目录

一、 概述

1.1 设计任务：

设计一个学生管理系统，实现以下功能：

1. 添加学生信息：包括姓名、性别、学号、课程和成绩。
2. 删除学生信息：根据学号删除学生信息。
3. 修改学生信息：根据学号修改学生的课程和成绩。
4. 按学科成绩排名：根据指定的学科成绩对学生进行排名。
5. 导出学生数据：将学生信息导出到 Excel 文件。
6. 在日志中记录使用者进行的操作
7. 在 GUI 中实现操作的回滚
8. 数据库的定时和手动备份以及恢复功能

1.2 涉及到的知识点

1.2.1 图形界面:

JavaFX 是一个用于构建客户端图形应用程序的开发工具包。它是 Java 平台的一部分，从 Java 8 开始成为 Java SE 的一部分。JavaFX 提供了丰富的图形化用户界面（GUI）组件和功能，使开发人员能够创建具有丰富交互和视觉效果的应用程序。

FXML（FXML Markup Language）是 JavaFX 的一种声明性 XML 格式，用于描述用户界面的结构和外观。它允许开发人员将用户界面的布局和样式与 Java 代码分离，以提高应用程序的可维护性和可重用性。FXML 文件描述了应用程序的场景图（Scene Graph），定义了界面的组件和它们的属性，以及它们之间的层次关系。FXML 文件使用标签和属性来描述用户界面的组件，例如按钮、标签、文本框等。开发人员可以使用 CSS（层叠样式表）来定义组件的外观和样式。FXML 还支持事件处理、绑定属性和资源束（Resource Bundle）等功能，使开发人员能够轻松地处理用户交互和应用程序逻辑。

JavaFX 应用程序使用 FXML 文件作为界面的模板，并使用 Java 代码将界面与应用程序逻辑连接起来。通过使用控制器类，开发人员可以在 FXML 文件中定义的组件上执行操作和事件处理。JavaFX 提供了丰富的 API 来操作和控制界面组件，以及处理用户输入和应用程序状态的变化。

1.2.2 后端：
SQLite3 是一种轻量级的嵌入式关系型数据库管理系统（RDBMS）。它是在公共领域中发布的开源软件，可以在许多操作系统上运行，包括 Windows、MacOS、Linux 和移动设备平台等。

SQLite3 的主要特点包括：

1. 嵌入式数据库：SQLite3 是一个嵌入式数据库，意味着它的数据库引擎可以直接嵌入到应用程序中，不需要单独的数据库服务器。这使得 SQLite3 成为许多移动应用、桌面应用和嵌入式系统的理想选择。
2. 轻量级：SQLite3 的设计目标之一是保持简单和轻量级。它的核心库非常小巧，只需要很少的资源即可运行。这使得 SQLite3 在资源受限的环境中表现出色，并且具有较快的启动和执行速度。
3. 零配置：与其他数据库管理系统不同，SQLite3 不需要复杂的配置或服务器设置。它使用单个文件作为数据库存储，并且可以直接在应用程序中创建和使用数据库。这使得它非常易于部署和管理。
4. 支持标准 SQL：SQLite3 支持标准的 SQL 查询语言，包括常见的 SQL 操作、聚合函数、子查询和连接等。它还支持事务处理，允许你在一系列数据库操作中保持一致性和完整性。
5. ACID 兼容：SQLite3 支持 ACID（原子性、一致性、隔离性和持久性）属性，确保数据库操作的可靠性和数据的完整性。它使用写日志和回滚日志来提供事务支持，并在发生故障时恢复数据库状态。
6. 多语言支持：SQLite3 提供了对多种编程语言的绑定，包括但不限于 C/C++、Java、Python、Ruby 和 .NET 等。这意味着你可以使用你喜欢的编程语言来与 SQLite3 数据库进行交互。

使用 SQL 语法进行数据库的 CRUD 操作(Create Read Update Delete)，SQL（Structured Query Language，结构化查询语言）是一种用于管理和操作关系型数据库的编程语言。它是由美国国家标准学会（ANSI）和国际标准化组织（ISO）标准化的，成为了关系型数据库管理系统（RDBMS）的通用语言。

SQL 提供了一组用于执行各种数据库操作的命令和语句，包括以下主要功能：

1. 数据定义语言（Data Definition Language，DDL）：DDL 用于创建、修改和删除数据库对象，如表、视图、索引等。常见的 DDL 命令包括 CREATE、ALTER 和 DROP。
2. 数据操作语言（Data Manipulation Language，DML）：DML 用于对数据库中的数据进行操作，包括插入、更新和删除数据。常见的 DML 命令包括 SELECT、INSERT、UPDATE 和 DELETE。

3. 数据查询语言（Data Query Language，DQL）：DQL 用于从数据库中检索数据，根据指定的条件过滤和排序结果。DQL 最常用的命令是 SELECT，它可以用于单个表或多个表之间的联接查询。
4. 数据控制语言（Data Control Language，DCL）：DCL 用于定义数据库的安全性和权限控制，包括授权用户的访问权限和撤销权限等。常见的 DCL 命令包括 GRANT 和 REVOKE。

SQL 是一种声明性语言，它允许开发人员通过编写简洁的语句来描述所需的操作，而无需关注底层的实现细节。SQL 的语法通常使用关键字、函数和运算符来表达查询条件和操作逻辑。

SQL 被广泛应用于各种数据库管理系统，包括常见的商业数据库（如 Oracle、MySQL、Microsoft SQL Server）以及开源数据库（如 SQLite、PostgreSQL）。它提供了灵活和强大的功能，使得开发人员可以高效地管理和操作大量结构化数据。无论是数据检索、数据更新还是数据分析，SQL 都是关系型数据库领域的重要工具之一。

Java annotation: Java 注解（Annotation）是一种元数据（Metadata）机制，它允许开发人员在代码中添加额外的信息，以提供给编译器、解释器或其他工具使用。注解可以应用于类、方法、字段和其他程序元素，以帮助开发人员提供关于程序结构、行为和配置的附加信息。

Java 注解的特点和用途如下：

1. 元数据信息：注解本身不会直接影响程序的执行逻辑，而是提供与程序元素相关的额外信息。例如，可以使用注解来标记特定的类、方法或字段，以指示其用途、约束或配置选项。
2. 编译时处理：Java 注解可以用于在编译时对代码进行静态检查和处理。编译器可以根据注解的信息执行特定的操作，例如生成额外的代码、执行代码验证或进行代码优化。
3. 运行时处理：某些注解可以在程序运行时被解释器或其他工具使用。这使得开发人员能够根据注解提供的信息动态地配置程序行为，例如通过反射机制来识别和处理注解。
4. 自定义扩展：Java 允许开发人员定义自己的自定义注解。这使得开发人员可以根据特定需求创建自定义注解，并在程序中使用它们来传达特定的意图和约束。

Java 常用数据结构：Map，HashMap，Stack

Java 多线程技术：以避免图形界面阻塞

Java 异常捕获知识: try/catch/finally，throws 等

Java 定时任务：ScheduledExecutorService

Java lambda 匿名函数：本次实验中主要用于多线程中

xerial.sqlite.jdbc: sqlite3 数据库驱动

Apache POI 库：实现对 Excel 的操作

Jackson json 库：实现对 json 对象的序列化与反序列化

Log4j2 日志库：记录有用的日志信息，用于后期分析

DAO(Data Access Object) 设计模式：隔离业务逻辑与数据库操作

二、总体设计方案

2.1 模块划分

src: 资源目录
程序采用 DAO 设计模式，将源代码划分在了几个文件夹中，
controller：目录存放与 FXML 页面绑定的 Java 控制器。
model：目录存放学生的数据模型以及增删改等操作的代码实现
util：目录存放了对数据库增删查改(CRUD)，备份与还原，Excel 导出功能，生成与日期相关的文件名，获取程序根路径，GUI 中 Ctrl+Z 撤销，消息弹窗，更新饼图等操作的封装。

入口函数: Main.java
resources 目录下包括了程序的图标，FXML GUI 的设计文件，Log4j2 的配置文件等

```
src/
└── main
    ├── java
    │   ├── com
    │   │   └── bobmaster
    │   │       └── studentsystem
    │   │           ├── Config.java
    │   │           ├── Main.java
    │   │           ├── controller
    │   │           │   ├── AboutPage.java
    │   │           │   ├── RootLayout.java
    │   │           │   └── SettingPageLayout.java
    │   │           ├── model
```

```
|   |           |   ├── Student.java
|   |           |   └── StudentDAO.java
|   |           └── util
|   |               ├── DBUtil.java
|   |               ├── ExportToExcel.java
|   |               ├── GenerateRandomTime.java
|   |               ├── GetCWD.java
|   |               ├── Revoke.java
|   |               ├── ShowMessage.java
|   |               └── UpdatePieChart.java
|   └── module-info.java
└── resources
    ├── com
    |   └── bobmaster
    |       └── studentsystem
    |           ├── AboutPage.fxml
    |           ├── RootLayout.fxml
    |           ├── SettingPage.fxml
    |           ├── assets
    |           |   └── aboutPage.png
    |           └── style.css
    ├── images
    |   ├── icon.png
    |   └── setting.png
    └── log4j2.properties
```

lib: 第三方 maven 包
xerial.sqlite.jdbc： sqlite3 数据库驱动
fasterxml.jackson.core: json 序列化反序列化包

apache.poi: 主要用到了操作 Excel 的 XSSFWorkbook
apache.logging.log4j.core: 日志工具

其他路径：
backup: 数据库备份存储路径
target: 编译产物
config.json: 用户配置文件
Program.log: 程序日志
student.db: 学生管理系统数据库文件

2.2 整体框图

```
├─── backup
├─── lib
├─── src
|   └─── main
|       ├─── java
|       |   ├─── com
|       |   |   └─── bobmaster
|       |   |       └─── studentsystem
|       |   |           ├─── Config.java
|       |   |           ├─── Main.java
|       |   |           ├─── controller
|       |   |           |   ├─── AboutPage.java
|       |   |           |   ├─── RootLayout.java
|       |   |           |   └─── SettingPageLayout.java
|       |   |           ├─── model
|       |   |           |   ├─── Student.java
|       |   |           |   └─── StudentDAO.java
|       |   |           └─── util
|       |   |               ├─── DBUtil.java
```

```
|       |   |               ├─── ExportToExcel.java
|       |   |               ├─── GenerateRandomTime.java
|       |   |               ├─── GetCWD.java
|       |   |               ├─── Revoke.java
|       |   |               ├─── ShowMessage.java
|       |   |               └─── UpdatePieChart.java
|       |   └─── module-info.java
|       └─── resources
|           ├─── com
|           |   └─── bobmaster
|           |       └─── studentsystem
|           |           ├─── AboutPage.fxml
|           |           ├─── RootLayout.fxml
|           |           ├─── SettingPage.fxml
|           |           ├─── assets
|           |           |   └─── aboutPage.png
|           |           └─── style.css
|           ├─── images
|           |   ├─── icon.png
|           |   └─── setting.png
|           └─── log4j2.properties
├─── student.db
├─── Program.log
├─── config.json
└─── target
    ├─── classes
    |   ├─── com
    |   |   └─── bobmaster
```

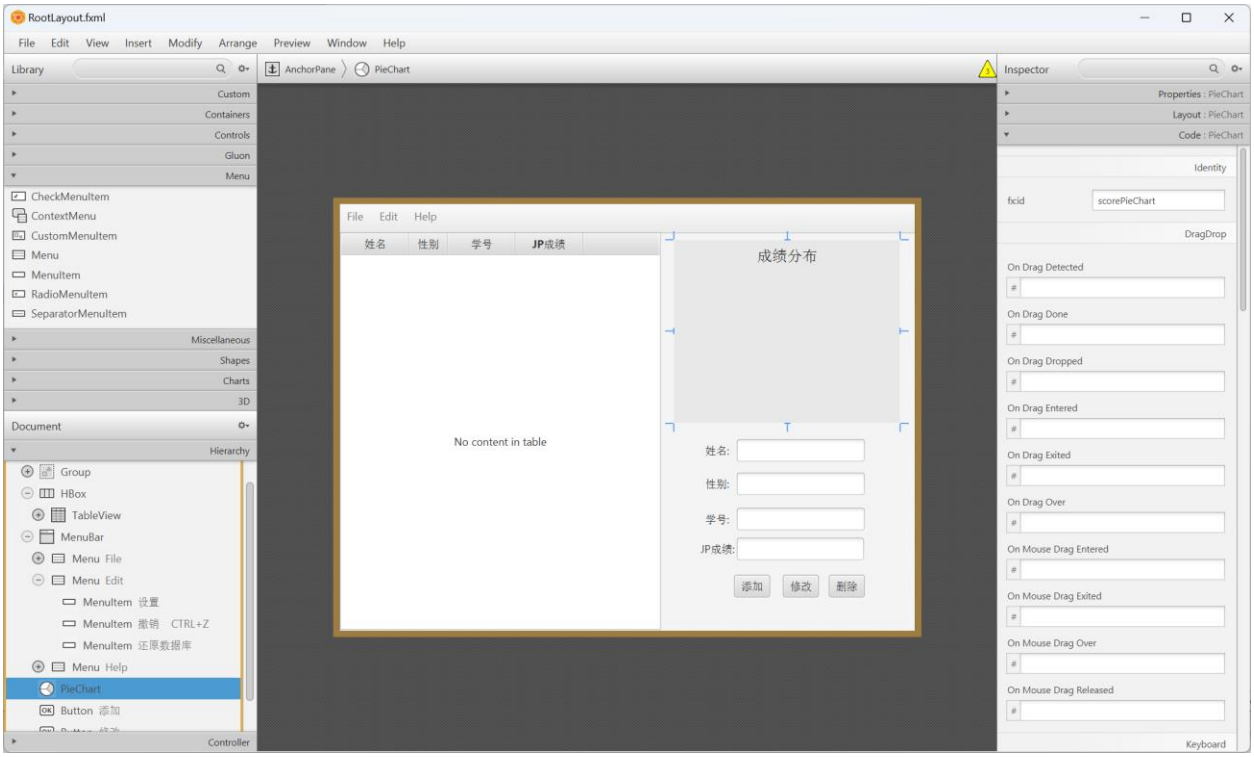
```
|   |       └── studentsystem
|   |           ├── AboutPage.fxml
|   |           ├── Config.class
|   |           ├── Main.class
|   |           ├── RootLayout.fxml
|   |           ├── SettingPage.fxml
|   |           ├── assets
|   |           |   └── aboutPage.png
|   |           ├── controller
|   |           |   ├── AboutPage.class
|   |           |   ├── RootLayout.class
|   |           |   └── SettingPageLayout.class
|   |           ├── model
|   |           |   ├── Student.class
|   |           |   └── StudentDAO.class
|   |           ├── style.css
|   |           └── util
|   |               ├── DBUtil.class
|   |               ├── ExportToExcel.class
|   |               ├── GenerateRandomTime.class
|   |               ├── GetCWD.class
|   |               ├── Revoke.class
|   |               ├── ShowMessage.class
|   |               └── UpdatePieChart.class
|   ├── images
|   |   ├── icon.png
|   |   └── setting.png
|   ├── log4j2.properties
```

