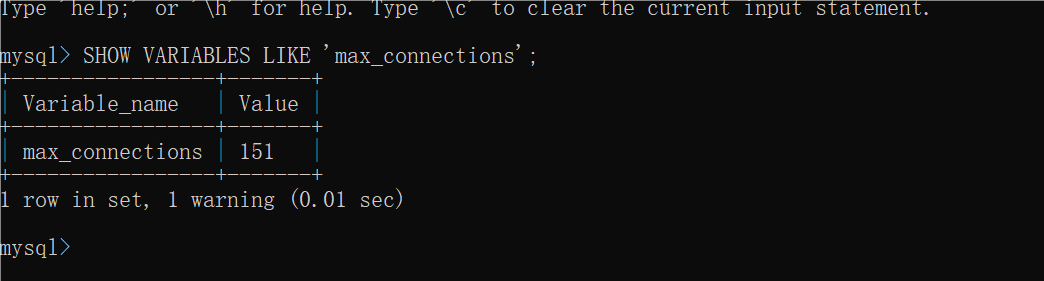
# 学生管理系统的设计与开发

叶泽兵  
日期：2024年8月5日

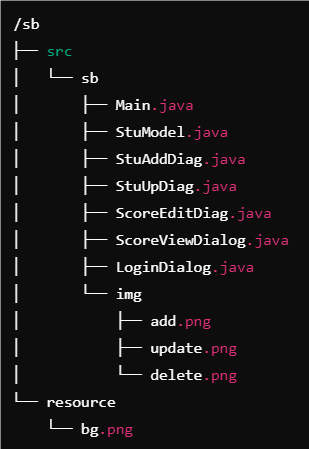
## 成果展示

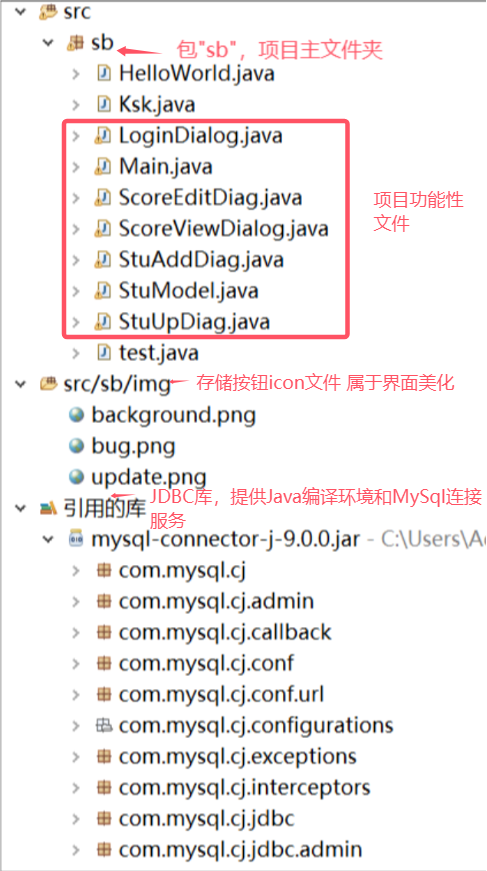
### 1.1功能概述

本数据库使用可编写Java程序的IDE（免费的eclipse IDE）和MySql制作而成。其中，本客户端旨在为用户提供可方便访问，管理，使用MySQL上的数据库实例的途径和方法。其数据库实例的最大连接数由MySQL设定默认为151。



