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班）** |
| **姓名** | **胡裕成** |
| **学号** | **B20210304101** |
| **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 | 胡裕成 | 学号 | B20210304101 | 班级 | 21软件01班 |
| 专业 | 软件工程 | | 指导教师姓名 | 杨刚、潘怡 | |
| 项目名称 | 基于C/S架构的餐饮管理系统 | | | | |
| **实训过程**：（说明实训各阶段的工作过程，包括需求分析、系统设计、编码实现、系统测试、交付实施等阶段的主要工作。）  需求分析:  餐饮管理系统包括以下功能：  员工管理：包括员工信息的增删改查等基本操作。  客户管理：包括客户信息的增删改查等基本操作。  菜品管理：包括菜品信息的增删改查等基本操作。  餐台管理：包括餐台信息的增删改查等基本操作。  点菜管理：对某客户、某一空闲餐台实行开台，同时实现点菜，保存客户点菜的信息,并记录开台时间。  结账管理：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，完成结账的操作，并记录成统计数据。  系统设计:  系统功能包括员工管理模块、客户管理模块、餐台管理模块、菜品管理模块、客户点菜模块、以及结账管理模块  编码实现:  利用JFrame界面实现人机交互；利用JDBC所提供的方法实现对数据库进行存取数据。  系统测试:  测试系统的完备性、实用性和正确性。  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,能够对员工，客户，菜品，餐台进行增删改查等管理，同时实现了点菜和结账，以及销售统计等功能 | | | | | |
| **实训总结**：（包括心得体会、存在的问题和改进方向。）  在本次的JavaC/S架构餐饮管理系统的实训中，我学习和掌握了许多开发技术和方法，同时收获了实际开发经验。在本次实训中，我认识到要在开发前充分考虑应用的需求和细节，对系统的功能模块、用户界面等进行详细的说明和设计，开发过程中要注意代码质量和代码风格的规范，提高可读性和可维护性。同时，我意识到，理论知识不足以解决实际问题，实践经验是实现应用程序的关键因素。通过本次实训，我对软件开发的流程有了更深入的理解，也提高了软件开发的实际能力。 | | | | | |
|  | | | | | |

基于C/S架构的餐饮管理系统

设计说明书

作者：胡裕成(B20210304101)

计算机科学与工程学院

2023年 06 月 15日

**摘要**

本餐饮管理系统是一个基于Java C/S架构开发的软件，旨在提供一个高效、稳定、易用的餐饮管理解决方案。该系统主要包含以下模块：

1. 管理模块：管理员登录后可以进行员工管理、客户管理、菜品管理、餐台管理等操作，同时也提供了销售统计功能，方便管理员进行决策和分析。
2. 点菜模块：顾客通过开台后点菜，可以选择菜品、数量等。
3. 结账模块：，结账时系统会自动计算费用并在支付费用后找零。

本系统实现了员工管理，客户管理，菜品管理，餐台管理，点菜管理，结账管理等功能，并且在每个模块中都通过精心设计的界面和功能实现了易用性和可扩展性。管理模块可以对员工、客户、菜品、餐台等信息进行管理，同时也提供了销售统计功能，可以帮助管理员更好地把握餐厅的运营状况。在点菜模块，系统提供了多种菜品的搜索方式，方便顾客查找和选择。在结账模块中，系统会自动计算费用并在支付费用后找零，方便顾客的使用。总而言之，该系统优化了前台界面和后台管理功能，提高了餐饮服务质量和管理效率。在今后的餐饮业中，类似这样的餐饮管理系统将会变得越来越普遍。

**关键词：**java ，C/S架构，餐饮管理系统

目录

[1 引言 1](#_Toc11882504)

[1.1 编写目的 1](#_Toc11882505)

[1.2 参考资料 1](#_Toc11882506)

[2 需求规约 2](#_Toc11882507)

[2.1 功能需求 2](#_Toc11882508)

[2.2 界面需求 3](#_Toc11882509)

[2.3 数据需求 3](#_Toc11882510)

[3 系统设计 4](#_Toc11882511)

[3.1 运行环境 4](#_Toc11882512)

[3.2 系统静态结构设计 4](#_Toc11882513)

[3.3 人机接口设计 5](#_Toc11882514)

[3.4 XXX模块设计说明 7](#_Toc11882515)

[3.5 XXX模块设计说明 7](#_Toc11882516)

[4 数据库设计 8](#_Toc11882517)

[4.1 数据库环境说明 8](#_Toc11882518)

[4.2 数据库的命名规则 8](#_Toc11882519)

[4.3 逻辑结构设计 8](#_Toc11882520)

[4.4 物理结构设计 9](#_Toc11882521)

[4.5 安全性设计 10](#_Toc11882522)

[5 测试用例设计 12](#_Toc11882523)

[附录 14](#_Toc11882524)

[附录1 程序运行结果 14](#_Toc11882525)

[附录2 程序源代码 15](#_Toc11882526)

# 1 引言

## 1.1 编写目的

本文档是“餐饮管理系统”的软件设计说明书。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] 文东. 数据库系统开发[M]. 北京: 北京科海电子出版社，2011．  
[2] 戴远泉，程宁，胡文杰. Java高级程序设计[M]. 北京：人民邮电出版社，2022.

# 2 需求规约

## 2.1 功能需求

餐饮管理系统包括以下功能：

员工管理：包括员工信息的增删改查等基本操作。

客户管理：包括客户信息的增删改查等基本操作。

菜品管理：包括菜品信息的增删改查等基本操作。

餐台管理：包括餐台信息的增删改查等基本操作。

点菜管理：对某客户、某一空闲餐台实行开台，同时实现点菜，保存客户点菜的信息,并记录开台时间。

结账管理：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，完成结账的操作，并记录成统计数据。

图2.1 餐饮管理系统功能模块图

2.1.1 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。

扩展点：

1. 用户注册，执行用户注册用例。

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 用户注册用例说明

用例名称：用户注册。

用例描述：餐厅管理员注册账号。

前置条件：用户输入账号、密码。

后置条件：用户注册成功。

活动步骤：

1. 管理员输入账号和密码。
2. 点击注册按钮，管理员注册成功。

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. 餐厅管理员点击新增员工按钮，弹出界面。
2. 餐厅管理员在新的界面里填写员工信息。
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. 餐厅管理员点击表格信息，弹出修改的界面。
2. 餐厅管理员在新的界面内的输入框中填写要修改的信息。
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. 餐厅管理员点击表格中该员工的id。
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. 餐厅管理员点击新增客户按钮，弹出界面。
2. 餐厅管理员在新的界面里填写客户信息。
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. 餐厅管理员点击表格信息，弹出修改的界面。
2. 餐厅管理员在新的界面内的输入框中填写要修改的信息。
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. 餐厅管理员点击表格中该客户的id。
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. 餐厅管理员点击新增菜品按钮，弹出界面。
2. 餐厅管理员在新的界面里填写菜品信息。
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. 餐厅管理员点击表格信息，弹出修改的界面。
2. 餐厅管理员在新的界面内的输入框中填写要修改的信息。
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. 餐厅管理员点击表格中该菜品的id。
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 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. 餐厅管理员点击新增餐台按钮，弹出界面。
2. 餐厅管理员在新的界面里填写餐台的座位数。
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. 餐厅管理员点击表格信息，弹出修改的界面。
2. 餐厅管理员在新的界面内的输入框中填写要修改的信息。
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. 餐厅管理员点击表格中该餐台的餐台号。
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。

2.2.7 点菜管理用例

用例名称：点菜管理。

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id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id以及人数，对客户进行开台操作。
2. 系统将根据人数为该用户分配合适的餐台。
3. 餐厅管理员根据菜品信息来查询菜品，提供给客户去选菜。
4. 选菜完成后点击加入购物车后再点击下单，菜品添加到购物车中。
5. 确认无误后点击保存，生成新的订单。

异常处理：

1. 餐厅管理员输入错误的客户id或着非法的人数时，系统会提示具体的输入错误，并且不会对客户进行开台操作。
2. 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。

2.2.8 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，删除，结账的操作。

前置条件：餐厅管理员输入订单号或客户id，对订单进行模糊查询。

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. 餐厅管理员输入订单号或客户id，对订单进行查询。
2. 系统将结果显示在表格中。
3. 餐厅管理员选定某一订单，弹出新的界面。
4. 在新的界面中显示订单的应付金额与折后价格以及实付金额。
5. 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

友好，操作简单；界面美观，用户体验感良好。

## 2.3 数据需求

表2.1 管理员实体

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 |
| 用户名 | 字符串 | 30 | 否 | 是 |
| 密码 | 字符串 | 30 | 否 | 否 |

表2.1 员工实体

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 |
| 员工号 | 整型 |  | 否 | 是 |
| 姓名 | 字符串 | 30 | 否 | 否 |
| 性别 | 字符 | 1 | 是 | 否 |
| 身份证号 | 字符串 | 18 | 是 | 否 |
| 出生日期 | 日期 |  | 是 | 否 |
| 住址 | 字符串 | 40 | 是 | 否 |
| 电话 | 字符串 | 11 | 是 | 否 |
| 职位 | 字符串 | 4 | 是 | 否 |
| 是否在职 | 字符 | 1 | 否 | 否 |

表2.1 客户实体

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 |
| 客户号 | 整型 |  | 否 | 是 |
| 姓名 | 字符串 | 30 | 否 | 否 |
| 性别 | 字符 | 1 | 是 | 否 |
| 电话 | 字符串 | 11 | 是 | 否 |
| Vip | 字符 | 1 | 否 | 否 |

表2.1 菜品实体

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 |
| 菜品号 | 整型 |  | 否 | 是 |
| 菜品名 | 字符串 | 30 | 否 | 否 |
| 价格 | 整型 |  | 否 | 否 |
| 口味 | 字符串 | 4 | 是 | 否 |
| 状态 | 字符 | 1 | 否 | 否 |

表2.1 餐台实体

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 类型 | 长度 | 是否为空 | 唯一标识 |
| 餐台号 | 整型 |  | 否 | 是 |
| 座位数 | 整型 |  | 否 | 否 |
| 状态 | 字符 | 1 | 否 | 否 |

