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** | **唐睿智** |
| **学号** | **B20200103213** |
| **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 | 唐睿智 | 学号 | B20200103213 | 班级 | 21软件01班 |
| 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**：（说明实训各阶段的工作过程，包括需求分析、系统设计、编码实现、系统测试、交付实施等阶段的主要工作。）  需求分析：6月5日，根据餐饮行业的特点和该企业的实际情况，本餐饮管理系统以餐饮业务为基础，突出管理，从专业角度出发，提供科学有效的管理模式。  系统设计：6月6日，根据具体情况，系统主要功能设计分为六个部分：员工管理，客户管理，餐台管理，菜品管理，点菜管理，结账管理。  编码实现：6月7日-14日，编码实现设计好的系统模块，编写数据库连接代码，编写每个部分的界面以及功能代码  系统测试：6月15日，对已编写好的系统进行用例测试  交付实施：6月16日 | | | | | |
| **实训成果**：  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。 | | | | | |
| **实训总结**：（包括心得体会、存在的问题和改进方向。）  点餐系统的架构有一定难度。虽然实现了任务书要求的功能，但还有很多自己想实现的因时间因素没有完成，例如把管理者使用的管理接口、服务员使用的开台接口以及客户使用的点餐接口区分开，还有多线程点餐等，还可以进一步优化。 | | | | | |

基于C/S架构的餐饮管理系统

设计说明书

作者：唐睿智（B20200103213）

计算机科学与工程学院

2023年 06 月 15日

**摘要**

该课程设计是基于C/S架构的餐饮管理系统，使用了java语言，运用了mysql数据库，系统分为了员工、客户、菜品、餐桌、点菜、结账六个模块进行管理。

**关键词：**java ，C/S架构，餐饮管理系统

目录

[已知技术参数和设计要求： 2](#_Toc138082695)

[1. 问题描述（功能要求）： 2](#_Toc138082696)

[2. 运行环境要求： 2](#_Toc138082697)

[3. 技术要求： 2](#_Toc138082698)

[实训工作量： 3](#_Toc138082699)

[工作计划： 3](#_Toc138082700)

[1 引言 1](#_Toc138082701)

[1.1 编写目的 1](#_Toc138082702)

[1.2 参考资料 1](#_Toc138082703)

[2 需求规约 2](#_Toc138082704)

[2.1 功能需求 2](#_Toc138082705)

[2.2 界面需求 5](#_Toc138082706)

[2.3 数据需求 5](#_Toc138082707)

[3 系统设计 9](#_Toc138082708)

[3.1 运行环境 9](#_Toc138082709)

[3.2 系统静态结构设计 9](#_Toc138082710)

[3.3 人机接口设计 10](#_Toc138082711)

[3.4 员工模块设计说明 21](#_Toc138082712)

[3.5 客户模块设计说明 35](#_Toc138082713)

[3.6 菜品模块设计说明 49](#_Toc138082714)

[3.7 餐桌模块设计说明 78](#_Toc138082715)

[3.8 开台模块设计说明 93](#_Toc138082716)

[3.9 点菜及结账模块设计说明 107](#_Toc138082717)

[4 数据库设计 128](#_Toc138082718)

[4.1 数据库环境说明 128](#_Toc138082719)

[4.2 数据库的命名规则 128](#_Toc138082720)

[4.3 逻辑结构设计 128](#_Toc138082721)

[4.4 物理结构设计 130](#_Toc138082722)

[4.5 安全性设计 130](#_Toc138082723)

[5 测试用例设计 131](#_Toc138082724)

[附录 136](#_Toc138082725)

[附录1 程序运行结果 136](#_Toc138082726)

[附录2 程序源代码 141](#_Toc138082727)

# 1 引言

## 1.1 编写目的

本文档是“基于C/S架构的餐饮管理系统”的软件设计说明书。编写目的是：明确软件需求、概要设计、详细设计和功能测试用例，为软件后续开发和维护提供指导。本文档的读者对象是需求分析人员、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] 马石安、魏文平.面向对象程序设计教程[M].北京：清华大学出版社.2007.8

[2] 杨厚群.数据结构[M]. 上海：上海交通大学出版社.2013.1

[3] 杨路明.C语言程序设计教程[M]. 北京：北京邮电大学出版社.2013.1

# 2 需求规约

## 2.1 功能需求

餐饮管理系统可大致分为：员工管理，客户管理，餐台管理，点菜管理，菜品管理，结账管理六大模块。分别实现了员工、客户的增、删、改、查，餐台的增加删除，点菜的加菜减菜，菜品种类的增加减少，结账的找零功能

根据餐饮行业的特点和该企业的实际情况，本餐饮管理系统以餐饮业务为基础，突出管理，从专业角度出发，提供科学有效的管理模式。

能够针对中餐为其菜品的多样化和特色化的服务提供标准化的管理。

能够提供符合餐饮企业自身要求的较科学化的标准化、流程化管理，解决餐饮行业专业人才欠缺的问题。

能够依据订餐、点菜、结账等环节的繁重化及复杂化问题，实现强化管理、降低成本、堵漏节流等效益。

能够针对企业的经营现状做出科学的分析，使得企业对市场的应变能力得到提高。

本餐饮管理系统采用Java语言进行开发，JDK采用16.0.2，开发工具使用Eclipse，数据库使用MySQL 5.6。

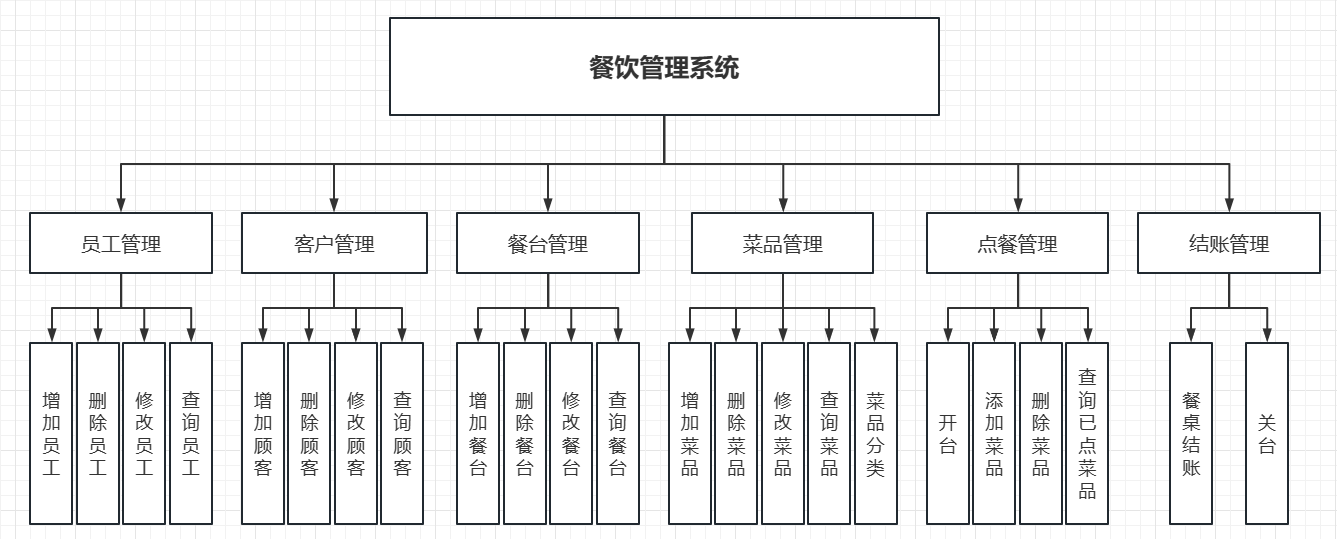


图2.1 餐饮管理系统功能模块图

2.1.1 员工管理用例说明

用例名称：员工管理

用例描述：对员工信息实现增删改查

前置条件：删除或修改信息必须是已存在的员工

后置条件：无

活动步骤：

1. 选择要进行的增删改查操作。
2. 系统检查员工的状态。
3. 系统进行相应操作。

扩展点：

无

异常处理：

1. 如果删除或修改的员工信息不存在，则用例终止。

2.1.2 客户管理用例说明

用例名称：客户管理

用例描述：对客户实现增删改查

前置条件：删除或修改信息必须是已存在的客户

后置条件：无

活动步骤：

1. 选择要进行的增删改查操作。
2. 系统检查客户的状态。
3. 系统进行相关操作。

扩展：

无

异常处理：

1. 如果要删除或者修改的客户信息不存在，则用例终止。

2.1.3 餐台管理用例说明

用例名称：餐台管理

用例描述：对餐台实现增删改查

前置条件：删除或修改信息必须是已存在的餐台

后置条件：无

活动步骤：

1. 选择要进行的增删改查操作。
2. 系统检查餐台的状态。
3. 系统进行相关操作。

扩展：

无

异常处理：

1. 如果要删除或者修改的餐台信息不存在，则用例终止。

2.1.4 菜品管理用例说明

用例名称：菜品管理

用例描述：对菜品分类、实现增删改查

前置条件：删除或修改信息必须是已存在的菜品类别或者菜品信息。

后置条件：无

活动步骤：

1. 选择要进行的增删改查操作。
2. 系统检查菜品类别、菜品的状态。
3. 系统进行相关操作。

扩展：

无

异常处理：

1. 如果要删除或者修改的菜品类别不存在，则用例终止。
2. 如果要删除或者修改的菜品信息不存在，则用例终止。

2.1.5 点菜用例说明

用例名称：点菜用例。

用例描述：对点菜用例的操作说明

前置条件：该餐桌已经开台且该订单还未完成

后置条件：无

活动步骤：

1. 管理员选择一个就餐中的餐桌。
2. 进行添加菜品或者删除菜品操作。
3. 系统检查信息，并对该订单进行相关操作。

扩展：

菜品信息、已点菜品

异常处理：

1. 如果该餐桌未开台或订单已完成，则用例终止。

2.1.6 结账管理用例说明

用例名称：结账管理

用例描述：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，通过手动输入优惠金额、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。

前置条件：

1. 该餐桌已经开台且该订单还未完成

后置条件：修改餐桌状态为空闲、订单状态为已完成。

活动步骤：

1. 管理员选择一个就餐中的餐桌。
2. 进行结账操作。
3. 找零，生成数据。

扩展：

无

异常处理：

1. 如果该餐桌未开台或订单已完成，则用例终止。

## 2.2 界面需求

## 2.3 数据需求

表2.1 管理员信息表user

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否允许为空 | 是否为主键 | 说明 |
| id | int | 10 | no | yes | 序号 |
| username | varchar | 20 | no | no | 用户名 |
| password | varchar | 20 | no | no | 密码 |

表2.2 员工信息表employee

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否允许为空 | 是否为主键 | 说明 |
| id | int | 10 | no | yes | 序号 |
| name | varchar | 20 | no | no | 用户名 |
| sex | varchar | 2 | yes | no | 性别 |
| identityID | varchar | 18 | yes | no | 身份证号 |
| address | varchar | 40 | yes | no | 家庭住址 |
| tel | varchar | 11 | yes | no | 电话 |
| position | varchar | 4 | no | no | 职位 |
| freeze | varchar | 4 | no | no | 是否在职 |

表2.3 客户信息表customer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否允许为空 | 是否为主键 | 说明 |
| id | int | 10 | no | yes | 序号 |
| name | varchar | 20 | no | no | 用户名 |
| sex | varchar | 4 | yes | no | 性别 |
| company | varchar | 20 | yes | no | 单位 |
| tel | varchar | 11 | yes | no | 电话 |
| cardID | varchar | 10 | no | no | 贵宾卡号 |

表2.4 菜品分类表category

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否允许为空 | 是否为主键 | 说明 |
| id | int | 10 | no | yes | 序号 |
| name | varchar | 20 | no | no | 名称 |
| describ | varchar | 20 | yes | no | 描述 |

表2.5 菜品信息表dish

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否允许为空 | 是否为主键 | 说明 |
| id | int | 10 | no | yes | 序号 |
| name | varchar | 20 | no | no | 菜品名 |
| category | int | 10 | no | no | 类别编号 |
| pic | blob | 50 | no | no | 图片 |
| code | varchar | 8 | no | no | 菜品代码 |
| unit | varchar | 4 | yes | no | 单位 |
| price | double | 6 | yes | no | 价格 |
| status | varchar | 4 | yes | no | 状态 |

表2.6 餐台信息表desk

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否允许为空 | 是否为主键 | 说明 |
| id | int | 10 | no | yes | 序号 |
| no | varchar | 8 | no | no | 餐台编号 |
| seating | int | 4 | no | no | 座位数 |
| status | varchar | 10 | no | no | 状态 |

表2.7 订单信息表order

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否允许为空 | 是否为主键 | 说明 |
| id | int | 10 | no | yes | 序号 |
| orderNo | varchar | 20 | no | yes | 订单编号 |
| deskID | int | 10 | no | no | 餐台号 |
| createtime | date | 40 | no | no | 就餐时间 |
| money | double | 6 | no | no | 金额 |
| customerID | int | 10 | no | no | 客户编号 |
| status | varchar | 4 | no | no | 状态 |
| number | int | 4 | no | no | 就餐人数 |

表2.8 订单明细表orderitem

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 是否允许为空 | 是否为主键 | 说明 |
| id | int | 10 | no | yes | 序号 |
| orderID | int | 10 | no | no | 订单编号 |
| dishID | int | 10 | no | no | 菜品编号 |
| amount | double | 4 | no | no | 菜品数量 |

# 3 系统设计

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：服务员对某一餐台点菜，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，系统自动统计数据，并显示找零完成结账的操作。

## 3.1 运行环境

JDK采用16.0.2，开发工具使用Eclipse，数据库使用MySQL 5.6。

## 3.2 系统静态结构设计

图3.1 点餐系统总体类图

系统静态结构说明

先登录进入系统主界面，选择员工、客户、餐桌、菜品、开台和点菜管理；

进入菜品管理有两个选择：菜品分类管理和菜品信息，再进行相关操作；

进入业务管理有两个选择：点菜和开台管理，进行相关操作；

点菜管理中可以进入结账管理，进行结账操作。

## 3.3 人机接口设计

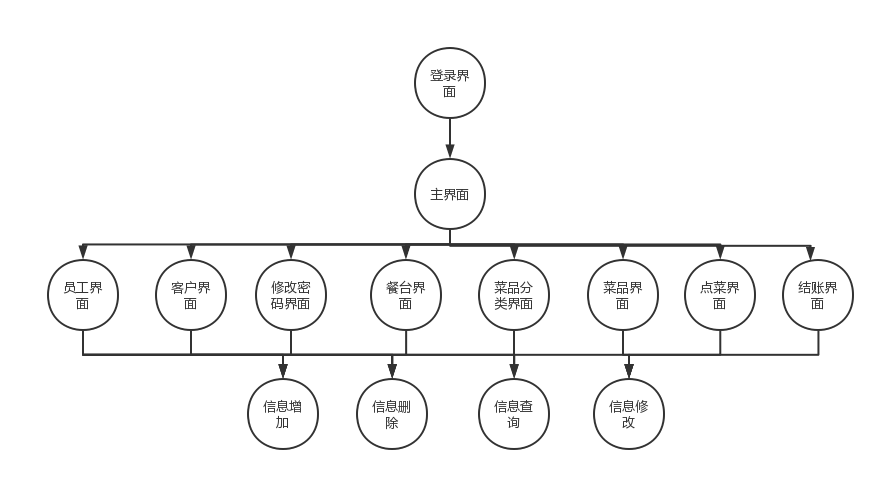
3.3.1 设计应用类

图3.2 餐饮管理的设计类图

3.3.2 登陆界面设计

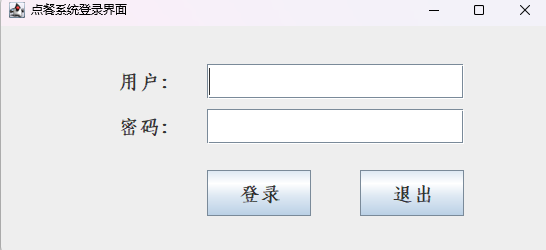


图3.3登陆界面

登陆界面包括用户、密码的输入，登陆和退出两个按钮，可以实现登陆操作。为使用者操作方便，输入完用户名后按Enter或者TAB可以切换至密码输入，密码输入完后按Enter即可进行登录操作。