如图所示最大连接数为151

项目结构：  




首先，本客户端用各Java类实现了如下功能和可视化界面方便用户操作：

**1. 学生信息管理**

增删改查功能：

添加学生：通过编写StuAddDiag类 弹出窗口，实现新生信息的录入。

修改学生信息：通过 StuUpDiag 弹出窗口，可以对现有学生信息进行编辑和更新。

删除学生信息：用户可以选择某一行记录，通过确认对话框实现永久删除操作。

查询学生信息：通过输入学生名字进行查询，并在界面上显示搜索结果。

清除查询：点击清除按钮后，可以显示所有学生记录，查询后不影响后续用户使用。

**数据表设计：**

stu 表：存储学生的基本信息，包括 stuID, stuName, stuSex, stuAge, stuJG, stuDept。

stuScore 表：存储学生的学分信息和额外说明，包括 stuID, stuSc, stuScno, stuNt，并与 stu 表通过外键关联。

其数据表之间的关系会在之后的章节用E-R图具体说明。

**2. 学生成绩管理**

成绩浏览和编辑：

在 Main 界面中双击某行时，如果该学生没有成绩信息，系统会询问是否添加，并弹出 类ScoreEditDiag 控制的窗口。

如果该学生已有成绩信息，系统会显示其成绩，并提供“编辑”按钮，进行编辑需要通过用户验证。

**3. 用户身份验证**

安全验证机制：

LoginDialog 类用于实现身份验证功能和窗口，用户在执行重要操作（如添加、修改、删除）时，必须输入正确的用户名和密码进行验证。

验证信息存储在 users 数据库的 admin 表中，保障系统安全。

本数据库系统具有如下特色：

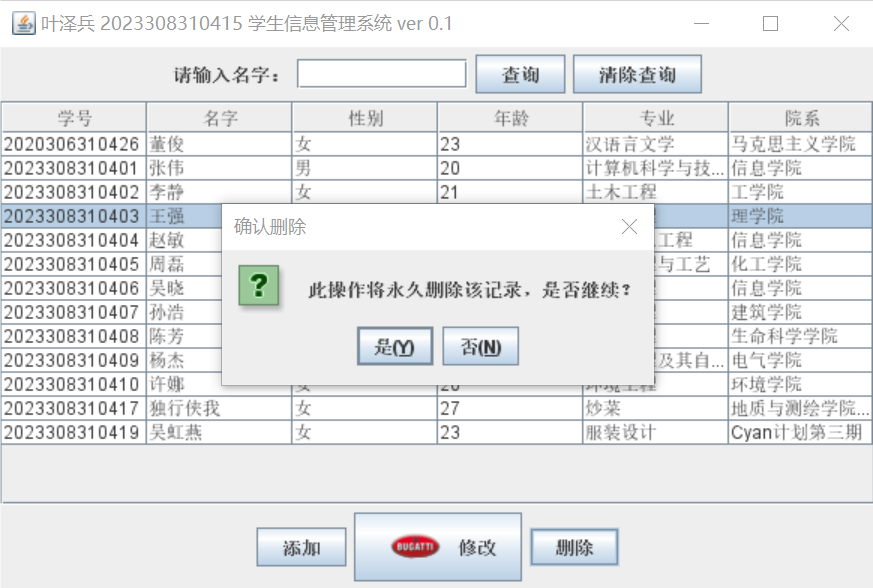
1. 灵活的数据管理

**支持多种操作**：系统支持多种数据操作，包括添加、修改、删除和查询，使得用户可以方便地管理学生信息。

**关联数据表**：stuScore 表和 stu 表通过外键关联，保证数据一致性和完整性。

2. 安全性和用户验证

