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** | **周瀚睿** |
| **学号** | **B20210304126** |
| **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 | 周瀚睿 | 学号 | B20210304126 | 班级 | 21软件01 |
| 专业 | 软件工程 | | 指导教师姓名 | 杨刚、潘怡 | |
| 项目名称 | 基于C/S架构的餐饮管理系统 | | | | |
| **实训过程**：（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：  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。  能够针对中餐因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、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1.7，开发工具使用Eclipse，数据库使用mysql5.0.  系统设计：  根据该企业的具体情况，系统主要功能设计有6大部分，分别为用户管理、员工管理、客户管理、餐台管理、菜品管理、菜品分类管理、业务管理、结账管理、支付管理。如图所示。  1.用户管理：对用户实现增删改查。  2.员工管理：对员工实现增删改查。  3.客户管理：对客户实现增删改查。  4.餐台管理：对餐台实现增删改查。  5.菜品管理：对菜品实现增删改查。  6.菜品分类管理：对菜品分类实现增删改查。  7.业务管理： 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。  8.结账管理：显示未支付的订单。  9.支付管理；收银员对某一餐台通过统计显示消费的菜品清单并统计出消费金额，通过手动输入实收金额进行找零的计算，并显示，完成结账的操作，并记录成统计数据。  编码实现：利用Java的GUI设计人机交互界面；利用Java的JDBC所提供的方法对数据库进行操纵。  系统测试：测试系统的完备性、实用性和可靠性。  交付实施：经测试成功后对系统进行维护和运营。 | | | | | |
| **实训成果**：  实现了餐饮管理系统的基本功能，界面美观、人机交互友好 | | | | | |
| **实训总结**：（包括心得体会、存在的问题和改进方向。）  通过这两周的实训，我对Java的GUI编程和JDBC操作数据库有了更深刻的认识，在编写代码时学会了合理利用数据结构和算法去解决实际遇到的问题，学会了如何设计界面，实现界面良好的人机交互功能，在遇到疑难问题时，先独立思考，并查阅网络资料，与同学进行讨论。 | | | | | |

基于C/S架构的餐饮管理系统

设计说明书

作者：B20210304126周瀚睿

计算机科学与工程学院

2023年 06 月 15日

**摘要**

本课程设计的主要内容为设计一个基于C/S架构的餐饮管理系统。

本课程设计基于Java GUI编程，数据库编程，Java JDBC编程。

本系统包含客户管理、员工管理、菜品管理、餐台管理、点菜管理和结账管理六大基本模块。客户管理实现对客户信息的增删改查；员工管理实现对员工信息的增删改查；菜品管理实现对菜品信息的增删改查；餐台管理实现对餐台信息的增删改查；点菜管理实现服务员对客户的人数进行开台，并进行点菜和退菜，实现购物车的功能；结账管理实现对订单的查询和修改功能。

**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](#_Toc28137)

[1. 问题描述（功能要求）： 2](#_Toc26491)

[2. 运行环境要求： 2](#_Toc24787)

[3. 技术要求： 2](#_Toc32080)

[实训工作量： 3](#_Toc7743)

[工作计划： 3](#_Toc26734)

[1 引言 1](#_Toc4699)

[1.1 编写目的 1](#_Toc3290)

[1.2 参考资料 1](#_Toc16251)

[2 需求规约 1](#_Toc24685)

[2.1 功能需求 1](#_Toc13700)

[2.2 界面需求 8](#_Toc10020)

[2.3 数据需求 8](#_Toc30087)

[3 系统设计 12](#_Toc32456)

[3.1 运行环境 13](#_Toc11063)

[3.2 系统静态结构设计 13](#_Toc7431)

[3.3 人机接口设计 14](#_Toc13097)

[3.3.6 用户接口5（客户管理界面）设计 18](#_Toc31550)

[3.4 登录模块设计说明 25](#_Toc23551)

[26](#_Toc17779)

[3.5 员工管理模块设计说明 26](#_Toc483)

[3.6 客户管理模块设计说明 29](#_Toc20059)

[3.7 餐台管理模块设计说明 31](#_Toc24332)

[3.8 菜品管理模块设计说明 33](#_Toc6241)

[3.9 业务管理模块设计说明 35](#_Toc280)

[3.10 支付管理模块设计说明 36](#_Toc17412)

[4 数据库设计 37](#_Toc4733)

[4.1 数据库环境说明 37](#_Toc30216)

[4.2 数据库的命名规则 37](#_Toc24687)

[4.3 逻辑结构设计 38](#_Toc7248)

[4.4 物理结构设计 40](#_Toc2271)

[4.4.1 数据库表 40](#_Toc28244)

[4.5 安全性设计 45](#_Toc22909)

[4.5.1 防止用户直接操作数据库的方法 45](#_Toc11929)

[4.5.2 用户帐号密码的加密方法 45](#_Toc28656)

[5 测试用例设计 45](#_Toc7316)

[附录 49](#_Toc27806)

[附录1 程序运行结果 49](#_Toc17107)

[附录2 程序源代码 57](#_Toc21673)

# 1 引言

## 1.1 编写目的

本文档是“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] 徐转运, 张扬, 王淼. Java高级程序设计[M]. 北京:清华大学出版社, 2014.

# 2 需求规约

## 2.1 功能需求

根据该企业的具体情况，系统主要功能设计有9大部分，分别为用户管理、员工管理、客户管理、餐台管理、菜品管理、菜品分类管理、业务管理、结账管理、支付管理。如图所示。

1.用户管理：对用户实现增删改查。

2.员工管理：对员工实现增删改查。

3.客户管理：对客户实现增删改查。

4.餐台管理：对餐台实现增删改查。

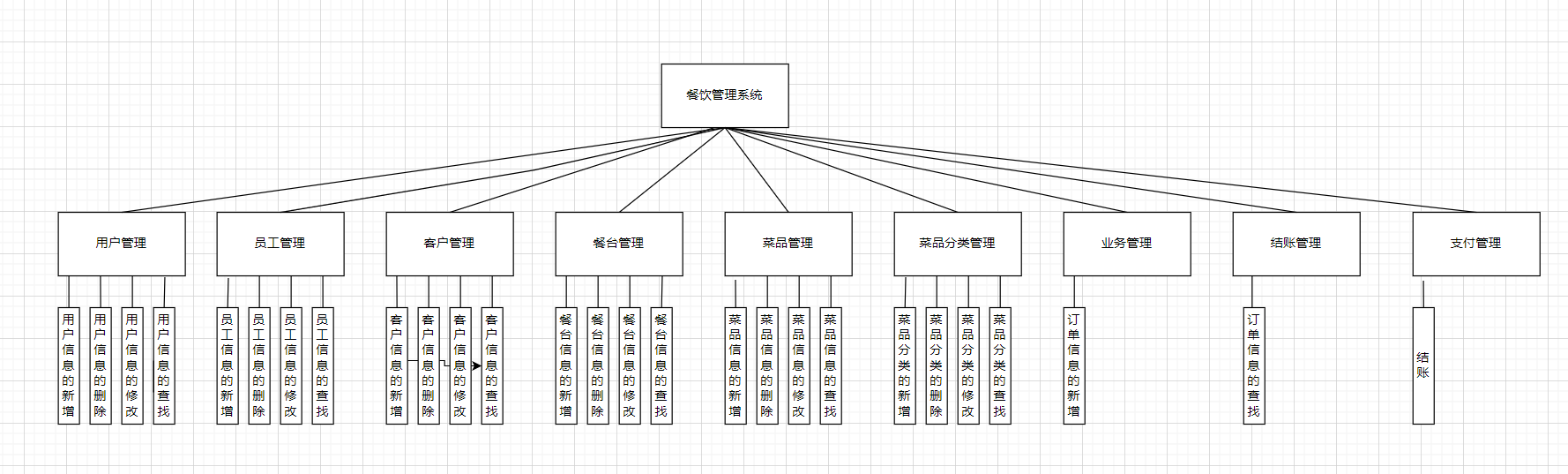
5.菜品管理：对菜品实现增删改查。

6.菜品分类管理：对菜品分类实现增删改查。

7.业务管理： 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。

8.结账管理：显示未支付的订单。

9.支付管理；收银员对某一餐台通过统计显示消费的菜品清单并统计出消费金额，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。



2.1.1 用户登录用例说明

用例名称：用户登录。

用例描述：餐厅管理员登录餐厅管理系统。

前置条件：管理员输入账号和密码。

后置条件：管理员登录成功。

活动步骤：

1. 管理员输入账号和密码
2. 点击登录按钮，管理员登录成功。

异常处理：

1. 用户账号或密码为空，用例终止。
2. 用户输入账号不存在或密码不正确，用例终止。
3. 用户输入账号或密码格式不正确，用例终止。

2.1.2 用户管理用例说明

用例名称：用户管理。

用例描述：餐厅管理员管理账号。

前置条件：用户输入账号、密码。

后置条件：用户添加，删除，查询成功。

活动步骤：

1. 管理员输入账号和密码。
2. 点击添加，删除，查询按钮，管理员添加，删除，查询成功。

异常处理：

1. 管理员输入账号或密码为空，用例终止。
2. 管理员输入账号或密码格式不对，用例终止。

2.2.3 员工管理用例说明

用例名称：员工管理。

用例描述：餐厅管理员对员工信息进行增删改查。

前置条件：餐厅管理员对员工信息进行筛选。

后置条件：餐厅管理员对员工信息进行一系列操作。

活动步骤：

查询操作：

1. 餐厅管理员在文本框内输入员工信息，包括员工id，员工姓名，若未输入员工信息，系统将查询到所有员工。
2. 餐厅管理员查询的信息将在表格中显示。

新增操作：

1. 餐厅管理员在文本框输入员工信息
2. 餐厅管理员点击添加按钮，完成添加
3. 员工信息添加成功后立即更新查询后的表格。

修改操作：

1. 餐厅管理员输入员工信息
2. 餐厅管理员点击修改按钮，完成修改
3. 员工信息修改成功后立即更新查询后的表格。

删除操作：

1. 餐厅管理员在文本框中输入该员工的id或点击该信息行。
2. 餐厅管理员点击删除按钮。
3. 员工信息删除后立即更新查询后表格。

异常处理：

新增操作：若输入员工信息有误，系统将提示具体哪一部分出错。

修改操作：若输入要修改的信息有误，系统将提示修改错误，将不会对表格信息进行更新。

2.2.4 客户管理用例

用例名称：客户管理。

用例描述：餐厅管理员对客户信息进行增删改查。

前置条件：餐厅管理员对客户信息进行筛选。

后置条件：餐厅管理员对客户信息进行一系列操作。

活动步骤：

查询操作：

1. 餐厅管理员在文本框内输入客户信息，包括客户id，客户姓名，若未输入客户信息，系统将查询到所有客户。
2. 餐厅管理员查询的信息将在表格中显示

新增操作：

1. 餐厅管理员在文本框输入客户信息
2. 餐厅管理员点击添加按钮，完成添加
3. 客户信息添加成功后立即更新查询后的表格

修改操作：

1. 餐厅管理员输入客户信息
2. 餐厅管理员点击修改按钮，完成修改
3. 客户信息修改成功后立即更新查询后的表格。

删除操作：

1. 餐厅管理员在文本框中输入该客户的id或点击该信息行。
2. 餐厅管理员点击删除按钮。
3. 客户信息删除后立即更新查询后表格。

异常处理：

新增操作：若输入客户信息有误，系统将提示具体哪一部分出错。

修改操作：若输入要修改的信息有误，系统将提示修改错误，将不会对表格信息进行更新。

2.2.5 菜品管理用例

用例名称：菜品管理。

用例描述：餐厅管理员对菜品信息进行增删改查。

前置条件：餐厅管理员对菜品信息进行筛选。

后置条件：餐厅管理员对菜品信息进行一系列操作。

活动步骤：

查询操作：

1. 餐厅管理员在文本框内输入菜品信息，包括菜品id，菜名，若未输入菜品信息，系统将查询到所有菜品。
2. 餐厅管理员查询的信息将在表格中显示

新增操作：

1. 餐厅管理员在文本框输入菜品信息

2. 餐厅管理员点击添加按钮，完成添加

3. 菜品信息添加成功后立即更新查询后的表格

修改操作：

1. 餐厅管理员输入菜品信息
2. 餐厅管理员点击修改按钮，完成修改

3. 菜品信息修改成功后立即更新查询后的表格。

删除操作：

1. 餐厅管理员在文本框中输入该菜品的id或点击该信息行。
2. 餐厅管理员点击删除按钮。
3. 菜品信息删除后立即更新查询后表格。

异常处理：

新增操作：若输入菜品信息有误，系统将提示具体哪一部分出错。

修改操作：若输入要修改的信息有误，系统将提示修改错误，将不会对表格信息进行更新。

2.2.6 菜品分类管理用例

用例名称：菜品分类管理。

用例描述：餐厅管理员对菜品分类信息进行增删改查。

前置条件：餐厅管理员对菜品分类信息进行筛选。

后置条件：餐厅管理员对菜品分类信息进行一系列操作。

活动步骤：

查询操作：

1. 餐厅管理员在文本框内输入菜品分类信息，包括菜品id，分类名，若未输入菜品信息，系统将查询到所有菜品。
2. 餐厅管理员查询的信息将在表格中显示

新增操作：

1. 餐厅管理员在文本框输入菜品分类信息

2. 餐厅管理员点击添加按钮，完成添加

3. 菜品分类信息添加成功后立即更新查询后的表格。

修改操作：

1. 餐厅管理员输入菜品分类信息
2. 餐厅管理员点击修改按钮，完成修改

3. 菜品分类信息修改成功后立即更新查询后的表格。

删除操作：

1. 餐厅管理员在文本框中输入该菜品分类的id或点击该信息行。
2. 餐厅管理员点击删除按钮。
3. 菜品分类信息删除后立即更新查询后表格。

异常处理：

新增操作：若输入菜品信息有误，系统将提示具体哪一部分出错。

修改操作：若输入要修改的信息有误，系统将提示修改错误，将不会对表格信息进行更新。

2.2.7 餐台管理用例

用例名称：餐台管理。

用例描述：餐厅管理员对餐台信息进行增删改查。

前置条件：餐厅管理员对餐台信息进行筛选。

后置条件：餐厅管理员对餐台信息进行一系列操作。

活动步骤：

查询操作：

1. 餐厅管理员在文本框内输入餐台号，若未输入餐台号，系统将查询到所有餐台。
2. 餐厅管理员查询的信息将在表格中显示

新增操作：

1. 餐厅管理员在文本框输入餐台信息

2. 餐厅管理员点击添加按钮，完成添加

3. 餐台信息添加成功后立即更新查询后的表格。

修改操作：

1. 餐厅管理员输入餐台信息
2. 餐厅管理员点击修改按钮，完成修改

3. 餐台信息修改成功后立即更新查询后的表格。

删除操作：

1. 餐厅管理员在文本框中输入该餐台的id或点击该信息行。
2. 餐厅管理员点击删除按钮。
3. 餐台信息删除后立即更新查询后表格。

异常处理：

修改操作：若输入要修改的信息有误，系统将提示修改错误，将不会对表格信息进行更新。

2.2.8 业务管理用例

用例名称：业务管理。

用例描述：餐厅管理员对客户进行开台，点菜，下单操作。

前置条件；餐厅管理员输入客户的客户名和人数，对客户进行开台操作。

后置条件：餐厅管理员为客户完成点单，下单操作。

活动步骤：

1. 餐厅管理员输入客户名以及人数，对客户进行开台操作。
2. 系统将为该用户分配空闲的餐台。
3. 餐厅管理员点击选择菜品，进入新界面，新界面包含菜品信息，提供给客户去选菜。
4. 选菜完成后点击加入后再点击下单，完成下单，生成订单。

异常处理：

1. 餐厅管理员输入的人数大于现有空闲餐台的所能容纳的最大人数时，系统将提示找不到合适的餐台。

2.2.9 结账管理用例

用例名称：结账管理。

用例描述：餐厅管理员对订单进行查询的操作。

前置条件：餐厅管理员对订单进行查询。

后置条件：餐厅管理员查看未支付订单。

活动步骤：

1. 餐厅管理员点击刷新，对订单进行查询。
2. 在文本框中显示未支付订单。

2.2.10 支付管理用例

用例名称：支付管理。

用例描述：餐厅管理员对订单进行查询，删除，结账的操作。

前置条件：餐厅管理员输入餐桌号和客户名，对订单进行查询。

后置条件：餐厅管理员对所选订单进行结账处理。

活动步骤：

1. 餐厅管理员输入餐桌号和客户名，对订单进行查询。
2. 在文本框中显示订单的应付金额与折后价格以及实付金额。
3. 餐厅管理员在实付金额处填写客户支付的金额，点击支付，系统将对该客户进行找零操作，同时更新查询后的表格。

异常处理：餐厅管理员输入金额小于折后价格，将提醒支付失败。

## 2.2 界面需求

1. 界面友好，操作简单。

2. 界面美观，用户体验感良好。

## 2.3 数据需求

表2.1 用户实体user

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 长度 | 是否为空 | 唯一标识 | 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 员工实体employees

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| id | int | 10 | no | yes | 序号 |
| name | varchar | 20 | no | no | 用户名 |
| sex | varchar | 2 | yes | no | 性别 |
| birthday | datatime | 8 | yes | no | 出生日期 |
| 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 客户实体customer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 长度 | 是否为空 | 唯一标识 | 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 菜品分类实体category

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 长度 | 是否为空 | 唯一标识 | 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 菜品信息实体dish

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| id | int | 10 | no | yes | 序号 |
| name | varchar | 20 | no | no | 菜品名 |
| categoryId | 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 餐台实体desk

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| id | int | 10 | no | yes | 序号 |
| no | varchar | 8 | no | yes | 餐台编号 |
| seating | int | 4 | no | no | 座位数 |
| status | varchar | 10 | no | no | 状态 |

表2.7 订单实体order

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 长度 | 是否为空 | 唯一标识 | 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 | 就餐日期时间 |
| 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 订单明细实体orderitem

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 长度 | 是否为空 | 唯一标识 | 说明 |
| id | int | 10 | no | yes | 序号 |
| orderId | int | 10 | no | no | 订单编号 |
| dishId | int | 10 | no | no | 菜品序号 |
| amount | double | 4 | no | no | 菜品数量 |

# 3 系统设计

餐饮管理系统的总体设计思路是建立一个集用户管理、员工管理、客户管理、餐台管理、菜品管理、菜品分类管理、业务管理、结账管理、支付管理功能于一体的系统，以提高餐厅的运营效率和管理水平。

下面是每个功能实现的思路：

用户管理：系统应能够管理餐厅的用户信息，包括用户名和密码。系统可以通过系统添加、修改和删除用户信息。在实现上可以设计一个用户管理模块，提供用户信息录入和查询功能，并将用户信息存储在数据库中。

员工管理：系统应能够管理餐厅的员工信息，包括用户名，性别，出生日期，身份证号，家庭住址，电话，职位，是否在职。管理员可以通过系统添加、修改和删除员工信息。在实现上，可以设计一个员工管理模块，提供员工信息录入和查询功能，并将员工信息存储在数据库中。

客户管理：系统应能够管理餐厅的客户信息，包括用户名，性别，单位，电话，贵宾卡好。管理员可以通过系统添加、修改和删除客户信息。在实现上，可以设计一个客户管理模块，提供客户信息录入和查询功能，并将客户信息存储在数据库中。

餐台管理：系统应能够管理餐厅的餐台信息，包括餐台编号，座位数，状态(空闲、使用中)。管理员可以通过系统添加、修改和删除餐台信息。在实现上，可以设计一个餐台管理模块，提供餐台信息录入和查询功能，并将餐台信息存储在数据库中。

菜品管理：系统应能够管理餐厅的菜品信息，包括菜品名，类别编号，图片，菜品代码，单位，价格，状态(可用、不可用)。管理员可以通过系统添加、修改和删除菜品信息。在实现上，可以设计一个菜品管理模块，提供菜品信息录入和查询功能，并将菜品信息存储在数据库中。

菜品分类管理：系统应能够管理餐厅的菜品分类信息，包括名称，描述。管理员可以通过系统添加、修改和删除菜品分类信息。在实现上，可以设计一个菜品分类管理模块，提供菜品分类信息录入和查询功能，并将菜品分类信息存储在数据库中。

业务管理：系统应具备接收、处理和跟踪订单的能力。用户可以通过系统下单，同时也可以通过系统查看订单状态和历史记录。在实现上，可以采用前后端分离的架构，前端提供用户订单输入界面和查询界面，后端负责处理订单逻辑和与数据库的交互。

结账管理：系统应具备接收、处理和跟踪订单的能力。用户可以通过结账管理找到未结账的订单。在实现上，可以设计一个结账管理模块，提供订单查询功能，通过数据库查询未支付的订单。

支付管理：系统应具备接收、处理和跟踪订单的能力。用户可以通过系统查询应收金额，通过输入实收金额后，计算找零。在实现上，可以通过数据库查询应收金额，并比较与实收金额的大小，通过计算给出找零。