3.3.3 主界面设计

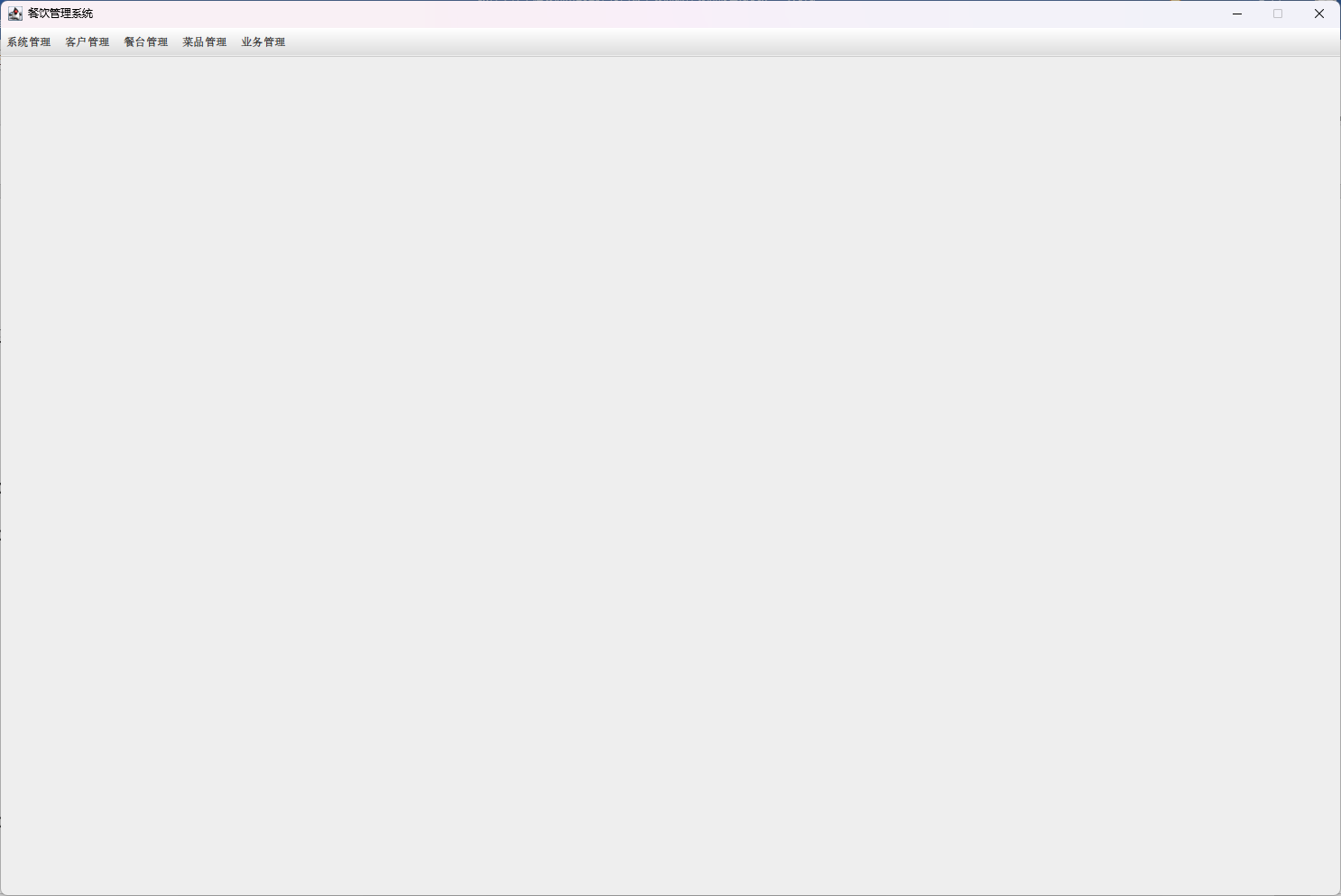


图3.4 系统主界面

主界面包括一行菜单栏，选择菜单栏中的组件可以在主界面的JPanel上进行切换不同界面的操作。

3.3.4 员工信息管理设计

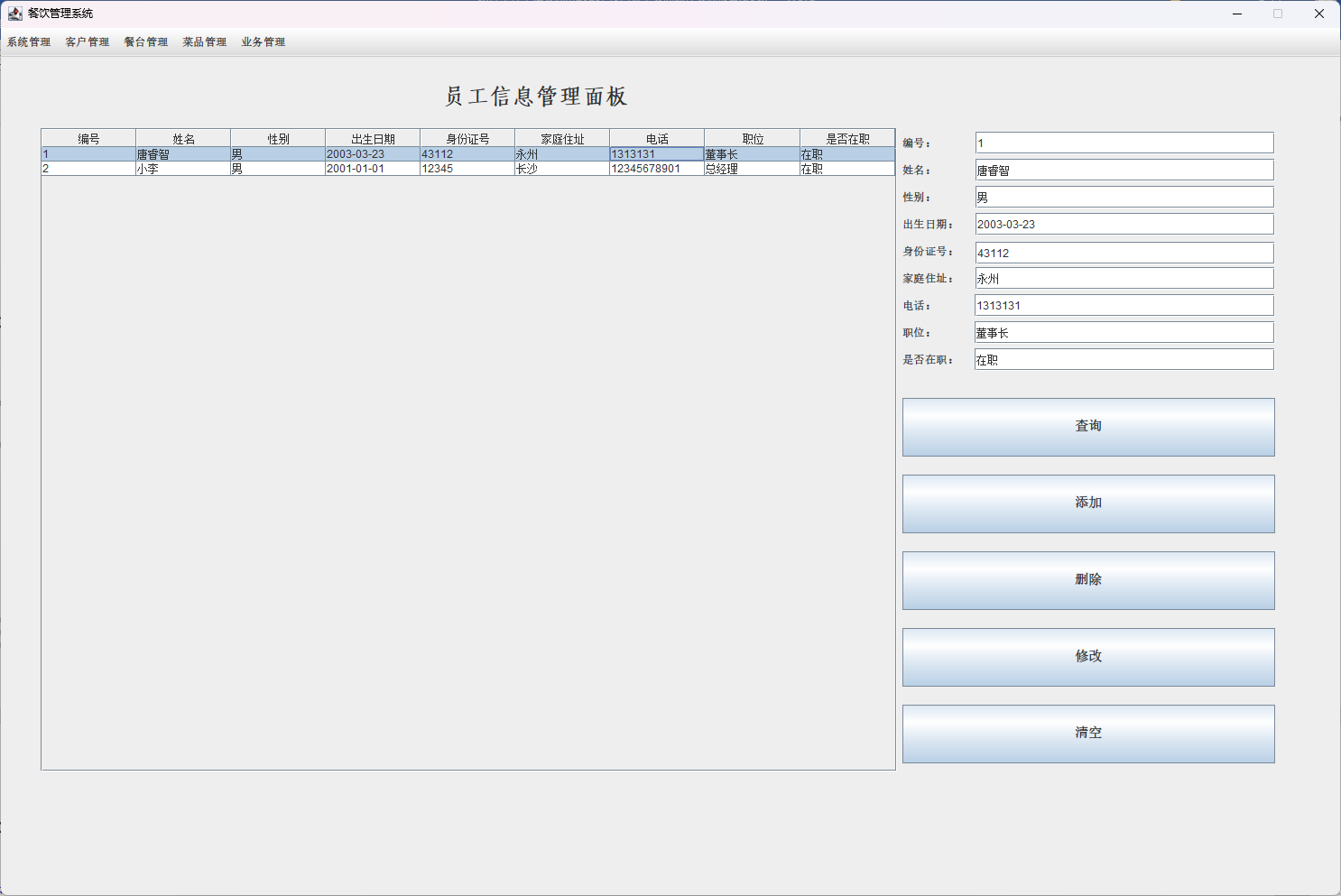


图3.5员工信息管理

员工信息界面包括员工信息表以及表中选中的某一行数据的详细信息，增删改查四个按钮分别进行对员工信息的相应操作。清空按钮进行清空右边数据栏的内容，不会改变员工信息。

3.3.5 客户信息管理设计

图3.6 客户信息管理

客户信息界面包括客户信息表以及表中选中的某一行数据的详细信息，增删改查四个按钮分别进行对客户信息的相应操作。清空按钮进行清空右边数据栏的内容，不会改变客户信息。

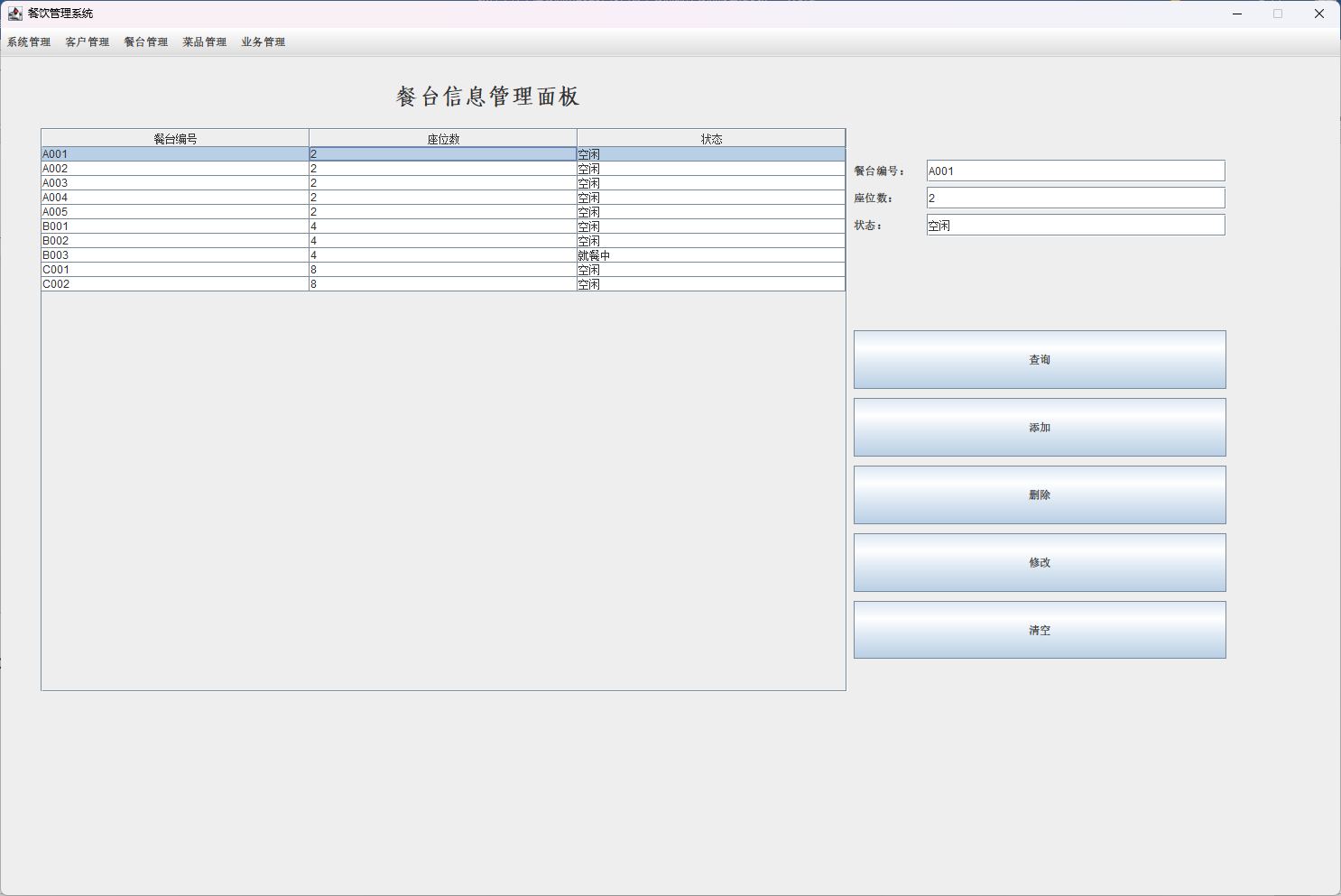
3.3.6餐台信息管理设计

图3.7 餐台信息管理

餐台信息界面包括餐台信息表以及表中选中的某一行数据的详细信息，增删改查四个按钮分别进行对餐台信息的相应操作。清空按钮进行清空右边数据栏的内容，不会改变餐台信息。

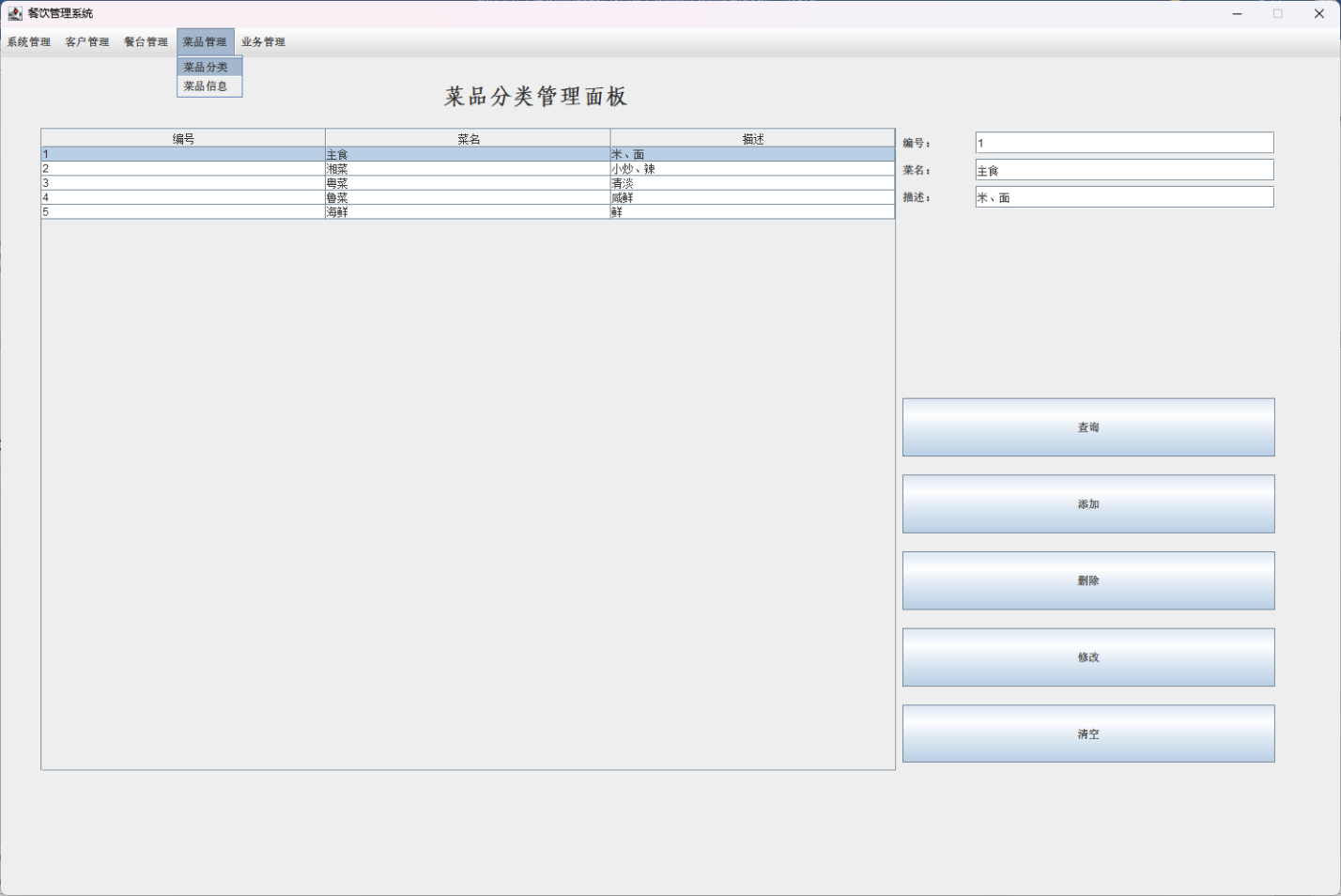
3.3.7 菜品分类信息管理设计

图3.8 菜品分类信息管理

菜品分类信息界面包括菜品分类信息表以及表中选中的某一行数据的详细信息，增删改查四个按钮分别进行对菜品分类信息的相应操作，清空按钮进行清空右边数据栏的内容，不会改变菜品分类信息。

3.3.8 菜品信息管理设计

图3.9菜品信息管理

菜品信息界面包括菜品信息表以及表中选中的某一行数据的详细信息，增删改查四个按钮分别进行对菜品信息的相应操作。清空按钮进行清空右边数据栏的内容，不会改变菜品分类信息。

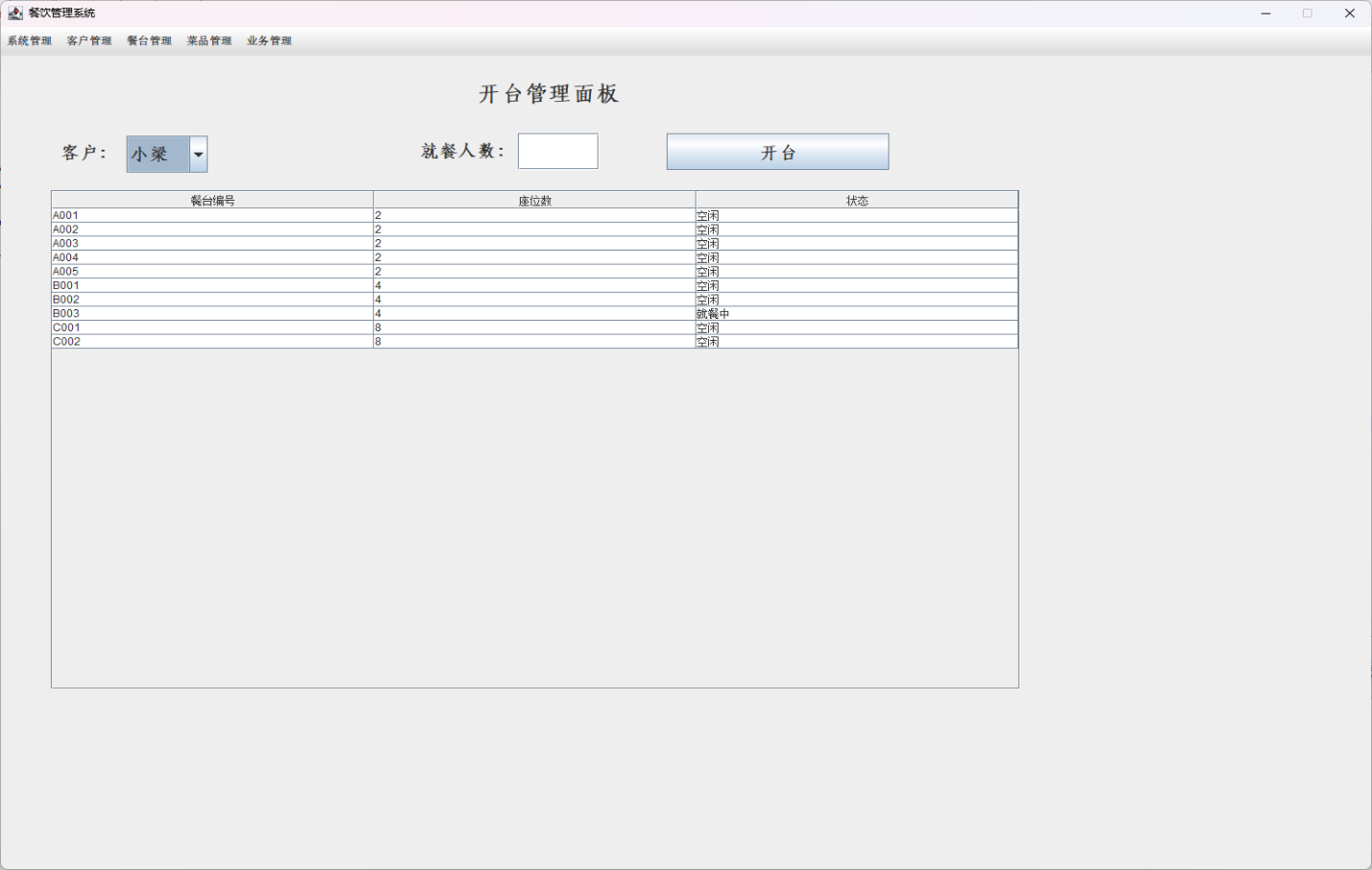
3.3.9 开台管理设计

图3.10开台管理

开台界面包括客户的选择（必须是已添加的客户），就餐的人数（就餐人数必须不大于选中餐台座位数，否则重新选择），以及实时的餐桌信息表和开台按钮。开台需要选择客户、选中一个空闲餐桌并输入就餐人数。开台后该餐桌状态会变为“就餐中”，并在订单记录中生成一条新数据。

3.3.10 点菜管理设计



图3.11点菜管理

点菜界面包括菜品信息表以及菜品类别（选中类别后更新表中相应类别的菜品信息），已点菜品表。餐台编号（必须是正在就餐且订单未支付的餐桌），当前订单编号，添加和删除菜品按钮。总金额以及结账按钮。点击结账按钮会弹出结账对话框。

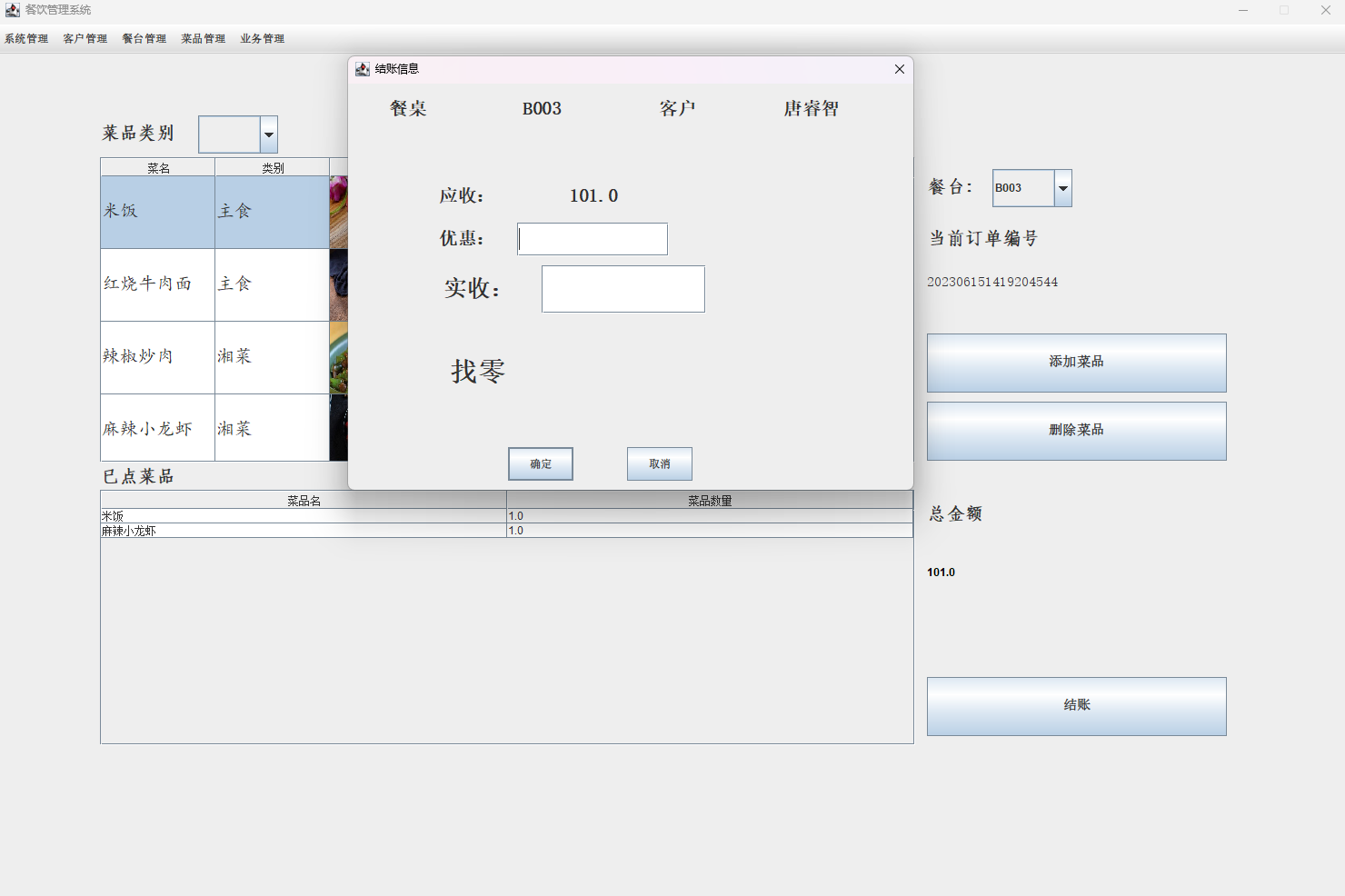
3.3.11 结账管理设计

图3.12结账管理

结账界面包括餐桌编号和客户名，应收金额。输入优惠金额以及实收金额后，点击确认按钮会自动计算找零金额，然后该餐桌状态变为“空闲”，订单状态变为“已支付”。

## 3.4 员工模块设计说明

员工实体类

**package** entity;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** String sex;

**private** String birthday;

**private** String identityID;

**private** String address;

**private** String tel;

**private** String position;

**private** String freeze;

**public** Employee() {}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getSex() {

**return** sex;

}

**public** **void** setSex(String sex) {

**this**.sex = sex;

}

**public** String getBirthday() {

**return** birthday;

}

**public** **void** setBirthday(String birthday) {

**this**.birthday = birthday;

}

**public** String getIdentityID() {

**return** identityID;

}

**public** **void** setIdentityID(String identityID) {

**this**.identityID = identityID;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getTel() {

**return** tel;

}

**public** **void** setTel(String tel) {

**this**.tel = tel;

}

**public** String getPosition() {

**return** position;

}

**public** **void** setPosition(String position) {

**this**.position = position;

}

**public** String getFreeze() {

**return** freeze;

}

**public** **void** setFreeze(String freeze) {

**this**.freeze = freeze;

}

}

员工操作类

**package** dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** entity.Employee;

**public** **class** EmployeeDAOImpl **implements** IBaseDAO<Employee>{

**boolean** isSuccess = **false**;

PreparedStatement ps = **null**;

ResultSet rs = **null**;

String sql = **null**;

Connection conn = JDBCConnection.*getConn*();

@Override

**public** ResultSet findAll() **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from Employee";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryById(**int** Id) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from Employee where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, Id);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryByName(String name) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from Employee where name like \"%\"?\"%\"";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, name);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** **boolean** add(Employee entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "insert into employee(id, name, sex, birthday, identityID, address, tel, position, freeze)"

+ "values(?,?,?,?,?,?,?,?,?)";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

ps.setString(2, entity.getName());

ps.setString(3, entity.getSex());

ps.setString(4, entity.getBirthday());

ps.setString(5, entity.getIdentityID());

ps.setString(6, entity.getAddress());

ps.setString(7, entity.getTel());

ps.setString(8, entity.getPosition());

ps.setString(9, entity.getFreeze());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** delete(Employee entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "delete from employee where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** update(Employee entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "update employee set name = ?, sex = ?, birthday = ?, identityID = ?, address = ?, tel = ?, position = ?, freeze = ?"

