分布式温控系统

软件概要设计说明书

面向对象方法

**单位名称: 14班C组**

**组长：胡旭强**

**组员：陈子迪、董升华、袁振宇、 孙嘉倪、石羽嘉**

**创建日期：2017年5月22日**

**文件修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修订人** | **修订日期** | **修订内容** |
| 1.0 | 全体 | 2017-03-24 | 确定细化需求，由需求画出架构示意图，分工进行细节需求规格分析。 |
| 胡旭强 | 2017-03-25 | 用户相关的用例图、用例描述、系统顺序图。 |
| 陈子迪 | 2017-03-25 | 空调管理员相关的用例图、用例描述、系统顺序图。 |
| 董升华 | 2017-03-25 | 写概述，定制文档规则和函数命名规则。协调各部分文档风格统一，整合各部分文档。 |
| 袁振宇 | 2017-03-25 | 项目背景思考和编写。 |
| 孙嘉倪 | 2017-03-25 | 前台管理员相关的用例图、用例描述、系统顺序图。 |
| 石羽嘉 | 2017-03-25 | 酒店管理员相关的用例图、用例描述、系统顺序图。 |
| 全体 | 2017-03-26 | 全体阅读总文档，统一细节用词，修改用词不严谨部分。 |
| 全体 | 2017-03-27 | 修改过耦合部分，为2.0版本修改留出余地。 |
| 2.0 | 全体 | 2017-4-9 | 确定分工 |
| 董升华 | 2017-4-9 | 业务知识背景，整理与排版 |
| 袁振宇 | 2017-4-9 | 概念属性方法表 |
| 其他成员 | 2017-4-9 | 各自部分的操作契约与部分业务流程图 |
| 全体成员 | 2017-4-17 | 进行格式化修改 |
| 3.0 | 全体成员 | 2017-5-12 | 攥写动态结构部分 |
| 4.0 | 全体成员 | 2017-5-20 | 攥写静态结构部分 |

**目录**

[1. 概述 6](#_Toc483428758)

[1.1．文档约定 6](#_Toc483428759)

[1.2．预期的读者和阅读建议 7](#_Toc483428760)

[1.3．产品的范围 7](#_Toc483428761)

[1.4．参考文献 8](#_Toc483428762)

[2. 项目背景描述 8](#_Toc483428763)

[2.1. 运行环境： 8](#_Toc483428764)

[2.2. 设计和实现上的限制： 8](#_Toc483428765)

[2.3. 假定和依赖： 8](#_Toc483428766)

[3.系统的技术架构说明 9](#_Toc483428767)

[4.系统的动态结构设计说明 10](#_Toc483428768)

[4.1 空调管理员模块 10](#_Toc483428769)

[4.1.1 监视从机状态 10](#_Toc483428770)

[4.1.1.1 系统事件列表 10](#_Toc483428771)

[4.1.1.2系统事件\_1：powerOn 11](#_Toc483428772)

[4.1.1.3 系统事件\_2:powerOff 12](#_Toc483428773)

[4.1.1.4 系统事件\_3 checkTem() 13](#_Toc483428774)

[4.1.2 主机设置模块 14](#_Toc483428775)

[4.1.2.1 系统事件列表 14](#_Toc483428776)

[4.1.2.2系统事件\_1：powerOn 15](#_Toc483428777)

[4.1.2.3 系统事件\_2:powerOff 16](#_Toc483428778)

[4.1.2.4 系统事件\_3：setMain 17](#_Toc483428779)

[4.1.3 主机调度 18](#_Toc483428780)

[4.1.3.1 系统事件列表 18](#_Toc483428781)

[4.1.3.2 系统事件\_1：handleCost 19](#_Toc483428782)

[4.1.3.3系统事件\_2: handleTemperature 20](#_Toc483428783)

[4.1.3.4系统事件\_3: handleWind 21](#_Toc483428784)

[4.2 用户模块 22](#_Toc483428785)

[4.2.1 基本用例（用户使用空调） 22](#_Toc483428786)

[4.2.1.1 系统事件列表 22](#_Toc483428787)

[4.2.1.2 系统事件1：设置目标温度 23](#_Toc483428788)

[4.2.1.3 系统事件（设置风速） 24](#_Toc483428789)

[4.2.1.4 系统事件3（显示花费） 25](#_Toc483428790)

[4.2.1.5 系统事件4（查看详单） 26](#_Toc483428791)

[4.3 统计模块 27](#_Toc483428792)

[4.3.1 基本用例（出具详单） 27](#_Toc483428793)

[4.3.1.1系统事件列表 27](#_Toc483428794)

[4.3.1.2系统事件\_1: void get\_detail\_bill 28](#_Toc483428795)

[4.3.2 基本用例（出具账单） 29](#_Toc483428796)

[4.3.2.1系统事件列表 29](#_Toc483428797)

[4.3.2.2系统事件\_1: void print\_bill 29](#_Toc483428798)

[4.4 酒店管理模块 30](#_Toc483428799)

[4.4.1 基本用例(分析消费情况) 30](#_Toc483428800)

[4.4.1.1系统事件列表 30](#_Toc483428801)

[4.4.1.2系统事件\_1: get\_report 31](#_Toc483428802)

[4.5通信模块 32](#_Toc483428803)

[4.5.1基本用例（主从机通信） 32](#_Toc483428804)

[4.5.1.1系统事件列表 32](#_Toc483428805)

[4.5.1.2系统事件\_1: send\_AimTemperature 33](#_Toc483428806)

[4.5.1.3系统事件\_2: send\_wind 34](#_Toc483428807)

[4.5.1.4系统事件\_3: get\_price 35](#_Toc483428808)

[4.5.1.7系统事件\_6: monitor\_state 36](#_Toc483428809)

[4.5.1.8系统事件\_7: initSlave 37](#_Toc483428810)

[4.6 持久化层设计 38](#_Toc483428811)

[4.6.1 报表信息持久化过程 38](#_Toc483428812)

[4.6.2 详单信息持久化过程 39](#_Toc483428813)

[4.7 StartUp设计 39](#_Toc483428814)

[5 系统静态结构下 40](#_Toc483428815)

[5.1 系统整体静态结构 40](#_Toc483428816)

[5.2 模块化静态结构 41](#_Toc483428817)

[5.2.1监视从机状态 41](#_Toc483428818)

[5.2.1.1类列表 41](#_Toc483428819)

[5.2.1.2 类图 42](#_Toc483428820)

[5.2.2 主机设置模块 42](#_Toc483428821)

[5.2..2.1类列表 42](#_Toc483428822)

[5.2.2.2类图 44](#_Toc483428823)

[5.2.3主机调度 44](#_Toc483428824)

[5.2.3.1类列表 44](#_Toc483428825)

[5.2.3.2类图 46](#_Toc483428826)

[5.2.4温控设置： 46](#_Toc483428827)

[5.2.4.1类列表： 46](#_Toc483428828)

[5.2.4.2 类图： 47](#_Toc483428829)

[5.2.5实时显示花费： 48](#_Toc483428830)

[5.2.5.1类列表 48](#_Toc483428831)

[5.2.5.2类图 48](#_Toc483428832)

[5.2.6查看报表 49](#_Toc483428833)

[5.2.6.1类列表 49](#_Toc483428834)

[5.2.6.2类图 49](#_Toc483428835)

[5.2.7出具账单 50](#_Toc483428836)

[5.2.7.1类列表 50](#_Toc483428837)

[5.2.7.2 类图 51](#_Toc483428838)

[5.2.8出具详单 51](#_Toc483428839)

[5.2.8.1类列表 51](#_Toc483428840)

[5.2.8.2 类图 52](#_Toc483428841)

[5.2.9分析报表 53](#_Toc483428842)

[5.2.9.1类列表 53](#_Toc483428843)

[5.2.9.2 类图 54](#_Toc483428844)

[5.2.10主机与从机间通信 54](#_Toc483428845)

[5.2.10.1类列表 54](#_Toc483428846)

[5.2.10.2类图 55](#_Toc483428847)

# 1. 概述

某快捷廉价酒店的自助计费式分布式温控系统核心由中央空调（主机）和房间空调（从机）两部分组成。客户可以在房间空调处进行设定温度和调节风速，同时在控制面板处了解所需支付的金额。这一切操作通过向中央空调发送温控请求来实现。中央空调有负载均衡的能力，可以保证所有房间的温控请求。几个管理员可以分别监测各个房间的状态，调整空调状态，和查看报表。除此之外，系统针对每个房间需要给出空调使用的详单方便客户退房时进行结账，以及酒店空调使用的各式统计报表。

## 1.1．文档约定

文档的描述内容是分布式温控系统的需求规格说明书；

文档中涉及的各项功能来源于系统的需求分析规格说明书；

文档采用基于UML建模语言的面向对象建模方式对对系统的功能进行结构设计；

功能的小节以需求分析规格说明书中的基本用例进行排序；

系统的结构设计由动态结构和静态结构组成，且静态结构通过动态结构汇总得到；

系统的动态结构组成：

每个基本用例及其子用例，对应有一张系统顺序图；

用例实现的设计就是说明系统顺序图中的每个系统事件，进入软件系统后由哪个软件对象接收，并由哪些其他的软件对象协同工作，按照需求分析规格说明的要求返回规定的结果给使用者；

每个系统事件的交互图使用UML的顺序图表示某一时刻软件对象的交互实例；

系统顺序图中有过多少个系统事件（比如，5个），该用例实现的交互图就有多少个（5个顺序图）

每个系统事件有明确的方法和参数定义。

系统的静态结构组成：

系统的静态结构主要由软件类及其之间的关系表示，即类图；软件类之间的关系主要使用定向关联（表示一个软件类的方法调用另一个软件类的方法），在明确两个类之间有相互调用的情况下使用双向关联

系统结构过于复杂的情况下，通过包图的方式描述系统的宏观静态结构；包图主要使用依赖关系表示包和包之间的调用关系。

文档中所有的UML图形均以 IBM RSA 8.0.3版本的建模工具进行绘制；

## 1.2．预期的读者和阅读建议

|  |  |  |
| --- | --- | --- |
| 编号 | **预期读者** | **阅读建议** |
| **1** | 房客 | 确认文档中给出的功能需求描述，了解如何使用本程序，能够熟练地进行基本操作，完成对从机空调的调制等操作。 |
| **2** | 管理员 | 确认文档中给出的功能需求描述，学会行使管理员权限，能够合理地调配、管理旅客用户组，对从控机空调进行初始化等操作。除此之外，还需通过数据分析来查知客户的消费倾向，从而改造出更合理的经营模式。 |
| **3** | 购买方 | 确认文档中给出的功能需求描述，了解本程序的适用范围及完整价值，进行理智且正确的消费、购买。 |
| **4** | 开发方 | 熟悉并掌握项目的各项功能要求，在客户需求的基础上提出自己的理念，随时进行功能扩展及程序的维护。 |

## 1.3．产品的范围

本系统分为中央空调(主机)、房间空调(从机)两部分，封装在两个不同的程序里。使用前将主机程序放在管理员主机上，而从机程序则放在每个房间内由用户操作的电脑上。

每次系统开机时，首先打开主机程序，而后依次打开从机程序，与主机进行连接并交互。

两个程序只能通过局域网来传送协议信息，从而进行互动及干预，不能直接由某个程序去实现另一个程序的功能。

1. 主机系统：供管理员使用，功能包括空调参数配置、用户组管理、监控用户状态、消费详单查询。负责控制主机及相应的电脑数据库，属于类似于后台管理与维护状态监控的模块。
2. 从机系统：供房客使用，功能包括温度调节、风力调节、费用显示，由面向用户的操作面板实现，且只能由用户控制，管理员不能直接干预。

## 1.4．参考文献

【1】软件工程模型与方法，肖丁 修佳鹏

【2】uml.org.