总体而言，餐饮管理系统的设计思路是通过集成各个功能模块，实现订单、餐台、员工、客户、菜品、找零的全面管理，从而提高餐厅的运营效率和管理水平。

## 3.1 运行环境

操作系统：Windows 11

数据库：MySQL 8.0.33

## 3.2 系统静态结构设计

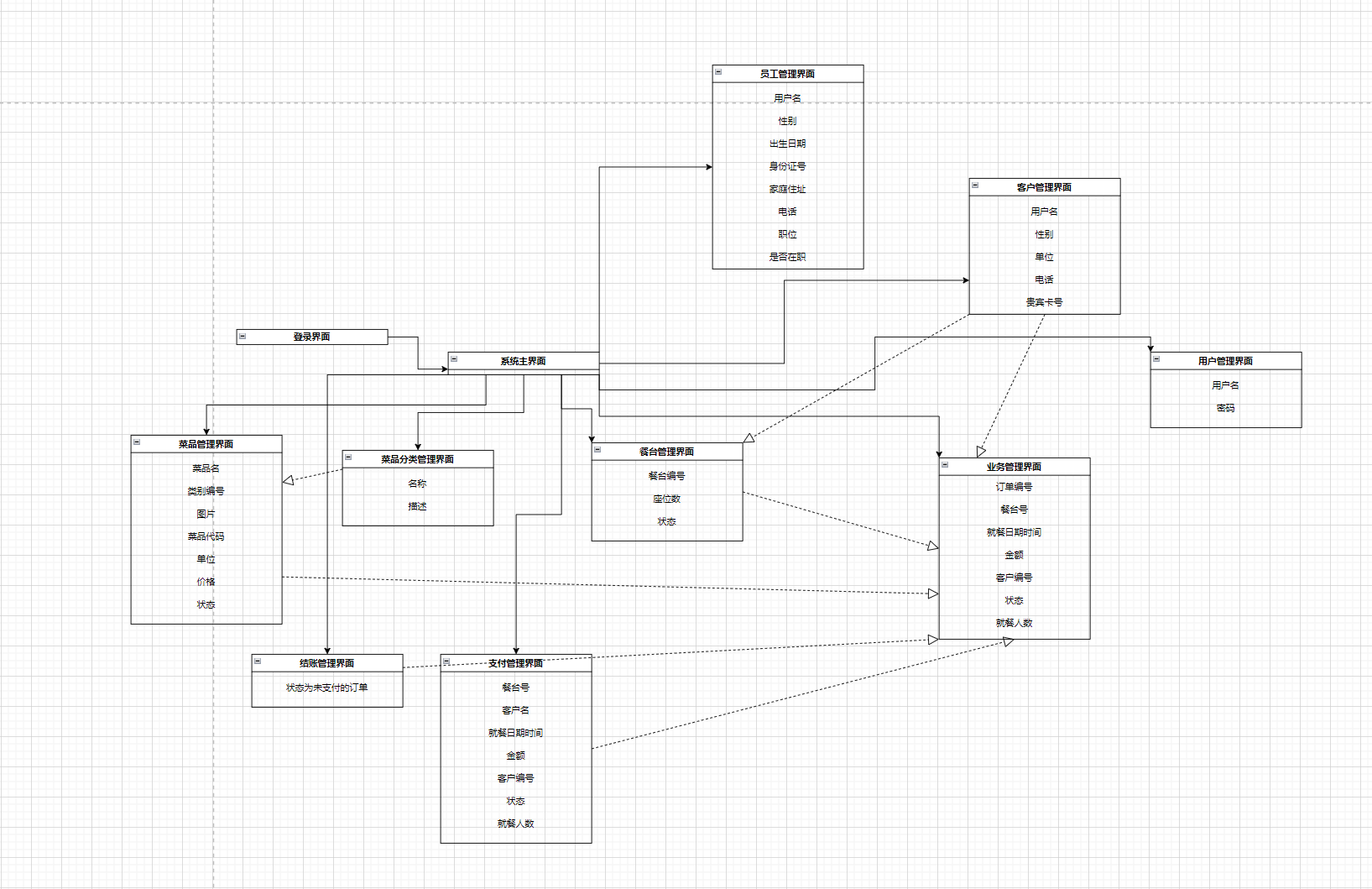


图3.1 餐饮管理系统总体类图

系统静态结构说明

餐厅管理员通过主界面可以跳转至不同的界面，在员工管理界面，管理员可以实现对员工信息的增删改查，在客户管理界面，管理员可以实现对客户信息的增删改查，在菜品管理界面，管理员可以实现对菜品信息的增删改查，在菜品分类管理界面，管理员可以实现对菜品分类信息的增删改查，在餐台界面，管理员可以实现对餐台信息的增删改查，在业务管理界面，管理员可以为顾客提供点菜服务，点菜完成后生成新的订单，在支付管理界面，管理员可以对订单进行查询和结账操作。

## 3.3 人机接口设计

3.3.1 设计应用类

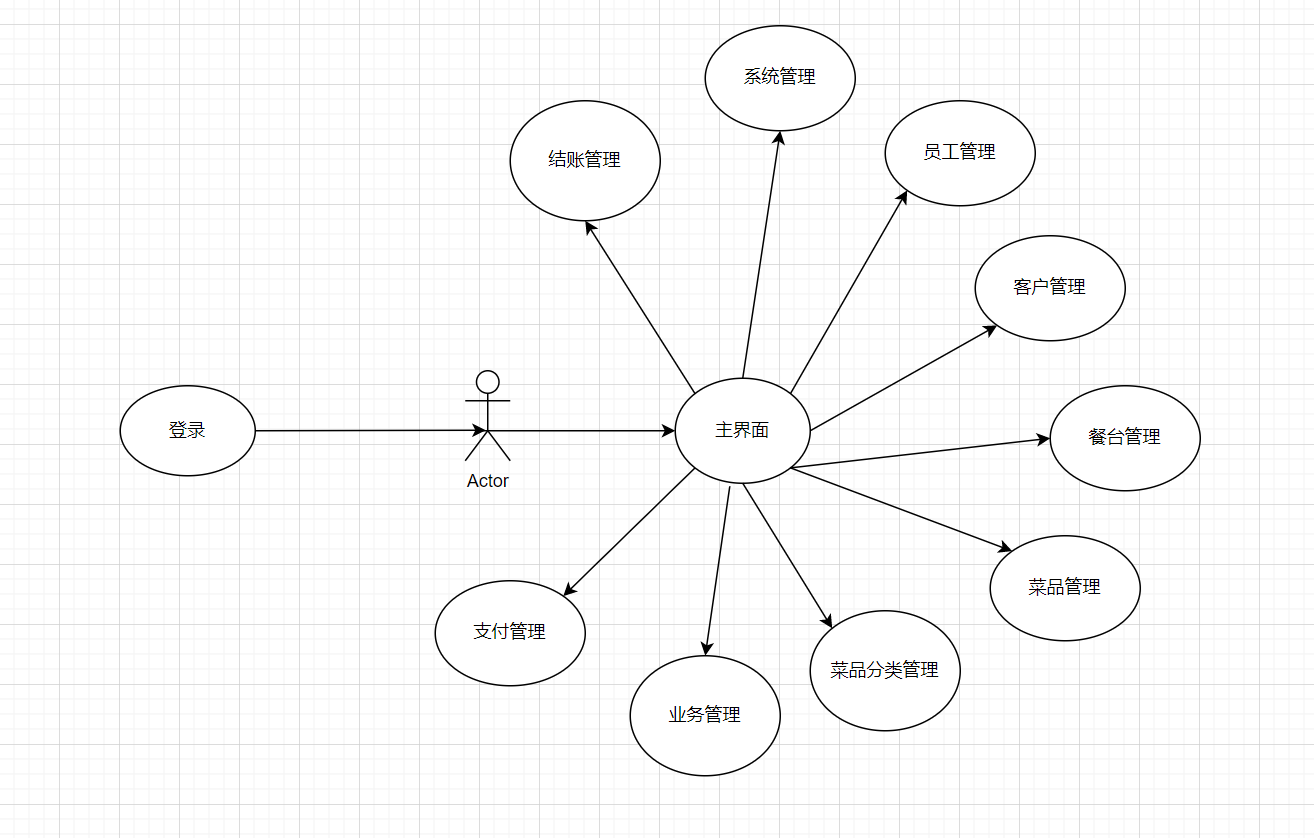


图3.3 餐饮管理系统设计类图

3.3.2 用户接口1（登录界面）设计

**界面主要控件：**

JTextField，输入框，接收账号和密码。

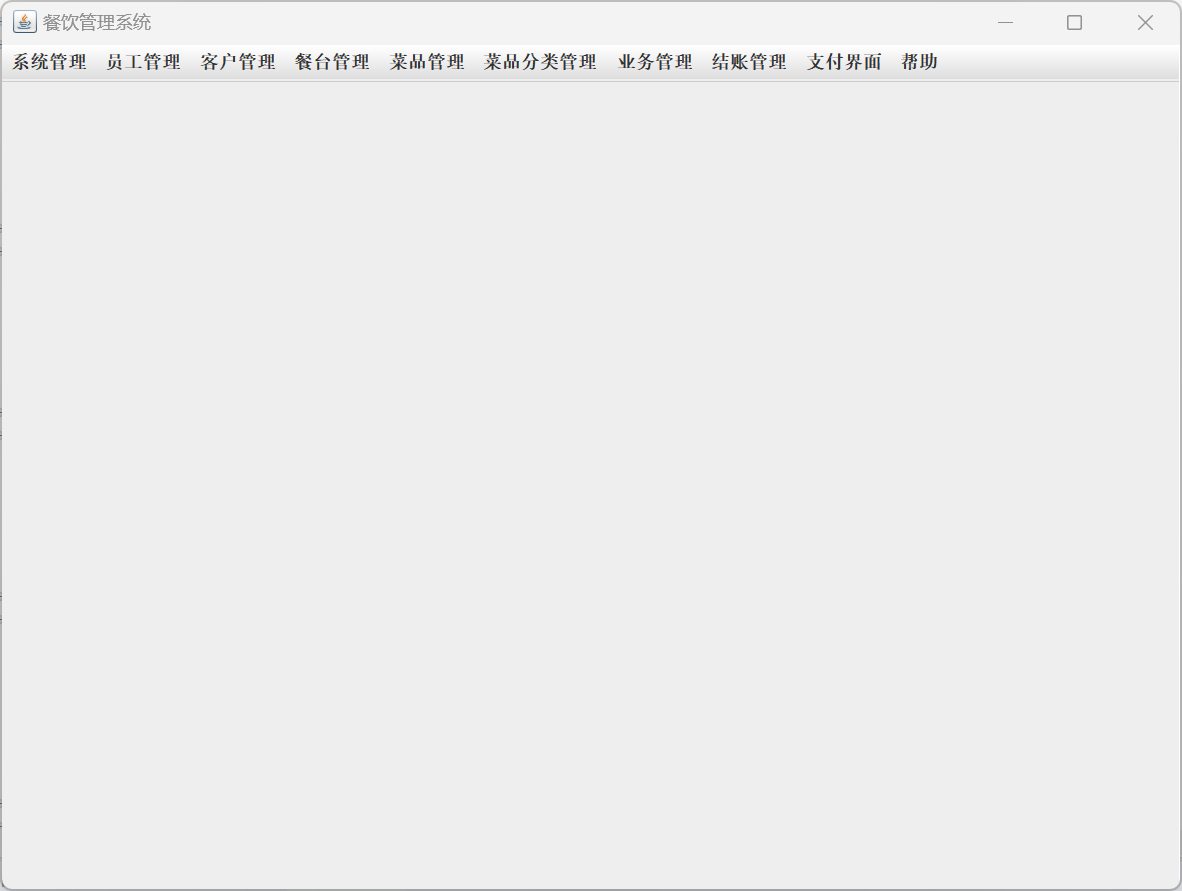
JButton，按钮，触发登录注册事件。

**实现功能：**

1. 餐厅管理员输入账号密码，点击登录按钮跳转餐厅管理界面；账号设置为10位数字，密码长度为8到16位的数字或字母，字母区分大小写。



3.3.3 用户接口2（餐饮管理系统主界面）设计



**界面主要控件：**

JButton，按钮，触发点击事件。

**实现功能：**

1. 点击员工管理，跳转至员工管理界面。
2. 点击客户管理，跳转至客户管理界面。
3. 点击餐台管理，跳转至餐台管理界面。
4. 点击菜品管理，跳转至菜品管理界面。
5. 点击菜品分类管理，跳转至菜品分类管理界面。
6. 点击业务管理功能，跳转至业务管理界面
7. 点击结账管理，跳转至结账管理界面。
8. 点击支付管理，跳转至支付管理界面。
9. 点击系统管理，跳转至系统管理界面。

3.3.4 用户接口3（系统管理界面）设计

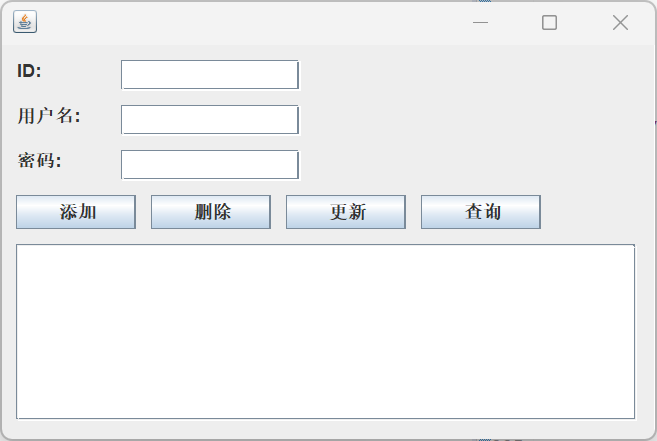
**界面主要控件：**

JButton，按钮，触发点击事件。

JTextField，输入框，接收用户id，用户名和密码。

**实现功能：**

1. 点击添加按钮，将进行用户的增添。
2. 通过输入信息，点击删除按钮，将用户信息删除。
3. 通过输入ID，点击查询按钮，将查询用户信息。



3.3.5 用户接口4（员工管理界面）设计

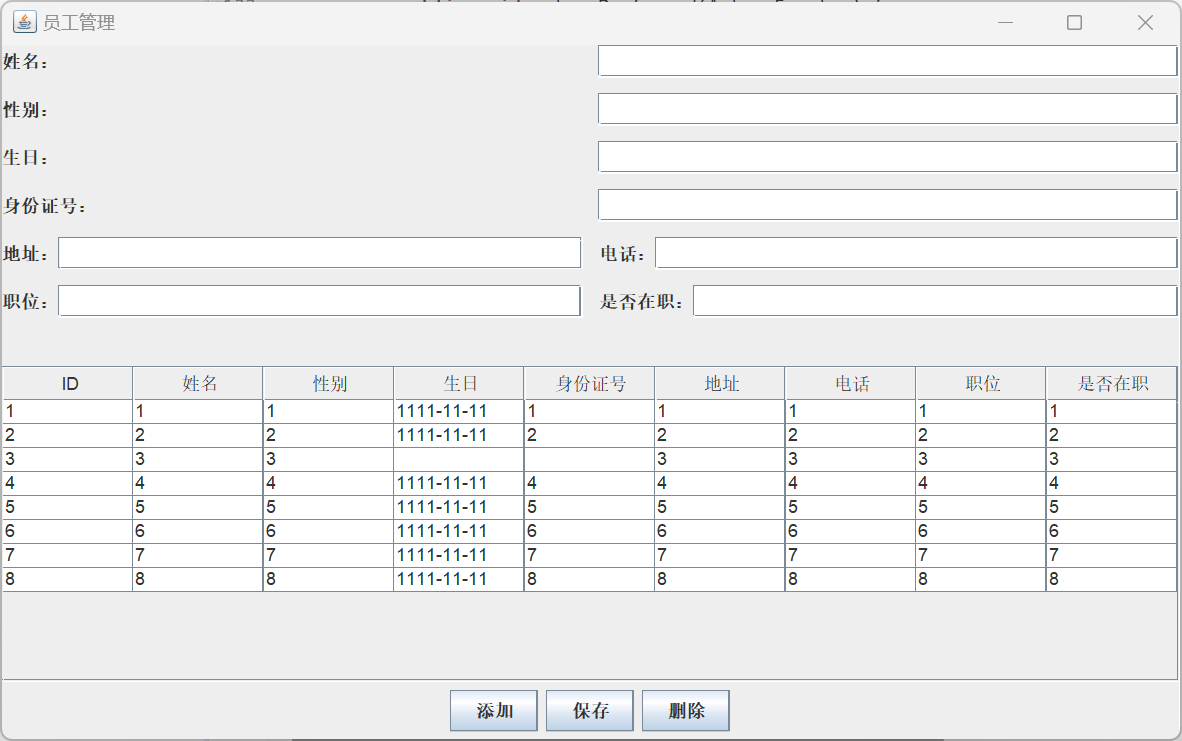
**界面主要控件：**

JButton，按钮，触发点击事件。

JTextField，输入框，接收员工姓名，性别，生日，身份证号，地址，电话，职位，是否在职。

**实现功能：**

1. 点击添加按钮，将对进行员工的增添。
2. 点击表格中的某个员工后再点击删除员工，将对该员工进行删除。
3. 通过输入信息，点击保存按钮，将对员工信息进行修改。



## 3.3.6 用户接口5（客户管理界面）设计

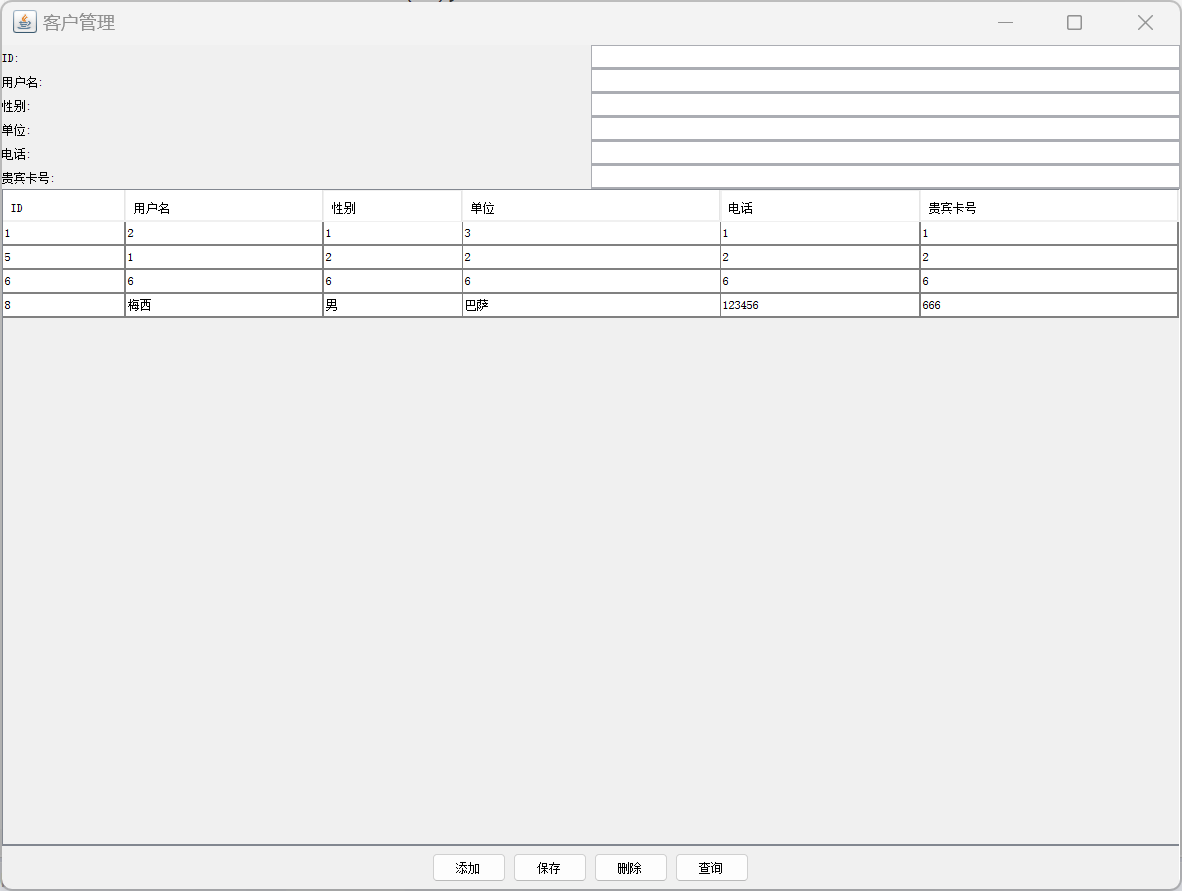
**界面主要控件：**

JButton，按钮，触发点击事件。

JTextField，输入框，接收客户id，用户名，性别，单位，电话，贵宾卡号。

**实现功能：**

1. 点击查询按钮，将对客户进行查询。
2. 点击添加按钮，将对客户进行增添。
3. 点击表格中的某个客户后再点击删除客户，将对该客户进行删除。
4. 点击表格，将对客户信息进行修改。