+ "where id = ?";

**try** {

ps = conn.prepareStatement(sql);

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

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

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

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

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

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

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

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

ps.setInt(9, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

}

员工工具类

**package** util;

**import** dao.EmployeeDAOImpl;

**public** **class** EmployeeDaoFactory {

**public** EmployeeDAOImpl getEmployeeDAO()

{

**return** **new** EmployeeDAOImpl();

}

}

**package** util;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.table.AbstractTableModel;

**import** dao.EmployeeDAOImpl;

**import** entity.Employee;

**public** **class** EmployeeTableModel **extends** AbstractTableModel{

**private** List<String> columnNames;

**private** List<List<String>> rowDates;

**private** EmployeeDaoFactory df;

**private** EmployeeDAOImpl impl;

**public** EmployeeTableModel() {

df = **new** EmployeeDaoFactory();

impl = df.getEmployeeDAO();

columnNames = **new** ArrayList<String>();

columnNames.add("编号");

columnNames.add("姓名");

columnNames.add("性别");

columnNames.add("出生日期");

columnNames.add("身份证号");

columnNames.add("家庭住址");

columnNames.add("电话");

columnNames.add("职位");

columnNames.add("是否在职");

}

**public** **void** findAll()

{

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.findAll();

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

row.add(rs.getString(4));

row.add(rs.getString(5));

row.add(rs.getString(6));

row.add(rs.getString(7));

row.add(rs.getString(8));

row.add(rs.getString(9));

rowDates.add(row);

}

}**catch**(SQLException e){

e.printStackTrace();

}

}

**public** **boolean** queryModelByName(String name) {

**boolean** flag = **true**;

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.queryByName(name);

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

row.add(rs.getString(4));

row.add(rs.getString(5));

row.add(rs.getString(6));

row.add(rs.getString(7));

row.add(rs.getString(8));

row.add(rs.getString(9));

rowDates.add(row);

}

}**catch**(Exception e){

flag = **false**;

}

**return** flag;

}

**public** **boolean** addModel(Employee ee) **throws** Exception {

**return** impl.add(ee);

}

**public** **boolean** deleteModel(Employee ee) **throws** Exception{

**return** impl.delete(ee);

}

**public** **boolean** updateModel(Employee ee) **throws** Exception{

**return** impl.update(ee);

}

@Override

**public** **int** getColumnCount() {

// **TODO** Auto-generated method stub

**return** **this**.columnNames.size();

}

@Override

**public** **int** getRowCount() {

// **TODO** Auto-generated method stub

**return** **this**.rowDates.size();

}

@Override

**public** Object getValueAt(**int** row, **int** col)

{

// **TODO** Auto-generated method stub

**if**(!rowDates.isEmpty())

**return** ((List<String>)**this**.rowDates.get(row)).get(col);

**else**

**return** **null**;

}

**public** String getColumnName(**int** column)

{

// **TODO** Auto-generated method stub

**return** (String)**this**.columnNames.get(column);

}

}

员工界面类

**package** boundry;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** javax.swing.JButton;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextField;

**import** util.EmployeeTableModel;

**import** javax.swing.SwingConstants;

**public** **class** EmployeeManagePane **extends** JPanel **implements** ActionListener{

//定义顶层标签

**private** JLabel titleLab;

//定义中间数据显示模块

**private** EmployeeTableModel eeModel;

**private** JTable jt;

**private** JScrollPane eePane;

//定义CRUD操作的板、标签、文本框及按钮

**private** JPanel contentPanel;

**private** JLabel id;

**private** JLabel name;

**private** JLabel sex;

**private** JLabel birthday;

**private** JLabel identityID;

**private** JLabel address;

**private** JLabel tel;

**private** JLabel position;

**private** JLabel freeze;

**private** JTextField idTxt = **new** JTextField();

**private** JTextField nameTxt = **new** JTextField();

**private** JTextField sexTxt = **new** JTextField();

**private** JTextField birthdayTxt = **new** JTextField();

**private** JTextField identityIDTxt = **new** JTextField();

**private** JTextField addressTxt = **new** JTextField();

**private** JTextField telTxt = **new** JTextField();

**private** JTextField positionTxt = **new** JTextField();

**private** JTextField freezeTxt = **new** JTextField();

**private** JButton queryBtn;

**private** JButton addBtn;

**private** JButton deleteBtn;

**private** JButton updateBtn;

**private** JButton clearBtn;

/\*\*

\* Create the panel.

\*/

**public** EmployeeManagePane() {

//设置布局及最佳大小

setLayout(**null**);

setPreferredSize(**new** Dimension(1395, 797));

//顶层标签

titleLab = **new** JLabel("员工信息管理面板");

titleLab.setHorizontalAlignment(SwingConstants.***CENTER***);

titleLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 25));

titleLab.setBounds(420, 10, 257, 54);

add(titleLab);

//中间数据显示模块

eeModel = **new** EmployeeTableModel();

jt = **new** JTable(eeModel);

jt.setRowSelectionAllowed(**true**);

eePane = **new** JScrollPane();

eePane.setBounds(0, 74, 949, 713);

eePane.setViewportView(jt);

refreshModel();

add(eePane);

//添加鼠标点击事件，选中二维表某一行后将数据显示在右边的操作文本框中。

jt.addMouseListener(**new** MouseListener() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

// **TODO** Auto-generated method stub

**int** row = jt.getSelectedRow();

idTxt.setText((String)eeModel.getValueAt(row, 0));

nameTxt.setText((String)eeModel.getValueAt(row, 1));

sexTxt.setText((String)eeModel.getValueAt(row, 2));

birthdayTxt.setText((String)eeModel.getValueAt(row, 3));

identityIDTxt.setText((String)eeModel.getValueAt(row, 4));

addressTxt.setText((String)eeModel.getValueAt(row, 5));

telTxt.setText((String)eeModel.getValueAt(row, 6));

positionTxt.setText((String)eeModel.getValueAt(row, 7));

freezeTxt.setText((String)eeModel.getValueAt(row, 8));

}

@Override

**public** **void** mousePressed(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseReleased(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseEntered(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseExited(MouseEvent e) {

// **TODO** Auto-generated method stub

}

});

//CRUD操作模块

contentPanel = **new** JPanel();

contentPanel.setBounds(950, 73, 435, 714);

contentPanel.setLayout(**null**);

id = **new** JLabel("编号：");

id.setBounds(5, 5, 80, 25);

contentPanel.add(id);

idTxt.setBounds(86, 5, 332, 25);

contentPanel.add(idTxt);

name = **new** JLabel("姓名：");

name.setBounds(5, 35, 80, 25);

contentPanel.add(name);

nameTxt.setBounds(86, 35, 332, 25);

contentPanel.add(nameTxt);

sex = **new** JLabel("性别：");

sex.setBounds(5, 65, 80, 25);

contentPanel.add(sex);

sexTxt.setBounds(86, 65, 332, 25);

contentPanel.add(sexTxt);

birthday = **new** JLabel("出生日期：");

birthday.setBounds(5, 95, 80, 25);

contentPanel.add(birthday);

birthdayTxt.setBounds(86, 95, 332, 25);

contentPanel.add(birthdayTxt);

identityID = **new** JLabel("身份证号：");

identityID.setBounds(5, 125, 80, 25);

contentPanel.add(identityID);

identityIDTxt.setBounds(86, 127, 332, 25);

contentPanel.add(identityIDTxt);

address = **new** JLabel("家庭住址：");

address.setBounds(5, 155, 80, 25);

contentPanel.add(address);

addressTxt.setBounds(86, 155, 332, 25);

contentPanel.add(addressTxt);

tel = **new** JLabel("电话：");

tel.setBounds(5, 185, 80, 25);

contentPanel.add(tel);

telTxt.setBounds(85, 185, 333, 25);

contentPanel.add(telTxt);

position = **new** JLabel("职位：");

position.setBounds(5, 215, 80, 25);

contentPanel.add(position);

positionTxt.setBounds(85, 215, 333, 25);

contentPanel.add(positionTxt);

freeze= **new** JLabel("是否在职：");

freeze.setBounds(5, 245, 80, 25);

contentPanel.add(freeze);

freezeTxt.setBounds(85, 245, 333, 25);

contentPanel.add(freezeTxt);

queryBtn = **new** JButton("查询");

queryBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

queryBtn.setBounds(5, 300, 413, 65);

contentPanel.add(queryBtn);

addBtn = **new** JButton("添加");

addBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

addBtn.setBounds(5, 385, 413, 65);

contentPanel.add(addBtn);

deleteBtn = **new** JButton("删除");

deleteBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

deleteBtn.setBounds(5, 470, 413, 65);

contentPanel.add(deleteBtn);

updateBtn = **new** JButton("修改");

updateBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

updateBtn.setBounds(5, 555, 413, 65);

contentPanel.add(updateBtn);

clearBtn = **new** JButton("清空");

clearBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

clearBtn.setBounds(5, 640, 413, 65);

contentPanel.add(clearBtn);

queryBtn.addActionListener(**this**);

addBtn.addActionListener(**this**);

deleteBtn.addActionListener(**this**);

updateBtn.addActionListener(**this**);

clearBtn.addActionListener(**this**);

add(contentPanel);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getSource() == queryBtn)//点击查询按钮

{

String name = nameTxt.getText().trim();//获取姓名文本框内容

**if** (name.length() != 0)//姓名输入有效，执行查询

{

eeModel = **new** EmployeeTableModel();

**if**(eeModel.queryModelByName(name) == **false**)

{

JOptionPane.*showMessageDialog*(**this**, "查无此人！");

}

jt.setModel(eeModel);//更新数据模块

} **else** {

refreshModel();//姓名输入为空，更新数据模块

}

}

**else** **if**(e.getSource() == addBtn)//点击添加按钮

{

entity.Employee ee = **new** entity.Employee();//获取文本框内容

ee.setId(Integer.*parseInt*(idTxt.getText()));

ee.setName(nameTxt.getText());

ee.setSex(sexTxt.getText());

ee.setBirthday(birthdayTxt.getText());

ee.setIdentityID(identityIDTxt.getText());

ee.setAddress(addressTxt.getText());

ee.setTel(telTxt.getText());

ee.setPosition(positionTxt.getText());

ee.setFreeze(freezeTxt.getText());

**try** {

**if** (eeModel.addModel(ee))//添加操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "添加成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if**(e.getSource() == deleteBtn)//点击删除按钮

{

**int** row = jt.getSelectedRow();//获取选中的行号

**if**(row == -1) {

JOptionPane.*showMessageDialog*(**this**, "请选择一行");

**return** ;

}

**int** id = Integer.*parseInt*((String) eeModel.getValueAt(row, 0));//获取编号

EmployeeTableModel temp = **new** EmployeeTableModel();

entity.Employee ee = **new** entity.Employee();

ee.setId(id);

**try** {

**if**(temp.deleteModel(ee))//删除操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "删除成功！");

refreshModel();

}

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == updateBtn)//点击修改按钮

{

entity.Employee ee = **new** entity.Employee();//获取文本框内容

ee.setId(Integer.*parseInt*(idTxt.getText()));

ee.setName(nameTxt.getText());

ee.setSex(sexTxt.getText());

ee.setBirthday(birthdayTxt.getText());

ee.setIdentityID(identityIDTxt.getText());

ee.setAddress(addressTxt.getText());

ee.setTel(telTxt.getText());

ee.setPosition(positionTxt.getText());

ee.setFreeze(freezeTxt.getText());

**try** {

**if** (eeModel.updateModel(ee))//修改操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "修改成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == clearBtn)

{

jt.getSelectionModel().clearSelection();

idTxt.setText("");

nameTxt.setText("");

sexTxt.setText("");

birthdayTxt.setText("");

identityIDTxt.setText("");

addressTxt.setText("");

telTxt.setText("");

positionTxt.setText("");

freezeTxt.setText("");

}

}

//更新数据显示模块的内容函数

**public** **void** refreshModel() {

eeModel = **new** EmployeeTableModel();

eeModel.findAll();

jt.setModel(eeModel);

}

}

## 3.5 客户模块设计说明

客户实体类

**package** entity;

**public** **class** Customer {

**private** **int** id;

**private** String name;

**private** String sex;

**private** String company;

**private** String tel;

**private** String cardID;

**public** Customer() {}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getSex() {

**return** sex;

}

**public** **void** setSex(String sex) {

**this**.sex = sex;

}

**public** String getCompany() {

**return** company;

}

**public** **void** setCompany(String company) {

**this**.company = company;

}

**public** String getTel() {

**return** tel;

}

**public** **void** setTel(String tel) {

**this**.tel = tel;

}

**public** String getCardID() {

**return** cardID;

}

**public** **void** setCardID(String cardID) {

**this**.cardID = cardID;

}

}

客户操作类

**package** dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** entity.Customer;

**public** **class** CustomerDAOImpl **implements** IBaseDAO<Customer>{

**boolean** isSuccess = **false**;

PreparedStatement ps = **null**;

ResultSet rs = **null**;

String sql = **null**;

Connection conn = JDBCConnection.*getConn*();

@Override

**public** ResultSet findAll() **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from customer";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryById(**int** Id) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from customer where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, Id);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryByName(String name) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from customer where name like \"%\"?\"%\"";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, name);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** **boolean** add(Customer entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "insert into customer(id, name, sex, company, tel, cardID)"

+ "values(?,?,?,?,?,?)";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

ps.setString(2, entity.getName());

ps.setString(3, entity.getSex());

ps.setString(4, entity.getCompany());

ps.setString(5, entity.getTel());

ps.setString(6, entity.getCardID());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** delete(Customer entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "delete from customer where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** update(Customer entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "update customer set name = ?, sex = ?, company = ?, tel = ?, cardID = ?"

+ "where id = ?";

**try** {

ps = conn.prepareStatement(sql);

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

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

ps.setString(3, entity.getCompany());

ps.setString(4, entity.getTel());

ps.setString(5, entity.getCardID());

ps.setInt(6, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

}

客户工具类

**package** util;

**import** dao.CustomerDAOImpl;

**public** **class** CustomerDaoFactory {

**public** CustomerDAOImpl getCustomerDAO() {

**return** **new** CustomerDAOImpl();

}

}

**package** util;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.table.AbstractTableModel;

**import** dao.CustomerDAOImpl;

**import** entity.Customer;

**public** **class** CustomerTableModel **extends** AbstractTableModel{

**private** List<String> columnNames;

**private** List<List<String>> rowDates;

**private** CustomerDaoFactory df;

**private** CustomerDAOImpl impl;

**public** CustomerTableModel() {

df = **new** CustomerDaoFactory();

impl = df.getCustomerDAO();

columnNames = **new** ArrayList<String>();

columnNames.add("编号");

columnNames.add("姓名");

columnNames.add("性别");

columnNames.add("单位");

columnNames.add("电话");

columnNames.add("贵宾卡号");

}

**public** **void** findAll()

{

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.findAll();

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

row.add(rs.getString(4));

row.add(rs.getString(5));

row.add(rs.getString(6));

rowDates.add(row);

}

}**catch**(SQLException e){

e.printStackTrace();

}

}

**public** **boolean** queryModelByName(String name) {

**boolean** flag = **true**;

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.queryByName(name);

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

row.add(rs.getString(4));

row.add(rs.getString(5));

row.add(rs.getString(6));

rowDates.add(row);

}

}**catch**(Exception e){

flag = **false**;

}

**return** flag;

}

**public** **boolean** addModel(Customer ee) **throws** Exception {

**return** impl.add(ee);

}

**public** **boolean** deleteModel(Customer ee) **throws** Exception{

**return** impl.delete(ee);

}

**public** **boolean** updateModel(Customer ee) **throws** Exception{

**return** impl.update(ee);

}

**public** List<String> getAllname()

{

List<String> name = **new** ArrayList<String>();

**try** {

ResultSet rs = impl.findAll();

**while** (rs.next())

{

name.add(rs.getString(2));

}

}**catch** (Exception e) {

e.printStackTrace();

}

**return** name;

}

**public** **int** getCustomerId(String name)

{

**int** id = 0;

**try** {

ResultSet rs = impl.queryByName(name);

**while**(rs.next())

{

id = Integer.*parseInt*(rs.getString(1));

}

}**catch**(SQLException e){

e.printStackTrace();

}

**return** id;

}

**public** String getCustomerName(**int** id)

{

String name = **null**;

**try** {

ResultSet rs = impl.queryById(id);

**while**(rs.next()) {

name = rs.getString(2);

}

}**catch**(SQLException e){

e.printStackTrace();

}

**return** name;

}

@Override

**public** **int** getColumnCount() {

// **TODO** Auto-generated method stub

**return** **this**.columnNames.size();

}

@Override

**public** **int** getRowCount() {

// **TODO** Auto-generated method stub

**return** **this**.rowDates.size();

}

@Override

**public** Object getValueAt(**int** row, **int** col)

{

// **TODO** Auto-generated method stub

**if**(!rowDates.isEmpty())

**return** ((List<String>)**this**.rowDates.get(row)).get(col);

**else**

**return** **null**;

}

**public** String getColumnName(**int** column)

{

// **TODO** Auto-generated method stub

**return** (String)**this**.columnNames.get(column);

}

}

客户界面类

**package** boundry;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** javax.swing.JButton;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextField;

**import** javax.swing.SwingConstants;

**import** util.CustomerTableModel;

**public** **class** CustomerManagePane **extends** JPanel **implements** ActionListener{

//定义顶层标签

**private** JLabel titleLab;

//定义中间数据显示模块

**private** CustomerTableModel cModel;

**private** JTable jt;

**private** JScrollPane eePane;

//定义CRUD操作的板、标签、文本框及按钮

**private** JPanel contentPanel;

**private** JLabel id;

**private** JLabel name;

**private** JLabel sex;

**private** JLabel company;

**private** JLabel tel;

**private** JLabel cardID;

**private** JTextField idTxt = **new** JTextField();

**private** JTextField nameTxt = **new** JTextField();

**private** JTextField sexTxt = **new** JTextField();

**private** JTextField companyTxt = **new** JTextField();

**private** JTextField telTxt = **new** JTextField();

**private** JTextField cardIDTxt = **new** JTextField();

**private** JButton queryBtn;

**private** JButton addBtn;

**private** JButton deleteBtn;

**private** JButton updateBtn;

**private** JButton clearBtn;

/\*\*

\* Create the panel.

\*/

**public** CustomerManagePane() {

//设置布局及最佳大小

setLayout(**null**);

setPreferredSize(**new** Dimension(1395, 797));

//顶层标签

titleLab = **new** JLabel("顾客信息管理面板");

titleLab.setHorizontalAlignment(SwingConstants.***CENTER***);

titleLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 25));

titleLab.setBounds(420, 10, 257, 54);

add(titleLab);

//中间数据显示模块

cModel = **new** CustomerTableModel();

jt = **new** JTable(cModel);

jt.setRowSelectionAllowed(**true**);

eePane = **new** JScrollPane();

eePane.setBounds(0, 74, 949, 713);

eePane.setViewportView(jt);

refreshModel();

add(eePane);

//添加鼠标点击事件，选中二维表某一行后将数据显示在右边的操作文本框中。

jt.addMouseListener(**new** MouseListener() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

// **TODO** Auto-generated method stub

**int** row = jt.getSelectedRow();

idTxt.setText((String)cModel.getValueAt(row, 0));

nameTxt.setText((String)cModel.getValueAt(row, 1));

sexTxt.setText((String)cModel.getValueAt(row, 2));

companyTxt.setText((String)cModel.getValueAt(row, 3));

telTxt.setText((String)cModel.getValueAt(row, 4));

cardIDTxt.setText((String)cModel.getValueAt(row, 5));

}

@Override

**public** **void** mousePressed(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseReleased(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseEntered(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseExited(MouseEvent e) {

// **TODO** Auto-generated method stub

}

});

//CRUD操作模块

contentPanel = **new** JPanel();

contentPanel.setBounds(950, 73, 435, 714);

contentPanel.setLayout(**null**);

id = **new** JLabel("编号：");

id.setBounds(5, 5, 80, 25);

contentPanel.add(id);

idTxt.setBounds(86, 5, 332, 25);

contentPanel.add(idTxt);

name = **new** JLabel("姓名：");

name.setBounds(5, 35, 80, 25);

contentPanel.add(name);

nameTxt.setBounds(86, 35, 332, 25);

contentPanel.add(nameTxt);

sex = **new** JLabel("性别：");

sex.setBounds(5, 65, 80, 25);

contentPanel.add(sex);

sexTxt.setBounds(86, 65, 332, 25);

contentPanel.add(sexTxt);

company = **new** JLabel("单位：");

company.setBounds(5, 95, 80, 25);

contentPanel.add(company);

companyTxt.setBounds(86, 95, 332, 25);

contentPanel.add(companyTxt);

tel = **new** JLabel("电话：");

tel.setBounds(5, 125, 80, 25);

contentPanel.add(tel);

telTxt.setBounds(86, 127, 332, 25);

contentPanel.add(telTxt);

cardID = **new** JLabel("贵宾卡号：");

cardID.setBounds(5, 155, 80, 25);

contentPanel.add(cardID);

cardIDTxt.setBounds(86, 155, 332, 25);

contentPanel.add(cardIDTxt);

queryBtn = **new** JButton("查询");

queryBtn.setBounds(5, 300, 413, 65);

contentPanel.add(queryBtn);

addBtn = **new** JButton("添加");

addBtn.setBounds(5, 385, 413, 65);

contentPanel.add(addBtn);

deleteBtn = **new** JButton("删除");

deleteBtn.setBounds(5, 470, 413, 65);

contentPanel.add(deleteBtn);

updateBtn = **new** JButton("修改");

updateBtn.setBounds(5, 555, 413, 65);

contentPanel.add(updateBtn);

clearBtn = **new** JButton("清空");

clearBtn.setBounds(5,640,413,64);

contentPanel.add(clearBtn);

queryBtn.addActionListener(**this**);

updateBtn.addActionListener(**this**);

deleteBtn.addActionListener(**this**);

addBtn.addActionListener(**this**);

clearBtn.addActionListener(**this**);

add(contentPanel);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getSource() == queryBtn)//点击查询按钮

{

String name = nameTxt.getText().trim();//获取姓名文本框内容

**if** (name.length() != 0)//姓名输入有效，执行查询

{

cModel = **new** CustomerTableModel();

**if**(cModel.queryModelByName(name) == **false**)

{

JOptionPane.*showMessageDialog*(**this**, "查无此人！");

}

jt.setModel(cModel);//更新数据模块

} **else** {

refreshModel();//姓名输入为空，更新数据模块

}

}

**else** **if**(e.getSource() == addBtn)//点击添加按钮

{

entity.Customer c = **new** entity.Customer();//获取文本框内容

c.setId(Integer.*parseInt*(idTxt.getText()));

c.setName(nameTxt.getText());

c.setSex(sexTxt.getText());

c.setCompany(companyTxt.getText());

c.setTel(telTxt.getText());

c.setCardID(cardIDTxt.getText());

**try** {

**if** (cModel.addModel(c))//添加操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "添加成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if**(e.getSource() == deleteBtn)//点击删除按钮

{

**int** row = jt.getSelectedRow();//获取选中的行号

**if**(row == -1) {

JOptionPane.*showMessageDialog*(**this**, "请选择一行");

**return** ;

}

**int** id = Integer.*parseInt*((String) cModel.getValueAt(row, 0));//获取编号

CustomerTableModel temp = **new** CustomerTableModel();

entity.Customer c = **new** entity.Customer();

c.setId(id);

**try** {

**if**(temp.deleteModel(c))//删除操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "删除成功！");

}

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

refreshModel();

}

**else** **if** (e.getSource() == updateBtn)//点击修改按钮

{

entity.Customer c = **new** entity.Customer();//获取文本框内容

c.setId(Integer.*parseInt*(idTxt.getText()));

c.setName(nameTxt.getText());

c.setSex(sexTxt.getText());

c.setCompany(companyTxt.getText());

c.setTel(telTxt.getText());

c.setCardID(cardIDTxt.getText());

**try** {

**if** (cModel.updateModel(c))//修改操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "修改成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == clearBtn)

{

jt.getSelectionModel().clearSelection();

idTxt.setText("");

nameTxt.setText("");

sexTxt.setText("");

companyTxt.setText("");

telTxt.setText("");

cardIDTxt.setText("");

}

}

//更新数据显示模块的内容函数

**public** **void** refreshModel() {

cModel = **new** CustomerTableModel();

cModel.findAll();

jt.setModel(cModel);

}

}

## 3.6 菜品模块设计说明

菜品实体类

**package** entity;

**import** java.sql.Blob;

**public** **class** Dish {

**private** **int** id;

**private** String name;

**private** **int** categoryId;

**private** Blob pic;

**private** String code;

**private** String unit;

**private** **double** price;

**private** String status;

**public** Dish() {}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getCategoryId() {

**return** categoryId;

}

**public** **void** setCategoryId(**int** categoryId) {

**this**.categoryId = categoryId;

}

**public** Blob getPic() {

**return** pic;

}

**public** **void** setPic(Blob pic) {

**this**.pic = pic;

}

**public** String getCode() {

**return** code;

}

**public** **void** setCode(String code) {

**this**.code = code;

}

**public** String getUnit() {

**return** unit;

}

**public** **void** setUnit(String unit) {

**this**.unit = unit;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

}

**package** entity;

**public** **class** Category {

**private** **int** id;

**private** String name;

**private** String describ;

**public** Category() {}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getDescrib() {

**return** describ;

}

**public** **void** setDescrib(String describ) {

**this**.describ = describ;

}

}

菜品操作类

**package** dao;

**import** java.io.ByteArrayOutputStream;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.sql.Blob;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** entity.Dish;

**public** **class** dishDAOImpl **implements** IBaseDAO<Dish>{

**boolean** isSuccess = **false**;

PreparedStatement ps = **null**;

ResultSet rs = **null**;

String sql = **null**;

Connection conn = JDBCConnection.*getConn*();

**public** ResultSet queryByCate(String category)

{

sql = "select d.name, c.name, d.pic, d.code, d.unit, d.price, d.status "

+ " from dish d,category c where d.categoryId = c.id and c.name = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, category);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet findAll() **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from dish";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryById(**int** Id) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from dish where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, Id);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryByName(String name) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from dish where name like \"%\"?\"%\"";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, name);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** **boolean** add(Dish entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "insert into dish(name, categoryId, pic, code, unit, price, status)"

+ "values(?,?,?,?,?,?,?)";

**try** {

ps = conn.prepareStatement(sql);

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

ps.setInt(2, entity.getCategoryId());

ps.setBlob(3, entity.getPic());

ps.setString(4, entity.getCode());

ps.setString(5, entity.getUnit());

ps.setDouble(6, entity.getPrice());

ps.setString(7, entity.getStatus());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** delete(Dish entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "delete from dish where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** update(Dish entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "update dish set name = ?, categoryId = ?, pic = ?, code = ?, unit = ?, price = ?, status = ?"