# 3 系统设计

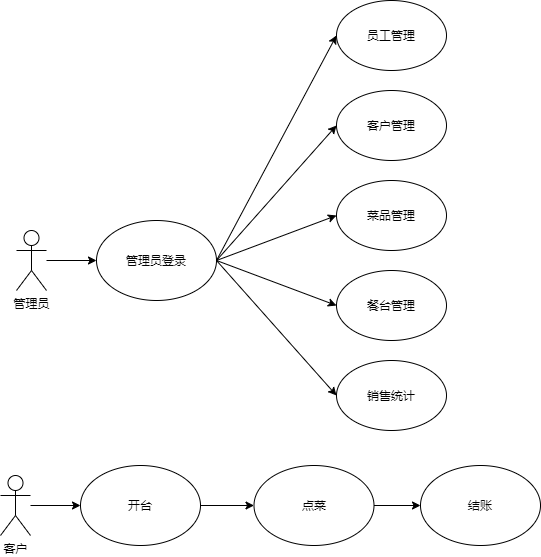
## 3.1 运行环境

操作系统: windows 10

软件环境: idea

## 3.2 人机接口设计

3.2.1 设计应用类

图3.3 餐饮管理系统的设计类图

3.2.2 用户接口1（登录界面）设计

图3.2 登录界面布局设计

主要控件：

账号输入框：用于输入所注册或登录的账号，最大长度为20个字符

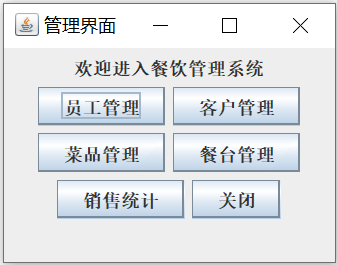
密码输入框：用于输入所注册或登录的密码，最大长度为20个字符

注册按钮：输入账号和密码后点击注册按钮，如果注册的账号为新账号，则注册成功。

登录按钮：输入账号和密码后点击登录按钮，如果账号密码均正确，则登录成功。

关闭按钮：点击登录按钮，将该页面关闭

3.2.3 用户接口2（管理界面）设计

图3.3 员工管理界面布局设计

主要控件：

员工管理按钮：点击进入员工管理界面

客户管理按钮：点击进入客户管理界面

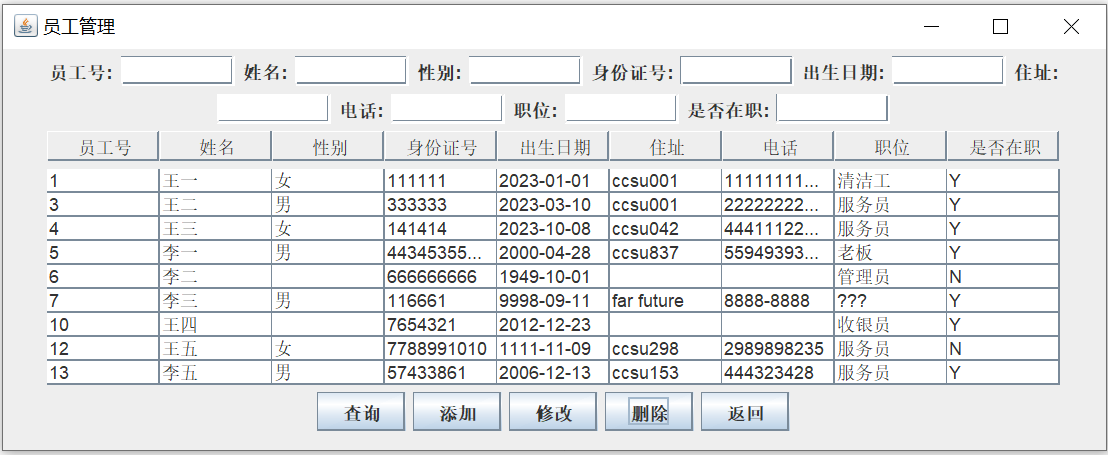
菜品管理按钮：点击进入菜品管理界面

餐台管理按钮：点击进入餐台管理界面

销售统计按钮：点击进入销售统计管理界面

关闭按钮：点击将该页面关闭

3.2.4 用户接口3（员工管理界面）设计

图3.4 员工管理界面布局设计

主要控件：

员工号输入框：用于输入员工号

姓名输入框：用于输入员工的姓名

性别输入框：用于输入员工的性别

身份证号输入框：用于输入员工的身份证号

出生日期输入框：用于输入员工的出生日期

住址输入框：用于输入员工的住址

电话输入框：用于输入员工的电话

职位输入框：用于输入员工的职位

是否在职输入框：用于输入员工的出生日期

查询按钮：点击根据向输入框输入的信息进行查询

添加按钮：点击保存输入的员工信息

修改按钮：点击将指定员工号的员工的信息修改成所输入的内容。

删除按钮：点击将指定员工号的员工的信息删除

返回按钮：点击退回到管理界面

3.2.5 用户接口4（客户管理界面）设计

图3.5 客户管理界面布局设计

主要控件：

客户号输入框：用于输入客户号

姓名输入框：用于输入客户的姓名

性别输入框：用于输入客户的性别

电话输入框：用于输入客户的电话

vip输入框：用于输入客户是否为vip

查询按钮：点击根据向输入框输入的信息进行查询

添加按钮：点击保存输入的客户信息

修改按钮：点击将指定客户号的客户的信息修改成所输入的内容。

删除按钮：点击将指定客户号的客户的信息删除

返回按钮：点击退回到管理界面

3.2.6 用户接口5（菜品管理界面）设计

图3.6 菜品管理界面布局设计

主要控件：

菜品号输入框：用于输入菜品号

菜名输入框：用于输入菜名

价格输入框：用于输入菜品的价格

口味输入框：用于输入菜品的口味

状态输入框：用于输入菜品的状态

查询按钮：点击根据向输入框输入的信息进行查询

添加按钮：点击保存输入的菜品信息

修改按钮：点击将指定菜品号的菜品的信息修改成所输入的内容。

删除按钮：点击将指定菜品号的菜品的信息删除

返回按钮：点击退回到管理界面

3.2.7 用户接口6（餐台管理界面）设计

图3.7 餐台管理界面布局设计

主要控件：

餐台号输入框：用于输入餐台号

座位数输入框：用于输入餐台的座位数

状态输入框：用于输入餐台的状态

查询按钮：点击根据向输入框输入的信息进行查询

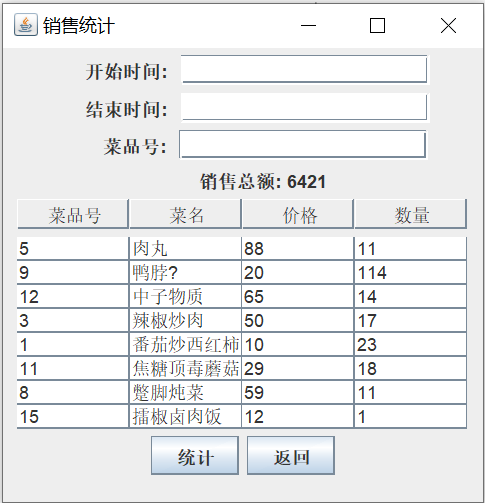
添加按钮：点击保存输入的餐台信息

修改按钮：点击将指定餐台号的餐台的信息修改成所输入的内容。

删除按钮：点击将指定餐台号的餐台的信息删除

返回按钮：点击退回到管理界面

3.2.8 用户接口7（销售统计界面）设计

图3.8 销售统计界面布局设计

主要控件：

开始时间输入框：用于输入需统计的开始时间

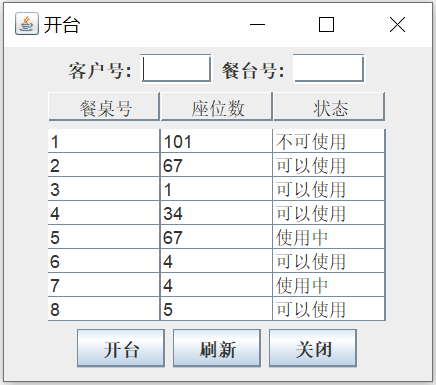
结束时间输入框：用于输入需统计的结束时间

菜品号输入框：用于输入需统计的菜品号

统计按钮：点击统计指定时间内指定菜品的销售情况

返回按钮：点击退回到管理界面

3.2.9 用户接口8（开台界面）设计

图3.9 开台界面布局设计

主要控件：

客户号输入框：用于输入开台的客户号

餐台号输入框：用于输入开台的餐台号

开台按钮：点击输入的客户号所对应的客户对指定的餐台进行开台，将开台信息保存，并跳转到点菜界面

刷新按钮：点击刷新餐台的信息显示

关闭按钮：点击将该页面关闭

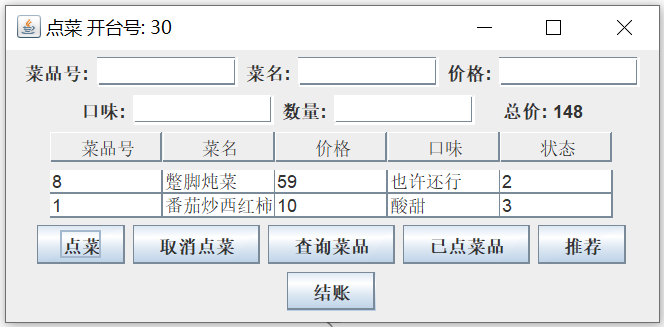
3.2.10 用户接口9（点菜界面）设计

图3.10 点菜界面布局设计

主要控件：

菜品号输入框：用于输入想要点的菜品号

菜名输入框：用于输入菜名

价格输入框：用于输入菜品的价格

口味输入框：用于输入菜品的口味

数量输入框：用于输入点菜的数量

点菜按钮：点击进行点菜，并将点菜信息保存

取消点菜按钮：点击取消指定的所点菜品，并修改或删除点菜信息

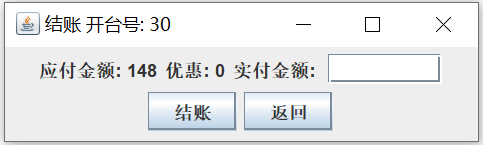
查询菜品按钮：点击根据向输入框输入的信息进行查询

已点菜品按钮：点击查看该客户已经点过的菜品

推荐按钮：点击根据客户的爱好口味推荐菜品。

结账按钮：点击跳转到结账界面进行结账

3.3.11 用户接口10（结账界面）设计

图3.11 员工管理界面布局设计

主要控件：

实付金额输入框：用于实付金额

结账按钮：点击进行结账，如果钱足够则进行找零并关闭该界面，否则提醒钱不够

返回按钮：点击返回点菜界面

# 4 数据库设计

## 4.1 数据库环境说明

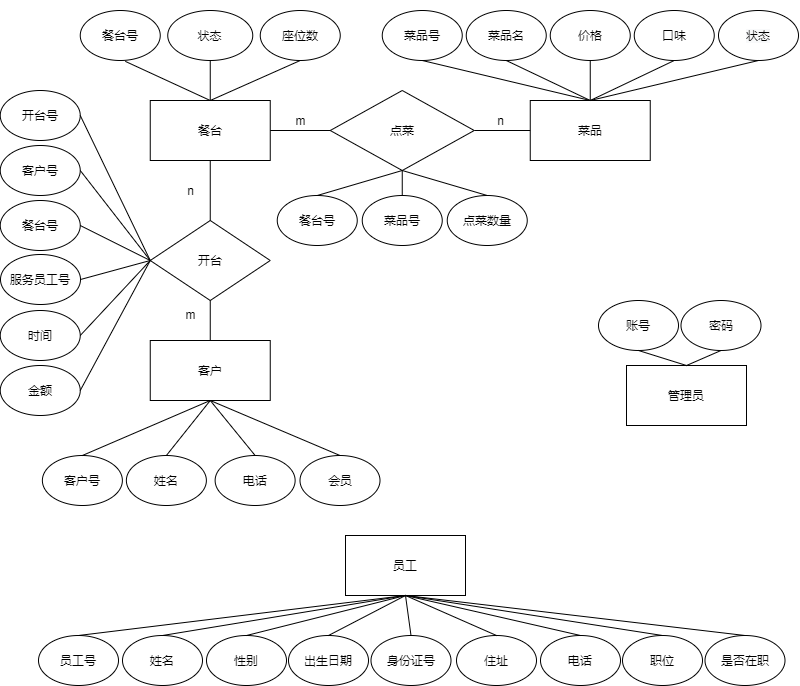
数据库系统: Mysql 8.0

编程工具: idea

## 4.2 数据库的命名规则

数据库以餐饮管理系统的英文命名，数据库中的表则以每个实体或联系的英文命名，以便开发时快速找到对应的表

## 4.3 逻辑结构设计

E-R图

关系模型

管理员表(账号, 密码)