3.3.7 用户接口6（餐台管理界面）设计

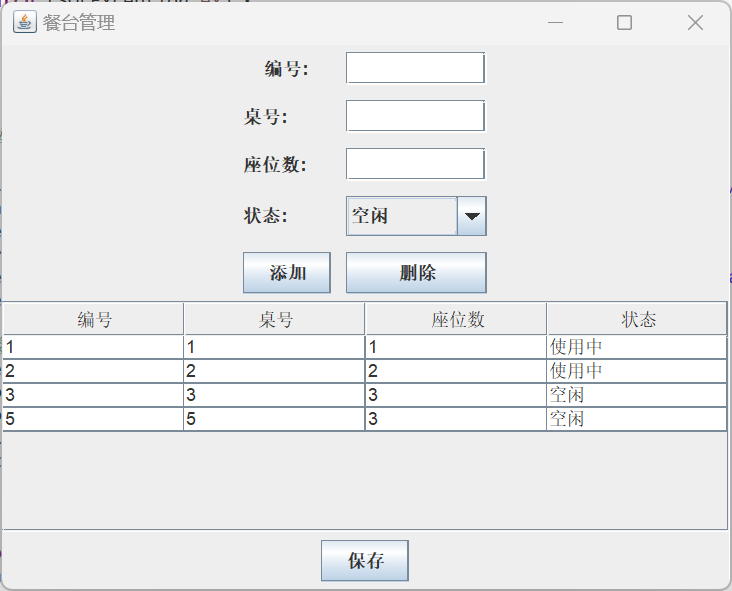
**界面主要控件：**

JButton，按钮，触发点击事件。

JTextField，输入框，接收餐台号，编号，座位数，状态。

**实现功能：**

1. 点击添加按钮，将对进行餐台的增添。
2. 点击表格中的某个餐台后再点击删除餐台，将对该餐台进行删除。
3. 点击表格，将对餐台信息进行修改。



3.3.8 用户接口7（菜品管理界面）设计

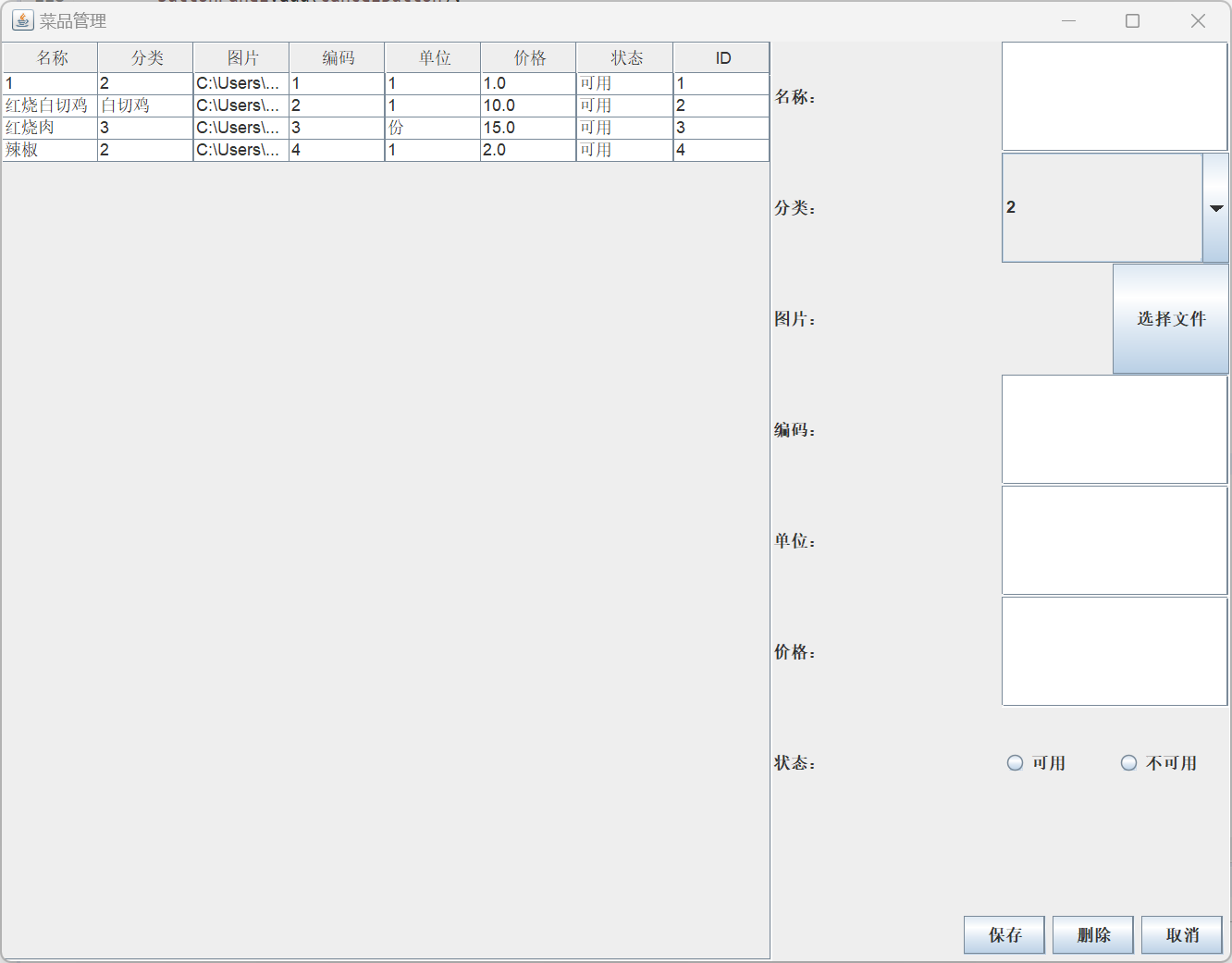
**界面主要控件：**

JButton，按钮，触发点击事件。

JTextField，输入框，接收菜品名称，分类，图片，编码，单位，价格，状态。

**实现功能：**

1. 点击保存按钮，将对进行菜品的增添。
2. 点击表格中的某个菜品后再点击删除菜品，将对该菜品进行删除。
3. 点击表格，将对菜品信息进行修改。

****

3.3.9 用户接口8（菜品分类管理界面）设计

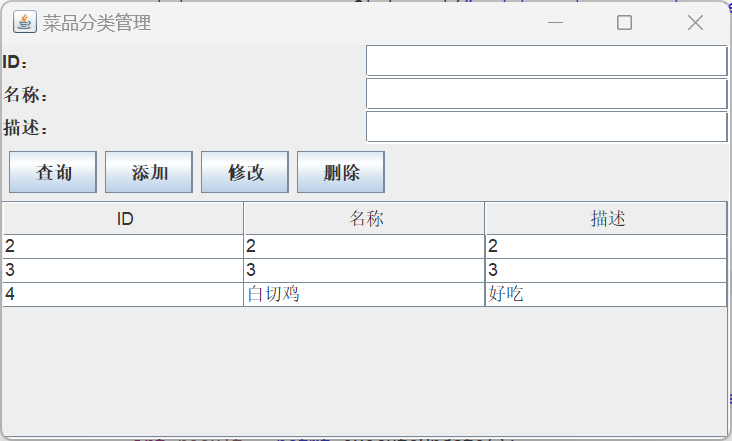
**界面主要控件：**

JButton，按钮，触发点击事件。

JTextField，输入框，接收菜品名称，描述。

**实现功能：**

1. 点击添加按钮，将对进行菜品的增添。
2. 点击表格中的某个菜品分类后再点击删除菜品，将对该菜品分类进行删除。
3. 点击表格，将对菜品分类信息进行修改。

****

3.3.10 用户接口9（点菜管理界面）设计

**界面主要控件：**

界面1：

JButton，按钮，触发点击事件。

JTextField，输入框，接收客户名，餐桌号和就餐人数。

界面2：

JButton，按钮，触发点击事件。

JTextField，输入框，接收菜名。

JComboBox，对菜品信息进行筛选。

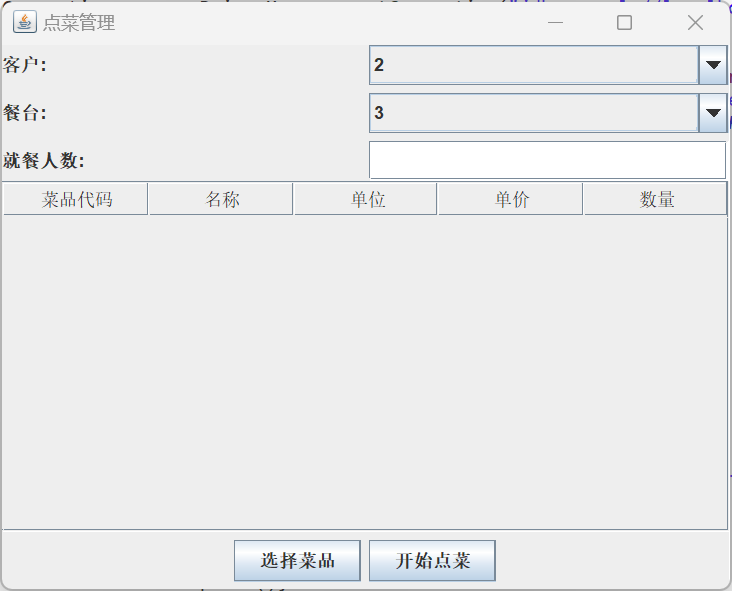
**实现功能：**

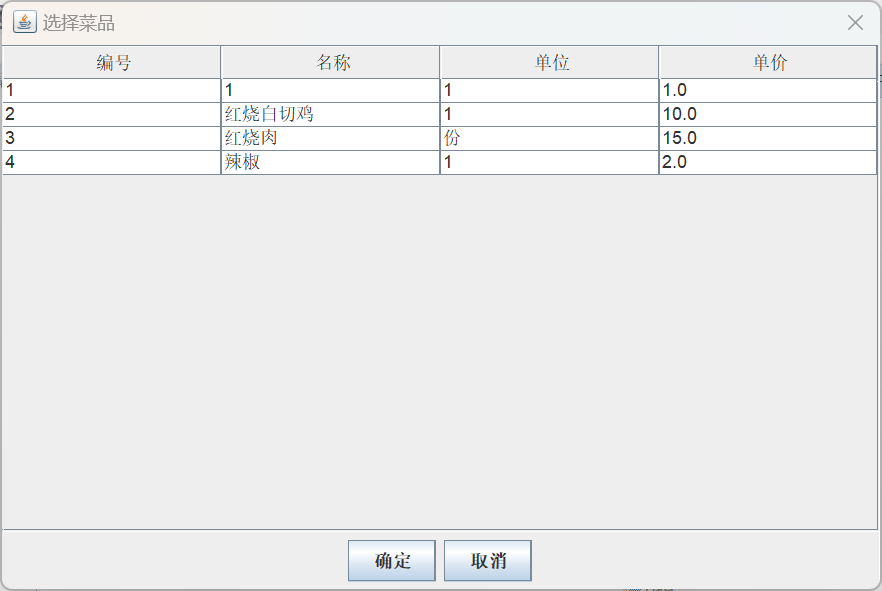
界面1：

1. 选择餐桌号，客户编号和就餐人数，点击选择菜品。
2. 点击选择菜品跳转至界面2

界面2：

1. 选择菜品，点击确认，菜品将添加到订单中

****

****

3.3.11 用户接口10（支付管理界面）设计

**界面主要控件：**

JButton，按钮，触发点击事件。

JTextField，输入框，接收客户名，餐桌号，应收金额，实收金额。

**实现功能：**

1. 选择餐桌号，客户名，对应到相应订单。
2. 输入实收金额，输出找零金额。
3. 点击确认，将对订单进行支付操作。



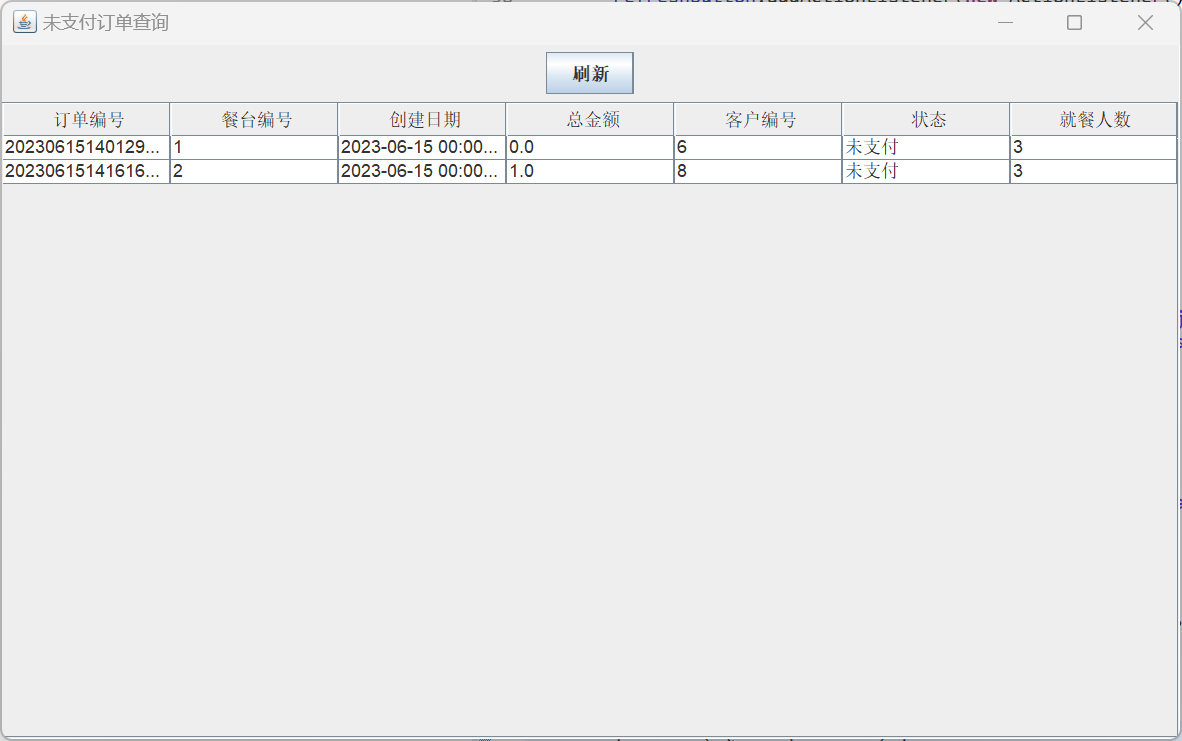
3.3.12 用户接口11（结账管理界面）设计

**界面主要控件：**

JButton，按钮，触发点击事件。

**实现功能：**

1. 点击刷新，显示未支付状态的订单。



## 3.4 登录模块设计说明

（1）功能：该模块主要是管理员对餐饮管理系统进行登录操作。

（2）测试要点：

1. 输入的账号或密码为空将提示账号或密码为空。

2. 输入的账号密码错误则提示账号密码有误。

（3）程序流程图如图所示：

## wps

登录模块程序流程图

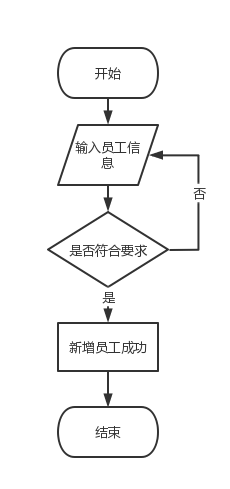
## 3.5 员工管理模块设计说明

（1）功能：实现管理员对员工的增删改查。

1. 新增员工，程序流程图如图所示：

测试要点：

1. 输入员工的信息，若员工信息无误且完整，点击添加，则提示新增成功。
2. 若信息不完整，且信息有误，系统会给出提醒。



新增员工信息流程图

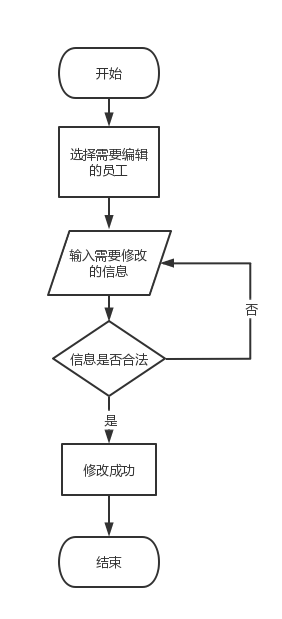
1. 删除员工，程序流程图如图所示：



删除员工流程图

测试要点：

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



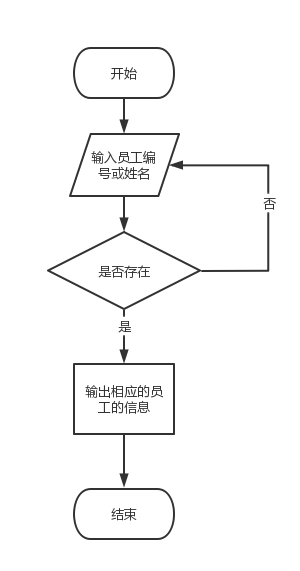
修改员工信息流程图

测试要点：

1. 在文本框中填写要修改的信息，若输入的信息合法则提示修改成功
2. 若输入的信息不合法，则提示信息不合法。
3. 查询员工，程序流程图如图所示：

测试要点：

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

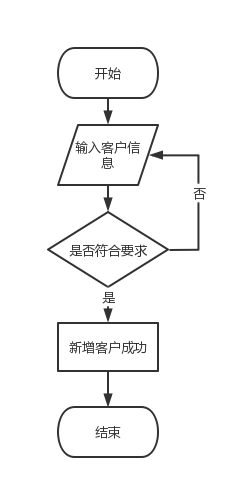


查询员工信息流程图

## 3.6 客户管理模块设计说明

（1）功能：实现管理员对客户的增删改查。

1. 新增客户，程序流程图如图所示：



新增客户流程图

测试要点：

1. 输入客户的信息，若客户信息无误且完整，点击添加，则提示新增成功。
2. 若信息不完整，且信息有误，系统会给出提醒。
3. 删除客户，程序流程图如图3.19所示：



图3.19删除客户流程图

测试要点：

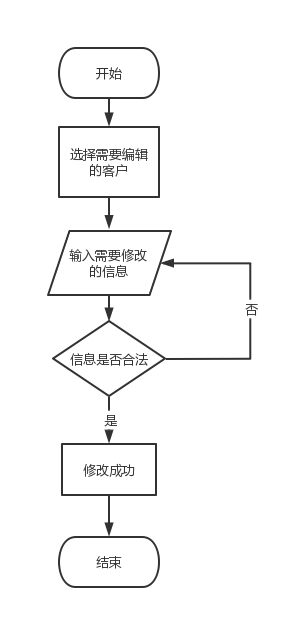
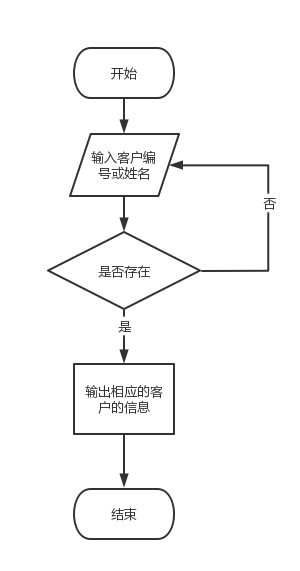
1. 点击表格中客户的信息，点击删除，系统提示删除成功。
2. 修改客户，程序流程图如图所示：

测试要点：

1. 在文本框中填写要修改的信息，点击保存，若输入的信息合法则提示修改成功。
2. 若输入的信息不合法，则提示信息不合法。
3. 查询客户，程序流程图如图所示：

测试要点：

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

修改客户流程图 查询客户流程图

## 3.7 餐台管理模块设计说明

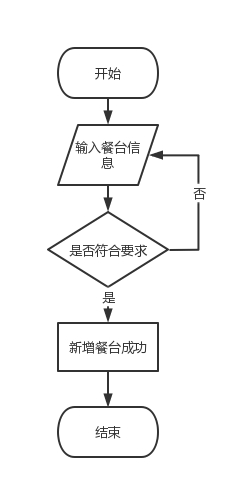


图3.22新增餐台流程图

测试要点：

1. 输入餐台的座位数，若输入为数字且合法，点击添加，则新增餐台成功，否则提示新增失败。
2. 删除餐台，程序流程图如图3.23所示：



图3.23删除餐台流程图

测试要点：

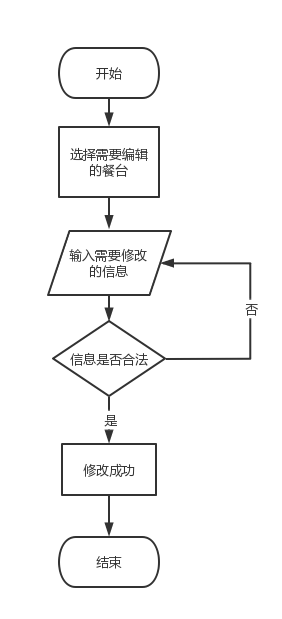
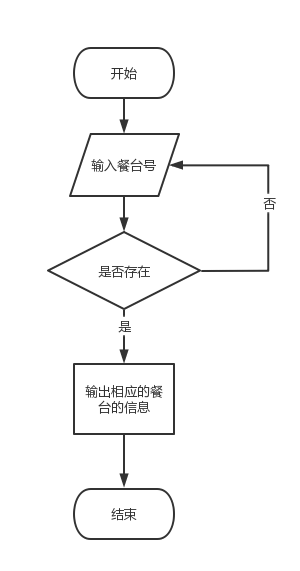
1. 点击餐台信息，点击删除按钮，系统提示删除成功
2. 修改餐台，程序流程图如所示：

测试要点：

1. 在文本框中填写要修改的信息，点击保存，若输入的信息合法则提示修改成功。
2. 若输入的信息不合法，则提示信息不合法。
3. 查询餐台，程序流程图如图所示：

测试要点：

1. 在文本框中输入餐台号，点击查询按钮。
2. 查询信息将在表格中显示。

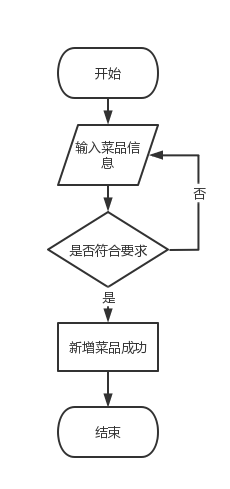
 

修改餐台流程图 查询餐台流程图

## 3.8 菜品管理模块设计说明

（1）功能：实现管理员对菜品信息的增删改查

1. 新增菜品，程序流程图如图所示：



新增菜品程序流程图

测试要点：

1. 输入菜品的信息，点击添加，若菜品信息无误且完整，则提示新增成功。
2. 若信息不完整，且信息有误，系统会给出提醒。
3. 删除菜品，程序流程图如图3.27所示：



