

面向对象程序设计课程设计

|  |  |
| --- | --- |
| **题目：** | **单项选择考试标准化系统设计** |
| **院系：** | **信息科学与工程学院** |
| **专业：** | **软件工程** |
| **班级：** | **软件21-2bf** |
| **指导老师：** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 性别 | 学号 | 学院专业班级 | 电话 |
| 彭飞 | 男 | 14214800889 | 软件21-2bf | 19918336389 |
| 张竣华 | 男 | 14214801902 | 软件21-2bf | 191656812915 |

二〇二三 年 十一月 二十日

目录

[一、课程设计目的 1](#_Toc8414)

[二、 系统功能 1](#_Toc17238)

[1. 基本要求 1](#_Toc26294)

[2. 软件环境 2](#_Toc14444)

[三、 系统概要设计 2](#_Toc15165)

[3.1TestMainFx类 2](#_Toc2207)

[3.2QuestionView类 2](#_Toc9707)

[3.3Teacher类 3](#_Toc26852)

[3.4ExamDAOImpl类 5](#_Toc3538)

[3.5DigitalClock类 6](#_Toc23082)

[3.6ExamMachine类 6](#_Toc30119)

[四、 系统详细设计 7](#_Toc19409)

[4.1TestMainFx类 7](#_Toc4318)

[4.2Teacher类 9](#_Toc26311)

[4.3ExamMachine类 10](#_Toc17105)

[五、 系统实现 15](#_Toc28308)

[5.1考试功能界面 15](#_Toc24596)

[5.2教师功能界面 18](#_Toc20457)

[5.3源代码 22](#_Toc2908)

[六、 收获及体会 22](#_Toc28771)

# 一、课程设计目的

将理论教学中涉及到的知识点贯穿起来，对不同的数据类型、程序控制结构、数据结构作比较和总结，结合设计题目进行综合性应用，对所学知识达到融会贯通的程度。并进一步巩固、加深对Java基本理论知识的掌握，进一步培养综合分析问题，解决问题的能力。

# 系统功能

### 1. 基本要求

1. 、用数据库保存试题库（每个试题包括题干、4个选项、标准答案）。
2. 、试题录入：可随时增加试题到试题库中。
3. 、试题抽取：每次从试题库中可以随机抽出N道题（N由键盘输入）。
4. 、答题:用户可以实现输入自己的答案。
5. 、自动判卷：系统可根据用户答案与标准答案的对比实现判卷并给出成绩。
6. 、退出

### 软件环境

编程环境：Java1.8 mysql8 javafx

集成开发环境：IntelliJ IDEA 2023.1.2

# 系统概要设计

本系统通过构建3个界面类，5个功能类来实现所有功能，分别是TestMainFx.Java、QuestionView.java、Teacher.java、DigitalClock.java、

ExamDAOImpl.java、ExamMachine.java、Question.java、Student.java

## 3**.1TestMainFx类**

### 3.1.1 登录场景

createLoginScene() 方法设置了登录界面，包括用户名、密码输入框和登录按钮。

handleLogin() 方法负责处理登录操作，根据用户名和密码检查学生或老师登录，并相应地显示考试场景或老师功能菜单。

## 3.2QuestionView类

类 QuestionView 主要是用来创建显示单个问题及其选项的视图。它实现了一个 VBox，将题干和对应的选项以一定的格式呈现出来。

### 3.2.1初始化问题视图

initQuestionView() 方法创建题干的标签，按照题号和题目内容的格式进行排列。将题干按照指定的最大字符数拆分成多行显示，以便适应界面布局。创建选项的单选按钮，根据问题中提供的选项内容生成对应的单选按钮。

### 3.2.2获取选中的答案

getSelectedAnswer()方法当用户选择了某个选项时，这个方法会返回用户所选的答案。通过 optionsGroup.getSelectedToggle() 获取选中的单选按钮。如果有选中的单选按钮，会返回其文本内容，即所选的答案选项。如果没有选中的单选按钮，则返回 null。

## 3.3Teacher类

### 3.3.1教师登录验证

loginTeacherCheck 方法通过用户名和密码进行教师的登录验证，使用一个 HashMap 存储了用户名和对应的密码。检查输入的用户名是否在教师信息的数据结构中，如果存在且密码匹配，则返回登录成功。

**3.3.2添加题目**

addQuestionToFile 方法提供了向题库中添加新题目的功能，调用了 ExamDAOImpl 的相关方法实现添加题目操作。

### 3.3.3批阅卷子

checkPaper 方法对学生的答案进行批改，计算得分。根据题目的数量和正确答案判断，计算每道题目的分数，并最终返回总分。

### 3.3.4创建功能场景

creatFunctionScene方法在 JavaFX 中创建教师的功能场景，提供了添加题目、添加教师信息和添加学生信息的按钮。每个按钮对应不同的功能，点击不同按钮会跳转到对应的界面。

