长 沙 学 院

软件工程基础实训Ⅱ 实训

|  |  |
| --- | --- |
| **题目** | **基于C/S架构的餐饮管理系统** |
| **学院** | **计算机科学与工程** |
| **专业(班级)** | **软件工程（21软件01）** |
| **姓名** | **沈沐** |
| **学号** | **B20210304115** |
| **指导教师** | **杨刚、潘怡** |
| **起止日期** | **2023.06.05～2023.06.16** |

实训任务书

**课程名称：软件工程基础实训II**

**实训题目：基于C/S架构的餐饮管理系统**

### 已知技术参数和设计要求：

1. 问题描述（功能要求）：

本次案例将完成一个基于C/S架构的餐饮管理系统。

（1）.进行餐饮管理系统的功能设计。

（2）.设计餐饮管理系统的数据库。

（3）.利用JAVA的GUI类设计开发友好的人机接口。

（4）.利用JAVA语言设计监听用户操作和JDBC进行数据库的处理。

餐饮管理系统主要有6大模块：

（1）.员工管理：对员工实现增删改查。

（2）.客户管理：对客户实现增删改查。

（3）.餐台管理：对餐台实现增删改查。

（4）.菜品管理：对菜品分类、菜品实现增删改查。

（5）.点菜管理：服务员对某客户、某一空闲餐台实行开台，同时实现点菜，将餐台号与所点的菜品对应起来，分别显示出来，并记录开台时间。

（6）.结账管理：收银员对某一餐台通过统计显示消费的菜品清单统计出消费金额，通过手动输入实收金额进行找零的计算，并显示，完成结账的操作，并记录成统计数据。

基础实训II有2周时间40学时，学生基本必做任务：独立完成基于C/S架构的餐饮管理系统。

选做扩展：菜品推荐、销售统计。

**2. 运行环境要求：**

（1）客户端：

windows操作系统，JDK1.6，Idea或Eclipse。

（2）服务器：

MySQL5.6或 SQL Server 数据库服务器。

3. 技术要求：

1） 掌握软件工程的需求分析和系统设计方法。

2) 掌握数据库的分析与设计，完成餐饮管理系统数据库的设计。

3) 掌握JAVA的GUI设计方法，完成登录、主界面、菜品分类管理界面、菜品管理界面等人机接口。

4） 掌握JAVA面向对象程序设计方法，设计系统所需的各种类。

5) 掌握JDBC方法，对数据库进行操作，完成主程序和菜品分类、菜品管理等功能。

### 实训工作量：

40课时

### 工作计划：

1.班级

21软件1-4

2.课时及教室安排

见实训课程工作计划表。

实训报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学生姓名 | 沈沐 | 学号 | B20210304115 | 班级 | 21软件01 |
| 专业 | 软件工程 | | 指导教师姓名 | 杨刚、潘怡 | |
| 项目名称 | 基于C/S架构的餐饮管理系统 | | | | |
| **实训过程**：  需求分析：在收到设计餐饮管理系统这一任务后，对任务进行解读，明白此系统需要根据实际出发，切合所需要求，提供科学有效的管理模式。可以针对点菜、结账、订餐台等复杂环节进行管理，以降低人工成本，提高日常工作效率。同时也可以针对该餐饮业的问题进行适当调节，使其可以更好应对市场，而不是被市场淘汰。  系统设计：根据该餐饮系统管理要求，设计所需实体，并分析其中所存在的联系，系统一共六个模块，分别为员工管理、客户管理、餐台管理、菜品管理、点菜管理以及结账管理。员工管理实现对员工资料的增删改查功能。客户管理需实现对客户资料的增删改查功能。餐台管理需要对餐台的相应资料实现增删改查这一功能。菜品管理则也是对菜品分类以及菜品实现增删改查功能。点菜管理是服务员对某客户、某一空闲餐台实行开台，同时实现点菜，将餐台号与所点的菜品对应起来，分别显示出来，并记录开台时间的功能。结账管理：收银员对某一餐台通过统计显示消费的菜品清单统计出消费金额，通过手动输入实收金额进行找零的计算，并显示完成结账的操作，并记录成统计数据。  编码实现：在需求分析与系统设计正式完成后，开始对餐饮管理系统有了一个大概思路，并开始有主向细开始实现编码，从主界面的编码实现，到数据库的设计分析，再到jdbc连接数据库，然后开始餐饮管理系统的各项功能具体实现。  系统测试：我在数据库存入相应例子，然后开始逐一测试。 | | | | | |
| **实训成果**：  完成基于C/S架构的餐饮管理系统这一设计。 | | | | | |
| **实训总结**：（包括心得体会、存在的问题和改进方向。）  本栏正文排版要求：宋体5号，行距18磅，段首缩进2字符。  为期两周的实训结束了，在这两周的课程中，我体会到了许多平常理论课上学不到的东西。以前的课堂学习让我们心里有了理论基础，实践起来也有所因循。都说“实践出真知”，在此次课程设计中我将理论课学到的知识点运用到实际，才发现远远没有想象中的那么简单。  在此次课程设计中，需要去设计一个餐饮管理系统。在日常生活中去外面吃饭，我还对餐饮方面的操作不以为然，自以为容易，真正操作起来也还是有些复杂和难度的。正常的管理员注册登录页面以及主页面的设计让我更加熟练了对页面按钮的设计与操作。也正式完整的在系统中运用了数据库，在连接数据库时遇到了一些小问题，但最终的得以解决，也让我对使用JDBC连接数据库更加熟练。在实现餐饮管理系统的各个功能时，我总会遇到许许多多的小问题，例如添加失败、删除失败等等。这让我明白了在实现代码时一定要细心和耐心。  在此餐饮管理系统中我还有许多不完善的地方，例如结账页面完成后无法直接更新至订单支付状态，需要后续再去订单详情修改。还有菜品客户等更新时不能直接更新，在更新完成后我需要刷新页面进入才能更新完成。我深知自己有许许多多不足的地方，也知道我的代码能力远远不如别人，为此，我会更加努力，付出更多时间与精力在此之中。在注重理论知识学习的同时更加还要去加强实践能力，不能光做到纸上谈兵。  通过这两周的实训和学习，我知道了此次实训的目的，也清楚目前自己的不足，那就是缺乏相应的知识与经验，对所学的专业知识不能够很好地运用于实践操作。正所谓“百闻不如一见”，经过这次自身的切身实践，我才深切地理会到了“走出课堂，投身实践”的必要性。课本上所学到的理论知识是为我们的实际操作指明了方向、提供相应的方法，真正的职业技巧是要我们从以后的实际工作中慢慢汲取的。而针对实际操作中遇到的一些特殊的问题,我们不能拘泥于课本,不可纯粹地“以本为本”。  在这次实训中我得到了许多收获，我第一次体会到JAVA工程师的工作是什么样子的，也发现了很多不曾注意到的细节。在实训的演示过程中，在实现项目的过程中尤其要注意的事项就是我们应该从小事做起、从基本做起，把职业能力训练与职业素质有机的结合起来。除此之外，多学习和收集与项目有关的资料、多请教经验丰富的老师，都是不可或缺的一步，这不仅能够很快的培养自己的管理技能及节省时间，避免做无用功，也可以让自己了解很多项目管理上的捷径，从而减少自己会遇到的挫折和错误。我也相信这对我接下来学习专业知识会起到很有效的帮助，在接下来的学习生涯里，我一定会从各个方面严格要求自己，认真严谨。 | | | | | |

基于C/S架构的餐饮管理系统

设计说明书

作者：B20210304115

计算机科学与工程学院

2023年 06 月 15日

**摘要**

随着国家市场经济的快速发展，餐饮业规模不断扩大、数量也不断的增长，手动管理模式在现代社会愈发显得效率低下，并且人工成本不低、提供决策信息方面难以满足现代社会需求，甚至制约了整个餐饮业的规模化发展以及整体服务水平的提升。为此，我们选择设计了此餐饮管理系统。这既可以节省人力资源，也可以提高管理效率以及工作效率，餐饮业可以提升一个新的水平。

本课程设计是采用Java这一程序语言设计一个基于C/S架构的餐饮管理系统，包含对员工、客户、菜品、餐台、点菜以及结账等相应系统管理。除了掌握餐饮管理系统的功能设计，同时还要求对技术方面掌握，如：数据库设计、分析类的方法、Java的gui设计、对数据库系统的增删改查操作等。

根据餐饮系统的流程，完成从用户登录、开台点菜、结账收银，到统计一条线的信息化管理。使用计算机对餐饮企业信息进行管理，具有手动管理所无法比拟的优点。例如:检索迅速、查找方便、可靠性高、存储量大、保密性好、使用寿命长、成本低等。因此本项目的研发内容就是通过开发一套餐饮管理系统，实现餐饮业务的信息化。

**关键词：**java ，C/S架构，餐饮管理系统

目录

[1 引言 1](#_Toc11882504)

[1.1 编写目的 1](#_Toc11882505)

[1.2 参考资料 1](#_Toc11882506)

[2 需求规约 2](#_Toc11882507)

[2.1 功能需求 2](#_Toc11882508)

[2.2 界面需求 3](#_Toc11882509)

[2.3 数据需求 3](#_Toc11882510)

[3 系统设计 4](#_Toc11882511)

[3.1 运行环境 4](#_Toc11882512)

[3.2 系统静态结构设计 4](#_Toc11882513)

[3.3 人机接口设计 5](#_Toc11882514)

[3.4 XXX模块设计说明 7](#_Toc11882515)

[3.5 XXX模块设计说明 7](#_Toc11882516)

[4 数据库设计 8](#_Toc11882517)

[4.1 数据库环境说明 8](#_Toc11882518)

[4.2 数据库的命名规则 8](#_Toc11882519)

[4.3 逻辑结构设计 8](#_Toc11882520)

[4.4 物理结构设计 9](#_Toc11882521)

[4.5 安全性设计 10](#_Toc11882522)

[5 测试用例设计 12](#_Toc11882523)

[附录 14](#_Toc11882524)

[附录1 程序运行结果 14](#_Toc11882525)

[附录2 程序源代码 15](#_Toc11882526)

# 1 引言

## 1.1 编写目的

本文档是“餐饮管理系统”的java系统设计说明书。编写目的是：明确用户对餐饮系统的需求，明确相应软件需求、餐饮系统概要设计以及详细设计和功能测试用例，以的为软件后续更好的开发和维护提供指导。本文档的读者对象是相应的餐饮系统需求分析人员、系统设计人员、系统开发人员、系统测试人员。

## 1.2 参考资料

[1] [王珊](https://baike.baidu.com/item/%E7%8E%8B%E7%8F%8A/7315332?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F%E6%A6%82%E8%AE%BA%EF%BC%88%E7%AC%AC5%E7%89%88%EF%BC%89/_blank)、[萨师煊](https://baike.baidu.com/item/%E8%90%A8%E5%B8%88%E7%85%8A/3100534?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F%E6%A6%82%E8%AE%BA%EF%BC%88%E7%AC%AC5%E7%89%88%EF%BC%89/_blank). 《数据库系统概论（第5版）》.北京.高等教育出版社，2014．

[2] 戴远泉、程宁、胡文杰.《Java高级程序设计实战教程（第2版）》.北京.人民邮电出版社，2022.