图3.27删除菜品流程图

测试要点：

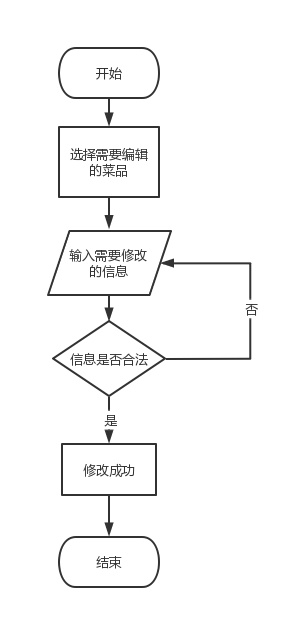
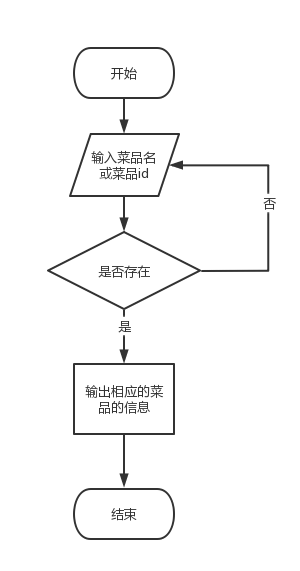
1. 点击菜品信息，点击删除按钮，系统提示删除成功。
2. 修改菜品，程序流程图如图3.28所示：

测试要点：

1. 在文本框中填写要修改的信息，若输入的信息合法则提示修改成功。
2. 若输入的信息不合法，则提示信息不合法。
3. 查询菜品，程序流程图如图3.29所示：

测试要点：

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

删除菜品程序流程图 查询菜品程序流程图

## 3.9 业务管理模块设计说明

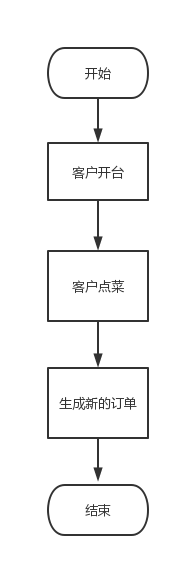
（1）功能：实现管理员对客户进行开台，点单操作。

（2）测试要点：

1. 输入客户的客户号，餐桌号以及人数，为客户开台。

2. 客户选菜，生成新的订单。

（3）程序流程图如图3.30所示：



业务管理程序流程图

## 3.10 支付管理模块设计说明

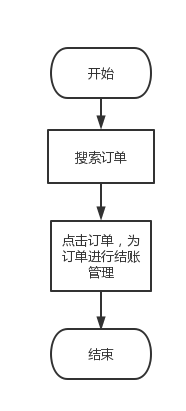
（1）功能：实现管理员对订单进行结账管理。

（2）测试要点：

1. 在输入框中输入餐桌号和客户名，定位到相应订单。

2. 在文本框中输入实收金额，找零

（3）程序流程图如图3.31所示：



支付管理流程图

# 4 数据库设计

## 4.1 数据库环境说明

数据库版本：MySQL 8.0.28

运行环境：Windows 11

## 4.2 数据库的命名规则

数据库名：cysjk

数据表命名规范：

（1）采用26个英文字母（区分大小写）和0-9的自然数（经常不需要）加上下划线’\_’组成，命名简洁明确，多个单词用下划线’\_’分隔。

（2）全部小写命名，禁止出现大写。

（3）禁止使用数据库关键字，如：name,time, datetime,password等。

（4）表名称不应该取得太长（一般不超过三个英文单词）。

（5）表的名称一般使用名词或者动宾短语。

（6）用单数形式表示名称。

（7）表必须填写描述信息（使用SQL语句建表时）。

字段命名规范：

（1）采用26个英文字母（区分大小写）和0-9个自然数（经常不需要）加下划线’\_’组成，命名简洁明确，多个单词用下划线’\_’分隔。

（2）全部小写命名，禁止出现大写。

（3）字段必须填写描述信息。

（4）禁止使用数据库关键字。

（5）字段名称一般采用名词或动宾短语。

（6）采用字段的名称必须是易于理解，一般不超过三个英文单词。

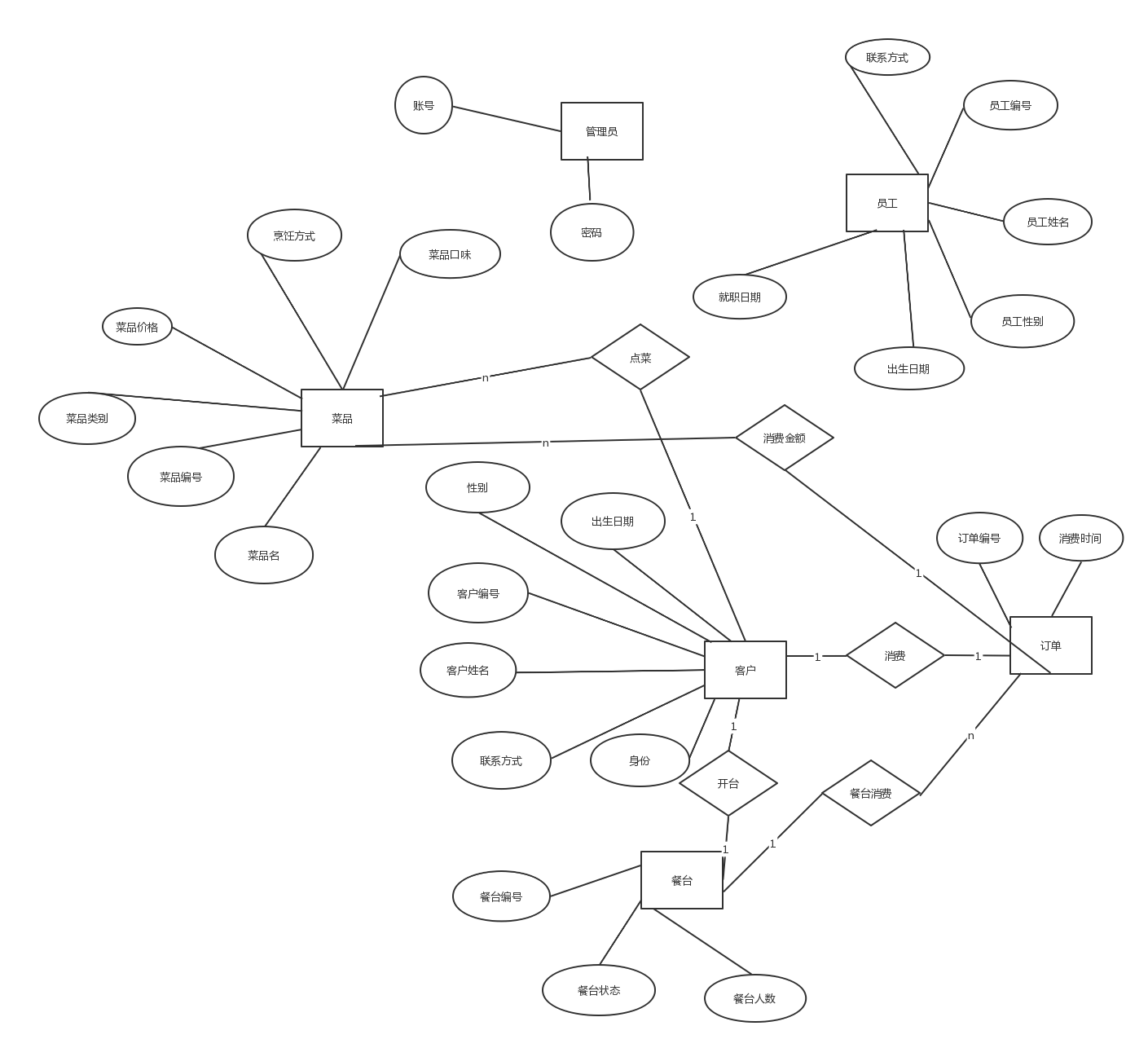
（7）在命名表的列时，不要重复表的名称。

（8）不要在列的名称中包含数据类型。

（9）字段命名使用完整命名，禁止缩写。

## 4.3 逻辑结构设计

E-R图



关系模型（主键用下划线标注，双下划线表示既是主键又是外键）

管理员（用户名，密码）

客户（客户编号，客户姓名，性别，单位，电话，贵宾卡号）

菜品（菜品编号，菜品名，类别编号，图片，菜品代码，单位，价格，状态）

员工（员工编号，员工姓名，员工性别，出生日期，身份证号，家庭住址，电话，职位，是否在职）

餐台（餐台编号，餐台状态，座位数）

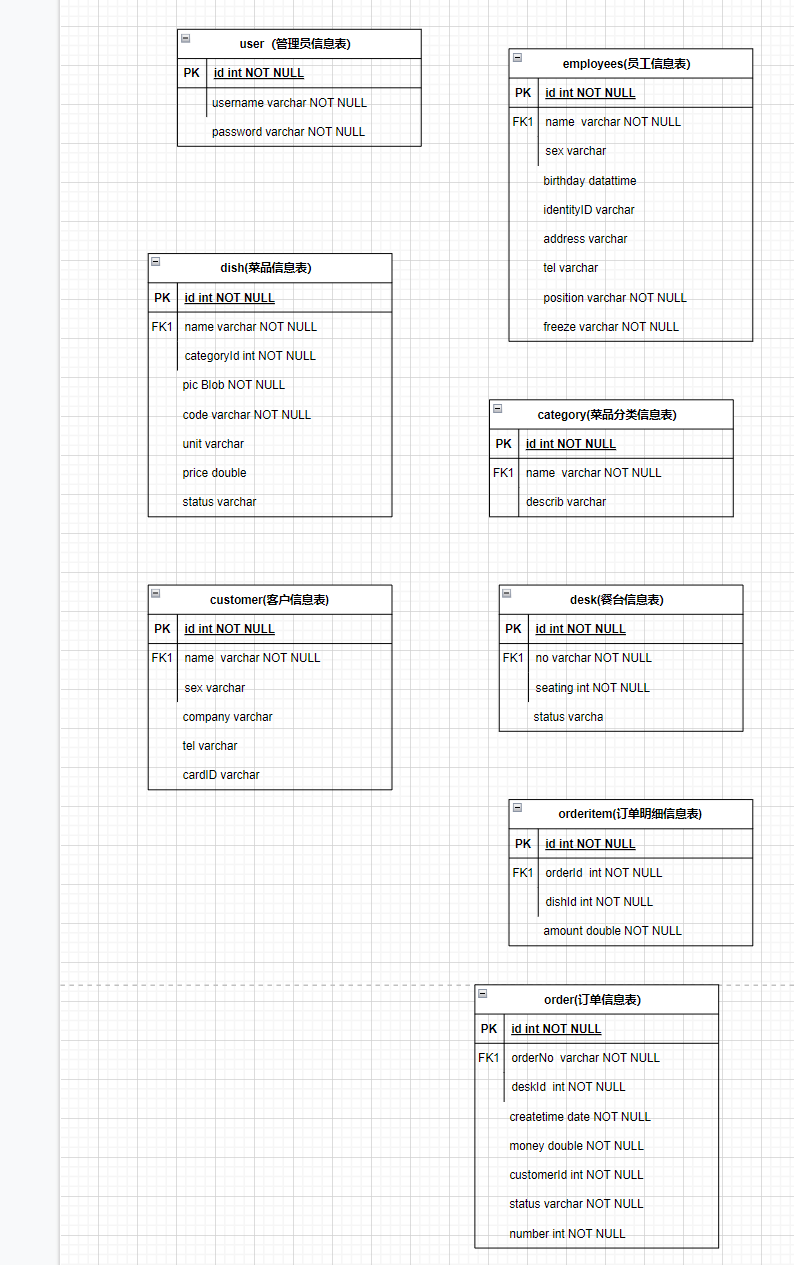
点单（订单编号，菜品编号，数量）

订单（订单编号，餐台编号，客户编号，开台时间，金额，状态，人数）

范式分析

所列关系均为第三范式，因为它消除了非主属性对主属性的传递依赖

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 | 存储管理员账号密码 |
| customer | 存储客户信息 |
| table | 存储餐台信息 |
| order | 存储订单信息 |
| employees | 存储员工信息 |
| dish | 存储菜品信息 |
| category | 存储菜品分类信息 |
| orderitem | 存储订单明细信息 |

表4.2 管理员信息表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | user | | | | | |
| 数据库用户 | | 数据库管理员 | | | | | |
| 主 键 | | id | | | | | |
| 其他排序字段 | | 无 | | | | | |
| 索引字段 | | 无 | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | username | char | N | N |  | 无 | 10位数字 |
| 2 | password | char | N | N |  | 无 | 8-16位数字或字母 |
| sql脚本 | | CREATE TABLE `user` (  `id` int NOT NULL primary key,  `adname` char(20) NOT NULL,  `passstr` char(20) DEFAULT NULL  ) | | | | | |
| 备注 | |  | | | | | |

表4.3 客户信息表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | customer | | | | | |
| 数据库用户 | | 数据库管理员 | | | | | |
| 主 键 | | id | | | | | |
| 其他排序字段 | | 无 | | | | | |
| 索引字段 | | 无 | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | id | char | N | Y |  | 无 | 无 |
| 2 | name | char | Y | N |  | 无 | 无 |
| 3 | sex | char | Y | N |  | 无 | 无 |
| 4 | company | date | Y | N |  | 无 | 无 |
| 5 | tel | char | Y | N |  | 无 | 无 |
| 6 | cardID | char | Y | N |  | 无 | 是、否中的一个 |
| sql脚本 | | CREATE TABLE `customer` (  `id` int NOT NULL primary key,  `name` char(10) DEFAULT NULL,  `sex` char(1) DEFAULT NULL,  `company` date DEFAULT NULL,  `tel` char(11) DEFAULT NULL,  `cardID` char(1) DEFAULT NULL  ) | | | | | |
| 备注 | |  | | | | | |

表4.4 员工信息表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | employees | | | | | |
| 数据库用户 | | 数据库管理员 | | | | | |
| 主 键 | | id | | | | | |
| 其他排序字段 | | 无 | | | | | |
| 索引字段 | | 无 | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | id | int | N | Y |  | 无 | 无 |
| 2 | name | char | Y | N |  | 无 | 无 |
| 3 | sex | char | Y | N |  | 无 | 无 |
| 4 | birthday | date | Y | N |  | 无 | 无 |
| 5 | identityID | char | Y | N |  | 无 | 无 |
| 6 | address | char | Y | N |  | 无 | 无 |
| sql脚本 | | CREATE TABLE `employees` (  `id` int NOT NULL primary key,  `name` char(10) DEFAULT NULL,  `sex` char(1) DEFAULT NULL,  `birthday` char(11) DEFAULT NULL,  `identityID` date DEFAULT NULL,  `address` date DEFAULT NULL ,  ） | | | | | |
| 备注 | |  | | | | | |

表4.5 菜品信息表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | dishes | | | | | |
| 数据库用户 | | 数据库管理员 | | | | | |
| 主 键 | | id | | | | | |
| 其他排序字段 | | 无 | | | | | |
| 索引字段 | | 无 | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | id | int | N | Y |  | 无 | 无 |
| 2 | name | char | Y | N |  | 无 | 无 |
| 3 | categoryID | int | Y | N |  | 无 | 无 |
| 4 | pic | Blob | Y | N |  | 无 | 无 |
| 5 | code | char | Y | N |  | 无 | 无 |
| 6 | unit | char | Y | N |  | 无 | 无 |
| sql脚本 | | CREATE TABLE `dishes` (  `id` int NOT NULL,  `name` char(10) DEFAULT NULL,  `categoryID` int DEFAULT NULL,  `pic` Blob DEFAULT NULL,  `code` char(10) DEFAULT NULL,  `unit` char(10) DEFAULT NULL, | | | | | |
| 备注 | |  | | | | | |

表4.6 餐台信息表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | desk | | | | | |
| 数据库用户 | | 数据库管理员 | | | | | |
| 主 键 | | id | | | | | |
| 其他排序字段 | | 无 | | | | | |
| 索引字段 | | 无 | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | id | int | N | Y |  | 无 | 无 |
| 2 | no | char | Y | N |  | 无 | 无 |
| 3 | seating | int | Y | N |  | 无 | 无 |
| sql脚本 | | CREATE TABLE `desk` (  `id` int NOT NULL primary key,  `no` char(10) DEFAULT NULL,  `seating` int DEFAULT NULL, | | | | | |
| 备注 | |  | | | | | |

表4.7 订单信息表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | order | | | | | |
| 数据库用户 | | 数据库管理员 | | | | | |
| 主 键 | | orderno | | | | | |
| 其他排序字段 | | 无 | | | | | |
| 索引字段 | | 无 | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | orderno | char | N | Y |  | 无 | 外键 |
| 2 | deskid | char | Y | N |  | 无 | 外键 |
| 3 | createtime | datetime | Y | N |  | 无 | 无 |
| 4 | money | int | Y | N |  | 无 | 无 |
| 5 | customerid | char | Y | N |  | 无 | 外键 |
| 6 | state | char | Y | N |  | 无 | 无 |
| 7 | number | int | Y | N |  | 无 | 无 |
| sql脚本 | | CREATE TABLE `orders` (  `orderno` char(20) NOT NULL COMMENT '订单编号' primay key,  `deskid` char(23) DEFAULT NULL COMMENT '餐台号',  `createtime` datetime DEFAULT NULL COMMENT '就餐时间',  `money` int DEFAULT NULL COMMENT '消费金额',  `customerid` char(23) DEFAULT NULL COMMENT '客户编号',  `state` char(3) DEFAULT NULL COMMENT '状态',  `number` int DEFAULT NULL COMMENT '就餐人数',  ) | | | | | |
| 备注 | |  | | | | | |

表4.8 菜品订单信息表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | orderitem | | | | | |
| 数据库用户 | | 数据库管理员 | | | | | |
| 主 键 | | orderid | | | | | |
| 其他排序字段 | | 无 | | | | | |
| 索引字段 | | 无 | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | orderid | int | N | Y |  | 无 | 外键 |
| 2 | dishid | int | N | Y |  | 无 | 外键 |
| 3 | amount | int | N | N |  | 无 | 无 |
| sql脚本 | | CREATE TABLE `orderdishes` (  `orderid` int NOT NULL COMMENT '订单编号' primary key,  `dishid` int NOT NULL COMMENT '菜品号',  `amount` int DEFAULT NULL COMMENT '数量',  ) | | | | | |
| 备注 | |  | | | | | |

## 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

1. 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。

2. 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

表5.1 登录-注册测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 登录 | yl-001 | 输入‘用户名’和‘密码’ | admin,123456 | 登录成功 | 通过 |
| yl-002 | aaa,111 | 登录失败 | 通过 |
| yl-003 | bbb,222 | 登录失败 | 通过 |
| yl-004 | aaa,空 | 登录失败 | 通过 |
| yl-005 | 空,111 | 登录失败 | 通过 |
| yl-006 | bbb,一个空格 | 登录失败 | 通过 |
| yl-007 | 一个空格，222 | 登录失败 | 通过 |

表5.2 员工管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 增加员工 | y2-001 | 点击添加按  钮，输入‘姓  名’，‘性别’，‘出生日期’，‘地址’，‘电话号码’，‘地址’，‘职位’，‘是否在职’ | 1,1,1111-11-11,1,1,1,1,1 | 新增成功 | 通过 |
| 删除员工 | y2-002 | 点击删除按钮 | 1 | 删除成功 | 通过 |
| 修改员工 | y2-003 | 点击表格 | 1->A | 修改成功 | 通过 |
| 查询员工 | y2-004 | 输入员工编  号或姓名，点击查询  按钮 | 1 | 查询成功 | 通过 |

表5.3 客户管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 增加客户 | y3-001 | 点击添加按  钮，输入‘姓  名’，‘性别’，‘出生日期’，‘电话号码’，‘贵宾卡号’ | 梅西，男，巴萨，123456，666 | 新增成功 | 通过 |
| 删除客户 | y3-002 | 点击删除按钮 | 梅西 | 删除成功 | 通过 |
| 修改客户 | y3-003 | 点击表格 | 123456->111111 | 修改成功 | 通过 |
| 查询客户 | y3-004 | 输入员工编  号或姓名，点击查询  按钮 | 梅西 | 查询成功 | 通过 |

表5.4 菜品管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 增加菜品 | y4-001 | 点击添加按  钮，输入‘菜品名’，‘图片’，‘代码’，‘份数’，‘价格’，‘状态’ | 红烧肉，3，图片，3，1，15.0，可用 | 新增成功 | 通过 |
| 删除菜品 | y4-002 | 点击删除按钮 | 红烧肉 | 删除成功 | 通过 |
| 修改菜品 | y4-003 | 点击表格 | 3->4 | 修改成功 | 通过 |
| 查询菜品 | y4-004 | 输入菜品编  号或菜名，点击查询  按钮 | 红烧肉 | 查询成功 | 通过 |

表5.5 餐台管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 增加餐台 | y5-001 | 点击添加按  钮，输入‘座位编号’，’座位数‘，’状态‘ | 1，1，空闲 | 新增成功 | 通过 |
| 删除餐台 | y5-002 | 点击删除按钮 | 1 | 删除成功 | 通过 |
| 修改餐台 | y5-003 | 点击表格 | 1->10 | 修改成功 | 通过 |
| 查询餐台 | y5-004 | 输入菜品编  号或菜名，点击查询  按钮 | 1 | 查询成功 | 通过 |

