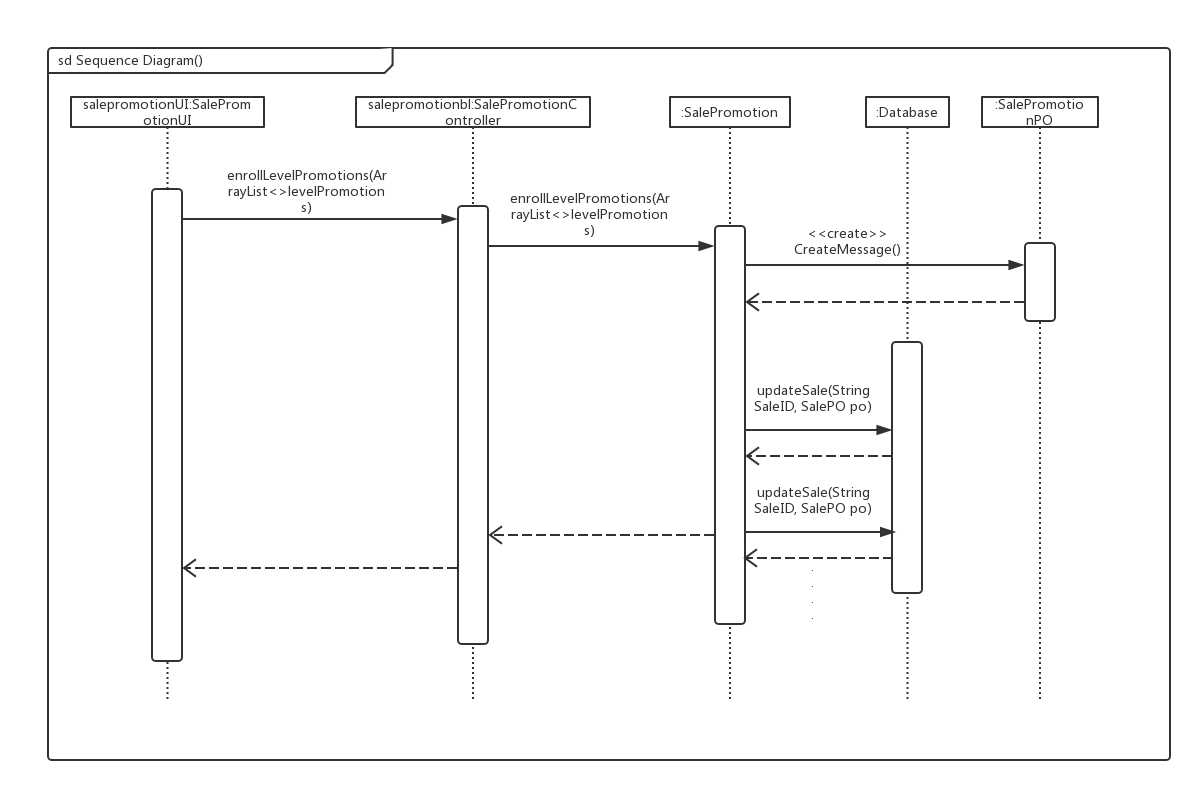
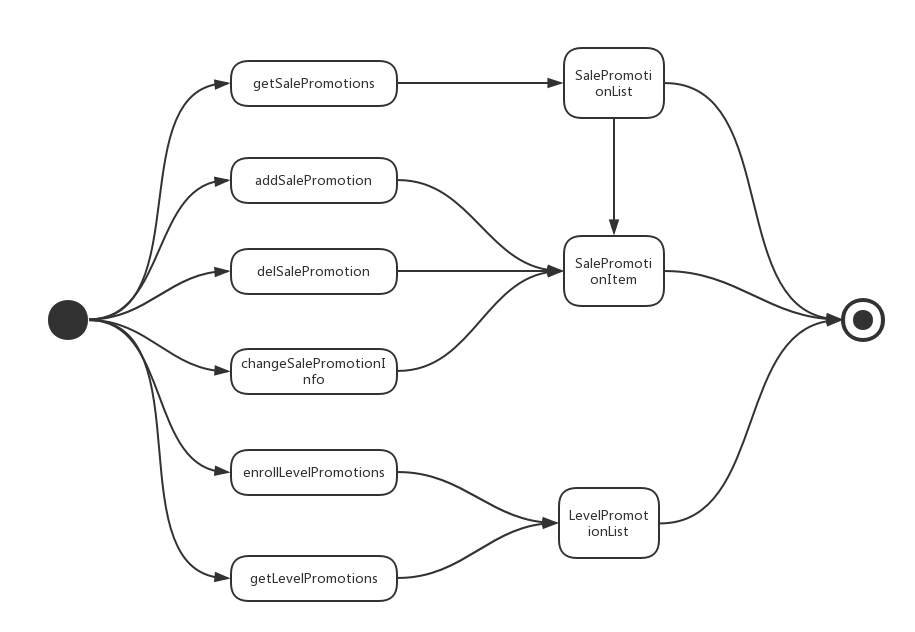