+ "where id = ?";

**try** {

ps = conn.prepareStatement(sql);

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

ps.setInt(2, entity.getCategoryId());

ps.setBlob(3, entity.getPic());

ps.setString(4, entity.getCode());

ps.setString(5, entity.getUnit());

ps.setDouble(6, entity.getPrice());

ps.setString(7, entity.getStatus());

ps.setInt(8, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

**public** Blob stringToBlob(String s) **throws** IOException, SQLException

{

File file = **new** File(s);

FileInputStream fis = **new** FileInputStream(file);

ByteArrayOutputStream baos = **new** ByteArrayOutputStream();

**int** len = -1;

**byte**[] buf = **new** **byte**[1024];

**while**((len = fis.read(buf)) != -1) {//汇总字节流到内存

baos.write(buf, 0, len);

}

baos.close();

fis.close();

**byte**[] bytes = baos.toByteArray();//从内存取出字节流数组

Blob pic = conn.createBlob();

pic.setBytes(1, bytes);//把字节流设置给blob类

**return** pic;

}

**public** String blobToString(Blob b) **throws** SQLException

{

**return** **new** String(b.getBytes((**long**)1, (**int**)b.length()));

}

}

**package** dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** entity.Category;

**public** **class** CategoryDAOImpl **implements** IBaseDAO<Category>{

**boolean** isSuccess = **false**;

PreparedStatement ps = **null**;

ResultSet rs = **null**;

String sql = **null**;

Connection conn = JDBCConnection.*getConn*();

@Override

**public** ResultSet findAll() **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from category";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryById(**int** Id) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from category where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, Id);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryByName(String name) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from category where name like \"%\"?\"%\"";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, name);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** **boolean** add(Category entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "insert into category(id, name, describ)"

+ "values(?,?,?)";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

ps.setString(2, entity.getName());

ps.setString(3, entity.getDescrib());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** delete(Category entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "delete from category where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** update(Category entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "update category set name = ?, describ = ?"

+ "where id = ?";

**try** {

ps = conn.prepareStatement(sql);

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

ps.setString(2, entity.getDescrib());

ps.setInt(3, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

}

菜品工具类

**package** util;

**import** dao.dishDAOImpl;

**public** **class** DishDaoFactory {

**public** dishDAOImpl getDishDao() {

**return** **new** dishDAOImpl();

}

}

**package** util;

**import** java.awt.Image;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.sql.Blob;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.List;

**import** java.util.Vector;

**import** javax.swing.ImageIcon;

**import** javax.swing.table.DefaultTableModel;

**import** dao.dishDAOImpl;

**import** entity.Dish;

**public** **class** DishTableModel **extends** DefaultTableModel{

Vector<String> columnNames;

Vector<Vector<Object>> rowDates;

**private** DishDaoFactory df;

**private** dishDAOImpl impl;

**public** DishTableModel() **throws** IOException

{

df = **new** DishDaoFactory();

impl = df.getDishDao();

columnNames = **new** Vector<String>();

columnNames.add("菜名");

columnNames.add("类别");

columnNames.add("图片");

columnNames.add("代码");

columnNames.add("单位");

columnNames.add("价格");

columnNames.add("状态");

}

**public** **void** findAll() **throws** IOException

{

CategoryTableModel cateModel = **new** CategoryTableModel();

Vector<Vector<Object>> rowDates = **new** Vector<Vector<Object>>();

**try** {

**int** k = 1;

ResultSet rs = impl.findAll();

**while**(rs.next()) {

Vector<Object> row = **new** Vector<Object>();

row.add(rs.getString(2));

row.add(cateModel.getCategoryById(Integer.*parseInt*(rs.getString(3))));

InputStream in = rs.getBlob(4).getBinaryStream();

String pic = "D:\\trz\\jpg1\\th"+k+".jpg";//为图片设置不同的地址

FileOutputStream out = **new** FileOutputStream(pic);

**int** len = -1;

**byte**[] buffer = **new** **byte**[1024];

**while**((len = in.read(buffer))!=-1) {//将图片写入本地硬盘中

out.write(buffer,0,len);

}

k++;

out.close();

in.close();

ImageIcon icon = **new** ImageIcon(pic);

icon.setImage(icon.getImage().getScaledInstance(100,100,Image.***SCALE\_DEFAULT***));//设置图片在JTable中的大小

Image img = icon.getImage();

row.add(img);

row.add(rs.getString(5));

row.add(rs.getString(6));

row.add(rs.getString(7));

row.add(rs.getString(8));

rowDates.add(row);

}

}**catch**(SQLException e){

e.printStackTrace();

}

**this**.setDataVector(rowDates, columnNames);

}

**public** **void** queryModelByCate(String category) **throws** IOException {

**boolean** flag = **true**;

CategoryTableModel cateModel = **new** CategoryTableModel();

Vector<Vector<Object>> rowDates = **new** Vector<Vector<Object>>();

**try** {

**int** k = 1;

ResultSet rs = impl.queryByCate(category);

**while**(rs.next()) {

Vector<Object> row = **new** Vector<Object>();

row.add(rs.getString(1));

row.add(rs.getString(2));

InputStream in = rs.getBlob(3).getBinaryStream();

String pic = "D:\\trz\\jpg1\\th"+k+".jpg";//为图片设置不同的地址

FileOutputStream out = **new** FileOutputStream(pic);

**int** len = -1;

**byte**[] buffer = **new** **byte**[1024];

**while**((len = in.read(buffer))!=-1) {//将图片写入本地硬盘中

out.write(buffer,0,len);

}

k++;

out.close();

in.close();

ImageIcon icon = **new** ImageIcon(pic);

icon.setImage(icon.getImage().getScaledInstance(100,100,Image.***SCALE\_DEFAULT***));//设置图片在JTable中的大小

Image img = icon.getImage();

row.add(img);

row.add(rs.getString(4));

row.add(rs.getString(5));

row.add(rs.getString(6));

row.add(rs.getString(7));

rowDates.add(row);

}

}**catch**(SQLException e){

flag = **false**;

}

**this**.setDataVector(rowDates, columnNames);

}

**public** List<String> getAllCategory() {

// **TODO** Auto-generated method stub

CategoryTableModel cateModel = **new** CategoryTableModel();

**return** cateModel.getAllCategory();

}

@Override

**public** Object getValueAt(**int** row, **int** column) {

// **TODO** Auto-generated method stub

**return** **super**.getValueAt(row, column);

}

**public** **int** getDishIdByName(String name)

{

**int** id = 0;

**try** {

ResultSet rs = impl.queryByName(name);

**while** (rs.next()) {

id = Integer.*parseInt*(rs.getString(1));

}

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** id;

}

**public** String getCategoryById(**int** id)

{

CategoryTableModel cateModel = **new** CategoryTableModel();

**return** cateModel.getCategoryById(id);

}

**public** **int** getIdByName(String name)

{

CategoryTableModel cateModel = **new** CategoryTableModel();

**return** cateModel.getIdByName(name);

}

**public** Blob sToBlob(String s) **throws** IOException, SQLException

{

**return** impl.stringToBlob(s);

}

**public** String blobToString(Blob b) **throws** SQLException

{

**return** impl.blobToString(b);

}

**public** **boolean** addModel(Dish d) **throws** Exception {

// **TODO** Auto-generated method stub

**return** impl.add(d);

}

**public** **boolean** deleteModel(Dish d) **throws** Exception {

// **TODO** Auto-generated method stub

**return** impl.delete(d);

}

**public** **boolean** updateModel(Dish d) **throws** Exception {

// **TODO** Auto-generated method stub

**return** impl.update(d);

}

}

**package** util;

**import** dao.CategoryDAOImpl;

**public** **class** CategoryDaoFactory {

**public** CategoryDAOImpl getCategoryDao() {

**return** **new** CategoryDAOImpl();

}

}

**package** util;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.table.AbstractTableModel;

**import** dao.CategoryDAOImpl;

**import** entity.Category;

**public** **class** CategoryTableModel **extends** AbstractTableModel{

**private** List<String> columnNames;

**private** List<List<String>> rowDates;

**private** CategoryDaoFactory df;

**private** CategoryDAOImpl impl;

**public** CategoryTableModel() {

df = **new** CategoryDaoFactory();

impl = df.getCategoryDao();

columnNames = **new** ArrayList<String>();

columnNames.add("编号");

columnNames.add("菜名");

columnNames.add("描述");

}

**public** **void** findAll()

{

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.findAll();

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

rowDates.add(row);

}

}**catch**(SQLException e){

e.printStackTrace();

}

}

**public** **boolean** queryModelById(**int** id) {

**boolean** flag = **true**;

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.queryById(id);

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

rowDates.add(row);

}

}**catch**(Exception e){

flag = **false**;

}

**return** flag;

}

**public** **boolean** queryModelByName(String name) {

**boolean** flag = **true**;

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.queryByName(name);

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

rowDates.add(row);

}

}**catch**(Exception e){

flag = **false**;

}

**return** flag;

}

**public** **boolean** addModel(Category ee) **throws** Exception {

**return** impl.add(ee);

}

**public** **boolean** deleteModel(Category ee) **throws** Exception{

**return** impl.delete(ee);

}

**public** **boolean** updateModel(Category ee) **throws** Exception{

**return** impl.update(ee);

}

**public** **int** getIdByName(String name)

{

**int** id = 0;

**try** {

ResultSet rs = impl.queryByName(name);

**while** (rs.next())

{

id = Integer.*parseInt*(rs.getString(1));

}

}**catch**(SQLException e){

e.printStackTrace();

}

**return** id;

}

**public** String getCategoryById(**int** id)

{

String cate = **null**;

**try** {

ResultSet rs = impl.queryById(id);

**while** (rs.next())

{

cate = rs.getString(2);

}

}**catch**(SQLException e){

e.printStackTrace();

}

**return** cate;

}

**public** List<String> getAllCategory()

{

List<String> category = **new** ArrayList<String>();

**try** {

ResultSet rs = impl.findAll();

**while**(rs.next())

{

category.add(rs.getString(2));

}

}**catch**(SQLException e){

e.printStackTrace();

}

**return** category;

}

@Override

**public** **int** getRowCount() {

// **TODO** Auto-generated method stub

**return** **this**.rowDates.size();

}

@Override

**public** String getColumnName(**int** column) {

// **TODO** Auto-generated method stub

**return** (String)**this**.columnNames.get(column);

}

@Override

**public** **int** getColumnCount() {

// **TODO** Auto-generated method stub

**return** **this**.columnNames.size();

}

@Override

**public** Object getValueAt(**int** row, **int** col) {

// **TODO** Auto-generated method stub

**if**(!rowDates.isEmpty())

**return** ((List<String>)**this**.rowDates.get(row)).get(col);

**else**

**return** **null**;

}

}

菜品界面类

**package** boundry;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** java.io.IOException;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.JButton;

**import** javax.swing.JComboBox;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextField;

**import** javax.swing.SwingConstants;

**import** util.DishTableModel;

**public** **class** DishManagePane **extends** JPanel **implements** ActionListener{

//定义顶层标签

**private** JLabel titleLab;

//定义中间数据显示模块

**private** DishTableModel model;

**private** JTable jt;

**private** JScrollPane eePane;

//定义CRUD操作的板、标签、文本框及按钮

**private** JPanel contentPanel;

**private** JLabel name;

**private** JLabel category;

**private** JLabel pic;

**private** JLabel code;

**private** JLabel unit;

**private** JLabel price;

**private** JLabel status;

**private** JTextField nameTxt = **new** JTextField();

**private** JComboBox<String> categoryBox;

**private** JTextField picTxt = **new** JTextField();

**private** JTextField codeTxt = **new** JTextField();

**private** JTextField unitTxt = **new** JTextField();

**private** JTextField priceTxt = **new** JTextField();

**private** JTextField statusTxt = **new** JTextField();

**private** JButton queryBtn;

**private** JButton addBtn;

**private** JButton deleteBtn;

**private** JButton updateBtn;

**private** JButton clearBtn;

/\*\*

\* Create the panel.

\* **@throws** IOException

\*/

**public** DishManagePane() **throws** IOException {

//设置布局及最佳大小

setLayout(**null**);

setPreferredSize(**new** Dimension(1366, 792));

//顶层标签

titleLab = **new** JLabel("菜品信息管理面板");

titleLab.setHorizontalAlignment(SwingConstants.***CENTER***);

titleLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 25));

titleLab.setBounds(420, 10, 257, 54);

add(titleLab);

model = **new** DishTableModel();

jt = **new** JTable();

jt.setFont(**new** Font("等线", Font.***PLAIN***, 20));

jt.setToolTipText("");

refreshModel();

eePane = **new** JScrollPane();

eePane.setBounds(10, 74, 909, 713);

eePane.setViewportView(jt);

add(eePane);

jt.addMouseListener(**new** MouseListener() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

// **TODO** Auto-generated method stub

**int** row = jt.getSelectedRow();

nameTxt.setText((String)model.getValueAt(row, 0));

categoryBox.setSelectedItem((String)model.getValueAt(row, 1));

//picTxt.setText(model.blobToString(model.getValueAt(row, 2)));

codeTxt.setText((String)model.getValueAt(row, 3));

unitTxt.setText((String)model.getValueAt(row, 4));

priceTxt.setText((String)model.getValueAt(row, 5));

statusTxt.setText((String)model.getValueAt(row, 6));

}

@Override

**public** **void** mousePressed(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseReleased(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseEntered(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseExited(MouseEvent e) {

// **TODO** Auto-generated method stub

}

});

//CRUD操作模块

contentPanel = **new** JPanel();

contentPanel.setBounds(928, 74, 435, 714);

contentPanel.setLayout(**null**);

name = **new** JLabel("菜名：");

name.setBounds(5, 10, 80, 25);

contentPanel.add(name);

nameTxt.setBounds(86, 10, 332, 25);

contentPanel.add(nameTxt);

category = **new** JLabel("类别：");

category.setBounds(5, 40, 80, 25);

contentPanel.add(category);

categoryBox = **new** JComboBox<String>();

fillCategory();

categoryBox.setSelectedIndex(-1);

categoryBox.setBounds(86, 40, 332, 25);

contentPanel.add(categoryBox);

pic = **new** JLabel("图片：");

pic.setBounds(5, 70, 80, 25);

contentPanel.add(pic);

picTxt.setBounds(86, 70, 332, 25);

contentPanel.add(picTxt);

code = **new** JLabel("代码：");

code.setBounds(5, 100, 80, 25);

contentPanel.add(code);

codeTxt.setBounds(86, 102, 332, 25);

contentPanel.add(codeTxt);

unit = **new** JLabel("单位：");

unit.setBounds(5, 130, 80, 25);

contentPanel.add(unit);

unitTxt.setBounds(86, 130, 332, 25);

contentPanel.add(unitTxt);

price = **new** JLabel("价格：");

price.setBounds(5, 160, 80, 25);

contentPanel.add(price);

priceTxt.setBounds(85, 160, 332, 25);

contentPanel.add(priceTxt);

status = **new** JLabel("状态：");

status.setBounds(5, 190, 80, 25);

contentPanel.add(status);

statusTxt.setBounds(85, 190, 332, 25);

contentPanel.add(statusTxt);

queryBtn = **new** JButton("查询");

queryBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

queryBtn.setBounds(5, 300, 413, 65);

contentPanel.add(queryBtn);

addBtn = **new** JButton("添加");

addBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

addBtn.setBounds(5, 385, 413, 65);

contentPanel.add(addBtn);

deleteBtn = **new** JButton("删除");

deleteBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

deleteBtn.setBounds(5, 470, 413, 65);

contentPanel.add(deleteBtn);

updateBtn = **new** JButton("修改");

updateBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

updateBtn.setBounds(5, 555, 413, 65);

contentPanel.add(updateBtn);

clearBtn = **new** JButton("清空");

clearBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

clearBtn.setBounds(5, 640, 413, 65);

contentPanel.add(clearBtn);

queryBtn.addActionListener(**this**);

addBtn.addActionListener(**this**);

deleteBtn.addActionListener(**this**);

updateBtn.addActionListener(**this**);

clearBtn.addActionListener(**this**);

add(contentPanel);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getSource() == queryBtn)//点击查询按钮

{

}

**else** **if**(e.getSource() == addBtn)//点击添加按钮

{

entity.Dish d = **new** entity.Dish();//获取文本框内容

d.setName(nameTxt.getText());

d.setCategoryId(model.getIdByName((String)categoryBox.getSelectedItem()));

**try** {

d.setPic(model.sToBlob(picTxt.getText()));

} **catch** (IOException | SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

d.setCode(codeTxt.getText());

d.setUnit(unitTxt.getText());

d.setPrice(Double.*parseDouble*(priceTxt.getText()));

d.setStatus(statusTxt.getText());

**try** {

**if** (model.addModel(d))//添加操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "添加成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if**(e.getSource() == deleteBtn)//点击删除按钮

{

**int** row = jt.getSelectedRow();//获取选中的行号

**if**(row == -1) {

JOptionPane.*showMessageDialog*(**this**, "请选择一行");

**return** ;

}

**int** id = model.getDishIdByName((String) model.getValueAt(row, 0));//获取编号

entity.Dish d = **new** entity.Dish();

d.setId(id);

DishTableModel temp = **null**;

**try** {

temp = **new** DishTableModel();

**if**(temp.deleteModel(d))//删除操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "删除成功！");

refreshModel();

}

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == updateBtn)//点击修改按钮

{

entity.Dish d = **new** entity.Dish();//获取文本框内容

d.setName(nameTxt.getText());

d.setCategoryId(model.getIdByName((String)categoryBox.getSelectedItem()));

**try** {

d.setPic(model.sToBlob(picTxt.getText()));

} **catch** (IOException | SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

d.setCode(codeTxt.getText());

d.setUnit(unitTxt.getText());

d.setPrice(Double.*parseDouble*(priceTxt.getText()));

d.setStatus(statusTxt.getText());

**try** {

**if** (model.updateModel(d))//修改操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "修改成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == clearBtn)

{

jt.getSelectionModel().clearSelection();

nameTxt.setText("");

categoryBox.setSelectedIndex(-1);

picTxt.setText("");

codeTxt.setText("");

unitTxt.setText("");

priceTxt.setText("");

statusTxt.setText("");

}

}

**public** **void** fillCategory() {

List<String> ListCate = **new** ArrayList<String>();

ListCate = model.getAllCategory();

**for** (String temp : ListCate)

{

categoryBox.addItem(temp);

}

}

**public** **void** refreshModel() **throws** IOException

{

model = **new** DishTableModel();

model.findAll();

jt.setModel(model);

jt.getColumnModel().getColumn(2).setCellRenderer(**new** ImageRenderer());

jt.setRowHeight(80);

jt.setRowSelectionAllowed(**true**);

}

}

**package** boundry;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** javax.swing.JButton;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextField;

**import** javax.swing.SwingConstants;

**import** util.CategoryTableModel;

**public** **class** CategoryManagePane **extends** JPanel **implements** ActionListener{

//定义顶层标签

**private** JLabel titleLab;

//定义中间数据显示模块

**private** CategoryTableModel cateModel;

**private** JTable jt;

**private** JScrollPane eePane;

//定义CRUD操作的板、标签、文本框及按钮

**private** JPanel contentPanel;

**private** JLabel id;

**private** JLabel name;

**private** JLabel describ;

**private** JTextField idTxt = **new** JTextField();

**private** JTextField nameTxt = **new** JTextField();

**private** JTextField describTxt = **new** JTextField();

**private** JButton queryBtn;

**private** JButton addBtn;

**private** JButton deleteBtn;

**private** JButton updateBtn;

**private** JButton clearBtn;

/\*\*

\* Create the panel.