**身份验证机制**：重要操作需要进行身份验证，提高了系统的安全性，防止未经授权的操作。



要执行删除记录操作 不仅要验证还要二次确认

3. 良好的用户体验

**图形化用户界面**：通过 Java Swing 实现，界面友好，操作直观。

**动态反馈**：用户在操作过程中可以收到即时的反馈信息，提高了交互体验。

4. 易于扩展和维护

**模块化设计**：系统采用面向对象的设计，功能模块划分清晰，便于维护和扩展。

**灵活的数据结构**：数据库设计合理，易于扩展新的功能或字段。

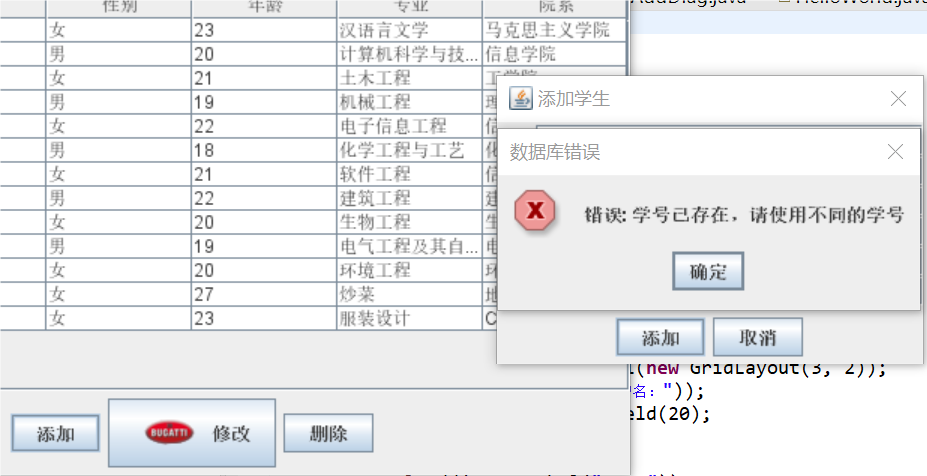
5. 较好的数据正确性保证  
在系统中，我们为不同的输入场景设置了相应的验证逻辑，确保每一个输入都符合预期的格式和范围。

**学生信息输入验证**

**学生ID（stuID）**：

**格式验证**：学生ID必须为12位的数字，并且遵循指定的规则（例如：2023-年份，30831-专业代码，0415-班级和学号）。

**唯一性验证**：通过数据库主键约束，确保每个学生ID都是唯一的，不允许重复。



学号唯一性约束

**姓名（stuName）**：

**非空验证**：学生姓名不能为空，输入框为空时将提示用户输入姓名。

**长度限制**：学生姓名的长度限制在20个字符以内，确保符合数据库字段长度限制。

**性别（stuSex）**：

**选项验证**：性别字段限制为特定的选项（如"男"或"女"），避免无效的性别输入。

**年龄（stuAge）**：

**范围验证**：学生年龄必须在合理的范围内（例如，16到99岁之间），避免输入异常的年龄数据。

**籍贯（stuJG）**：

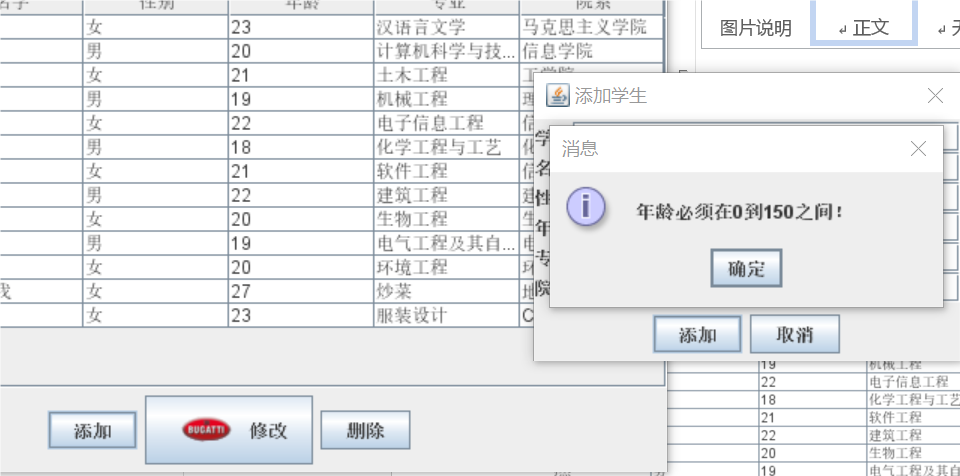
**长度限制**：籍贯字段限制为40个字符以内，确保数据库存储不超限。

**院系（stuDept）**：

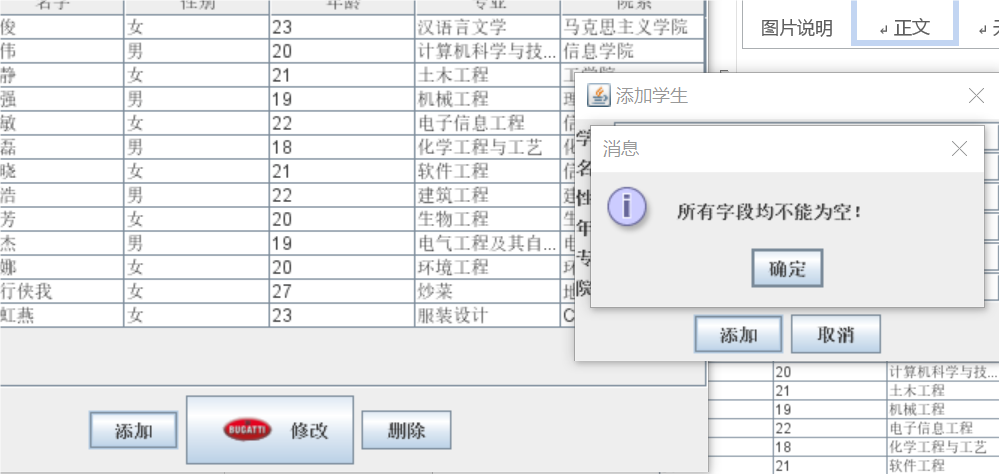
**非空验证**：院系字段不能为空，并且限制在40个字符以内，确保输入符合实际要求。