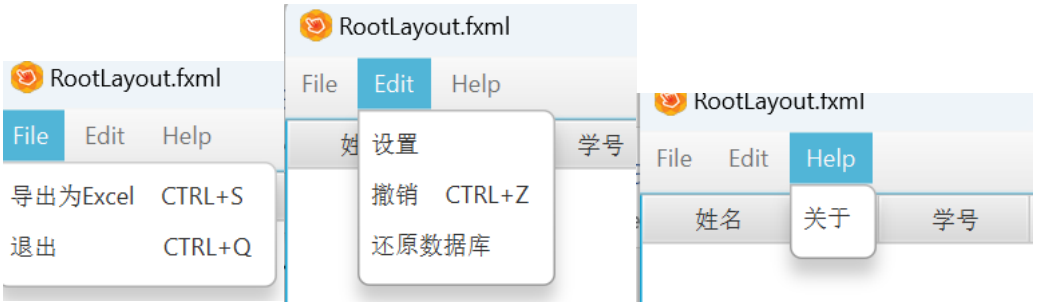
```
|   └──── module-info.class
└──── generated-sources
    └──── annotations
```

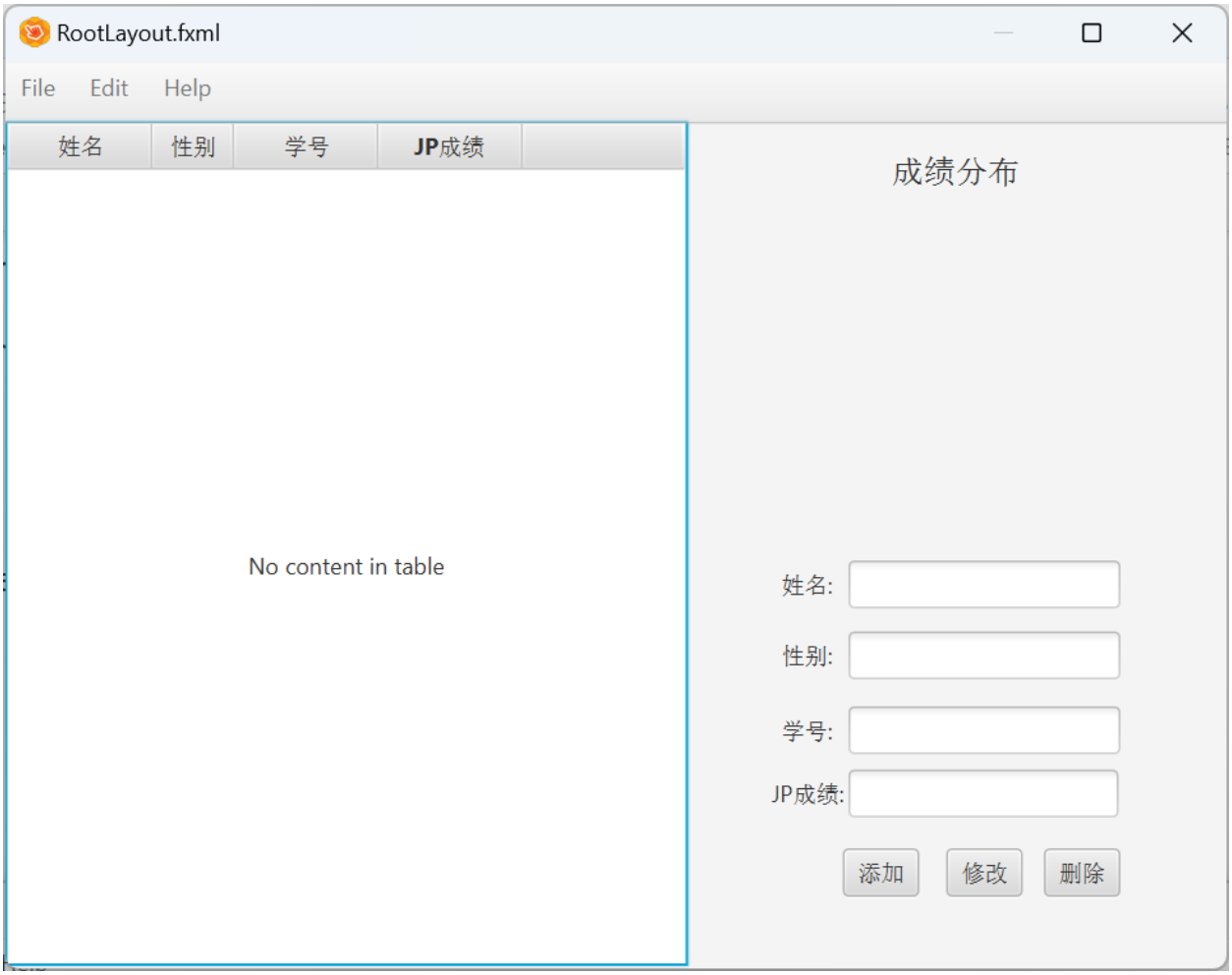
2.3 GUI 模块设计界面

图形界面主要采用 JavaFX Scene Builder 设计

根界面





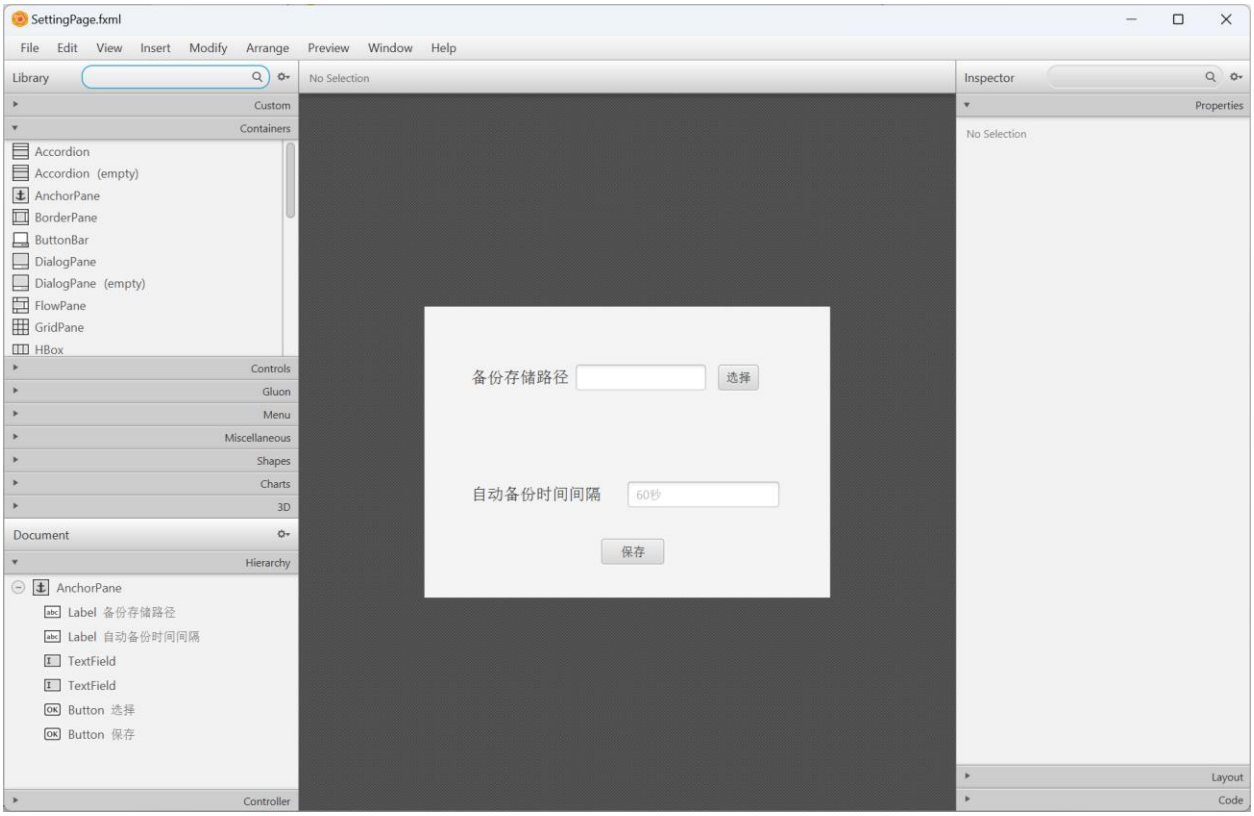

RootLayout.fxml
File
Edit
Help
姓名
性别
学号
JP成绩
No content in table
成绩分布
姓名:
性别:
学号:
JP成绩:
添加
修改
删除

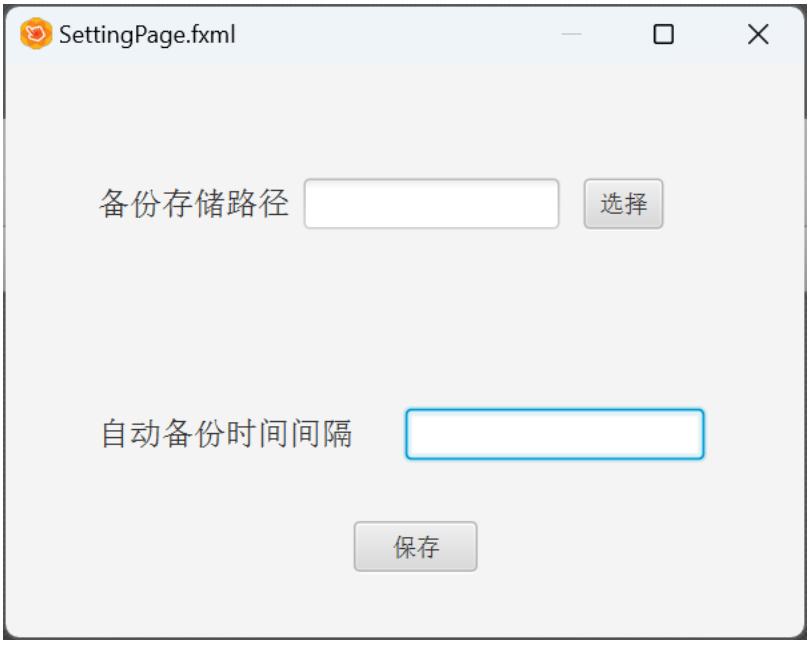
关于界面


AboutPage.fxml
File
Edit
View
Insert
Modify
Arrange
Preview
Window
Help
Library
No Selection
Inspector
Custom
Containers
Accordion
Accordion (empty)
AnchorPane
BorderPane
ButtonBar
DialogPane
DialogPane (empty)
FlowPane
Controls
Gluon
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
AnchorPane
ImageView
Label 学生成绩管理系统...
Label 版本: v0.0.1
Controller
Properties
Layout
Code
No Selection
学生成绩管理系统
Java课设作品
通信2102龚智勋
Powered by openjdk20.0.1+javaFX
版本: v0.0.1

