**4.1.4 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系统结构描述文档。

orderbl模块承担实现订单的功能，主要是订单的生成、撤销等。

（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。在展示层和业务逻辑层之间添加businesslogicservice.orderblservice.OrderBLService接口。在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，添加OrderController，这样OrderController会将对生成订单、撤销订单、浏览客户订单、浏览酒店订单、执行订单、查看未执行订单、撤销异常订单的业务逻辑处理委托给Order对象。OrderPO是作为订单的持久性对象被添加到设计模型中去的。OrderList是OrderLineItem的容器类。UserInfo、PromotionInfo、UpdateCredit、UpdateRoom是根据依赖倒置原则，为了消除循环依赖而产生的接口。

orderbl模块的设计如图4-1-4-1所示。

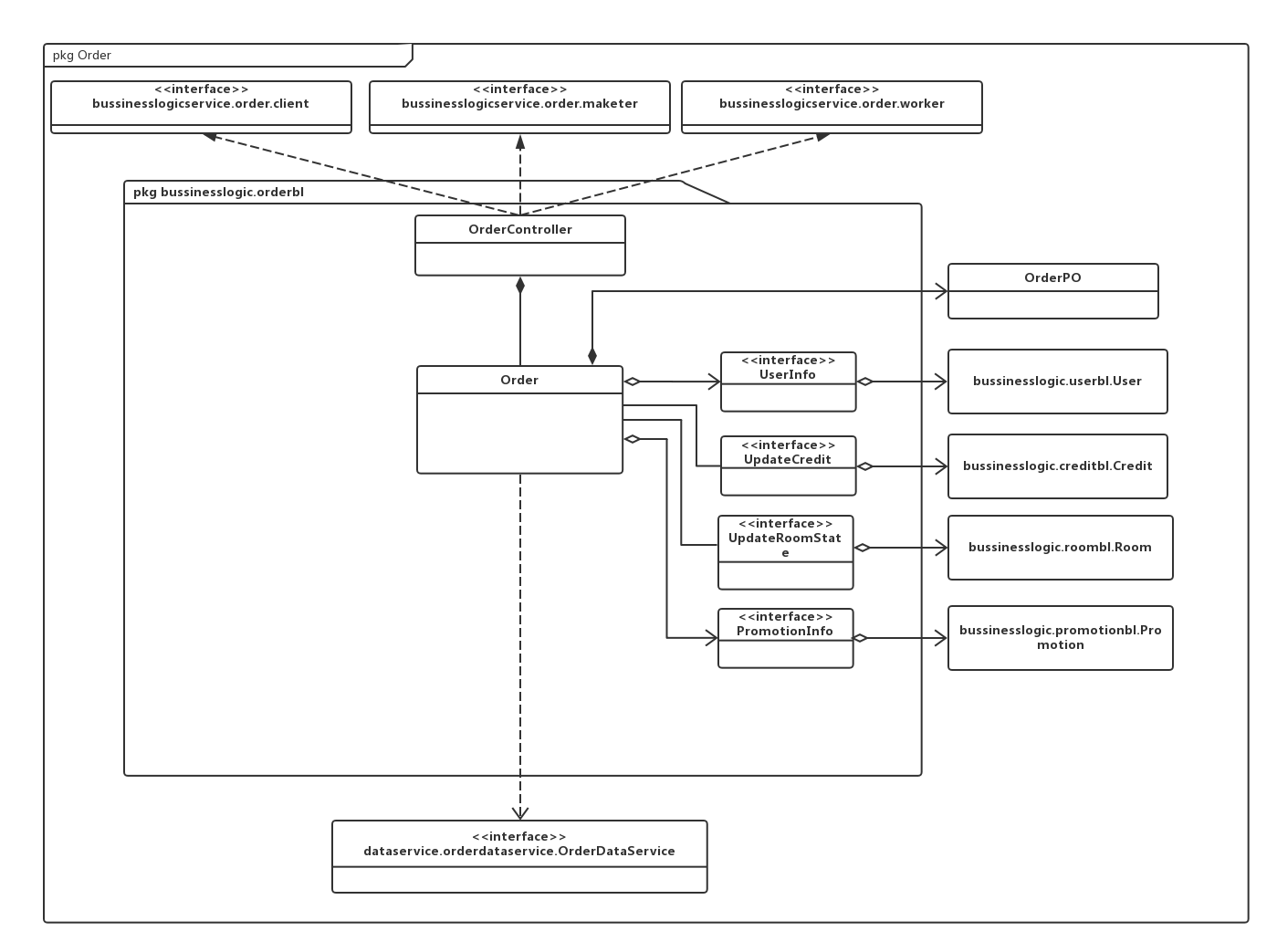


图4-1-4-1 orderbl模块各个类的设计

orderbl模块各个类的职责如表4-1-4-1所示。

**表4-1-4-1 orderbl模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| OrderController | 负责实现订单界面所需要的服务 |
| Order | 订单的领域模型对象，拥有订单所持有的订单ID、订单时间等信息，可以帮助完成订单界面所需要的服务 |

（3）模块内部类的接口规范

OrderController和Order的接口规范如表4-1-4-2和表4-1-4-3所示。

**表4-1-4-2 OrderController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| OrderController.makeOrder | 语法 | | public ResultMessage  makeOrder(OrderVo vo) |
| 前置条件 | | 用户确认订单信息无误 |
| 后置条件 | | 调用Order领域对象的makeOrder方法 |
| OrderController.cancelOrder | 语法 | | public ResultMessage  cancelOrder(String orderId) |
| 前置条件 | | 用户或网站营销人员已登陆并确认撤销订单 |
| 后置条件 | | 调用Order领域对象的cancelOrder方法 |
| OrderController.queryOrderById | 语法 | | public OrderVo  queryOrderById(String orderId) |
| 前置条件 | | 订单号输入格式正确 |
| 后置条件 | | 调用Order领域对象的queryOrderById方法 |
| OrderController.queryOrderByHotel | 语法 | | public ArrayList<OrderVo>  queryOrder(String hotelId,String userId) |