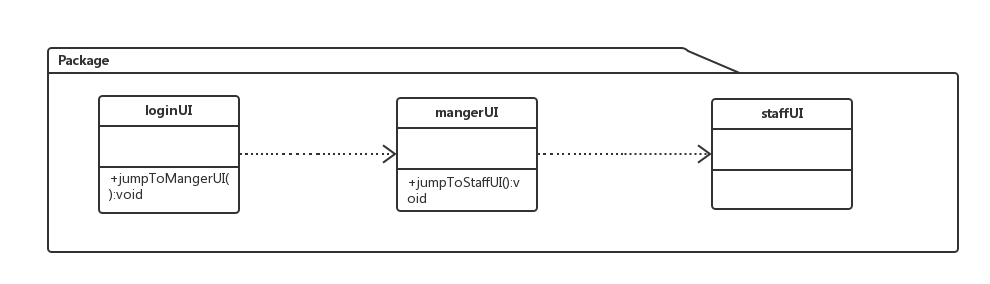
图5-6 管理等级制度

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。

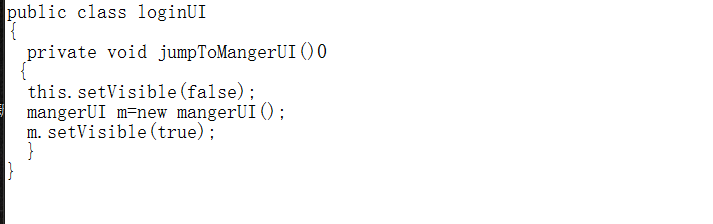