输入合法性验证



输入合法性验证：年龄



在所有输入合法验证的基础上：填充完整验证

**学生成绩输入验证**

**学分（stuSc）**：

**格式验证**：学分为带一位小数的数字，确保学分输入合法并符合格式要求。

**范围验证**：学分的范围应在合理区间内（如0.0到100.0），防止输入异常值。

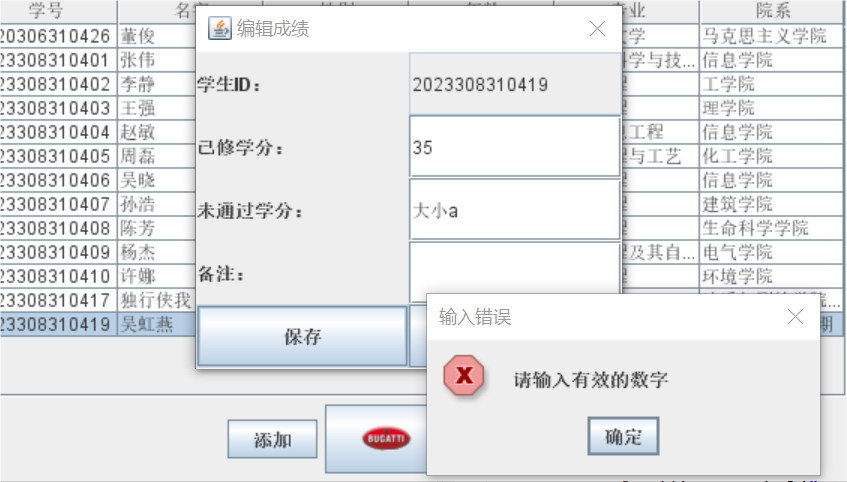
**未通过学分（stuScno）**：

**格式验证**：未通过学分为带一位小数的数字，要求输入格式与学分一致。

**范围验证**：未通过学分的范围也应在合理区间内（如0.0到100.0），确保数据合理性。

**额外说明（stuNt）**：

* + **长度限制**：说明字段限制在255个字符以内，确保数据库存储不超限。



编辑成绩窗口（由类ScoreEditDialog控制）输入合法验证

其他验证环节不一一赘述，详情请见作业提交压缩包的演示视频。

### **1.2 管理系统各界面介绍**

**主界面布局：**

使用 JPanel、JTable、JScrollPane 等 Swing 组件实现用户友好的界面布局。

在 Main 类中，主界面分为三个部分：顶部为查询和清除功能，中间为学生信息显示表格，底部为增删改按钮。

图标和背景：

自定义按钮图标：在按钮上添加自定义的图标文件，如“添加”、“修改”、“删除”等。

友好的用户交互：

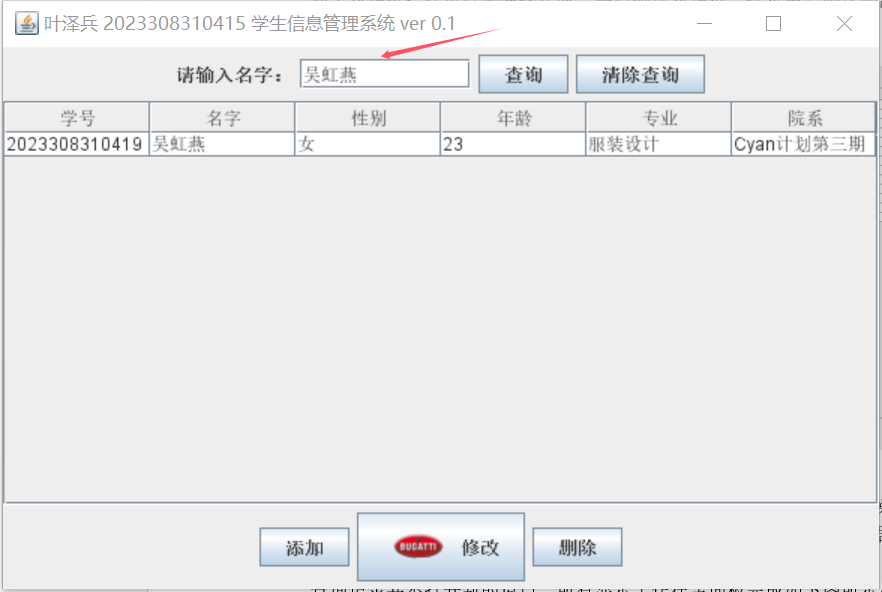
弹出对话框：在进行删除操作时，会有确认对话框，提示用户确认删除。

信息提示：在用户输入错误或操作不当时，通过 JOptionPane 提示相关信息。  


*主界面*

如上图所示，主界面由三个图形区组成：最顶上的显示查询框和查询操作按钮，这是供广大数据库使用者使用的；中间的显示学生信息记录，最下面的显示供数据库管理员使用的三个功能按钮，分别对应记录添加，记录修改，记录删除。

查询记录并不打开新的窗口，所有显示工作在主面板完成，如下图所示：



*查找学生“吴虹燕”*

在我们的学生管理数据库系统中，用户验证功能起到了至关重要的作用。它不仅确保了系统的安全性和稳定性，同时也为敏感操作提供了必要的访问控制，防止未经授权的用户进行重要的操作。以下是项目中用户验证窗口和功能的简要介绍，以及其在系统中的地位和重要性。

**用户验证窗口和功能：**

在我们的系统中，用户验证功能主要通过 LoginDialog 窗口实现。这个窗口用于在用户执行某些敏感操作（例如，添加、修改或删除学生信息）时，对用户身份进行验证，确保只有授权用户才能进行这些操作。