### 3.3.5添加学生信息

addStudentScene 方法实现了添加学生信息的界面，输入学生用户名和密码，提交后将信息添加至数据结构。

### 3.3.6添加教师信息

addTeacherScene 方法提供了添加教师信息的界面，类似于添加学生信息，输入教师用户名和密码后，将信息添加至数据结构。

### 3.3.7添加题目

addQuestionsScene 方法实现了向题库中添加新题目的界面，可以输入题目、选项和答案，提交后将题目信息添加至题库。

## 3.4ExamDAOImpl类

ExamDAOImpl 是一个数据访问对象（DAO）的实现，用于管理数据库中的考试、问题、教师和学生信息。

### 3.4.1题目仓库

questionRepository 方法从数据库中获取题目信息，将题目的标题、选项和答案封装为 Question 对象，并存储在一个 HashSet 集合中返回。

### 3.4.2获取所有教师信息

getAllTeachers 方法从数据库中获取所有教师的用户名和密码信息，存储在一个 HashMap 中并返回。

### 3.4.3获取所有学生信息

getAllstudent 方法从数据库中获取所有学生的用户名和密码信息，同样存储在一个 HashMap 中并返回。

### 3.4.4添加题目

addquestion方法向数据库的题目表中添加新的题目，包括题目标题、选项和答案。在添加之前，检查答案是否在选项中，确保数据的有效性。

### 3.4.5添加学生信息

addstudent 方法向数据库的学生表中添加新的学生信息，包括学生的用户名和密码。

### 3.4.6添加教师信息

addteacher 方法向数据库的教师表中添加新的教师信息，包括教师的用户名和密码。

## 3.5DigitalClock类

startDigitalClock 方法用于创建一个 Label 控件作为显示倒计时的时钟。它设置了控件的样式、字体等属性，创建了一个 DigitalClock 对象，并在一个单独的线程中启动这个时钟。

## 3.6ExamMachine类

**3.6.1 生成试卷**

generateRandomizedPaper 方法用于生成随机试卷

### 3.6.2 考试场景

createExamScene() 方法创建了考试界面，包括选择题目数量的输入框和开始考试按钮。

handleStartExam(int numQuestions) 方法处理开始考试的操作，生成随机试卷并展示给考生。

### 3.6.3 答题流程和显示

createExamVisual(ArrayList<Question> paper) 方法创建了答题界面，显示考试题目并提供上下翻页功能。

handleSubmitAnswers(ArrayList<Question> paper) 方法处理考生提交答案的操作，收集学生答案并进行批改，然后展示最终成绩。

### 3.6.4 结果显示

createResultScene(float score) 方法展示考试结果，显示最终成绩。

## 3.7QuestionTableView类

在 TableView 中展示问题及其标题、选项、答案和ID，允许直接在表格中编辑标题、选项和答案，提供连接数据库的功能，可以更新和删除问题。

# 系统详细设计

## 4.1TestMainFx类

### 4.1.1登录界面的构建

1. 使用 BorderPane 作为布局。
2. 创建登录界面元素：TextField 和 PasswordField 用于输入用户名和密码，Button 用于触发登录。
3. 设置背景图片作为界面背景。
4. 通过 VBox 和 Line 组合实现输入框的样式化。

### 4.1.2登录处理方法 (handleLogin 方法)

1. 当用户点击登录按钮时触发，获取输入的用户名和密码。
2. 实例化 ExamMachine、Teacher 和 Student 对象。
3. 调用 loginStudentCheck 和 loginTeacherCheck 方法验证用户名和密码是否匹配学生或教师的信息。
4. 如果学生信息匹配，则调用 examMachine.createExamScene() 方法创建考试界面。
5. 如果教师信息匹配，则调用 teacher.createFunctionScene() 方法创建功能界面。
6. 如果用户名或密码不匹配，弹出错误提示框。

### 4.1.3主要方法

1. createLoginScene(): 构建登录场景。
2. handleLogin(TextField usernameField, PasswordField passwordField): 处理登录逻辑。

## 4.2Teacher类

### 4.2.1变量

Box：用于存储教师的用户名和密码信息，是一个 Map<String, Integer> 类型。

### 4.2.2构造函数

public Teacher(String name, Integer password)：构造函数接收用户名和密码，初始化教师对象，并加载所有教师信息到 Box 中。