| 前置条件 | | 用户已选择酒店 |
| 后置条件 | | 调用Order领域对象的queryOrder方法 |
| OrderController.queryRoomOrder | 语法 | | public ArrayList<OrderVo>  queryRoomOrder(String hotelId,  String roomId) |
| 前置条件 | | 酒店工作人员已选择房间 |
| 后置条件 | | 调用Order领域对象的queryRoomOrder方法 |
| OrderController.queryUserOrder | 语法 | | public ArrayList<OrderVo>  queryUserOrder(String userId) |
| 前置条件 | | 用户选择查询订单 |
| 后置条件 | | 调用Order领域对象的queryUserOrder方法 |
| OrderController.queryUnexcutedOrder | 语法 | | public ArrayList<OrderVo>  queryUnexcutedOrder() |
| 前置条件 | | 网站营销人员选择查询未执行订单 |
| 后置条件 | | 调用Order领域对象的queryUnexcutedOrder方法 |
| OrderController.queryHotelOrder | 语法 | | public ArrayList<OrderVo>  queryHotelOrder(String hoteIId) |
| 前置条件 | | 酒店工作人员选择查询酒店订单 |
| 后置条件 | | 调用Order领域对象的queryHotelOrder方法 |
| OrderController.excuteOrder | 语法 | | public ResultMessage  excuteOrder(String orderId) |
| 前置条件 | | 酒店工作人员确认执行订单 |
| 后置条件 | | 调用Order领域对象的excuteOrder方法 |
| OrderController.putUpOrder | 语法 | | public ResultMessage  putUpOrder(String orderId) |
| 前置条件 | | 酒店工作人员确认补录订单 |
| 后置条件 | | 调用Order领域对象的putUpOrder方法 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| Order.makeOrder | | 生成订单 | |
| Order.cancelOrder | | 撤销订单 | |
| Order.queryOrderById | | 通过订单ID查找订单 | |
| Order.queryOrderByHotel | | 查找用户在酒店订单 | |
| Order.queryRoomOrder | | 查找房间订单 | |
| Order.queryUserOrder | | 查找用户订单 | |
| Order.queryUnexcutedOrder | | 查找位置性能订单 | |
| Order.queryHotelOrder | | 查找酒店订单 | |
| Order.excuteOrder | | 执行订单 | |
| Order.putUpOrder | | 补录订单 | |