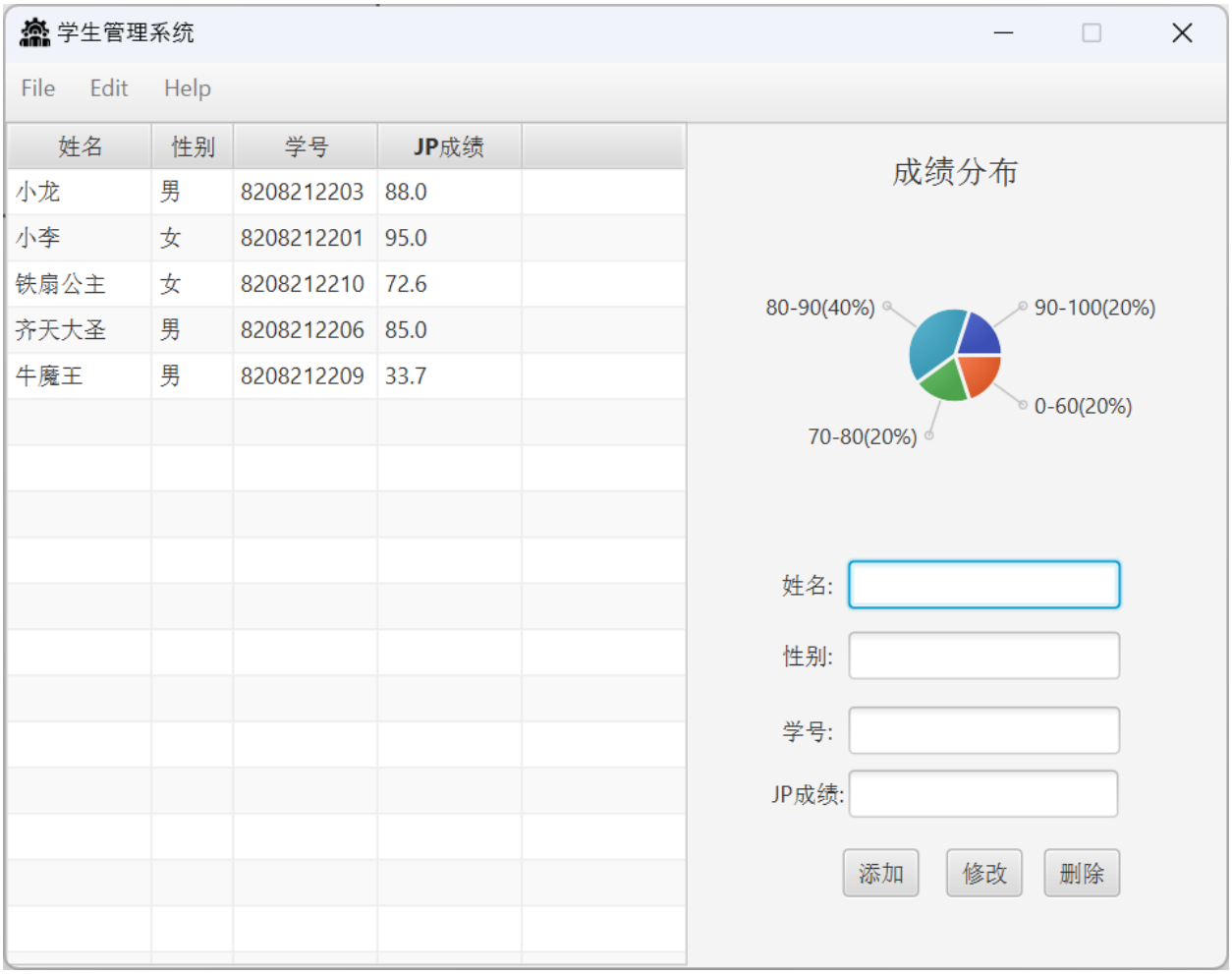

AboutPage.fxml
学生成绩管理系统
Java课设作品
通信2102龚智勋
Powered by openjdk20.0.1+javaFX
版本: v0.0.1

用户设置界面

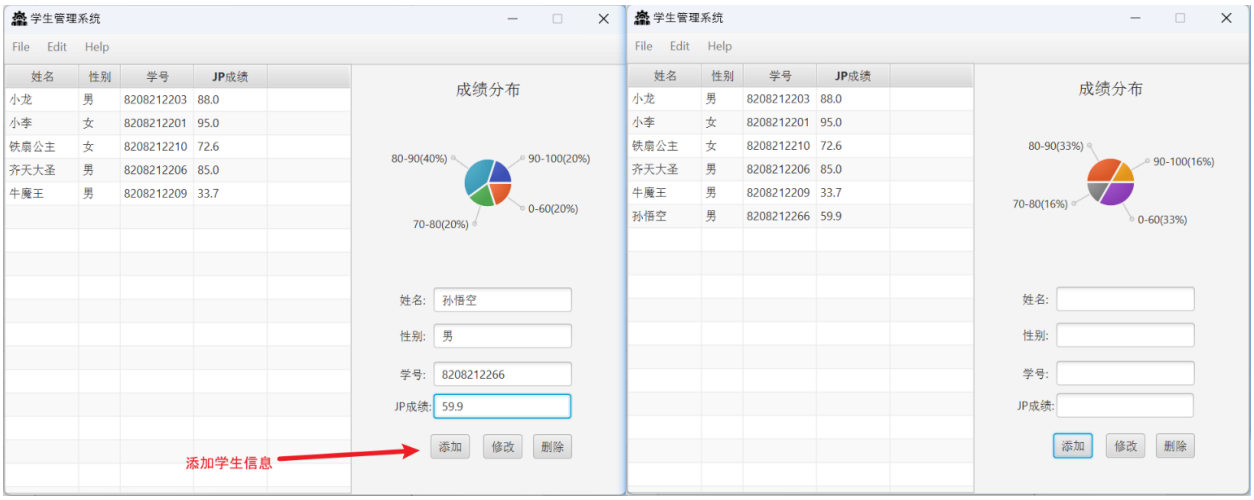

SettingPage.fxml
File Edit View Insert Modify Arrange Preview Window Help
Library
Custom
Containers
Accordion
Accordion (empty)
AnchorPane
BorderPane
ButtonBar
DialogPane
DialogPane (empty)
FlowPane
GridPane
HBox
Controls
Gluon
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
AnchorPane
Label 备份存储路径
Label 自动备份时间间隔
TextField
TextField
Button 选择
Button 保存
Controller
No Selection
备份存储路径
选择
自动备份时间间隔
60秒
保存
Inspector
Properties
No Selection
Layout
Code