# 2. 项目背景描述

## 2.1. 运行环境：

子系统的运行环境要求：

|  |  |  |
| --- | --- | --- |
| 编号 | **名称** | **运行环境** |
| 1 | 主机 | Windows8.1及以上版本的系统 |
| 2 | 数据库 | SQL server 2012 |
| 3 | 从机 | Windows8.1及以上版本的系统 |

## 2.2. 设计和实现上的限制：

各子系统运行和开发所需要的技术条件和限制说明

|  |  |  |
| --- | --- | --- |
| 限制因素 | 限制说明 | 备注 |
| 必须采用的技术、工具、编程语言、数据库等 | C/S混合结构，采用SQL server数据库，qt |  |
| 不能使用的技术、工具、编程语言、数据库等 | 无特殊限制。 |  |
| 企业策略、政策法规、业界标准 | 必须遵守中华人民共和国的相关法律法规，以及项目小组的基本意识准则，同时不能拿低于行业标准的残次品应付了事。 |  |
| 硬件限制 | 无特殊限制。 |  |
| 性能限制 | 要求.net 3.0及以上版本。正常windows7以上平台的PC均可，在xp上可能会无法正常运行。 |  |

## 2.3. 假定和依赖：

定义各子系统运行和开发中可能的假定条件和依赖因素

表格 0‑1假设因素

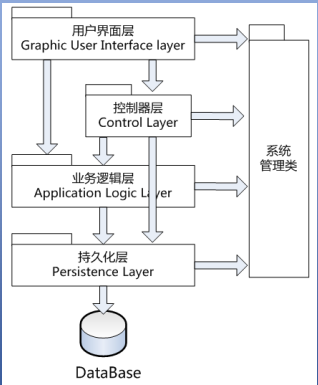
|  |  |  |
| --- | --- | --- |
| 编号 | 假设 | 备注 |
| 1 | 客户端操作系统windows8.1及以上版本 |  |
| 2 | 对现有业务、功能描述与实际情况基本相符合，对需求的变更不影响系统框架大的调整。 |  |

表格 0‑2依赖因素

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 依赖 | 依赖说明 | 备注 |
| 1 | SQL server | 业务数据存储在SQL server数据库中 |  |
| 2 | Windows8.1及以上版本 | 系统的开发及使用均在windows8.1及以上版本的系统上 |  |

# 3.系统的技术架构说明

系统选择的技术基础架构为分层结构，选择层次化的设计模型。本文档主要关注控制器层、业务逻辑层和持久化层的设计，如下图所示：



# 4.系统的动态结构设计说明

该系统的控制器层选择外观控制器，负责接收各种系统事件；根据系统顺序图、操作契约和领域模型，依次对各个系统事件进行用例实现，即动态结构设计。之后，进行了持久化层的设计，分别对报表信息持久化、详单信息持久化。最后设计了startUp功能来实现系统的初始化操作。

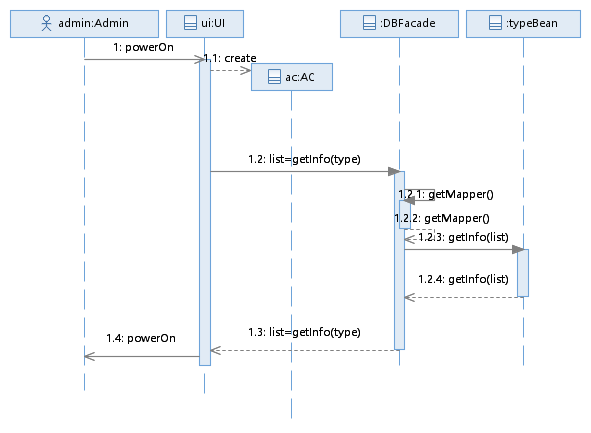
## 4.1 空调管理员模块

### 4.1.1 监视从机状态

#### 4.1.1.1 系统事件列表

|  |  |  |
| --- | --- | --- |
| 系统事件名称 | 用例 | 参数说明 |
| 1 double checkTem() | 监控温度（子用例） | 1、参数为空  2、返回值为房间当前温度 |
| 2 void powerOn() | 打开中央空调 | 1、参数为空  2、修改空调状态为1，即开机 |
| 3 void powerOff() | 关闭中央空调 | 1、参数为空  2、修改空调状态为0，即关机 |

#### 4.1.1.2系统事件\_1：powerOn



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Admin | 空调管理员 | 空调管理员，用于对中央空调进行管理，记录中央空调基本信息 |
| UI | 软件类 | 空调管理界面 |
| DBFacade | 软件类 | 在数据库操作时，用于维护领域类和负责其持 久化的对应 Mapper 类之间的对应关系 |
| AC | 概念类 | 中央空调类，用于产生中央空调对象 |
| typeBean | 软件类 | Type 是泛指，表示任意一个需要持久化的领域 类类名，TypeBean 专门负责该领域类的持久化 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| powerOn | null | null | 打开中央空调 |
| create | null | null | 创建中央空调实例 |
| getInfo | type：参数类型 | list:中央空调初始化设置参数 | 得到中央空调初始配置参数 |
| getMapper | null | typeBean:持久化类 | 得到该领域类对应的持久化的类 |

3、流程说明

①空调管理员执行开机

②创建中央空调对象

③将初始化设置从内存导入对象，类型为list<type>

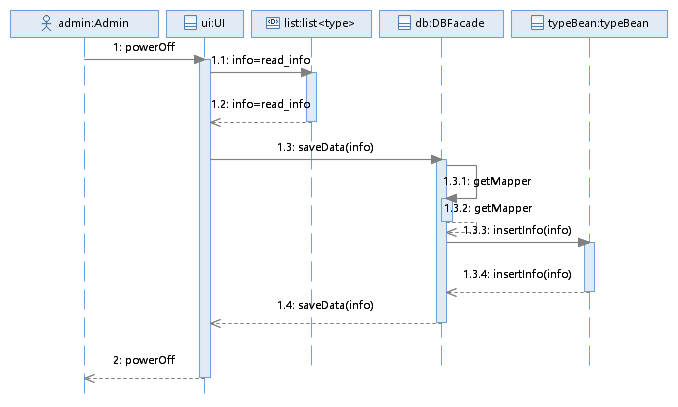
（包括：list<setInfo>//空调工作信息

list<customer>//用户信息

list<room>//房间信息）

④利用从list<setInfo>中获得的参数，为从机设置参数

#### 4.1.1.3 系统事件\_2:powerOff



1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Admin | 空调管理员 | 空调管理员，用于对中央空调进行管理，记录中央空调基本信息 |
| UI | 软件类 | 空调管理界面 |
| DBFacade | 软件类 | 在数据库操作时，用于维护领域类和负责其持 久化的对应 Mapper 类之间的对应关系 |
| AC | 概念类 | 中央空调类，用于产生中央空调对象 |
| typeBean | 软件类 | Type 是泛指，表示任意一个需要持久化的领域 类类名，TypeBean 专门负责该领域类的持久化 |
| list<type> | 数据类型 | Type 是泛指，与 TypeBean一一对应，持久化数据在软件执行过程中开辟内存空间，以链表形式存储 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| powerOff | null | null | 关闭中央空调 |
| readInfo | null | Info:空调当前状态 | 获取到中央空调当前状态 |
| saveData | Info：空调当前状态 | null | 保存中央空调目前状态 |
| getMapper | null | typeBean:持久化类 | 得到该领域类对应的持久化的类 |
| insertInfo | Info：空调当前状态 | null | 保存中央空调当前状态 |

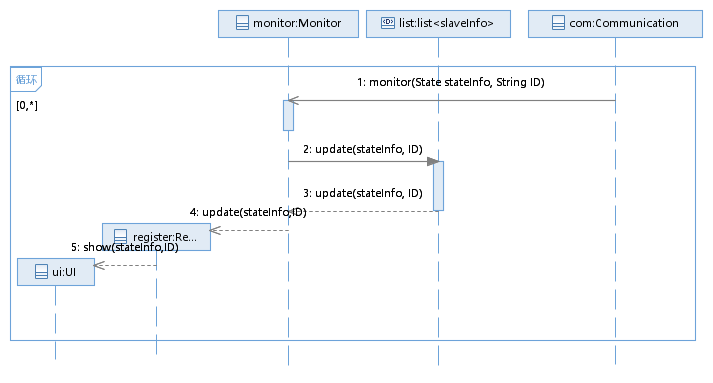
3、流程说明

①管理员点击关机按钮

②关机前将当前空调状态list<Info>存入数据库

③执行关机

#### 4.1.1.4 系统事件\_3 checkTem()



1、类说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 性质 | | 功能说明 | |
| Monitor | | 软件类 | | 检测从机当前状态变化，当从机状态变化时，会收到发来的更新包 |
| Register | | 软件类 | | 控制器，负责逻辑处理 |
| UI | | 软件类 | | 管理员接触到的用户界面 |
| AC | | 概念类 | | 中央空调类，用于产生中央空调对象 |
| list<slaveInfo> | | 数据类型 | | 记录从机当前状态 |
| Communication | | 软件类 | | 主控机部分，负责主从机之间的通信 | |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| monitor | State stateInfo:记录从控机当前状态数据的数据结构  String ID：房间号 | null | 监控得到从控机更新的当前状态 |
| update | State stateInfo:记录从控机当前状态数据的数据结构  String ID：房间号 | null | 根据房间号对相应房间进行状态更新 |
| show | State stateInfo:记录从控机当前状态数据的数据结构  String ID：房间号 | null | 将更新后的状态反馈到前端界面 |

3、流程说明

①从控机状态发生变化，向主控机发送更新信息

②通过Communication类与主机进行通信

②主机收到更新包，更新从机状态，并在前端显示

### 4.1.2 主机设置模块

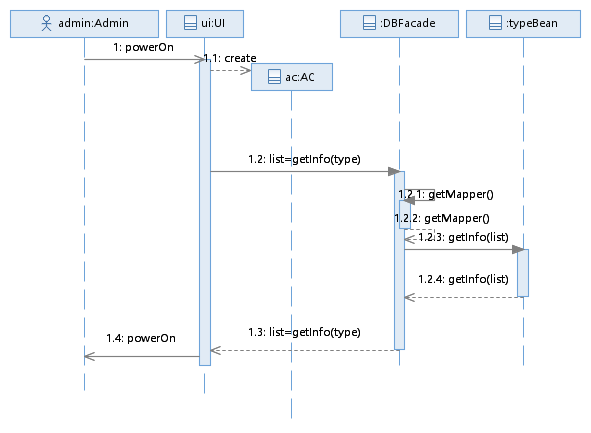
#### 4.1.2.1 系统事件列表

|  |  |  |
| --- | --- | --- |
| 系统事件名称 | 用例 | 参数说明 |
| 1 void powerOn() | 打开中央空调 | 1、参数为空  2、修改空调状态为1，即开机 |

|  |  |  |
| --- | --- | --- |
| 2 void powerOff() | 关闭中央空调 | 1、参数为空  2、修改空调状态为0，即关机 |

|  |  |  |
| --- | --- | --- |
| 3 void setMain(setList setInfo) | 设置主机初始化信息 | 1、参数为setList类型的数据结构，包括工作模式、最高最低温度、缺省工作温度、单位功率费用、缺省风速 |