**表4-1-4-3 Order的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Order.makeOrder | 语法 | | public ResultMessage  makeOrder(OrderVo vo) |
| 前置条件 | | 启动生成订单回合 |
| 后置条件 | | 系统自动给该订单生成订单编号并持久化保留该订单信息 |
| Order.cancelOrder | 语法 | | public ResultMessage  cancelOrder(String orderId) |
| 前置条件 | | 启动撤销订单回合 |
| 后置条件 | | 系统返回是否撤销成功 |
| Order.queryOrderById | 语法 | | public OrderVo  queryOrderById(String orderId) |
| 前置条件 | | 启动查找订单回合 |
| 后置条件 | | 系统查找是否存在该订单，根据输入的订单id返回订单信息 |
| Order.queryOrderByHotel | 语法 | | public ArrayList<OrderVo>  queryOrder(String hotelId,String userId) |
| 前置条件 | | 启动查找订单回合 |
| 后置条件 | | 系统查找是否存在该酒店，根据输入的酒店id返回订单信息 |
| Order.queryRoomOrder | 语法 | | public ArrayList<OrderVo>  queryRoomOrder(String hotelId,  String roomId) |
| 前置条件 | | 启动查找订单回合 |
| 后置条件 | | 系统查找是否存在该房间，根据输入的订单id返回该房间的预订订单列表 |
| Order.queryUserOrder | 语法 | | public ArrayList<OrderVo>  queryUserOrder(String userId) |
| 前置条件 | | 启动查找订单回合 |
| 后置条件 | | 系统查询是否存在该用户id，并根据id返回该用户的订单信息列表 |
| Order.queryUnexcutedOrder | 语法 | | public ArrayList<OrderVo>  queryUnexcutedOrder() |
| 前置条件 | | 启动查找订单回合 |
| 后置条件 | | 系统返回未执行的订单列表 |
| Order.queryHotelOrder | 语法 | | public ArrayList<OrderVo>  queryHotelOrder(String hoteIId) |
| 前置条件 | | 启动查找订单回合 |
| 后置条件 | | 系统查找是否存在该酒店，根据输入的酒店id返回订单信息 |
| Order.excuteOrder | 语法 | | public ResultMessage  excuteOrder(String orderId) |
| 前置条件 | | 启动执行订单回合 |
| 后置条件 | | 系统返回执行订单操作是否成功 |
| Order.putUpOrder | 语法 | | public ResultMessage  putUpOrder(String orderId) |
| 前置条件 | | 启动补录订单回合 |
| 后置条件 | | 系统返回订单补录是否成功 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| DataFactory.getOrderData() | | 得到Order数据的服务的引用 | |
| OrderDataService.insert(OrderPO po) | | 插入OrderPO单一持久化对象 | |
| OrderDataService.update(OrderPO po) | | 更新OrderPO单一持久化对象 | |
| OrderDataService.findById (String orderId) | | 得到该orderid的OrderPO持久化对象 | |
| OrderDataService.findByUser(String userId) | | 得到user的所有的OrderPO持久化对象 | |
| OrderDataService.findUOByHotel(String hotelId, String userId) | | 得到user在该酒店下的所有的OrderPO持久化对象 | |
| OrderDataService.findHotelOrder(String hotelId) | | 得到worker所在酒店下的所有的OrderPO持久化对象 | |
| OrderDataService.findByRoom(String hotelId,String roomId) | | 得到worker所在酒店下的该Room的所有OrderPO持久化对象 | |
| OrderDataService.findUnexcutedOrder() | | 得到当天所有未执行订单的OrderPO持久化对象 | |
| OrderDataService.putUpOrder(String orderId) | | 补录该订单 | |
| User.getUserInfo() | | 得到当前用户信息 | |
| Promotion.getPromotion() | | 得到当前促销策略 | |
| Credit.updateCredit(int value) | | 更新Credit记录 | |
| Room.checkIn(String roomId) | | 更新房间入住状态 | |
| Room.leave(String roomId) | | 更新房间离开状态 | |