### 4.2.3登录校验

public boolean loginTeacherCheck(String name, Integer pwd)：校验教师的用户名和密码是否正确。

### 4.2.4题库操作

public boolean addQuestionToFile(String title, String[] options, String answer)：往题库中添加新的题目。

### 4.2.5批阅试卷

public static float checkPaper(ArrayList<Question> paper, String[] answer)：批阅试卷并计算得分。它接收一个题目集合和学生的答案数组作为参数。该方法模拟了提交答案和批阅的过程，通过线程睡眠来模拟等待的效果。根据学生的答案与题目的正确答案进行比对，计算学生的得分并返回。

### 4.2.6UI交互

1. creatFunctionScene(BorderPane layout)：创建教师功能场景。这个场景包括添加题目、添加教师信息和添加学生信息的按钮。
2. addStudentScene(BorderPane layout)：创建添加学生信息的界面。提供输入学生用户名和密码的功能。
3. addTeacherScene(BorderPane layout)：创建添加教师信息的界面。提供输入教师用户名和密码的功能。
4. addQuestionsScene(BorderPane layout)：创建添加题目信息的界面。提供输入题目和答案的功能。允许教师将题目添加到题库中。

## 4.3ExamMachine类

ExamMachine 类实现了考试系统的核心逻辑和界面，详细描述每个部分的设计和功能：

### 4.3.1变量

1. questionRepository：类型：Set<Question>描述：存储所有题目的集合。
2. questionViews：类型：List<QuestionView>描述：用于管理题目的视图列表。
3. paper：类型：ArrayList<Question>描述：存储随机生成的试卷。
4. currentPage：类型：Integer描述：记录当前所处题目页码。
5. layout：类型：BorderPane描述：界面布局容器。

### 4.3.2方法

（1）构造函数

ExamMachine(BorderPane layout)输出启动信息，调用 load() 方法加载题库，输出初始化完成信息。

（2）加载题目信息

load():使用 ExamDAOImpl 类的实例化对象从数据库中获取题目信息。将获取的题目信息存储到 questionRepository 中。

（3）生成随机试卷

generateRandomizedPaper(int num)从题库中获取题目，对题库中的题目进行随机排序，为每个题目的选项重新排列顺序，生成一份新的试卷，返回指定数量的题目组成的试卷。

### 4.3.3界面

（1）创建考试场景

createExamScene()创建界面上的元素，包含一个输入题目数量的文本框和一个开始考试按钮，按钮点击后，调用 generateRandomizedPaper 方法生成试卷。

（2）创建考试视图

createExamVisual(ArrayList<Question> paper)清空界面，遍历试卷中的每道题目，生成对应的题目视图并添加到题目视图列表中。显示当前页的题目。

（3）显示题目页

showQuestionPage(BorderPane examBox, ArrayList<Question> paper)显示当前页的题目，创建包含当前题目的视图，显示当前题目的内容和导航按钮，更新界面显示为当前页的题目内容。

（4）处理提交的答案

handleSubmitAnswers(ArrayList<Question> paper)收集学生提交的答案，遍历题目视图列表，获取学生选择的答案，计算学生得分并显示结果。

（5）创建成绩页面

createResultScene(float score)清空界面。显示考试得分的界面。

界面：包含输入题目数量的文本框和开始考试按钮。

## 4.4QuestionTableView类

继承自 VBox，是一个自定义的 JavaFX 控件，包含了一个 TableView 控件，用于显示问题信息，创建了多个 TableColumn 用于显示问题的不同属性（标题、选项、答案、ID）。

### 4.4.1关键组件

（1）TableView（表格视图）： 包含问题及其详细信息。

（2）TableColumns（表格列）： 表示 Question 的不同属性（标题、选项、答案、ID）。

（3）Cell Factories（单元格工厂）： 定义了表格列的自定义逻辑，用于编辑和显示单元格内容。

### 4.4.2功能

（1）编辑： 允许在表格中直接编辑标题、选项和答案。

（2）右键菜单： 提供右键菜单选项，可以删除问题。

（3）数据库交互： 当用户在表格中编辑问题信息时，会调用相关的数据库操作，将修改同步到数据库中。

## 4.5DigitalClock类

这个 DigitalClock 类是一个倒计时器，这个类提供了一个简单的倒计时器功能，可以在界面上显示倒计时，并在计时结束时执行指定的操作，例如提交考试答案。

### 4.5.1类结构

DigitalClock 类：

实现了 Runnable 接口，可以作为一个线程运行，包含了一个 Label 对象用于显示倒计时，使用 Duration 对象来表示时间段。

### 4.5.2成员变量

1. isRunning（boolean）： 用于控制计时器是否正在运行。
2. timerLabel（Label）： 用于显示倒计时的标签。
3. duration（Duration）： 表示倒计时的时间段。
4. examMachine（ExamMachine）： 引用了一个 ExamMachine 对象，用于执行特定操作。