#### 4.1.2.2系统事件\_1：powerOn



时序图说明

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Admin | 空调管理员 | 空调管理员，用于对中央空调进行管理，记录中央空调基本信息 |
| UI | 软件类 | 空调管理界面 |
| DBFacade | 软件类 | 在数据库操作时，用于维护领域类和负责其持 久化的对应 Mapper 类之间的对应关系 |
| AC | 概念类 | 中央空调类，用于产生中央空调对象 |
| typeBean | 软件类 | Type 是泛指，表示任意一个需要持久化的领域 类类名，TypeBean 专门负责该领域类的持久化 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| powerOn | null | null | 打开中央空调 |
| create | null | null | 创建中央空调实例 |
| getInfo | type：参数类型 | list:中央空调初始化设置参数 | 得到中央空调初始配置参数 |
| getMapper | null | typeBean:持久化类 | 得到该领域类对应的持久化的类 |

3、流程说明

①空调管理员执行开机

②创建中央空调对象

③将初始化设置从内存导入对象，类型为list<type>

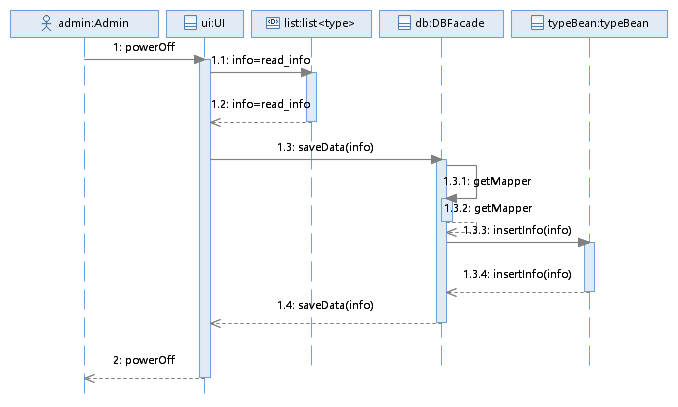
（包括：list<setInfo>//空调工作信息

list<customer>//用户信息

list<room>//房间信息）

④利用从list<setInfo>中获得的参数，为从机设置参数

#### 4.1.2.3 系统事件\_2:powerOff



1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Admin | 空调管理员 | 空调管理员，用于对中央空调进行管理，记录中央空调基本信息 |
| UI | 软件类 | 空调管理界面 |
| DBFacade | 软件类 | 在数据库操作时，用于维护领域类和负责其持 久化的对应 Mapper 类之间的对应关系 |
| AC | 概念类 | 中央空调类，用于产生中央空调对象 |
| typeBean | 软件类 | Type 是泛指，表示任意一个需要持久化的领域 类类名，TypeBean 专门负责该领域类的持久化 |
| list<type> | 数据类型 | Type 是泛指，与 TypeBean一一对应，持久化数据在软件执行过程中开辟内存空间，以链表形式存储 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| powerOff | null | null | 关闭中央空调 |
| readInfo | null | Info:空调当前状态 | 获取到中央空调当前状态 |
| saveData | Info：空调当前状态 | null | 保存中央空调目前状态 |
| getMapper | null | typeBean:持久化类 | 得到该领域类对应的持久化的类 |
| insertInfo | Info：空调当前状态 | null | 保存中央空调当前状态 |

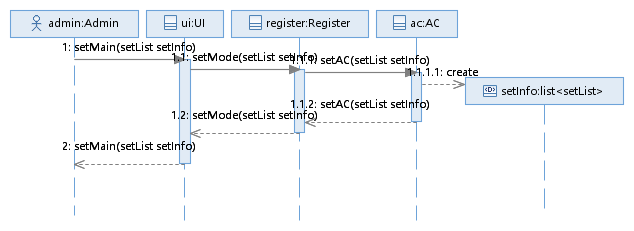
3、流程说明

①管理员点击关机按钮

②关机前将当前空调状态list<Info>存入数据库

③执行关机

#### 4.1.2.4 系统事件\_3：setMain

**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Admin | 空调管理员 | 空调管理员，用于对中央空调进行管理，记录中央空调基本信息 |
| UI | 软件类 | 空调管理界面 |
| DBFacade | 软件类 | 在数据库操作时，用于维护领域类和负责其持 久化的对应 Mapper 类之间的对应关系 |
| AC | 概念类 | 中央空调类，用于产生中央空调对象 |
| Register | 软件类 | 控制器，负责逻辑处理 |
| list<setList> | 数据类型 | 用于存储主机的设置参数 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| setMain | setList setInfo | null | 在用户交互界面设置主机的工作参数 |
| setMode | setList setInfo | null | 向控制部分传送主机的设置参数 |
| setAC | setList setInfo | null | 设置空调参数 |
| create | null | null | 创建空调参数表 |

3、流程说明

①在开机前，在用户交互界面UI弹出对话框，要求用户配置主机的工作参数

②UI设置完毕后将参数发送给控制器

③控制器将参数赋给空调类

④创建空调参数表并保存信息

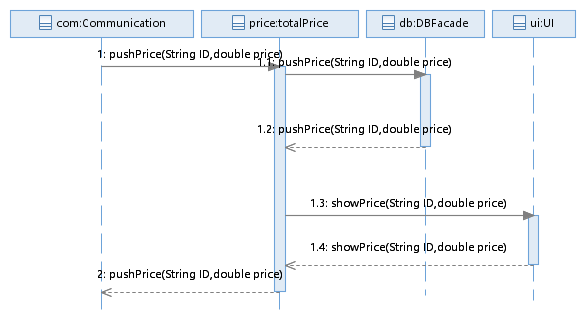
注：不允许主机运行后再次修改工作参数

### 4.1.3 主机调度

#### 4.1.3.1 系统事件列表

|  |  |  |
| --- | --- | --- |
| 系统事件名称 | 用例 | 参数说明 |
| 1 void handleCost(String ID, double cost) | **处理计费请求（子用例）** | **1、参数为房间号及费用**  **2、返回值为空** |
| 2 bool handleTemperature(int temp，int salveID) | **处理调温请求（子用例）** | 1. **temp：int类型，表示理想温度** 2. **slaveID：int 类型，发出请求的从控机号** 3. **若返回值为true，表示处理成功；为false，表示失败** |
| 3 bool handleWind(int wind，int slaveID) | **处理调风请求（子用例）** | 1. **temp：int类型，表示理想温度** 2. **slaveID：int 类型，发出请求的从控机号** 3. **若返回值为true，表示处理成功；为false，表示失败** |

#### 4.1.3.2 系统事件\_1：handleCost



1、类说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 性质 | | 功能说明 | |
| Communication | | 软件类 | | 主控机部分，负责主从机之间的通信 | |
| totalPrice | 软件类 | | 费用计算模块，负责传递当前某房间的费用和计算总费用 | |
| UI | 软件类 | | 空调管理界面 | |
| DBFacade | 软件类 | | 在数据库操作时，用于维护领域类和负责其持 久化的对应 Mapper 类之间的对应关系 | |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| pushPrice | String ID:房间号  double price：房间目前累计费用 | null | 传递房间号ID产生的空调费用 |
| showPrice | String ID:房间号  double price：房间目前累计费用 | null | 在前端显示某房间空调费用 |

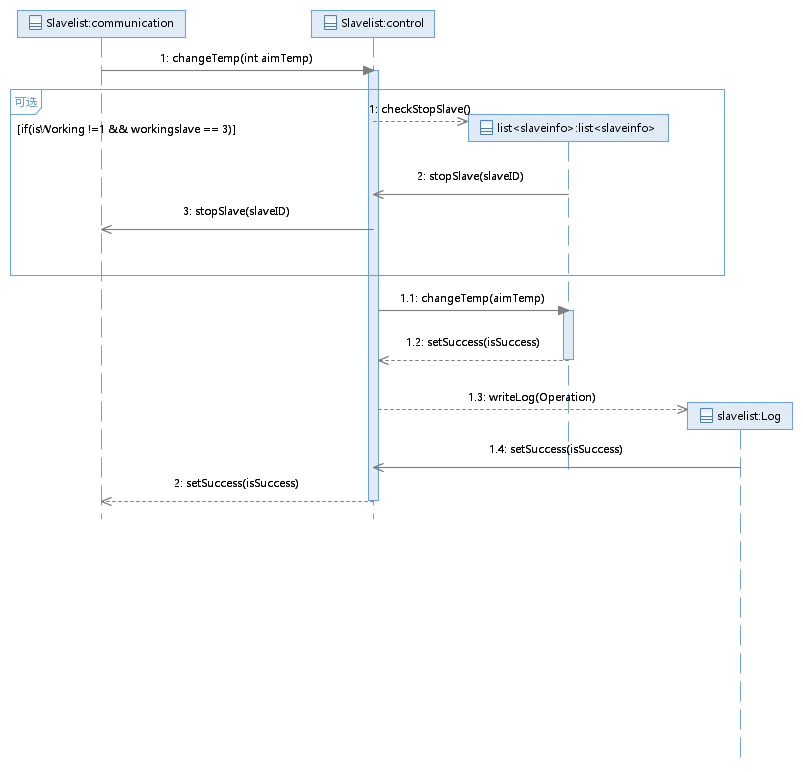
3、流程说明

①从机费用有变化时，通过通讯模块向主机调度模块发送房间费用信息

②主机调度模块收到费用信息后，写入数据库某房间的费用数据

③在空调管理员交互界面显示房间累积的费用

#### 4.1.3.3系统事件\_2: handleTemperature



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Communication | 软件类 | 中央空调通信类，控制主控机和从控机的通信 |
| Control | 软件类 | 对从控机的请求进行控制处理 |
| Log | 软件类 | 控制日志操作 |
| List<slavelinfo> | 数据类型 | 位于主控机，用于记录各从机的各项信息 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| changeTemp | aimTemp:目标温度 | Null | 更改目标温度请求 |
| checkStopSlave | null | Bool：true为可以挂起，false为没有可以挂起 | 检查是否有可以挂起的从机 |
| stopSlave | SlaveID：要挂起的从机ID | Null | 向指定从机发出挂起命令 |
| writLog | Operation:某项操作 | Null | 将动作记录至日志 |
| setSuccess | isSuccess：表示设置是否成功 | Null | 返回设定是否成功 |