LoginDialog 窗口



*数据库管理员验证窗口*

LoginDialog 是一个自定义的登录窗口，它通过 Swing 界面与用户进行交互。以下是窗口的主要组成部分：

用户名和密码输入框：

用户需要在对应的输入框中输入其用户名和密码。

对输入的内容进行格式验证，确保输入的合法性。

登录按钮：

用户在输入完用户名和密码后，点击登录按钮提交验证信息。

验证通过后，用户可以继续进行后续操作。

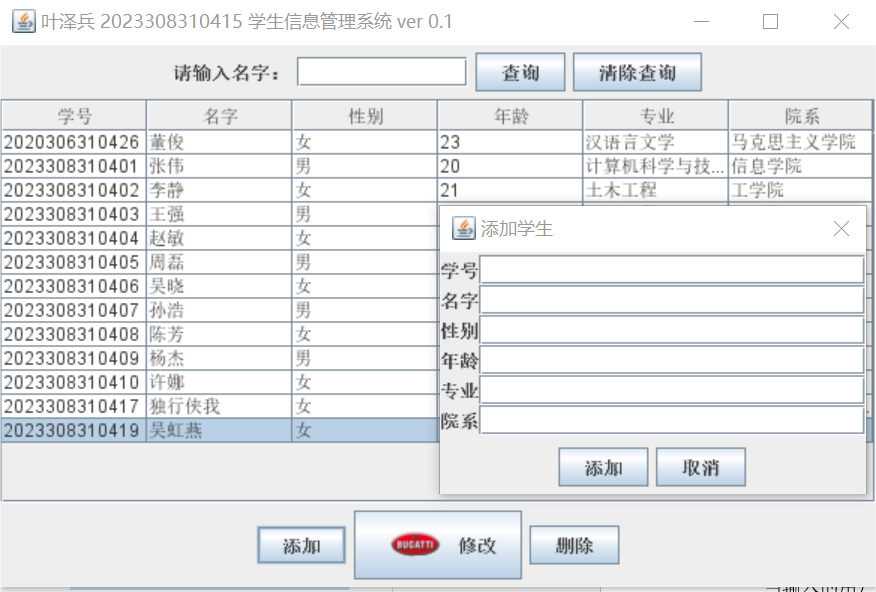
取消按钮：

用户可以点击取消按钮关闭登录窗口，放弃当前的敏感操作。

反馈信息：

当输入的用户名或密码不正确时，系统会通过弹出对话框或标签提示用户进行修正。

**添加学生窗口（类StuAddDiag控制）**

****

*学生信息添加窗口，其有众多方法保证管理员正确录入信息*

功能：

StuAddDiag 是一个自定义的对话框窗口，用于在学生管理系统中添加新的学生记录。这个窗口为用户提供了一个友好的界面，用于输入学生的详细信息，并将其保存到数据库中。以下是这个窗口的主要功能和组成部分：

输入字段：

学生ID (stuID)：用于唯一标识学生。是一个必填字段，用户输入的值必须符合预定格式（例如，12位数字）。

学生姓名 (stuName)：记录学生的姓名。