### 4.5.3方法和功能

1. 构造函数： 接受一个 Label 和一个 ExamMachine 实例，初始化倒计时器的相关属性。
2. startDigitalClock 方法： 静态方法，用于创建一个新的 DigitalClock 实例，并启动倒计时线程。
3. run 方法： 实现了倒计时的主要逻辑，在倒计时运行期间更新界面上的倒计时显示，并在计时结束时执行指定操作。
4. formatDuration 方法： 将 Duration 对象格式化为小时、分钟、秒的字符串形式。
5. 运行逻辑：

在 run 方法中，通过 Platform.runLater 更新界面上的倒计时显示，每隔一秒钟更新一次显示，直到倒计时结束，倒计时结束后，执行 examMachine 的 handleSubmitAnswers 方法。进行试卷提交。

### 4.5.4线程控制

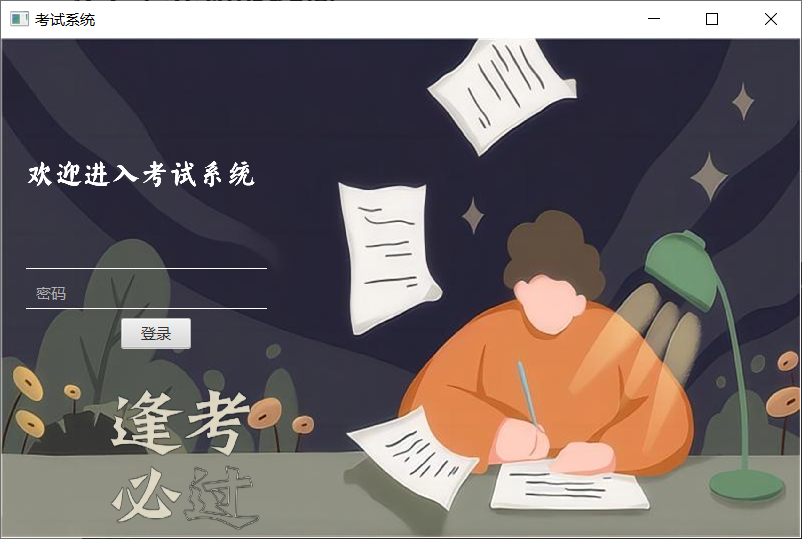
使用 isRunning 控制线程是否继续运行，倒计时结束或计时被中断时，通过 Platform.runLater 更新界面上的显示信息。