员工表(员工号, 姓名, 性别, 出生日期, 身份证号, 住址, 电话, 职位, 是否在职)

客户表(客户号, 姓名, 性别, 电话, 会员)

菜品表(菜品号, 菜品名, 价格, 口味, 状态)

餐台表(餐台号，座位数，状态)

开台表(开台号，客户号，餐台号，服务员工号，开台时间，实收金额)

点菜号(开台号，菜品号，点菜数量)

范式分析

3FN

## 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 | 管理员表，记录管理员账号和密码 |
| staff | 员工表，记录员工的员工号、姓名等基本信息 |
| customer | 客户表，记录客户的客户号、姓名等基本信息 |
| dish | 菜品表，记录菜品的菜品号、菜名、价格等基本信息 |
| desk | 餐台表，记录餐台的餐台号、座位数等基本信息 |
| order\_desk | 开台号，存放客户开台的开台记录，包括开台时间、实付金额等信息 |
| order\_dish | 点菜表，存放开台后的点菜记录，包括开台号、所点菜品号、点菜数量等信息 |

表4.2 管理员表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | User | | | | | |
| 数据库用户 | | root | | | | | |
| 主 键 | | Username | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 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 | Varchar(20) | N | Y |  |  | 账号 |
| 2 | password | Varchar(20) | N | N |  |  | 加密后的密码 |
| sql脚本 | |  | | | | | |
| 备注 | |  | | | | | |

表4.2 员工表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | staff | | | | | |
| 数据库用户 | | root | | | | | |
| 主 键 | | Id | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 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 |  |  | 自动增加/员工号 |
| 2 | name | Varchar(20) | N | N |  |  | 姓名 |
| 3 | sex | enum('男', '女') | Y | N |  |  | 性别 |
| 4 | Identity\_id | Varchar(18) | Y | N |  |  | 身份证 |
| 5 | birthday | date | Y | N |  |  | 出生日期 |
| 6 | address | Varchar(40) | Y | N |  |  | 住址 |
| 7 | phone | Varchar(11) | Y | N |  |  | 电话 |
| 8 | position | Varchar(4) | Y | N |  |  | 职位 |
| 9 | freeze | enum('Y', 'N') | N | N |  | ‘Y’ | 是否在职 |
| sql脚本 | | create table staff (  id int null auto\_increment,  name varchar(20) not null,  sex enum('男', '女') null,  identity\_id varchar(18) null,  birthday date null,  address varchar(40) null,  phone varchar(11) null,  position varchar(4) null,  freeze enum('Y', 'N') default 'Y' not null,  constraint primary key (id) ); | | | | | |
| 备注 | |  | | | | | |

表4.3 客户表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | customer | | | | | |
| 数据库用户 | | root | | | | | |
| 主 键 | | Id | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 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 |  |  | 自动增加/客户号 |
| 2 | name | Varchar(20) | N | N |  |  | 姓名 |
| 3 | sex | enum('男', '女') | Y | N |  |  | 性别 |
| 4 | phone | Varchar(11) | Y | N |  |  | 电话 |
| 5 | vip | enum('Y', 'N') | N | N |  | ‘Y’ | 会员 |
| sql脚本 | | create table customer (  id int null auto\_increment,  name varchar(20) not null,  sex enum('男', '女') null,  phone varchar(11) null,  vip enum('Y', 'N') default 'N' not null,  constraint primary key (id) ); | | | | | |
| 备注 | |  | | | | | |

表4.4 菜品表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | dish | | | | | |
| 数据库用户 | | root | | | | | |
| 主 键 | | Id | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 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 |  |  | 自动增加/菜品号 |
| 2 | name | Varchar(20) | N | N |  |  | 菜品名 |
| 3 | price | int | Y | N |  |  | 价格 |
| 4 | taste | Varchar(4) | Y | N |  |  | 口味 |
| 5 | state | enum('Y', 'N') | N | N |  | ‘Y’ | 状态 |
| sql脚本 | | create table dish (  id int null auto\_increment,  name varchar(20) not null,  price int not null,  taste varchar(4) null,  state enum('Y', 'N') default 'Y' null,  constraint primary key (id) ); | | | | | |
| 备注 | |  | | | | | |

表4.5 餐台表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | desk | | | | | |
| 数据库用户 | | root | | | | | |
| 主 键 | | Id | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 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 |  |  | 自动增加/餐台号 |
| 2 | seating | int | Y | N |  |  | 位置数 |
| 3 | state | enum('Y', 'N') | N | N |  | ‘Y’ | 状态 |
| sql脚本 | | create table desk (  id int auto\_increment primary key,  seating int null,  state enum ('可以使用', '使用中', '不可使用') default '可以使用' null ); | | | | | |
| 备注 | |  | | | | | |

表4.6 开台表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | order\_desk | | | | | |
| 数据库用户 | | root | | | | | |
| 主 键 | | order\_id | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 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 | order\_id | int | N | Y |  |  | 自动增加/开台号 |
| 2 | customer\_id | int | N | N |  |  | 客户号 |
| 3 | desk\_id | int | N | N |  |  | 餐台号 |
| 4 | waiter\_id | int | N | N |  |  | 服务员工号 |
| 5 | time | datetime | N | N |  |  | 开台时间 |
| 6 | paid\_in\_amount | int | N | N |  | 0 | 实收金额 |
| sql脚本 | | create table order\_desk (  order\_id int auto\_increment primary key,  customer\_id int not null,  desk\_id int null,  waiter\_id int not null,  time datetime null,  paid\_in\_amount int default 0 null ); | | | | | |
| 备注 | |  | | | | | |

表4.7 点菜表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表 名 | | order\_dish | | | | | |
| 数据库用户 | | root | | | | | |
| 主 键 | | order\_id, dish\_id | | | | | |
| 其他排序字段 | |  | | | | | |
| 索引字段 | |  | | | | | |
| 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 | order\_id | int | N | N |  |  | 开台号 |
| 3 | dish\_id | int | N | N |  |  | 所点菜品号 |
| 5 | count | int | Y | N |  | 1 | 所点菜品数量 |
| sql脚本 | | create table order\_dish (  order\_id int not null,  dish\_id int not null,  count int default 1 null ); | | | | | |
| 备注 | |  | | | | | |

## 4.5 安全性设计

管理数据库内容时，必须管理员登录后才可以进行管理，否则只能进行点菜

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

管理员登录后，该用户认定为是管理员，否则认定为是顾客

（2）存取控制

将数据存入数据库前，必须先检查数据的正确性，检查通过后，才能存入数据库

（3）数据加密

注册登录使用的密码将会进行加密处理，防止密码在数据库中明文出现

4.5.2 用户帐号密码的加密方法

账号不做加密处理；

密码加密方法为：

每两位真实密码中插入一位随机字符，再将真实密码的每一位加上当前的位置，剩余的位数用随机字符补齐，将真实密码的长度加上加密后的密码的第3位得到加密后的密码的最后一位

# 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 | 1 | 输入‘账号’和‘密码’ | admin，123456 | 登录成功 | 通过 |
| 2 | admin，1234567 | 登录失败 | 通过 |
| 3 | admin1，123456 | 登录失败 | 通过 |
| 4 | admin,空 | 登录失败 | 通过 |
| 5 | 空,123456 | 登录失败 | 通过 |
| 6 | admin,一个空格 | 登录失败 | 通过 |
| 7 | 一个空格，123456 | 登录失败 | 通过 |
| 8 | admin，密码长度超过32 | 登录失败 | 通过 |
| 9 | 用户名超过36，123456 | 登录失败 | 通过 |
| 注册 | 10 | 输入账号，密码 | User1,123456,不存在该账号 | 创建成功 | 通过 |
| 11 | User1,123456,存在该账号 | 创建失败 | 通过 |
| 12 | User1, 空 | 创建失败 | 通过 |
| 13 | 空,123456 | 创建失败 | 通过 |