学生性别 (stuSex)：记录学生的性别，通常是一个下拉菜单，包含选项如“男”和“女”。

学生年龄 (stuAge)：记录学生的年龄，通常是一个整数值。

学生籍贯 (stuJG)：记录学生的籍贯或专业名称。

学生部门 (stuDept)：记录学生所属的部门或学院。

按钮操作：

确认按钮 (Confirm)：用户点击后，将输入的学生信息保存到数据库中。

取消按钮 (Cancel)：用户点击后，关闭对话框而不保存数据。

输入验证：

格式验证：确保所有必填字段都被填写，并且数据格式符合预期。

唯一性验证：检查 stuID 是否已存在于数据库中，以避免重复记录。

数据提交：

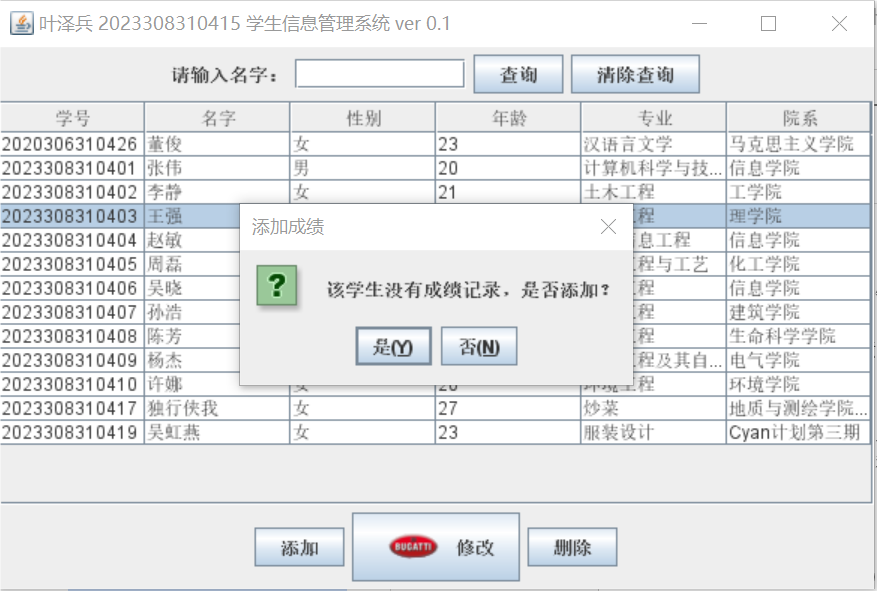
当用户点击确认按钮时，系统将从输入字段中提取数据，并将其插入到 stu 表中。

如果存在输入错误或数据不符合要求，系统会提示用户进行修正。

**添加学生成绩窗口（ScoreEditDiag）的功能**

功能

ScoreEditDiag 是学生管理系统中用于管理学生成绩的窗口。这个窗口允许用户输入和编辑学生的成绩信息，并将这些信息存储在数据库的 stuScore 表中。通过 ScoreEditDiag，用户可以详细查看和管理学生的已修学分、未通过学分和额外说明。以下是 ScoreEditDiag 的主要功能：