（4）业务逻辑层的动态模型

在订单业务中，分为生成订单、撤销订单等业务。

图4-1-4-2表明了生成订单业务逻辑处理的相关对象之间的协作。

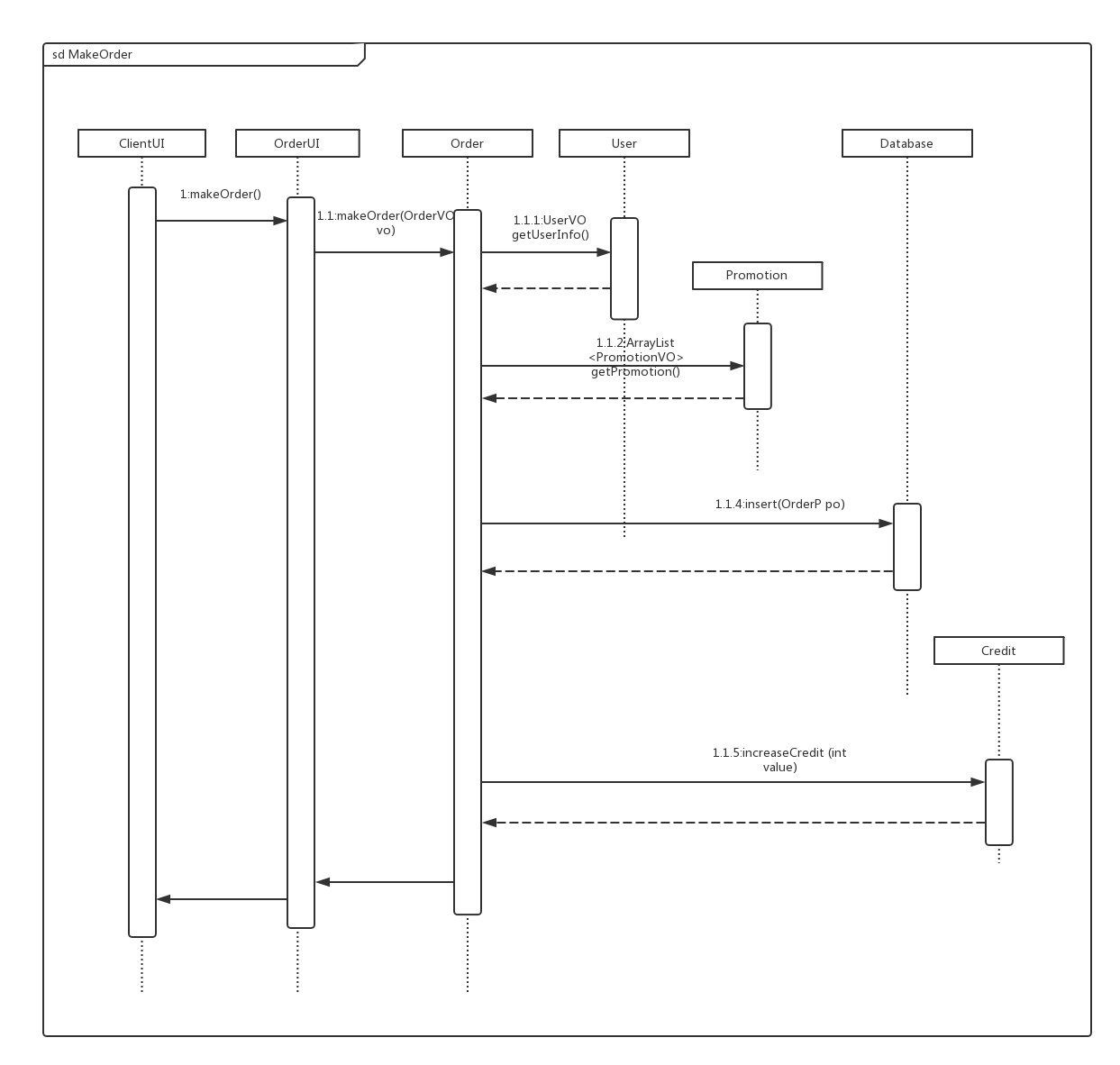


图4-1-4-2 生成订单的顺序图

图4-1-4-3表明了撤销订单业务逻辑处理的相关对象之间的协作。

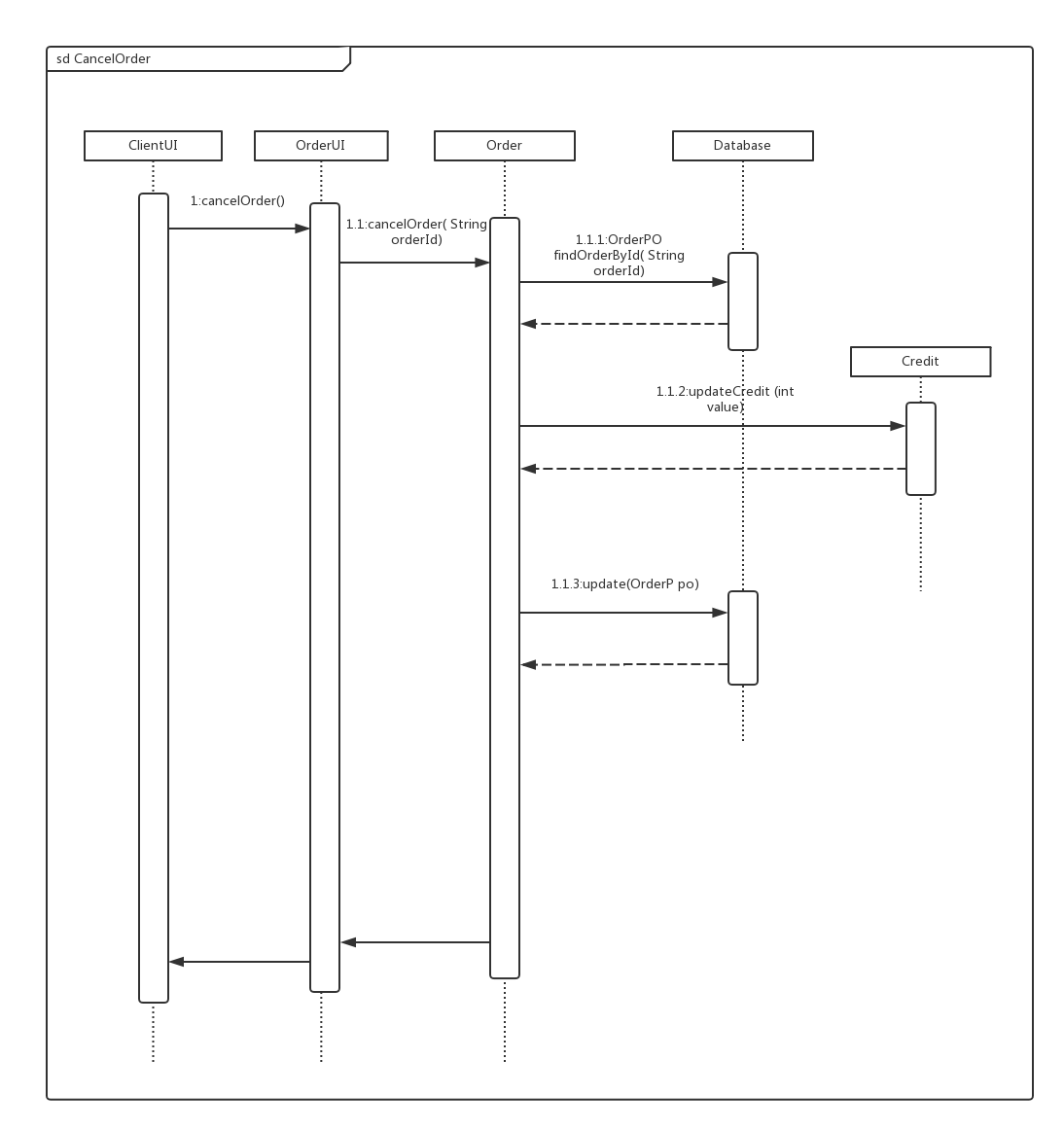