[3] 谭义红.《Java面向对象程序设计案例教程》 北京.北京邮电大学出版社.2017

[4] 明日科技.《Java从入门到精通（第5版）》.北京.清华大学出版社.2019

# 2 需求规约

## 2.1 功能需求

（1）员工管理：对员工实现增删改查。

（2）客户管理：对客户实现增删改查。

（3）餐台管理：对餐台实现增删改查。

（4）菜品管理：对菜品分类、菜品实现增删改查。

（5）点菜管理：服务员对某客户、某一空闲餐台实行开台，同时实现点菜，将餐台号与所点的菜品对应起来，分别显示出来，并记录开台时间。

（6）结账管理：收银员对某一餐台通过统计显示消费的菜品清单统计出消费金额，通过手动输入实收金额进行找零的计算，并显示，完成结账的操作，并记录成统计数据。

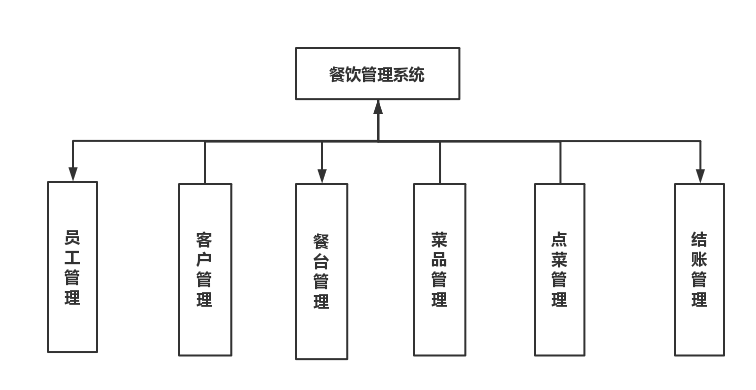


图2.1 餐饮管理系统功能模块图

2.1.1 员工管理用例说明

用例名称：员工管理

用例描述：管理员对该餐厅员工信息进行登记、修改、删除、查询。

前置条件：管理员请求员工管理。

后置条件：管理员对员工进行管理。

活动步骤：

1. 管理员请求对员工进行管理。
2. 系统检查员工的信息。
3. 系统对员工进行修改。
4. 管理员完成员工管理。

2.1.2 客户管理用例说明

用例名称：客户管理

用例描述：管理员对该餐厅客户信息进行登记、修改、删除、查询。

前置条件：管理员请求客户管理。

后置条件：管理员对客户进行管理。

活动步骤：

1. 管理员请求对客户进行管理。
2. 系统检查客户的信息。
3. 系统对客户进行修改。
4. 管理员完成客户管理。

2.1.3 餐台管理用例说明

用例名称：餐台管理

用例描述：管理员对该餐厅餐台信息进行查询或修改状态。

前置条件：管理员请求餐台信息管理。

后置条件：管理员对餐台信息进行管理。

活动步骤：

1. 管理员请求对餐台信息进行管理。
2. 系统检查餐台的信息。
3. 系统对餐台进行状态修改。
4. 管理员完成餐台信息管理。

2.1.4 菜品管理用例说明

用例名称：菜品管理

用例描述：管理员对该餐厅菜品信息进行查询、添加、删除、修改。

前置条件：管理员请求菜品信息管理。

后置条件：管理员对菜品信息进行管理。

活动步骤：

1. 管理员请求对菜品信息进行管理。
2. 系统检查菜品的信息。
3. 系统对菜品进行管理修改。
4. 管理员完成菜品信息管理。

2.1.5 点菜管理用例说明

用例名称：点菜管理

用例描述：管理员对客户进行点单管理。

前置条件：管理员请求点单信息管理。

后置条件：管理员对点单信息进行管理。

活动步骤：

1. 管理员请求对点菜信息进行管理。
2. 系统检查菜品的信息。
3. 系统对菜品采用添加或删除进行点单。
4. 系统完成点单。
5. 管理员完成点单管理。

2.1.6 结账管理用例说明

用例名称：结账管理

用例描述：管理员对系统订单进行结账管理。

前置条件：管理员请求餐台订单信息进行结账管理。

后置条件：管理员对餐台订单进行结账管理。

活动步骤：

1. 管理员请求对订单信息进行结账管理。
2. 系统检查订单的总价格信息。
3. 系统对订单状态进行管理。
4. 系统完成订单结账。
5. 管理员完成订单结账管理。

## 2.2 界面需求

用户对餐饮管理系统要求

1.产品必须处于用户控制之下

界面是用来控制系统的，所有系统开放给用户的功能，都应该被用户良好地控制，而做到这一原则，需要满足以下的几点:

(1)界面操作简洁明了，不强迫用户进入不必要或者不想不愿意进入的动作

(2)提供灵活的交互，例如可以提供多种交互方式供用户选择。

(3)允许用户交互可以被中断和取消，例如在添加信息时，用户可以在中途选择取消添加.

(4)应允许用户和出现在屏幕上的对象进行直接交互

2.可以减少对用户记忆方面的负担

(1)减少对短期记忆的要求。系统在使用过程中，不让用户记忆大量的内容.

(2)有意义的缺省不必要的资料

(3)定义直觉性的捷径。尽量设计用户熟悉或常用的操作方式，让用户更方便快捷的完成与界面的交互。

(4)界面的视觉布局基于真实世界。例如：系统图标尽量反应现实世界的物理或感性信息

(5)以逐一进展的方式进入系统揭示信息。逐步提示用户操作，而不是一次抛出大量的提示信息。

3.保持界面一致性

保持界面一致性可以让用户以自己习惯的方式进行界面操作，减少用户学习和适应的时间。例：软件升级时，需要兼容以前版本的一系列设计和操作。

## 2.3 数据需求

表2.1 用户实体

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| 序号 | 整形 | 10 | 否 | 是 | 序号 |
| 用户名 | 字符串 | 20 | 否 | 否 | 用户名 |
| 密码 | 字符串 | 20 | 否 | 否 | 密码 |

表2.2员工实体

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| 序号 | 整形 | 10 | 否 | 是 | 序号 |
| 用户名 | 字符串 | 20 | 否 | 否 | 用户名 |
| 性别 | 字符串 | 2 | 否 | 是 | 性别 |
| 生日 | 日期时间 | 8 | 是 | 是 | 出生日期 |
| 身份证号 | 字符串 | 18 | 是 | 是 | 身份证号码 |
| 住址 | 字符串 | 40 | 是 | 否 | 家庭住址 |
| 电话 | 字符串 | 11 | 是 | 否 | 电话号码 |
| 职位 | 字符串 | 4 | 否 | 否 | 职位 |
| 是否在职 | 字符串 | 4 | 否 | 否 | 是否在职 |

表2.3客户实体

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| 序号 | 整形 | 10 | 否 | 是 | 序号 |
| 用户名 | 字符串 | 20 | 否 | 否 | 用户名 |
| 性别 | 字符串 | 4 | 是 | 是 | 性别 |
| 公司 | 字符串 | 20 | 是 | 否 | 单位 |
| 电话 | 字符串 | 11 | 是 | 否 | 电话号码 |
| 会员号 | 字符串 | 20 | 否 | 是 | 贵宾卡号 |

表2.4菜品分类实体

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| 序号 | 整形 | 10 | 否 | 是 | 序号 |
| 名称 | 字符串 | 20 | 否 | 否 | 名称 |
| 描述 | 字符串 | 20 | 是 | 否 | 描述 |

表2.5菜品实体

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| 序号 | 整形 | 10 | 否 | 是 | 序号 |
| 菜品名 | 字符串 | 20 | 否 | 否 | 菜品名 |
| 类别编号 | 整形 | 10 | 否 | 是 | 类别编号 |
| 图片 | 二进制 | 50 | 否 | 否 | 图片 |
| 菜品代码 | 字符串 | 8 | 否 | 是 | 菜品代码 |
| 单位 | 字符串 | 4 | 是 | 是 | 菜品单位 |
| 价格 | 字符串 | 6 | 是 | 否 | 价格 |
| 状态 | 字符串 | 4 | 是 | 否 | 状态 |

表2.6餐台实体

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| 序号 | 整形 | 10 | 否 | 是 | 序号 |
| 餐台编号 | 字符串 | 8 | 否 | 是 | 餐台编号 |
| 座位数 | 整形 | 10 | 否 | 否 | 座位数 |
| 状态 | 字符串 | 10 | 否 | 否 | 状态为：已预订、就餐中、已结账 |

表2.7订单信息实体

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| 序号 | 整形 | 10 | 否 | 是 | 序号 |
| 订单编号 | 字符串 | 20 | 否 | 是 | 订单编号 |
| 餐台号 | 整形 | 10 | 否 | 否 | 餐台号 |
| 就餐日期时间 | 日期时间 | 40 | 否 | 否 | 就餐日期时间 |

表2.8订单明细信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| 序号 | 整形 | 10 | 否 | 是 | 序号 |
| 订单编号 | 整形 | 10 | 否 | 是 | 订单编号 |
| 菜品序号 | 整形 | 10 | 否 | 否 | 菜品序号 |
| 菜品数量 | 双精度浮点数 | 4 | 否 | 否 | 菜品数量 |