3、流程说明

调用调度模块请求调整目标温度

判断是否正在工作或者正在工作的从机数量小于3

若不满足，则查看各从机的状态，寻找可以挂起的从机

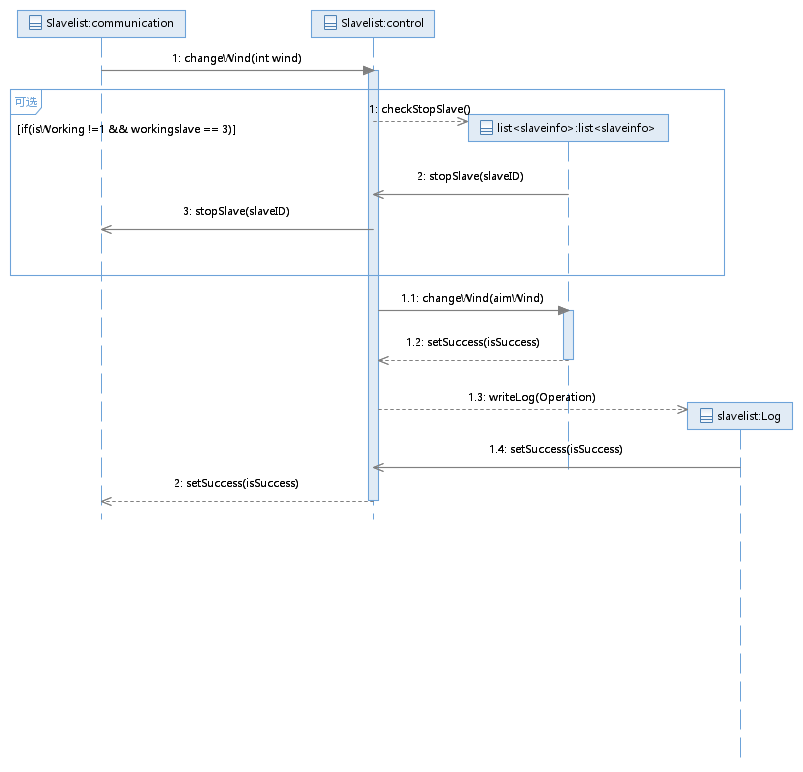
发出挂起指令。

若满足，改变目标温度

将该动作记录至日志文件

返回设置成功

#### 4.1.3.4系统事件\_3: handleWind

****

**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Communication | 软件类 | 中央空调通信类，控制主控机和从控机的通信 |
| Control | 软件类 | 对从控机的请求进行控制处理 |
| Log | 软件类 | 控制日志操作 |
| List<slavelinfo> | 数据类型 | 位于主控机，用于记录各从机的各项信息 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| changeWind | aimWind:目标风速 | Null | 更改风速请求 |
| checkStopSlave | null | Bool：true为可以挂起，false为没有可以挂起 | 检查是否有可以挂起的从机 |
| stopSlave | SlaveID：要挂起的从机ID | Null | 向指定从机发出挂起命令 |
| writLog | Operation:某项操作 | Null | 将动作记录至日志 |
| setSuccess | isSuccess：表示设置是否成功 | Null | 返回设定是否成功 |

3、流程说明

调用调度模块请求调整风速

判断是否正在工作或者正在工作的从机数量小于3

若不满足，则查看各从机的状态，寻找可以挂起的从机

发出挂起指令。

若满足，改变风速

将该动作记录至日志文件

返回设置成功

## 4.2 用户模块

### 4.2.1 基本用例（用户使用空调）

#### 4.2.1.1 系统事件列表

|  |  |  |
| --- | --- | --- |
| 系统事件名称 | 用例 | 参数说明 |
| 1、bool set\_temperature（int temp） | 温度设置（子用例） | 1、Temp：int类型；表示预期温度  2、返回值为0，,表示设置失败；返回值为1表示设置成功 |
| 2、bool set\_wind（int speed） | 风速设置（子用例） | 1、speed：int类型；表示预期风速；0表示弱风，1表示中风，2表示强风  2、返回值为0，表示设置失败；返回值为1表示设置成功 |
| 3、float show\_currentcost（string roomNum） | 实时显示花费（子用例） | 1. roomNum：string型； 2. 返回值为当前花费，浮点型 |
| 4、Report show\_report（string roomNum） | 查看详单（子用例） | 1.roomNum：string型；  2.返回值为用户详单，一个数据结构，其中包含时间、具体操作、花费 |

#### 4.2.1.2 系统事件1：设置目标温度

**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Customer | 系统使用者 | 使用从控机的外部对象 |
| UI | 概念类 | 与客户交互的界面 |
| Register | 软件类 | 控制器，从控机界面与逻辑处理模块的接口 |
| tempChange | 软件类 | 模拟温度变化（室温变化以及空调启动温度变化），控制从控机本地温度变化 |
| Communication | 软件类 | 从控机通信类，控制主控机和从控机的通信 |

1. 消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| Set\_temperature（int temp） | Temp:int类型，需要设置的温度 | bool res:true表示设置成功；false表示设置失败 | 发送目标温度 |
| Check（int temp） | Temp:int类型，需要设置的温度 | bool res:true表示温度合理；false表示温度不合理 | 判断温度是否位于合理范围内 |
| send\_targetTemp（int temp） | temp:int类型，需要设置的温度 | bool result:true表示温度设置成功；false表示设置不成功 | 控制器向通信类发送设置温度请求 |
| Air\_tempchange（int current\_temp） | Current\_temp:int类型，房间当前温度 | null | 温度模拟模块进行空调启动时的温度模拟 |
| natural\_temp（int current\_temp） | current\_temp:int类型，房间当前温度 | null | 温度模拟模块进行室温模拟 |
| Show\_currentTemp（int current\_temp） | Current\_temp：int类型，房间当前温度 | null | 从机界面显示当前温度 |
| Send\_temperature（int temp） | Temp：int类型，房间目标温度 | bool result:true表示温度设置成功；false表示设置不成功 | 界面向控制器发送目标温度 |
| Get\_currentTemp（int current\_temp） | Current\_temp：int类型，房间当前温度 | null | 控制器发当前温度给界面 |

3、流程说明

用户通过界面操作，设置期望温度

控制器检查该温度是否位于合理范围

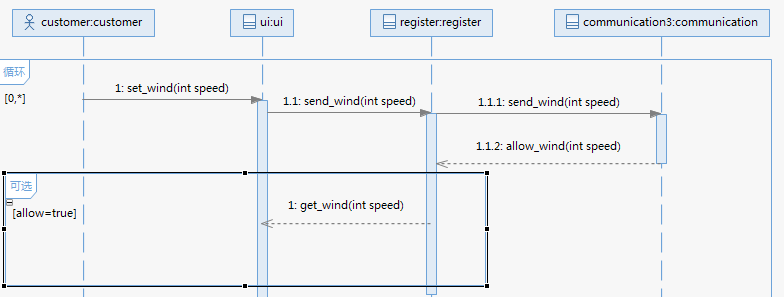
若范围合理，则通过从控机通信类发出调温请求，否则返回

从控机通信类得到主控机控制类周期性返回调节后的温度直至温度达到目标值

从控机接受调节后的温度。达到目标值后，利用本地温度变化函数模拟实际的室温变化

当室温与理想值相差超过1度后，再次发起调温请求，如此反复

#### 4.2.1.3 系统事件（设置风速）



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Customer | 系统使用者 | 使用从控机的外部对象 |
| UI | 概念类 | 与客户交互界面 |
| Register | 软件类 | 控制器，从控机界面与逻辑处理模块的接口 |
| Communication3 | 软件类 | 位于从控机方，从控机通信类，控制主控机和从控机的通信 |

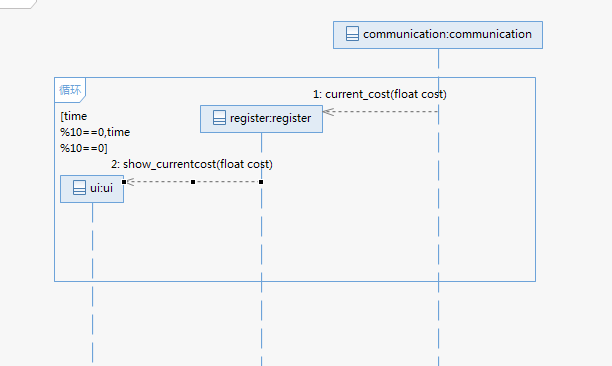
2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| set\_wind（int speed） | speed:int类型，理想风速 | bool res:true表示设置成功；false表示设置失败 | 用户设置风速，并在界面显示 |
| send\_wind（int  speed） | speed:int类型，理想风速 | bool res:true表示发送风速请求成功；false表示发送失败 | 界面向控制器传递理想风速；控制器向通信类传递理想风速 |
| allow\_wind（int speed） | speed:int类型，理想风速; | bool res:true表示请求被应答；false表示请求未应答 | 主机应答调节风速请求 |
| get\_wind（int speed） | speed:int类型，理想风速 | null | 控制器向ui发送空调允许调风成功信息 |

3、流程说明

* 用户通过界面操作，设置期望风速
* 通过从控机通信类发出调风速请求
* 若在规定时间内，得到主控机通信类的回复表示请求被响应，则界面显示期望风速

#### 4.2.1.4 系统事件3（显示花费）



**时序图说明：**

1. 类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| UI | 概念类 | 与客户交互的界面 |
| Register | 软件类 | 控制器，从控机界面与逻辑处理模块的接口 |
| Communication | 软件类 | 位于从控机方，从控机通信类，控制主控机和从控机的通信 |

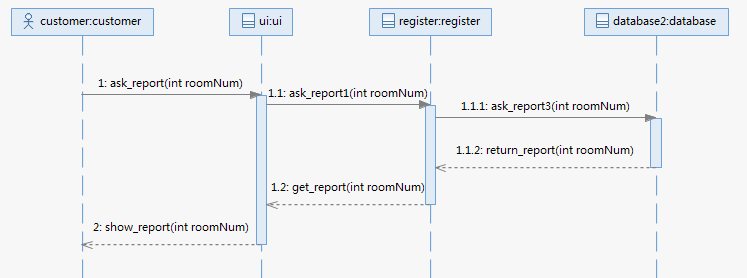
2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| current\_cost（float cost） | cost:float类型，当前价格 | null | 通信类想控制器定时传递当前消费 |
| show\_currentcost（float cost） | cost:float类型，当前价格 | null | 定时显示当前价格到界面 |

3、流程说明

* 从控机通信类通过通信模块定时循环发送当前费用的请求
* 收到主控机的费用请求后，显示于界面

#### 4.2.1.5 系统事件4（查看详单）



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Customer | 系统使用者 | 使用从控机的外部对象 |
| UI | 概念类 | 与客户交互界面 |
| Register | 软件类 | 控制器，从控机界面与逻辑处理模块的接口 |
| database | 软件类 | 数据库，位于从控机方，存储每一个房间的操作及花费 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| Ask\_report（int roomNum） | roomNum:int类型，房间号 | Bool型，true表示请求成功false表示请求失败 | 用户在退房时，要求查看详单 |
| Ask\_report1（int  roomNum） | roomNum:int类型，房间号 | Bool型，true表示发成功，false表示发送失败 | Ui界面向控制器发送查看详单请求 |
| Ask\_report3（int roomNum） | roomNum:int类型，房间号 | Bool型，true表示查看成功。False表示查看失败 | 控制器向数据库发送查看详单请求 |
| Return\_report（int roomNum） | roomNum:int类型，房间号 | Report：详单数据结构 | Database数据库返回查询的结果 |
| show\_report（int  roomNum） | roomNum:int类型，房间号 | Report：详单数据结构 | 用户查看到详单。 |
| Get\_report（int roomNum） | roomNum：int类型，房间号 | Report：详单数据结构 | 控制器返回给UI界面 |