\*/

**public** CategoryManagePane() {

//设置布局及最佳大小

setLayout(**null**);

setPreferredSize(**new** Dimension(1395, 797));

//顶层标签

titleLab = **new** JLabel("菜品分类管理面板");

titleLab.setHorizontalAlignment(SwingConstants.***CENTER***);

titleLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 25));

titleLab.setBounds(420, 10, 257, 54);

add(titleLab);

//中间数据显示模块

cateModel = **new** CategoryTableModel();

jt = **new** JTable(cateModel);

jt.setRowSelectionAllowed(**true**);

eePane = **new** JScrollPane();

eePane.setBounds(0, 74, 949, 713);

eePane.setViewportView(jt);

refreshModel();

add(eePane);

//添加鼠标点击事件，选中二维表某一行后将数据显示在右边的操作文本框中。

jt.addMouseListener(**new** MouseListener() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

// **TODO** Auto-generated method stub

**int** row = jt.getSelectedRow();

idTxt.setText((String)cateModel.getValueAt(row, 0));

nameTxt.setText((String)cateModel.getValueAt(row, 1));

describTxt.setText((String)cateModel.getValueAt(row, 2));

}

@Override

**public** **void** mousePressed(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseReleased(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseEntered(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseExited(MouseEvent e) {

// **TODO** Auto-generated method stub

}

});

//CRUD操作模块

contentPanel = **new** JPanel();

contentPanel.setBounds(950, 73, 435, 714);

contentPanel.setLayout(**null**);

id = **new** JLabel("编号：");

id.setBounds(5, 5, 80, 25);

contentPanel.add(id);

idTxt.setBounds(86, 5, 332, 25);

contentPanel.add(idTxt);

name = **new** JLabel("菜名：");

name.setBounds(5, 35, 80, 25);

contentPanel.add(name);

nameTxt.setBounds(86, 35, 332, 25);

contentPanel.add(nameTxt);

describ = **new** JLabel("描述：");

describ.setBounds(5, 65, 80, 25);

contentPanel.add(describ);

describTxt.setBounds(86, 65, 332, 25);

contentPanel.add(describTxt);

queryBtn = **new** JButton("查询");

queryBtn.setBounds(5, 300, 413, 65);

contentPanel.add(queryBtn);

addBtn = **new** JButton("添加");

addBtn.setBounds(5, 385, 413, 65);

contentPanel.add(addBtn);

deleteBtn = **new** JButton("删除");

deleteBtn.setBounds(5, 470, 413, 65);

contentPanel.add(deleteBtn);

updateBtn = **new** JButton("修改");

updateBtn.setBounds(5, 555, 413, 65);

contentPanel.add(updateBtn);

clearBtn = **new** JButton("清空");

clearBtn.setBounds(5,640,413,64);

contentPanel.add(clearBtn);

queryBtn.addActionListener(**this**);

updateBtn.addActionListener(**this**);

deleteBtn.addActionListener(**this**);

addBtn.addActionListener(**this**);

clearBtn.addActionListener(**this**);

add(contentPanel);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getSource() == queryBtn)//点击查询按钮

{

String name = nameTxt.getText().trim();//获取姓名文本框内容

**if** (name.length() != 0)//姓名输入有效，执行查询

{

cateModel = **new** CategoryTableModel();

**if**(cateModel.queryModelByName(name) == **false**)

{

JOptionPane.*showMessageDialog*(**this**, "查无此人！");

}

jt.setModel(cateModel);//更新数据模块

} **else** {

refreshModel();//姓名输入为空，更新数据模块

}

}

**else** **if**(e.getSource() == addBtn)//点击添加按钮

{

entity.Category cate = **new** entity.Category();//获取文本框内容

cate.setId(Integer.*parseInt*(idTxt.getText()));

cate.setName(nameTxt.getText());

cate.setDescrib(describTxt.getText());

**try** {

**if** (cateModel.addModel(cate))//添加操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "添加成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if**(e.getSource() == deleteBtn)//点击删除按钮

{

**int** row = jt.getSelectedRow();//获取选中的行号

**if**(row == -1) {

JOptionPane.*showMessageDialog*(**this**, "请选择一行");

**return** ;

}

**int** id = Integer.*parseInt*((String) cateModel.getValueAt(row, 0));//获取编号

CategoryTableModel temp = **new** CategoryTableModel();

entity.Category cate = **new** entity.Category();

cate.setId(id);

**try** {

**if**(temp.deleteModel(cate))//删除操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "删除成功！");

}

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

refreshModel();

}

**else** **if** (e.getSource() == updateBtn)//点击删除按钮

{

entity.Category cate = **new** entity.Category();//获取文本框内容

cate.setId(Integer.*parseInt*(idTxt.getText()));

cate.setName(nameTxt.getText());

cate.setDescrib(describTxt.getText());

**try** {

**if** (cateModel.updateModel(cate))//修改操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "修改成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == clearBtn)

{

jt.getSelectionModel().clearSelection();

idTxt.setText("");

nameTxt.setText("");

describTxt.setText("");

}

}

//更新数据显示模块的内容函数

**public** **void** refreshModel() {

cateModel = **new** CategoryTableModel();

cateModel.findAll();

jt.setModel(cateModel);

}

}

## 3.7 餐桌模块设计说明

餐桌实体类

**package** entity;

**public** **class** Desk {

**private** **int** id;

**private** String no;

**private** **int** seating;

**private** String status;

**public** Desk() {}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getNo() {

**return** no;

}

**public** **void** setNo(String no) {

**this**.no = no;

}

**public** **int** getSeating() {

**return** seating;

}

**public** **void** setSeating(**int** seating) {

**this**.seating = seating;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

}

餐桌操作类

**package** dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** entity.Desk;

**public** **class** deskDAOImpl **implements** IBaseDAO<Desk>{

**boolean** isSuccess = **false**;

PreparedStatement ps = **null**;

ResultSet rs = **null**;

String sql = **null**;

Connection conn = JDBCConnection.*getConn*();

**public** ResultSet getFreeDesk()

{

sql = "select no from desk where status = '空闲'";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

**public** ResultSet getBusyDesk()

{

sql = "select no from desk where status = '就餐中'";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

**public** **boolean** updateBusyStatus(**int** id)

{

sql = "update desk set status = '就餐中' "

+ "where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, id);

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

**public** **boolean** updateFreeStatus(String deskNo)

{

sql = "update desk set status = '空闲' "

+ "where no = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, deskNo);

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** ResultSet findAll() **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from desk";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryById(**int** Id) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from desk where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, Id);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryByName(String deskNo) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from desk where no = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, deskNo);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** **boolean** add(Desk entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "insert into desk(no, seating, status)"

+ "values(?,?,?)";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, entity.getNo());

ps.setInt(2, entity.getSeating());

ps.setString(3, entity.getStatus());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** delete(Desk entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "delete from desk where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** update(Desk entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "update desk set no = ?, seating = ?, status = ?"

+ "where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, entity.getNo());

ps.setInt(2, entity.getSeating());

ps.setString(3, entity.getStatus());

ps.setInt(4, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

}

餐桌工具类

**package** util;

**import** dao.deskDAOImpl;

**public** **class** DeskDaoFactory {

**public** deskDAOImpl getDeskDao()

{

**return** **new** deskDAOImpl();

}

}

**package** util;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.table.AbstractTableModel;

**import** dao.deskDAOImpl;

**import** entity.Desk;

**public** **class** DeskTableModel **extends** AbstractTableModel{

**private** List<String> columnNames;

**private** List<List<String>> rowDates;

**private** DeskDaoFactory df;

**private** deskDAOImpl impl;

**public** DeskTableModel() {

df = **new** DeskDaoFactory();

impl = df.getDeskDao();

columnNames = **new** ArrayList<String>();

columnNames.add("餐台编号");

columnNames.add("座位数");

columnNames.add("状态");

}

**public** **void** findAll()

{

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.findAll();

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(2));

row.add(rs.getString(3));

row.add(rs.getString(4));

rowDates.add(row);

}

}**catch**(SQLException e){

e.printStackTrace();

}

}

**public** **boolean** queryModelByName(String name) {

**boolean** flag = **true**;

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.queryByName(name);

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(2));

row.add(rs.getString(3));

row.add(rs.getString(4));

rowDates.add(row);

}

}**catch**(Exception e){

flag = **false**;

}

**return** flag;

}

**public** List<String> getFreeDesk()

{

List<String> desk = **new** ArrayList<String>();

**try** {

ResultSet rs = impl.getFreeDesk();

**while** (rs.next())

{

desk.add(rs.getString(1));

}

}**catch** (Exception e) {

e.printStackTrace();

}

**return** desk;

}

**public** List<String> getBusyDesk()

{

List<String> desk = **new** ArrayList<String>();

**try** {

ResultSet rs = impl.getBusyDesk();

**while** (rs.next())

{

desk.add(rs.getString(1));

}

}**catch** (Exception e) {

e.printStackTrace();

}

**return** desk;

}

**public** **int** getDeskId(String deskNo)

{

**int** id = 0;

**try** {

ResultSet rs = impl.queryByName(deskNo);

**while** (rs.next()) {

id = Integer.*parseInt*(rs.getString(1));

}

}**catch**(SQLException e){

e.printStackTrace();

}

**return** id;

}

**public** String getDeskNo(**int** id)

{

String deskNo = **null**;

**try** {

ResultSet rs = impl.queryById(id);

**while** (rs.next()) {

deskNo = rs.getString(2);

}

}**catch**(SQLException e){

e.printStackTrace();

}

**return** deskNo;

}

**public** **boolean** addModel(Desk ee) **throws** Exception {

**return** impl.add(ee);

}

**public** **boolean** deleteModel(Desk ee) **throws** Exception{

**return** impl.delete(ee);

}

**public** **boolean** updateModel(Desk ee) **throws** Exception{

**return** impl.update(ee);

}

**public** **boolean** updateBusyStatus(**int** id) {

**return** impl.updateBusyStatus(id);

}

**public** **boolean** updateFreeStatus(String deskNo)

{

**return** impl.updateFreeStatus(deskNo);

}

**public** List<String> getAllCustomer()

{

CustomerTableModel cModel = **new** CustomerTableModel();

**return** cModel.getAllname();

}

**public** String getCustomerName(**int** id)

{

CustomerTableModel cModel = **new** CustomerTableModel();

**return** cModel.getCustomerName(id);

}

**public** **int** getCustomerId(String cName)

{

CustomerTableModel cModel = **new** CustomerTableModel();

**return** cModel.getCustomerId(cName);

}

@Override

**public** **int** getColumnCount() {

// **TODO** Auto-generated method stub

**return** **this**.columnNames.size();

}

@Override

**public** **int** getRowCount() {

// **TODO** Auto-generated method stub

**return** **this**.rowDates.size();

}

@Override

**public** Object getValueAt(**int** row, **int** col)

{

// **TODO** Auto-generated method stub

**if**(!rowDates.isEmpty())

**return** ((List<String>)**this**.rowDates.get(row)).get(col);

**else**

**return** **null**;

}

**public** String getColumnName(**int** column)

{

// **TODO** Auto-generated method stub

**return** (String)**this**.columnNames.get(column);

}

}

餐桌界面类

**package** boundry;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** javax.swing.JButton;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextField;

**import** javax.swing.SwingConstants;

**import** util.DeskTableModel;

**public** **class** DeskManagePane **extends** JPanel **implements** ActionListener{

//定义顶层标签

**private** JLabel titleLab;

//定义中间数据显示模块

**private** DeskTableModel deskModel;

**private** JTable jt;

**private** JScrollPane eePane;

//定义CRUD操作的板、标签、文本框及按钮

**private** JPanel contentPanel;

**private** JLabel no;

**private** JLabel seating;

**private** JLabel status;

**private** JTextField noTxt = **new** JTextField();

**private** JTextField seatingTxt = **new** JTextField();

**private** JTextField statusTxt = **new** JTextField();

**private** JButton queryBtn;

**private** JButton addBtn;

**private** JButton deleteBtn;

**private** JButton updateBtn;

**private** JButton clearBtn;

/\*\*

\* Create the panel.

\*/

**public** DeskManagePane() {

//设置布局及最佳大小

setLayout(**null**);

setPreferredSize(**new** Dimension(1395, 797));

//顶层标签

titleLab = **new** JLabel("餐台信息管理面板");

titleLab.setHorizontalAlignment(SwingConstants.***CENTER***);

titleLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 25));

titleLab.setBounds(367, 10, 257, 54);

add(titleLab);

//中间数据显示模块

deskModel = **new** DeskTableModel();

jt = **new** JTable(deskModel);

jt.setRowSelectionAllowed(**true**);

eePane = **new** JScrollPane();

eePane.setBounds(0, 74, 894, 625);

eePane.setViewportView(jt);

refreshModel();

add(eePane);

//添加鼠标点击事件，选中二维表某一行后将数据显示在右边的操作文本框中。

jt.addMouseListener(**new** MouseListener() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

// **TODO** Auto-generated method stub

**int** row = jt.getSelectedRow();

noTxt.setText((String)deskModel.getValueAt(row, 0));

seatingTxt.setText((String)deskModel.getValueAt(row, 1));

statusTxt.setText((String)deskModel.getValueAt(row, 2));

}

@Override

**public** **void** mousePressed(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseReleased(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseEntered(MouseEvent e) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseExited(MouseEvent e) {

// **TODO** Auto-generated method stub

}

});

//CRUD操作模块

contentPanel = **new** JPanel();

contentPanel.setBounds(896, 74, 430, 626);

contentPanel.setLayout(**null**);

no = **new** JLabel("餐台编号：");

no.setBounds(5, 35, 80, 25);

contentPanel.add(no);

noTxt.setBounds(86, 35, 332, 25);

contentPanel.add(noTxt);

seating = **new** JLabel("座位数：");

seating.setBounds(5, 65, 80, 25);

contentPanel.add(seating);

seatingTxt.setBounds(86, 65, 332, 25);

contentPanel.add(seatingTxt);

status = **new** JLabel("状态：");

status.setBounds(5, 95, 80, 25);

contentPanel.add(status);

statusTxt.setBounds(86, 95, 332, 25);

contentPanel.add(statusTxt);

queryBtn = **new** JButton("查询");

queryBtn.setBounds(5, 224, 413, 65);

contentPanel.add(queryBtn);

addBtn = **new** JButton("添加");

addBtn.setBounds(5, 299, 413, 65);

contentPanel.add(addBtn);

deleteBtn = **new** JButton("删除");

deleteBtn.setBounds(5, 374, 413, 65);

contentPanel.add(deleteBtn);

updateBtn = **new** JButton("修改");

updateBtn.setBounds(5, 449, 413, 65);

contentPanel.add(updateBtn);

clearBtn = **new** JButton("清空");

clearBtn.setBounds(5,524,413,64);

contentPanel.add(clearBtn);

queryBtn.addActionListener(**this**);

updateBtn.addActionListener(**this**);

deleteBtn.addActionListener(**this**);

addBtn.addActionListener(**this**);

clearBtn.addActionListener(**this**);

add(contentPanel);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getSource() == queryBtn)//点击查询按钮

{

String no = noTxt.getText().trim();//获取姓名文本框内容

**if** (no.length() != 0)//姓名输入有效，执行查询

{

deskModel = **new** DeskTableModel();

**if**(deskModel.queryModelByName(no) == **false**)

{

JOptionPane.*showMessageDialog*(**this**, "查无此人！");

}

jt.setModel(deskModel);//更新数据模块

} **else** {

refreshModel();//姓名输入为空，更新数据模块

}

}

**else** **if**(e.getSource() == addBtn)//点击添加按钮

{

entity.Desk desk = **new** entity.Desk();//获取文本框内容

desk.setNo(noTxt.getText());

desk.setSeating(Integer.*parseInt*(seatingTxt.getText()));

desk.setStatus(statusTxt.getText());

**try** {

**if** (deskModel.addModel(desk))//添加操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "添加成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if**(e.getSource() == deleteBtn)//点击删除按钮

{

**int** row = jt.getSelectedRow();//获取选中的行号

**if**(row == -1) {

JOptionPane.*showMessageDialog*(**this**, "请选择一行");

**return** ;

}

**int** id = deskModel.getDeskId((String) deskModel.getValueAt(row, 0));//获取编号

DeskTableModel temp = **new** DeskTableModel();

entity.Desk desk = **new** entity.Desk();

desk.setId(id);

**try** {

**if**(temp.deleteModel(desk))//删除操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "删除成功！");

}

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

refreshModel();

}

**else** **if** (e.getSource() == updateBtn)//点击修改按钮

{

entity.Desk desk = **new** entity.Desk();//获取文本框内容

desk.setId(deskModel.getDeskId(noTxt.getText()));

desk.setNo(noTxt.getText());

desk.setSeating(Integer.*parseInt*(seatingTxt.getText()));

desk.setStatus(statusTxt.getText());

**try** {

**if** (deskModel.updateModel(desk))//修改操作有效，更新数据模块

{

JOptionPane.*showMessageDialog*(**this**, "修改成功！");

refreshModel();

}

}**catch** (Exception e1) {

// **TODO**: handle exception

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == clearBtn)

{

jt.getSelectionModel().clearSelection();

noTxt.setText("");

seatingTxt.setText("");

statusTxt.setText("");

}

}

//更新数据显示模块的内容函数

**public** **void** refreshModel() {

deskModel = **new** DeskTableModel();

deskModel.findAll();

jt.setModel(deskModel);

}

}

## 3.8 开台模块设计说明

开台订单实体类

**package** entity;

**public** **class** Order {

**private** **int** id;

**private** String orderNo;

**private** **int** deskId;

**private** String createtime;

**private** **double** money;

**private** **int** customerId;

**private** String status;

**private** **int** number;

**public** Order() {}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getOrderNo() {

**return** orderNo;

}

**public** **void** setOrderNo(String orderNo) {

**this**.orderNo = orderNo;

}

**public** **int** getDeskId() {

**return** deskId;

}

**public** **void** setDeskId(**int** deskId) {

**this**.deskId = deskId;

}

**public** String getCreatetime() {

**return** createtime;

}

**public** **void** setCreatetime(String createtime) {

**this**.createtime = createtime;

}

**public** **double** getMoney() {

**return** money;

}

**public** **void** setMoney(**double** money) {

**this**.money = money;

}

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** **int** getNumber() {

**return** number;

}

**public** **void** setNumber(**int** number) {

**this**.number = number;

}

}

开台操作类

**package** dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** entity.Order;

**public** **class** OrderDAOImpl **implements** IBaseDAO<Order>{

**boolean** isSuccess = **false**;

PreparedStatement ps = **null**;

ResultSet rs = **null**;

String sql = **null**;

Connection conn = JDBCConnection.*getConn*();

**public** ResultSet getOrderByDeskNo(String deskNo)

{

sql = "select o.id, o.money, o.orderNo from desk d left join `order` o on d.id = o.deskId "

+ " where o.status = '未支付' and d.status = '就餐中' and d.no = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1,deskNo);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

**public** **boolean** updateMoney(**double** money, **int** id)

{

sql = "update `order` set money = ? where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setDouble(1, money);

ps.setInt(2, id);

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

**public** **boolean** updateStatus(String orderNo)

{

sql = "update `order` set status = '已支付'"

+ "where orderNo = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, orderNo);

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** ResultSet findAll() **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select o.orderNo, d.no, o.createtime, o.money, c.name, o.status, o.number from `order` o,customer c,desk d "

+ "where o.deskId = d.id and o.customerId = c.id";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryById(**int** Id) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from `order` where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, Id);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryByName(String orderNo) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from `order` where orderNo = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, orderNo);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** **boolean** add(Order entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "insert into `order`(orderNo, deskId, createtime, money, customerId, status, number)"

+ "values(?,?,?,?,?,?,?)";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, entity.getOrderNo());

ps.setInt(2, entity.getDeskId());

ps.setString(3, entity.getCreatetime());

ps.setDouble(4, entity.getMoney());

ps.setInt(5, entity.getCustomerId());

ps.setString(6, entity.getStatus());

ps.setInt(7, entity.getNumber());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** delete(Order entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "delete from `order` where orderNo = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, entity.getOrderNo());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** update(Order entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "update `order` set orderNo = ?, deskId = ?, createtime = ?, money = ?, customerId = ?, status = ?, number = ?"

+ "where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, entity.getOrderNo());

ps.setInt(2, entity.getDeskId());

ps.setString(3, entity.getCreatetime());

ps.setDouble(4, entity.getMoney());

ps.setInt(5, entity.getCustomerId());

ps.setString(6, entity.getStatus());

ps.setInt(7, entity.getNumber());

ps.setInt(8, entity.getId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

}

开台工具类

**package** util;

**import** dao.OrderDAOImpl;

**public** **class** OrderDaoFactory {

**public** OrderDAOImpl getOrderDao()

{

**return** **new** OrderDAOImpl();

}

}

**package** util;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.table.AbstractTableModel;

**import** dao.OrderDAOImpl;

**import** entity.Order;

**public** **class** OrderTableModel **extends** AbstractTableModel{

**private** List<String> columnNames;

**private** List<List<String>> rowDates = **new** ArrayList<List<String>>();

**private** OrderDaoFactory df;

**private** OrderDAOImpl impl;

**public** OrderTableModel() {

df = **new** OrderDaoFactory();

impl = df.getOrderDao();

columnNames = **new** ArrayList<String>();

columnNames.add("订单编号");

columnNames.add("餐台号");

columnNames.add("就餐日期");

columnNames.add("金额");

columnNames.add("客户编号");

columnNames.add("状态");

columnNames.add("就餐人数");

}

**public** **void** findAll()

{

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.findAll();

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

row.add(rs.getString(4));

row.add(rs.getString(5));

row.add(rs.getString(6));

row.add(rs.getString(7));

rowDates.add(row);

}

}**catch**(SQLException e){

e.printStackTrace();

}

}

**public** **boolean** queryModelByName(String name) {

**boolean** flag = **true**;

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.queryByName(name);

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

row.add(rs.getString(3));

row.add(rs.getString(4));

row.add(rs.getString(5));

row.add(rs.getString(6));

row.add(rs.getString(7));

rowDates.add(row);

}

}**catch**(Exception e){

flag = **false**;

}

**return** flag;

}

**public** **boolean** addModel(Order ee) **throws** Exception {

**return** impl.add(ee);

}

**public** **boolean** deleteModel(Order ee) **throws** Exception{

**return** impl.delete(ee);

}

**public** **boolean** updateModel(Order ee) **throws** Exception{

**return** impl.update(ee);

}

**public** **boolean** updateMoney(**double** money, **int** orderId)

{

**return** impl.updateMoney(money, orderId);

}

**public** **boolean** updateStatus(String orderNo)

{

**return** impl.updateStatus(orderNo);

}

**public** **double** getMoney(String deskNo) **throws** SQLException

{

**double** money = 0;

**try** {

ResultSet rs = impl.getOrderByDeskNo(deskNo);

**while**(rs.next()) {

money = rs.getDouble(2);

}

} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** money;

}

**public** **int** getOrderId(String deskNo)

{

**int** id = 0;

**try** {

ResultSet rs = impl.getOrderByDeskNo(deskNo);

**while**(rs.next()) {

id = rs.getInt(1);

}

} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** id;

}

@Override

**public** **int** getColumnCount() {

// **TODO** Auto-generated method stub

**return** **this**.columnNames.size();

}

@Override

**public** **int** getRowCount() {

// **TODO** Auto-generated method stub

**return** **this**.rowDates.size();

}

@Override

**public** Object getValueAt(**int** row, **int** col)

{

// **TODO** Auto-generated method stub

**if**(!rowDates.isEmpty())

**return** ((List<String>)**this**.rowDates.get(row)).get(col);

**else**

**return** **null**;

}

**public** String getColumnName(**int** column)

{

// **TODO** Auto-generated method stub

**return** (String)**this**.columnNames.get(column);

}

}

**package** util;

**import** java.text.DateFormat;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** ChangeOrderEvent {

**private** entity.Order order = **new** entity.Order();

**private** Date date;

**private** OrderTableModel orderModel = **new** OrderTableModel();

**private** DeskTableModel deskModel = **new** DeskTableModel();

**public** ChangeOrderEvent(){}

**public** **void** init\_order(**int** deskId, **int** cId, **int** num) **throws** Exception

{

date = **new** Date();

DateFormat dft = **new** SimpleDateFormat("yyyy-MM-dd");

DateFormat dfNo = **new** SimpleDateFormat("yyyyMMddHHmmss");

order.setCreatetime(dft.format(date));

order.setOrderNo(dfNo.format(date) + (**int**)((Math.*random*()\*9+1)\*1000));

order.setDeskId(deskId);

order.setCustomerId(cId);

order.setMoney(0);

order.setStatus("未支付");

order.setNumber(num);

orderModel.addModel(order);

deskModel.updateBusyStatus(deskId);

}

**public** **void** finish\_order(String orderNo, String deskNo)

{

orderModel.updateStatus(orderNo);

deskModel.updateFreeStatus(deskNo);

}

}

开台界面类

**package** boundry;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.JButton;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextField;

**import** javax.swing.SwingConstants;

**import** util.DeskTableModel;

**import** util.ChangeOrderEvent;

**import** javax.swing.JComboBox;

**public** **class** OrderManagePane **extends** JPanel **implements** ActionListener{

//定义顶层标签及订单客户信息模块

**private** JLabel titleLab;

**private** JPanel topPanel;

//定义中间数据显示模块

**private** DeskTableModel deskModel = **new** DeskTableModel();

**private** JTable jt;

**private** JScrollPane eePane;

**private** JComboBox<String> cIdBox;

JButton OrderBtn;

**private** JTextField textField;

/\*\*

\* Create the panel.