3 系统设计

首先设计该餐饮管理系统的注册、登录界面，在登录成功后在主界面开始对各个功能设计。

## 3.1 运行环境

设备处理器：AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz

Windows系统版本：Windows 10 家庭中文版

Java代码运行环境：IntelliJ IDEA 2023.1.2 (Community Edition)

JDBC 数据库连接jar包：mysq-connector-j-8.0.33.jar

数据库系统：MySQL

数据库编程工具：Windows CMD命令行

## 3.2 系统静态结构设计

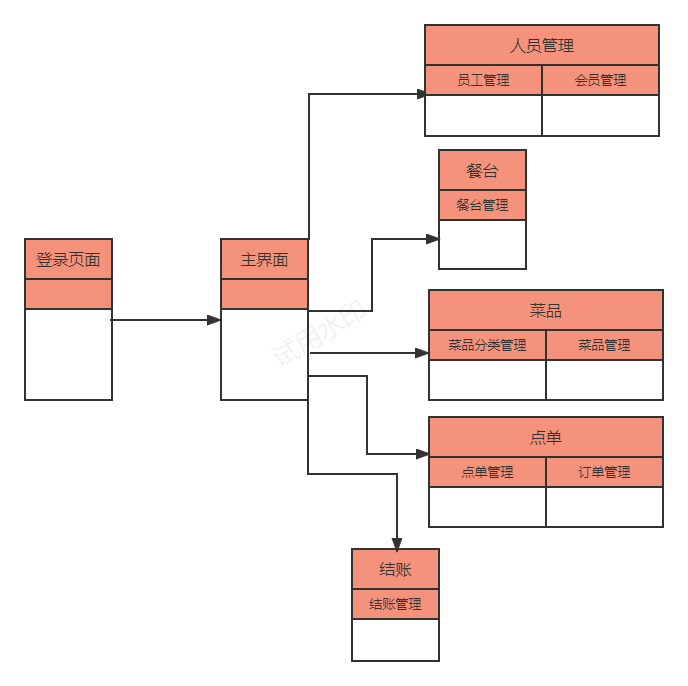


图3.1 航空售票系统总体类图

系统静态结构说明

该餐饮管理系统中的餐厅管理员可以注册登录后进入主界面，再通过主界面可以跳转至不同的界面，在人员的管理界面，管理员可以实行对员工、会员信息的增删改查，在菜品管理界面，管理员可以实现对菜品分类信息以及菜品信息的增删改查，在餐台界面，管理员可以实现对餐台信息的增删改查，在点菜管理界面，管理员可以为顾客提供点菜服务，点菜完成后生成新的订单，并可以查询，在结账管理界面，管理员可以对订单进行结账操作。

## 3.3 人机接口设计

3.3.1 设计应用类

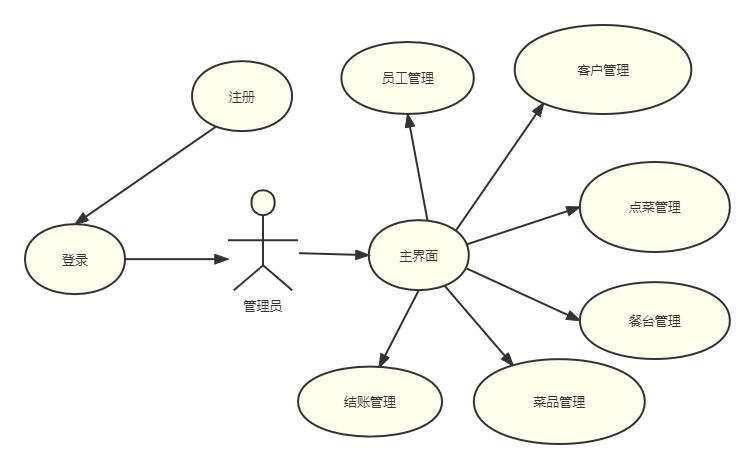


图3.3 餐饮管理系统的设计类图

3.3.2 用户接口1（登录界面）设计

**界面主要控件：**

JTextField，输入框，接收账号和密码。

JButton，按钮，触发登录注册事件。

**实现功能：**

管理员输入账号密码，点击登录按钮跳转餐厅管理界面。



**图3.2 登录界面布局设计**

3.3.3 用户接口2（注册界面）设计

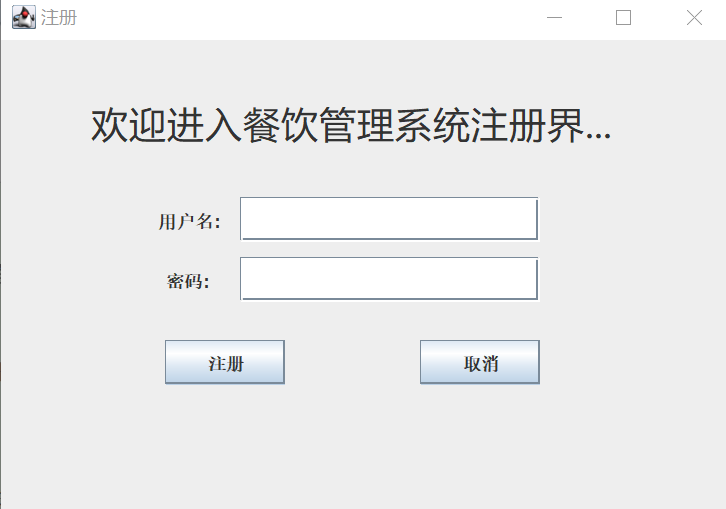
**界面主要控件：**

JTextField，输入框，接收账号和密码。

JButton，按钮，触发点击事件

**实现功能：**

1. 餐厅管理员输入自己的账号密码，点击注册按钮完成注册；
2. 进入登录界面登录系统进入主界面。



**图3.3注册界面布局设计**

## 3.4 注册模块设计说明

（1）功能：该模块是餐饮系统对管理员进行注册操作。

（2）测试要点：

1. 输入的账号或密码为空将提示账号或密码为空。

2. 输入的账号密码错误则提示账号密码有误。

（3）程序流程图如图3.4.1所示：

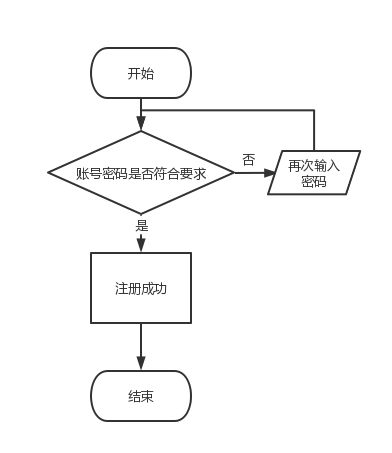


图3.4.1注册模块流程图

## 3.5 登录模块设计说明

（1）功能：该模块主要是管理员对餐饮管理系统进行登录操作。

（2）测试要点：

1. 输入的账号或密码为空将提示账号或密码为空。

2. 输入的账号密码错误则提示账号密码有误。

（3）程序流程图如图3.5.1所示：

## wps

图3.5.1登录模块程序流程图

## 3.6 员工管理模块设计说明

（1）功能：实现管理员对员工的增删改查。

1. 新增员工，程序流程图如图3.6.1所示：

测试要点：

1. 点击新增员工按钮，弹出新增员工窗口。
2. 输入员工的信息，若员工信息无误且完整，则提示新增成功。
3. 若信息不完整，且信息有误，系统会给出提醒。

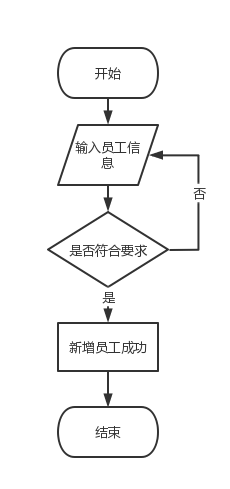


图3.6.1新增员工信息流程图

1. 删除员工，程序流程图如图3.6.2所示：



图3.6.2删除员工流程图

测试要点：

1. 点击表格中员工的信息，点击删除，系统提示删除成功。
2. 修改员工，程序流程图如图3.6.3所示：

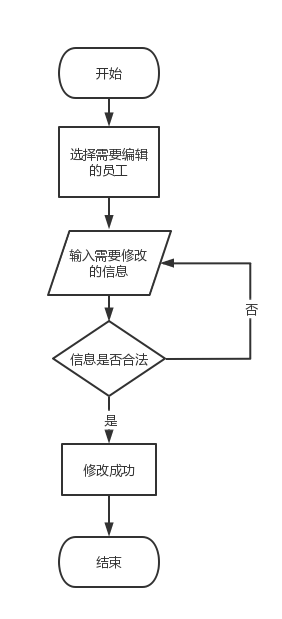


图3.6.3修改员工信息流程图

测试要点：

1. 点击表格中员工的信息，弹出新的界面。
2. 在新的界面中填写要修改的信息，若输入的信息合法则提示修改成功
3. 若输入的信息不合法，则提示信息不合法。
4. 查询员工，程序流程图如图3.6.4所示：

测试要点：

1. 在文本框中输入员工的姓名和id，点击查询按钮。
2. 查询信息将在表格中显示。

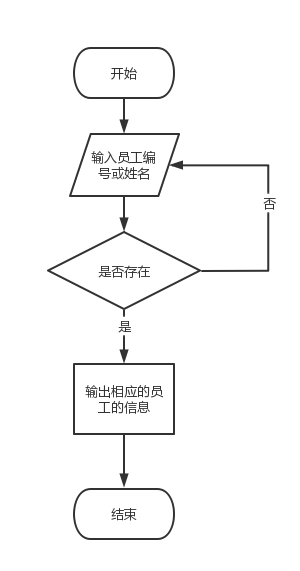


图3.6.4查询员工信息流程图

## 3.7 客户管理模块设计说明

（1）功能：实现管理员对客户的增删改查。

1. 新增客户，程序流程图如图3.7.1所示：

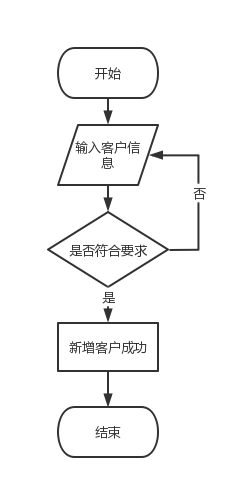


图3.7.1新增客户流程图

测试要点：

1. 点击新增客户按钮，弹出新增客户窗口。
2. 输入客户的信息，若客户信息无误且完整，则提示新增成功。
3. 若信息不完整，且信息有误，系统会给出提醒。
4. 删除客户，程序流程图如图3.7.2所示：