SettingPage.fxml
备份存储路径
选择
自动备份时间间隔
保存

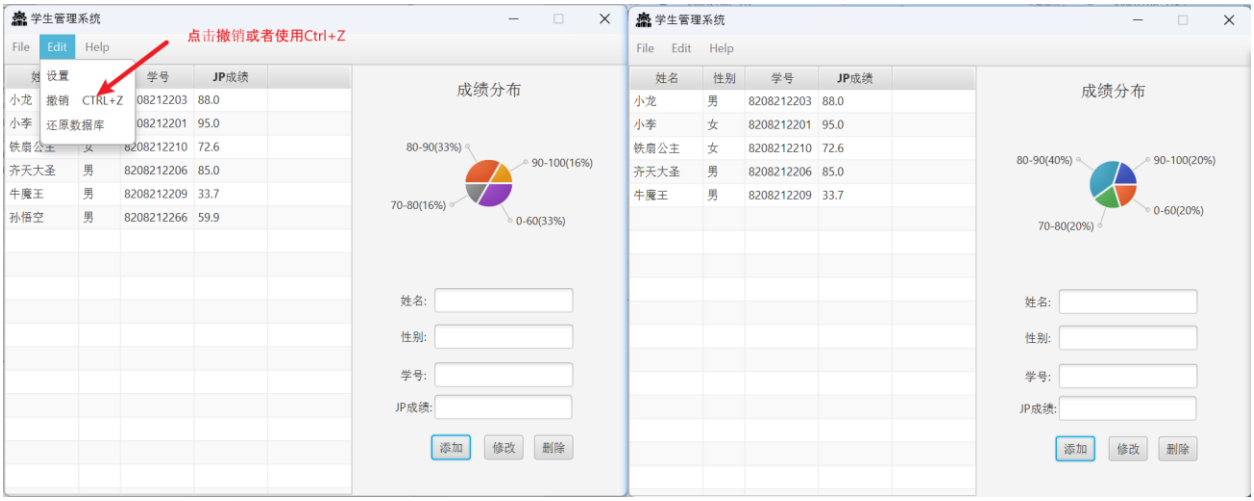
三、 成果展示



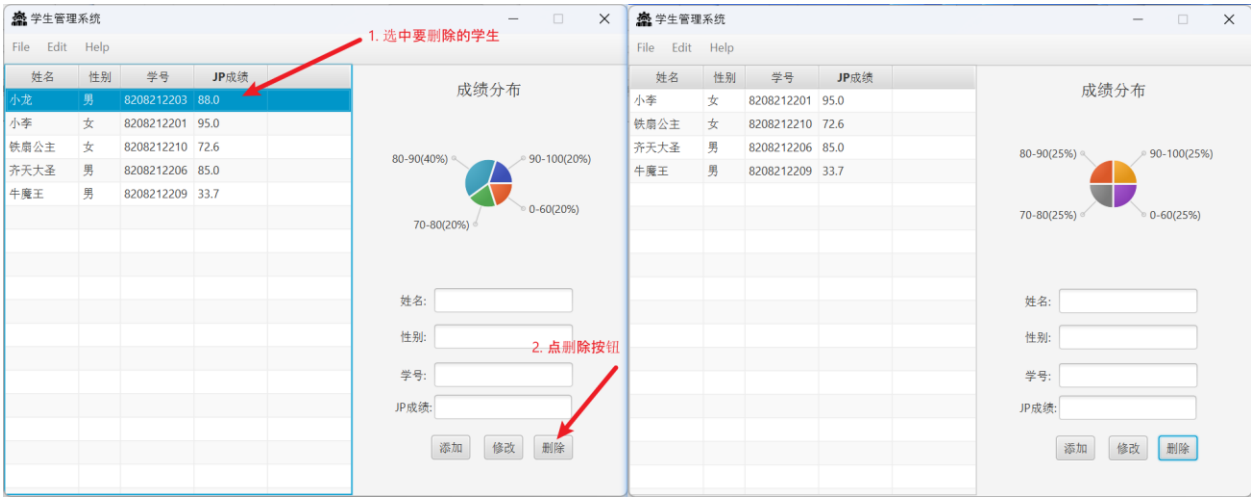
添加学生信息



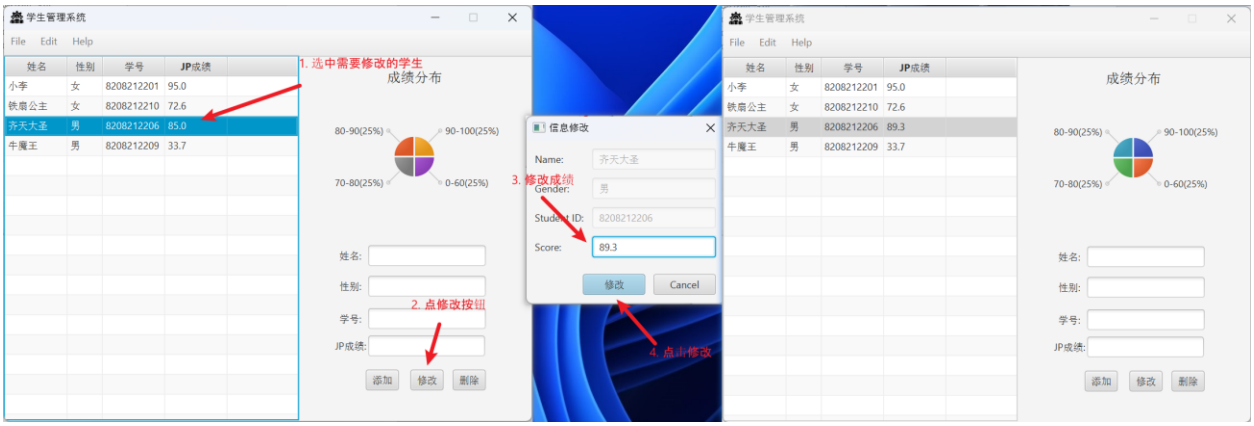
撤回更改



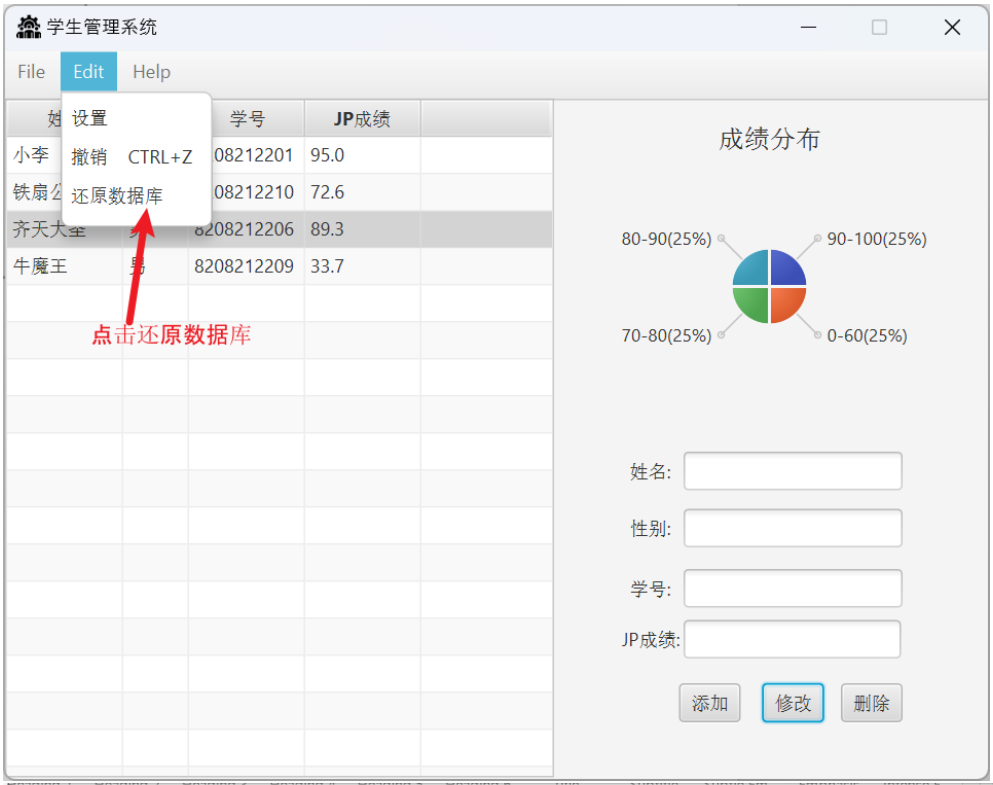
删除学生

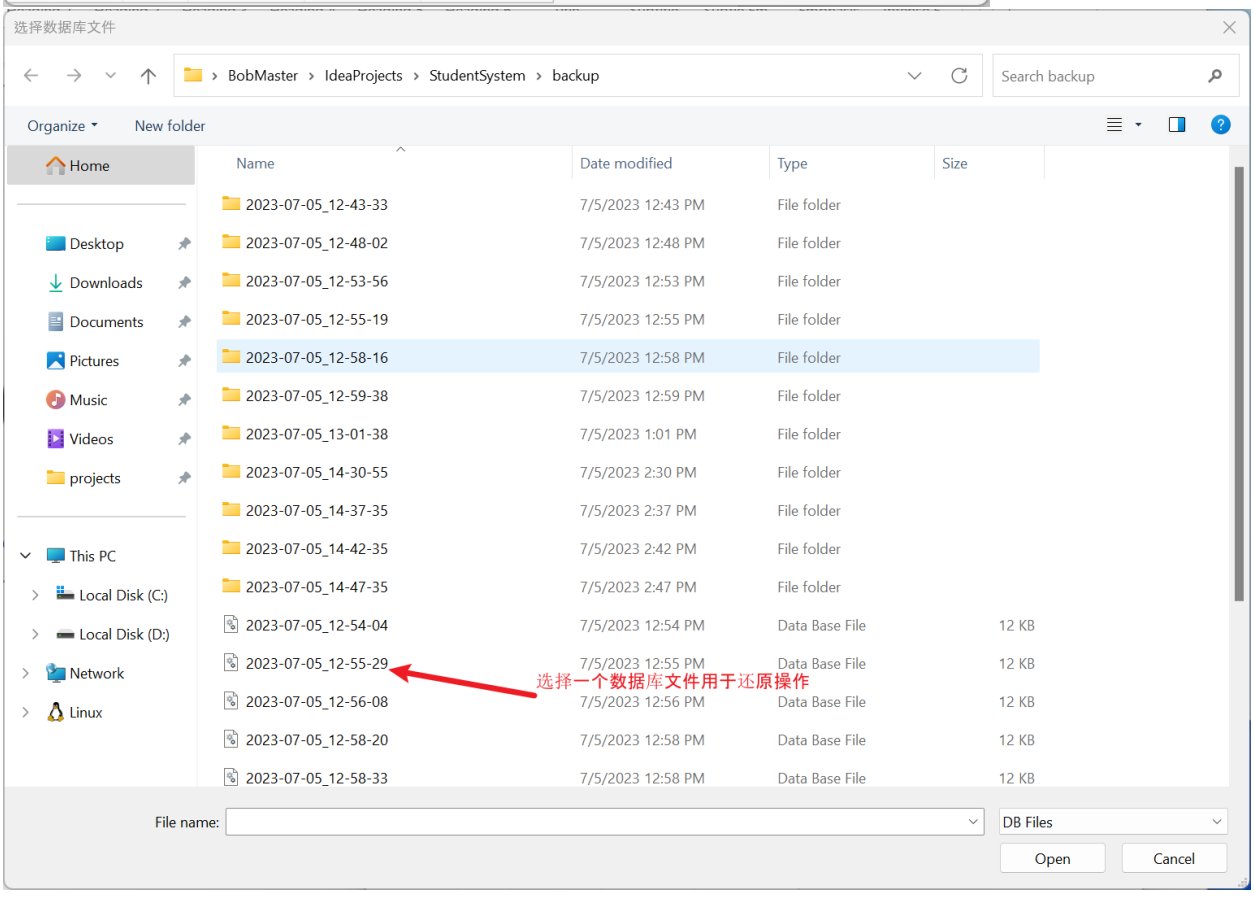


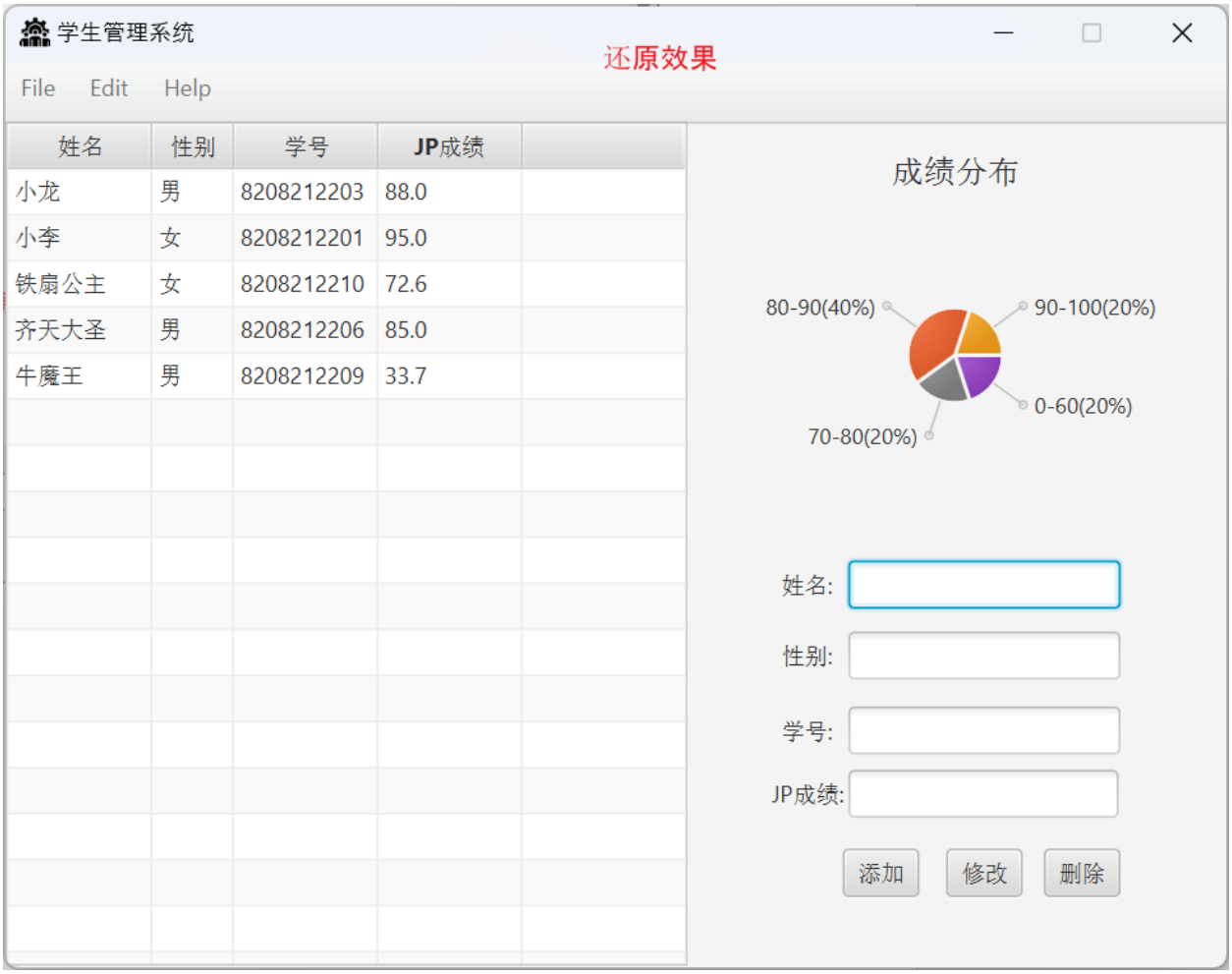
修改学生



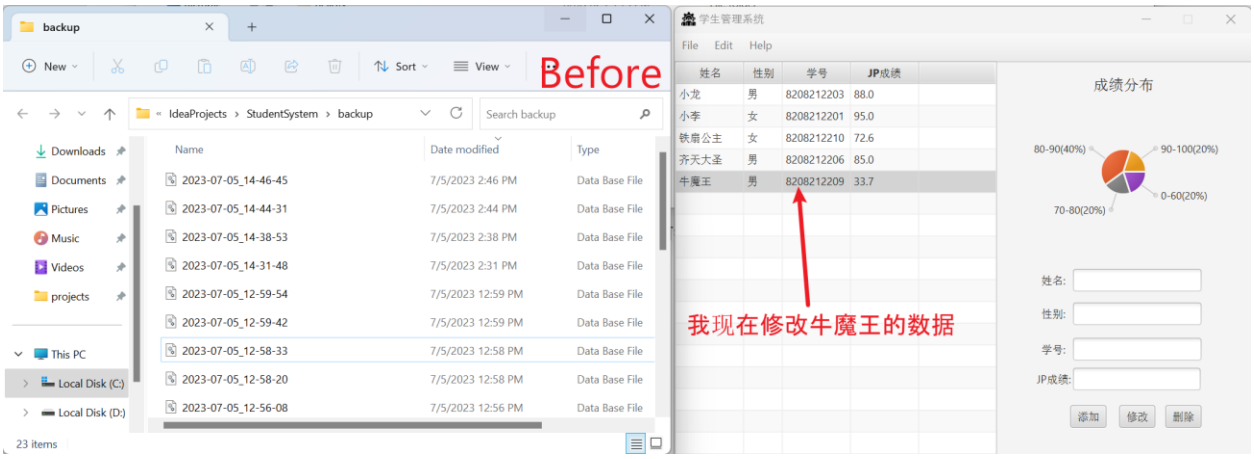
从备份恢复数据库

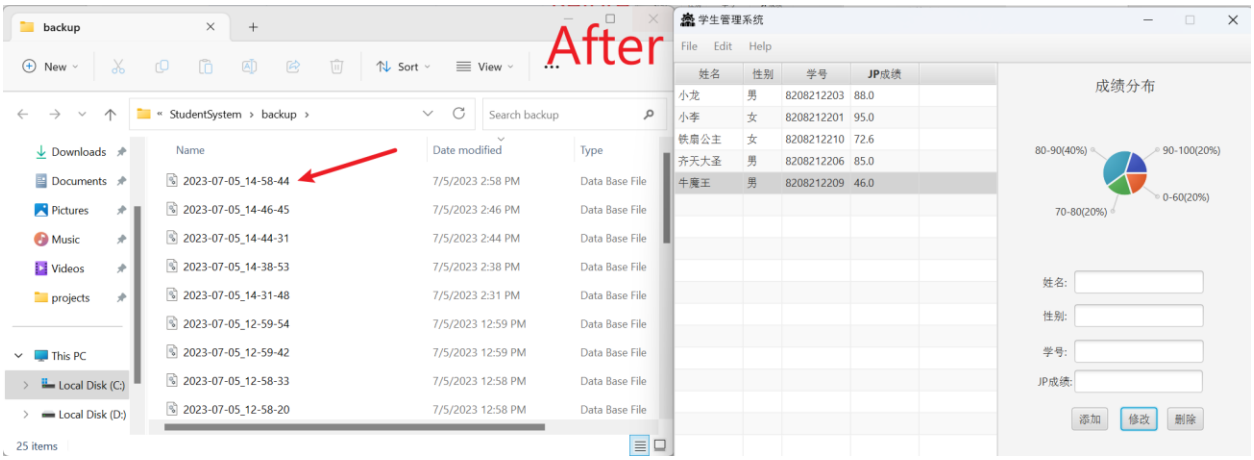

学生管理系统
File
Edit
Help
设置
撤销 CTRL+Z
还原数据库
学号
JP成绩
小李
08212201
95.0
铁扇公
08212210
72.6
齐天大圣
8208212206
89.3
牛魔王
男
8208212209
33.7
点击还原数据库
成绩分布
80-90(25%)
90-100(25%)
70-80(25%)
0-60(25%)
姓名:
性别:
学号:
JP成绩:
添加
修改
删除


选择数据库文件
BobMaster › IdeaProjects › StudentSystem › backup
Search backup
Organize
New folder
Home
Desktop
Downloads
Documents
Pictures
Music
Videos
projects
This PC
Local Disk (C:)
Local Disk (D:)
Network
Linux
Name
Date modified
Type
Size
2023-07-05_12-43-33 7/5/2023 12:43 PM File folder
2023-07-05_12-48-02 7/5/2023 12:48 PM File folder
2023-07-05_12-53-56 7/5/2023 12:53 PM File folder
2023-07-05_12-55-19 7/5/2023 12:55 PM File folder
2023-07-05_12-58-16 7/5/2023 12:58 PM File folder
2023-07-05_12-59-38 7/5/2023 12:59 PM File folder
2023-07-05_13-01-38 7/5/2023 1:01 PM File folder
2023-07-05_14-30-55 7/5/2023 2:30 PM File folder
2023-07-05_14-37-35 7/5/2023 2:37 PM File folder
2023-07-05_14-42-35 7/5/2023 2:42 PM File folder
2023-07-05_14-47-35 7/5/2023 2:47 PM File folder
2023-07-05_12-54-04 7/5/2023 12:54 PM Data Base File 12 KB
2023-07-05_12-55-29 7/5/2023 12:55 PM Data Base File 12 KB
2023-07-05_12-56-08 7/5/2023 12:56 PM Data Base File 12 KB
2023-07-05_12-58-20 7/5/2023 12:58 PM Data Base File 12 KB
2023-07-05_12-58-33 7/5/2023 12:58 PM Data Base File 12 KB
选择一个数据库文件用于还原操作
File name:
DB Files
Open
Cancel