表5.2 员工管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 添加员工信息 | 1 | 输入姓名, 性别, 出生日期, 身份证号, 住址, 电话, 职位, 是否在职 | 张三，男，1999-01-01，111，长沙学院，12345，管理员，Y | 添加成功 | 通过 |
| 2 | 张三，空，空，空，空，空，空,空 | 添加成功 | 通过 |
| 3 | 空，男，1999-01-01，111，长沙学院，12345，管理员，Y | 添加失败 | 通过 |
| 4 | 空，空，空，空，空，空，空,空 | 添加失败 | 通过 |
| 删除员工信息 | 5 | 输入员工号 | 1，存在该员工 | 删除员工号为1的员工信息 | 通过 |
| 6 | 1，不存在该员工 | 删除失败 | 通过 |
| 7 | 空 | 删除失败 | 通过 |
| 修改员工信息 | 8 | 输入员工号，姓名, 性别, 出生日期, 身份证号, 住址, 电话, 职位, 是否在职 | 1，张三，男，1999-01-01，111，长沙学院，12345，管理员，Y | 将员工号为1的员工的信息修改为所输入的信息 | 通过 |
| 9 | 1，张三，空，空，222，空，空，空,空 | 将员工号为1的员工的身份证号信息修改为222 | 通过 |
| 10 | 空，张三，男，1999-01-01，111，长沙学院，12345，管理员，Y | 修改失败 | 通过 |
| 查询员工信息 | 11 | 输入员工号，姓名, 性别, 出生日期, 身份证号, 住址, 电话, 职位, 是否在职 | 1，空，空，空，空，空，空，空,空 | 查询员工号为1的员工 | 通过 |
| 12 | 1，张三，男，1999-01-01，111，长沙学院，12345，管理员，Y | 查询员工号为1的员工 | 通过 |
| 13 | 空，空，男，空，空，空，空，空,空 | 查询所有性别为男的员工 | 通过 |
| 14 | 2，空，空，空，空，空，空，空,空，不存在员工号为2的员工 | 查询失败 | 通过 |
| 15 | 空，空，空，空，空，空，空，服务员,空，不存在服务员 | 查询失败 | 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 |
| 添加客户信息 | 1 | 输入姓名, 性别, 电话, 是否为vip | 李四，男，12345，Y | 添加成功 | 通过 |
| 2 | 李四，空，空，空 | 添加成功 | 通过 |
| 3 | 空，男，12345，Y | 添加失败 | 通过 |
| 4 | 空，空，空，空 | 添加失败 | 通过 |
| 删除客户信息 | 5 | 输入客户号 | 1，存在该客户 | 删除客户号为1的客户信息 | 通过 |
| 6 | 1，不存在该客户 | 删除失败 | 通过 |
| 7 | 空 | 删除失败 | 通过 |
| 修改客户信息 | 8 | 输入客户号，姓名, 性别, 电话, 是否为vip | 1，李四，男，12345，Y | 将客户号为1的客户的信息修改为所输入的信息 | 通过 |
| 9 | 1，李四，空，空，N | 将客户号为1的客户的vip信息修改为N | 通过 |
| 10 | 空，李四，男，12345，Y | 修改失败 | 通过 |
| 查询客户信息 | 11 | 输入客户号，姓名, 性别, 电话, 是否为vip | 1，空，空，空，空 | 查询客户号为1的客户 | 通过 |
| 12 | 1，李四，男，12345，Y | 查询客户号为1的客户 | 通过 |
| 13 | 空，空，空，空，Y | 查询所有vip客户 | 通过 |
| 14 | 2，空，空，空，空，不存在客户号为2的客户 | 查询失败 | 通过 |
| 15 | 空，空，空，11111,空，不存在电话为11111的客户 | 查询失败 | 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 |
| 添加菜品信息 | 1 | 输入菜名, 价格, 口味, 状态 | 番茄炒蛋，5，甜，Y | 添加成功 | 通过 |
| 2 | 番茄炒蛋，空，空，空 | 添加成功 | 通过 |
| 3 | 空，5，甜，Y | 添加失败 | 通过 |
| 4 | 空，空，空，空 | 添加失败 | 通过 |
| 删除菜品信息 | 5 | 输入菜品号 | 3，存在菜品号为3的菜品 | 删除菜品号为3的菜品信息 | 通过 |
| 6 | 3，不存在菜品号为3的菜品 | 删除失败 | 通过 |
| 7 | 空 | 删除失败 | 通过 |
| 修改菜品信息 | 8 | 输入菜品号，菜名, 价格, 口味, 状态 | 3，番茄炒蛋，5，甜，Y | 将菜品号为1的菜品的信息修改为所输入的信息 | 通过 |
| 9 | 3，辣椒炒肉，空，空，空 | 将菜品号为3的菜品的菜名改为‘辣椒炒肉’ | 通过 |
| 10 | 空，番茄炒蛋，5，甜，Y | 修改失败 | 通过 |
| 查询菜品信息 | 11 | 输入菜品号，菜名, 价格, 口味, 状态 | 3，空，空，空，空 | 查询菜品号为3的菜品 | 通过 |
| 12 | 3，番茄炒蛋，5，甜，Y | 查询菜品号为3的菜品 | 通过 |
| 13 | 空，空，空，辣，空 | 查询所有口味为辣的菜品 | 通过 |
| 14 | 3，空，空，空，空，不存在菜品号为3的菜品 | 查询失败 | 通过 |
| 15 | 空，土豆丝，空，空,空，不存在菜名为土豆丝的菜品 | 查询失败 | 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 |
| 添加餐台信息 | 1 | 输入座位数, 状态 | 8, 可以使用 | 添加成功 | 通过 |
| 2 | 8, 空 | 添加成功 | 通过 |
| 3 | 空，可以使用 | 添加失败 | 通过 |
| 4 | 空，空 | 添加失败 | 通过 |
| 删除餐台信息 | 5 | 输入餐台号 | 4，存在4号餐台 | 删除4号餐台的信息 | 通过 |
| 6 | 4，不存在4号餐台 | 删除失败 | 通过 |
| 7 | 空 | 删除失败 | 通过 |
| 修改餐台信息 | 8 | 输入餐台号，座位数, 状态 | 4,8, 可以使用 | 将4号餐台的信息修改为所输入的信息 | 通过 |
| 9 | 4,9,空 | 将4号餐台的座位数修改为9 | 通过 |
| 10 | 空,8, 可以使用 | 修改失败 | 通过 |
| 查询餐台信息 | 11 | 输入餐台号，座位数, 状态 | 4，空，空 | 查询4号餐台 | 通过 |
| 12 | 4，8，可以使用 | 查询4号餐台 | 通过 |
| 13 | 空，空, 可以使用 | 查询所有可以使用的餐台 | 通过 |
| 14 | 4，空，空，不存在4号餐台 | 查询失败 | 通过 |
| 15 | 空，16,空，不存在16个座位的餐台 | 查询失败 | 通过 |

表5.5 点菜管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | 用例编号 | 操作步骤  （输入条件） | 测试数据 | 预期结果 | 实际结果/判定 |
| 开台 | 1 | 输入客户号, 餐台号 | 6, 7 | 添加成功 | 通过 |
| 2 | 6, 空 | 添加失败 | 通过 |
| 3 | 空，7 | 添加失败 | 通过 |
| 4 | 空，空 | 添加失败 | 通过 |
| 5 | 6, 7，不存在6号客户 | 添加失败 | 通过 |
| 6 | 6, 7，不存在7号餐台 | 添加失败 | 通过 |
| 7 | 6, 7，7号餐台不可使用 | 添加失败 | 通过 |
| 点菜 | 8 | 输入菜品号，数量 | 8，2，存在8号菜品 | 点8号菜品两份 | 通过 |
| 9 | 8，空，存在8号菜品 | 点8号菜品一份 | 通过 |
| 10 | 空，2 | 点菜失败 | 通过 |
| 11 | 8，空，不存在8号菜品 | 点菜失败 | 通过 |
| 取消点菜 | 12 | 输入菜品号，数量 | 8，2，已点8号菜品 | 取消点过的8号菜品两份 | 通过 |
| 13 | 8，空，已点8号菜品 | 取消点过的所有8号菜品 | 通过 |
| 14 | 空，2 | 取消点菜失败 | 通过 |
| 15 | 8，空，未点8号菜品 | 取消点菜失败 | 通过 |

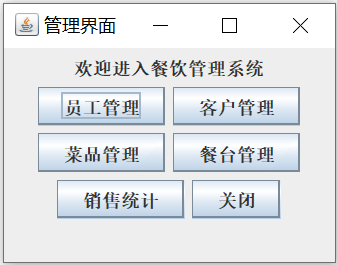
表5.5 结账管理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |
| 结账 | 1 | 输入实收金额 | 101，需付100 | 找零1元 | 通过 |
| 2 | 99，需付100 | 结账失败 | 通过 |
| 3 | 空，需付100 | 结账失败 | 通过 |

# 附录

## 附录1 程序运行结果

图1 登录界面

图2 管理界面

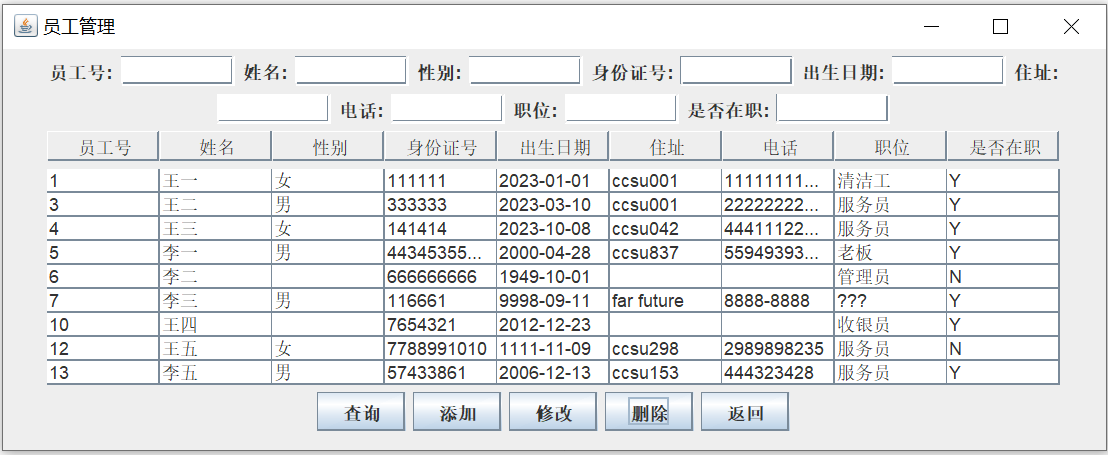
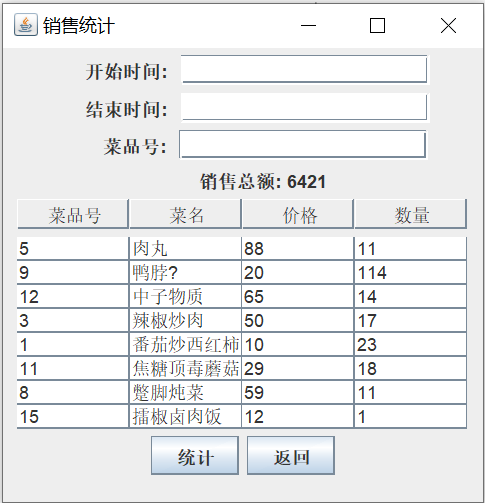
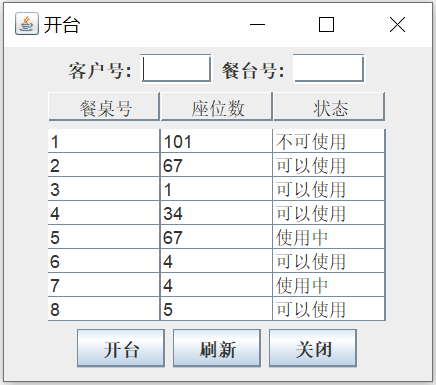
图3 员工管理界面