图4-1-4-3 撤销订单的顺序图

图4-1-4-4表明了执行订单业务逻辑处理的相关对象之间的协作。

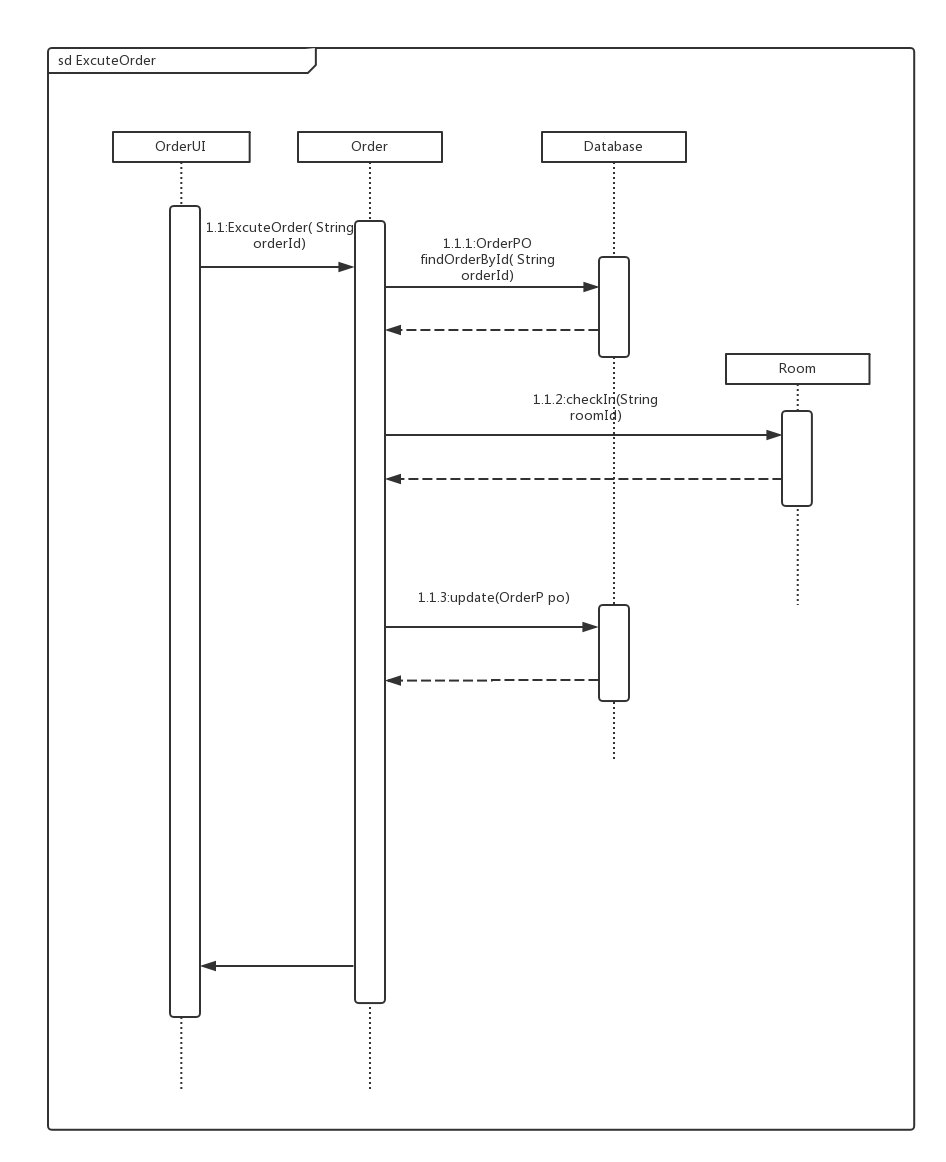


图4-1-4-4 执行订单的顺序图

图4-1-4-5表明了查询订单业务逻辑处理的相关对象之间的协作。

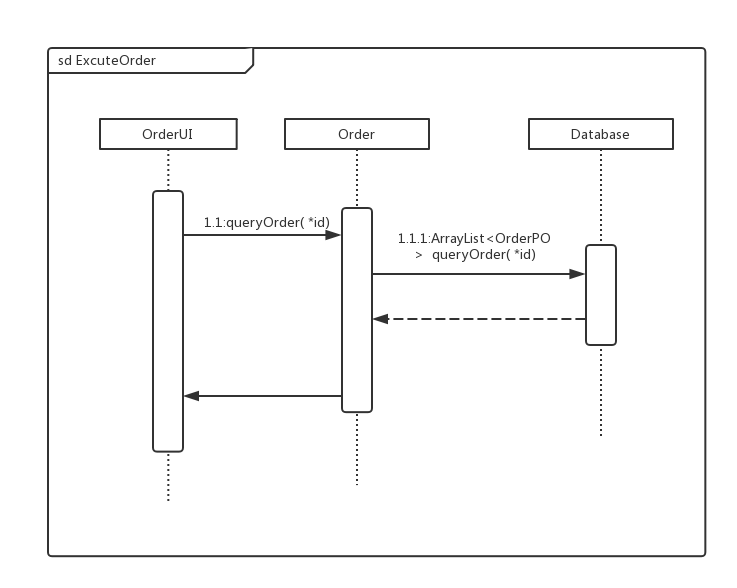