表5.6 业务管理测试用例

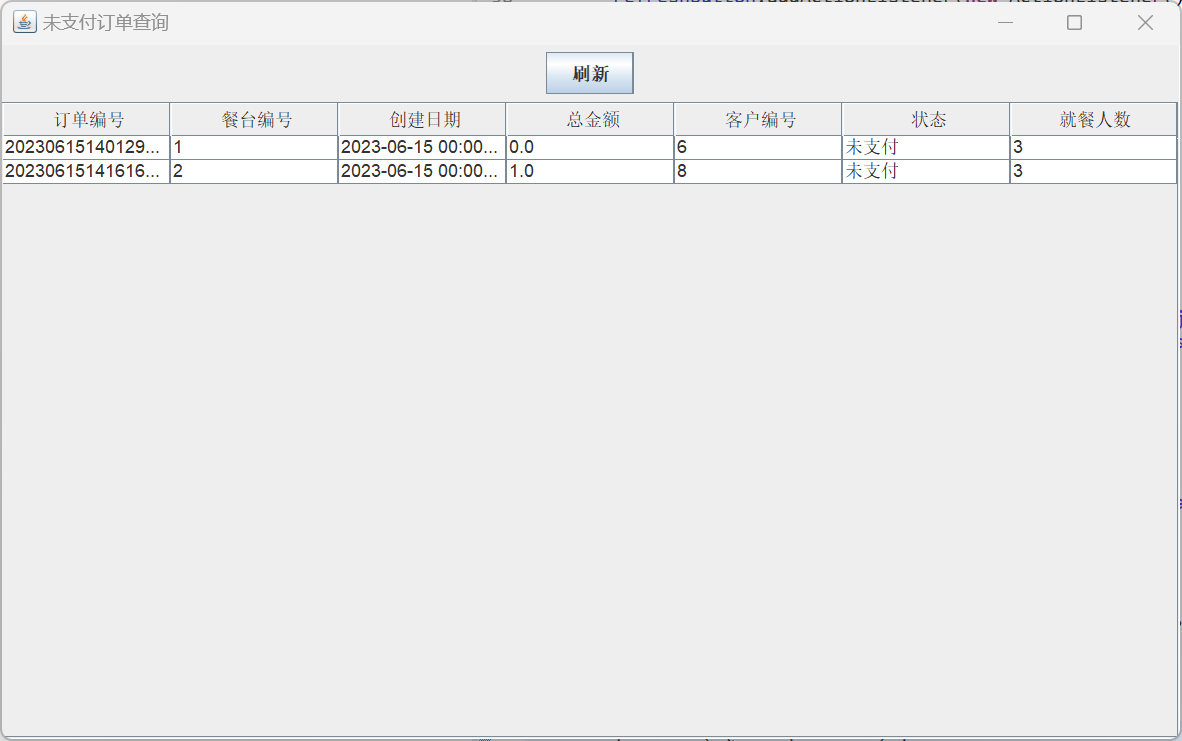
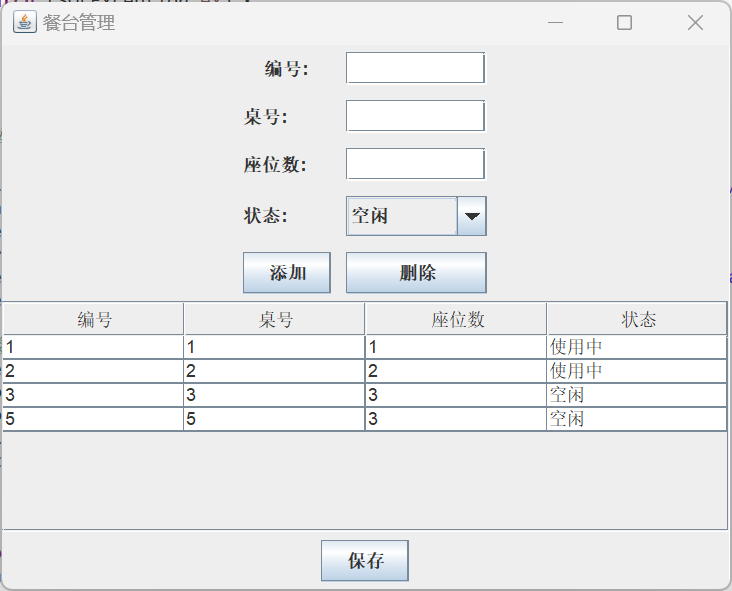
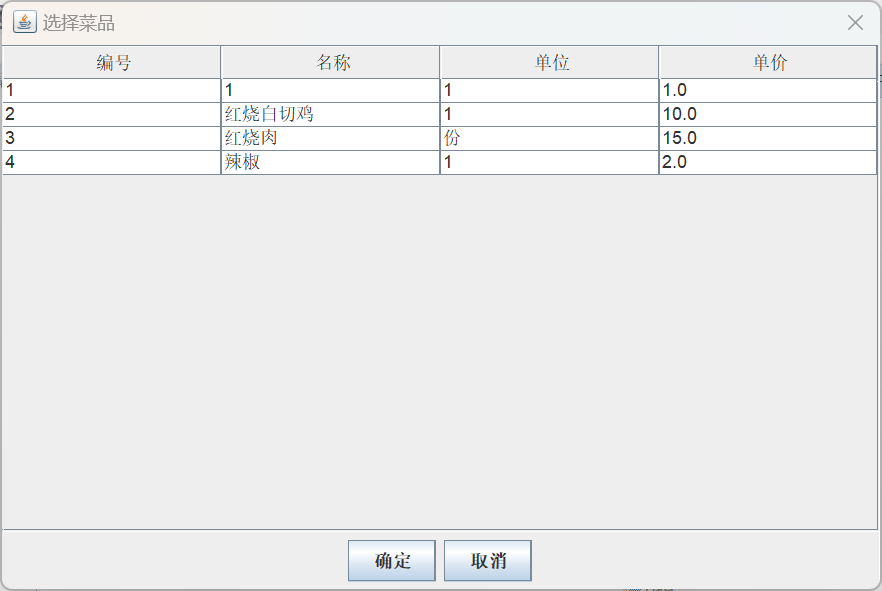
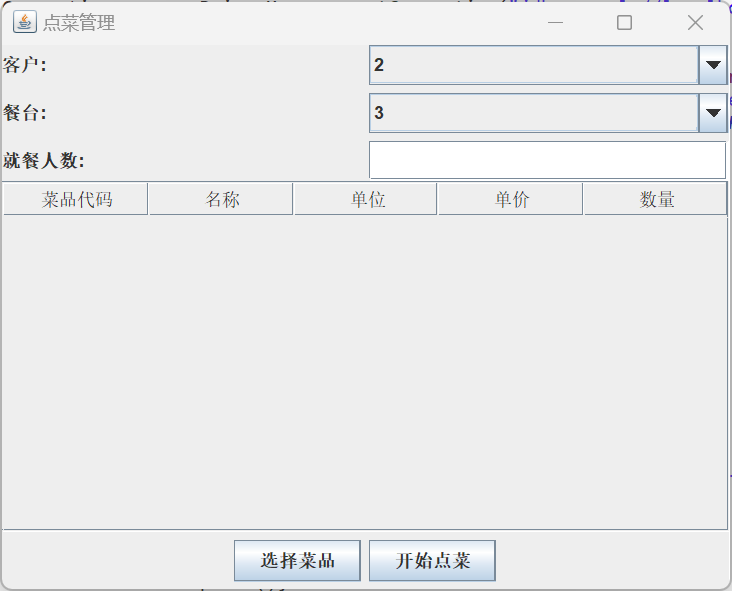
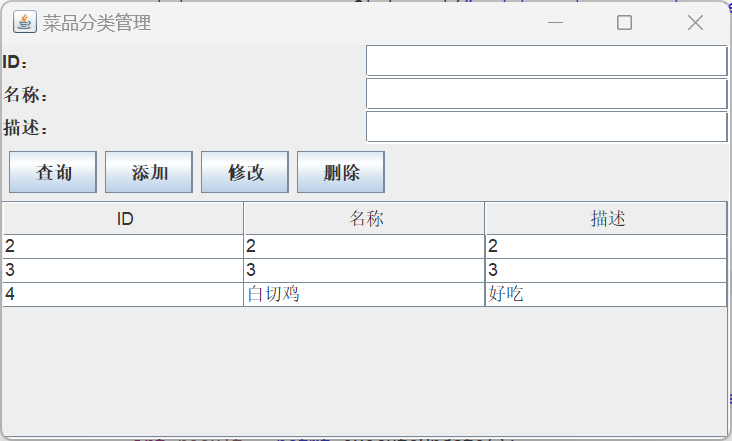
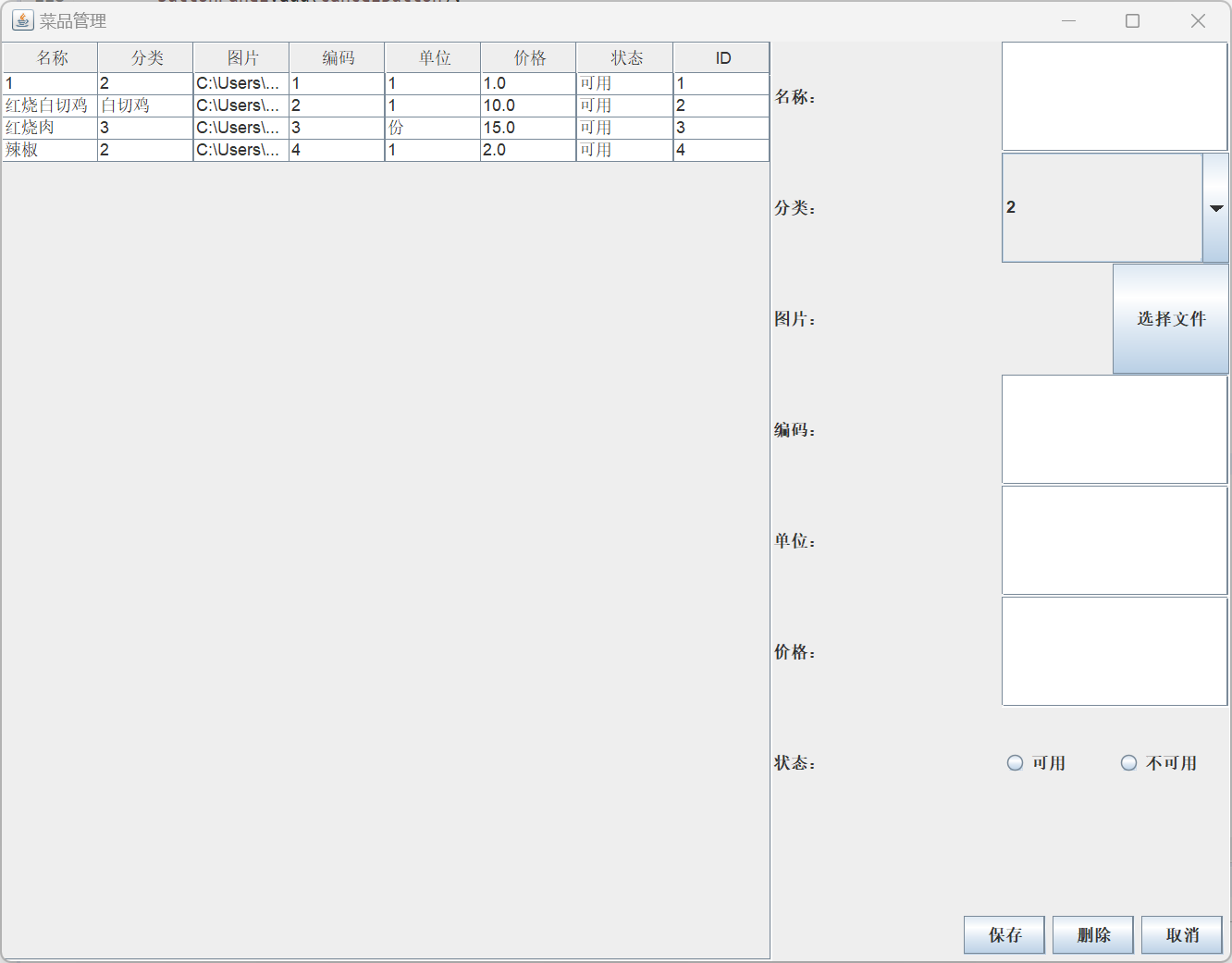
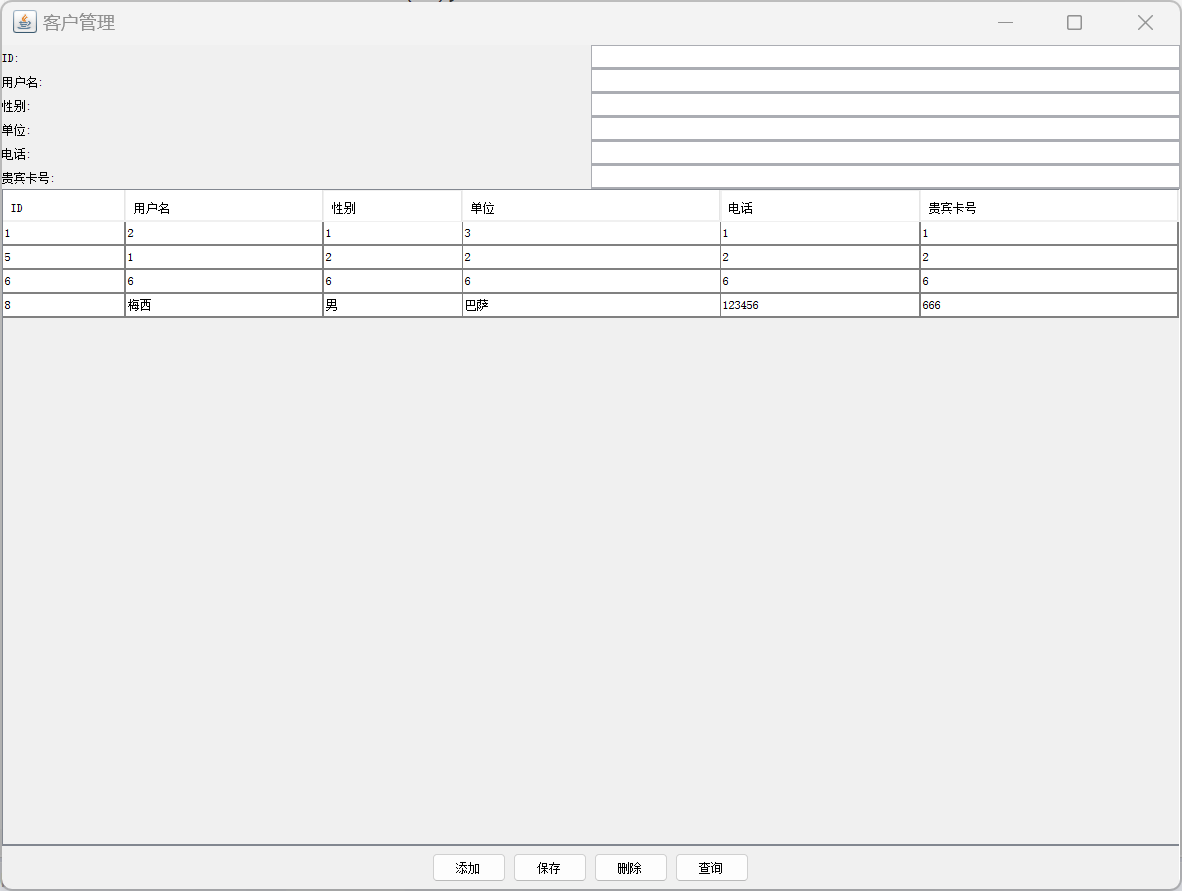
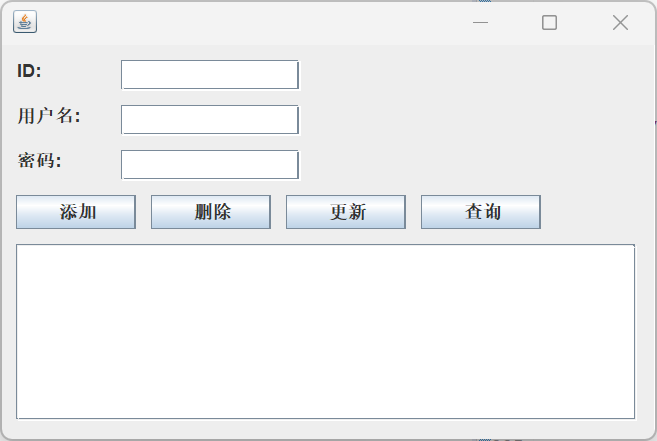
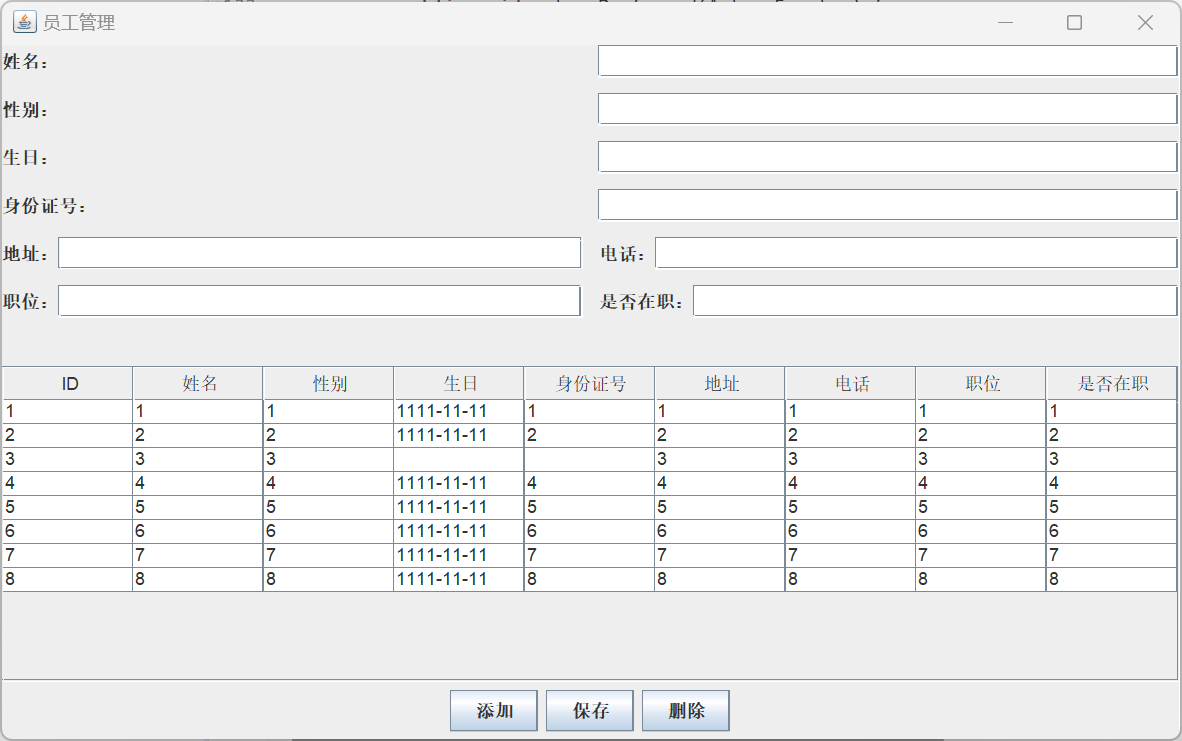
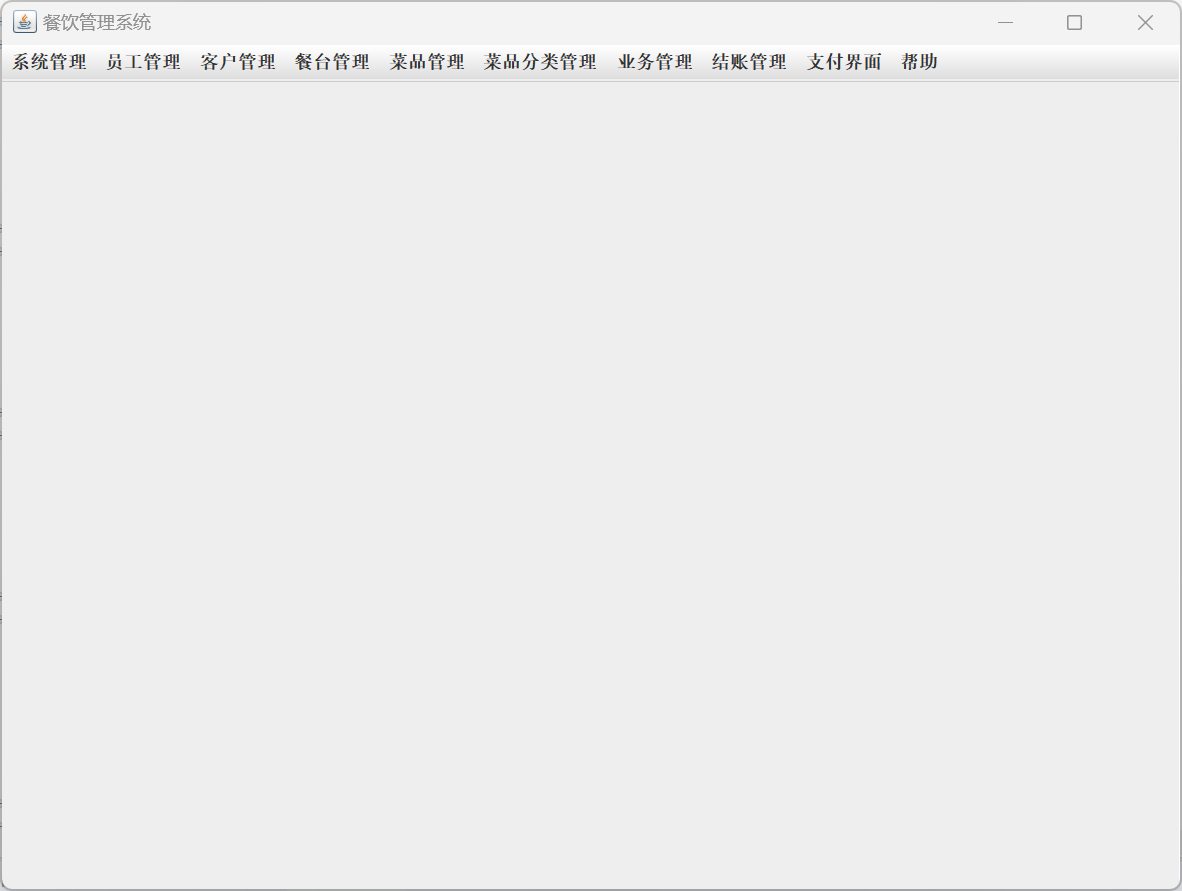
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 开台 | y6-001 | 输入客户名，餐桌号，就餐人数 | 1，1，2 | 开台成功 | 通过 |
| 生成订单 | y6-002 | 点菜后点击保存 | 无 | 操作成功 | 通过 |