图4 客户管理界面

图5 菜品管理界面

图6 餐台管理界面

图7 销售统计界面

图8 开台界面

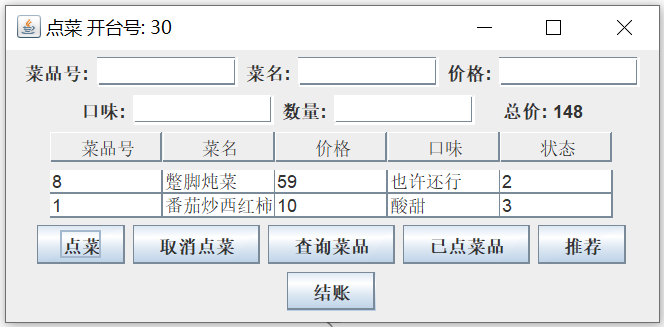
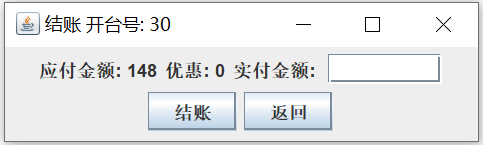
图9 点菜界面

图10 结账界面

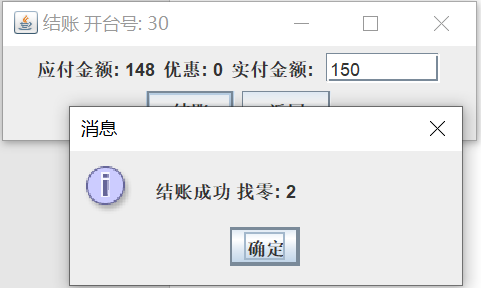


图11 结账后找零

## 附录2 程序源代码

1.管理员注册登录

//查询该管理员账号的密码  
public String getUserPassword(String username)  
{  
 String password = null;  
 String sql = "select \* from user where username = ?;";  
  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setString(1, username);  
 ResultSet rs = preStmt.executeQuery();  
  
 //获取密码  
 if (rs.next())  
 { password = rs.getString("password"); }  
  
 //关闭查询  
 preStmt.close();  
 rs.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
  
 //返回查询结果  
 return *declassify*(password);  
}  
  
//注册管理员信息  
public boolean logonUser(String username, String password)  
{  
 String sql = "insert into user(username, password) value (?,?);";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setString(1, username);  
 preStmt.setString(2, *encrypt*(password));  
 preStmt.execute();  
 preStmt.close();  
 }  
 catch (SQLException e)  
 { return false; }  
 return true;  
}

//将密码加密  
static String encrypt(String password)  
{  
 char[] passwordChars = password.toCharArray();  
 char[] newChars = new char[29];  
 int index = 0;  
 for (int i = 0; i < 29; i++)  
 {  
 if (index == password.length() || (i + 1) % 3 == 0) {  
 newChars[i] = '!';  
 newChars[i] += new Random().nextInt(94);  
 } else {  
 passwordChars[index] += (i + 1);  
 newChars[i] = passwordChars[index++];  
 }  
 }  
 newChars[2] = '!';  
 char passwordLength = newChars[2] += new Random().nextInt(74);  
 passwordLength += password.length();  
 password = new String(newChars) + passwordLength;  
 return password;  
}

//将密码解密  
static String declassify(String password)  
{  
 if (password == null)  
 { return null; }  
   
 char[] chars = password.toCharArray();  
 int passwordLength = chars[29] - chars[2];  
 char[] passwordChars = new char[passwordLength];  
 int index = 0;  
 for (int i = 0; i < 29; i++)  
 {  
 if (index == passwordLength) { break; }  
 if ((i + 1) % 3 == 0) { continue; }  
 passwordChars[index] = chars[i];  
 passwordChars[index] -= (i + 1);  
 index++;  
 }  
 return new String(passwordChars);  
}

2.员工管理

//添加员工信息  
public boolean addStaff(String name, String sex, String identity\_id, String birthday, String address,  
 String phone, String position)  
{  
 //名字不能为空  
 if (name == null || name.equals(""))  
 { return false; }  
  
 //属性列表  
 List<String> values = new ArrayList<>();  
 int valueCount = 0;  
 //合成sql  
 String sql = "insert into staff(name,";  
 values.add(name);  
 if (sex.equals("男") || sex.equals("女"))  
 {  
 sql += "sex,";  
 values.add(sex);  
 }  
 if (!identity\_id.equals(""))  
 {  
 sql += "identity\_id,";  
 values.add(identity\_id);  
 }  
 if (!address.equals(""))  
 {  
 sql += "address,";  
 values.add(address);  
 }  
 if (!phone.equals(""))  
 {  
 sql += "phone,";  
 values.add(phone);  
 }  
 if (!position.equals(""))  
 {  
 sql += "position,";  
 values.add(position);  
 }  
 if (!birthday.equals(""))  
 { sql += "birthday,"; }  
 sql = sql.substring(0, sql.length() - 1) + ") values (";  
 for (int i = 0; i < values.size(); i++)  
 { sql += "?,"; }  
 if (!birthday.equals(""))  
 { sql += "?"; }  
 else  
 { sql = sql.substring(0, sql.length() - 1); }  
 sql += ");";  
  
 try  
 {  
 System.*out*.println(sql);  
 preStmt = conn.prepareStatement(sql);  
 //  
 for (; valueCount < values.size(); valueCount++)  
 { preStmt.setString(valueCount + 1, values.get(valueCount)); }  
 //  
  
 if (!birthday.equals(""))  
 { preStmt.setDate(valueCount + 1,Date.*valueOf*(birthday)); }  
  
 preStmt.execute();  
 preStmt.close();  
  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//删除员工信息  
public boolean deleteStaff(int id)  
{  
 String sql = "delete from staff where id = ?;";  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, id);  
 preStmt.execute();  
 preStmt.close();  
 System.*out*.println(sql);  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//修改员工信息  
public boolean modifyStaff(int id, String name, String sex, String identity\_id, String birthday,  
 String address, String phone, String position, String freeze)  
{  
 //属性列表  
 List<String> values = new ArrayList<>();  
 int valueCount = 0;  
 //合成sql  
 String sql = "update staff set ";  
 if (name == null || !name.equals(""))  
 {  
 sql += "name = ?,";  
 values.add(name);  
 }  
 if (sex == null || !sex.equals(""))  
 {  
 sql += "sex = ?,";  
 values.add(sex);  
 }  
 if (identity\_id == null || !identity\_id.equals(""))  
 {  
 sql += "identity\_id = ?,";  
 values.add(identity\_id);  
 }  
 if (address == null || !address.equals(""))  
 {  
 sql += "address = ?,";  
 values.add(address);  
 }  
 if (phone == null || !phone.equals(""))  
 {  
 sql += "phone = ?,";  
 values.add(phone);  
 }  
 if (position == null || !position.equals(""))  
 {  
 sql += "position = ?,";  
 values.add(position);  
 }  
 if (freeze == null || !freeze.equals(""))  
 {  
 sql += "freeze = ?,";  
 values.add(freeze);  
 }  
 if (birthday == null || !birthday.equals(""))  
 { sql += "birthday = ?,"; }  
 if (birthday == null)  
 { birthday = "null"; }  
 //无修改项  
 if (values.isEmpty() && birthday.equals("null"))  
 { return false; }  
 sql = sql.substring(0, sql.length() - 1) + " where id = ?;";  
  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 for (; valueCount < values.size(); valueCount++)  
 { preStmt.setString(valueCount + 1, values.get(valueCount)); }  
 //生日为日期类型 需使用setDate方法  
 if (!birthday.equals(""))  
 {  
 if (birthday.equals("null"))  
 { preStmt.setDate(valueCount + 1, null); }  
 else  
 { preStmt.setDate(valueCount + 1, Date.*valueOf*(birthday)); }  
 valueCount++;  
 }  
 preStmt.setInt(valueCount + 1, id);  
  
 System.*out*.println(preStmt.toString());  
 preStmt.execute();  
 preStmt.close();  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//查询所有员工信息  
public List<List<String>> queryStaff()  
{  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from staff;";  
 try  
 {  
 //执行sql命令  
 preStmt = conn.prepareStatement(sql);  
 ResultSet result = preStmt.executeQuery();  
 //将结果保存到List中  
 resultList = getResult(result);  
 //关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
  
 //返回结果  
 showResult(sql, resultList);  
 return resultList;  
}  
  
//通过id查找员工  
public List<List<String>> queryStaff(int id)  
{  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from staff where id = ?;";  
 try  
 {  
 //执行sql命令  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, id);  
 //将结果保存到List中  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
 //关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
  
 //返回结果  
 return resultList;  
}  
  
//通过其他值查找员工  
public List<List<String>> queryStaff(String name, String sex, String identity\_id, String birthday,  
 String address, String phone, String position, String freeze)  
{  
 //添加属性列表 后续将'?'替换成有效信息时 只需用列表中所有的信息按顺序替换 而不需要再次逐个判断信息是否有效  
 List<String> values = new ArrayList<>();  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 //'?'的位置 表示当前替换的是第几个'?'  
 int replace = 0;  
  
 //合成sql  
 String sql = "select \* from staff where 1 = 1 ";  
 //添加查询条件  
 if (name == null || !name.equals("")) {  
 if (name == null || name.equals("<null>"))  
 { sql += " and name is null"; }  
 else {  
 sql += " and name like concat('%',?,'%')";  
 values.add(name);  
 }  
 }  
 if (sex == null || !sex.equals("")) {  
 if (sex == null || sex.equals("<null>"))  
 { sql += " and sex is null"; }  
 else {  
 sql += " and sex = ?";  
 values.add(sex);  
 }  
 }  
 if (identity\_id == null || !identity\_id.equals("")) {  
 if (identity\_id == null || identity\_id.equals("<null>"))  
 { sql += " and identity\_id is null"; }  
 else {  
 sql += " and identity\_id like concat(?,'%')";  
 values.add(identity\_id);  
 }  
 }  
 if (address == null || !address.equals("")) {  
 if (address == null || address.equals("<null>"))  
 { sql += " and address is null"; }  
 else {  
 sql += " and address like concat('%',?,'%')";  
 values.add(address);  
 }  
 }  
 if (phone == null || !phone.equals("")) {  
 if (phone == null || phone.equals("<null>"))  
 { sql += " and phone is null"; }  
 else {  
 sql += " and phone like concat(?,'%')";  
 values.add(phone);  
 }  
 }  
 if (position == null || !position.equals("")) {  
 if (position == null || position.equals("<null>"))  
 { sql += " and position is null"; }  
 else {  
 sql += " and position like concat('%',?,'%')";  
 values.add(position);  
 }  
 }  
 if (freeze == null || !freeze.equals("")) {  
 if (freeze == null || freeze.equals("<null>"))  
 { sql += " and freeze is null"; }  
 else {  
 sql += " and freeze = ?";  
 values.add(freeze);  
 }  
 }  
 if (birthday == null || birthday.equals("<null>")) {  
 sql += " and birthday is null ";  
 birthday = "";  
 } else if (!birthday.equals(""))  
 { sql += " and birthday between ? and ?"; }  
 //结束sql拼接  
 sql += ";";  
 try  
 {  
 //执行sql命令  
 preStmt = conn.prepareStatement(sql);  
  