图3.7.2删除客户流程图

测试要点：

1. 点击表格中客户的信息，点击删除，系统提示删除成功。
2. 修改客户，程序流程图如图3.7.3所示：

测试要点：

1. 点击表格中客户的信息，弹出新的界面。
2. 在新的界面中填写要修改的信息，若输入的信息合法则提示修改成功。
3. 若输入的信息不合法，则提示信息不合法。
4. 查询客户，程序流程图如图3.7.4所示：

测试要点：

1. 在文本框中输入客户的姓名和id，点击查询按钮。
2. 查询信息将在表格中显示。

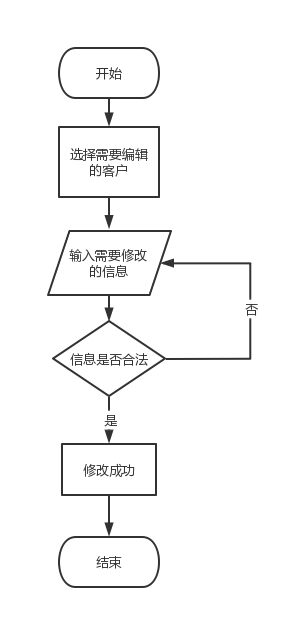
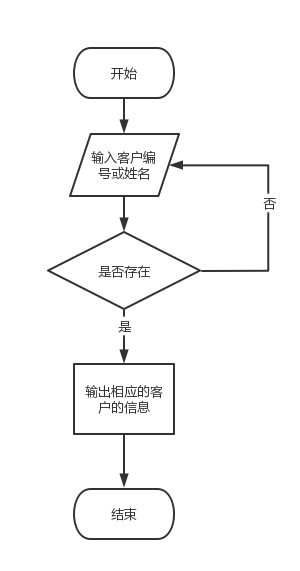
 

图3.7.3，3.7.4修改客户流程图示：

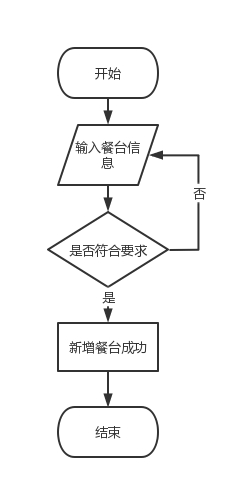


图3.7.5新增餐台流程图

测试要点：

1. 点击新增餐台按钮，弹出新增餐台界面。
2. 输入餐台的座位数，若输入为数字且合法，则新增餐台成功，否则提示新增失败。
3. 删除餐台，程序流程图如图3.7.6所示：



图3.7.6删除餐台流程图

测试要点：

1. 点击餐台信息，点击删除按钮，系统提示删除成功
2. 修改餐台，程序流程图如图3.7.8所示：

测试要点：

1. 点击表格中餐台的信息，弹出新的界面。
2. 在新的界面中填写要修改的信息，若输入的信息合法则提示修改成功。
3. 若输入的信息不合法，则提示信息不合法。
4. 查询餐台，程序流程图如图3.7.9所示：

测试要点：

1. 在文本框中输入餐台号，点击查询按钮。
2. 查询信息将在表格中显示。

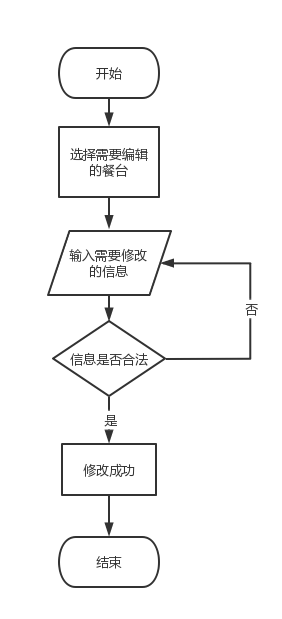
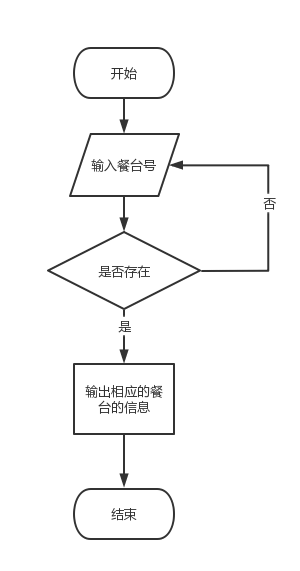
 

图3.7.8修改餐台流程图 图3.7.9查询餐台流程图

## 3.8 菜品管理模块设计说明

（1）功能：实现管理员对菜品信息的增删改查

1. 新增菜品，程序流程图如图3.8.1所示：

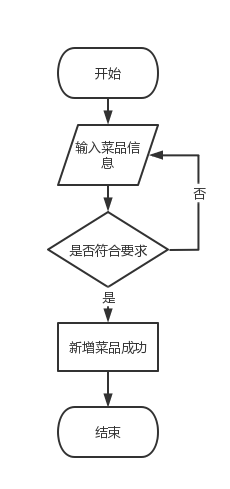


图3.8.1新增菜品程序流程图

测试要点：

1. 点击新增菜品按钮，弹出新增菜品窗口。
2. 输入菜品的信息，若菜品信息无误且完整，则提示新增成功。
3. 若信息不完整，且信息有误，系统会给出提醒。
4. 删除菜品，程序流程图如图3.8.2所示：



图3.8.2删除菜品流程图

测试要点：

1. 点击菜品信息，点击删除按钮，系统提示删除成功。
2. 修改菜品，程序流程图如图3.8.3所示：

测试要点：

1. 点击表格中餐台的信息，弹出新的界面。
2. 在新的界面中填写要修改的信息，若输入的信息合法则提示修改成功。
3. 若输入的信息不合法，则提示信息不合法。
4. 查询菜品，程序流程图如图3.8.4所示：

测试要点：

1. 在文本框中输入菜品名或菜品id，点击查询按钮。
2. 查询信息将在表格中显示。

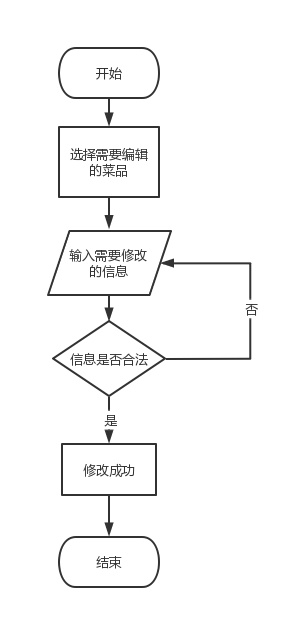
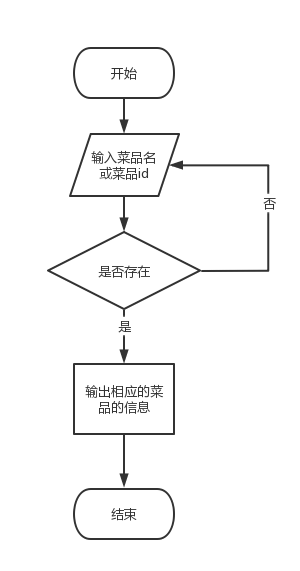
 

图3.8.3删除菜品程序流程图 图3.8.4查询菜品程序流程图

## 3.9 点单管理模块设计说明

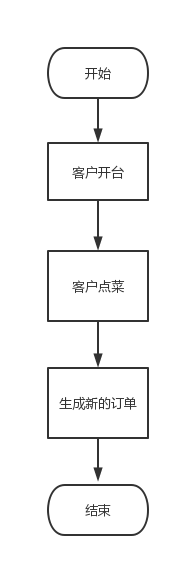
（1）功能：实现管理员对客户进行开台，点单操作。

（2）测试要点：

1. 输入客户的客户号以及人数，为客户开台。

2. 客户选菜，生成新的订单。

（3）程序流程图如图3.9.1所示：



3.9.1点菜管理程序流程图

## 3.10 结账管理模块设计说明

（1）功能：实现管理员对订单进行结账管理。

（2）测试要点：

1. 在输入框中输入订单编号或客户编号，查询的订单将在表格中显示。

2. 点击订单，跳转至结账界面，将会根据客户的身份对客户进行优惠。

3. 在新的界面输入金额，找零

（3）程序流程图如图3.10所示：

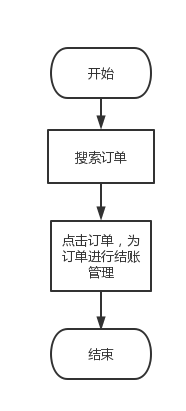


图3.10订单管理流程图

# 4 数据库设计

## 4.1 数据库环境说明

数据库系统采用MySQL，MySQL是一个快速的、多线程、多用户和健壮的SQL数据库服务器。编程工具采用Windows CMD命令行，相对来说更加灵活、高效，不仅可以省去可视化工具的安装和配置，而且利于脚本化管理和调度。

## 4.2 数据库的命名规则

1、采用26个英文字母(区分大小写)和0-9的自然数加上下划线'\_'组成，命名简洁明确，多个单词用下划线'\_'分隔,一个项目一个数据库，多个项目慎用同一个数据库