表5.7 结账管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 删除订单 | y7-001 | 点击订单后点击删除按钮 | 202306122224223410 | 删除成功 | 通过 |
| 结账操作 | Y7-002 | 点击订单后输入金额 | 15.0 | 支付成功 | 通过 |

# 附录

## 附录1 程序运行结果



## 附录2 程序源代码

LoginUI.java

package zs;7

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.\*;

public class LoginUI extends JFrame {

private static final String DB\_URL = "jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8";

private static final String DB\_USER = "root";

private static final String DB\_PASS = "123456";

private JTextField usernameField;

private JPasswordField passwordField;

private JButton loginButton;

public LoginUI() {

super("欢迎进入餐饮管理系统");

// 创建文本框组件

usernameField = new JTextField(20);

passwordField = new JPasswordField(20);

// 创建按钮组件

loginButton = new JButton("登录");

// 添加事件监听器

loginButton.addActionListener(new LoginButtonListener());

// 创建面板并添加组件

JPanel inputPanel = new JPanel(new GridLayout(2, 2));

inputPanel.add(new JLabel("用户名："));

inputPanel.add(usernameField);

inputPanel.add(new JLabel("密码："));

inputPanel.add(passwordField);

JPanel buttonPanel = new JPanel(new FlowLayout());

buttonPanel.add(loginButton);

// 添加组件到窗口中

getContentPane().add(inputPanel, BorderLayout.NORTH);

getContentPane().add(buttonPanel, BorderLayout.SOUTH);

// 设置窗口大小和位置

setSize(400, 150);

setLocationRelativeTo(null);

}

private boolean validateUser(String username, String password) {

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASS)) {

String sql = "SELECT COUNT(\*) FROM user WHERE username=? AND password=?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, username);

stmt.setString(2, password);

ResultSet rs = stmt.executeQuery();

rs.next();

return rs.getInt(1) > 0;

} catch (SQLException ex) {

ex.printStackTrace();

return false;

}

}

private class LoginButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

if (validateUser(username, password)) {

JOptionPane.showMessageDialog(LoginUI.this, "登录成功");

dispose(); // 关闭登录窗口

// TODO: 跳转到主界面

MainFrame mainFrame = new MainFrame();

mainFrame.setVisible(true);

} else {

JOptionPane.showMessageDialog(LoginUI.this, "用户名或密码错误", "错误", JOptionPane.ERROR\_MESSAGE);

passwordField.setText("");

passwordField.requestFocus();

}

}

}

public static void main(String[] args) {

LoginUI frame = new LoginUI();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

MainFrame.java

package zs;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class MainFrame extends JFrame {

private JMenuBar menuBar;

private JMenu systemMenu;

private JMenuItem userMenuItem;

private JMenuItem tableMenuItem;

private JMenuItem dishMenuItem;

private JMenu businessMenu;

private JMenuItem orderMenuItem;

private JMenuItem payMenuItem;

private JMenu helpMenu;

private JMenuItem aboutMenuItem;

private JMenu tableMenu;

private JMenu dishMenu;

private JMenu payMenu;

private JMenu employeeMenu;

private JMenuItem employeeMenuItem;

private JMenu customerMenu;

private JMenuItem customerMenuItem;

private JMenu ChangeMenu;

private JMenuItem ChangeMenuItem;

private JMenu categoryMenu;

private JMenuItem categoryMenuItem;

public MainFrame() {

this.setTitle("餐饮管理系统");

this.setSize(800, 600);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE); // 修改为DO\_NOTHING\_ON\_CLOSE，不执行默认关闭操作

// 菜单栏

menuBar = new JMenuBar();

systemMenu = new JMenu("系统管理");

userMenuItem = new JMenuItem("用户管理");

tableMenuItem = new JMenuItem("餐台管理");

dishMenu = new JMenu("菜品管理");

dishMenuItem = new JMenuItem("菜品管理");

dishMenu.add(dishMenuItem);

systemMenu.add(userMenuItem);

tableMenu = new JMenu("餐台管理");

tableMenu.add(tableMenuItem);

businessMenu = new JMenu("业务管理");

orderMenuItem = new JMenuItem("订单管理");

payMenuItem = new JMenuItem("结账管理");

payMenu = new JMenu("结账管理");

employeeMenu = new JMenu("员工管理");

employeeMenuItem = new JMenuItem("员工管理");

employeeMenu.add(employeeMenuItem);

customerMenu = new JMenu("客户管理");

customerMenuItem = new JMenuItem("客户管理");

customerMenu.add(customerMenuItem);

ChangeMenu = new JMenu("支付界面");

ChangeMenuItem = new JMenuItem("支付界面");

ChangeMenu.add(ChangeMenuItem);

categoryMenu = new JMenu("菜品分类管理");

categoryMenuItem = new JMenuItem("菜品分类管理");

categoryMenu.add(categoryMenuItem);

businessMenu.add(orderMenuItem);

payMenu.add(payMenuItem);

helpMenu = new JMenu("帮助");

aboutMenuItem = new JMenuItem("关于");

helpMenu.add(aboutMenuItem);

menuBar.add(systemMenu);

menuBar.add(employeeMenu);

menuBar.add(customerMenu);

menuBar.add(tableMenu);

menuBar.add(dishMenu);

menuBar.add(categoryMenu);

menuBar.add(businessMenu);

menuBar.add(payMenu);

menuBar.add(ChangeMenu);

menuBar.add(helpMenu);

this.setJMenuBar(menuBar);

userMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

UserFrame userFrame = new UserFrame();

userFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

userFrame.setVisible(true);

}

});

employeeMenuItem.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

EmployeeFrame employeeFrame = new EmployeeFrame();

employeeFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

employeeFrame.setVisible(true);

}

});

customerMenuItem.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

CustomerFrame customerFrame = new CustomerFrame();

customerFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

customerFrame.setVisible(true);

}

});

tableMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

TableFrame tableFrame = new TableFrame();

tableFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

tableFrame.setVisible(true);

}

});

dishMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

DishFrame dishFrame = new DishFrame();

dishFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

dishFrame.setVisible(true);

}

});

categoryMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

CategoryFrame categoryFrame = new CategoryFrame();

categoryFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

categoryFrame.setVisible(true);

}

});

orderMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

OrderFrame orderFrame = new OrderFrame();

orderFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

orderFrame.setVisible(true);

}

});

payMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

UnpaidOrderFrame payFrame = new UnpaidOrderFrame();

payFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

payFrame.setVisible(true);

}

});

ChangeMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ChangeFrame ChangeFrame = new ChangeFrame();

ChangeFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); // 设置关闭操作为隐藏窗口

ChangeFrame.setVisible(true);

}

});

aboutMenuItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(MainFrame.this, "餐饮管理系统 v1.0\n作者：XXX", "关于", JOptionPane.INFORMATION\_MESSAGE);

}

});

// 添加窗口监听器，防止误关闭主窗口

this.addWindowListener(new java.awt.event.WindowAdapter() {

@Override

public void windowClosing(java.awt.event.WindowEvent e) {

int option = JOptionPane.showConfirmDialog(MainFrame.this, "真的要退出程序吗？", "提示", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

System.exit(0); // 确认关闭

}

}

});

}

public static void main(String[] args) {

new MainFrame().setVisible(true);

}

}

EmployeeFrame.java

**package** zs;

**import** java.awt.BorderLayout;

**import** java.awt.GridLayout;

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

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

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.JTextArea;

**import** javax.swing.JTextField;

**import** javax.swing.SwingUtilities;

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

**public** **class** EmployeeFrame **extends** JFrame {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JPanel contentPane;

**private** JTextField nameField;

**private** JTextField sexField;

**private** JTextField birthdayField;

**private** JTextField identityIDField;

**private** JTextArea addressArea;

**private** JTextField telField;

**private** JTextField positionField;

**private** JTextField freezeField;

**private** JTable table;

**private** DefaultTableModel model;

**private** JButton addButton;

**private** JButton saveButton;

**private** JButton deleteButton;

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

**public** EmployeeFrame() {

setTitle("员工管理");

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 800, 500);

contentPane = **new** JPanel();

setContentPane(contentPane);

contentPane.setLayout(**new** BorderLayout(0, 0));

JPanel inputPanel = **new** JPanel();

contentPane.add(inputPanel, BorderLayout.***NORTH***);

inputPanel.setLayout(**new** GridLayout(7, 1, 10, 10));

JLabel nameLabel = **new** JLabel("姓名：");

inputPanel.add(nameLabel);

nameField = **new** JTextField();

inputPanel.add(nameField);

nameField.setColumns(10);

JLabel sexLabel = **new** JLabel("性别：");

inputPanel.add(sexLabel);

sexField = **new** JTextField();

inputPanel.add(sexField);

sexField.setColumns(10);

JLabel birthdayLabel = **new** JLabel("生日：");

inputPanel.add(birthdayLabel);

birthdayField = **new** JTextField();

inputPanel.add(birthdayField);

birthdayField.setColumns(10);

JLabel identityIDLabel = **new** JLabel("身份证号：");

inputPanel.add(identityIDLabel);

identityIDField = **new** JTextField();

inputPanel.add(identityIDField);

identityIDField.setColumns(18);

JLabel addressLabel = **new** JLabel("地址：");

JPanel addressPanel = **new** JPanel(**new** BorderLayout());

addressPanel.add(addressLabel, BorderLayout.***WEST***);

addressArea = **new** JTextArea();

addressPanel.add(**new** JScrollPane(addressArea), BorderLayout.***CENTER***);

inputPanel.add(addressPanel);

JLabel telLabel = **new** JLabel("电话：");

telField = **new** JTextField();

JPanel telPanel = **new** JPanel(**new** BorderLayout());

telPanel.add(telLabel, BorderLayout.***WEST***);

telPanel.add(telField, BorderLayout.***CENTER***);

inputPanel.add(telPanel);

JLabel positionLabel = **new** JLabel("职位：");

positionField = **new** JTextField();

JPanel positionPanel = **new** JPanel(**new** BorderLayout());

positionPanel.add(positionLabel, BorderLayout.***WEST***);

positionPanel.add(positionField, BorderLayout.***CENTER***);

inputPanel.add(positionPanel);

JLabel freezeLabel = **new** JLabel("是否在职：");

freezeField = **new** JTextField();

JPanel freezePanel = **new** JPanel(**new** BorderLayout());

freezePanel.add(freezeLabel, BorderLayout.***WEST***);

freezePanel.add(freezeField, BorderLayout.***CENTER***);

inputPanel.add(freezePanel);

JPanel buttonPanel = **new** JPanel();

contentPane.add(buttonPanel, BorderLayout.***SOUTH***);

addButton = **new** JButton("添加");

buttonPanel.add(addButton);

saveButton = **new** JButton("保存");

buttonPanel.add(saveButton);

deleteButton = **new** JButton("删除");

buttonPanel.add(deleteButton);

model = **new** DefaultTableModel(**new** Object[][] {}, **new** String[] { "ID", "姓名", "性别", "生日", "身份证号", "地址", "电话", "职位", "是否在职" });

table = **new** JTable(model);

JScrollPane scrollPane = **new** JScrollPane(table);

contentPane.add(scrollPane, BorderLayout.***CENTER***);

loadEmployeeData(); // 加载已有的员工数据

addButton.addActionListener(**new** ActionListener() {

@Override

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

model.addRow(**new** Object[] {0, nameField.getText(), sexField.getText(), birthdayField.getText(), identityIDField.getText(), addressArea.getText(), telField.getText(), positionField.getText(), freezeField.getText()});

clearFields();

}

});

saveButton.addActionListener(**new** ActionListener() {

@Override

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

**try** {

Class.*forName*("com.mysql.cj.jdbc.Driver");

Connection conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=true&characterEncoding=utf-8", "root", "123456");

PreparedStatement stmt = **null**;

**for** (**int** i = 0; i < model.getRowCount(); i++) {

**if** (model.getValueAt(i, 1) != **null** && ((String)model.getValueAt(i, 1)).trim().length() > 0) {

id = (**int**) model.getValueAt(i, 0);

**if** (id != 0) {

stmt = conn.prepareStatement("UPDATE employees SET name=?, sex=?, birthday=?, identityID=?, address=?, tel=?, position=?, freeze=? WHERE id=?");

stmt.setString(1, (String)model.getValueAt(i, 1));

stmt.setString(2, (String)model.getValueAt(i, 2));

stmt.setString(3, (String)model.getValueAt(i, 3));

stmt.setString(4, (String)model.getValueAt(i, 4));

stmt.setString(5, (String)model.getValueAt(i, 5));

stmt.setString(6, (String)model.getValueAt(i, 6));

stmt.setString(7, (String)model.getValueAt(i, 7));

stmt.setString(8, (String)model.getValueAt(i, 8));

stmt.setInt(9, id);

} **else** {

stmt = conn.prepareStatement("INSERT INTO employees (name, sex, birthday, identityID, address, tel, position, freeze) VALUES (?, ?, ?, ?, ?, ?, ?, ?)", Statement.***RETURN\_GENERATED\_KEYS***);

stmt.setString(1, (String)model.getValueAt(i, 1));

stmt.setString(2, (String)model.getValueAt(i, 2));

stmt.setString(3, (String)model.getValueAt(i, 3));

stmt.setString(4, (String)model.getValueAt(i, 4));

stmt.setString(5, (String)model.getValueAt(i, 5));

stmt.setString(6, (String)model.getValueAt(i, 6));

stmt.setString(7, (String)model.getValueAt(i, 7));

stmt.setString(8, (String)model.getValueAt(i, 8));

}

stmt.executeUpdate();

**if** (id == 0) {

ResultSet rs = stmt.getGeneratedKeys();

**if** (rs.next()) {

id = rs.getInt(1);

model.setValueAt(id, i, 0);

}

rs.close();

}

}

}

stmt.close();

conn.close();

JOptionPane.*showMessageDialog*(**null**, "保存成功", "提示", JOptionPane.***INFORMATION\_MESSAGE***);

} **catch** (SQLException | ClassNotFoundException ex) {

ex.printStackTrace();

JOptionPane.*showMessageDialog*(**null**, "保存失败：" + ex.getMessage(),"错误", JOptionPane.***ERROR\_MESSAGE***);

}

}

});

deleteButton.addActionListener(**new** ActionListener() {

@Override

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

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

// 获取选中行的索引，如果没有选中任何行则返回-1。

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

**int** confirm = JOptionPane.*showConfirmDialog*(**null**, "确定删除该员工信息吗？", "确认删除", JOptionPane.***YES\_NO\_OPTION***);

**if** (confirm == JOptionPane.***YES\_OPTION***) {

**int** id = (**int**) model.getValueAt(row, 0);

**try** {

Class.*forName*("com.mysql.cj.jdbc.Driver");

Connection conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=true&characterEncoding=utf-8", "root", "123456");

PreparedStatement stmt = conn.prepareStatement("DELETE FROM employees WHERE id=?");

stmt.setInt(1, id);

stmt.executeUpdate();

stmt.close();

conn.close();

model.removeRow(row);

clearFields();

JOptionPane.*showMessageDialog*(**null**, "删除成功", "提示", JOptionPane.***INFORMATION\_MESSAGE***);

} **catch** (SQLException | ClassNotFoundException ex) {

ex.printStackTrace();

JOptionPane.*showMessageDialog*(**null**, "删除失败：" + ex.getMessage(), "错误", JOptionPane.***ERROR\_MESSAGE***);

}

}

} **else** {

JOptionPane.*showMessageDialog*(**null**, "请先选择要删除的员工信息", "提示", JOptionPane.***WARNING\_MESSAGE***);

}

}

});

table.getSelectionModel().addListSelectionListener(event -> {

**if** (table.getSelectedRow() != -1) {

id = (**int**) model.getValueAt(table.getSelectedRow(), 0);

nameField.setText((String)model.getValueAt(table.getSelectedRow(), 1));

sexField.setText((String)model.getValueAt(table.getSelectedRow(), 2));

birthdayField.setText((String)model.getValueAt(table.getSelectedRow(), 3));

identityIDField.setText((String)model.getValueAt(table.getSelectedRow(), 4));

addressArea.setText((String)model.getValueAt(table.getSelectedRow(), 5));

telField.setText((String)model.getValueAt(table.getSelectedRow(), 6));

positionField.setText((String)model.getValueAt(table.getSelectedRow(), 7));

freezeField.setText((String)model.getValueAt(table.getSelectedRow(), 8));

}

});

}

**private** **void** clearFields() {

nameField.setText(**null**);

sexField.setText(**null**);

birthdayField.setText(**null**);

identityIDField.setText(**null**);

addressArea.setText(**null**);

telField.setText(**null**);

positionField.setText(**null**);

freezeField.setText(**null**);

}

**private** **void** loadEmployeeData() {

**try** {

Class.*forName*("com.mysql.cj.jdbc.Driver");

Connection conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=true&characterEncoding=utf-8", "root", "123456");

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM employees");

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

**int** id = rs.getInt("id");

String name = rs.getString("name");

String sex = rs.getString("sex");

String birthday = rs.getString("birthday");

String identityID = rs.getString("identityID");

String address = rs.getString("address");

String tel = rs.getString("tel");

String position = rs.getString("position");

String freeze = rs.getString("freeze");

model.addRow(**new** Object[] {id, name, sex, birthday, identityID, address, tel, position, freeze});

}

rs.close();

stmt.close();

conn.close();

} **catch** (SQLException | ClassNotFoundException ex) {

ex.printStackTrace();

JOptionPane.*showMessageDialog*(**null**, "加载数据失败：" + ex.getMessage(),"错误", JOptionPane.***ERROR\_MESSAGE***);

}

}

**public** **static** **void** main(String[] args) {

SwingUtilities.*invokeLater*(() -> {

EmployeeFrame frame = **new** EmployeeFrame();

frame.setVisible(**true**);

});

}

}

CustomerFrame.java

package zs;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.table.\*;

public class CustomerFrame extends JFrame {

private static final String DB\_URL = "jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8";

private static final String DB\_USER = "root";

private static final String DB\_PASS = "123456";

private JTextField idField, nameField, sexField, companyField, telField, cardIdField;

private JButton addButton, updateButton, deleteButton, searchButton;

private JTable table;

public CustomerFrame() {

super("客户管理");

// 创建表格模型

DefaultTableModel model = new DefaultTableModel(new Object[][] {},

new String[] { "ID", "用户名", "性别", "单位", "电话", "贵宾卡号" });

// 创建表格组件

table = new JTable(model);

JScrollPane scrollPane = new JScrollPane(table);

// 设置表格列宽

table.getColumnModel().getColumn(0).setPreferredWidth(30);

table.getColumnModel().getColumn(1).setPreferredWidth(80);

table.getColumnModel().getColumn(2).setPreferredWidth(40);

table.getColumnModel().getColumn(3).setPreferredWidth(120);

table.getColumnModel().getColumn(4).setPreferredWidth(80);

table.getColumnModel().getColumn(5).setPreferredWidth(120);

// 创建文本框组件

idField = new JTextField(10);

nameField = new JTextField(20);

sexField = new JTextField(5);

companyField = new JTextField(30);

telField = new JTextField(20);

cardIdField = new JTextField(30);

// 创建按钮组件

addButton = new JButton("添加");

updateButton = new JButton("保存");

deleteButton = new JButton("删除");

searchButton = new JButton("查询");

// 添加事件监听器

addButton.addActionListener(new AddButtonListener());

updateButton.addActionListener(new UpdateButtonListener());

deleteButton.addActionListener(new DeleteButtonListener());

searchButton.addActionListener(new SearchButtonListener());

// 创建面板并添加组件

JPanel inputPanel = new JPanel(new GridLayout(6, 2));

inputPanel.add(new JLabel("ID:"));

inputPanel.add(idField);

inputPanel.add(new JLabel("用户名:"));

inputPanel.add(nameField);

inputPanel.add(new JLabel("性别:"));

inputPanel.add(sexField);

inputPanel.add(new JLabel("单位:"));

inputPanel.add(companyField);

inputPanel.add(new JLabel("电话:"));

inputPanel.add(telField);

inputPanel.add(new JLabel("贵宾卡号:"));

inputPanel.add(cardIdField);

JPanel buttonPanel = new JPanel(new FlowLayout());

buttonPanel.add(addButton);

buttonPanel.add(updateButton);

buttonPanel.add(deleteButton);

buttonPanel.add(searchButton);

// 添加组件到窗口中

getContentPane().add(inputPanel, BorderLayout.NORTH);

getContentPane().add(scrollPane, BorderLayout.CENTER);

getContentPane().add(buttonPanel, BorderLayout.SOUTH);

// 设置窗口大小和位置

setSize(800, 600);

setLocationRelativeTo(null);

// 加载表格数据

loadTableData();

}

private void clearFields() {

idField.setText("");

nameField.setText("");

sexField.setText("");

companyField.setText("");

telField.setText("");

cardIdField.setText("");

}

private void loadTableData() {

DefaultTableModel model = (DefaultTableModel) table.getModel();

// 清空表格数据

model.setRowCount(0);

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASS)) {

String sql = "SELECT \* FROM customer";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String sex = rs.getString("sex");

String company = rs.getString("company");

String tel = rs.getString("tel");