\*

\*/

**public** OrderManagePane(){

//设置布局及最佳大小

setLayout(**null**);

setPreferredSize(**new** Dimension(1395, 797));

//顶层标签及订单客户信息模块

titleLab = **new** JLabel("开台管理面板");

titleLab.setHorizontalAlignment(SwingConstants.***CENTER***);

titleLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 25));

titleLab.setBounds(420, 10, 257, 54);

add(titleLab);

topPanel = **new** JPanel();

topPanel.setBounds(10, 74, 1050, 65);

add(topPanel);

topPanel.setLayout(**null**);

//中间数据显示模块

eePane = **new** JScrollPane();

eePane.setBounds(10, 149, 1050, 570);

refreshOrderPane();

add(eePane);

JLabel cId = **new** JLabel("客户：");

cId.setBounds(10, 10, 62, 42);

topPanel.add(cId);

cId.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

cIdBox = **new** JComboBox<String>();

cIdBox.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

cIdBox.setBounds(82, 13, 88, 42);

fillBox();

cIdBox.setSelectedIndex(-1);

topPanel.add(cIdBox);

JLabel number = **new** JLabel("就餐人数：");

number.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

number.setBounds(399, 6, 114, 46);

topPanel.add(number);

textField = **new** JTextField();

textField.setBounds(506, 10, 88, 42);

topPanel.add(textField);

OrderBtn = **new** JButton("开台");

OrderBtn.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

OrderBtn.setBounds(667, 10, 241, 42);

OrderBtn.addActionListener(**this**);

topPanel.add(OrderBtn);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

**if** (e.getSource() == OrderBtn)

{

**int** num = Integer.*parseInt*(textField.getText());

**int** row = jt.getSelectedRow();//获取选中的行号

**if**(row == -1 || deskModel.getValueAt(row, 2).equals("就餐中")) {

JOptionPane.*showMessageDialog*(**this**, "请选择一个空闲餐桌");

**return** ;

}

**else** **if**(Integer.*parseInt*((String)deskModel.getValueAt(row, 1)) < num)

{

JOptionPane.*showMessageDialog*(**this**, "请选择更大的餐桌");

**return** ;

}

**int** deskId = deskModel.getDeskId((String) deskModel.getValueAt(row, 0));

**int** cId = deskModel.getCustomerId((String) cIdBox.getSelectedItem());

**try** {

**new** ChangeOrderEvent().init\_order(deskId, cId, num);;

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

refreshOrderPane();

}

}

**public** **void** fillBox()

{

List<String> customerList = **new** ArrayList<String>();

customerList = deskModel.getAllCustomer();

**for** (String temp : customerList)

{

cIdBox.addItem(temp);

}

}

**public** **void** refreshOrderPane()

{

deskModel = **new** DeskTableModel();

deskModel.findAll();

jt = **new** JTable();

jt.setModel(deskModel);

jt.setRowSelectionAllowed(**true**);

eePane.setViewportView(jt);

}

}

## 3.9 点菜及结账模块设计说明

点菜订单信息明细实体类

**package** entity;

**public** **class** OrderItem {

**private** **int** id;

**private** **int** orderId;

**private** **int** dishId;

**private** **double** amount;

**public** OrderItem() {}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **int** getOrderId() {

**return** orderId;

}

**public** **void** setOrderId(**int** orderId) {

**this**.orderId = orderId;

}

**public** **int** getDishId() {

**return** dishId;

}

**public** **void** setDishId(**int** dishId) {

**this**.dishId = dishId;

}

**public** **double** getAmount() {

**return** amount;

}

**public** **void** setAmount(**double** amount) {

**this**.amount = amount;

}

}

点菜操作类

**package** dao;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** entity.OrderItem;

**public** **class** OrderItemDAOImpl **implements** IBaseDAO<OrderItem>{

**boolean** isSuccess = **false**;

PreparedStatement ps = **null**;

ResultSet rs = **null**;

String sql = **null**;

Connection conn = JDBCConnection.*getConn*();

**public** ResultSet queryByDeskNo(String deskNo)

{

sql = "select d.name, oi.amount from orderitem oi, dish d, `order` o "

+ "where d.id = oi.dishId and o.id = oi.orderId and o.orderNo in "

+ "(select a.orderNo from `order` a,desk b where a.deskId = b.id and a.status = '未支付' and b.status = '就餐中' and b.no = ?)";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, deskNo);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

**public** ResultSet checkMoney(**int** orderId)

{

sql = "select SUM(oi.amount \* d.price) from orderitem oi, dish d where oi.dishId = d.id and orderId = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, orderId);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

**public** ResultSet getOrderNo(String deskNo)

{

sql = "select o.orderNo from desk d, `order` o where o.deskId = d.id and d.no = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, deskNo);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

**public** ResultSet getCname(String orderNo)

{

sql = "select c.name from customer c, `order` o where o.customerId = c.id and o.orderNo = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setString(1, orderNo);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

**public** **boolean** updateAmount(OrderItem entity)

{

sql = "update orderitem set amount = amount + 1 "

+ " where orderId = ?, dishId = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getOrderId());

ps.setInt(2, entity.getDishId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** ResultSet findAll() **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select d.name, oi.amount from orderitem oi, dish d where d.id = oi.dishId";

**try** {

ps = conn.prepareStatement(sql);

rs = ps.executeQuery();

}**catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryById(**int** Id) **throws** SQLException {

// **TODO** Auto-generated method stub

sql = "select \* from orderitem where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, Id);

rs = ps.executeQuery();

}**catch**(SQLException e) {

e.printStackTrace();

}

**return** rs;

}

@Override

**public** ResultSet queryByName(String name) **throws** SQLException {

// **TODO** Auto-generated method stub

**return** **null**;

}

@Override

**public** **boolean** add(OrderItem entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "insert into orderitem(orderId, dishId, amount)"

+ "values(?,?,?)";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getOrderId());

ps.setInt(2, entity.getDishId());

ps.setDouble(3, entity.getAmount());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** delete(OrderItem entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "delete from orderitem where orderId = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getOrderId());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

@Override

**public** **boolean** update(OrderItem entity) **throws** Exception {

// **TODO** Auto-generated method stub

sql = "update orderitem set orderId = ?, dishId = ?, amount = ?"

+ "where id = ?";

**try** {

ps = conn.prepareStatement(sql);

ps.setInt(1, entity.getId());

ps.setInt(2, entity.getOrderId());

ps.setInt(3, entity.getDishId());

ps.setDouble(4, entity.getAmount());

isSuccess = ps.executeUpdate() != 0 ? **true** : **false**;

}**catch**(Exception e) {

isSuccess = **false**;

e.printStackTrace();

}

**return** isSuccess;

}

}

点菜工具类

**package** util;

**import** dao.OrderItemDAOImpl;

**public** **class** OrderItemDaoFactory {

**public** OrderItemDAOImpl getOrderItemDao()

{

**return** **new** OrderItemDAOImpl();

}

}

**package** util;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.table.AbstractTableModel;

**import** dao.OrderItemDAOImpl;

**import** entity.OrderItem;

**public** **class** OrderItemTableModel **extends** AbstractTableModel{

**private** List<String> columnNames;

**private** List<List<String>> rowDates = **new** ArrayList<List<String>>();

**private** OrderItemDaoFactory df;

**private** OrderItemDAOImpl impl;

**public** OrderItemTableModel() {

df = **new** OrderItemDaoFactory();

impl = df.getOrderItemDao();

columnNames = **new** ArrayList<String>();

columnNames.add("菜品名");

columnNames.add("菜品数量");

}

**public** **void** findAll()

{

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.findAll();

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

rowDates.add(row);

}

}**catch**(SQLException e){

e.printStackTrace();

}

}

**public** **boolean** queryModelBydeskNo(String deskNo) {

**boolean** flag = **true**;

rowDates = **new** ArrayList<List<String>>();

**try** {

ResultSet rs = impl.queryByDeskNo(deskNo);

**while**(rs.next()) {

List<String> row = **new** ArrayList<String>();

row.add(rs.getString(1));

row.add(rs.getString(2));

rowDates.add(row);

}

}**catch**(Exception e){

flag = **false**;

}

**return** flag;

}

**public** **boolean** addModel(OrderItem ee) **throws** Exception {

**return** impl.add(ee);

}

**public** **boolean** deleteModel(OrderItem ee) **throws** Exception{

**return** impl.delete(ee);

}

**public** **boolean** updateModel(OrderItem ee) **throws** Exception{

**return** impl.update(ee);

}

**public** **boolean** updateAmount(OrderItem oi)

{

**return** impl.updateAmount(oi);

}

**public** **double** getMoney(**int** orderId)

{

**double** money = 0;

**try** {

ResultSet rs = impl.checkMoney(orderId);

**while** (rs.next()) {

money = rs.getDouble(1);

}

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** money;

}

**public** String getOrderNo(String deskNo)

{

String orderNo = **null**;

**try** {

ResultSet rs = impl.getOrderNo(deskNo);

**while** (rs.next()) {

orderNo = rs.getString(1);

}

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** orderNo;

}

**public** String getCname(String orderNo)

{

String cName = **null**;

**try** {

ResultSet rs = impl.getCname(orderNo);

**while** (rs.next()) {

cName = rs.getString(1);

}

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** cName;

}

@Override

**public** **int** getColumnCount() {

// **TODO** Auto-generated method stub

**return** **this**.columnNames.size();

}

@Override

**public** **int** getRowCount() {

// **TODO** Auto-generated method stub

**return** **this**.rowDates.size();

}

@Override

**public** Object getValueAt(**int** row, **int** col)

{

// **TODO** Auto-generated method stub

**if**(!rowDates.isEmpty())

**return** ((List<String>)**this**.rowDates.get(row)).get(col);

**else**

**return** **null**;

}

**public** String getColumnName(**int** column)

{

// **TODO** Auto-generated method stub

**return** (String)**this**.columnNames.get(column);

}

}

**package** util;

**import** java.text.DateFormat;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** ChangeOrderEvent {

**private** entity.Order order = **new** entity.Order();

**private** Date date;

**private** OrderTableModel orderModel = **new** OrderTableModel();

**private** DeskTableModel deskModel = **new** DeskTableModel();

**public** ChangeOrderEvent(){}

**public** **void** init\_order(**int** deskId, **int** cId, **int** num) **throws** Exception

{

date = **new** Date();

DateFormat dft = **new** SimpleDateFormat("yyyy-MM-dd");

DateFormat dfNo = **new** SimpleDateFormat("yyyyMMddHHmmss");

order.setCreatetime(dft.format(date));

order.setOrderNo(dfNo.format(date) + (**int**)((Math.*random*()\*9+1)\*1000));

order.setDeskId(deskId);

order.setCustomerId(cId);

order.setMoney(0);

order.setStatus("未支付");

order.setNumber(num);

orderModel.addModel(order);

deskModel.updateBusyStatus(deskId);

}

**public** **void** finish\_order(String orderNo, String deskNo)

{

orderModel.updateStatus(orderNo);

deskModel.updateFreeStatus(deskNo);

}

}

**package** util;

**public** **class** ChangeDishOrderEvent {

**int** orderId;

**int** dishId;

**double** money;

entity.OrderItem orderitem = **new** entity.OrderItem();

OrderTableModel oM = **new** OrderTableModel();

OrderItemTableModel oiM = **new** OrderItemTableModel();

**public** ChangeDishOrderEvent(){}

**public** ChangeDishOrderEvent(**int** orderId, **int** dishId)

{

**this**.orderId = orderId;

**this**.dishId = dishId;

orderitem.setOrderId(orderId);

orderitem.setDishId(dishId);

orderitem.setAmount(1);

}

**public** **void** addDish() **throws** Exception

{

oiM.addModel(orderitem);

money = oiM.getMoney(orderId);

oM.updateMoney(money, orderId);

}

**public** **void** deleteDish() **throws** Exception

{

oiM.deleteModel(orderitem);

money = oiM.getMoney(orderId);

oM.updateMoney(money, orderId);

}

}

点菜界面类

**package** boundry;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.ItemEvent;

**import** java.awt.event.ItemListener;

**import** java.io.IOException;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.swing.JButton;

**import** javax.swing.JComboBox;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.SwingConstants;

**import** util.OrderTableModel;

**import** util.ChangeDishOrderEvent;

**import** util.DishTableModel;

**import** util.OrderItemTableModel;

**import** java.awt.Color;

**public** **class** DishesOrderesManagePane **extends** JPanel **implements** ActionListener{

**private** JLabel titleLab;

**private** JLabel dishId;

**private** JComboBox<String> dishBox;

**private** JPanel contentPane;

**private** JScrollPane dishPane = **new** JScrollPane();

**private** JLabel title2Lab;

**private** JScrollPane oiPane = **new** JScrollPane();

**private** DishTableModel dmodel = **new** DishTableModel();

**private** OrderItemTableModel omodel = **new** OrderItemTableModel();

**private** JTable jt\_dish;

**private** JTable jt\_orderitem;

**private** JLabel deskId;

**private** JComboBox<String> deskBox;

**private** JButton addBtn;

**private** JButton deleteBtn;

**private** JButton checkBtn;

**private** JLabel orderNoLab;

**private** JLabel orderNo;

**private** JLabel moneyLab;

**private** JLabel money;

**private** OrderTableModel tempModel = **new** OrderTableModel();

/\*\*

\* Create the panel.

\* **@throws** IOException

\*/

**public** DishesOrderesManagePane() **throws** IOException {

//设置布局及最佳大小

setLayout(**null**);

setPreferredSize(**new** Dimension(1279, 765));

//顶层标签

titleLab = **new** JLabel("点菜管理面板");

titleLab.setHorizontalAlignment(SwingConstants.***CENTER***);

titleLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 25));

titleLab.setBounds(421, 10, 257, 54);

add(titleLab);

dishId = **new** JLabel("菜品类别");

dishId.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

dishId.setBounds(10, 63, 98, 36);

add(dishId);

dishBox = **new** JComboBox<String>();

dishBox.setBounds(118, 63, 88, 42);

add(dishBox);

dishPane.setBounds(10, 109, 897, 336);

refreshDishPane();

add(dishPane);

title2Lab = **new** JLabel("已点菜品");

title2Lab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

title2Lab.setBounds(10, 441, 98, 36);

add(title2Lab);

oiPane.setBounds(10, 475, 897, 281);

omodel = **new** OrderItemTableModel();

jt\_orderitem = **new** JTable(omodel);

oiPane.setViewportView(jt\_orderitem);

add(oiPane);

contentPane = **new** JPanel();

contentPane.setLayout(**null**);

contentPane.setBounds(910, 109, 359, 646);

addBtn = **new** JButton("添加菜品");

addBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

addBtn.setBounds(10, 194, 330, 65);

addBtn.addActionListener(**this**);

contentPane.add(addBtn);

deleteBtn = **new** JButton("删除菜品");

deleteBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

deleteBtn.setBounds(10, 269, 330, 65);

deleteBtn.addActionListener(**this**);

contentPane.add(deleteBtn);

checkBtn = **new** JButton("结账");

checkBtn.setFont(**new** Font("宋体", Font.***BOLD***, 15));

checkBtn.setBounds(10, 572, 330, 65);

checkBtn.addActionListener(**this**);

contentPane.add(checkBtn);

orderNoLab = **new** JLabel("当前订单编号");

orderNoLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

orderNoLab.setLocation(10, 62);

orderNoLab.setSize(160, 52);

contentPane.add(orderNoLab);

orderNo = **new** JLabel();

orderNo.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

orderNo.setLocation(10, 112);

orderNo.setSize(330, 52);

contentPane.add(orderNo);

deskId = **new** JLabel("餐台：");

deskId.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

deskId.setBounds(10, 10, 62, 42);

contentPane.add(deskId);

deskBox = **new** JComboBox<String>();

deskBox.setBounds(82, 13, 88, 42);

contentPane.add(deskBox);

add(contentPane);

fillBox();

dishBox.setSelectedIndex(-1);

dishBox.addItemListener(**new** ItemListener() {

@Override

**public** **void** itemStateChanged(ItemEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getStateChange() == ItemEvent.***SELECTED***) {

String dish = (String) dishBox.getSelectedItem();

**try** {

dmodel = **new** DishTableModel();

dmodel.queryModelByCate(dish);

jt\_dish = **new** JTable();

jt\_dish.setFont(**new** Font("楷体", Font.***PLAIN***, 20));

jt\_dish.setModel(dmodel);

jt\_dish.getColumnModel().getColumn(2).setCellRenderer(**new** ImageRenderer());

jt\_dish.setRowHeight(80);

jt\_dish.setRowSelectionAllowed(**true**);

dishPane.setViewportView(jt\_dish);

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

}

});

deskBox.setSelectedIndex(-1);

deskBox.addItemListener(**new** ItemListener() {

@Override

**public** **void** itemStateChanged(ItemEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getStateChange() == ItemEvent.***SELECTED***)

{

**try** {

refreshOrderItemPane((String) deskBox.getSelectedItem());

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

}

});

moneyLab = **new** JLabel("总金额");

moneyLab.setFont(**new** Font("华文仿宋", Font.***BOLD***, 20));

moneyLab.setBounds(10, 365, 160, 52);

contentPane.add(moneyLab);

money = **new** JLabel();

money.setForeground(**new** Color(0, 0, 0));

money.setBounds(10, 427, 330, 59);

contentPane.add(money);

}

**public** **void** fillBox()

{

List<String> ListCate = **new** ArrayList<String>();

List<String> ListDesk = **new** ArrayList<String>();

util.DeskTableModel m = **new** util.DeskTableModel();

ListCate = dmodel.getAllCategory();

ListDesk = m.getBusyDesk();

**for** (String temp : ListCate)

{

dishBox.addItem(temp);

}

**for** (String temp : ListDesk) {

deskBox.addItem(temp);

}

}

**public** **void** refreshDishPane() **throws** IOException

{

dmodel = **new** DishTableModel();

dmodel.findAll();

jt\_dish = **new** JTable();

jt\_dish.setFont(**new** Font("楷体", Font.***PLAIN***, 20));

jt\_dish.setModel(dmodel);

jt\_dish.getColumnModel().getColumn(2).setCellRenderer(**new** ImageRenderer());

jt\_dish.setRowHeight(80);

jt\_dish.setRowSelectionAllowed(**true**);

dishPane.setViewportView(jt\_dish);

}

**public** **void** refreshOrderItemPane(String deskNo) **throws** SQLException

{

omodel = **new** OrderItemTableModel();

omodel.queryModelBydeskNo(deskNo);

jt\_orderitem = **new** JTable();

jt\_orderitem.setModel(omodel);

jt\_orderitem.setRowSelectionAllowed(**true**);

oiPane.setViewportView(jt\_orderitem);

money.setText(String.*valueOf*((tempModel.getMoney(deskNo))));

orderNo.setText((String) omodel.getOrderNo(deskNo));

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getSource() == addBtn)

{

**int** row = jt\_dish.getSelectedRow();//获取选中的行号

**if**(row == -1) {

JOptionPane.*showMessageDialog*(**this**, "请选择一项菜品");

**return** ;

}

**int** dishId = dmodel.getDishIdByName((String)jt\_dish.getValueAt(row, 0));

**if** ((String) deskBox.getSelectedItem() == **null**) {

JOptionPane.*showMessageDialog*(**this**, "请选择一个就餐中的餐台");

**return** ;

}

**int** orderId = tempModel.getOrderId((String)deskBox.getSelectedItem());

**try** {

**new** ChangeDishOrderEvent(orderId, dishId).addDish();

refreshOrderItemPane((String) deskBox.getSelectedItem());

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == deleteBtn)

{

**int** row = jt\_orderitem.getSelectedRow();//获取选中的行号

**if**(row == -1) {

JOptionPane.*showMessageDialog*(**this**, "请选择一项菜品");

**return** ;

}

**int** dishId = dmodel.getDishIdByName((String)jt\_orderitem.getValueAt(row, 0));

**if** ((String) deskBox.getSelectedItem() == **null**) {

JOptionPane.*showMessageDialog*(**this**, "请选择一个就餐中的餐台");

**return** ;

}

**int** orderId = tempModel.getOrderId((String)deskBox.getSelectedItem());

**try** {

**new** ChangeDishOrderEvent(orderId, dishId).deleteDish();

refreshOrderItemPane((String) deskBox.getSelectedItem());

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

**else** **if** (e.getSource() == checkBtn)

{

**if** ((String) deskBox.getSelectedItem() == **null**) {

JOptionPane.*showMessageDialog*(**this**, "请选择一个就餐中的餐台");

**return** ;

}

String deskNo\_ = (String)deskBox.getSelectedItem();

String cName\_ = omodel.getCname(orderNo.getText());

String money\_ = money.getText();

String orderNo\_ = orderNo.getText();

**new** GiveChangeDialog(deskNo\_, cName\_, money\_, orderNo\_);

}

}

}

**package** boundry;

**import** java.awt.BorderLayout;

**import** javax.swing.JButton;

**import** javax.swing.JDialog;

**import** javax.swing.JPanel;

**import** javax.swing.border.EmptyBorder;

**import** util.ChangeOrderEvent;

**import** javax.swing.SwingConstants;

**import** javax.swing.JLabel;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.JTextField;

**import** java.awt.Color;

**public** **class** GiveChangeDialog **extends** JDialog **implements** ActionListener{

**private** **final** JPanel contentPanel = **new** JPanel();

**private** JTextField realText;

**private** JTextField offText;

**private** **double** sMoney;

**private** JLabel change = **new** JLabel();

**private** String orderNo;

**private** String deskNo;

/\*\*

\* Create the dialog.