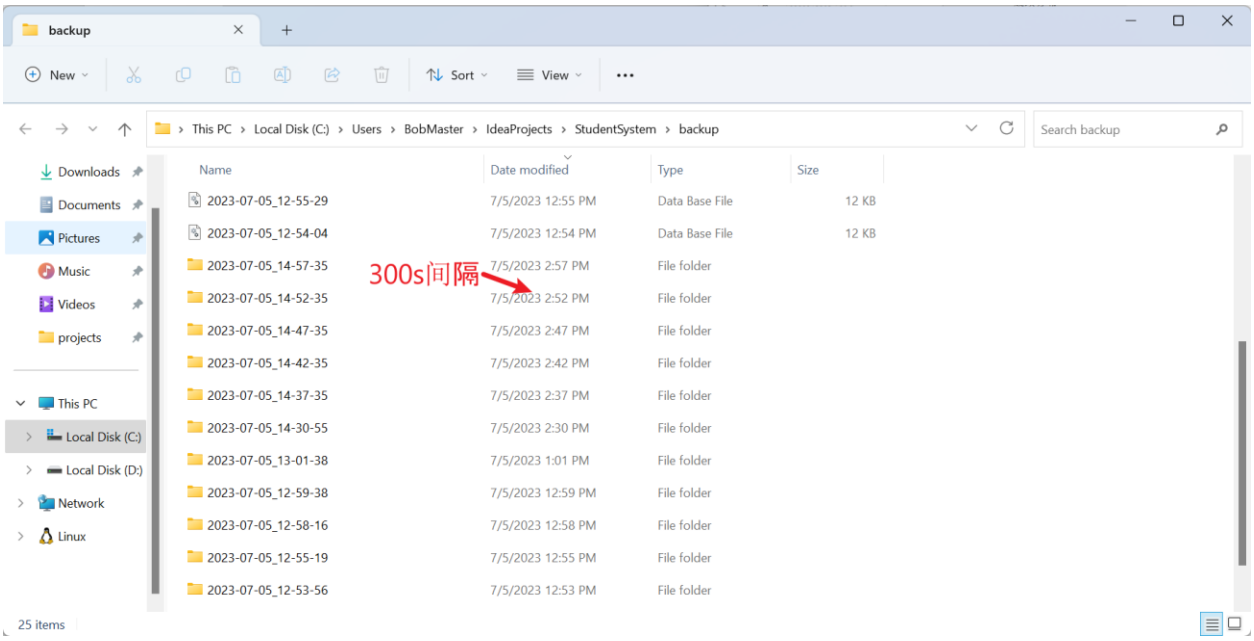


手动备份是用户进行任何一项添加，修改，删除操作之前都会进行一次备份

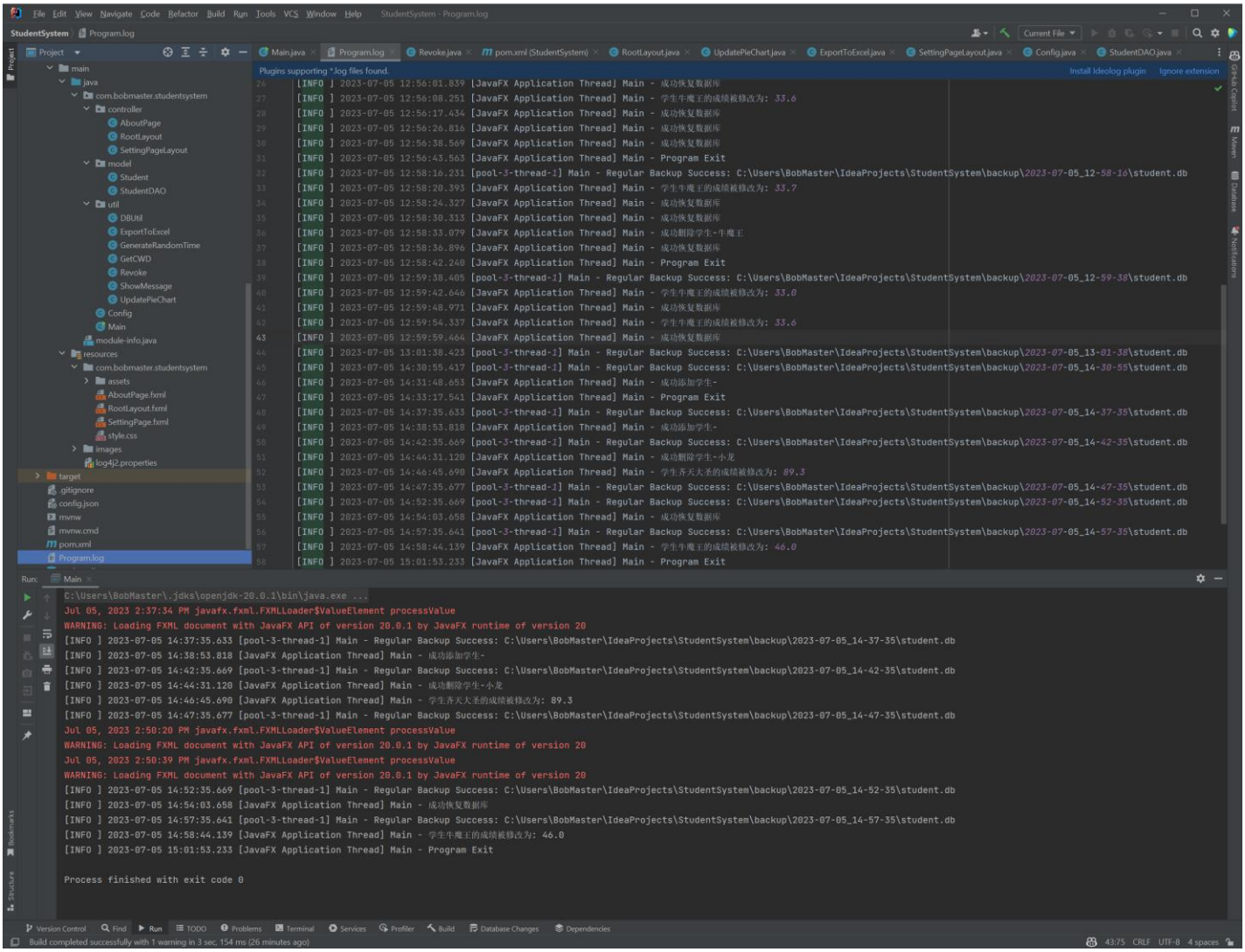




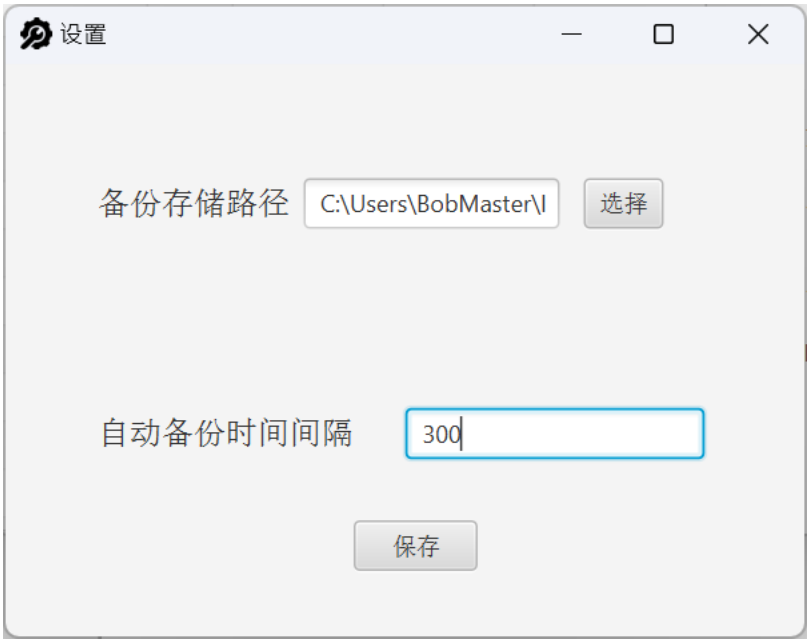
自动备份会根据用户设定的时间间隔进行自动备份，默认为 60 秒



日志记录功能


StudentSystem - Program.log
Program.log
Plugins supporting *.log files found.
Install Ideolog plugin
Ignore extension
[INFO] 2023-07-05 12:56:01.839 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 12:56:08.251 [JavaFX Application Thread] Main - 学生牛魔王的成绩被修改为: 33.6
[INFO] 2023-07-05 12:56:17.434 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 12:56:26.816 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 12:56:38.569 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 12:56:43.563 [JavaFX Application Thread] Main - Program Exit
[INFO] 2023-07-05 12:58:16.231 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_12-58-16\student.db
[INFO] 2023-07-05 12:58:20.393 [JavaFX Application Thread] Main - 学生牛魔王的成绩被修改为: 33.7
[INFO] 2023-07-05 12:58:24.327 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 12:58:30.313 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 12:58:33.079 [JavaFX Application Thread] Main - 成功删除学生-牛魔王
[INFO] 2023-07-05 12:58:36.896 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 12:58:42.240 [JavaFX Application Thread] Main - Program Exit
[INFO] 2023-07-05 12:59:38.405 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_12-59-38\student.db
[INFO] 2023-07-05 12:59:42.646 [JavaFX Application Thread] Main - 学生牛魔王的成绩被修改为: 33.0
[INFO] 2023-07-05 12:59:48.971 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 12:59:54.337 [JavaFX Application Thread] Main - 学生牛魔王的成绩被修改为: 33.6
[INFO] 2023-07-05 12:59:59.464 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 13:01:38.423 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_13-01-38\student.db
[INFO] 2023-07-05 14:30:55.417 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-30-55\student.db
[INFO] 2023-07-05 14:31:48.653 [JavaFX Application Thread] Main - 成功添加学生-
[INFO] 2023-07-05 14:33:17.541 [JavaFX Application Thread] Main - Program Exit
[INFO] 2023-07-05 14:37:35.633 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-37-35\student.db
[INFO] 2023-07-05 14:38:53.818 [JavaFX Application Thread] Main - 成功添加学生-
[INFO] 2023-07-05 14:42:35.669 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-42-35\student.db
[INFO] 2023-07-05 14:44:31.120 [JavaFX Application Thread] Main - 成功删除学生-小龙
[INFO] 2023-07-05 14:46:45.690 [JavaFX Application Thread] Main - 学生齐天大圣的成绩被修改为: 89.3
[INFO] 2023-07-05 14:47:35.677 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-47-35\student.db
[INFO] 2023-07-05 14:52:35.669 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-52-35\student.db
[INFO] 2023-07-05 14:54:03.658 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 14:57:35.641 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-57-35\student.db
[INFO] 2023-07-05 14:58:44.139 [JavaFX Application Thread] Main - 学生牛魔王的成绩被修改为: 46.0
[INFO] 2023-07-05 15:01:53.233 [JavaFX Application Thread] Main - Program Exit
Run: Main
C:\Users\BobMaster\.jdks\openjdk-20.0.1\bin\java.exe ...
Jul 05, 2023 2:37:34 PM javafx.fxml.FXMLLoader$ValueElement processValue
WARNING: Loading FXML document with JavaFX API of version 20.0.1 by JavaFX runtime of version 20
[INFO] 2023-07-05 14:37:35.633 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-37-35\student.db
[INFO] 2023-07-05 14:38:53.818 [JavaFX Application Thread] Main - 成功添加学生-
[INFO] 2023-07-05 14:42:35.669 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-42-35\student.db
[INFO] 2023-07-05 14:44:31.120 [JavaFX Application Thread] Main - 成功删除学生-小龙
[INFO] 2023-07-05 14:46:45.690 [JavaFX Application Thread] Main - 学生齐天大圣的成绩被修改为: 89.3
[INFO] 2023-07-05 14:47:35.677 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-47-35\student.db
Jul 05, 2023 2:50:20 PM javafx.fxml.FXMLLoader$ValueElement processValue
WARNING: Loading FXML document with JavaFX API of version 20.0.1 by JavaFX runtime of version 20
Jul 05, 2023 2:50:39 PM javafx.fxml.FXMLLoader$ValueElement processValue
WARNING: Loading FXML document with JavaFX API of version 20.0.1 by JavaFX runtime of version 20
[INFO] 2023-07-05 14:52:35.669 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-52-35\student.db
[INFO] 2023-07-05 14:54:03.658 [JavaFX Application Thread] Main - 成功恢复数据库
[INFO] 2023-07-05 14:57:35.641 [pool-3-thread-1] Main - Regular Backup Success: C:\Users\BobMaster\IdeaProjects\StudentSystem\backup\2023-07-05_14-57-35\student.db
[INFO] 2023-07-05 14:58:44.139 [JavaFX Application Thread] Main - 学生牛魔王的成绩被修改为: 46.0
[INFO] 2023-07-05 15:01:53.233 [JavaFX Application Thread] Main - Program Exit
Process finished with exit code 0
Build completed successfully with 1 warning in 3 sec, 154 ms (26 minutes ago)