String cardId = rs.getString("cardID");

model.addRow(new Object[] { id, name, sex, company, tel, cardId });

}

} catch (SQLException ex) {

ex.printStackTrace();

}

}

private class AddButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String idText = idField.getText();

if (idText.isEmpty()) {

JOptionPane.showMessageDialog(CustomerFrame.this, "请输入有效的ID", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

int id;

try {

id = Integer.parseInt(idText);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(CustomerFrame.this, "请输入有效的ID", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

String name = nameField.getText();

String sex = sexField.getText();

String company = companyField.getText();

String tel = telField.getText();

String cardId = cardIdField.getText();

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASS)) {

String sql = "INSERT INTO customer (id, name, sex, company, tel, cardID) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, id);

stmt.setString(2, name);

stmt.setString(3, sex);

stmt.setString(4, company);

stmt.setString(5, tel);

stmt.setString(6, cardId);

stmt.executeUpdate();

loadTableData();

clearFields();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

private class UpdateButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String idText = idField.getText();

if (idText.isEmpty()) {

JOptionPane.showMessageDialog(CustomerFrame.this, "请输入有效的ID", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

int id;

try {

id = Integer.parseInt(idText);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(CustomerFrame.this, "请输入有效的ID", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASS)) {

String sql = "SELECT \* FROM customer WHERE id=?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, id);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

String name = rs.getString("name");

String sex = rs.getString("sex");

String company = rs.getString("company");

String tel = rs.getString("tel");

String cardId = rs.getString("cardID");

// 获取当前记录的其他字段值

String newName = nameField.getText();

String newSex = sexField.getText();

String newCompany = companyField.getText();

String newTel = telField.getText();

String newCardId = cardIdField.getText();

// 只更新修改的字段，保持其他字段不变

if (!newName.isEmpty()) {

name = newName;

}

if (!newSex.isEmpty()) {

sex = newSex;

}

if (!newCompany.isEmpty()) {

company = newCompany;

}

if (!newTel.isEmpty()) {

tel = newTel;

}

if (!newCardId.isEmpty()) {

cardId = newCardId;

}

// 执行更新操作

sql = "UPDATE customer SET name=?, sex=?, company=?, tel=?, cardID=? WHERE id=?";

stmt = conn.prepareStatement(sql);

stmt.setString(1, name);

stmt.setString(2, sex);

stmt.setString(3, company);

stmt.setString(4, tel);

stmt.setString(5, cardId);

stmt.setInt(6, id);

stmt.executeUpdate();

JOptionPane.showMessageDialog(CustomerFrame.this, "记录已成功更新", "提示", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(CustomerFrame.this, "无法找到指定的ID记录", "错误", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

private class DeleteButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String idText = idField.getText();

int id = -1;

if (!idText.isEmpty()) {

try {

id = Integer.parseInt(idText);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(CustomerFrame.this, "请输入有效的ID", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

}

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASS)) {

String sql;

PreparedStatement stmt;

if (id != -1) {

// 通过ID删除记录

sql = "DELETE FROM customer WHERE id=?";

stmt = conn.prepareStatement(sql);

stmt.setInt(1, id);

} else {

// 通过选定行删除记录

int selectedRow = table.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(CustomerFrame.this, "请选择要删除的行或输入有效的ID", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

id = (int) table.getValueAt(selectedRow, 0);

sql = "DELETE FROM customer WHERE id=?";

stmt = conn.prepareStatement(sql);

stmt.setInt(1, id);

}

stmt.executeUpdate();

loadTableData();

clearFields();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

private class SearchButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String searchText = nameField.getText().trim();

DefaultTableModel model = (DefaultTableModel) table.getModel();

model.setRowCount(0);

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASS)) {

String sql = "SELECT \* FROM customer WHERE id=? OR name=? OR sex=? OR company=? OR tel=? OR cardID=?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, searchText);

stmt.setString(2, searchText);

stmt.setString(3, searchText);

stmt.setString(4, searchText);

stmt.setString(5, searchText);

stmt.setString(6, searchText);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String sex = rs.getString("sex");

String company = rs.getString("company");

String tel = rs.getString("tel");

String cardId = rs.getString("cardID");

if (searchText.equals(String.valueOf(id)) || searchText.equalsIgnoreCase(name) ||

searchText.equalsIgnoreCase(sex) || searchText.equalsIgnoreCase(company) ||

searchText.equalsIgnoreCase(tel) || searchText.equalsIgnoreCase(cardId)) {

model.addRow(new Object[]{id, name, sex, company, tel, cardId});

}

}

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (Exception e) {

e.printStackTrace();

}

CustomerFrame frame = new CustomerFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

});

}

}

DishFrame.java

package zs;

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import java.io.Serializable;

class Category implements Serializable {

private int id;

private String name;

public Category(int id, String name) {

this.id = id;

this.name = name;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

@Override

public String toString() {

return name;

}

}

public class DishFrame extends JFrame {

private JTable table;

private DefaultTableModel model;

private JTextField nameField;

private JComboBox categoryComboBox;

private JLabel imageLabel;

private String imagePath;

private JTextField codeField;

private JTextField unitField;

private JTextField priceField;

private JRadioButton availableRadio;

private JRadioButton unavailableRadio;

private int selectedId = -1;

public DishFrame() {

setTitle("菜品管理");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(900, 700);

JPanel panel = new JPanel(new BorderLayout());

getContentPane().add(panel);

// 表格

table = new JTable();

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

// 表单

JPanel formPanel = new JPanel(new BorderLayout());

JPanel fieldPanel = new JPanel(new GridLayout(0, 2));

formPanel.add(fieldPanel, BorderLayout.NORTH);

fieldPanel.add(new JLabel("名称："));

nameField = new JTextField();

fieldPanel.add(nameField);

fieldPanel.add(new JLabel("分类："));

categoryComboBox = new JComboBox();

loadCategories();

fieldPanel.add(categoryComboBox);

fieldPanel.add(new JLabel("图片："));

JPanel imagePanel = new JPanel(new BorderLayout());

imageLabel = new JLabel();

imageLabel.setPreferredSize(new Dimension(80, 80));

imagePanel.add(imageLabel, BorderLayout.WEST);

JButton imageButton = new JButton("选择文件");

imagePanel.add(imageButton, BorderLayout.CENTER);

fieldPanel.add(imagePanel);

fieldPanel.add(new JLabel("编码："));

codeField = new JTextField();

fieldPanel.add(codeField);

fieldPanel.add(new JLabel("单位："));

unitField = new JTextField();

fieldPanel.add(unitField);

fieldPanel.add(new JLabel("价格："));

priceField = new JTextField();

fieldPanel.add(priceField);

fieldPanel.add(new JLabel("状态："));

JPanel statusPanel = new JPanel(new GridLayout(0, 2));

ButtonGroup statusGroup = new ButtonGroup();

availableRadio = new JRadioButton("可用");

unavailableRadio = new JRadioButton("不可用");

statusGroup.add(availableRadio);

statusGroup.add(unavailableRadio);

statusPanel.add(availableRadio);

statusPanel.add(unavailableRadio);

fieldPanel.add(statusPanel);

JButton saveButton = new JButton("保存");

JButton deleteButton = new JButton("删除");

JButton cancelButton = new JButton("取消");

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

buttonPanel.add(saveButton);

buttonPanel.add(deleteButton);

buttonPanel.add(cancelButton);

formPanel.add(buttonPanel, BorderLayout.SOUTH);

panel.add(formPanel, BorderLayout.EAST);

// 加载数据

model = new DefaultTableModel(new Object[]{"名称", "分类", "图片", "编码", "单位", "价格", "状态", "ID"}, 0) {

@Override

public boolean isCellEditable(int row, int column) {

return false;

}

};

table.setModel(model);

table.getSelectionModel().addListSelectionListener(new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if (!e.getValueIsAdjusting()) {

int row = table.getSelectedRow();

if (row != -1) {

selectedId = (int) model.getValueAt(row, 7);

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

Category category = (Category) model.getValueAt(row, 1);

categoryComboBox.setSelectedItem(category);

String pic = (String) model.getValueAt(row, 2);

if (pic != null && !pic.isEmpty()) {

imagePath = pic;

ImageIcon icon = new ImageIcon(imagePath);

imageLabel.setIcon(icon);

} else {

imageLabel.setIcon(null);

}

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

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

priceField.setText(model.getValueAt(row, 5).toString());

String status = (String) model.getValueAt(row, 6);

if (status.equals("可用")) {

availableRadio.setSelected(true);

} else {

unavailableRadio.setSelected(true);

}

}

}

}

});

saveButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (nameField.getText().isEmpty()) {

JOptionPane.showMessageDialog(null, "名称不能为空", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

if (categoryComboBox.getSelectedItem() == null) {

JOptionPane.showMessageDialog(null, "分类不能为空", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

if (selectedId == -1 && imagePath == null) {

JOptionPane.showMessageDialog(null, "请选择一个图片文件，或者先添加一个新的菜品", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

String name = nameField.getText().trim();

Category category = (Category) categoryComboBox.getSelectedItem();

String code = codeField.getText().trim();

String unit = unitField.getText().trim();

double price = Double.parseDouble(priceField.getText().trim());

String status = availableRadio.isSelected() ? "可用" : "不可用";

if (selectedId == -1) {

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

String sql = "INSERT INTO dish(name, categoryId, code, unit, price, status, pic) VALUES(?, ?, ?, ?, ?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, name);

stmt.setInt(2, category.getId());

stmt.setString(3, code);

stmt.setString(4, unit);

stmt.setDouble(5, price);

stmt.setString(6, status);

stmt.setString(7, imagePath);

stmt.executeUpdate();

stmt.close();

conn.close();

JOptionPane.showMessageDialog(null, "添加成功！", "提示", JOptionPane.INFORMATION\_MESSAGE);

loadDishes();

resetFields();

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "添加失败：" + ex.getMessage(), "错误", JOptionPane.ERROR\_MESSAGE);

}

} else {

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

String sql = "UPDATE dish SET name=?, categoryId=?, code=?, unit=?, price=?, status=?, pic=? WHERE id=?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, name);

stmt.setInt(2, category.getId());

stmt.setString(3, code);

stmt.setString(4, unit);

stmt.setDouble(5, price);

stmt.setString(6, status);

stmt.setString(7, imagePath);

stmt.setInt(8, selectedId);

stmt.executeUpdate();

stmt.close();

conn.close();

JOptionPane.showMessageDialog(null, "修改成功！", "提示", JOptionPane.INFORMATION\_MESSAGE);

loadDishes();

resetFields();

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "修改失败：" + ex.getMessage(), "错误", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (selectedId == -1) {

JOptionPane.showMessageDialog(null, "请选择要删除的记录", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

int dialogResult = JOptionPane.showConfirmDialog(null, "确定要删除吗？", "删除记录", JOptionPane.YES\_NO\_OPTION);

if (dialogResult == JOptionPane.YES\_OPTION) {

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

String sql = "DELETE FROM dish WHERE id=?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, selectedId);

stmt.executeUpdate();

stmt.close();

conn.close();

JOptionPane.showMessageDialog(null, "删除成功！", "提示", JOptionPane.INFORMATION\_MESSAGE);

loadDishes();

resetFields();

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "删除失败：" + ex.getMessage(), "错误", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

cancelButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

resetFields();

}

});

imageButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter("图片文件", "jpg", "jpeg", "png", "gif");

fileChooser.setFileFilter(filter);

int result = fileChooser.showOpenDialog(null);

if (result == JFileChooser.APPROVE\_OPTION) {

imagePath = fileChooser.getSelectedFile().getAbsolutePath();

ImageIcon icon = new ImageIcon(imagePath);

imageLabel.setIcon(icon);

}

}

});

setVisible(true);

setLocationRelativeTo(null);

loadDishes();

}

private void loadCategories() {

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

String sql = "SELECT \* FROM category";

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

Category category = new Category(rs.getInt("id"), rs.getString("name"));

categoryComboBox.addItem(category);

}

rs.close();

stmt.close();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "加载分类失败：" + ex.getMessage(), "错误", JOptionPane.ERROR\_MESSAGE);

}

}

private void loadDishes() {

model.setRowCount(0);

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

String sql = "SELECT dish.\*, category.name AS categoryName FROM dish INNER JOIN category ON dish.categoryId=category.id";

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

Category category = new Category(rs.getInt("categoryId"), rs.getString("categoryName"));

model.addRow(new Object[]{

rs.getString("name"),

category,

rs.getString("pic"),

rs.getString("code"),

rs.getString("unit"),

rs.getDouble("price"),

rs.getString("status"),

rs.getInt("id")

});

}

rs.close();

stmt.close();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "加载数据失败：" + ex.getMessage(), "错误", JOptionPane.ERROR\_MESSAGE);

}

}

private void resetFields() {

selectedId = -1;

nameField.setText("");

categoryComboBox.setSelectedIndex(0);

imageLabel.setIcon(null);

imagePath = null;

codeField.setText("");

unitField.setText("");

priceField.setText("");

availableRadio.setSelected(true);

}

public static void main(String[] args) {

DishFrame frame = new DishFrame();

}

}

DishDialog.java

**package** zs;

**import** javax.swing.\*;

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

**import** java.awt.\*;

**import** java.sql.\*;

**public** **class** DishDialog **extends** JDialog {

**private** OrderFrame orderFrame;

**private** DefaultTableModel model;

**private** JTable table;

**public** DishDialog(OrderFrame orderFrame) {

**super**(orderFrame, "选择菜品", **true**);

**this**.orderFrame = orderFrame;

setSize(600, 400);

// 表格

String[] columnNames = {"编号", "名称", "单位", "单价"};

model = **new** DefaultTableModel(columnNames, 0);

table = **new** JTable(model);

JScrollPane scrollPane = **new** JScrollPane(table);

getContentPane().add(scrollPane, BorderLayout.***CENTER***);

// 加载菜品数据到表格中

**try** {

Connection conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/cysjk?useSSL=false", "root", "123456");

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM dish");

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

Object[] rowData = {rs.getString("code"), rs.getString("name"), rs.getString("unit"), rs.getDouble("price")};

model.addRow(rowData);

}

rs.close();

stmt.close();

conn.close();

} **catch** (SQLException ex) {

ex.printStackTrace();

}

// 确定和取消按钮

JPanel buttonPanel = **new** JPanel();

JButton okButton = **new** JButton("确定");

buttonPanel.add(okButton);

JButton cancelButton = **new** JButton("取消");

buttonPanel.add(cancelButton);

getContentPane().add(buttonPanel, BorderLayout.***SOUTH***);

okButton.addActionListener(e -> {

**int**[] rows = table.getSelectedRows();

**if** (rows.length == 0) {

JOptionPane.*showMessageDialog*(**null**, "请选择要点的菜品", "提示", JOptionPane.***WARNING\_MESSAGE***);

**return**;

}

DefaultTableModel orderModel = orderFrame.getOrderTableModel();

**for** (**int** row : rows) {

String code = (String)model.getValueAt(row, 0);

String name = (String)model.getValueAt(row, 1);

String unit = (String)model.getValueAt(row, 2);

**double** price = (**double**)model.getValueAt(row, 3);

Object[] rowData = {code, name, unit, price, 1};

orderModel.addRow(rowData);

}

dispose();

});

cancelButton.addActionListener(e -> dispose());

}

}

CategoryFrame.java

package zs;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class CategoryFrame extends JFrame implements ActionListener {

private JLabel label1, label2, label3;

private JTextField tf1, tf2, tf3;

private JButton btn1, btn2, btn3, btn4;

private JTable table;

private DefaultTableModel model;

private Connection conn;

private PreparedStatement pstmt;

private ResultSet rs;

public CategoryFrame() {

super("菜品分类管理");

// 初始化界面组件

label1 = new JLabel("ID：");

label2 = new JLabel("名称：");

label3 = new JLabel("描述：");

tf1 = new JTextField(10);

tf2 = new JTextField(10);

tf3 = new JTextField(10);

btn1 = new JButton("查询");

btn2 = new JButton("添加");

btn3 = new JButton("修改");

btn4 = new JButton("删除");

JPanel p1 = new JPanel(new GridLayout(3, 2));

p1.add(label1);

p1.add(tf1);

p1.add(label2);

p1.add(tf2);

p1.add(label3);

p1.add(tf3);

JPanel p2 = new JPanel(new FlowLayout(FlowLayout.LEFT));

p2.add(btn1);

p2.add(btn2);

p2.add(btn3);

p2.add(btn4);

model = new DefaultTableModel();

model.addColumn("ID");

model.addColumn("名称");

model.addColumn("描述");

table = new JTable(model);

JScrollPane sp = new JScrollPane(table);

JPanel p3 = new JPanel(new BorderLayout());

p3.add(sp, BorderLayout.CENTER);

p3.add(p2, BorderLayout.NORTH);

add(p1, BorderLayout.NORTH);

add(p3, BorderLayout.CENTER);

// 添加事件监听器

btn1.addActionListener(this);

btn2.addActionListener(this);

btn3.addActionListener(this);

btn4.addActionListener(this);

// 连接数据库并显示数据

try {

Class.forName("com.mysql.cj.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/cysjk", "root", "123456");

displayData();

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}

setSize(500, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btn1) { // 查询操作

int id = Integer.parseInt(tf1.getText().trim());

try {

pstmt = conn.prepareStatement("select \* from category where id=?");

pstmt.setInt(1, id);

rs = pstmt.executeQuery();

if (rs.next()) {

tf2.setText(rs.getString("name"));

tf3.setText(rs.getString("describ"));

} else {

JOptionPane.showMessageDialog(null, "未找到该分类！");

}

} catch (SQLException ex) {

ex.printStackTrace();

}

} else if (e.getSource() == btn2) { // 添加操作

String name = tf2.getText().trim();

String describ = tf3.getText().trim();

try {

pstmt = conn.prepareStatement("insert into category(name, describ) values (?, ?)");

pstmt.setString(1, name);

pstmt.setString(2, describ);

int result = pstmt.executeUpdate();

if (result > 0) {

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

displayData();

} else {

JOptionPane.showMessageDialog(null, "添加失败！");

}

} catch (SQLException ex) {

ex.printStackTrace();

}

} else if (e.getSource() == btn3) { // 修改操作

int id = Integer.parseInt(tf1.getText().trim());

String name = tf2.getText().trim();

String describ = tf3.getText().trim();

try {

pstmt = conn.prepareStatement("update category set name=?, describ=? where id=?");

pstmt.setString(1, name);

pstmt.setString(2, describ);

pstmt.setInt(3, id);

int result = pstmt.executeUpdate();

if (result > 0) {

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

displayData();

} else {

JOptionPane.showMessageDialog(null, "修改失败！");

}

} catch (SQLException ex) {

ex.printStackTrace();

}

} else if (e.getSource() == btn4) { // 删除操作

int id = Integer.parseInt(tf1.getText().trim());

try {

pstmt = conn.prepareStatement("delete from category where id=?");

pstmt.setInt(1, id);

int result = pstmt.executeUpdate();

if (result > 0) {

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

displayData();

} else {

JOptionPane.showMessageDialog(null, "删除失败！");

}

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

private void displayData() throws SQLException {

model.setRowCount(0);

pstmt = conn.prepareStatement("select \* from category");

rs = pstmt.executeQuery();

while (rs.next()) {

Object[] rowData = new Object[3];

rowData[0] = rs.getInt("id");

rowData[1] = rs.getString("name");

rowData[2] = rs.getString("describ");

model.addRow(rowData);

}

}

public static void main(String[] args) {

new CategoryFrame();

}

}

TableFrame.java

package zs;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class TableFrame extends JFrame {

private JLabel idLabel;

private JTextField idField;

private JLabel noLabel;

private JTextField noField;

private JLabel seatingLabel;

private JTextField seatingField;

private JLabel statusLabel;

private JComboBox<String> statusComboBox;

private JButton addButton;

private JButton deleteButton;

private JButton saveButton;

private JTable table;

private DefaultTableModel model;

public TableFrame() {

this.setTitle("餐台管理");

this.setSize(500, 400);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();

idLabel = new JLabel("编号:");

c.gridx = 0;

c.gridy = 0;

c.insets = new Insets(5, 5, 5, 5);

panel.add(idLabel, c);

idField = new JTextField(10);

c.gridx = 1;

c.gridy = 0;

c.fill = GridBagConstraints.HORIZONTAL;

panel.add(idField, c);

noLabel = new JLabel("桌号:");

c.gridx = 0;

c.gridy = 1;

panel.add(noLabel, c);

noField = new JTextField(10);

c.gridx = 1;

c.gridy = 1;

panel.add(noField, c);

seatingLabel = new JLabel("座位数:");

c.gridx = 0;

c.gridy = 2;

panel.add(seatingLabel, c);

seatingField = new JTextField(10);

c.gridx = 1;

c.gridy = 2;

panel.add(seatingField, c);

statusLabel = new JLabel("状态:");

c.gridx = 0;

c.gridy = 3;

panel.add(statusLabel, c);

statusComboBox = new JComboBox<String>();

statusComboBox.addItem("空闲");

statusComboBox.addItem("使用中");

c.gridx = 1;

c.gridy = 3;

panel.add(statusComboBox, c);

addButton = new JButton("添加");

c.gridx = 0;

c.gridy = 4;

c.gridwidth = 1;

panel.add(addButton, c);

deleteButton = new JButton("删除");

c.gridx = 1;

c.gridy = 4;

c.gridwidth = 1;

panel.add(deleteButton, c);

this.add(panel, BorderLayout.NORTH);

// 表格

String[] columnNames = {"编号", "桌号", "座位数", "状态"};

model = new DefaultTableModel(columnNames, 0);

table = new JTable(model);

JScrollPane scrollPane = new JScrollPane(table);

this.add(scrollPane, BorderLayout.CENTER);