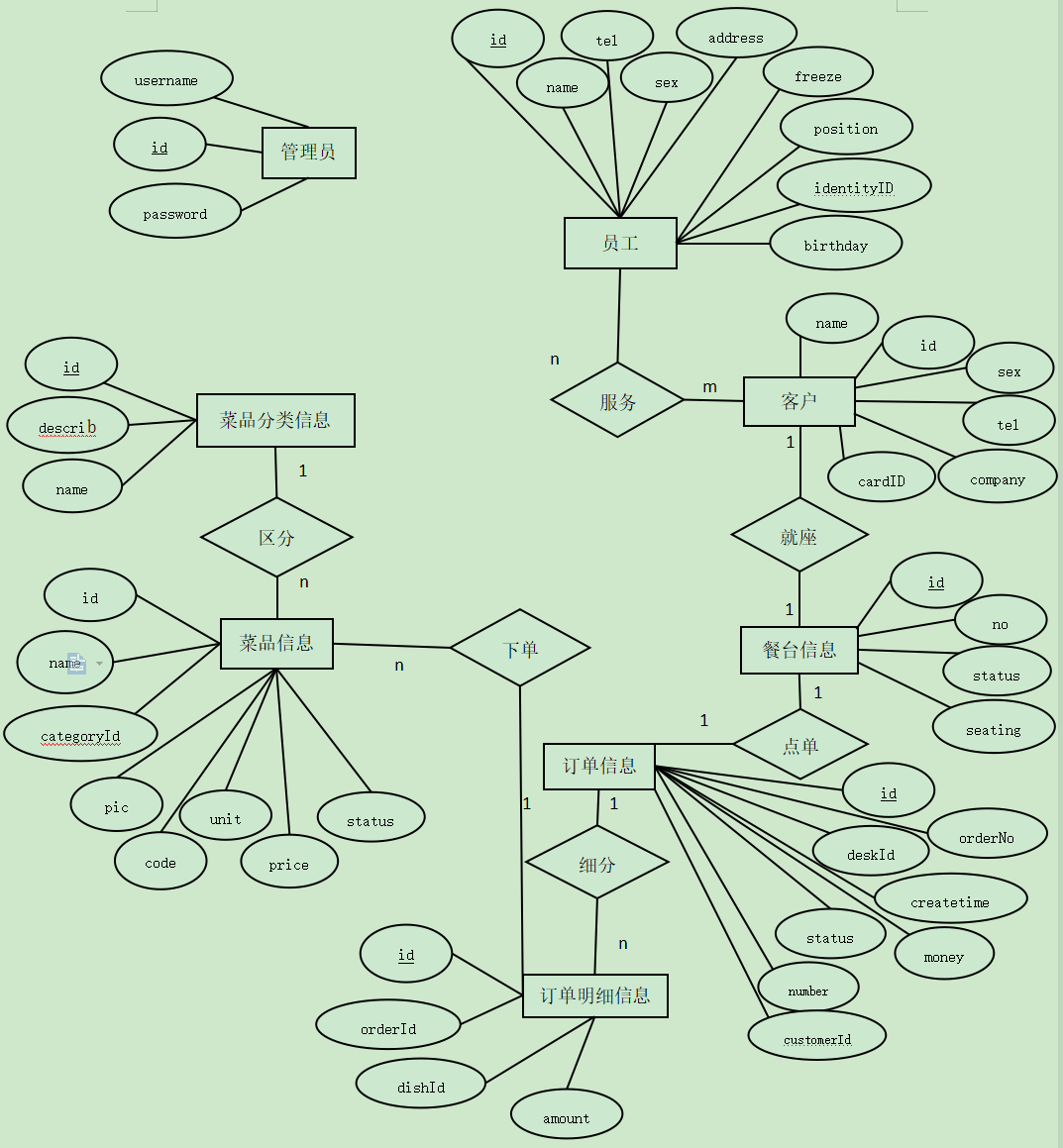
2、禁止使用数据库关键字，如：name，time ，datetime，password等

3、表名称不宜过长（一般最多使用三个英文单词）

4、多用单数形式表示名称，例：使用 employee，而不是 employees

## 4.3 逻辑结构设计

E-R图



关系模型

管理员（id，username，password）

员工（id，name，sex，tel，address，freeze，position，identityID，birthday）

客户（id，name，sex，tel，company，cardID）

餐台信息（id，no，status，seating）

订单信息（id，orderNo，deskId，createtime，money，status，number，customerId）

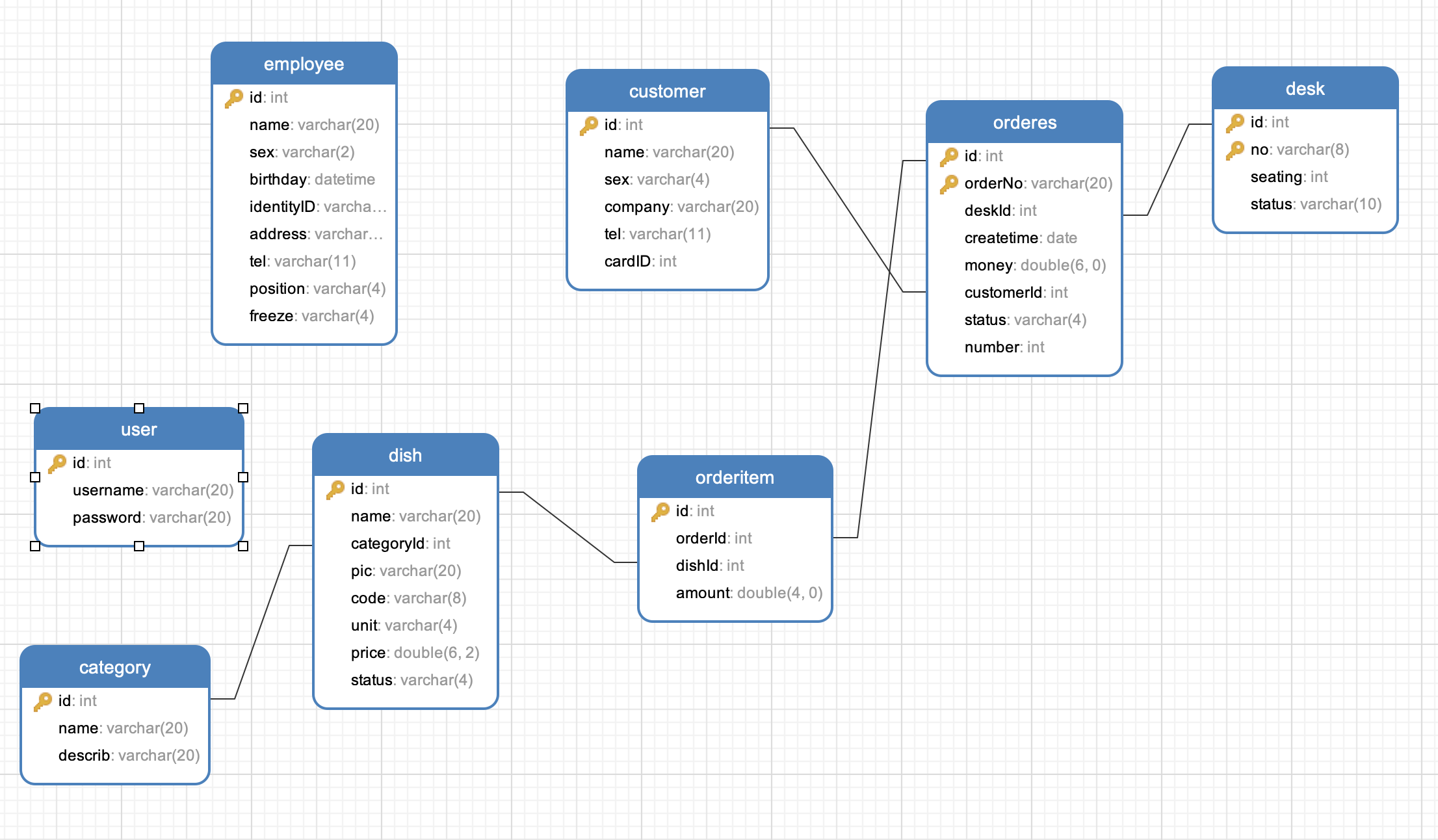
订单明细信息（id，orderId，dishId，amount）

菜品信息（id，name，categoryId，pic，code，unit，price，status）

菜品分类信息（id，describ，name）

范式分析：属于BCNF

逻辑模型图：



## 4.4 物理结构设计

4.4.1 数据库表

表4.1 表汇总

|  |  |
| --- | --- |
| 表 名 | 功 能 说 明 |
| user | 存储管理员信息的数据 |
| employee | 存储员工信息的数据 |
| category | 存储菜品分类信息的数据 |
| dish | 存储菜品信息的数据 |
| desk | 存储餐台信息的数据 |
| customer | 存储顾客信息的数据 |
| dishorder | 存储订单信息的数据 |
| orderitem | 存储订单明细信息的数据 |

表4.2 user表

|  |  |
| --- | --- |
| 表 名 | User |
| 数据库用户 | id、username、password |
| 主 键 | id |
| 其他排序字段 | 无 |
| 索引字段 | id |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 数据类型（精确范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 约束条件/说明 |
| 1 | id | Int（10） | N | Y | Primary key | 序号 |
| 2 | username | Varchar（20） | N | N | 无 | 用户名 |
| 3 | password | Varchar（20） | N | N | 无 | 密码 |

表4.3 employee表

|  |  |
| --- | --- |
| 表 名 | employee |
| 数据库用户 | id、name、sex、birthday、identityID、address、tel、position、freeze |
| 主 键 | id |
| 其他排序字段 | 无 |
| 索引字段 | id |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 约束条件/说明 |
| 1 | id | Int(10) | N | Y | Primary key | 序号 |
| 2 | name | Varchar(20) | N | N | 无 | 用户名 |
| 3 | sex | Varchar(2) | N | Y | 无 | 性别 |
| 4 | birthday | Datetime | Y | Y | 无 | 出生日期 |
| 5 | identityID | Varchar（18） | Y | Y | 无 | 身份证号码 |
| 6 | address | Varchar（40） | Y | N | 无 | 家庭住址 |
| 7 | tel | Varchar（11） | Y | N | 无 | 电话 |
| 8 | position | Varchar（4） | N | N | 无 | 职位 |
| 9 | freeze | Varchar（4） | N | N | 无 | 是否在职 |

表4.4 customer表

|  |  |
| --- | --- |
| 表 名 | customer |
| 数据库用户 | id、name、sex、company、tel、cardID |
| 主 键 | id |
| 其他排序字段 | 无 |
| 索引字段 | id |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 数据类型（精确范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 约束条件/说明 |
| 1 | id | Int（10） | N | Y | Primary key | 序号 |
| 2 | name | Varchar（20） | N | N | 无 | 用户名 |
| 3 | sex | Varchar（4） | Y | Y | 无 | 性别 |
| 4 | company | Varchar(20) | Y | N |  | 单位 |
| 5 | tel | Varchar(11) | Y | N |  | 电话 |
| 6 | cardID | Varchar(10) | N | Y |  | 贵宾卡号 |