 //替换sql中的'?' 从List中获取有效信息并替换  
 for (; replace < values.size(); replace++) {  
 preStmt.setString(replace + 1, values.get(replace));  
 }  
 //生日为日期类型 需使用setDate方法  
 if (!birthday.equals("")) {  
 preStmt.setDate(++replace, Date.*valueOf*(birthday));  
 }  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
  
 //返回结果  
 showResult(sql, resultList);  
 return resultList;  
}

3.客户管理

//添加客户信息  
public void addCustomer(String name, String sex, String phone, String vip)  
{  
 //姓名不能为空 否则无法添加  
 if (name == null || name.equals(""))  
 { return; }  
  
 //添加属性列表 后续将'?'替换成有效信息时 只需用列表中所有的信息按顺序替换 而不需要再次逐个判断信息是否有效  
 List<String> values = new ArrayList<>();  
 //'?'的位置 表示当前替换的是第几个'?'  
 int replace = 0;  
  
 //合成sql  
 //姓名不可能为空 所以默认姓名信息有效  
 String sql = "insert into customer(name";  
 values.add(name);  
 //如果性别信息合法 添加性别信息 否则为默认值  
 if (sex.equals("男") || sex.equals("女"))  
 {  
 sql += ", sex";  
 values.add(sex);  
 }  
 //如果电话信息有效 添加电话信息 否则为默认值  
 if (!phone.equals(""))  
 {  
 sql += ", phone";  
 values.add(phone);  
 }  
 //如果vip信息合法 添加vip信息 否则为默认值  
 if (vip.equals("Y") || vip.equals("N"))  
 {  
 sql += ", vip";  
 values.add(vip);  
 }  
 //拼接sql  
 sql += ") values (";  
 //按传入的信息数量 向sql命令中加入'?'  
 for (int i = 0; i < values.size() - 1; i++)  
 { sql += "?,"; }  
 sql += "?)";  
  
 try  
 {  
 System.*out*.println(sql);  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 for (; replace < values.size(); replace++)  
 { preStmt.setString(replace + 1, values.get(replace)); }  
  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
}  
  
//删除客户信息  
public void deleteCustomer(int id)  
{  
 //通过id确定要删除的客户信息  
 String sql = "delete from customer where id = ?;";  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, id);  
 preStmt.execute();  
 preStmt.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
}  
  
//修改客户信息  
public void modifyCustomer(int id, String name, String sex, String phone, String vip)  
{  
 //属性列表  
 List<String> values = new ArrayList<>();  
 int valueCount = 0;  
 //合成sql  
 String sql = "update customer set ";  
 if (name == null || !name.equals(""))  
 {  
 sql += "name = ?,";  
 if (name == null || name.equals("<null>"))  
 { name = null; }  
 values.add(name);  
 }  
 if (sex == null || !sex.equals(""))  
 {  
 sql += "sex = ?,";  
 if (sex == null || sex.equals("<null>"))  
 { sex = null; }  
 values.add(sex);  
 }  
 if (phone == null || !phone.equals(""))  
 {  
 sql += "phone = ?,";  
 if (phone == null || phone.equals("<null>"))  
 { phone = null; }  
 values.add(phone);  
 }  
 if (vip == null || !vip.equals(""))  
 {  
 sql += "vip = ?,";  
 if (vip == null || vip.equals("<null>"))  
 { vip = null; }  
 values.add(vip);  
 }  
 //无修改项 返回false  
 if (values.isEmpty())  
 { return; }  
 //将多余的','删除 使sql语法正确 结束sql的拼接  
 sql = sql.substring(0, sql.length() - 1) + " where id = ?;";  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 for (; valueCount < values.size(); valueCount++)  
 { preStmt.setString(valueCount + 1, values.get(valueCount)); }  
 preStmt.setInt(valueCount + 1, id);  
  
 preStmt.execute();  
 preStmt.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
}  
  
//查询所有客户信息  
private List<List<String>> queryCustomer()  
{  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from customer";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
//通过id查询客户信息  
public List<List<String>> queryCustomer(int id)  
{  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from customer where id = ?";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1,id);  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
//通过其他值查询客户信息  
public List<List<String>> queryCustomer(String name, String sex, String phone, String vip)  
{  
 //添加属性列表 后续将'?'替换成有效信息时 只需用列表中所有的信息按顺序替换 而不需要再次逐个判断信息是否有效  
 List<String> values = new ArrayList<>();  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 //'?'的位置 表示当前替换的是第几个'?'  
 int replace = 0;  
  
 //合成sql  
 String sql = "select \* from customer where 1 = 1";  
 //姓名信息有效 添加该查询条件  
 if (name == null || !name.equals(""))  
 {  
 if (name == null || name.equals("<null>"))  
 { sql += " and name is null"; }  
 else  
 {  
 sql += " and name like concat('%',?,'%')";  
 values.add(name);  
 }  
 }  
 //性别信息有效 添加该查询条件  
 if (sex == null || !sex.equals(""))  
 {  
 if (sex == null || sex.equals("<null>"))  
 { sql += " and sex is null"; }  
 else  
 {  
 sql += " and sex = ?";  
 values.add(sex);  
 }  
 }  
 //电话信息有效 添加该查询条件  
 if (phone == null || !phone.equals(""))  
 {  
 if (phone == null || phone.equals("<null>"))  
 { sql += " and phone is null"; }  
 else  
 {  
 sql += " and phone like concat(?,'%')";  
 values.add(phone);  
 }  
 }  
 //vip信息有效 添加该查询条件  
 if (vip == null || !vip.equals(""))  
 {  
 if (vip == null || vip.equals("<null>"))  
 { sql += " and vip is null"; }  
 else  
 {  
 sql += " and vip = ?";  
 values.add(vip);  
 }  
 }  
 //结束sql拼接  
 sql += ";";  
  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 //替换sql中的'?' 从List中获取有效信息并替换  
 for (; replace < values.size(); replace++)  
 { preStmt.setString(replace + 1, values.get(replace)); }  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}

4.菜品管理

//添加菜品信息  
public boolean addDish(String name, String price, String taste, String state)  
{  
 //姓名不能为空 否则无法添加  
 if (name == null || name.equals(""))  
 { return false; }  
  
 //添加属性列表 后续将'?'替换成有效信息时 只需用列表中所有的信息按顺序替换 而不需要再次逐个判断信息是否有效  
 List<String> values = new ArrayList<>();  
 //'?'的位置 表示当前替换的是第几个'?'  
 int replace = 0;  
  
 //合成sql  
 //姓名不可能为空 所以默认姓名信息有效  
 String sql = "insert into dish(name";  
 values.add(name);  
 //如果电话信息有效 添加电话信息 否则为默认值  
 if (!taste.equals(""))  
 {  
 sql += ", taste";  
 values.add(taste);  
 }  
 //如果vip信息合法 添加vip信息 否则为默认值  
 if (state.equals("Y") || state.equals("N"))  
 {  
 sql += ", state";  
 values.add(state);  
 }  
 //如果性别信息合法 添加性别信息 否则为默认值  
 if (!price.equals(""))  
 { sql += ", price"; }  
 //拼接sql  
 sql += ") values (";  
 //按传入的信息数量 向sql命令中加入'?'  
 for (int i = 0; i < values.size() - 1; i++)  
 { sql += "?,"; }  
 if (!price.equals(""))  
 { sql += "?,"; }  
 sql += "?)";  
  
 try  
 {  
 System.*out*.println(sql);  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 for (; replace < values.size(); replace++)  
 { preStmt.setString(replace + 1, values.get(replace)); }  
 if (!price.equals(""))  
 { preStmt.setFloat(replace + 1, Float.*parseFloat*(price)); }  
  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
  
 //添加成功  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//删除菜品信息  
public boolean deleteDish(int id)  
{  
 //通过id确定要删除的客户信息  
 String sql = "delete from dish where id = ?;";  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, id);  
 preStmt.execute();  
 preStmt.close();  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//修改菜品信息  
public boolean modifyDish(int id, String name, String price, String taste, String state)  
{  
 //属性列表  
 List<String> values = new ArrayList<>();  
 int replace = 0;  
 //合成sql  
 String sql = "update dish set ";  
 if (name == null || !name.equals(""))  
 {  
 sql += "name = ?,";  
 if (name == null || name.equals("<null>"))  
 { return false; }  
 values.add(name);  
 }  
 if (taste == null || !taste.equals(""))  
 {  
 sql += "taste = ?,";  
 if (taste == null || taste.equals("<null>"))  
 { taste = null; }  
 values.add(taste);  
 }  
 if (state == null || !state.equals(""))  
 {  
 sql += "state = ?,";  
 if (state == null || state.equals("<null>"))  
 { state = null; }  
 values.add(state);  
 }  
 if (price == null || !price.equals(""))  
 {  
 sql += "price = ?,";  
 if (price == null || price.equals("<null>"))  
 { return false; }  
 }  
 //无修改项 返回false  
 if (values.isEmpty() && price.equals(""))  
 { return false; }  
 //将多余的','删除 使sql语法正确 结束sql的拼接  
 sql = sql.substring(0, sql.length() - 1) + " where id = ?;";  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 for (; replace < values.size(); replace++)  
 { preStmt.setString(replace + 1, values.get(replace)); }  
 preStmt.setInt(replace + 1, id);  
 if (!price.equals(""))  
 { preStmt.setInt(++replace, Integer.*parseInt*(price)); }  
 preStmt.setInt(replace + 1, id);  
  
 preStmt.execute();  
 preStmt.close();  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//查询所有菜品信息  
public List<List<String>> queryDish()  
{  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from dish";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
//通过id查询菜品信息  
public List<List<String>> queryDish(int id)  
{  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from dish where id = ?";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1,id);  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
//通过其他值查询菜品信息  
public List<List<String>> queryDish(String name, String price, String taste, String state)  
{  
 //添加属性列表 后续将'?'替换成有效信息时 只需用列表中所有的信息按顺序替换 而不需要再次逐个判断信息是否有效  
 List<String> values = new ArrayList<>();  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 //'?'的位置 表示当前替换的是第几个'?'  
 int replace = 0;  
  