\*/

**public** GiveChangeDialog(String deskNo, String cName, String sumMoney, String orderNo) {

setTitle("结账信息");

setBounds(100, 100, 638, 486);

getContentPane().setLayout(**new** BorderLayout());

contentPanel.setBorder(**new** EmptyBorder(5, 5, 5, 5));

getContentPane().add(contentPanel, BorderLayout.***CENTER***);

**this**.sMoney = Double.*parseDouble*(sumMoney);

**this**.orderNo = orderNo;

**this**.deskNo = deskNo;

contentPanel.setLayout(**null**);

{

JLabel deskLab = **new** JLabel("餐桌");

deskLab.setHorizontalAlignment(SwingConstants.***CENTER***);

deskLab.setFont(**new** Font("宋体", Font.***BOLD***, 20));

deskLab.setBounds(10, 10, 110, 37);

contentPanel.add(deskLab);

}

{

JLabel customerLab = **new** JLabel("客户");

customerLab.setHorizontalAlignment(SwingConstants.***CENTER***);

customerLab.setFont(**new** Font("宋体", Font.***BOLD***, 20));

customerLab.setBounds(307, 10, 110, 37);

contentPanel.add(customerLab);

}

{

JLabel desk = **new** JLabel(deskNo);

desk.setHorizontalAlignment(SwingConstants.***CENTER***);

desk.setFont(**new** Font("宋体", Font.***BOLD***, 20));

desk.setBounds(130, 10, 167, 37);

contentPanel.add(desk);

}

{

JLabel customer = **new** JLabel(cName);

customer.setHorizontalAlignment(SwingConstants.***CENTER***);

customer.setFont(**new** Font("宋体", Font.***BOLD***, 20));

customer.setBounds(427, 10, 167, 37);

contentPanel.add(customer);

}

{

JLabel offLab = **new** JLabel("优惠：");

offLab.setHorizontalAlignment(SwingConstants.***CENTER***);

offLab.setFont(**new** Font("宋体", Font.***BOLD***, 20));

offLab.setBounds(76, 153, 110, 37);

contentPanel.add(offLab);

}

{

JLabel realLab = **new** JLabel("实收：");

realLab.setHorizontalAlignment(SwingConstants.***CENTER***);

realLab.setFont(**new** Font("宋体", Font.***BOLD***, 25));

realLab.setBounds(76, 200, 134, 53);

contentPanel.add(realLab);

}

realText = **new** JTextField();

realText.setFont(**new** Font("宋体", Font.***PLAIN***, 25));

realText.setBounds(213, 200, 181, 53);

contentPanel.add(realText);

realText.setColumns(10);

JLabel moneyLab = **new** JLabel("应收：");

moneyLab.setHorizontalAlignment(SwingConstants.***CENTER***);

moneyLab.setFont(**new** Font("宋体", Font.***BOLD***, 20));

moneyLab.setBounds(76, 106, 110, 37);

contentPanel.add(moneyLab);

JLabel money = **new** JLabel(sumMoney);

money.setHorizontalAlignment(SwingConstants.***CENTER***);

money.setFont(**new** Font("宋体", Font.***BOLD***, 20));

money.setBounds(186, 106, 167, 37);

contentPanel.add(money);

JLabel changeLab = **new** JLabel("找零");

changeLab.setHorizontalAlignment(SwingConstants.***CENTER***);

changeLab.setFont(**new** Font("宋体", Font.***BOLD***, 30));

changeLab.setBounds(76, 292, 134, 53);

contentPanel.add(changeLab);

change = **new** JLabel("");

change.setForeground(Color.***RED***);

change.setHorizontalAlignment(SwingConstants.***CENTER***);

change.setFont(**new** Font("宋体", Font.***BOLD***, 30));

change.setBounds(213, 292, 204, 53);

contentPanel.add(change);

{

JButton okButton = **new** JButton("确定");

okButton.setBounds(176, 400, 72, 37);

contentPanel.add(okButton);

okButton.setActionCommand("ok");

okButton.addActionListener(**this**);

getRootPane().setDefaultButton(okButton);

}

{

JButton cancelButton = **new** JButton("取消");

cancelButton.setBounds(307, 400, 72, 37);

contentPanel.add(cancelButton);

cancelButton.addActionListener(**this**);

cancelButton.setActionCommand("Cancel");

}

{

offText = **new** JTextField();

offText.setFont(**new** Font("宋体", Font.***PLAIN***, 20));

offText.setColumns(10);

offText.setBounds(186, 153, 167, 37);

contentPanel.add(offText);

}

**this**.setVisible(**true**);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**if** (e.getActionCommand().equals("ok"))

{

**double** off = Double.*parseDouble*(offText.getText());

**double** real = Double.*parseDouble*(realText.getText());

change.setText(String.*valueOf*(real + off - sMoney));

**new** ChangeOrderEvent().finish\_order(orderNo, deskNo);

}

**else** **if**(e.getActionCommand().equals("Cancel"))

{

dispose();

}

}

}

# 4 数据库设计

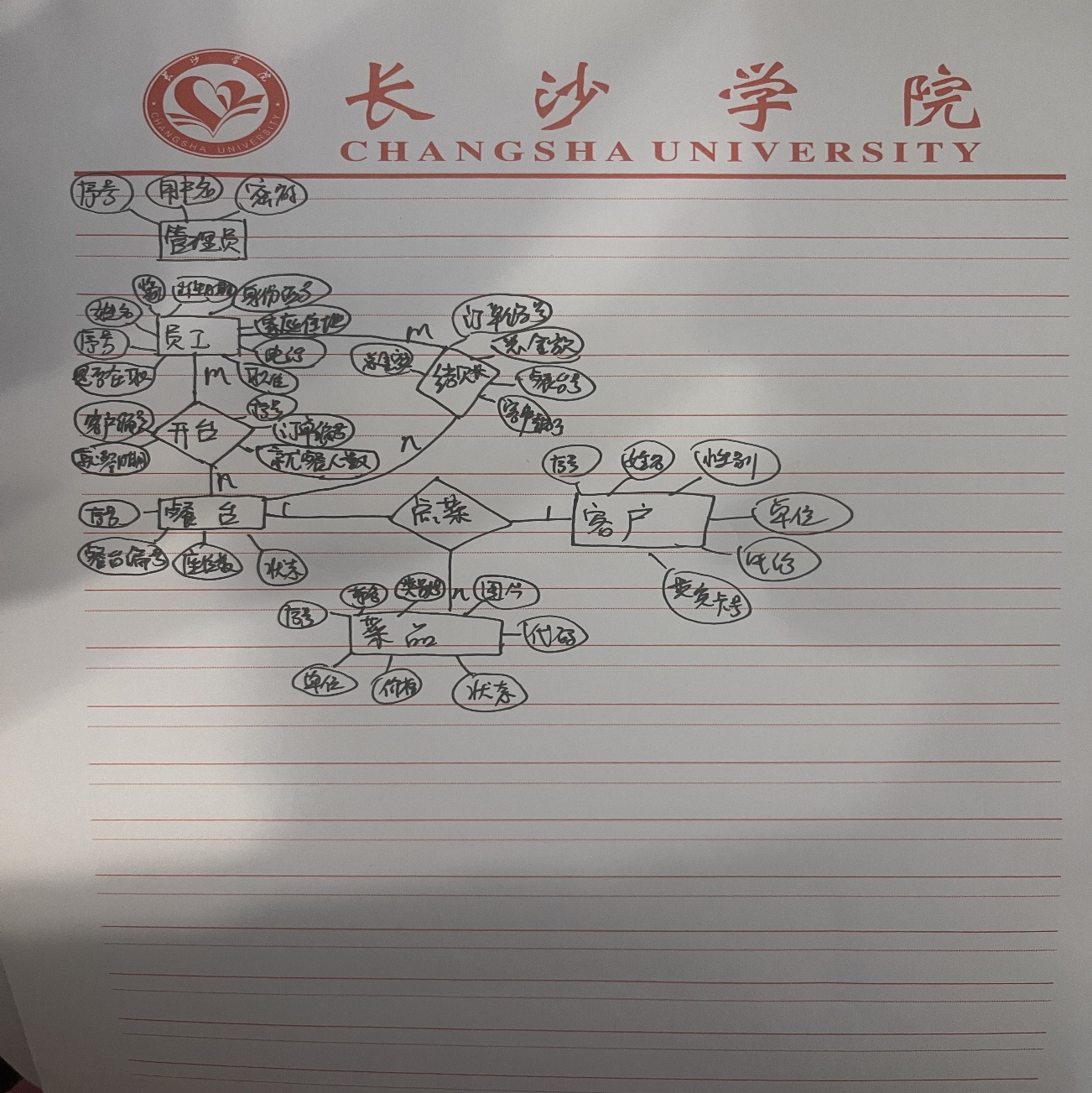
## 4.1 数据库环境说明

数据库使用MySQL 5.6

## 4.2 数据库的命名规则

数据库的命名根据所实现功能的英文翻译定义。

## 4.3 逻辑结构设计

E-R图

关系模型

user(id,username,password)

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

customer(id,name,sex,company,tel,cardID)

category(id,name,describ)

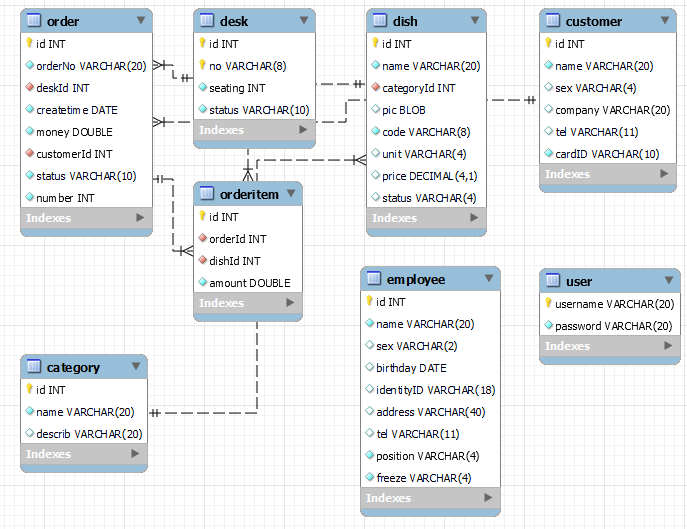
dish(id,name,pic,code,unit,price,status)

desk(id,no,seating,status)

order(id,orderNo,deskId,creatingtime,money,customerId,status,number)

范式分析

不存在决定因素非码的非平凡函数依赖，因此最高属于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 |
| employee | 存储员工信息 |
| customer | 存储客户信息 |
| category | 存储菜品分类信息 |
| dish | 存储菜品信息 |
| desk | 存储餐台信息 |
| order | 存储订单信息 |
| orderitem | 存储点餐明细信息 |

## 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）用户标识和鉴定

（2）存取控制

（3）数据加密

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应用软件，通过应用软件访问数据库，而没有其他途径操作数据库。（3）用户帐号密码的加密方法。对用户帐号的密码进行加密处理，确保在任何地方都不会出现密码的明文。

# 5 测试用例设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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’ | trz,123456 | 登录成功 | 通过 |
| yl-002 | admin，1234567 | 登录失败 | 通过 |
| yl-003 | admin1，123456 | 登录失败 | 通过 |
| yl-004 | admin,空 | 登录失败 | 通过 |
| yl-005 | 空,123456 | 登录失败 | 通过 |
| yl-006 | admin,一个空格 | 登录失败 | 通过 |
| yl-007 | 一个空格，123456 | 登录失败 | 通过 |
| yl-008 | admin，密码长度超过32 | 登录失败 | 通过 |
| yl-009 | 用户名超过36，123456 | 登录失败 | 通过 |

表5.1 登录测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |
| 菜品分类增加 | yl-010 | 输入“名称”和“描述” | 湘菜，辣 | 增加成功 | 通过 |
| yl-011 | 湘菜，空 | 增加成功 | 通过 |
| yl-012 | 空，辣 | 增加失败 | 通过 |
| 菜品分类删除 | yl-013 | 选中某行数据 | 选中一行数据 | 删除成功 | 通过 |
| yl-014 | 选中两行数据 | 删除成功 | 通过 |
| yl-015 | 不选 | 删除失败 | 通过 |
| 菜品类别修改 | yl-016 | 选中“名称”或“描述” | 修改名称为海鲜 | 修改成功 | 通过 |
| yl-017 | 修改描述为鲜美 | 修改成功 | 通过 |

表5.2菜品分类管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |
| 客户增加 | yl-018 | 输入“姓名”，“性别”,“单位”，‘电话’以及选择‘贵宾卡号’ | 唐睿智，男，华为，1，HJ001 | 增加成功 | 通过 |
| yl-019 | 空，男，阿里，2，HJ002 | 增加失败 | 通过 |
| yl-020 | 唐睿智，男，腾讯,134，空 | 增加失败 | 通过 |
| yl-021 | 小梁，男，空，1，HJ002 | 增加成功 | 通过 |
| 客户删除 | yl-022 | 选中某行数据 | 选中一行数据 | 删除成功 | 通过 |
| yl-023 | 选中两行数据 | 删除成功 | 通过 |
| yl-024 | 不选 | 删除失败 | 通过 |
| 客户修改 | yl-025 | 选择某一数据 | 修改用户名为楠楠 | 修改成功 | 通过 |
| yl-026 | 修改性别为女 | 修改成功 | 通过 |
| yl-027 | 修改贵宾卡号为空 | 修改成功 | 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 |
| 菜品增加 | yl-028 | 输入“名称”，选择“类别”，输入“图片”绝对地址，输入“代码”，“单位”，“单价”，“状态” | 辣椒炒肉，湘菜，D:\\t\\order3.jpg，XC001，份，49，在售 | 增加成功 | 通过 |
| yl-029 | 麻辣小龙虾，湘菜，D:\\t\\order4.jpg，XC002，份，空，在售 | 增加成功 | 通过 |
| yl-030 | 佛跳墙，粤菜，空，YC001，份，299，在售 | 增加失败 | 通过 |
| 菜品删除 | yl-031 | 选中某行数据 | 选中一行数据 | 删除成功 | 通过 |
| yl-032 | 选中两行数据 | 删除成功 | 通过 |
| yl-033 | 不选 | 删除失败 | 通过 |
| 菜品修改 | yl-034 | 选中某行数据 | 修改名称为蒜蓉烤生蚝 | 修改成功 | 通过 |
| yl-035 | 修改代码为HX002 | 修改成功 | 通过 |
| yl-036 | 修改单位为打 | 修改成功 | 通过 |
| yl-037 | 修改单价为88 | 修改成功 | 通过 |
| yl-038 | 修改状态为售罄 | 修改成功 | 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 |
| 餐台增加 | yl-039 | 输入“餐台编号”，“座位数”，“状态” | A001，2，空闲 | 增加成功 | 通过 |
| yl-040 | A002，空，空 | 增加失败 | 通过 |
| yl-041 | A003，空，空闲 | 增加失败 | 通过 |
| 餐台删除 | yl-042 | 选中某行数据 | 选中一行数据 | 删除成功 | 通过 |
| yl-043 | 选中两行数据 | 删除成功 | 通过 |
| yl-044 | 不选 | 删除失败 | 通过 |
| 餐台修改 | yl-045 | 选择某行数据 | 修改编号为A004 | 修改成功 | 通过 |
| yl-046 | 修改座位数为4 | 修改成功 | 通过 |
| yl-047 | 修改状态为就餐中 | 修改成功 | 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 |
| 员工增加 | yl-048 | 输入“姓名”，“性别”，“出生日期”，“身份证号”，“家庭住址”，“电话”，“职位”，“是否在职” | 唐睿智，男，2003-03-23,4311，永州，151，董事长，在职 | 增加成功 | 通过 |
| yl-049 | 小李，空，空，空，空，空，空，空 | 增加成功 | 通过 |
| 员工删除 | yl-050 | 选中某行数据 | 选中一行数据 | 删除成功 | 通过 |
| yl-051 | 选中两行数据 | 删除成功 | 通过 |
| yl-052 | 不选 | 删除失败 | 通过 |
| 员工修改 | yl-053 | 选中某行数据 | 修改姓名为小何 | 修改成功 | 通过 |
| yl-054 | 修改是否在职为离职 | 修改成功 | 通过 |

表5.6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 |
| 开台 | yl-056 | 选择“客户名”，输入“就餐人数”（不大于餐台座位数），选择空闲餐台。 | 小梁，2，选择A001（座位数2，状态空闲） | 开台成功 | 通过 |
| yl-057 | 小梁，4，选择B001（座位数4，状态空闲） | 开台失败 | 通过 |
| yl-058 | 小梁，2，选择A002（座位数2，状态就餐中） | 开台失败 | 通过 |

表5.7开台管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |
| 菜品增加 | yl-059 | 餐桌下拉框（只会显示就餐中的餐桌）选择 “餐桌号”，菜品表选中一项“菜品”。 | 选中B003餐桌，选中米饭 | 增加成功 | 通过 |
| yl-060 | 选中B003餐桌，未选中菜品 | 增加失败 | 通过 |
| 菜品删除 | yl-061 | 餐桌下拉框（只会显示就餐中的餐桌）选择 “餐桌号”，已点菜品表选中一行记录。 | 选中B003餐桌，选中麻辣小龙虾 | 删除成功 | 通过 |
| yl-062 | 选中B003餐桌，未选菜品 | 删除失败 | 通过 |
| 结账 | yl-063 | 餐桌下拉框（只会显示就餐中的餐桌）选择 “餐桌号”。 | 选中B003餐桌 | 进入结账界面 | 通过 |
| yl-064 | 未选中餐桌 | 没有进入结账 | 通过 |

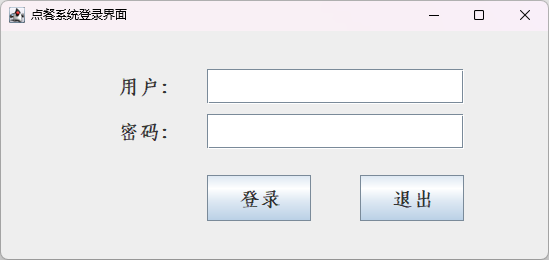
表5.8点菜管理测试用例

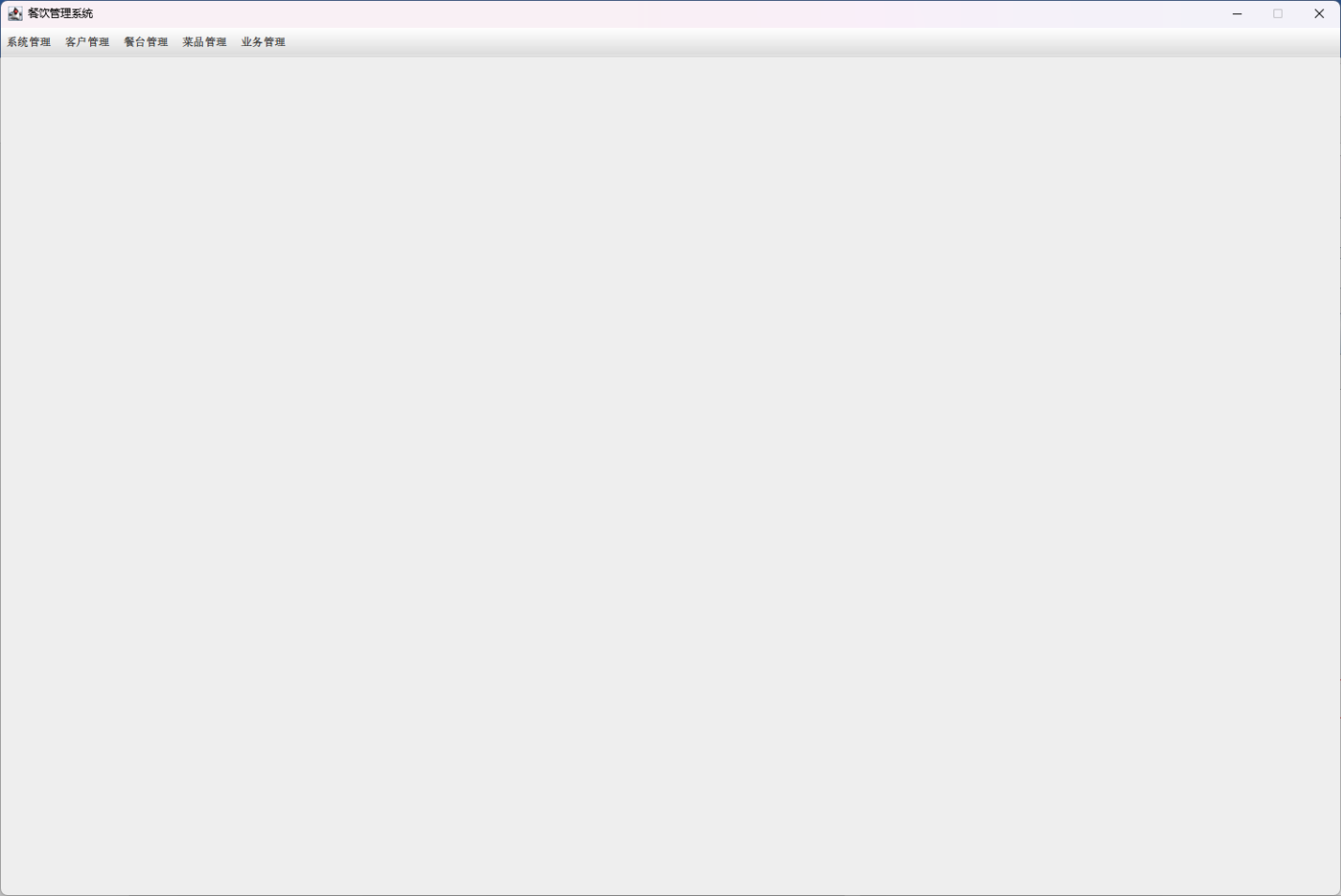
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 确认结账 | yl-065 | 输入“优惠”金额，“实收”金额。 | 应收101。输入0,102 | 结账成功 | 通过 |
| yl-066 | 应收101。输入0,90 | 结账失败 | 通过 |
| yl-067 | 应收101。输入空，空 | 结账失败 | 通过 |