表4.5 category表

|  |  |
| --- | --- |
| 表 名 | category |
| 数据库用户 | id、name、describ |
| 主 键 | id |
| 其他排序字段 | 无 |
| 索引字段 | id |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 数据类型（精确范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 约束条件/说明 |
| 1 | id | Int（10） | N | Y | Primary key | 序号 |
| 2 | name | Varchar（20） | N | N | 无 | 名称 |
| 3 | describ | Varchar（20） | Y | N | 无 | 描述 |

表4.6 dish表

|  |  |
| --- | --- |
| 表 名 | dish |
| 数据库用户 | id、name、categoryId、pic、code、unit、price、status |
| 主 键 | id |
| 其他排序字段 | categoryId |
| 索引字段 | id |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 约束条件/说明 |
| 1 | id | Int(10) | N | Y | Primary key | 序号 |
| 2 | name | Varchar(20) | N | N | 无 | 菜品名 |
| 3 | categoryId | Int(10) | N | Y | Foreign key | 类别编号 |
| 4 | pic | Blob(50) | N | N | 无 | 图片 |
| 5 | code | Varchar（8） | N | Y | 无 | 菜品代码 |
| 6 | unit | Varchar（4） | Y | Y | 无 | 单位 |
| 7 | price | Datetime（6） | Y | N | 无 | 价格 |
| 8 | status | Varchar（4） | Y | N | 无 | 状态 |

表4.7 desk表

|  |  |
| --- | --- |
| 表 名 | desk |
| 数据库用户 | id、no、seating、status |
| 主 键 | id |
| 其他排序字段 | no |
| 索引字段 | id |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 数据类型（精确范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 约束条件/说明 |
| 1 | id | Int（10） | N | Y | Primary key | 序号 |
| 2 | no | Varchar（8） | N | Y | Foreign key | 餐台编号 |
| 3 | seating | Int(4） | N | N | 无 | 座位数 |
| 4 | status | Varchar(10) | N | N | 无 | 状态为：已预订、就餐中、已结账 |

表4.8 order表

|  |  |
| --- | --- |
| 表 名 | order |
| 数据库用户 | id、orderNo、deskId、createtime、money、customerId、status、number |
| 主 键 | id |
| 其他排序字段 | orderNo、deskId |
| 索引字段 | id |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 约束条件/说明 |
| 1 | id | Int(10) | N | Y | Primary key | 序号 |
| 2 | orderNo | Varchar(20) | N | Y | Foreign key | 订单编号 |
| 3 | deskId | Int(10) | N | N | Foreign key | 餐台号 |
| 4 | createtime | Datetime | N | N | 无 | 就餐日期时间 |

表4.9 orderitem

|  |  |
| --- | --- |
| 表 名 | orderitem |
| 数据库用户 | id、orderId、dishId、amount |
| 主 键 | id |
| 其他排序字段 | orderId、dishId |
| 索引字段 | id |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名称 | 数据类型（精确范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 约束条件/说明 |
| 1 | id | Int（10） | N | Y | Primary key | 序号 |
| 2 | orderId | Int(10） | N | N | Foreign key | 订单编号 |
| 3 | dishId | Int(10） | N | N | Foreign key | 菜品序号 |
| 4 | amount | Double(4) | N | N | 无 | 菜品数量 |

## 4.5 安全性设计

4.5.1 防止用户直接操作数据库的方法

1.给用户限定数据库访问的权限:可以通过在数据库中设置用户权限的方式，限定用户对数据库的访问权限，实现防止用户直接操作数据库的目的。可以为用户设置只读权限，这样用户就只能查询数据库，而不能对数据库进行增、删、改等操作。也可以设置只能访问部分数据表，这样可以避免用户直接操作整个数据库。

2、写存储过程，通过使用存储过程访问数据库数据，然后对存储过程设置权限

3、为使用该餐饮管理系统的用户设立分组，在数据库中对用户进行权限设置。

4.5.2 用户帐号密码的加密方法

防止用户直接操作数据库的方法。用户只能用帐号登录到系统，然后再通过餐饮系统对数据库进行访问，无法通过其它途径登录数据库对其进行修改。

用户帐号密码加密的方法。对登录用户的帐号密码进行加密处理，确保在任何地方都不会出现密码的明文。

# 5 测试用例设计

# 5 测试用例设计

表5.1 登录-注册测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 登录 | yl-001 | 输入‘用户名’和‘密码’ | 小王，123456 | 登录成功 | 通过 |
| yl-002 | 小吴，123456 | 登录成功 | 通过 |
| yl-003 | 小邓，123456 | 登录成功 | 通过 |
| yl-004 | 小蛋黄，123456 | 登录成功 | 通过 |
| yl-005 | 小蛋黄，123123 | 登录成功 | 通过 |
| yl-006 | 2942990081,一个空格 | 登录失败 | 通过 |
| yl-007 | 一个空格，1237236182 | 登录失败 | 通过 |
| yl-008 | 2942990081，密码长度超过32 | 登录失败 | 通过 |
| yl-009 | 用户名超过36，1237236182 | 登录失败 | 通过 |
| 注册 | yl-010 | 用户名，密码 | 11,1111 | 创建成功 | 通过 |
| y1-011 | 用户名，密码 | 88888888888888,1 | 创建失败 | 通过 |

表5.2 员工管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 增加员工 | y2-001 | 点击添加按  钮，输入‘姓  名’，‘性别’，‘出生日期’，‘地址’，‘电话号码’ | 小王，男，1234567,  1988-12-07 00:00:00,湖南长沙，112345 | 新增成功 | 通过 |
| 删除员工 | y2-002 | 点击删除按钮 | 小王 | 删除成功 | 通过 |
| 修改员工 | y2-003 | 点击表格 | 小王->小刘 | 修改成功 | 通过 |
| 查询员工 | y2-004 | 输入员工任一属性，点击查询  按钮 | 4 | 查询成功 | 通过 |

表5.3 客户管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 增加客户 | y3-001 | 点击添加按  钮，输入‘姓  名’，‘性别’，‘公司’，‘电话号码’，‘会员号’ | 张三，女，清华大学，156789,1000001 | 新增成功 | 通过 |
| 删除客户 | y3-002 | 点击删除按钮 | 张三 | 删除成功 | 通过 |
| 修改客户 | y3-003 | 点击表格 | 女->男 | 修改成功 | 通过 |
| 查询客户 | y3-004 | 输入员工任一属性，点击查询  按钮 | 1000001 | 查询成功 | 通过 |

表5.4 菜品管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 增加菜品 | y4-001 | 点击添加按  钮，输入‘菜品编号’，‘菜品名’，‘菜品分类编号’，‘图片’，‘代码’，‘单位’，‘价格’，‘状态’ | 1，菲力牛排，1，无，flnp，份，45.00，在售 | 新增成功 | 通过 |
| 删除菜品 | y4-002 | 点击删除按钮 | 1 | 删除成功 | 通过 |
| 修改菜品 | y4-003 | 点击表格 | 45.00->40.00 | 修改成功 | 通过 |
| 查询菜品 | y4-004 | 输入菜品任一属性，点击查询  按钮 | 1 | 查询成功 | 通过 |

表5.5 餐台管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 增加餐台 | y5-001 | 点击添加按  钮，输入‘座位数’ | 10 | 新增成功 | 通过 |
| 删除餐台 | y5-002 | 点击删除按钮 | T220623148 | 删除成功 | 通过 |
| 修改餐台 | y5-003 | 点击表格 | 10->20 | 修改成功 | 通过 |
| 查询餐台 | y5-004 | 输入菜品编  号或菜名，点击查询  按钮 | T220623148 | 查询成功 | 通过 |

表5.6 点菜管理测试用例

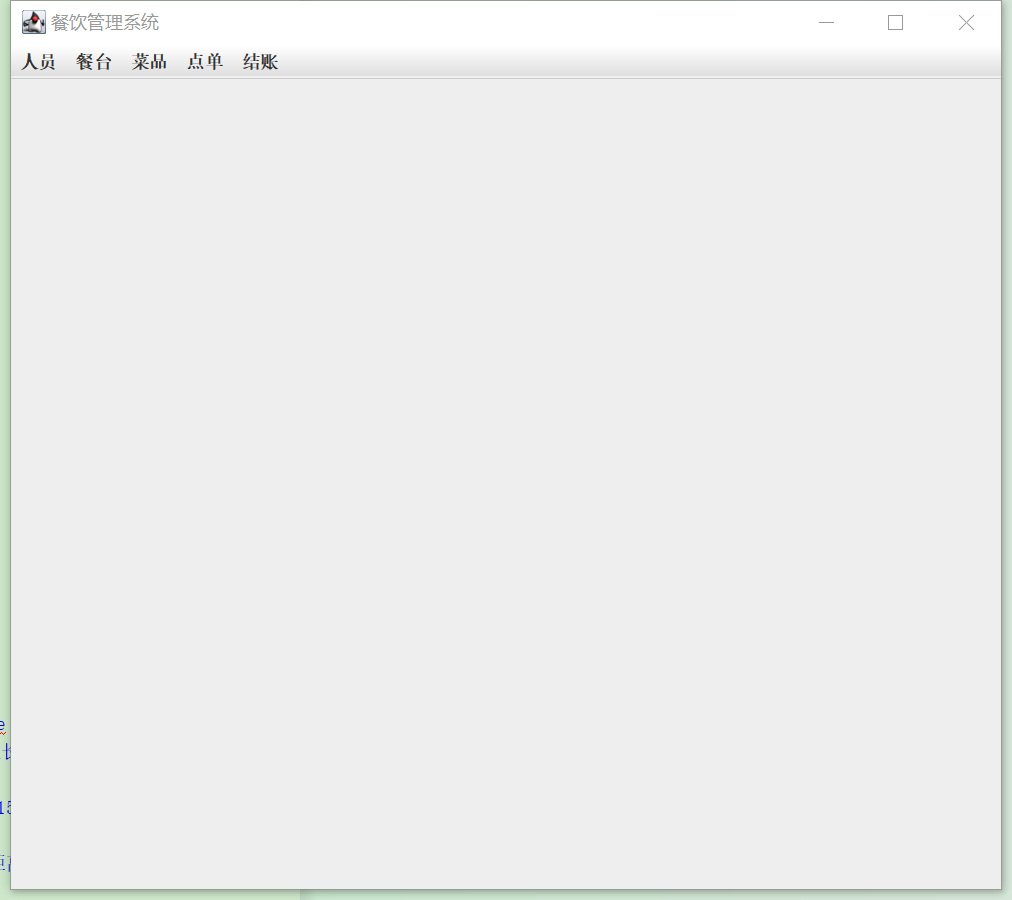
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 开台 | y6-001 | 输入客户id和人数 | 1，5 | 开台成功 | 通过 |
| 生成订单 | y6-002 | 点菜后点击保存 | 无 | 操作成功 | 通过 |