**2.界面跳转**

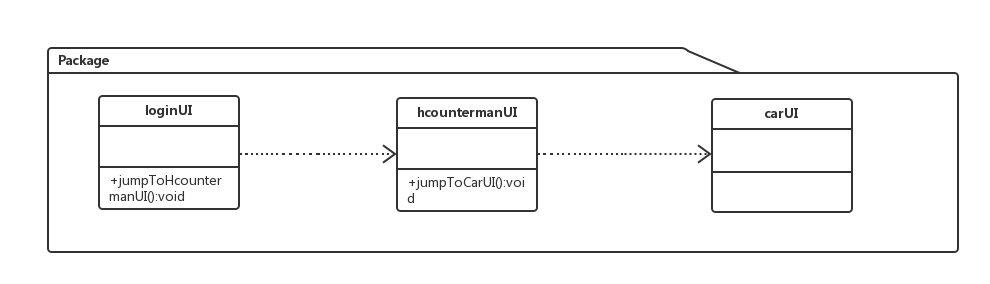
采用分散式控制风格如图



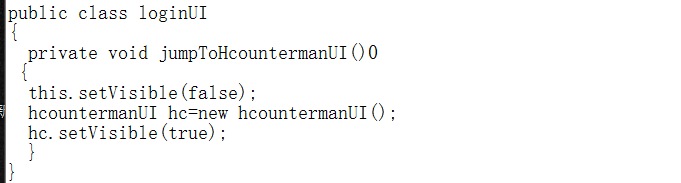
分散式控制风格类图



分散式控制风格代码



分散式控制风格类图



分散式控制风格代码