设置界面


设置
备份存储路径
C:\Users\BobMaster\I
选择
自动备份时间间隔
300
保存

关于界面


关于
学生成绩管理系统
Java课设作品
通信2102龚智勋
Powered by openjdk20.0.1+javaFX
版本: v0.0.1

四、结语

本次 Java 课程设计，我学到了利用 JavaFX 实现跨平台的 GUI 程序开发，学会了 Log4j 这一日志库的基本使用，Java JSON 序列化和反序列化，数据库的增删查改等基本操作，多线程以及匿名函数的使用，此外对各个实例的生命周期有了更深了解。

五、参考文献

1. JavaFX Tutorial with All Details and Examples
https://www.swtestacademy.com/javafx-tutorial/

2. Database Operations in JavaFX with Complete Example
https://www.swtestacademy.com/database-operations-javafx/

3. Introduction to JavaFx
https://www.baeldung.com/javafx

4. JavaFX Tutorial - JavaFX PieChart
http://www.java2s.com/Tutorials/Java/JavaFX/0810__JavaFX_Pie_Chart.htm

5. JavaFX Documentation Project
https://fxdocs.github.io/docs/html5/

6. Intro to the Jackson ObjectMapper
https://www.baeldung.com/jackson-object-mapper-tutorial

7. Log4j2 – Logging to Both File and Console
https://www.baeldung.com/java-log4j2-file-and-console

六、附录-程序源代码

Main.java

```
package com.bobmaster.studentsystem;

import com.bobmaster.studentsystem.model.Student;
import com.bobmaster.studentsystem.util.DBUtil;
import com.bobmaster.studentsystem.util.ExportToExcel;
import com.bobmaster.studentsystem.util.GenerateRandomTime;
import com.bobmaster.studentsystem.util.Revoke;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.TableView;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import java.io.IOException;
import java.sql.SQLException;
import java.util.Map;
import java.util.Stack;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;

import static java.util.concurrent.TimeUnit.SECONDS;


public class Main extends Application {
    private Stage primaryStage;
    private AnchorPane rootLayout;
    private Scene scene;
    // DbPath stack
    public static Stack<String> dbPathStack = new Stack<>();

    public static final Logger logger = LogManager.getLogger(Main.class);

    @Override
    public void start(Stage primaryStage) throws IOException {
        this.primaryStage = primaryStage;
        this.primaryStage.setTitle("学生管理系统");
```

```
        this.primaryStage.getIcons().add(new
Image("file:src/main/resources/images/icon.png"));
        initRootLayout();

        Map <String, Object> map = (Map<String, Object>)
Config.LoadConfig(Map.class);

        // Create Schedule backup task
        ScheduledExecutorService scheduler = Executors.newScheduledThreadPool(1);
        scheduler.scheduleAtFixedRate(() -> {
            try {
                String dbPath = map.get("backupPath").toString() + "\\" +
                        GenerateRandomTime.Generate() + "\\student.db";
                DBUtil.dbBackup(dbPath);
                logger.info("Regular Backup Success: " + dbPath);
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        }, 0, Long.parseLong(map.get("backupInterval").toString()), SECONDS);
        // close scheduler when RootLayout exit
        primaryStage.setOnCloseRequest(event -> {
            scheduler.shutdown();
            logger.info("Program Exit");
        });
    }

    private void initRootLayout() {
        try {
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class.getResource("RootLayout.fxml"));
            rootLayout = loader.load();

            // keybinding
            initKeyBinding();

            scene = new Scene(rootLayout);
            primaryStage.setScene(scene);
            primaryStage.setResizable(false);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void initKeyBinding() {
```

```
        rootLayout.setOnKeyPressed(event -> {
            // exit when press Ctrl+Q
            if (event.isControlDown() && event.getCode().equals(KeyCode.Q)) {
                try {
                    DBUtil.dbDisconnect();
                } catch (SQLException e) {
                    throw new RuntimeException(e);
                }
                logger.info("Program Exit");
                System.exit(0);
            }

            // export to excel when press Ctrl+S
            if (event.isControlDown() && event.getCode().equals(KeyCode.S)) {
                ExportToExcel.exportToExcel();
            }

            // revoke when press Ctrl+Z
            if (event.isControlDown() && event.getCode().equals(KeyCode.Z)) {
                try {
                    TableView<Student> node = (TableView<Student>)
scene.lookup("#studentTable");
                    Revoke.revokeEdit(dbPathStack, node);
                } catch (SQLException | ClassCastException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        });
    }


    public static void main(String[] args) throws SQLException{
        DBUtil.dbInit();
        launch(args);
    }
}
```

Config.java

```
package com.bobmaster.studentsystem;
import com.bobmaster.studentsystem.util.GetCWD;
import com.fasterxml.jackson.databind.ObjectMapper;
```

```java
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Config {
    public static void SaveConfig(Object object) throws IOException {
        ObjectMapper objectMapper = new ObjectMapper();
        objectMapper.writeValue(new File(GetCWD.getCWD() + "\\config.json"),
object);
    }

    public static Object LoadConfig(Class<?> clazz) throws IOException {
        File file = new File(GetCWD.getCWD() + "\\config.json");
        if (!file.exists()) {
            Map<String, Object> map = new HashMap<>();
            map.put("backupPath", GetCWD.getCWD() + "\\backup");
            map.put("backupInterval", 60);
            SaveConfig(map);
        }
        ObjectMapper objectMapper = new ObjectMapper();
        return objectMapper.readValue(file, clazz);
    }


}
```

controller

AboutPage.java

```java
package com.bobmaster.studentsystem.controller;

import javafx.fxml.FXML;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;

public class AboutPage {
    @FXML
    public AnchorPane aboutPagePane;

    public void handleHideAboutPage(MouseEvent mouseEvent) {
        aboutPagePane.getScene().getWindow().hide();
    }
```

```
}
```

RootLayout.java

```
package com.bobmaster.studentsystem.controller;

import com.bobmaster.studentsystem.Main;
import com.bobmaster.studentsystem.model.Student;
import com.bobmaster.studentsystem.model.StudentDAO;
import com.bobmaster.studentsystem.util.*;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.chart.PieChart;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.GridPane;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.io.IOException;
import java.sql.SQLException;

public class RootLayout {
    @FXML
    public TextField nameText;
    @FXML
    public TextField idText;
    @FXML
    public TextField genderText;
    @FXML
    public TextField scoreText;
    @FXML
    public TableView<Student> studentTable;
    @FXML
    public TableColumn<Student, String> stuNameColumn;
    @FXML
    public TableColumn<Student, String> stuIdColumn;
```

```
    @FXML
    public TableColumn<Student, String> stuGenderColumn;
    @FXML
    public TableColumn<Student, Double> stuScoreColumn;
    @FXML
    public PieChart scorePieChart;

    public void handleAboutPage(ActionEvent actionEvent) {
        try {
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class.getResource("AboutPage.fxml"));
            AnchorPane root = loader.load();
            Stage stage = new Stage();
            stage.setTitle("关于");
            stage.getIcons().add(new
Image("file:src/main/resources/images/icon.png"));
            stage.setScene(new Scene(root));
            stage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void handleSettingPage(ActionEvent actionEvent) {
        try {
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class.getResource("SettingPage.fxml"));
            AnchorPane root = loader.load();
            Stage stage = new Stage();
            stage.setTitle("设置");
            stage.getIcons().add(new
Image("file:src/main/resources/images/setting.png"));
            stage.setScene(new Scene(root));
            stage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void handleExit(ActionEvent actionEvent) {
        Main.logger.info("Program Exit");
        System.exit(0);
    }

    public void handleExportToExcel(ActionEvent actionEvent) {
```

```
        ExportToExcel.exportToExcel();
    }

    // initialize the controller class
    @FXML
    public void initialize() {
        // initialize the student table with the four columns
        stuNameColumn.setCellValueFactory(cellData ->
cellData.getValue().studentNameProperty());
        stuIdColumn.setCellValueFactory(cellData ->
cellData.getValue().studentIdProperty());
        stuGenderColumn.setCellValueFactory(cellData ->
cellData.getValue().studentGenderProperty());
        stuScoreColumn.setCellValueFactory(cellData ->
cellData.getValue().studentScoreProperty().asObject());
        try {
            ObservableList<Student> studentData = StudentDAO.searchStudents();
            populateStudents(studentData);
            // PieChart
            UpdatePieChart.updatePieChart(scorePieChart);
        } catch (SQLException | IOException e) {
            e.printStackTrace();
        }

    }

    // populate the student table with the data
    private void populateStudents(ObservableList<Student> studentData) {
        studentTable.setItems(studentData);
    }

    // Insert a student
    @FXML
    public void handleAddStudent(MouseEvent mouseEvent) throws SQLException{
        // if all the text fields are not empty
        if (!nameText.getText().isEmpty() && !idText.getText().isEmpty()
                && !genderText.getText().isEmpty()
&& !scoreText.getText().isEmpty()) {
            // check if the student Score is a number
            try {
                Double.parseDouble(scoreText.getText());
                // if the student id is not in the database
                if (StudentDAO.searchStudent(idText.getText()) == null) {
                    try {
                        // backup the database
```

```
                        String DbPath = GetCWD.getCWD() + "\\backup\\" +
GenerateRandomTime.Generate() + ".db";
                        Main.dbPathStack.push(DbPath);
                        DBUtil.dbBackup(DbPath);
                        StudentDAO.insertStudent(nameText.getText(),
                                genderText.getText(),
                                idText.getText(),
                                Double.parseDouble(scoreText.getText()));
                        // clear the text fields
                        nameText.clear();
                        genderText.clear();
                        idText.clear();
                        scoreText.clear();
                        Main.logger.info("成功添加学生-" + nameText.getText());
                        ShowMessage.showAlert(Alert.AlertType.INFORMATION, "成功
", "添加学生数据成功! ");
                        // PieChart
                        UpdatePieChart.updatePieChart(scorePieChart);
                    } catch (SQLException | IOException e) {
                        Main.logger.error("添加学生数据失败! ");
                        ShowMessage.showErrorAlert("错误", "添加学生数据失败! ");
                        e.printStackTrace();
                    }

                    ObservableList<Student> studentData =
StudentDAO.searchStudents();
                    populateStudents(studentData);
                } else {
                    // if the student id is in the database
                    ShowMessage.showErrorAlert("错误", "该学号已存在! ");
                }
            } catch (NumberFormatException e) {
                ShowMessage.showErrorAlert("错误", "成绩必须为数字! ");
            }
        } else {
            // if any of the text fields is empty
            ShowMessage.showErrorAlert("错误", "请填写完整! ");
        }
    }

    // Delete a student
    @FXML
    public void handleDeleteStudent(MouseEvent mouseEvent) {
        Student selectedStudent =
studentTable.getSelectionModel().getSelectedItem();
```

```
        if (selectedStudent != null) {
            try {
                // backup the database
                String DbPath = GetCWD.getCWD() + "\\backup\\" +
GenerateRandomTime.Generate() + ".db";
                Main.dbPathStack.push(DbPath);
                DBUtil.dbBackup(DbPath);
                StudentDAO.deleteStudent(selectedStudent.getStudentId());
                Main.logger.info("成功删除学生-" + selectedStudent.getName());
                ShowMessage.showAlert(Alert.AlertType.INFORMATION, "成功", "学生"
+
                        selectedStudent.getName() + "的数据删除成功！");
                // PieChart
                UpdatePieChart.updatePieChart(scorePieChart);
            } catch (SQLException | IOException e) {
                Main.logger.error("删除学生数据失败！");
                ShowMessage.showErrorAlert("错误", "删除学生数据失败！");
                e.printStackTrace();
            }

            try {
                ObservableList<Student> studentData =
StudentDAO.searchStudents();
                populateStudents(studentData);
            } catch (SQLException e) {
                ShowMessage.showErrorAlert("错误", e.getMessage());
                e.printStackTrace();
            }

        } else {
            ShowMessage.showErrorAlert("错误", "请选择要删除的学生！");
        }
    }

    // Update a student
    @FXML
    public void handleUpdateStudent(MouseEvent mouseEvent) {
        Student selectedStudent =
studentTable.getSelectionModel().getSelectedItem();
        if (selectedStudent != null) {
            Dialog<Student> dialog = new Dialog<>();
            dialog.setTitle("信息修改");

            ButtonType modifyButtonType = new ButtonType("修改",
ButtonBar.ButtonData.OK_DONE);
```

```
            dialog.getDialogPane().getButtonTypes().addAll(modifyButtonType,
ButtonType.CANCEL);

            GridPane gridPane = new GridPane();
            gridPane.setPadding(new Insets(10));
            gridPane.setHgap(10);
            gridPane.setVgap(10);

            TextField nameTextField = new TextField(selectedStudent.getName());
            nameTextField.setDisable(true);
            TextField genderTextField = new
TextField(selectedStudent.getGender());
            genderTextField.setDisable(true);
            TextField studentIdTextField = new
TextField(selectedStudent.getStudentId());
            studentIdTextField.setDisable(true);
            TextField scoreTextField = new
TextField(Double.toString(selectedStudent.getScore()));

            gridPane.add(new Label("Name:"), 0, 0);
            gridPane.add(nameTextField, 1, 0);
            gridPane.add(new Label("Gender:"), 0, 1);
            gridPane.add(genderTextField, 1, 1);
            gridPane.add(new Label("Student ID:"), 0, 2);
            gridPane.add(studentIdTextField, 1, 2);
            gridPane.add(new Label("Score:"), 0, 3);
            gridPane.add(scoreTextField, 1, 3);

            dialog.getDialogPane().setContent(gridPane);

            dialog.setResultConverter(dialogButton -> {
                if (dialogButton == modifyButtonType) {
                    selectedStudent.setScore(Double.parseDouble(scoreTextField.ge
tText()));
                    try {
                        // backup the database
                        String DbPath = GetCWD.getCWD() + "\\backup\\" +
GenerateRandomTime.Generate() + ".db";
                        Main.dbPathStack.push(DbPath);
                        DBUtil.dbBackup(DbPath);
                        // update the student score
                        StudentDAO.updateScore(selectedStudent.getStudentId(),
selectedStudent.getScore());
                        Main.logger.info("学生" + selectedStudent.getName() + "的
成绩被修改为: "
```

```
                                + selectedStudent.getScore());
                        // PieChart
                        UpdatePieChart.updatePieChart(scorePieChart);
                    } catch (SQLException | IOException e) {
                        throw new RuntimeException(e);
                    }
                    return selectedStudent;
                }
                return null;
            });

            dialog.showAndWait().ifPresent(result -> {
                studentTable.refresh();
            });
        }
    }

    // handle Revoke
    @FXML
    public void handleRevoke(ActionEvent actionEvent) {
        try {
            Revoke.revokeEdit(Main.dbPathStack, studentTable);

        } catch (SQLException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    // restore DB
    public void handleRestoreDB(ActionEvent actionEvent) {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("选择数据库文件");
        fileChooser.getExtensionFilters().addAll(
                new FileChooser.ExtensionFilter("DB Files", "*.db")
        );
        fileChooser.setInitialDirectory(new File(GetCWD.getCWD() +
"\\backup\\"));
        File file = fileChooser.showOpenDialog(null);
        if (file != null) {
            try {
                DBUtil.dbRestore(file.getAbsolutePath());
                Main.logger.info("成功恢复数据库");
```

```
                ShowMessage.showAlert(Alert.AlertType.INFORMATION, "成功", "数据
库恢复成功！");

                ObservableList<Student> studentData =
StudentDAO.searchStudents();
                populateStudents(studentData);
                // PieChart
                UpdatePieChart.updatePieChart(scorePieChart);
            } catch (IOException | SQLException e) {
                Main.logger.error("数据库恢复失败");
                ShowMessage.showErrorAlert("错误", "数据库恢复失败！");
                e.printStackTrace();
            }
        }
    }
}
```