表5.7 结账管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 删除订单 | y7-001 | 点击订单后点击删除按钮 | Z220626767 | 删除成功 | 通过 |
| 结账操作 | Y7-002 | 点击订单后输入金额 | 500 | 支付成功 | 通过 |

# 附录

## 附录1 程序运行结果

图1.主界面

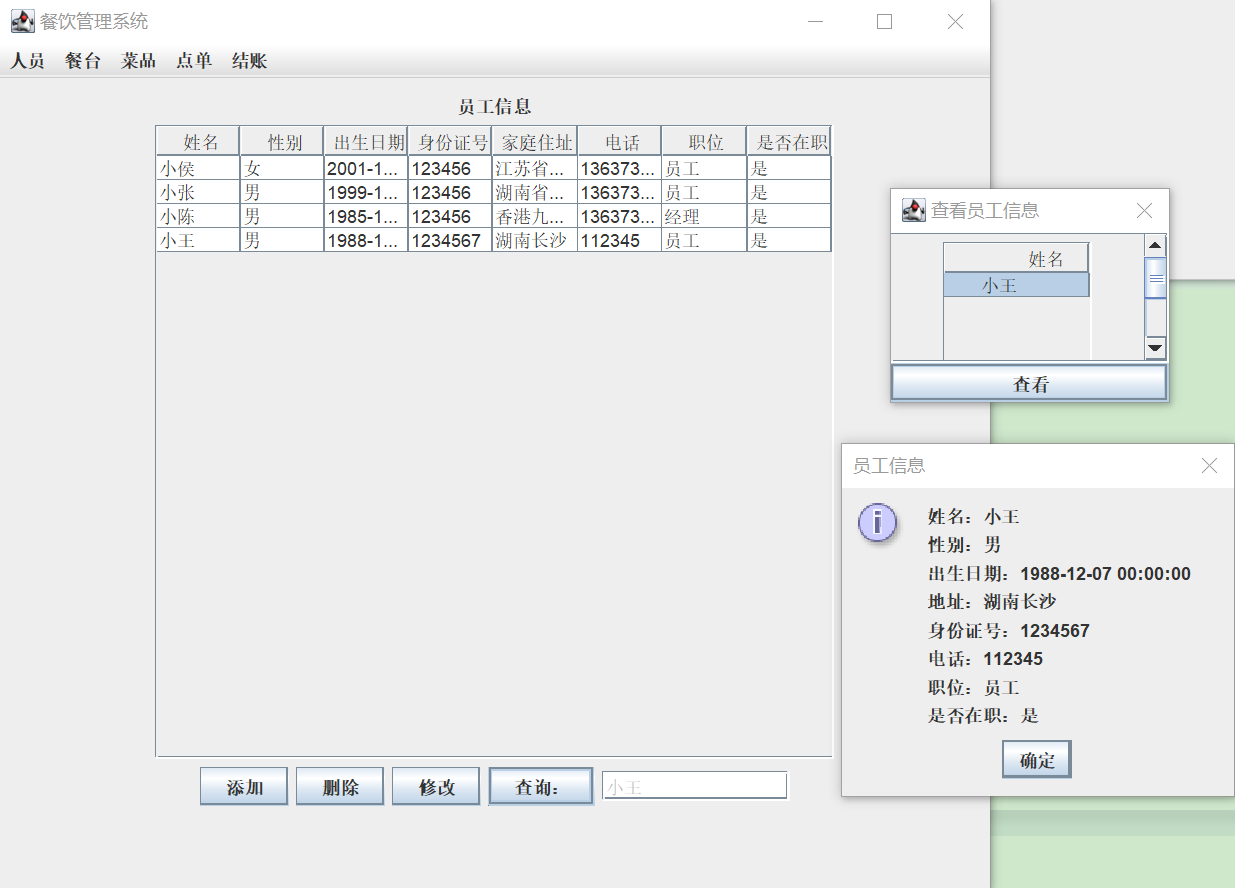
图2.示例查询界面

图3.结账界面

## 

## 附录2 程序源代码

注：源代码以文件为单位附列在此。

1、CheckoutFrame.java

package retuarants.catering.frame;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class CheckoutFrame extends JFrame implements ActionListener {

private JTextField tableNumberField;

private JTextField totalAmountField;

private JTextField receivedAmountField;

private JTextField changeAmountField;

private JButton calculateButton;

private JButton checkoutButton;

public CheckoutFrame() {

setTitle("结账管理");

setSize(300, 200);

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

setLocationRelativeTo(null);

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new GridLayout(5, 2, 10, 10));

JLabel tableNumberLabel = new JLabel("餐台号");

tableNumberField = new JTextField();

JLabel totalAmountLabel = new JLabel("消费金额");

totalAmountField = new JTextField();

JLabel receivedAmountLabel = new JLabel("实收金额");

receivedAmountField = new JTextField();

JLabel changeAmountLabel = new JLabel("找零金额");

changeAmountField = new JTextField();

calculateButton = new JButton("计算");

checkoutButton = new JButton("结账");

mainPanel.add(tableNumberLabel);

mainPanel.add(tableNumberField);

mainPanel.add(totalAmountLabel);

mainPanel.add(totalAmountField);

mainPanel.add(receivedAmountLabel);

mainPanel.add(receivedAmountField);

mainPanel.add(changeAmountLabel);

mainPanel.add(changeAmountField);

mainPanel.add(calculateButton);

mainPanel.add(checkoutButton);

calculateButton.addActionListener(this);

checkoutButton.addActionListener(this);

setContentPane(mainPanel);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == calculateButton) {

calculateChangeAmount();

} else if (e.getSource() == checkoutButton) {

performCheckout();

}

}

private void calculateChangeAmount() {

double totalAmount = Double.parseDouble(totalAmountField.getText());

double receivedAmount = Double.parseDouble(receivedAmountField.getText());

double changeAmount = receivedAmount - totalAmount;

changeAmountField.setText(String.valueOf(changeAmount));

}

private void performCheckout() {

// 结账操作的逻辑

String tableNumber = tableNumberField.getText();

double totalAmount = Double.parseDouble(totalAmountField.getText());

double receivedAmount = Double.parseDouble(receivedAmountField.getText());

double changeAmount = Double.parseDouble(changeAmountField.getText());

// 执行结账操作

// ...

// 显示结账结果

JOptionPane.showMessageDialog(this, "结账成功");

// 关闭当前窗口

dispose();

}

public static void main(String[] args) {

new CheckoutFrame().setVisible(true);

}

}

2、EmployeeManagePane2.java

package retuarants.catering.pane;

import retuarants.category.dialog.EmployeeAddDialog;

import retuarants.category.dialog.EmployeeAlterDialog;

import retuarants.category.dialog.empLookTableDialog;

import retuarants.catering.dao.IBaseDao;

import retuarants.catering.entity.Employee;

import retuarants.catering.util.ChangeEmployeeEvent;

import retuarants.catering.util.EmployeeDaoFactory;

import retuarants.catering.util.EmployeeTableModel;

import retuarants.catering.util.JDBConnection;

import javax.swing.\*;

import javax.swing.table.DefaultTableColumnModel;

import java.awt.\*;

import java.awt.event.FocusEvent;

import java.awt.event.FocusListener;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class EmployeeManagePane2 extends JPanel {

private JPanel panelTop = null;

private JLabel labHeader = null;

private JPanel panelBottom = null;

private JButton add = null, delete = null,alter=null,lookup = null;

private JTextField lookupto;

private JScrollPane scroll = null;

private JTable table = null;

private EmployeeTableModel model = null;

private List listEmployee = null;

public EmployeeTableModel getModel() {

if (null == model) {

model = new EmployeeTableModel(listEmployee);//修改table中的值

model.addTableModelListener(new ChangeEmployeeEvent(model));

return model;

}

return model;

}

public JLabel getLabHeader() {

if (null == labHeader) {

labHeader = new JLabel("员工信息");

return labHeader;

}

return labHeader;

}

public JTable getTable() {

if (null == table) {

table = new JTable(getModel());

table.setEnabled(true);

table.setRowSelectionAllowed(true);

DefaultTableColumnModel dcm = (DefaultTableColumnModel) table.getColumnModel();

dcm.getColumn(0).setMinWidth(0);

dcm.getColumn(0).setMaxWidth(0);

return table;

}

return table;

}

public JScrollPane getPanTable() {

if (null == scroll) {

scroll = new JScrollPane();

scroll.setViewportView(getTable());

return scroll;

}

return scroll;

}

public JPanel getPanelTop() {

if (null == panelTop) {

panelTop = new JPanel();

panelTop.setLayout(new FlowLayout(FlowLayout.CENTER));

panelTop.add(getLabHeader());

return panelTop;

}

return panelTop;

}

public JPanel getPanelBottom() {

if (null == panelBottom) {

panelBottom = new JPanel();

panelBottom.setLayout(new FlowLayout(FlowLayout.CENTER));

panelBottom.add(getAdd());

panelBottom.add(getDelete());

panelBottom.add(getalter());

panelBottom.add(lookup());

panelBottom.add(lookupto());

return panelBottom;

}

return panelBottom;

}

public JButton getAdd() {

if (null == add) {

add = new JButton("添加");

add.addActionListener(e -> new EmployeeAddDialog());

return add;

}

return add;

}

public JButton getDelete() {

if (null == delete) {

delete = new JButton("删除");

delete.addActionListener(e -> {

int[] rows = getTable().getSelectedRows();

if (rows.length > 0) {

int flag = JOptionPane.showConfirmDialog(null, "确定删除?");

if (flag == JOptionPane.YES\_OPTION)

deleteEmployee();