# 系统实现

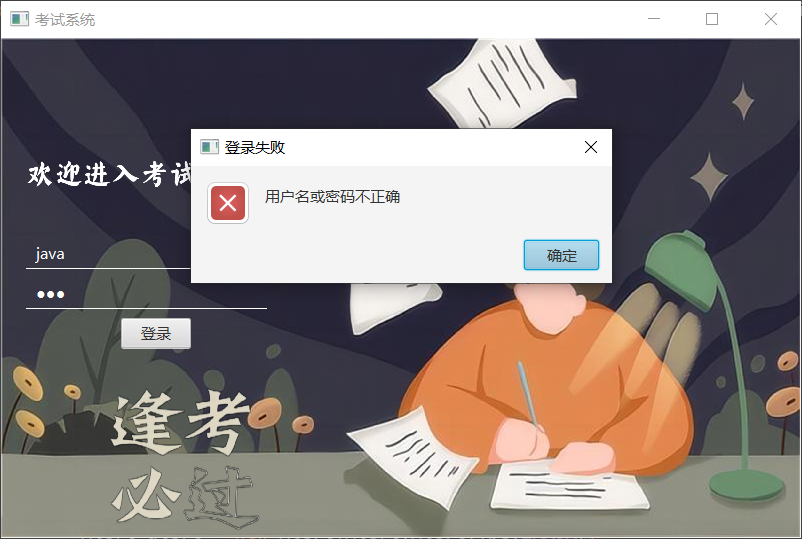
## 5.1考试功能界面

这一部分我们对我们程序运行的界面和功能做一个简单的展示。包括学生登入界面，登陆失败，题目数量选择界面，试卷评分界面。

### 5.1.1学生登入界面



**5.1.2登陆失败界面**



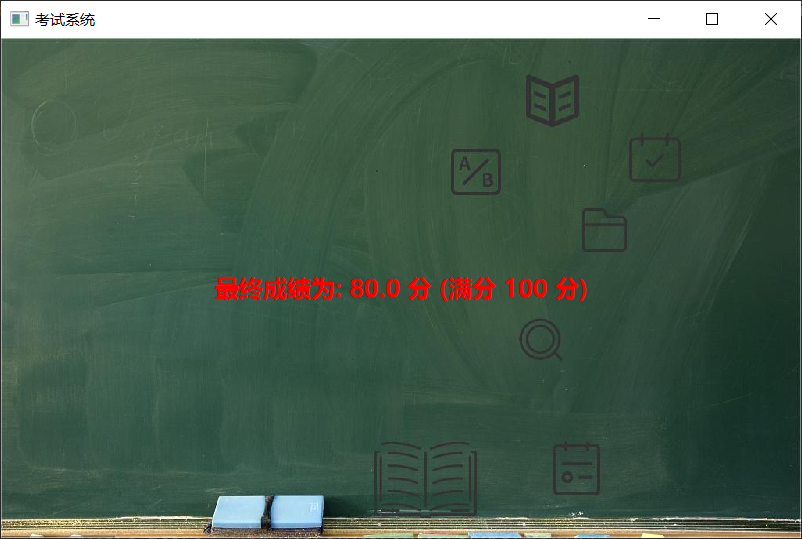
**5.1.3题目选择界面**



### 5.1.4考试界面



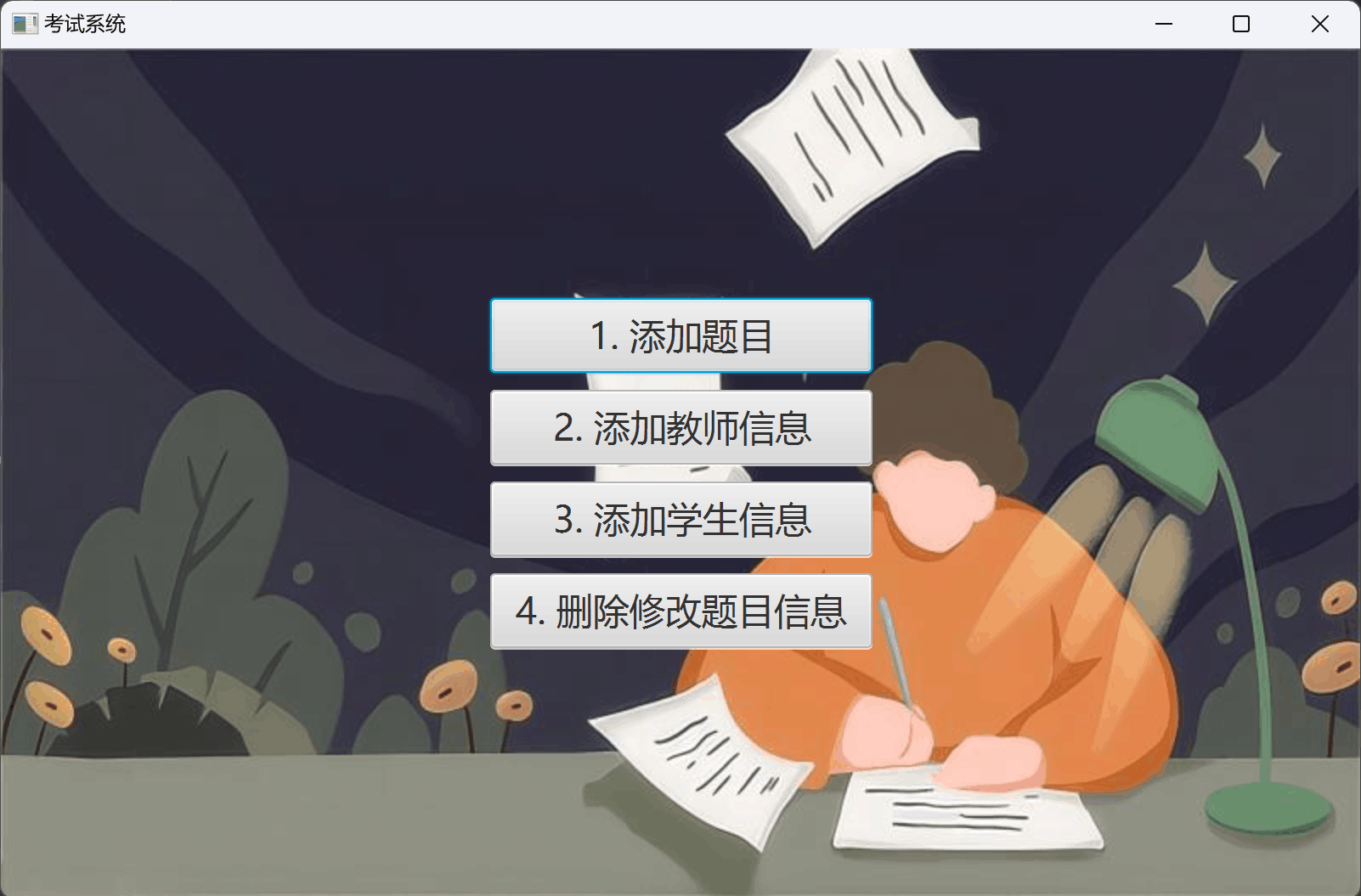
### 5.1.5试卷评分界面



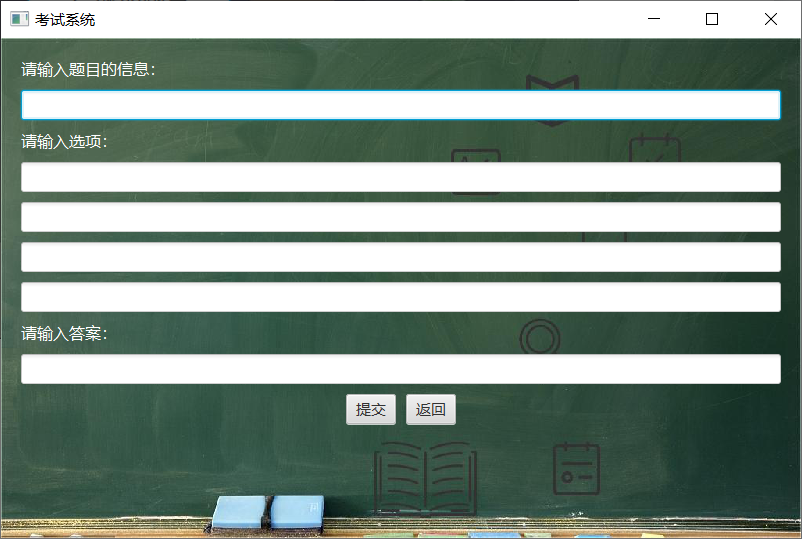