SettingPageLayout.java

```
package com.bobmaster.studentsystem.controller;

import com.bobmaster.studentsystem.Config;
import com.bobmaster.studentsystem.util.ShowMessage;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;
import javafx.scene.input.MouseEvent;
import javafx.stage.DirectoryChooser;

import java.io.File;
import java.io.IOException;
import java.util.Map;

public class SettingPageLayout {
    @FXML
    public TextField backUpPathText;

    @FXML
    public TextField backUpIntervalText;

    @FXML
    public Map<String, Object> map = null;

    @FXML
    public void initialize() throws IOException {
        map = (Map<String, Object>) Config.LoadConfig(Map.class);
```

```
        backUpPathText.setText(map.get("backupPath").toString());
        backUpIntervalText.setText(map.get("backupInterval").toString());
    }

    @FXML
    public void handleSettingBackUpPath(MouseEvent mouseEvent) {
        // popup a directory chooser dialog
        DirectoryChooser directoryChooser = new DirectoryChooser();
        directoryChooser.setTitle("选择备份路径");
        File file = directoryChooser.showDialog(null);
        if (file != null) {
            backUpPathText.setText(file.getAbsolutePath());
        }
    }

    @FXML
    public void handleSettingSave(MouseEvent mouseEvent) throws IOException {
        map.put("backupPath", backUpPathText.getText());
        map.put("backupInterval",
Integer.parseInt(backUpIntervalText.getText()));
        Config.SaveConfig(map);
        ShowMessage.showAlert(Alert.AlertType.INFORMATION, "保存成功", "定时任务间
隔重启生效");
        backUpPathText.getScene().getWindow().hide();
    }
}
```

model

Student.java

```
package com.bobmaster.studentsystem.model;
import javafx.beans.property.*;
public class Student {
    // Declare Students Table Columns
    private final StringProperty name;
    private final StringProperty gender;
    private final StringProperty studentId;
    private final DoubleProperty score;

    // Constructor
    public Student() {
        this.name = new SimpleStringProperty();
        this.gender = new SimpleStringProperty();
        this.studentId = new SimpleStringProperty();
```

```
        this.score = new SimpleDoubleProperty();
    }

    // name
    public String getName() {
        return name.get();
    }

    public void setName(String name) {
        this.name.set(name);
    }

    public StringProperty studentNameProperty() {
        return name;
    }
    // gender
    public String getGender() {
        return gender.get();
    }

    public void setGender(String gender) {
        this.gender.set(gender);
    }

    public StringProperty studentGenderProperty() {
        return gender;
    }

    // studentId
    public String getStudentId() {
        return studentId.get();
    }

    public void setStudentId(String studentId) {
        this.studentId.set(studentId);
    }

    public StringProperty studentIdProperty() {
        return studentId;
    }

    // score
    public double getScore() {
        return score.get();
    }
```

```
    public void setScore(double score) {
        this.score.set(score);
    }

    public DoubleProperty studentScoreProperty() {
        return score;
    }
}
```

StudentDAO.java

```
package com.bobmaster.studentsystem.model;

import com.bobmaster.studentsystem.util.DBUtil;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.sql.*;

public class StudentDAO {
    // ******************************
    // select a student
    // ******************************
    public static Student searchStudent(String studentId) throws SQLException {
        // Declare a SELECT statement
        String selectStmt = "SELECT * FROM student WHERE studentId = '" +
studentId + "'";

        // Execute SELECT statement
        try {
            // Get ResultSet from dbExecuteQuery method
            ResultSet rsStudent = DBUtil.dbExecuteQuery(selectStmt);

            // Return student object
            return getStudentFromResultSet(rsStudent);
        } catch (SQLException e) {
            System.out.println("While searching a student with " + studentId + "
id, an error occurred: " + e);
            // Return exception
            throw e;
        }
    }

    // use ResultSet from DB as parameter and set Student Object's attributes and
return student object.
```

```
    private static Student getStudentFromResultSet(ResultSet rs) throws
SQLException {
        Student stu = null;
        if (rs.next()) {
            stu = new Student();
            stu.setName(rs.getString("name"));
            stu.setGender(rs.getString("gender"));
            stu.setStudentId(rs.getString("studentId"));
            stu.setScore(rs.getDouble("score"));
        }
        return stu;
    }

    // ******************************
    // select all students
    // ******************************
    public static ObservableList<Student> searchStudents() throws SQLException {
        // Declare a SELECT statement
        String selectStmt = "SELECT * FROM student";

        // Execute SELECT statement
        try {
            // Get ResultSet from dbExecuteQuery method
            ResultSet rsStudents = DBUtil.dbExecuteQuery(selectStmt);
            // Send ResultSet to the getStudentList method and get student object
            ObservableList<Student> studentList = getStudentList(rsStudents);

            // Return student object
            return studentList;
        } catch (SQLException e) {
            System.out.println("SQL select operation has been failed: " + e);
            // Return exception
            throw e;
        }
    }

    // Select * from students operation
    private static ObservableList<Student> getStudentList(ResultSet rs) throws
SQLException {
        ObservableList<Student> studentList =
FXCollections.observableArrayList();
        while (rs.next()) {
            Student stu = new Student();
            stu.setName(rs.getString("name"));
            stu.setGender(rs.getString("gender"));
```

```
            stu.setStudentId(rs.getString("studentId"));
            stu.setScore(rs.getDouble("score"));
            // Add student to the ObservableList
            studentList.add(stu);
        }
        return studentList;
    }

    // ******************************
    // insert a student
    // ******************************
    public static void insertStudent(String name, String gender, String
studentId, double score) throws SQLException {
        // Declare a Insert statement
        String insertStmt =
                "INSERT INTO student\n" +
                        "(name, gender, studentId, score)\n" +
                        "VALUES\n" +
                        "('"+name+"','"+"'"+gender+"','"+"'"+studentId+"','"+score+"
);";
        // Execute INSERT operation
        try {
            DBUtil.dbExecuteUpdate(insertStmt);
        } catch (SQLException e) {
            System.out.print("Error occurred while INSERT Operation: " + e);
            throw e;
        }
    }

    // ******************************
    // update a student
    // ******************************
    public static void updateScore(String studentId, double score) throws
SQLException {
        // Declare a UPDATE statement
        String updateStmt =
                "BEGIN TRANSACTION;\n" +
                        "UPDATE student\n" +
                        "SET score = "+score+"\n" +
                        "WHERE studentId = '"+studentId+"';\n" +
                        "COMMIT;";
        // Execute UPDATE operation
        try {
            DBUtil.dbExecuteUpdate(updateStmt);
        } catch (SQLException e) {
```

```
            System.out.print("Error occurred while UPDATE Operation: " + e);
            throw e;
        }
    }

    // ******************************
    // delete a student
    // ******************************
    public static void deleteStudent(String studentId) throws SQLException{
        // Declare a DELETE statement
        String deleteStmt =
                "BEGIN TRANSACTION;\n" +
                        "DELETE FROM student\n" +
                        "WHERE studentId = '"+studentId+"';\n" +
                        "COMMIT;";
        // Execute DELETE operation
        try {
            DBUtil.dbExecuteUpdate(deleteStmt);
        } catch (SQLException e) {
            System.out.print("Error occurred while DELETE Operation: " + e);
        }
    }
}
```

util

DBUtil.java

```
package com.bobmaster.studentsystem.util;
import java.io.File;
import java.sql.*;
public class DBUtil {
    // Declare sqlite3 URL
    private static final String DB_URL = "jdbc:sqlite:student.db";

    // Connection
    private static Connection conn = null;

    // Connect to database
    public static void dbConnect() throws SQLException {
        try {
            conn = DriverManager.getConnection(DB_URL);
        } catch (SQLException e) {
            System.out.println("Connection failed: " + e.getMessage());
            throw e;
        }
    }
```

```
    // Close connection
    public static void dbDisconnect() throws SQLException {
        try {
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (SQLException e) {
            System.out.println("Disconnection failed: " + e.getMessage());
            throw e;
        }
    }

    // DB Execute Query Operation
    public static ResultSet dbExecuteQuery(String queryStmt) throws SQLException
{
        Statement stmt = null;
        ResultSet resultSet = null;
        try {
            dbConnect();
            stmt = conn.createStatement();
            resultSet = stmt.executeQuery(queryStmt);
        } catch (SQLException e) {
            System.out.println("Problem occurred at executeQuery operation: " +
e.getMessage());
            throw e;
        }
        return resultSet;
    }

    // DB Execute Update Operation (For Update/Insert/Delete)
    public static void dbExecuteUpdate(String sqlStmt) throws SQLException {
        Statement stmt = null;
        try {
            dbConnect();
            stmt = conn.createStatement();
            stmt.executeUpdate(sqlStmt);
        } catch (SQLException e) {
            System.out.println("Problem occurred at executeUpdate operation: " +
e.getMessage());
            throw e;
        }
    }

    // DB initialize
```

```
    public static void dbInit() throws SQLException {
        Statement stmt = null;
        try {
            dbConnect();
            stmt = conn.createStatement();
            stmt.execute("CREATE TABLE IF NOT EXISTS student (" +
                            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                            "name TEXT," +
                            "gender TEXT," +
                            "studentId TEXT," +
                            "score REAL"+
                            ");"
            );
        } catch (SQLException e) {
            System.out.println("Problem occurred at dbInit operation: " +
e.getMessage());
            throw e;
        }
    }

    // DB backup
    public static void dbBackup(String path) throws SQLException {
        // check if directory exists, if not create it
        String dir = path.substring(0, path.lastIndexOf("\\"));
        File file = new File(dir);
        if (!file.exists()) {
            file.mkdirs();
        }

        Statement stmt = null;
        try {
            dbConnect();
            stmt = conn.createStatement();
            stmt.execute("backup to " + path);
        } catch (SQLException e) {
            System.out.println("Problem occurred at dbBackup operation: " +
e.getMessage());
            throw e;
        }
    }

    // DB restore
    public static void dbRestore(String path) throws SQLException {
        Statement stmt = null;
        try {
```

```
            dbConnect();
            stmt = conn.createStatement();
            stmt.execute("restore from " + path);
        } catch (SQLException e) {
            System.out.println("Problem occurred at dbRestore operation: " +
e.getMessage());
            throw e;
        }
    }
}
```