*若学生没有成绩记录，会弹出输入引导窗口*



*添加成绩窗口*

输入字段：

学生ID (stuID)：用于标识学生。这一字段是不可编辑的，只显示被选择学生的 ID。

已修学分 (stuSc)：记录学生已修得的学分，允许用户输入带一位小数的数值。

未通过学分 (stuScno)：记录学生未通过的学分，同样允许输入带一位小数的数值。

额外说明 (stuNt)：用于输入与学生成绩相关的备注或说明。

按钮操作：

保存按钮 (Save)：用于保存当前窗口中的成绩信息到数据库。

取消按钮 (Cancel)：用于关闭窗口而不保存数据。

输入验证：

格式验证：确保学分输入为带一位小数的数值，并检查所有必填字段是否完整。

逻辑验证：可能包括已修学分不能小于未通过学分等逻辑判断。

数据提交：

将用户输入的数据提交到 stuScore 表中。如果是新的记录，进行插入操作；如果是已有记录，进行更新操作。

数据库交互：

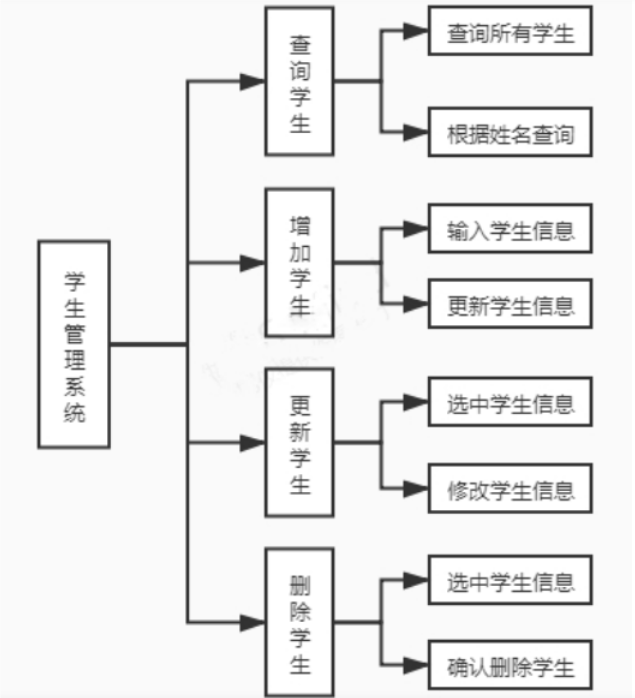
在操作前，从数据库中检查是否已有该学生的成绩记录。如果已有，进行更新；如果没有，则提示添加新的记录。