## 5.2教师功能界面

这一部分我们对我们程序教师功能运行的界面和功能做一个简单的展示。包括功能选择界面，

### 5.2.1功能选择界面



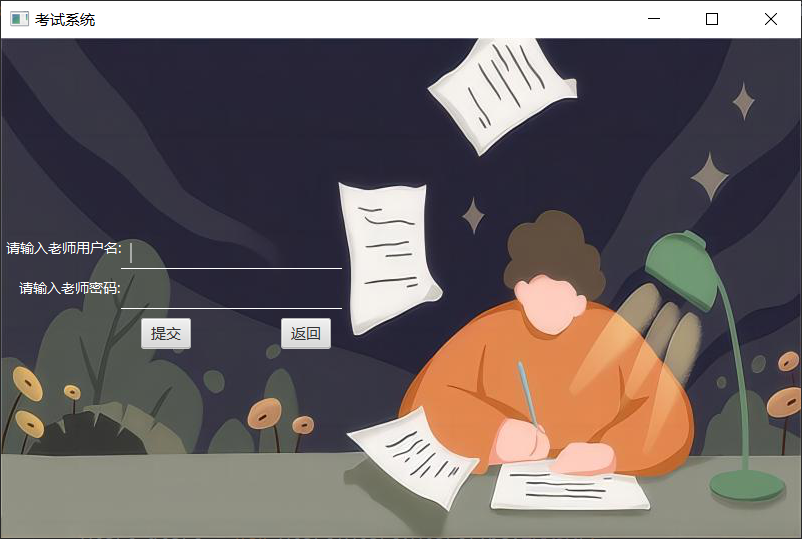
### 5.2.2添加题目界面



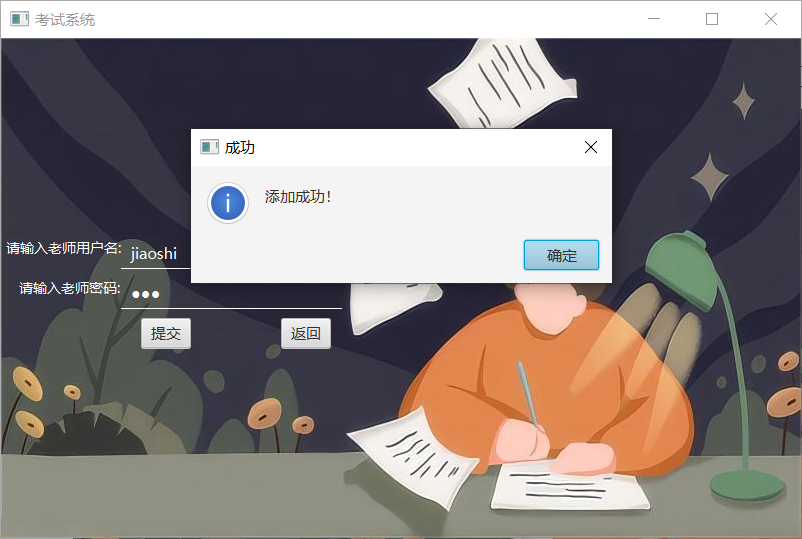
### 5.2.4添加成功界面



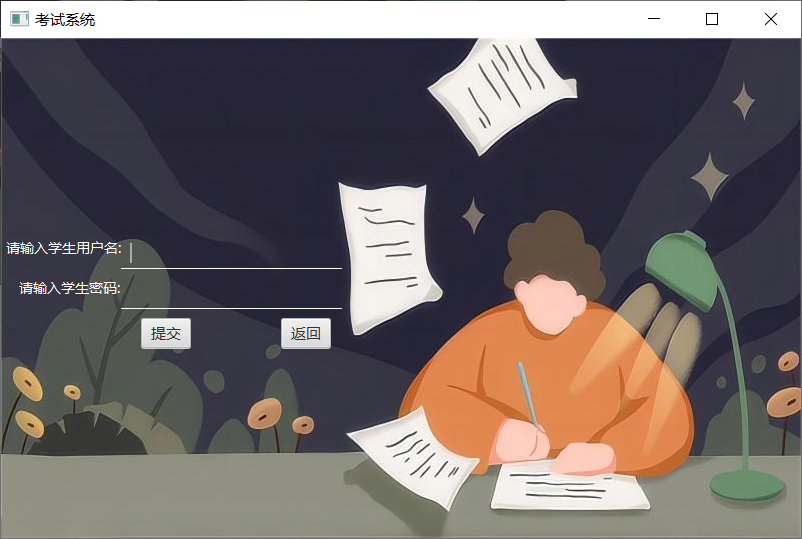
### 5.2.5添加教师界面



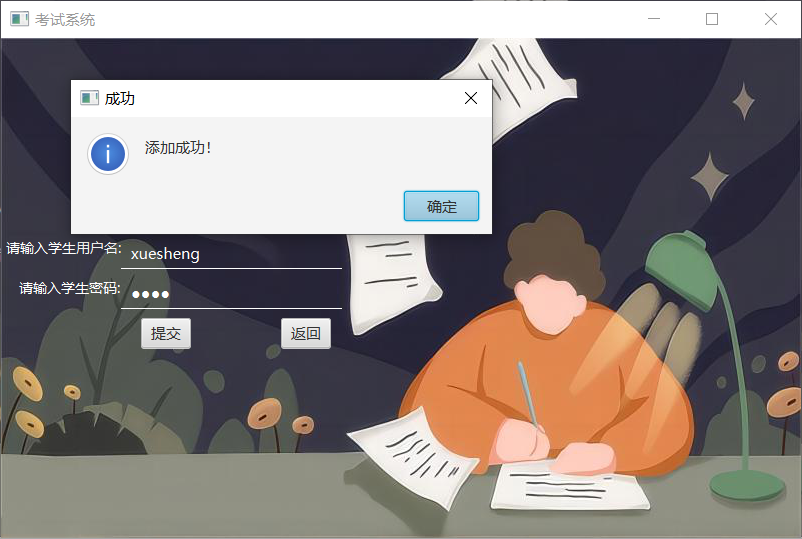