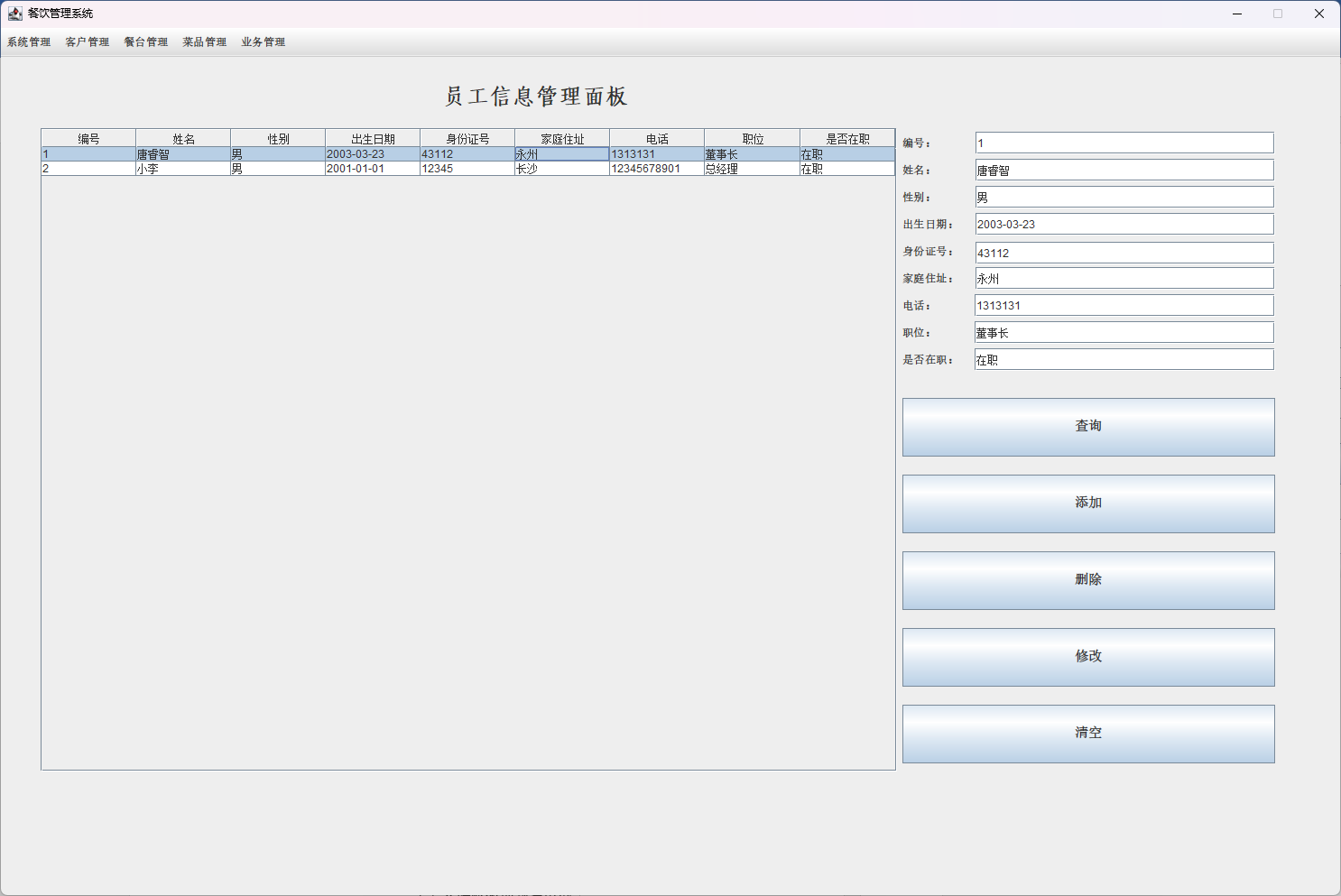
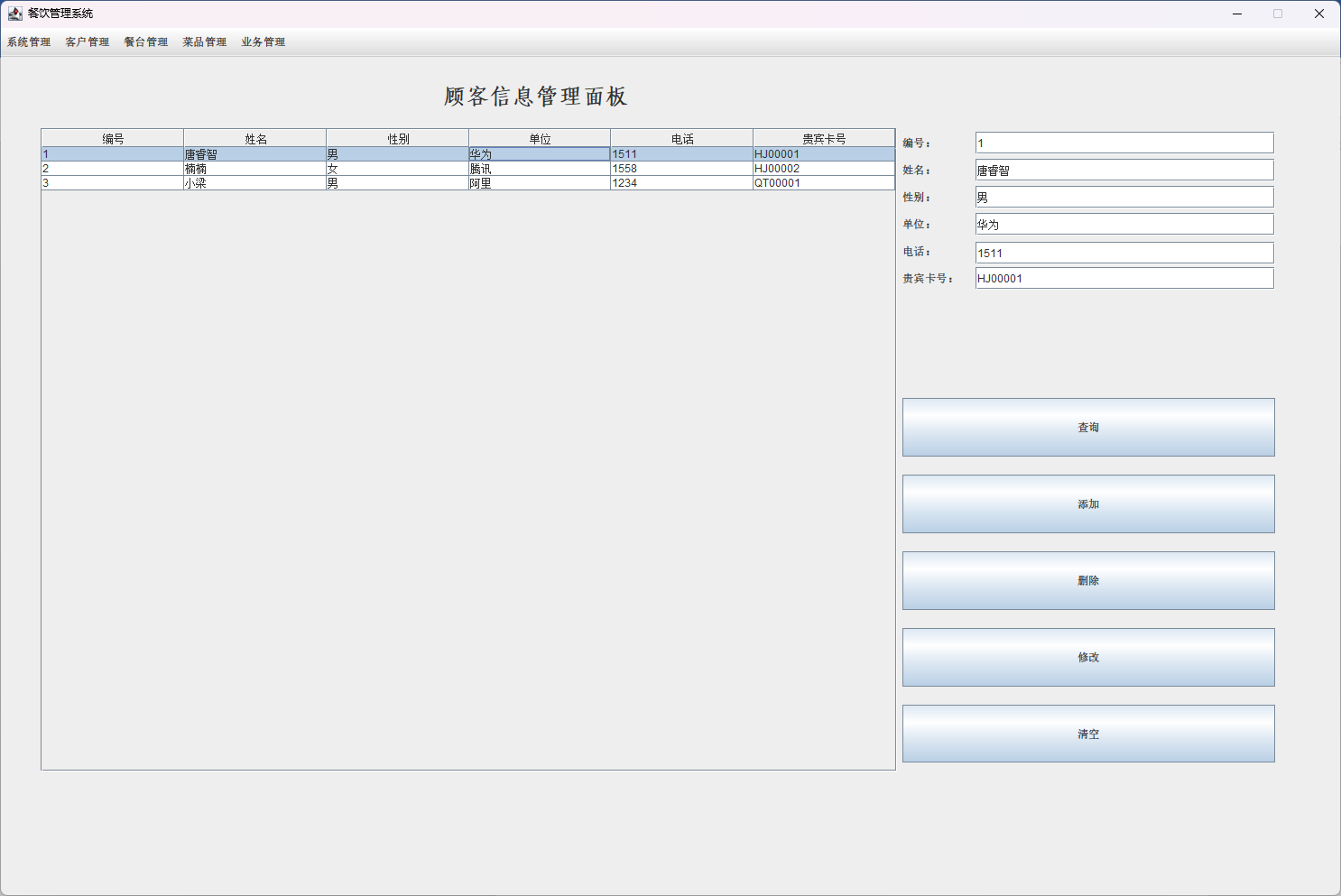
表5.9结账管理测试用例

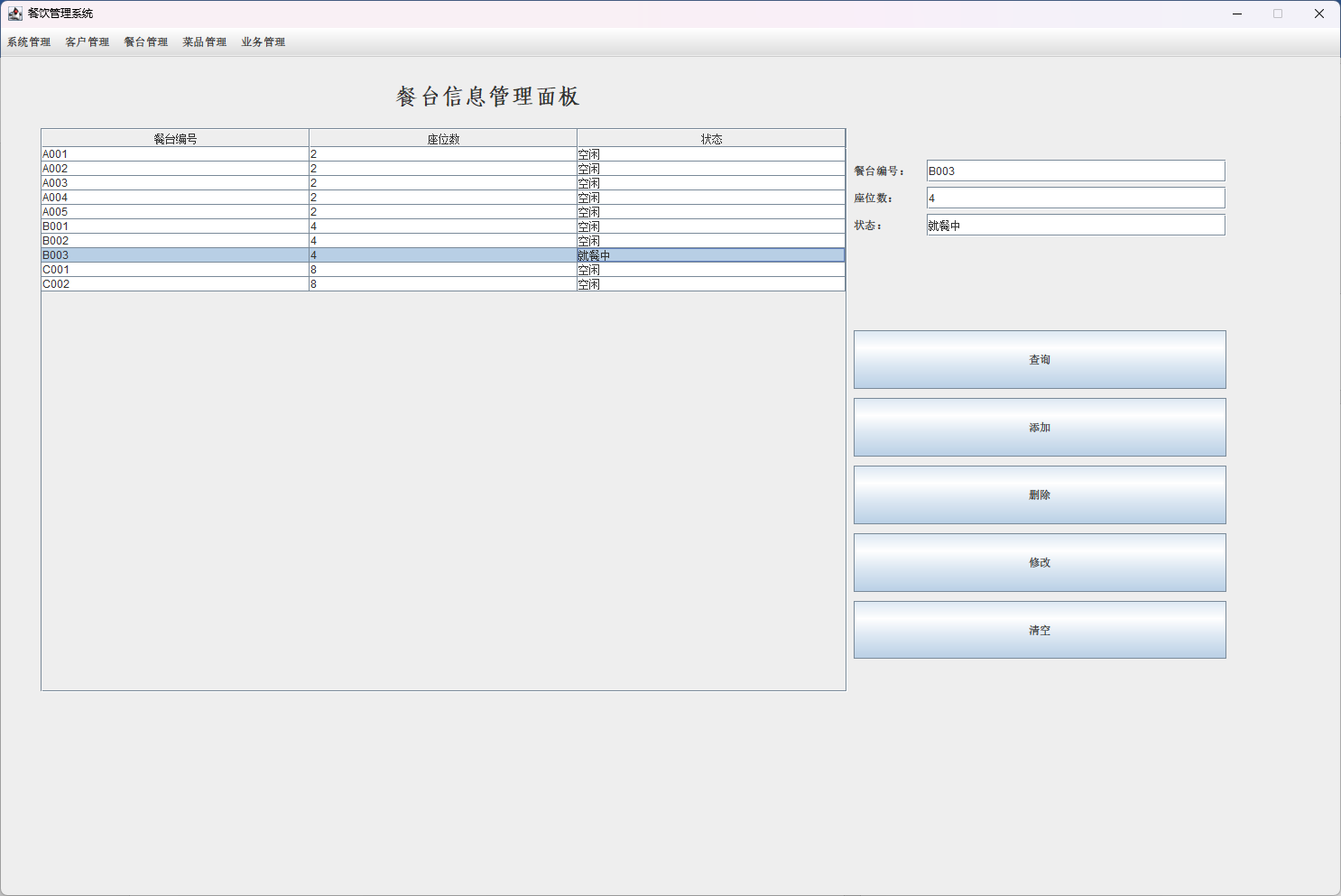
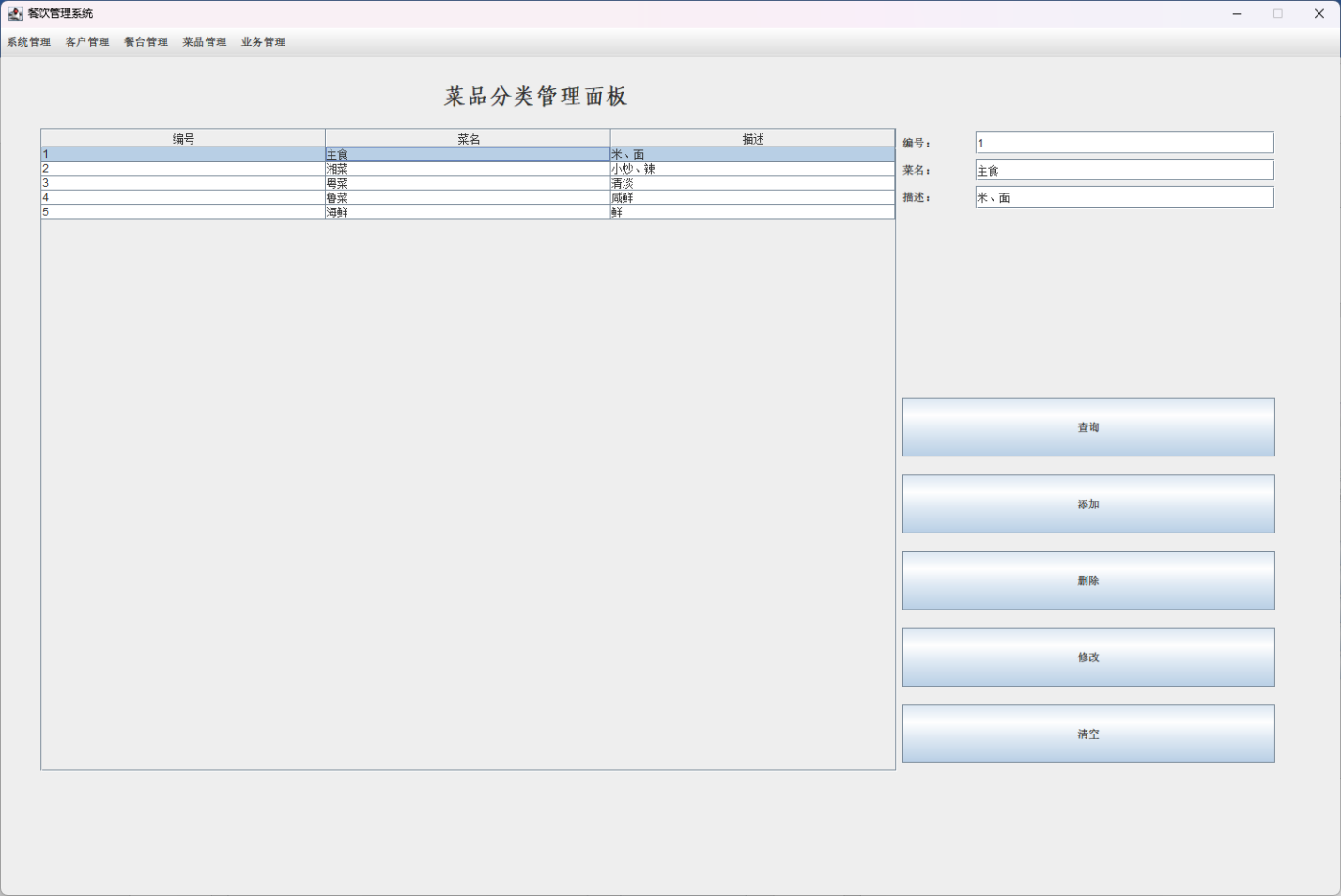
# 附录

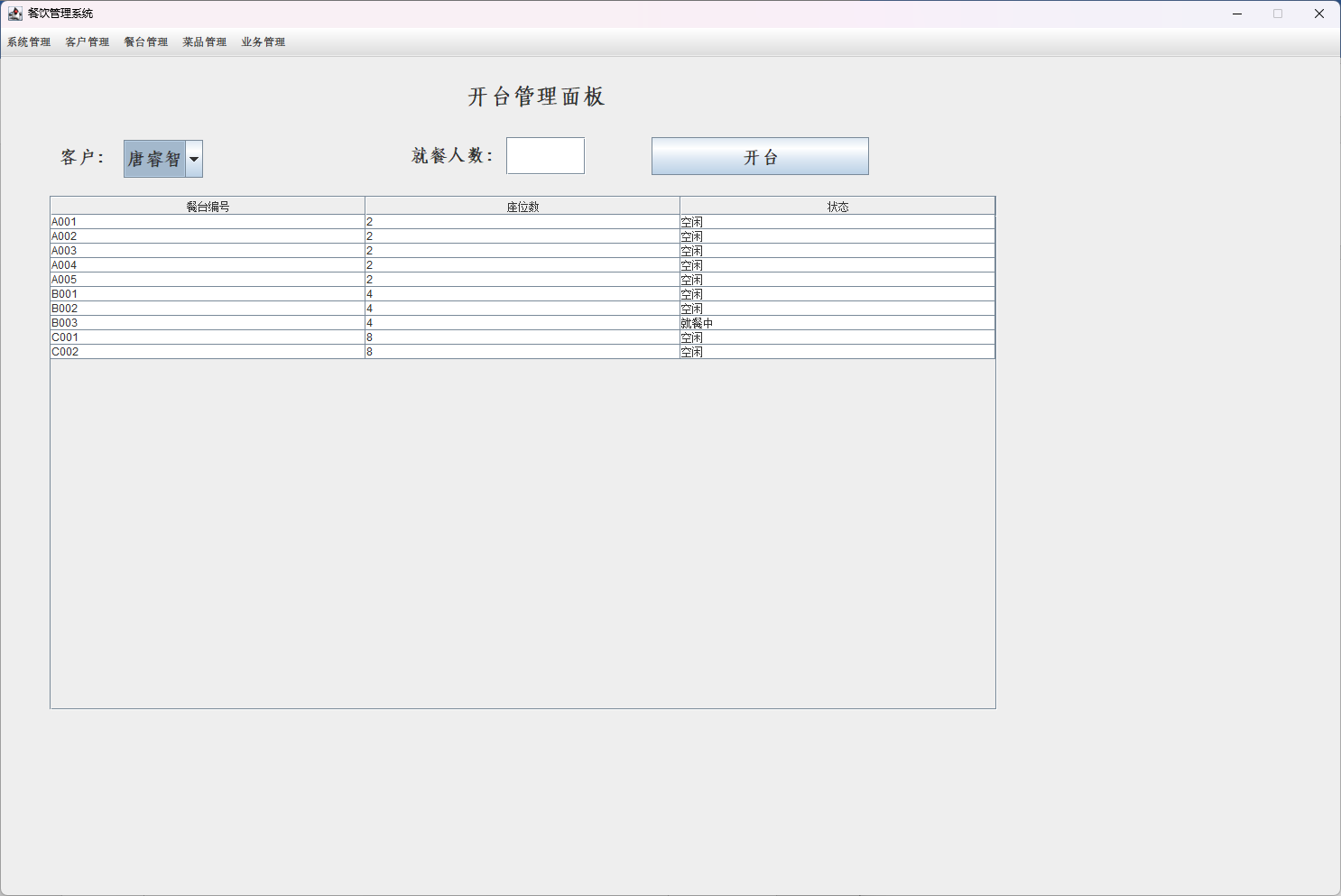
## 附录1 程序运行结果

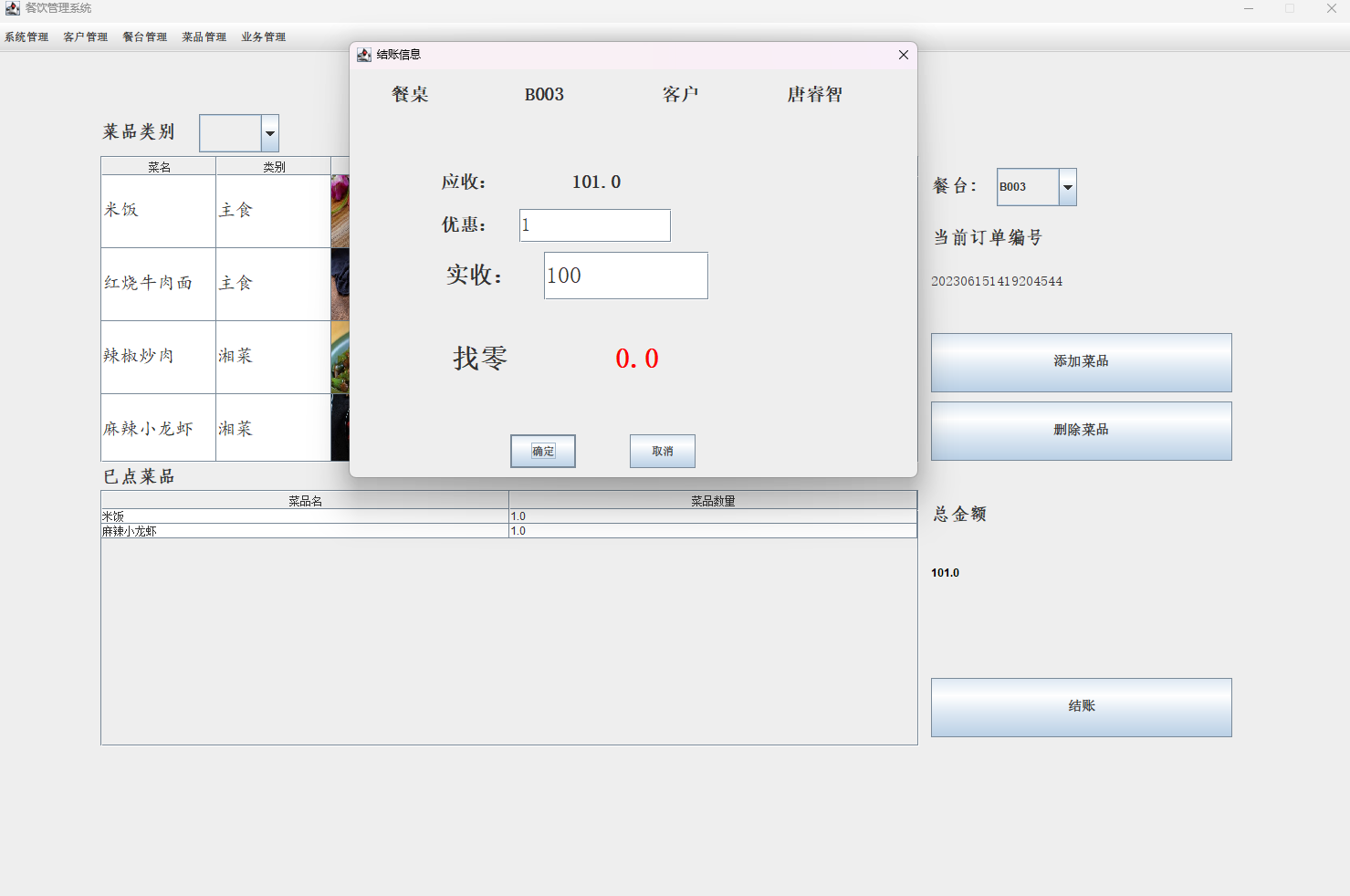












## 附录2 程序源代码

entity包：



DAO包：



util包：



boundry包：