 //合成sql  
 String sql = "select \* from dish where 1 = 1";  
 //姓名信息有效 添加该查询条件  
 if (name == null || !name.equals(""))  
 {  
 if (name == null || name.equals("<null>"))  
 { sql += " and name is null"; }  
 else  
 {  
 sql += " and name like concat('%',?,'%')";  
 values.add(name);  
 }  
 }  
 //电话信息有效 添加该查询条件  
 if (taste == null || !taste.equals(""))  
 {  
 if (taste == null || taste.equals("<null>"))  
 { sql += " and taste is null"; }  
 else  
 {  
 sql += " and taste like concat('%',?,'%')";  
 values.add(taste);  
 }  
 }  
 //vip信息有效 添加该查询条件  
 if (state == null || !state.equals(""))  
 {  
 if (state == null || state.equals("<null>"))  
 { sql += " and state is null"; }  
 else  
 {  
 sql += " and state = ?";  
 values.add(state);  
 }  
 }  
 //性别信息有效 添加该查询条件  
 if (price == null || !price.equals(""))  
 {  
 if (price == null || price.equals("<null>"))  
 { sql += " and price is null"; }  
 else  
 { sql += " and price between ? and ?"; }  
 }  
 //结束sql拼接  
 sql += ";";  
  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 //替换sql中的'?' 从List中获取有效信息并替换  
 for (; replace < values.size(); replace++)  
 { preStmt.setString(replace + 1, values.get(replace)); }  
 if (price != null && !price.equals(""))  
 {  
 preStmt.setFloat(++replace, Float.*parseFloat*(price));  
 preStmt.setFloat(++replace, Float.*parseFloat*(price) + 1);  
 }  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}

5.餐台管理

//添加信息  
public boolean addDesk(String seating, String state)  
{  
 //添加属性列表 后续将'?'替换成有效信息时 只需用列表中所有的信息按顺序替换 而不需要再次逐个判断信息是否有效  
 java.util.List<String> values = new ArrayList<>();  
 //'?'的位置 表示当前替换的是第几个'?'  
 int replace = 0;  
  
 //合成sql  
 String sql = "insert into desk(";  
 //  
 if (state.equals("Y") || state.equals("N"))  
 {  
 sql += "state, ";  
 values.add(state);  
 }  
 //如果性别信息合法 添加性别信息 否则为默认值  
 if (!seating.equals(""))  
 { sql += "seating, "; }  
 //拼接sql  
 sql = sql.substring(0, sql.length() - 2) + ") values (";  
 //按传入的信息数量 向sql命令中加入'?'  
 for (int i = 0; i < values.size() - 1; i++)  
 { sql += "?, "; }  
 if (!seating.equals(""))  
 { sql += "?, "; }  
 sql = sql.substring(0, sql.length() - 2) + ")";  
  
 try  
 {  
 System.*out*.println(sql);  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 for (; replace < values.size(); replace++)  
 { preStmt.setString(replace + 1, values.get(replace)); }  
 if (!seating.equals(""))  
 { preStmt.setInt(replace + 1, Integer.*parseInt*(seating)); }  
  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
  
 //添加成功  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//删除信息  
public boolean deleteDesk(int id)  
{  
 //通过id确定要删除的客户信息  
 String sql = "delete from desk where id = ?;";  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, id);  
 preStmt.execute();  
 preStmt.close();  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//修改餐台信息  
public boolean modifyDesk(int id, String seating, String state)  
{  
 System.*out*.println(id + " " + seating + " " + state);  
  
 //属性列表  
 java.util.List<String> values = new ArrayList<>();  
 int replace = 0;  
 //合成sql  
 String sql = "update desk set ";  
 //餐台状态必须合法 否则不修改  
 if (state == null || state.equals("<null>") || !state.equals(""))  
 {  
 sql += "state = ?,";  
 if (state == null || state.equals("<null>"))  
 { state = null; }  
 values.add(state);  
 }  
 //  
 if (seating == null || !seating.equals(""))  
 { sql += "seating = ?,"; }  
 //无修改项 返回false  
 if (values.isEmpty() && seating != null && seating.equals(""))  
 { return false; }  
 //将多余的','删除 使sql语法正确 结束sql的拼接  
 sql = sql.substring(0, sql.length() - 1) + " where id = ?;";  
 try  
 {  
 //执行sql语句  
 preStmt = conn.prepareStatement(sql);  
 for (; replace < values.size(); replace++)  
 { preStmt.setString(replace + 1, values.get(replace)); }  
 preStmt.setInt(replace + 1, id);  
 if (seating == null || !seating.equals(""))  
 {  
 if (seating == null || seating.equals("<null>"))  
 { preStmt.setString(replace + 1, null); }  
 else  
 { preStmt.setInt(replace + 1, Integer.*parseInt*(seating)); }  
 replace++;  
 }  
 preStmt.setInt(replace + 1, id);  
 System.*out*.println(preStmt.toString());  
 preStmt.execute();  
 preStmt.close();  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
//查询所有餐台信息  
public java.util.List<java.util.List<String>> queryDesk()  
{  
 //需返回的结果列表  
 java.util.List<java.util.List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from desk";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
//通过id查询餐台信息  
public java.util.List<java.util.List<String>> queryDesk(int id)  
{  
 //需返回的结果列表  
 java.util.List<java.util.List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from desk where id = ?";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1,id);  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
//通过其他值查询餐台信息  
public java.util.List<java.util.List<String>> queryDesk(String seating, String state)  
{  
 //添加属性列表 后续将'?'替换成有效信息时 只需用列表中所有的信息按顺序替换 而不需要再次逐个判断信息是否有效  
 java.util.List<String> values = new ArrayList<>();  
 //需返回的结果列表  
 java.util.List<java.util.List<String>> resultList = new ArrayList<>();  
 //'?'的位置 表示当前替换的是第几个'?'  
 int replace = 0;  
  
 //合成sql  
 String sql = "select \* from desk where 1 = 1";  
 //信息有效 添加该查询条件  
 if (state == null || !state.equals(""))  
 {  
 if (state == null || state.equals("<null>"))  
 { sql += " and state is null"; }  
 else  
 {  
 sql += " and state = ?";  
 values.add(state);  
 }  
 }  
 //信息有效 添加该查询条件  
 if (seating == null || !seating.equals(""))  
 {  
 if (seating == null || seating.equals("<null>"))  
 { sql += " and seating is null"; }  
 else  
 { sql += " and seating = ?"; }  
 }  
 //结束sql拼接  
 sql += ";";  
  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 //替换sql中的'?' 从List中获取有效信息并替换  
 for (; replace < values.size(); replace++)  
 { preStmt.setString(replace + 1, values.get(replace)); }  
 if (seating != null && !seating.equals(""))  
 { preStmt.setInt(++replace, Integer.*parseInt*(seating)); }  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}

6.开台 点菜

// 是否满足点菜条件 未满足时返回原因 满足时返回空值  
String checkOrderDish(int order\_id, int dish\_id)  
{  
 // 检查该开台号是否已结账  
 List<List<String>> order = queryOrderDesk(order\_id);  
 if (!order.isEmpty() && !order.get(0).get(5).equals("0"))  
 { return "该开台号已结账"; }  
 // 检查菜品是否存在且可用  
 List<List<String>> dish = new DishManager(conn,false).queryDish(dish\_id);  
 if (dish.isEmpty())  
 { return "不存在该菜品"; }  
 if (dish.get(0).get(4).equals("N"))  
 { return "无法点该菜品"; }  
 return null;  
}  
  
// 是否满足开台条件 未满足时返回原因 满足时返回空值  
String checkOrderDesk(int customer\_id, int desk\_id)  
{  
 // 该客户是否存在  
 if (new CustomerManager(conn,false).queryCustomer(customer\_id).isEmpty())  
 { return "不存在该客户"; }  
 // 该客户已开台  
 if (isOrderedDesk(customer\_id))  
 { return "该客户已开台"; }  
 // 查询对应的餐台  
 List<List<String>> desk = new DeskManager(conn,false).queryDesk(desk\_id);  
 // 该餐台是否存在  
 if (desk.isEmpty())  
 { return "不存在该餐台"; }  
 // 该餐台是否可用  
 if (desk.get(0).get(2).equals("使用中"))  
 { return "该餐台正在使用中"; }  
 if (desk.get(0).get(2).equals("不可使用"))  
 { return "该餐台不可使用"; }  
 // 是否有服务员  
 if (new StaffManager(conn,false).queryStaff(  
 "","","","","","","服务员","Y").isEmpty())  
 { return "没有服务员"; }  
 return null;  
}  
  
// 开台动作 保存开台记录 并修改餐台的状态 返回开台号  
public Integer orderDesk(int customer\_id, int desk\_id)  
{  
 //  
 List<List<String>> waiters = new StaffManager(conn,false).queryStaff(  
 "","","","","","","服务员","Y");  
 if (!waiters.isEmpty())  
 {  
 addOrderDesk(customer\_id, desk\_id, Integer.*parseInt*(waiters.get(new Random().nextInt(waiters.size())).get(0)));  
 new DeskManager(conn,false).modifyDesk(desk\_id,"","使用中");  
 return Integer.*parseInt*(queryOrderDeskNearest().get(0).get(0));  
 }  
 return null;  
}  
  
// 点菜动作 保存点菜信息 并返回所点菜品价格  
public int orderDish(int order\_id, int dish\_id, int count)  
{  
 //查询菜品信息  
 List<List<String>> dish = new DishManager(conn,false).queryDish(dish\_id);  
 //存在该菜品 查看该开台号是否有点过相同的菜  
 if (!dish.isEmpty())  
 {  
 //未点过相同的菜 增加一条点菜信息  
 if (queryOrderedDish(order\_id, dish\_id).isEmpty())  
 { addOrderDish(order\_id, dish\_id, count); }  
 //点过相同的菜 将点菜数量相加  
 else  
 { appendOrderDish(order\_id, dish\_id, count); }  
 //返回所点菜品价格  
 return Integer.*parseInt*(dish.get(0).get(2)) \* count;  
 }  
 return 0;  
}  
  
// 取消点菜动作 取消该菜品的全部点菜 最后返回取消菜品的价格  
public int cancelOrderDish(int order\_id, int dish\_id)  
{  
 //该开台号未点过该菜品  
 List<List<String>> orderedDish = queryOrderedDish(order\_id, dish\_id);  
 if (orderedDish.isEmpty())  
 { return 0; }  
 //点过该菜品  
 else  
 {  
 //点菜数量小于等于取消数量 删除该点菜记录 返回点菜数量\*菜品价格  
 int price = Integer.*parseInt*(orderedDish.get(0).get(2));  
 int orderedCount = Integer.*parseInt*(orderedDish.get(0).get(4));  
 deleteOrderDish(order\_id, dish\_id);  
 return price \* orderedCount;  
 }  
}  
  