3、流程说明

* 用户退房时，要求查看详单。
* 酒店查询数据库，通过房间号，筛选出该房间的所有记录，返回详单
* 打印出详单

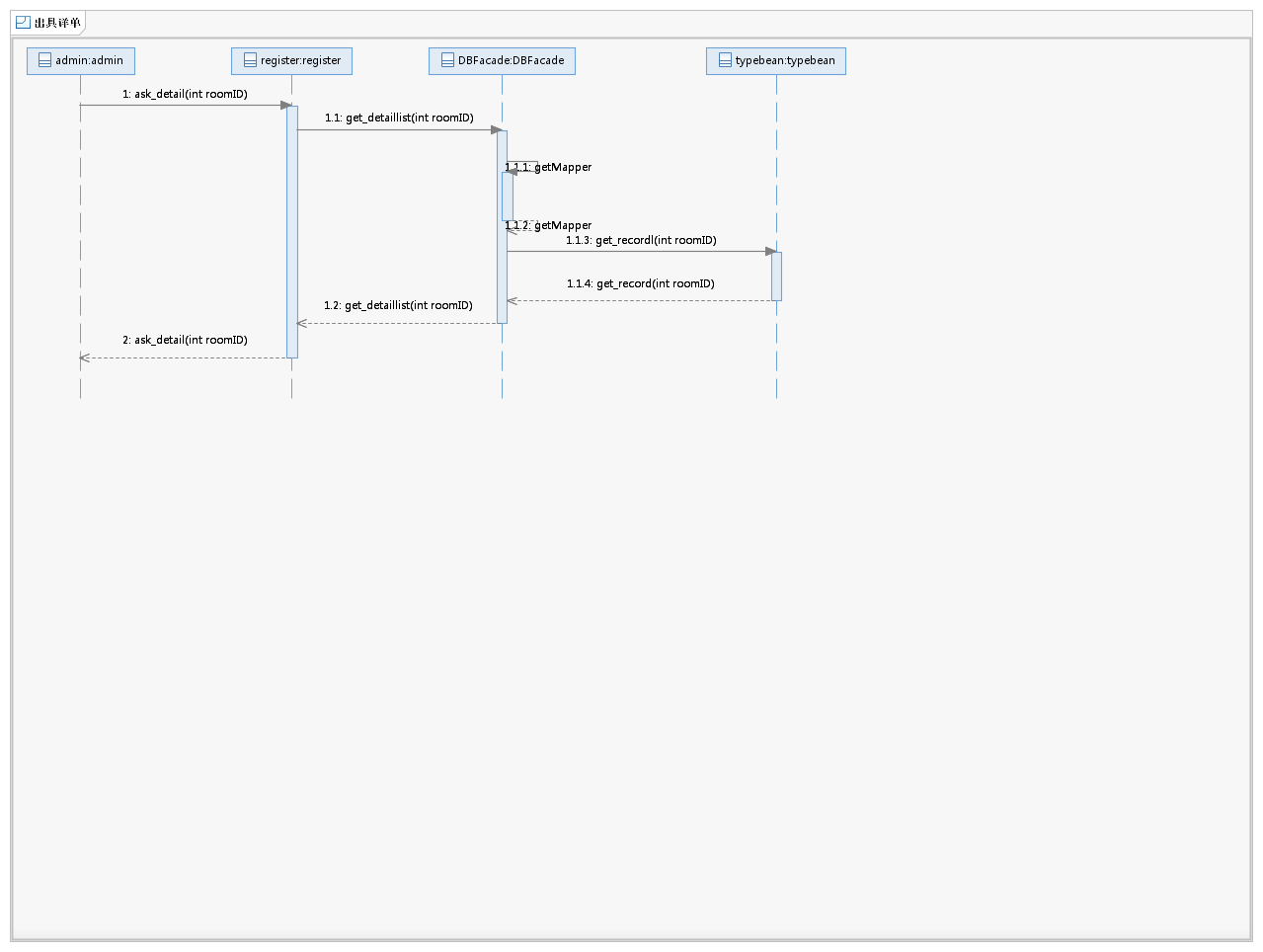
## 4.3 统计模块

### 4.3.1 基本用例（出具详单）

#### 4.3.1.1系统事件列表

|  |  |  |
| --- | --- | --- |
| 系统事件名称 | 用例 | 参数说明 |
| 1 void get\_detail\_bill（detailbill dbill） | 出具详单(基本用例) | 1、 dbill: detailbill 结构类型，表示整个消费详单（历史温度、历史风速、时间、功率、费用等） |

#### 4.3.1.2系统事件\_1: void get\_detail\_bill



**时序图说明：**

1、类说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | | 性质 | 功能说明 |
| admin:admin | | 前台管理人员 | 前台管理员（服务员），使用系统的外部类 |
| Register | | 软件类 | 控制器，前台管理界面与逻辑处理模块的接口 |
| DBFacade | 软件类 | | 在数据库操作时，用于维护领域类和负责其持久化的对应Mapper类之间的对应关系 |
| TypeBean | 软件类 | | Type 是泛指，表示任意一个需要持久化的领域 类类名，TypeBean 专门负责该领域类的持久化 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| ask\_detail(int roomID) | roomID | Detail detaillist：由多条记录组成的详单 | 将详单及房间号进行传递 |
| get\_detaillist(int roomID) | roomID | Detail detail：由多条记录组成的详单 | 构造消费详单 |
| getMapper | Null | Null | 查询需要的接口类型，并进行选择 |
| get\_record（int roomID) | roomID | Detail record：获取所有的消费细则 | 获取消费记录 |

3、流程说明

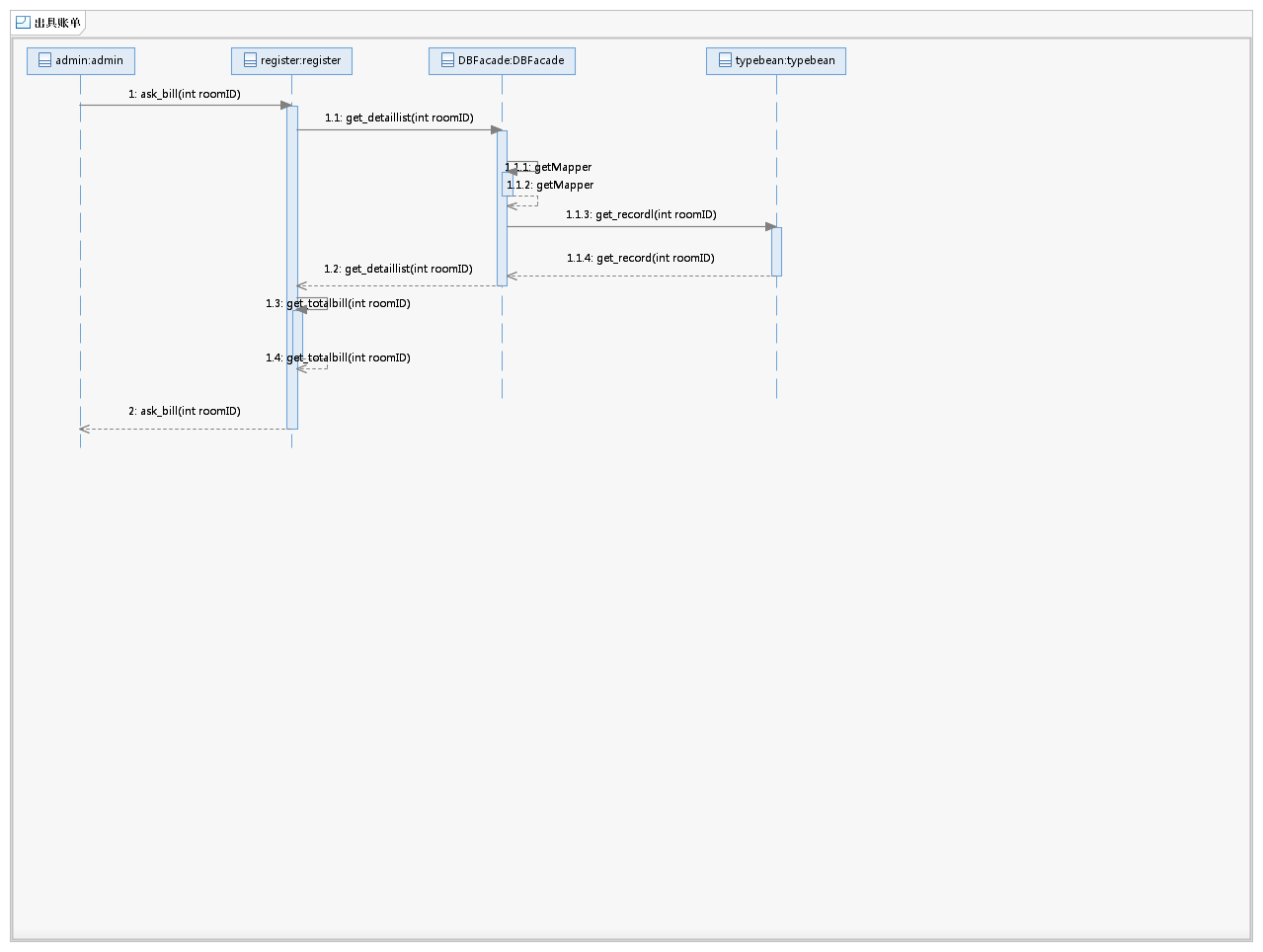
* 前台管理员向系统发出详单请求，传送发出请求的id
* 控制器查询消费详单，从数据库中获取该从控机的消费详情
* 返回消费详情

### 4.3.2 基本用例（出具账单）

#### 4.3.2.1系统事件列表

|  |  |  |
| --- | --- | --- |
| 系统事件名称 | 用例 | 参数说明 |
| 1 void print\_bill（detailbill dbill） | 出具账单(基本用例) | 1、dbill: detailbill 结构类型，表示整个消费详单（历史温度、历史风速、折扣率、单价、总功率等） |

#### 4.3.2.2系统事件\_1: void print\_bill



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| admin | 前台管理人员 | 前台管理员（服务员），使用系统的外部类 |
| Register | 软件类 | 控制器，前台管理界面与逻辑处理模块的接口 |
| DBFacade | 软件类 | 在数据库操作时，用于维护领域类和负责其持久化的对应Mapper类之间的对应关系 |
| TypeBean | 软件类 | Type 是泛指，表示任意一个需要持久化的领域 类类名，TypeBean 专门负责该领域类的持久化 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| ask\_bill(int roomID) | roomID | Double totalcost计算后得到的消费总账单 | 将账单和房间号进行传递 |
| get\_detaillist(int roomID) | roomID | Detail detail：由多条记录组成的详单 | 构造消费详单 |
| getMapper | Null | Null | 查询需要的接口类型，并进行选择 |
| get\_record（int roomID) | roomID | Detail record：获取所有的消费细则 | 获取消费记录 |
| Gettotalbillint roomID) | roomID | Double totalcost计算相应的总消费 | 计算消费的总价 |