**3.依赖视角**

下图是客户端和服务器端各自的包的依赖关系

下图是客户端和服务器端各自的包之间的依赖关系。

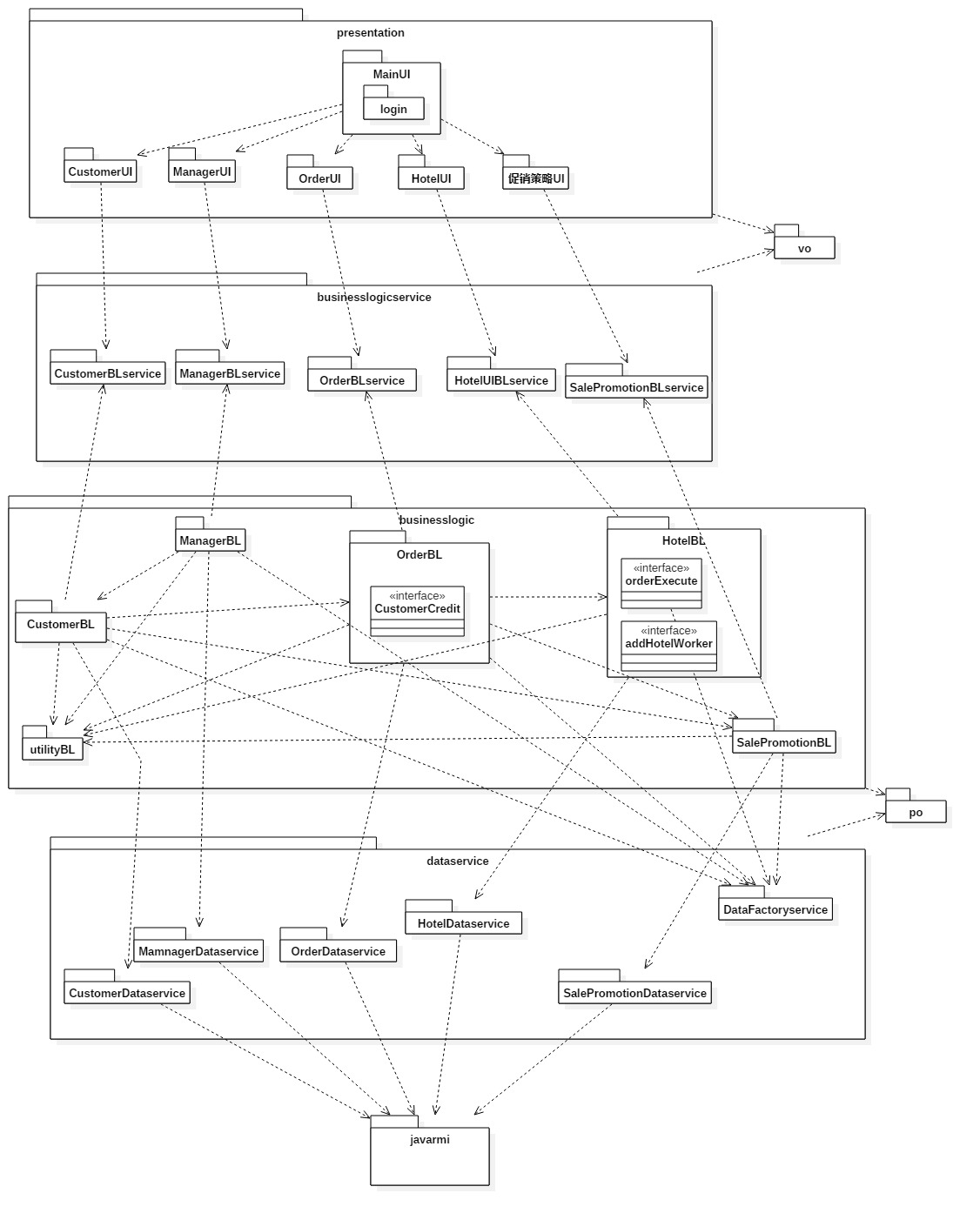


图1 酒店预订系统客户端开发包图

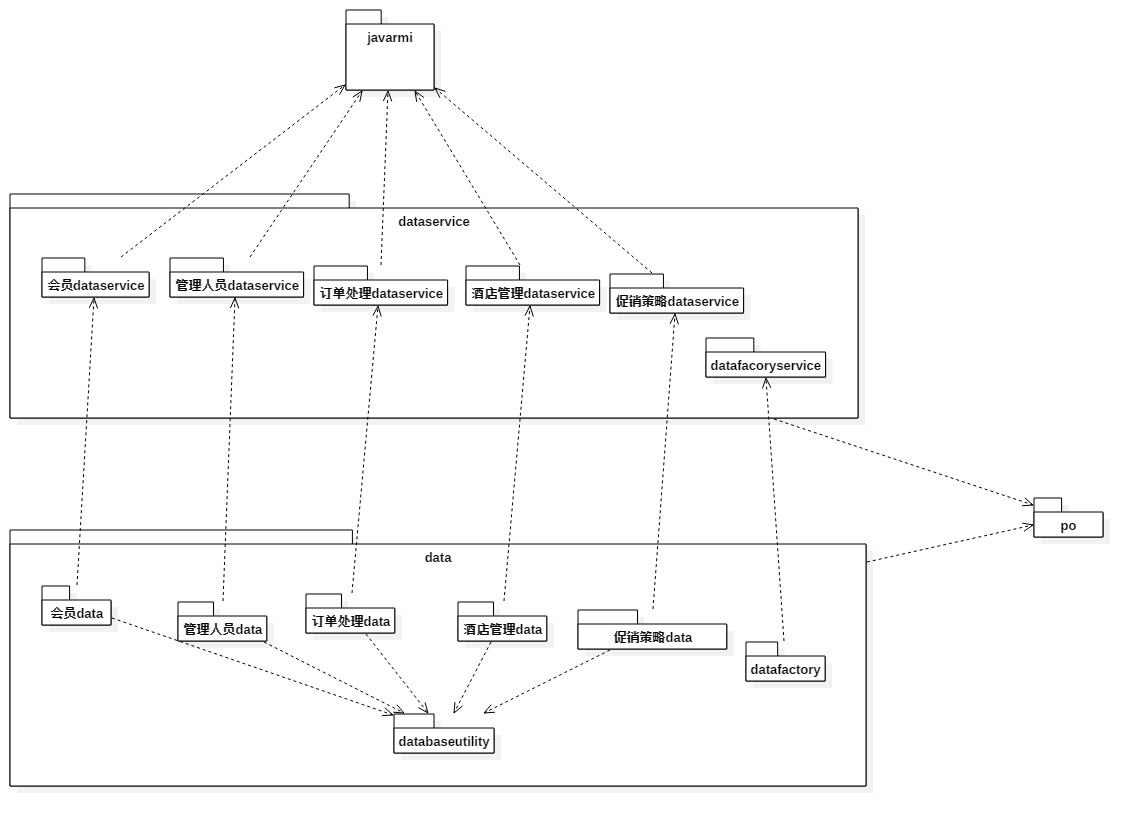


图2 酒店预订系统服务器端开发包图