### 5.2.6添加成功界面



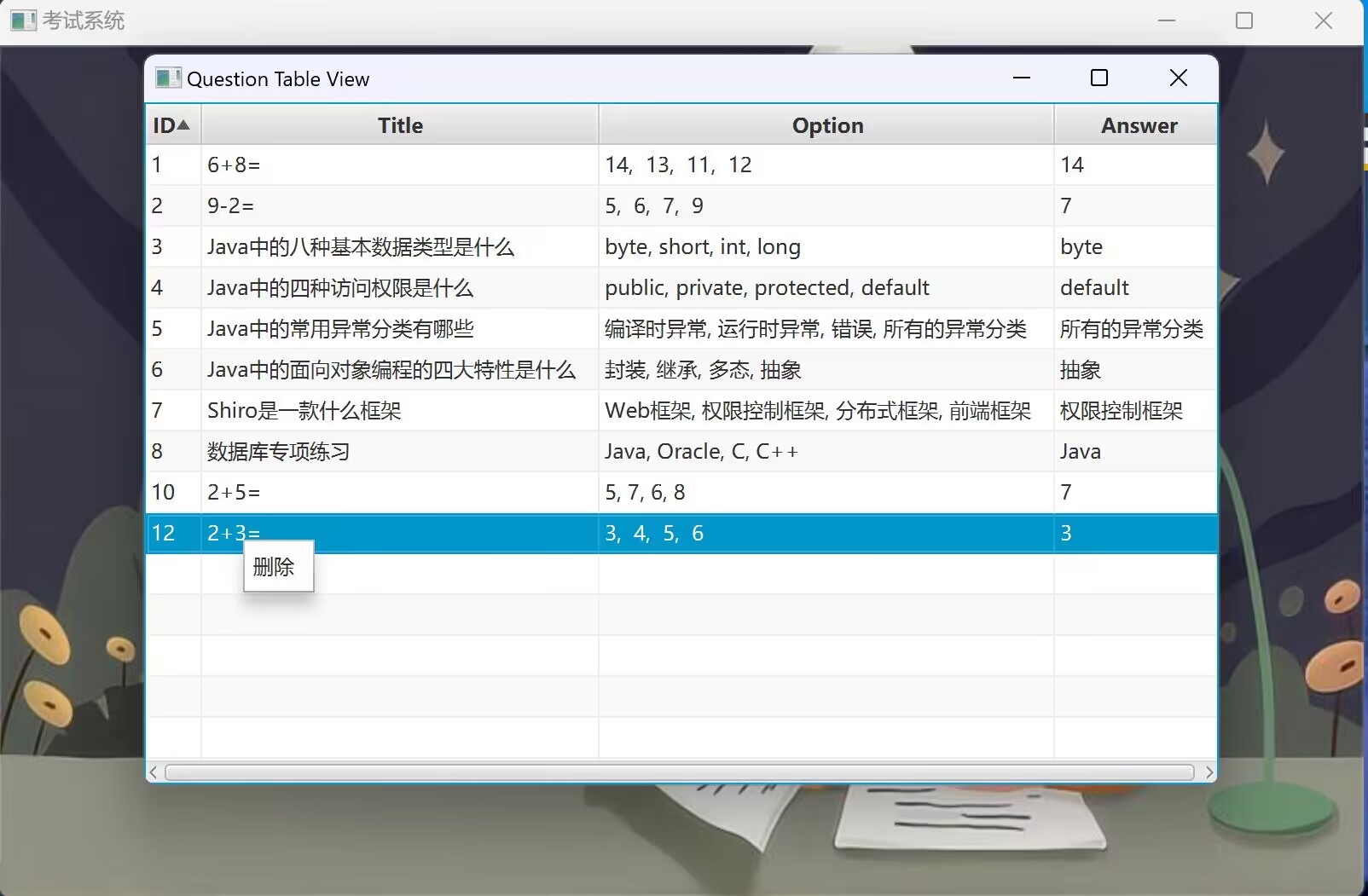
### 5.2.7添加学生界面



### 5.2.8添加成功界面



### 5.2.9题目修改界面



## 5.3源代码

由于代码过长，我们将在最后附录部分给出所有代码。

# 收获及体会

在构建这个考试系统的过程中，我着重关注了系统的模块化设计和用户交互体验。首先，我将不同功能划分到不同的类中，这有助于保持代码的清晰和可维护性。Teacher和Student类分别处理教师和学生的登录验证以及信息管理，而ExamMachine类则是系统的核心，负责试卷生成和考试流程管理。

数据管理方面，我采用了DAO设计模式，将数据库操作抽象到接口中，这样我能够灵活地切换不同的数据源或扩展系统的功能。通过文件操作，我实现了从文本文件中读取题库信息的功能，并运用随机化算法来生成随机试卷，以确保试卷的多样性和公平性。

用户交互方面，我充分利用了JavaFX提供的各种组件，比如按钮、输入框和弹窗，使得系统的界面友好且易于操作。我着重处理异常情况，通过合理的异常处理机制，例如使用try-catch块和弹窗提醒用户异常情况，确保系统的稳定性和用户体验。

整个系统结构清晰，代码可读性高，这有助于未来扩展和维护。通过这个系统，用户可以进行登录验证、添加题目和学生信息等操作，而系统还预留了提升功能和用户体验的空间，以便随着需求的变化不断完善。这个过程让我更深入地理解了面向对象设计的重要性，并提高了我的系统设计和开发能力。