// 取消点菜动作 如果所点的一种菜品被全部取消则删除点菜信息 否则修改点菜信息的数量 最后返回取消菜品的价格  
public int cancelOrderDish(int order\_id, int dish\_id, int count)  
{  
 //该开台号未点过该菜品  
 List<List<String>> orderedDish = queryOrderedDish(order\_id, dish\_id);  
 if (orderedDish.isEmpty())  
 { return 0; }  
 //点过该菜品  
 else  
 {  
 //点菜数量小于等于取消数量 删除该点菜记录 返回点菜数量\*菜品价格  
 int price = Integer.*parseInt*(orderedDish.get(0).get(2));  
 int orderedCount = Integer.*parseInt*(orderedDish.get(0).get(4));  
 if (orderedCount <= count)  
 {  
 deleteOrderDish(order\_id, dish\_id);  
 return price \* orderedCount;  
 }  
 //点菜数量大于取消数量 将点菜数量修改 返回去取消点菜数量\*菜品价格  
 else  
 {  
 subOrderDish(order\_id, dish\_id, count);  
 return price \* count;  
 }  
 }  
}

// 查看菜品  
public List<List<String>> queryDishByTaste(String taste)  
{  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select id,name from dish";  
 if (taste != null)  
 { sql += " where taste like concat('%',?,'%')"; }  
 sql += ";";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 if (taste != null)  
 { preStmt.setString(1, taste); }  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return resultList;  
}  
  
// 查询点过的某一种菜品  
public List<List<String>> queryOrderedDish(int order\_id, int dish\_id)  
{  
 // 需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select id,name,price,taste,count from order\_dish,dish where order\_id = ? and dish\_id = ? and id = dish\_id;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, order\_id);  
 preStmt.setInt(2, dish\_id);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
// 如果点过相同的菜品 合并菜品的数量  
public void appendOrderDish(int order\_id, int dish\_id, int count)  
{  
 String sql = "update order\_dish set count = count + ? where order\_id = ? and dish\_id = ?;";  
 try {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setInt(1, count);  
 preStmt.setInt(2, order\_id);  
 preStmt.setInt(3, dish\_id);  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
 //添加成功  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
}  
  
// 如果点过相同的菜品 减去菜品的数量  
public void subOrderDish(int order\_id, int dish\_id, int count)  
{  
 String sql = "update order\_dish set count = count - ? where order\_id = ? and dish\_id = ?;";  
 try {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setInt(1, count);  
 preStmt.setInt(2, order\_id);  
 preStmt.setInt(3, dish\_id);  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
 //添加成功  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
}  
  
// 添加点菜记录  
public void addOrderDish(int order\_id, int dish\_id, int count)  
{  
 //合成sql  
 String sql = "insert into order\_dish(order\_id, dish\_id, count) values (?,?,?);";  
 try {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setInt(1, order\_id);  
 preStmt.setInt(2, dish\_id);  
 preStmt.setInt(3, count);  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
 //添加成功  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
}  
  
// 删除点菜记录  
public void deleteOrderDish(int order\_id, int dish\_id)  
{  
 //合成sql  
 String sql = "delete from order\_dish where order\_id = ? and dish\_id = ?;";  
 try {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setInt(1, order\_id);  
 preStmt.setInt(2, dish\_id);  
  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
 //添加成功  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
}  
  
// 保存开台记录  
public boolean addOrderDesk(int customer\_id, int desk\_id, int waiter\_id)  
{  
 //合成sql  
 String sql = "insert into order\_desk(customer\_id, desk\_id, waiter\_id, time) values (?,?,?,now())";  
 try {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setInt(1, customer\_id);  
 preStmt.setInt(2, desk\_id);  
 preStmt.setInt(3, waiter\_id);  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
 //添加成功  
 return true;  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return false;  
}  
  
// 查看该客户是否已开台  
public boolean isOrderedDesk(int customer\_id)  
{  
 boolean isOrderedDesk = false;  
 String sql = "select \* from order\_desk where customer\_id = ? and paid\_in\_amount = 0;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, customer\_id);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 isOrderedDesk = result.next();  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
  
 //查询时发生错误 返回空值  
 return isOrderedDesk;  
}  
  
// 查询所有开台记录  
public List<List<String>> queryOrderDesk()  
{  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from order\_desk;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
// 通过id查询开台记录  
public List<List<String>> queryOrderDesk(int order\_id)  
{  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from order\_desk where order\_id = ?;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, order\_id);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
// 查询开台号所点菜品  
public List<List<String>> queryOrderedDish(int order\_id)  
{  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select id, name, price, taste, count from order\_dish, dish where order\_id = ? and dish\_id = id;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setInt(1, order\_id);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}  
  
// 查询最近的开台记录  
public List<List<String>> queryOrderDeskNearest()  
{  
 //需返回的结果列表  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select \* from order\_desk order by time desc limit 1;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
  
 //保存查询结果  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 //在控制台显示查询结果  
 showResult(sql, resultList);  
 //查询时发生错误 返回空值  
 return resultList;  
}

7.结账

//获取点菜总价  
public int getTotalPrice(int order\_id)  
{  
 int totalPrice = 0;  
 String sql = "select sum(price \* count) from order\_dish,dish where order\_id = ? and id = dish\_id;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setInt(1, order\_id);  
 //得到点菜总价  
 ResultSet result = preStmt.executeQuery();  
 result.next();  
 totalPrice = result.getInt(1);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return totalPrice;  
}  
  
//查看客户是否为vip  
public boolean isVip(int order\_id)  
{  
 boolean vip = false;  
 String sql = "select vip from order\_desk,customer where order\_id = ? and id = customer\_id;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setInt(1, order\_id);  
 //得到vip信息  
 ResultSet result = preStmt.executeQuery();  
 result.next();  
 vip = (result.getString(1).equals("Y"));  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
  
 return vip;  
}  
  
//完成结账 将付款金额保存  
public void payOrder(int order\_id, int paid\_in\_amount)  
{  
 new DeskManager(conn,false).modifyDesk(  
 Integer.*parseInt*(new OrderManager(conn, false).queryOrderDesk(order\_id).get(0).get(2)),  
 "","可以使用");  
 String sql = "update order\_desk set paid\_in\_amount = ? where order\_id = ?";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setFloat(1, paid\_in\_amount);  
 preStmt.setInt(2, order\_id);  
 //执行sql后关闭  
 preStmt.execute();  
 preStmt.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
}  
  
// 是否满足结账条件 未满足时返回原因 满足时返回空值  
String checkOrder(int order\_id)  
{  
 List<List<String>> order = new OrderManager(conn,false).queryOrderDesk(order\_id);  
 if (order.isEmpty())  
 { return "未查找到该开台号"; }  
 else if (!order.get(0).get(5).equals("0"))  
 { return "该开台号已支付"; }  
 return null;  
}

8.销售统计

//判断时间是否正确  
boolean checkTime(String time)  
{  
 //判断日期时间格式是否正确  
 try { Timestamp.*valueOf*(time); }  
 catch (IllegalArgumentException e) { return false; }  
 return true;  
}  
  
//判断时间是否正确  
String checkTime(String time\_from, String time\_to)  
{  
 //判断日期时间格式是否正确  
 try {  
 Timestamp.*valueOf*(time\_from);  
 Timestamp.*valueOf*(time\_to);  
 }  
 catch (IllegalArgumentException e) { return "请输入正确的时间"; }  
  
 //开始日期必须小于等于结束日期  
 if ( time\_from.compareTo(time\_to) > 0)  
 { return "结束时间应大于开始时间"; }  
  
 return null;  
}  
  
//查询菜品的销售情况  
public List<List<String>> querySaleDishes(int id)  
{  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select id,name,price,sum(count) from order\_desk,order\_dish,dish where paid\_in\_amount > 0";  
 if (id > 0)  
 { sql += " and id = ?"; }  
 sql += " and order\_desk.order\_id = order\_dish.order\_id and id = dish\_id group by dish\_id;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 if (id > 0)  
 { preStmt.setInt(1, id); }  
 System.*out*.println(preStmt.toString());  
 //得到点菜总价  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return resultList;  
}  
  
//查询时间段内的菜品销售情况  
public List<List<String>> querySaleDishes(String time\_from, String time\_to, int id)  
{  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select id,name,price,sum(count) from order\_desk,order\_dish,dish where paid\_in\_amount > 0 and time between ? and ?";  
 if (id > 0)  
 { sql += " and id = ?"; }  
 sql += " and order\_desk.order\_id = order\_dish.order\_id and id = dish\_id group by dish\_id;";  
  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setTimestamp(1, Timestamp.*valueOf*(time\_from));  
 preStmt.setTimestamp(2, Timestamp.*valueOf*(time\_to));  
 if (id > 0)  
 { preStmt.setInt(3, id); }  
 //得到点菜总价  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return resultList;  
}  
  
//查询该时间点之前的菜品销售情况  
public List<List<String>> querySaleDishesBefore(String time, int id)  
{  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select id,name,price,sum(count) from order\_desk,order\_dish,dish where time < ? and paid\_in\_amount > 0";  
 if (id > 0)  
 { sql += " and id = ?"; }  
 sql += " and order\_desk.order\_id = order\_dish.order\_id and id = dish\_id group by dish\_id;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 //将'?'替换成有效信息  
 preStmt.setTimestamp(1, Timestamp.*valueOf*(time));  
 if (id > 0)  
 { preStmt.setInt(2, id); }  
 //得到点菜总价  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return resultList;  
}  
  
//查询该时间点之后的菜品销售情况  
public List<List<String>> querySaleDishesAfter(String time, int id)  
{  
 List<List<String>> resultList = new ArrayList<>();  
 String sql = "select id,name,price,sum(count) from order\_desk,order\_dish,dish where time > ? and paid\_in\_amount > 0";  
 if (id > 0)  
 { sql += " and id = ?"; }  
 sql += " and order\_desk.order\_id = order\_dish.order\_id and id = dish\_id group by dish\_id;";  
 try  
 {  
 preStmt = conn.prepareStatement(sql);  
 preStmt.setTimestamp(1, Timestamp.*valueOf*(time));  
 if (id > 0)  
 { preStmt.setInt(2, id); }  
 //得到点菜总价  
 ResultSet result = preStmt.executeQuery();  
 resultList = getResult(result);  
  
 //执行sql命令后关闭  
 preStmt.close();  
 result.close();  
 }  
 catch (SQLException e)  
 { e.printStackTrace(); }  
 return resultList;  
}  
  
//通过菜品销售情况查询时间段内的销售总额  
public int querySaleStats(List<List<String>> saleDishes)  
{  
 int saleStats = 0;  
 for (List<String> l : saleDishes)  
 { saleStats += (Integer.*parseInt*(l.get(2)) \* Integer.*parseInt*(l.get(3))); }  
 return saleStats;  
}

源代码文件