3、流程说明

* 前台工作人员发出获取消费详单请求
* 控制器从消费记录数据库中获取消费详情
* 计算总价并打印之。
* 内存中不做任何修改

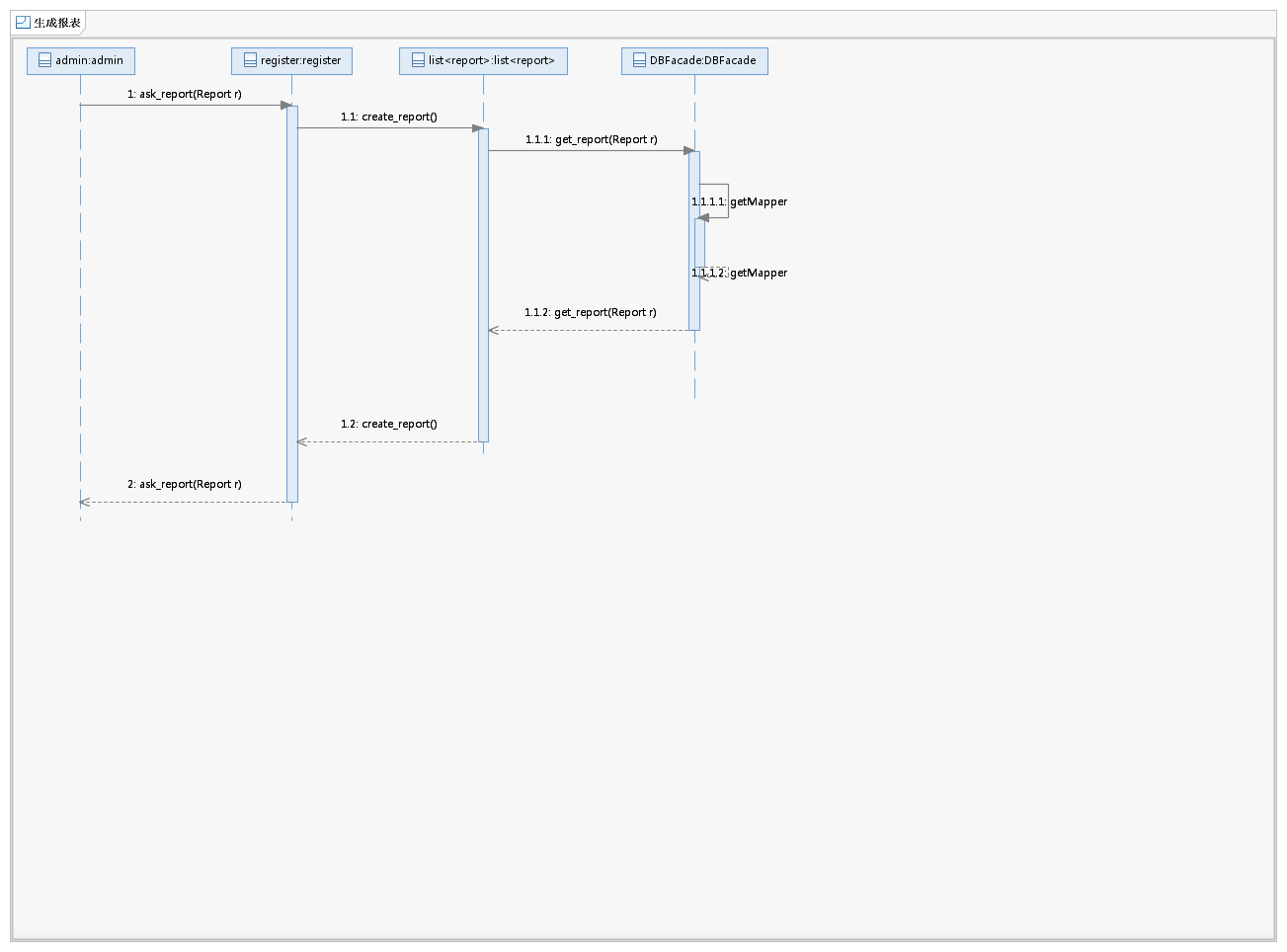
## 4.4 酒店管理模块

### 4.4.1 基本用例(分析消费情况)

#### 4.4.1.1系统事件列表

|  |  |  |  |
| --- | --- | --- | --- |
| 系统事件名称 | 用例 | | 参数说明 |
| 1. void get\_report(Report rt) | | 分析顾客消费情况(基本用例) | 1. rt：Report结构类型，表示由从控机消费历史信息得出的报表（温度偏好、风速偏好、消费金额偏好、用户组偏好等） |

#### 4.4.1.2系统事件\_1: get\_report



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| admin | 酒店管理员 | 酒店管理员，使用系统的外部类 |
| register | 软件类 | 控制器，酒店管理界面与逻辑处理模块的接口 |
| list<report> | 数据类型 | 用于存储生成的报表 |
| DBFacade | 软件类 | 在数据库操作时，用于维护领域类和负责其持久化的对应Mapper类之间的对应关系 |

2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| ask\_report(Report rt) | rt | Report rt:文字类型的报表 | 将报表的文字模块及图表展现给管理员 |
| Create\_report() | null | success | 创建报表 |
| getMapper | Null | Null | 查询需要的接口类型，并进行选择 |
| get\_record(Report r) | r | record r空调使用记录 | 获取空调使用记录 |

3、流程说明

* 酒店管理员获取报表并进行分析。
* 控制器对数据库中内容进行分析形成报表。
* 更新内存中报表类。

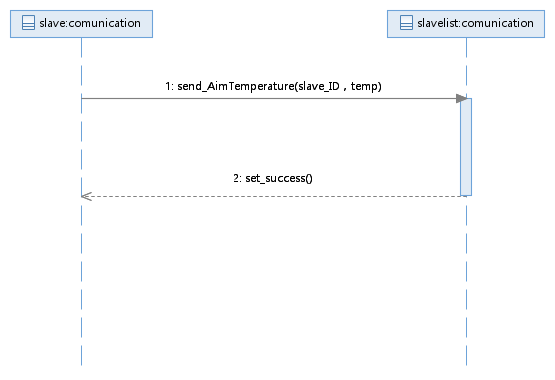
## 4.5通信模块

### 4.5.1基本用例（主从机通信）

#### 4.5.1.1系统事件列表

|  |  |  |
| --- | --- | --- |
| 系统事件名称 | 用例 | 参数说明 |
| 1 void send\_AimTemperature(slave\_ID，temp) | 设置温度(子用例) | 1. 参数为设置的温度,从控机ID 2. 无返回值 |
| 2 void send\_wind（wind,slave\_ID） | 设置风力（子用例） | 1. 参数为设置的风速，从控机ID 2. 无返回值 |
| 3 void get\_price(slave\_ID) | 获取当前消费价格（子用例） | 1、参数为从控机ID，无返回值 |
| 4 void monitor\_state（slave\_ID,temp） | 发送当前温度(子用例) | 1. 参数为从控机ID,当前温度值 2. 无返回值 |
| 5 void initSlave(slave\_ID,mode,upper\_tem,lower\_tem,default\_aim,winspeed) | 获取从机初始化信息（子用例） | 1、参数为从控机ID，主机工作模式，最高目标温度，最低目标温度，默认目标温度，默认风速  2、无返回值 |

#### 4.5.1.2系统事件\_1: send\_AimTemperature



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| slave:communication | 软件类 | 负责从控机的通信 |
| slavelist:communication | 软件类 | 负责主控机的通信 |

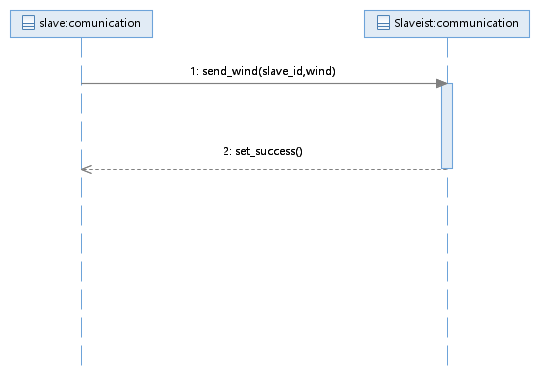
2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| Send\_AimTemperature(slave\_ID,temp) | temp:客户要求设定温度  Slave\_ID:发出此消息的从控机ID | Void:由于是异步消息，无返回值 | 发送目标温度 |
| Set\_success() | 无 | Bool：设置成功为true，失败为false | 返回是否设置成功 |

3、流程说明

* 从控机向主控机发送设置温度请求。
* 主控机收到请求，返回设置是否成功。

#### 4.5.1.3系统事件\_2: send\_wind



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Slave:communication | 软件类 | 负责从控机的通信 |
| Slavelist:communication | 软件类 | 负责主控机的通信 |

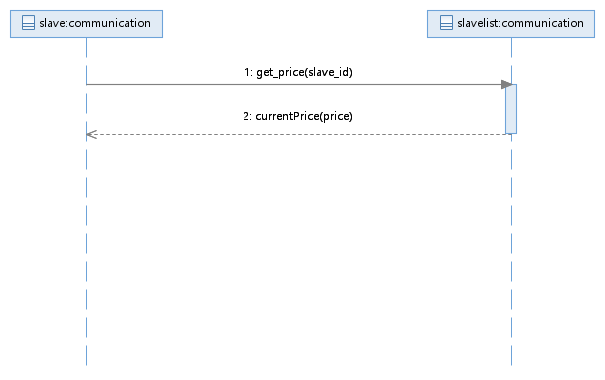
2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| Send\_wind(slave\_ID，wind) | wind:客户要求设定的风速  Slave\_ID:发出此消息的从控机ID | Void:由于是异步消息，无返回值 | 发送理想风速 |
| Set\_success() | 无 | Bool：设置成功为true，失败为false | 返回是否设置成功 |

3、流程说明

* 从控机向主控机发送设置风力请求。
* 主控机收到请求，返回设置是否成功。

#### 4.5.1.4系统事件\_3: get\_price



**时序图说明：**

1. 类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Slave:communication | 软件类 | 负责从控机的通信 |
| Slavelist:communication | 软件类 | 负责主控机的通信 |

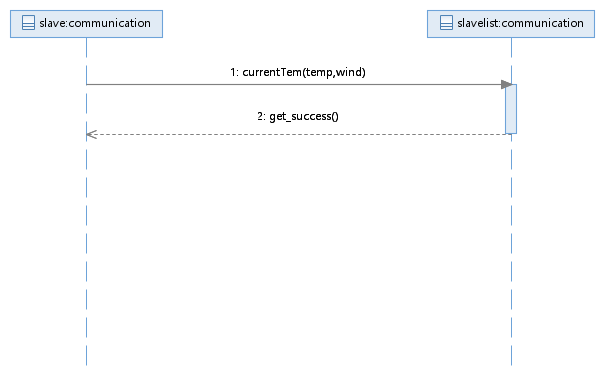
2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| get\_price(slave\_ID) | Slave\_ID:发出此消息的从控机ID | Void:由于是异步消息，无返回值 | 请求当前房间价格。 |
| currentPrice(price) | price:当前目标房间价格 | Void:由于是异步消息，无返回值 | 给予目标房间当前房费 |