图4-1-4-5 查询订单的顺序图

图4-1-4-6 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。

PutUpOrder被UI调用进入exception状态的order，之后通过putUpOrder，order进入已执行状态，之后通过endOrder进入结束状态；如果MakeOrder方法被UI调用，Order进入Make状态，之后通过makeOrder方法，进入Unexcute状态，在此状态下，如果通过excute，order则进入Excute状态，如果通过cancel，order进入Canceled状态，之后通过endOrder进入结束状态

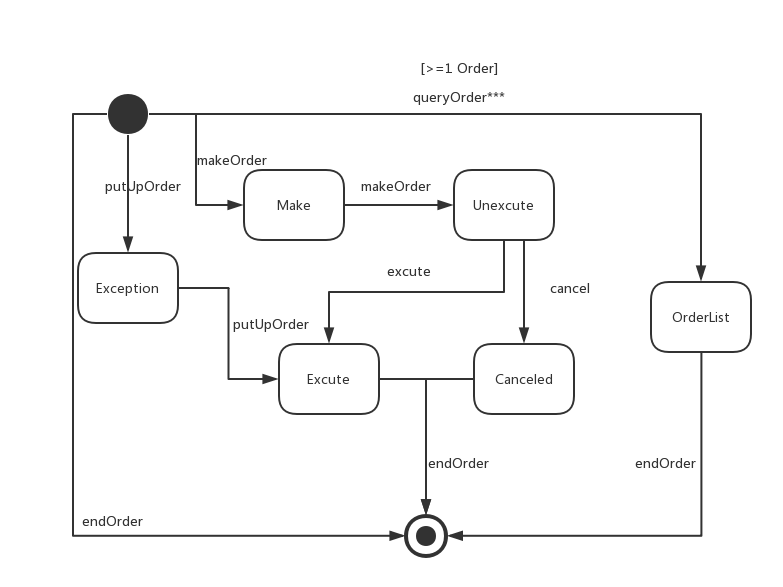


图4-1-4-6 Order对象状态图