ExportToExcel.java

```
package com.bobmaster.studentsystem.util;
import com.bobmaster.studentsystem.Main;
import javafx.scene.control.Alert;
import javafx.stage.FileChooser;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;


import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import java.util.Optional;

public class ExportToExcel {
    public static void exportToExcel() {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("导出为 Excel");
        fileChooser.getExtensionFilters().add(new
FileChooser.ExtensionFilter("Excel Files", "*.xlsx"));
        fileChooser.setInitialFileName("学生信息_" +
GenerateRandomTime.Generate()
                + ".xlsx");

        File file = fileChooser.showSaveDialog(null);
        if (file != null) {
            try {
                ResultSet rs = DBUtil.dbExecuteQuery("SELECT name, gender,
studentId, " +
                        "score FROM student ORDER BY score DESC");
```

```
                XSSFWorkbook workbook = new XSSFWorkbook();
                workbook.getProperties().getCoreProperties().setCreator("龚智勋
");
                workbook.getProperties().getCoreProperties().setCreated(Optional.
of(new Date()));
                Sheet sheet = workbook.createSheet("学生综合信息表");

                // Create header row
                Row headerRow = sheet.createRow(0);
                headerRow.createCell(0).setCellValue("姓名");
                headerRow.createCell(1).setCellValue("性别");
                headerRow.createCell(2).setCellValue("学号");
                headerRow.createCell(3).setCellValue("JP 成绩");

                // Write student data
                int rowNum = 1;
                while (rs.next()) {
                    String name = rs.getString("name");
                    String gender = rs.getString("gender");
                    String studentId = rs.getString("studentId");
                    double score = rs.getDouble("score");

                    Row row = sheet.createRow(rowNum++);
                    row.createCell(0).setCellValue(name);
                    row.createCell(1).setCellValue(gender);
                    row.createCell(2).setCellValue(studentId);
                    row.createCell(3).setCellValue(score);
                }

                // Auto size columns
                for (int i = 0; i < 4; i++) {
//                    sheet.autoSizeColumn(i);
                    if (i == 0) {
                        sheet.setColumnWidth(i, 256 * 6);
                    } else if (i == 1) {
                        sheet.setColumnWidth(i, 256 * 6);
                    } else if (i == 2) {
                        sheet.setColumnWidth(i, 256 * 12);
                    } else {
                        sheet.setColumnWidth(i, 256 * 6);
                    }
                }

                // Write workbook to file
                try {
```

```
                    FileOutputStream fileOutputStream = new
FileOutputStream(file);
                    workbook.write(fileOutputStream);
                    workbook.close();
                } catch (IOException e) {
                    ShowMessage.showErrorAlert("导出失败", "Excel 文件导出失败
"+e.getMessage());
                    e.printStackTrace();
                }
                Main.logger.info("学生信息导出至: "+file.getAbsolutePath());
                ShowMessage.showAlert(Alert.AlertType.INFORMATION, "导出成功",
"Excel 文件导出至: "+
                        file.getAbsolutePath());

            } catch (SQLException e ) {
                ShowMessage.showErrorAlert("导出失败", "Excel 文件导出失败
"+e.getMessage());
                e.printStackTrace();
            }

        }
    }
}
```

GenerateRandomTime.java

```
package com.bobmaster.studentsystem.util;

import java.text.SimpleDateFormat;
import java.util.Date;

public class GenerateRandomTime {
    public static String Generate() {
        return new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss").format(new Date());
    }
}
```

GetCWD.java

```
package com.bobmaster.studentsystem.util;

import java.nio.file.Paths;

public class GetCWD {
    public static String getCWD() {
        return Paths.get("").toAbsolutePath().toString();
    }
```

```
}
```

Revoke.java

```
package com.bobmaster.studentsystem.util;
import com.bobmaster.studentsystem.model.Student;
import com.bobmaster.studentsystem.model.StudentDAO;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.chart.PieChart;
import javafx.scene.control.Alert;
import javafx.scene.control.TableView;

import java.io.IOException;
import java.sql.SQLException;
import java.util.Stack;

public class Revoke {
    public static void revokeEdit(Stack<String> stack, TableView<Student>
studentTableView) throws SQLException, IOException {
        if (stack.size() != 0) {
            String dbPath = stack.pop();
            DBUtil.dbRestore(dbPath);
            ObservableList<Student> studentData = StudentDAO.searchStudents();
            studentTableView.setItems(studentData);
            // Update pieChart
            Scene scene = studentTableView.getScene();
            UpdatePieChart.updatePieChart((PieChart)
scene.lookup("#scorePieChart"));
        } else {
            ShowMessage.showAlert(Alert.AlertType.WARNING, "撤销失败", "没有可撤销
的操作");
        }
    }
}
```

ShowMessage.java

```
package com.bobmaster.studentsystem.util;

import javafx.scene.control.Alert;

public class ShowMessage {
    public static void showAlert(Alert.AlertType alertType, String title, String
message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
```

```
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }

    public static void showErrorAlert(String title, String message) {
        showAlert(Alert.AlertType.ERROR, title, message);
    }
}
```

UpdatePieChart.java

```
package com.bobmaster.studentsystem.util;

import com.bobmaster.studentsystem.Main;
import com.bobmaster.studentsystem.controller.RootLayout;
import com.bobmaster.studentsystem.model.StudentDAO;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.chart.PieChart;
import javafx.scene.control.Label;

import java.io.IOException;
import java.sql.SQLException;

public class UpdatePieChart {
    public static void updatePieChart(PieChart pieChart) throws SQLException,
IOException {
        ObservableList<PieChart.Data> pieChartData =
FXCollections.observableArrayList();
        // calculate the number of students in each score range
        int[] scoreRange = new int[6];
        for (int i = 0; i < 5; i++) {
            scoreRange[i] = 0;
        }
        for (int i = 0; i < StudentDAO.searchStudents().size(); i++) {
            if (StudentDAO.searchStudents().get(i).getScore() < 60) {
                scoreRange[0]++;
            } else if (StudentDAO.searchStudents().get(i).getScore() < 70) {
                scoreRange[1]++;
            } else if (StudentDAO.searchStudents().get(i).getScore() < 80) {
                scoreRange[2]++;
            } else if (StudentDAO.searchStudents().get(i).getScore() < 90) {
                scoreRange[3]++;
```

```
            } else if (StudentDAO.searchStudents().get(i).getScore() <= 100) {
                scoreRange[4]++;
            }
        }
        int total = 0;
        for (int i = 0; i < 5; i++) {
            total += scoreRange[i];
        }
        // add data to pie chart
        pieChartData.add(new PieChart.Data("0-60(" + scoreRange[0]*100/total +
"%)", scoreRange[0]));
        pieChartData.add(new PieChart.Data("60-70(" + scoreRange[1]*100/total +
"%)", scoreRange[1]));
        pieChartData.add(new PieChart.Data("70-80(" + scoreRange[2]*100/total +
"%)", scoreRange[2]));
        pieChartData.add(new PieChart.Data("80-90(" + scoreRange[3]*100/total +
"%)", scoreRange[3]));
        pieChartData.add(new PieChart.Data("90-100(" + scoreRange[4]*100/total +
"%)", scoreRange[4]));

        // set pieChart data
        pieChart.setData(pieChartData);
    }
}
```

resources

AutoPage.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Label?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane fx:id="aboutPagePane" minHeight="0.0" minWidth="0.0"
onMouseClicked="#handleHideAboutPage" prefHeight="259.0" prefWidth="416.0"
xmlns="http://javafx.com/javafx/20.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.bobmaster.studentsystem.controller.AboutPage">
   <children>
      <ImageView fitHeight="176.0" fitWidth="154.0" layoutX="14.0" layoutY="27.0"
pickOnBounds="true" preserveRatio="true">
         <image>
```

```
            <Image url="@assets/aboutPage.png" />
         </image>
      </ImageView>
      <Label layoutX="168.0" layoutY="14.0" prefHeight="37.0" prefWidth="196.0"
text="学生成绩管理系统&#10;">
         <font>
            <Font name="System Bold" size="24.0" />
         </font>
      </Label>
      <Label layoutX="316.0" layoutY="228.0" prefHeight="17.0" prefWidth="63.0"
text="版本: v0.0.1" />
      <Label layoutX="96.0" layoutY="228.0" prefHeight="17.0" prefWidth="187.0"
text="Powered by openjdk20.0.1+javaFX" />
      <Label layoutX="220.0" layoutY="51.0" prefHeight="17.0" prefWidth="73.0"
text="Java 课设作品" />
      <Label layoutX="214.0" layoutY="68.0" text="通信 2102 龚智勋" />
   </children>
</AnchorPane>
```

RootLayout.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.Cursor?>
<?import javafx.scene.Group?>
<?import javafx.scene.chart.PieChart?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.HBox?>

<AnchorPane focusTraversable="true" prefHeight="472.0" prefWidth="637.0"
stylesheets="@style.css" xmlns="http://javafx.com/javafx/20.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.bobmaster.studentsystem.controller.RootLayout">
    <children>
      <Group layoutX="405.0" layoutY="260.0">
         <children>
              <Label layoutY="4.0" text="姓名:">
```

```
                <opaqueInsets>
                    <Insets />
                </opaqueInsets>
            </Label>
            <TextField fx:id="nameText" layoutX="35.0" prefHeight="25.0"
prefWidth="142.0" />
            <Label layoutY="41.0" text="性别:" />
            <TextField fx:id="genderText" layoutX="35.0" layoutY="37.0"
prefHeight="25.0" prefWidth="142.0" />
            <Label layoutY="80.0" text="学号:" />
            <TextField fx:id="idText" layoutX="35.0" layoutY="76.0"
prefHeight="25.0" prefWidth="142.0" />
          <Label layoutX="-5.0" layoutY="113.0" text="JP 成绩:" />
          <TextField fx:id="scoreText" layoutX="35.0" layoutY="109.0"
prefHeight="25.0" prefWidth="141.0" />
        </children>
      </Group>
      <HBox layoutY="31.0" prefHeight="440.0" prefWidth="356.0">
        <children>
            <TableView fx:id="studentTable" editable="true" prefHeight="455.0"
prefWidth="356.0">
                <columns>
                    <TableColumn fx:id="stuNameColumn" prefWidth="75.0" text="
姓名" />
                    <TableColumn fx:id="stuGenderColumn"
prefWidth="42.000030517578125" text="性别" />
                <TableColumn fx:id="stuIdColumn" prefWidth="75.0" text="学号"
/>
                <TableColumn fx:id="stuScoreColumn" prefWidth="75.0" text="JP
成绩" />
                </columns>
            </TableView>
        </children>
      </HBox>
        <MenuBar layoutY="1.0" opacity="0.67" prefHeight="31.0"
prefWidth="637.0">
          <menus>
            <Menu mnemonicParsing="false" text="File">
                <items>
                    <MenuItem mnemonicParsing="false"
onAction="#handleExportToExcel" text="导出为 Excel    CTRL+S" />
                    <MenuItem fx:id="quit" mnemonicParsing="false"
onAction="#handleExit" text="退出                CTRL+Q" />
                </items>
            </Menu>
```

```
                <Menu mnemonicParsing="false" text="Edit">
                    <items>
                  <MenuItem mnemonicParsing="false" onAction="#handleSettingPage"
text="设置" />
                        <MenuItem mnemonicParsing="false"
onAction="#handleRevoke" text="撤销    CTRL+Z" />
                  <MenuItem mnemonicParsing="false" onAction="#handleRestoreDB"
text="还原数据库" />
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Help">
               <items>
                  <MenuItem mnemonicParsing="false" onAction="#handleAboutPage"
text="关于" />
               </items></Menu>
            </menus>
         <padding>
            <Insets bottom="5.0" />
         </padding>
         <cursor>
            <Cursor fx:constant="HAND" />
         </cursor>
        </MenuBar>
      <PieChart fx:id="scorePieChart" layoutX="371.0" layoutY="40.0"
legendSide="RIGHT" prefHeight="202.0" prefWidth="249.0" title="成绩分布" />
        <Button fx:id="addStudentBtn" layoutX="437.0" layoutY="410.0"
mnemonicParsing="false" onMouseClicked="#handleAddStudent" text="添加">
        </Button>
        <Button fx:id="updateStudentBtn" layoutX="491.0" layoutY="410.0"
mnemonicParsing="false" onMouseClicked="#handleUpdateStudent" text="修改" />
        <Button fx:id="deleteStudentBtn" layoutX="542.0" layoutY="410.0"
mnemonicParsing="false" onMouseClicked="#handleDeleteStudent" text="删除" />
    </children>
</AnchorPane>
```

SettingPage.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>
```

```
<AnchorPane prefHeight="286.0" prefWidth="399.0"
xmlns="http://javafx.com/javafx/20.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.bobmaster.studentsystem.controller.SettingPageLayout">
   <children>
      <Label layoutX="46.0" layoutY="57.0" text="备份存储路径">
         <font>
            <Font size="16.0" />
         </font>
      </Label>
      <Label layoutX="46.0" layoutY="172.0" text="自动备份时间间隔">
         <font>
            <Font size="16.0" />
         </font>
      </Label>
      <TextField fx:id="backUpIntervalText" layoutX="200.0" layoutY="172.0"
promptText="60 秒" />
      <TextField fx:id="backUpPathText" layoutX="149.0" layoutY="57.0"
prefHeight="25.0" prefWidth="128.0" />
      <Button layoutX="289.0" layoutY="57.0" mnemonicParsing="false"
onMouseClicked="#handleSettingBackUpPath" text="选择" />
      <Button layoutX="174.0" layoutY="228.0" mnemonicParsing="false"
onMouseClicked="#handleSettingSave" prefHeight="25.0" prefWidth="62.0" text="保存
" />
   </children>
</AnchorPane>
```

style.css

```
.menuItem {
    -fx-background-radius: 5;
}

.context-menu {
    -fx-background-insets: 0, 1, 2;
    -fx-background-radius: 0 6 6 6, 0 5 5 5, 0 4 4 4;
    -fx-padding: 0.333333em 0.083333em 0.666667em 0.083333em; /* 4 1 8 1 */
}
```

log4j2.properties

```
appender.file.type = File
appender.file.name = LOGFILE
appender.file.fileName = Program.log
appender.file.layout.type = PatternLayout
appender.file.layout.pattern = [%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1}
- %msg%n
appender.file.filter.threshold.type = ThresholdFilter
appender.file.filter.threshold.level = info
```

```
appender.console.type = Console
appender.console.name = STDOUT
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = [%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS}
[%t] %c{1} - %msg%n

rootLogger = debug, STDOUT, LOGFILE
```