3、流程说明

* 从控机向主控机发送获得当前消费情况请求。
* 主控机收到请求，返回当前消费情况

#### 4.5.1.7系统事件\_6: monitor\_state



**时序图说明：**

1、类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Slave:communication | 软件类 | 负责从控机的通信 |
| Slavelist:communication | 软件类 | 负责主控机的通信 |

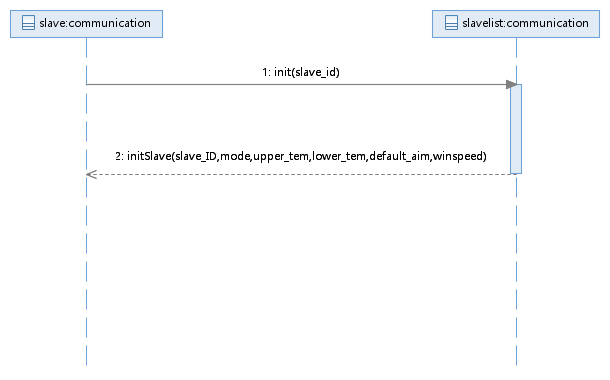
2、消息说明

|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| currentTem(temp,wind) | Temp:目标房间当前温度。  Wind:目标房间当前风力。 | Void:由于是异步消息，无返回值 | 对获取房间状态请求的回应。 |
| get\_success() | 无 | Bool：设置成功为true，失败为false | 返回是否处理成功 |

3、流程说明

* 从控机向主控机发送获取当前状态信息（温度和风速）
* 主控机收到请求，是否处理成功

#### 4.5.1.8系统事件\_7: initSlave



**时序图说明：**

1. 类说明

|  |  |  |
| --- | --- | --- |
| 名称 | 性质 | 功能说明 |
| Slave:communication | 软件类 | 负责从控机的通信 |
| Slavelist:communication | 软件类 | 负责主控机的通信 |

2、消息说明

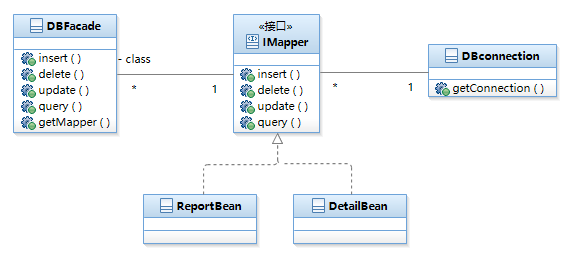
|  |  |  |  |
| --- | --- | --- | --- |
| 消息名 | 消息参数 | 返回类型 | 消息说明 |
| Init(slave\_id) | Slave\_ID:发出此消息的从控机ID | Void:由于是异步消息，无返回值 | 从机发送初始化请求 |
| initSlave(slave\_ID,mode,upper\_tem,lower\_tem,default\_aim,winspeed) | Slave\_id:从控机ID，mode:主机工作模式，  Upper\_tem:最高目标温度，  Lower\_tem:最低目标温度，  Default:默认目标温度，  Winspeed:默认风速 | Void:由于是异步消息，无返回值 | 主机向从机返回各项初始化的状态信息 |

3、流程说明

* 从控机向主控机发送初始化请求。
* 主控机收到请求，返回各项初始化信息。

## 4.6 持久化层设计

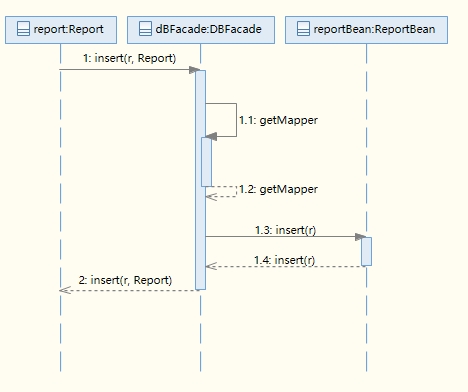
设计持久化层：对报表信息和详单信息需要进行持久化。持久化层的设计如下：



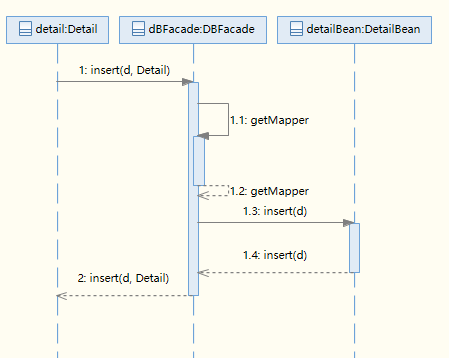
ReportBean：负责报表信息持久化；

DetailBean：负责详单信息持久化

### 4.6.1 报表信息持久化过程



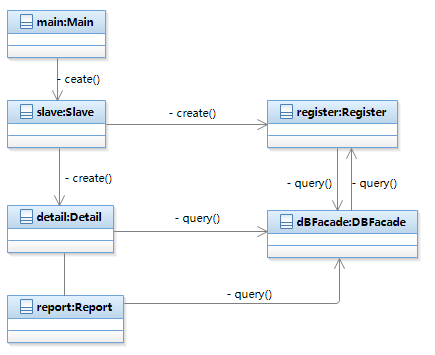
### 详单信息持久化过程



## 4.7 StartUp设计

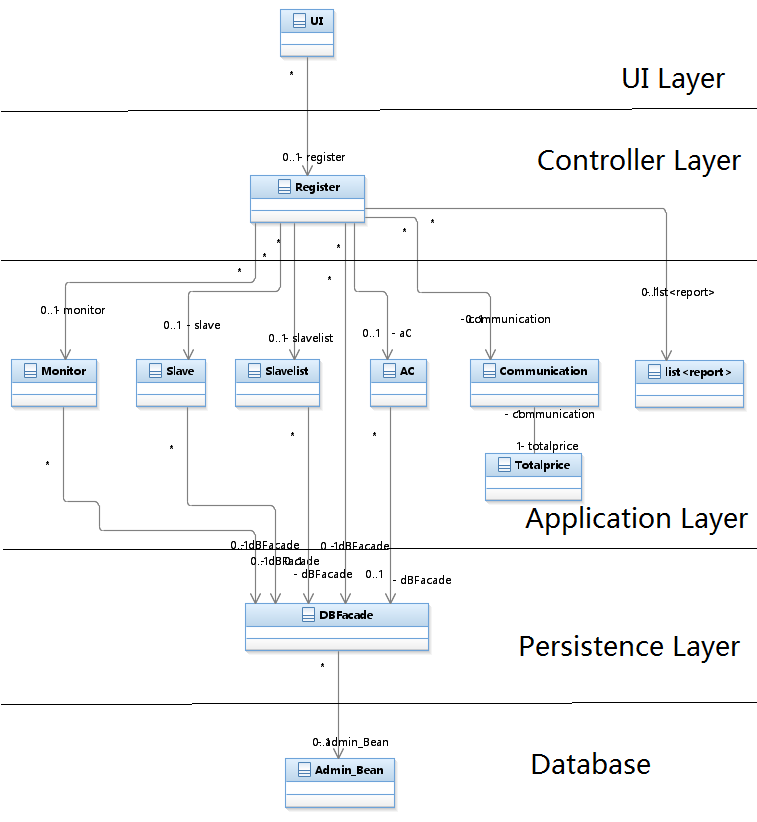
通过设置一个startUp功能来完成系统的相关初始化工作。根据StartUp的原则和分析结果，可有如下初始化内容：选择从机对象为根对象。

1. 创建Slave（从机），Register（外部控制器），Report（报表），Detail（详单）；
2. 建立Report（报表），Detail（详单）之间的关联；
3. 建立Detail（详单）与Salve（从机）之间的关联；
4. 建立Register与Detail（详单）的关联。



# 系统静态结构下

## 5.1 系统整体静态结构



## 5.2 模块化静态结构

### 5.2.1监视从机状态

#### 5.2.1.1类列表

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| UI | slave\_UI | Button btn  Plane plane  Label label | 1. powerOn()  2. create()  3. powerOff()  4. show(stateInfo,ID) | |

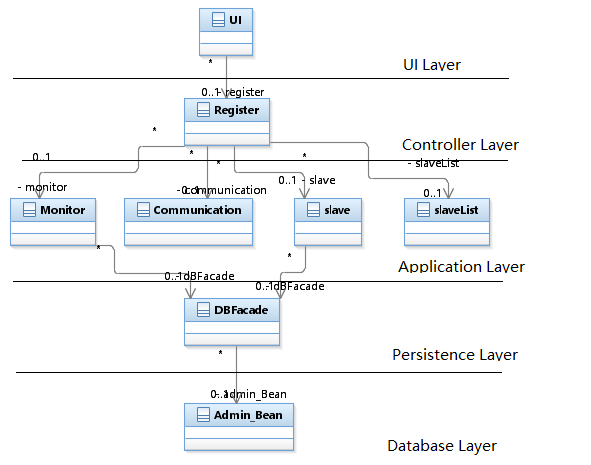
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Back | DBFacade |  | 1.getInfo(type)  2.getMapper()  3.saveData(info) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Bean | Admin\_Bean |  | 1. getInfo(list)  2. insertInfo(info) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Action | air\_register |  | 1.update(stateInfo,ID) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Service | Monitor | String ID  State stateInfo | 1.upate(stateInfo,ID)  2.monitor(State stateInfo, String ID) |
| communication | State stateInfo  String ID | 1.monitor(State stateInfo, String ID) |
| slave | Int workMode  Int windspeed  Double highTem  Double lowTem  Doubel price |  |
| slaveList | String ID  Int windSpeed  Double tem |  |

#### 5.2.1.2 类图



### 5.2.2 主机设置模块

#### 5.2..2.1类列表

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| UI | slave\_UI | Button btn  Plane plane  Label label | 1. powerOn()  2. create()  3. powerOff()  4. show(stateInfo,ID)  5.setMain()  6.setMode(setList setInfo) |

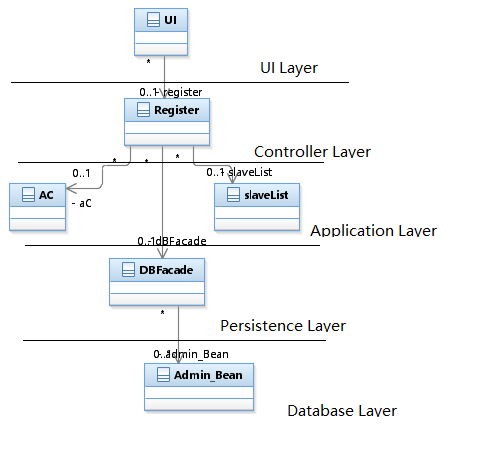
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Back | DBFacade |  | 1.getInfo(type)  2.getMapper()  3.saveData(info) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Bean | Admin\_Bean |  | 1. getInfo(list)  2. insertInfo(info) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Action | air\_register |  | 1.setMode(setList setInfo)  2.setAC(setList setInfo) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Service | ac | Int workMode  Int windspeed  Double highTem  Double lowTem  Doubel price |  |
| slaveList | String ID  Int windSpeed  Double tem |  |