} else

JOptionPane.showMessageDialog(null, "请选择要删除的行！");

});

return delete;

}

return delete;

}

public JButton getalter() {

if (null == alter) {

alter = new JButton("修改");

alter.addActionListener(e -> {

int selectedRow = table.getSelectedRow();

if (selectedRow != -1) {

new EmployeeAlterDialog(getModel().getRow(selectedRow));//传去所选的emp的id

}

else {

JOptionPane.showMessageDialog(null, "请选择要修改的数据行！");

}

});return alter;

}

return alter;

}

public JButton lookup() {

if (null == lookup) {

lookup = new JButton("查询：");

lookup.addActionListener(e -> {

Connection conn = JDBConnection.getConn();

PreparedStatement ps ;

ResultSet rs ;

String sql="select \* FROM employee where name LIKE ? OR sex LIKE ? OR birthday LIKE ? OR identityID LIKE ? OR address LIKE ? OR tel LIKE ? OR position LIKE ? OR freeze LIKE ?";

try {

ps = conn.prepareStatement(sql);

ps.setString(1,"%"+lookupto.getText());

ps.setString(2, "%"+lookupto.getText());

ps.setString(3, "%"+lookupto.getText());

ps.setString(4, "%"+lookupto.getText());

ps.setString(5, "%"+lookupto.getText());

ps.setString(6, "%"+lookupto.getText());

ps.setString(7, "%"+lookupto.getText());

ps.setString(8, "%"+lookupto.getText());

rs = ps.executeQuery();

ArrayList<Employee> list=new ArrayList<>();

if (rs.next()) do {

Employee empl = new Employee();

empl.setId(rs.getInt(1));

empl.setName(rs.getString(2));//性名

empl.setSex(rs.getString(3));//性别

empl.setBirthday(rs.getString(4));//生日

empl.setIdentityID(rs.getString(5));//身份证号

empl.setAddress(rs.getString(6));//家庭住址

empl.setTel(rs.getString(7));//电话

empl.setPosition(rs.getString(8));//职位

empl.setFreeze(rs.getString(9));//是否在职x

list.add(empl);

} while (rs.next());

new empLookTableDialog(list);

}

catch (SQLException ex) {

JOptionPane.showMessageDialog(null, "请输入正确信息");

throw new RuntimeException(ex);

}

});

return lookup;

}

return lookup;

}//查询

public JTextField lookupto(){

if(null==lookupto){

lookupto=new JTextField("请输入想要查询的信息");

lookupto.setForeground(Color.lightGray);

lookupto.selectAll();

lookupto.addFocusListener(new FocusListener() {

@Override

public void focusGained(FocusEvent e) {

lookupto.setForeground(Color.BLACK);

lookupto.setText("");

}

@Override

public void focusLost(FocusEvent e) {

lookupto.setForeground(Color.lightGray);

}

});

return lookupto;

}

return lookupto;

}//查询文本框

public void deleteEmployee() {

int[] rows = getTable().getSelectedRows();

ArrayList<Employee> list = new ArrayList<>();

IBaseDao dao = EmployeeDaoFactory.getDao();

for (int i = rows.length - 1; i >= 0; i--) {

list.add(getModel().getRow(rows[i]));

getModel().deleteRow(rows[i]);

}

dao.deleteList(list);

getTable().updateUI();

list.clear();

}

public void initData() {

listEmployee = new ArrayList();

IBaseDao dao = EmployeeDaoFactory.getDao();

listEmployee = dao.getList();

}

public EmployeeManagePane2() {

initData();

this.setLayout(new BorderLayout());

add(getPanelTop(), BorderLayout.NORTH);

add(getPanelBottom(), BorderLayout.SOUTH);

add(getPanTable(), BorderLayout.CENTER);

this.setSize(new Dimension(675,600));

}

public static void main(String[] args) {

new EmployeeManagePane2();

}

}

3、EmployeeDaoImpl.java

package retuarants.catering.dao;

import retuarants.catering.entity.Employee;

import retuarants.catering.util.JDBConnection;

import javax.swing.\*;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class EmployeeDaoImpl implements IBaseDao{

ResultSet rs = null;

@Override

public List getList() {

// TODO Auto-generated method stub

Connection conn = JDBConnection.getConn();

String s1 = "select \* from employee order by id asc";

List<Employee> list = new ArrayList<>();

try {

rs = conn.prepareStatement(s1).executeQuery();

} catch (SQLException e) {

System.out.println("取出全部数据出错！");

JOptionPane.showMessageDialog(null, "取出全部数据出错!");

e.printStackTrace();

}

try {

if (rs.next()) do {

Employee empl = new Employee();

empl.setId(rs.getInt(1));

empl.setName(rs.getString(2));

empl.setSex(rs.getString(3));

empl.setBirthday(rs.getString(4));

empl.setIdentityID(rs.getString(5));

empl.setAddress(rs.getString(6));

empl.setTel(rs.getString(7));

empl.setPosition(rs.getString(8));

empl.setFreeze(rs.getString(9));

list.add(empl);

} while (rs.next());

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "取出全部数据出错!");

e.printStackTrace();

} finally {

try {

rs.close();

conn.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "关闭数据连接时出错!");

e.printStackTrace();

}

}

return list;

}//将emplpyee表中所有数据放到list里面

public void saveEmployee(Employee emo) {

// TODO Auto-generated method stub

String s1;

Connection conn = JDBConnection.getConn();

PreparedStatement ps = null;

try {

s1 = "insert into Employee(name,sex,birthday,identityID,address,tel,position,freeze) values(?,?,?,?,?,?,?,?)";

ps = conn.prepareStatement(s1);

ps.setString(1, emo.getName());

ps.setString(2, emo.getSex());

ps.setString(3, emo.getBirthday());

ps.setString(4, emo.getIdentityID());

ps.setString(5, emo.getAddress());

ps.setString(6, emo.getTel());

ps.setString(7, emo.getPosition());

ps.setString(8, emo.getFreeze());

ps.executeUpdate();

} catch (SQLException e) {

System.out.println("添加数据时出错！");

JOptionPane.showMessageDialog(null, "添加数据时出错!");

e.printStackTrace();

} finally {

try {

assert ps != null;

ps.close();

conn.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "关闭数据连接时出错!");

e.printStackTrace();

}

}

}//将emo对象中的所有数据存在数据库中mployee表的属性中去

@Override

public void saveList(List list) {

// TODO Auto-generated method stub

String s1;

Connection conn = JDBConnection.getConn();

PreparedStatement ps = null;

try {

for (Object o : list) {

Employee empl = (Employee) o;

String name = empl.getName();

String sex = empl.getSex();

String birthday = empl.getBirthday();

String identityID = empl.getIdentityID();

String address = empl.getAddress();

String tel = empl.getTel();

String position = empl.getPosition();

String freeze = empl.getFreeze();

s1 = "insert into Employee(name,sex,birthday,identityID,address,tel,position,freeze) values(?,?,?,?,?,?,?,?)";

ps = conn.prepareStatement(s1);

ps.setString(1, name);

ps.setString(2, sex);

ps.setString(3, birthday);

ps.setString(4, identityID);

ps.setString(5, address);

ps.setString(6, tel);

ps.setString(7, position);

ps.setString(8, freeze);

ps.executeUpdate();

}

} catch (SQLException e) {

System.out.println("添加数据时出错！");

JOptionPane.showMessageDialog(null, "添加数据时出错!");

e.printStackTrace();

} finally {

try {

assert ps != null;

ps.close();

conn.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "关闭数据连接时出错!");

e.printStackTrace();

}

}

}//将list中的的emp都存入数据库

@Override

public void deleteList(List list) {

// TODO Auto-generated method stub

String str = "delete from employee where id=?";

int id;

Iterator it = list.iterator();

Connection conn = JDBConnection.getConn();

PreparedStatement ps = null;

try {

while (it.hasNext()) {

id = ((Employee) it.next()).getId();

ps = conn.prepareStatement(str);

ps.setInt(1, id);

ps.executeUpdate();

}

} catch (SQLException ignored) {

} finally {

try {

assert ps != null;

ps.close();

conn.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "关闭数据连接时出错!");

e.printStackTrace();

}

}

}

@Override

public void update(List list) {

// TODO Auto-generated method stub

Connection conn = JDBConnection.getConn();

PreparedStatement ps = null;

int id;

Iterator it = list.iterator();

try{

while(it.hasNext()){

Employee empl =(Employee)it.next();

id = ((Employee) it.next()).getId();

String name = empl.getName();

String sex=empl.getSex();

String birthday=empl.getBirthday();

String identityID=empl.getIdentityID();

String address=empl.getAddress();

String tel=empl.getTel();

String position=empl.getPosition();

String freeze=empl.getFreeze();

String str = "update employee set name=?,sex=?,birthday=?,identityID=?,address=?,tel=?,position=?,freeze=? where id=?";

ps = conn.prepareStatement(str);

ps.setString(1, name);

ps.setString(2, sex);

ps.setString(3, birthday);

ps.setString(4, identityID);

ps.setString(5, address);

ps.setString(6, tel);

ps.setString(7, position);

ps.setString(8, freeze);

ps.setInt(9, id);

ps.executeUpdate();

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "修改数据时出错!");

e.printStackTrace();

} finally {

try {

assert ps != null;

ps.close();

conn.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(null, "关闭数据连接时出错!");

e.printStackTrace();

}

}

}

}

1. EmployeeTableModel.java

break;

case 4:

empl.setIdentityID(value);

break;

case 5:

empl.setAddress(value);

break;

case 6:

empl.setTel(value);

break;

case 7:

empl.setPosition(value);

break;

case 8:

empl.setFreeze(value);

}

fireTableDataChanged();

}

public boolean isCellEditable(int row, int column) {

return true;

}

/\*\*

\* setValueAt方法是使增加或修改值的时候生效,aValue就是你在单元格填的值, 要把这些值保存到数据源中

\*/

public void setValueAt(Object aValue, int rowIndex, int columnIndex) {

Employee empl = (Employee) list.get(rowIndex);

setPropertyValueByCol(empl, aValue.toString(), columnIndex);

this.fireTableCellUpdated(rowIndex, columnIndex);

}

public static List getChangeList() {

return changeList;

}

public static void setChangeList(List changeList) {

EmployeeTableModel.changeList = changeList;

}

}