// 按钮

JPanel buttonPanel = new JPanel();

saveButton = new JButton("保存");

buttonPanel.add(saveButton);

this.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String idStr = idField.getText();

String no = noField.getText();

String seatsStr = seatingField.getText();

String status = (String)statusComboBox.getSelectedItem();

if (idStr.isEmpty() || no.isEmpty() || seatsStr.isEmpty()) {

JOptionPane.showMessageDialog(null, "请输入编号、桌号和座位数", "提示", JOptionPane.WARNING\_MESSAGE);

return;

}

try {

int id = Integer.parseInt(idStr);

int seats = Integer.parseInt(seatsStr);

// 添加到表格中

Object[] rowData = {id, no, seats, status};

model.addRow(rowData);

// 添加到数据库中

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

String sql = "INSERT INTO desk(id, no, seating, status) VALUES(?, ?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, id);

stmt.setString(2, no);

stmt.setInt(3, seats);

stmt.setString(4, status);

stmt.executeUpdate();

// 关闭数据库连接

stmt.close();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(null, "编号和座位数必须是整数", "提示", JOptionPane.WARNING\_MESSAGE);

return;

}

}

});

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int rowIndex = table.getSelectedRow();

if (rowIndex == -1) {

JOptionPane.showMessageDialog(null, "请选择要删除的行", "提示", JOptionPane.WARNING\_MESSAGE);

return;

}

// 从表格中删除

model.removeRow(rowIndex);

// 从数据库中删除

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

String sql = "DELETE FROM desk WHERE id=" + table.getValueAt(rowIndex, 0);

Statement stmt = conn.createStatement();

stmt.executeUpdate(sql);

// 关闭数据库连接

stmt.close();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

});

saveButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

// 先清空表中数据

Statement stmt = conn.createStatement();

stmt.executeUpdate("DELETE FROM desk");

// 再保存表格中所有数据

for (int i = 0; i < model.getRowCount(); i++) {

int id = (int)model.getValueAt(i, 0);

String no = (String)model.getValueAt(i, 1);

int seats = (int)model.getValueAt(i, 2);

String status = (String)model.getValueAt(i, 3);

String sql = "INSERT INTO desk(id, no, seating, status) VALUES(" + id + ", '" + no + "', " + seats + ", '" + status + "')";

stmt.executeUpdate(sql);

}

// 关闭数据库连接

stmt.close();

conn.close();

JOptionPane.showMessageDialog(null, "保存成功", "提示", JOptionPane.INFORMATION\_MESSAGE);

} catch (SQLException ex) {

ex.printStackTrace();

}

}

});

// 加载数据库中的数据到表格中

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=true&useUnicode=yes&characterEncoding=UTF-8", "root", "123456");

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM desk");

while (rs.next()) {

Object[] rowData = {rs.getInt("id"), rs.getString("no"), rs.getInt("seating"), rs.getString("status")};

model.addRow(rowData);

}

// 关闭数据库连接

rs.close();

stmt.close();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

public static void main(String[] args) {

new TableFrame().setVisible(true);

}

}

UserFrame.java

package zs;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

public class UserFrame extends JFrame {

static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

static final String DB\_URL = "jdbc:mysql://localhost:3306/cysjk";

static final String USERNAME = "root";

static final String PASSWORD = "123456";

private JPanel contentPane;

private JTextField idField;

private JTextField usernameField;

private JTextField passwordField;

private Connection conn = null;

private PreparedStatement pstmt = null;

public UserFrame() {

try {

// 加载 MySQL JDBC 驱动程序

Class.forName(JDBC\_DRIVER);

// 建立数据库连接

conn = DriverManager.getConnection(DB\_URL, USERNAME, PASSWORD);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 450, 300);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblId = new JLabel("ID:");

lblId.setBounds(10, 10, 60, 15);

contentPane.add(lblId);

idField = new JTextField();

idField.setBounds(80, 10, 120, 21);

contentPane.add(idField);

idField.setColumns(10);

JLabel lblUsername = new JLabel("用户名:");

lblUsername.setBounds(10, 40, 60, 15);

contentPane.add(lblUsername);

usernameField = new JTextField();

usernameField.setBounds(80, 40, 120, 21);

contentPane.add(usernameField);

usernameField.setColumns(10);

JLabel lblPassword = new JLabel("密码:");

lblPassword.setBounds(10, 70, 60, 15);

contentPane.add(lblPassword);

passwordField = new JTextField();

passwordField.setBounds(80, 70, 120, 21);

contentPane.add(passwordField);

passwordField.setColumns(10);

JButton addButton = new JButton("添加");

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int id = Integer.parseInt(idField.getText());

String username = usernameField.getText();

String password = passwordField.getText();

addUser(id, username, password);

}

});

addButton.setBounds(10, 100, 80, 23);

contentPane.add(addButton);

JButton deleteButton = new JButton("删除");

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int id = Integer.parseInt(idField.getText());

deleteUser(id);

}

});

deleteButton.setBounds(100, 100, 80, 23);

contentPane.add(deleteButton);

JButton updateButton = new JButton("更新");

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int id = Integer.parseInt(idField.getText());

String username = usernameField.getText();

String password = passwordField.getText();

updateUser(id, username, password);

}

});

updateButton.setBounds(190, 100, 80, 23);

contentPane.add(updateButton);

JButton queryButton = new JButton("查询");

queryButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int id = Integer.parseInt(idField.getText());

queryUser(id);

}

});

queryButton.setBounds(280, 100, 80, 23);

contentPane.add(queryButton);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(10, 133, 414, 118);

contentPane.add(scrollPane);

JTextArea textArea = new JTextArea();

textArea.setEditable(false);

scrollPane.setViewportView(textArea);

// 添加窗口关闭事件监听器，释放资源

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

close();

}

});

}

public void addUser(int id, String username, String password) {

try {

// 实现插入数据操作

String sql = "INSERT INTO user(id, username, password) VALUES (?, ?, ?)";

pstmt = conn.prepareStatement(sql);

pstmt.setInt(1, id);

pstmt.setString(2, username);

pstmt.setString(3, password);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(null, "插入数据成功");

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "插入数据失败，请检查输入是否正确");

} finally {

// 关闭PreparedStatement对象

if(pstmt != null) {

try {

pstmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

public void deleteUser(int id) {

try {

// 实现删除数据操作

String sql = "DELETE FROM user WHERE id=?";

pstmt = conn.prepareStatement(sql);

pstmt.setInt(1, id);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(null, "删除数据成功");

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "删除数据失败，请检查输入是否正确");

} finally {

// 关闭PreparedStatement对象

if(pstmt != null) {

try {

pstmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

public void updateUser(int id, String username, String password) {

try {

// 实现更新数据操作

String sql = "UPDATE user SET username=?, password=? WHERE id=?";

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, username);

pstmt.setString(2, password);

pstmt.setInt(3, id);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(null, "更新数据成功");

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "更新数据失败，请检查输入是否正确");

} finally {

// 关闭PreparedStatement对象

if(pstmt != null) {

try {

pstmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

public void queryUser(int id) {

try {

// 实现查询数据操作

String sql = "SELECT \* FROM user WHERE id=?";

pstmt = conn.prepareStatement(sql);

pstmt.setInt(1, id);

ResultSet rs = pstmt.executeQuery();

JScrollPane scrollPane = (JScrollPane) contentPane.getComponentAt(10, 133);

JTextArea textArea = (JTextArea) scrollPane.getViewport().getView();

textArea.setText("");

while(rs.next()) {

int userId = rs.getInt("id");

String username = rs.getString("username");

String password = rs.getString("password");

textArea.append("id: " + userId + ", username:" + username + ", password: " + password + "");

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "查询数据失败，请检查输入是否正确");

}finally { // 关闭PreparedStatement对象

if(pstmt != null) {

try {

pstmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

public void close() { // 关闭PreparedStatement对象

if(pstmt != null) {

try {

pstmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

public static void main(String[] args) {

UserFrame frame = new UserFrame();

frame.setVisible(true);

}

}

UnpaidOrderFrame.java

package zs;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class UnpaidOrderFrame extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JTable table;

private DefaultTableModel model;

private JButton refreshButton;

public UnpaidOrderFrame() {

setTitle("未支付订单查询");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 800, 500);

contentPane = new JPanel();

setContentPane(contentPane);

contentPane.setLayout(new BorderLayout(0, 0));

JPanel buttonPanel = new JPanel();

contentPane.add(buttonPanel, BorderLayout.NORTH);

refreshButton = new JButton("刷新");

buttonPanel.add(refreshButton);

model = new DefaultTableModel(new Object[][] {}, new String[] {"订单编号", "餐台编号", "创建日期", "总金额", "客户编号", "状态", "就餐人数"});

table = new JTable(model);

JScrollPane scrollPane = new JScrollPane(table);

contentPane.add(scrollPane, BorderLayout.CENTER);

refreshButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

refreshTable();

}

private void refreshTable() {

model.setRowCount(0);

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=false", "root", "123456");

String sql = "SELECT \* FROM `order` WHERE status='未支付'";

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

int id = rs.getInt("id");

String orderNo = rs.getString("orderNo");

int tableId = rs.getInt("deskId");

Timestamp createTime = rs.getTimestamp("createtime");

double totalPrice = rs.getDouble("money");

int customerId = rs.getInt("customerId");

String status = rs.getString("status");

int peopleCount = rs.getInt("number");

model.addRow(new Object[] {orderNo, tableId, createTime, totalPrice, customerId, status, peopleCount});

}

rs.close();

stmt.close();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

public static void main(String[] args) {

UnpaidOrderFrame frame = new UnpaidOrderFrame();

frame.setVisible(true);

}

}

OrderFrame.java

package zs;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.math.BigDecimal;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

public class OrderFrame extends JFrame {

private JLabel customerLabel;

private JComboBox<String> customerComboBox;

private JLabel tableLabel;

private JComboBox<String> tableComboBox;

private JLabel numLabel;

private JTextField numField;

private JButton startButton;

private JTable table;

private DefaultTableModel model;

private int customerId = -1; // 添加一个customerId变量

public OrderFrame() {

this.setTitle("点菜管理");

this.setSize(500, 400);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(3, 2, 5, 5));

customerLabel = new JLabel("客户:");

panel.add(customerLabel);

customerComboBox = new JComboBox<String>();

panel.add(customerComboBox);

tableLabel = new JLabel("餐台:");

panel.add(tableLabel);

tableComboBox = new JComboBox<String>();

panel.add(tableComboBox);

numLabel = new JLabel("就餐人数:");

panel.add(numLabel);

numField = new JTextField();

panel.add(numField);

this.add(panel, BorderLayout.NORTH);

// 表格

String[] columnNames = {"菜品代码", "名称", "单位", "单价", "数量"};

model = new DefaultTableModel(columnNames, 0);

table = new JTable(model);

JScrollPane scrollPane = new JScrollPane(table);

this.add(scrollPane, BorderLayout.CENTER);

// 按钮

JPanel buttonPanel = new JPanel();

JButton dishButton = new JButton("选择菜品");

buttonPanel.add(dishButton);

startButton = new JButton("开始点菜");

buttonPanel.add(startButton);

this.add(buttonPanel, BorderLayout.SOUTH);

dishButton.addActionListener(e -> showDishDialog());

customerComboBox.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String selectedCustomer = (String) customerComboBox.getSelectedItem();

customerId = getCustomerId(selectedCustomer);

}

});

startButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String customer = (String)customerComboBox.getSelectedItem();

String table = (String)tableComboBox.getSelectedItem();

String numStr = numField.getText();

if (customer.isEmpty() || table.isEmpty() || numStr.isEmpty()) {

JOptionPane.showMessageDialog(null, "请输入客户、餐台和就餐人数", "提示", JOptionPane.WARNING\_MESSAGE);

return;

}

try {

int num = Integer.parseInt(numStr);

// 计算总金额

double totalMoney = 0.0;

for (int i = 0; i < model.getRowCount(); i++) {

Object value = model.getValueAt(i, 3);

if (value instanceof Double) {

double price = (Double) value;

int quantity = (int)model.getValueAt(i, 4);

totalMoney += price \* quantity;

} else if (value instanceof String) {

double price = Double.parseDouble((String)value);

int quantity = (int)model.getValueAt(i, 4);

totalMoney += price \* quantity;

} else {

// 处理其他类型的数据

}

}

// 添加到数据库中

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=false", "root", "123456");

// 生成订单号

SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");

String orderNo = sdf.format(new Date()) + String.format("%04d", (int)(Math.random() \* 10000));

String sql = "INSERT INTO `order`(orderNo, deskId, createtime, money, customerId, status, number) VALUES(?, ?, ?, ?, ?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS);

stmt.setString(1, orderNo);

stmt.setInt(2, Integer.parseInt(table));

stmt.setTimestamp(3, new Timestamp(System.currentTimeMillis()));

stmt.setDouble(4, totalMoney); // 将总金额设置为double类型

stmt.setInt(5, customerId); // 将customerId添加到SQL语句中

stmt.setString(6, "未支付"); // 假设默认为未支付

stmt.setInt(7, num);

stmt.executeUpdate();

ResultSet rs = stmt.getGeneratedKeys();

rs.next();

int orderId = rs.getInt(1);

for (int i = 0; i < model.getRowCount(); i++) {

String code = (String)model.getValueAt(i, 0);

int quantity = (int)model.getValueAt(i, 4);

sql = "INSERT INTO orderitem(orderId, dishId, amount) VALUES(?, ?, ?)";

PreparedStatement stmt2 = conn.prepareStatement(sql);

stmt2.setInt(1, orderId);

stmt2.setInt(2, Integer.parseInt(code));

stmt2.setDouble(3, quantity);

stmt2.executeUpdate();

stmt2.close();

}

// 修改desk表中的状态

sql = "UPDATE desk SET status='使用中' WHERE no=?";

PreparedStatement stmt3 = conn.prepareStatement(sql);

stmt3.setString(1, table);

stmt3.executeUpdate();

stmt3.close();

// 关闭数据库连接

rs.close();

stmt.close();

conn.close();

JOptionPane.showMessageDialog(null, "点菜成功", "提示", JOptionPane.INFORMATION\_MESSAGE);

} catch (SQLException ex) {

ex.printStackTrace();

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(null, "就餐人数和菜品数量必须是整数", "提示", JOptionPane.WARNING\_MESSAGE);

return;

} catch (Exception ex) {

ex.printStackTrace();

}

}

});

// 加载客户和餐台列表

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=false", "root", "123456");

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM customer");

while (rs.next()) {

customerComboBox.addItem(rs.getString("name"));

}

rs.close();

stmt.close();

stmt = conn.createStatement();

rs = stmt.executeQuery("SELECT \* FROM desk WHERE status='空闲'");

while (rs.next()) {

tableComboBox.addItem(rs.getString("no"));

}

rs.close();

stmt.close();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

private int getCustomerId(String customerName) {

int customerId = -1;

try {

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cysjk?useSSL=false", "root", "123456");

String sql = "SELECT id FROM customer WHERE name=?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, customerName);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

customerId = rs.getInt("id");

}

rs.close();

stmt.close();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

return customerId;

}

public DefaultTableModel getOrderTableModel() {

Component comp = getContentPane().getComponent(1);

if (!(comp instanceof JScrollPane)) {

// 第二个组件不是滚动面板，返回一个空模型

return new DefaultTableModel();

}

JTable orderTable = (JTable) ((JScrollPane) comp).getViewport().getView();

return (DefaultTableModel) orderTable.getModel();

}

private void showDishDialog() {

DishDialog dialog = new DishDialog(this);

dialog.setVisible(true);

}

public static void main(String[] args) {

new OrderFrame().setVisible(true);

}

}

ChangeFrame.java

package zs;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class ChangeFrame extends JFrame implements ActionListener {

private JComboBox<String> deskNumberBox, customerNameBox;

private JTextField receivableAmountField, receivedAmountField, changeAmountField;

public ChangeFrame() {

setTitle("找零界面");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(6, 2));

JLabel deskNumberLabel = new JLabel("餐桌号：");

deskNumberBox = new JComboBox<>();

panel.add(deskNumberLabel);

panel.add(deskNumberBox);

JLabel customerNameLabel = new JLabel("客户名：");

customerNameBox = new JComboBox<>();

panel.add(customerNameLabel);

panel.add(customerNameBox);

JLabel receivableAmountLabel = new JLabel("应收金额：");

receivableAmountField = new JTextField(10);

receivableAmountField.setEditable(false);

panel.add(receivableAmountLabel);

panel.add(receivableAmountField);

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

receivedAmountField = new JTextField(10);

panel.add(receivedAmountLabel);

panel.add(receivedAmountField);

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

changeAmountField = new JTextField(10);

changeAmountField.setEditable(false);

panel.add(changeAmountLabel);

panel.add(changeAmountField);

JButton confirmButton = new JButton("确认");

confirmButton.addActionListener(this);

panel.add(new JLabel(""));

panel.add(confirmButton);

add(panel);

setVisible(true);

// 初始化餐桌号和客户名列表

initDeskNumbers();

initCustomerNames();

}

private void initDeskNumbers() {

Connection conn = null;

try {

Class.forName("com.mysql.cj.jdbc.Driver");

String url = "jdbc:mysql://localhost:3306/cysjk";

String user = "root";

String password = "123456";

conn = DriverManager.getConnection(url, user, password);

String query = "SELECT no FROM desk ORDER BY id";

PreparedStatement stmt = conn.prepareStatement(query);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

deskNumberBox.addItem(rs.getString("no"));

}

rs.close();

stmt.close();

} catch (ClassNotFoundException | SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "连接数据库失败！", "错误", JOptionPane.ERROR\_MESSAGE);

} finally {

if (conn != null) {

try {

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

}

private void initCustomerNames() {

Connection conn = null;

try {

Class.forName("com.mysql.cj.jdbc.Driver");

String url = "jdbc:mysql://localhost:3306/cysjk";

String user = "root";

String password = "123456";

conn = DriverManager.getConnection(url, user, password);

String query = "SELECT name FROM customer ORDER BY id";

PreparedStatement stmt = conn.prepareStatement(query);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

customerNameBox.addItem(rs.getString("name"));

}

rs.close();

stmt.close();

} catch (ClassNotFoundException | SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "连接数据库失败！", "错误", JOptionPane.ERROR\_MESSAGE);

} finally {

if (conn != null) {

try {

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals("确认")) {

String deskNumber = (String) deskNumberBox.getSelectedItem();

String customerName = (String) customerNameBox.getSelectedItem();

// 连接MySQL数据库

Connection conn = null;

try {

Class.forName("com.mysql.cj.jdbc.Driver");

String url = "jdbc:mysql://localhost:3306/cysjk";

String user = "root";

String password = "123456";

conn = DriverManager.getConnection(url, user, password);

// 查询desk表和customer表获取餐桌号和客户名对应的信息

int deskId = 0;

int customerId = 0;

String deskQuery = "SELECT id FROM desk WHERE no=?";

PreparedStatement deskStmt = conn.prepareStatement(deskQuery);

deskStmt.setString(1, deskNumber);

ResultSet deskRs = deskStmt.executeQuery();

if (deskRs.next()) {

deskId = deskRs.getInt("id");

} else {

JOptionPane.showMessageDialog(this, "该餐桌号不存在！", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

deskRs.close();

deskStmt.close();

String customerQuery = "SELECT id FROM customer WHERE name=?";

PreparedStatement customerStmt = conn.prepareStatement(customerQuery);

customerStmt.setString(1, customerName);

ResultSet customerRs = customerStmt.executeQuery();

if (customerRs.next()) {

customerId = customerRs.getInt("id");

} else {

JOptionPane.showMessageDialog(this, "该客户名不存在！", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

customerRs.close();

customerStmt.close();

// 查询order表获取应收金额

double receivableAmount = 0;

String orderQuery = "SELECT money FROM `order` WHERE deskId=? AND customerId=?";

PreparedStatement orderStmt = conn.prepareStatement(orderQuery);

orderStmt.setInt(1, deskId);

orderStmt.setInt(2, customerId);

ResultSet orderRs = orderStmt.executeQuery();

if (orderRs.next()) {

receivableAmount = orderRs.getDouble("money");

} else {

JOptionPane.showMessageDialog(this, "该订单不存在！", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

orderRs.close();

orderStmt.close();

double receivedAmount = 0;

try {

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

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "请输入正确的实收金额！", "错误", JOptionPane.ERROR\_MESSAGE);

return;

}

double changeAmount = receivedAmount - receivableAmount;

if (changeAmount < 0) {

JOptionPane.showMessageDialog(this, "实收金额不能小于应收金额！", "错误", JOptionPane.ERROR\_MESSAGE);

} else {

// 修改订单状态为已支付

String updateQuery = "UPDATE `order` SET status='已支付' WHERE deskId=? AND customerId=?";

PreparedStatement updateStmt = conn.prepareStatement(updateQuery);

updateStmt.setInt(1, deskId);

updateStmt.setInt(2, customerId);

updateStmt.executeUpdate();

updateStmt.close();

// 更新desk表中对应餐桌的status属性为"空闲"

String updateDeskQuery = "UPDATE desk SET status='空闲' WHERE id=?";

PreparedStatement updateDeskStmt = conn.prepareStatement(updateDeskQuery);

updateDeskStmt.setInt(1, deskId);

updateDeskStmt.executeUpdate();

updateDeskStmt.close();

receivableAmountField.setText(String.format("%.2f", receivableAmount));

changeAmountField.setText(String.format("%.2f", changeAmount));

JOptionPane.showMessageDialog(this, "支付成功！", "提示", JOptionPane.INFORMATION\_MESSAGE);

}

} catch (ClassNotFoundException | SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "连接数据库失败！", "错误", JOptionPane.ERROR\_MESSAGE);

} finally {

if (conn != null) {

try {

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

}

}

public static void main(String[] args) {

new ChangeFrame();

}

}