#### 5.2.2.2类图



### 5.2.3主机调度

#### 5.2.3.1类列表

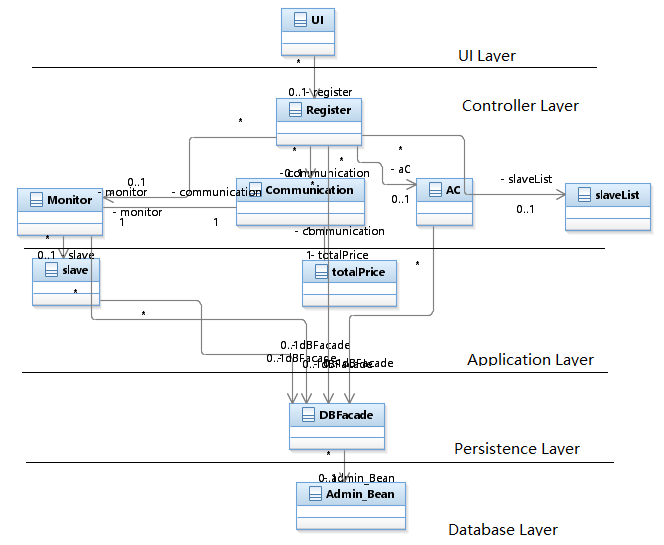
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| UI | slave\_UI | Button btn  Plane plane  Label label | 1.showPrice(String ID, double price) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Back | DBFacade |  | 1.pushPrice(String ID, double price) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Service | Monitor | String ID  State stateInfo | 1.monitor(State stateInfo, String ID) |
| communication | State stateInfo  String ID | 1.pushPrice(String ID, double Price)  2.changeTemp(int aimTemp)  3.stopSlave(slaveID)  4.setSuccess(isSuccess)  5.changeWind(int wind)  6. |
| slave | Int workMode  Int windspeed  Double highTem  Double lowTem  Doubel price |  |
| slaveList | String slaveID  Int windSpeed  Double tem  Int aimTemp | 1.changeTemp(int aimTemp)  2.stopSlave(slaveID)  3.setSuccess(isSuccess)  4.checkStopSlave()  5.writeLog(Operation)  6.changeWind(aimWind) |
| totalPrice | String ID  Double price | 1.pushPrice(String ID, double Price)  2.showPirce(String ID, double Price)  3. |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| Bean | Admin\_Bean |  | 1. getInfo(list)  2. insertInfo(info) |

#### 5.2.3.2类图



### 5.2.4温控设置：

#### 5.2.4.1类列表：

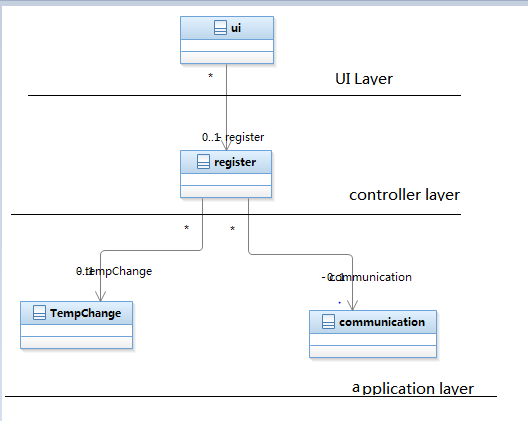
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| UI | **UI** | JButton btn  Plane plane  Label label  ComboBox cb | 1.void send\_temperature(int temp)  2.void send\_wind(int speed) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
| controller | **register** |  | 1.bool send\_targetTemp(int temp)  2.void send\_wind(int speed)  3.bool result=checke\_valid(int temp) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |
|  | **TempChange** | Int change\_func  Int cur\_temp | 1. void **natural\_temp（int current\_temp）** 2. **Void Air\_tempchange（int current\_temp）** |

|  |  |  |  |
| --- | --- | --- | --- |
| application | communication | Int slaveID | 1..bool request\_wind(int slaveID,int speed)  2.bool end\_temp(int temp) |

#### 5.2.4.2 类图：



### 5.2.5实时显示花费：

#### 5.2.5.1类列表

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| UI | **UI** | Plane plane  Label label  ComboBox cb | Void Show\_currentcost(float cost) |

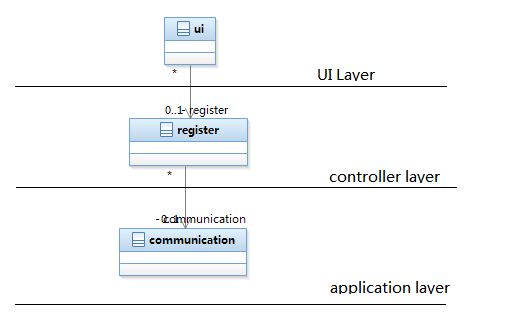
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| controller | **register** |  | 1.void current\_cost(float cost) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| application | **ommunication** | Int slaveID | 1.void current\_price\_request(float cost) |

#### 5.2.5.2类图



### 5.2.6查看报表

#### 5.2.6.1类列表

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| UI | **1.slave\_UI** | JButton btn  Plane plane  Label label  ComboBox cb | **Bool Ask\_report（int roomNum）** |

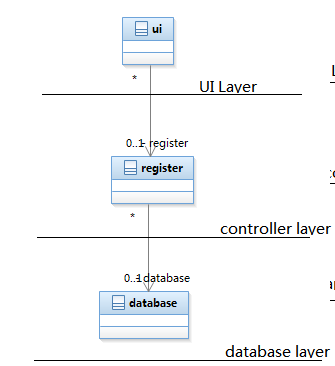
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Register | **1.salve\_register** |  | **Bool Ask\_report1（int roomNum）** |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| database | **1.slave\_database** | Int slaveID | **Bool Ask\_report3（int roomNum）** |

#### 5.2.6.2类图



### 5.2.7出具账单

### 5.2.7.1类列表

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| UI layer | **1.** admin | JButton btn  Plane plane  Label label  ComboBox cb | 1.void on\_confirm\_click()  2.void on\_cancel\_click()  3.void ask\_detail(int roomID) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Controller | **1.** Register | String Operator | 1.void get\_detaillist(int roomID)  2.void bill(int slaveID) |

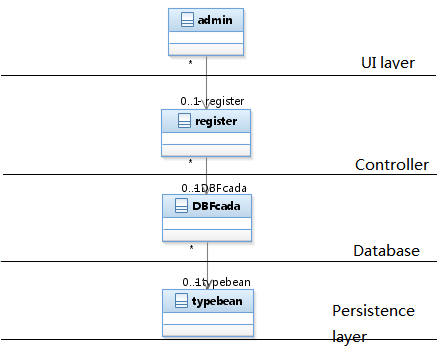
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Database | **1.** **DBFacade** | consumeDetail dbill  qtime time | 1.void getMapper()  2.void get\_record（int roomID） |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Persistence layer | **1.** TypeBean |  | 1.void get\_consumeDetail()  2.void getinfo(consumeDetail dbill)  3.void insertinfo(consumeDetail dbill) |

#### 5.2.7.2 类图



### 5.2.8出具详单

#### 5.2.8.1类列表

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| UI layer | **1.** admin | JButton btn  Plane plane  Label label  ComboBox cb | 1.void on\_confirm\_click()  2.void on\_cancel\_click()  3.void ask\_bill(int roomID) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Controller | **1.** Register | String Operator | 1.void get\_detaillist(int roomID)  2.void Get\_totalbill(int roomID) |

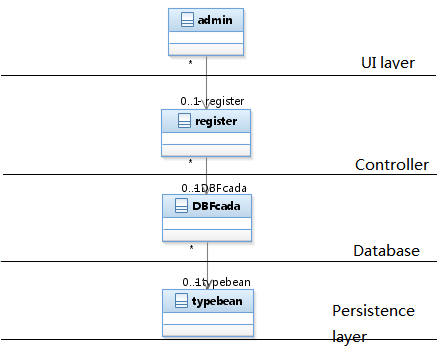
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Database | **1.** **DBFacade** | consumeDetail dbill  qtime time | 1.void getMapper()  2.void get\_record（int roomID） |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Persistence layer | **1.** TypeBean |  | 1.void get\_consumeDetail()  2.void getinfo(consumeDetail dbill)  3.void insertinfo(consumeDetail dbill) |

#### 5.2.8.2 类图



### 5.2.9分析报表

#### 5.2.9.1类列表

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| UI | **1.admin** | JButton btn  Plane plane  Label label  ComboBox cb | 1.void on\_confirm\_click()  2.void on\_cancel\_click()  3.void ask\_report(Report rt) |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Controller | **1.** **register** | String Operator | 1.void Create\_report() |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Application | **3.** **list<report>** | reporttable rt | 1.void get\_record(Report r) |

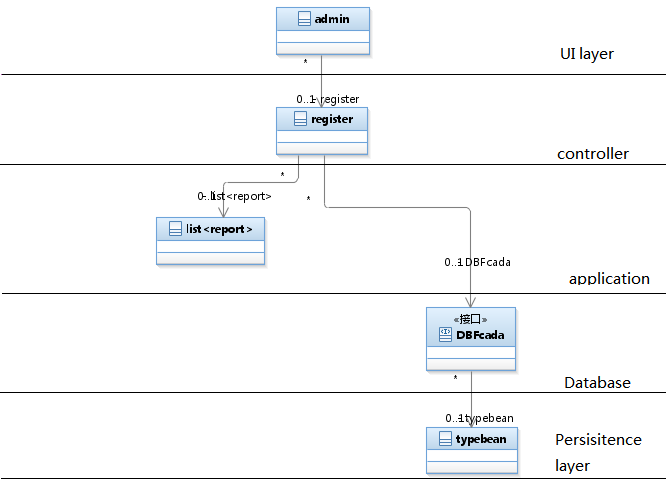
|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Database | **1.** **DBFacade** | consumeDetail dbill | 1.Void getMapper() |

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Persistence layer | **1.** TypeBean | String Operator  consumeDetail dbill | 1.void get\_consumeDetail() |

#### 5.2.9.2 类图



### 5.2.10主机与从机间通信

#### 5.2.10.1类列表

|  |  |  |  |
| --- | --- | --- | --- |
| 层级 | 类名称 | 类属性 | 类方法 |

|  |  |  |  |
| --- | --- | --- | --- |
| Application | **1 slave** | BOOL connection | 1 void send\_AimTemperature(temp,slaveID)  2 void send\_wind（wind,slave\_ID）  3 void get\_price(slave\_ID)  4 void monitor\_state（slave\_ID, temp）  5void initSlave(slave\_ID,mode,upper\_tem,lower\_tem,default\_aim,winspeed) |
| **2 slavelist** | BOOL connection | 1 void send\_AimTemperature(temp,slaveID)  2 void send\_wind（wind,slave\_ID）  3 void get\_price(slave\_ID)  4 void monitor\_state（slave\_ID, temp）  5 void initSlave(slave\_ID,mode,upper\_tem,lower\_tem,default\_aim,winspeed) |

#### 5.2.10.2类图