### **1.3 使用效果（演示视频）**

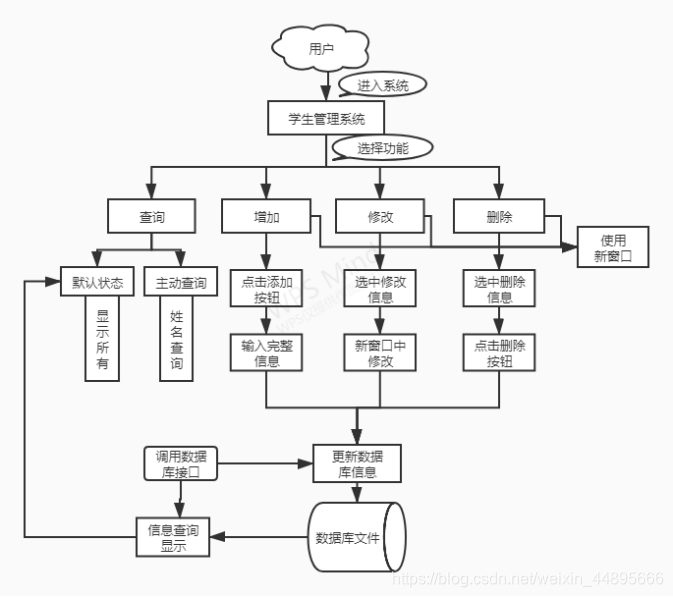
详见压缩包中的演示视频。

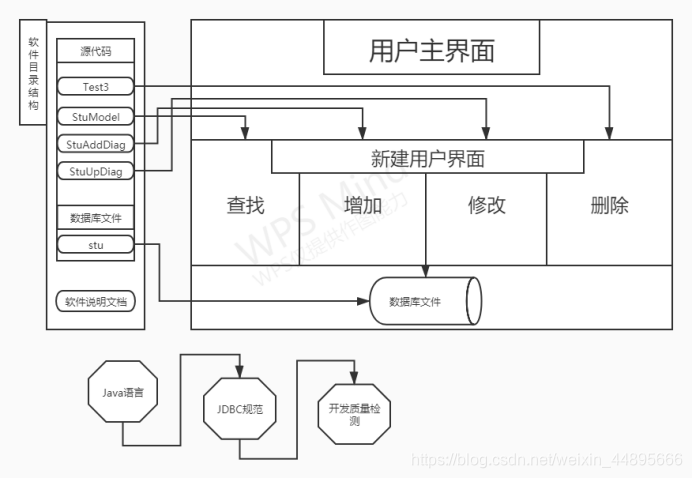
## 第二章 项目结构图设计

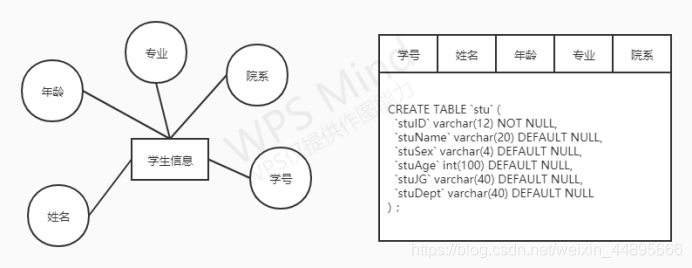
### **2.1客户端功能结构**

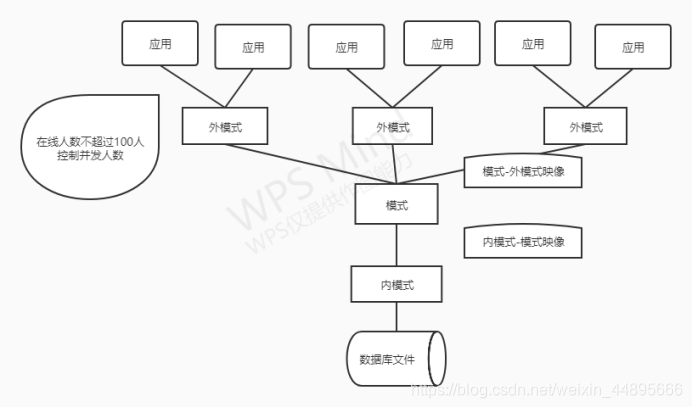


### 2.2软件架构设计



逻辑架构（使用wps绘制）  


开发架构  


数据架构（表stu）  


运行架构（在线人数实际上最高可到达151人）

## 第三章 项目源码

### 3.1类Main

1. *// filename : Main.java*
2. package sb;
3. import java.awt.Graphics;
4. import java.awt.Image;
5. import javax.swing.ImageIcon;
6. import javax.swing.JLabel;
7. import javax.swing.JPanel;
8. import javax.swing.JScrollPane;
9. import javax.swing.JTable;
10. import javax.swing.\*;
11. import java.awt.event.\*;
12. import java.sql.Connection;
13. import java.sql.DriverManager;
14. import java.sql.PreparedStatement;
15. import java.sql.ResultSet;
16. import java.sql.Statement;
17. public class Main extends JFrame implements ActionListener {
18. *// 定义一些控件*
19. JPanel jp1, jp2;
20. JLabel jl1;
21. JButton jb1, jb2, jb3, jb4, jb5; *// 添加“清除查询”按钮 jb5*
22. JTable jt;
23. JScrollPane jsp;
24. JTextField jtf;
25. StuModel sm;
26. *// 定义连接数据库的变量*
27. Statement stat = null;
28. PreparedStatement ps;
29. Connection ct = null;
30. ResultSet rs = null;
31. public static void main(String[] args) {
32. Main main = new Main();
33. }
34. *// 构造函数*
35. public Main() {
36. setTitle("叶泽兵 2023308310415 学生信息管理系统 ver 0.1 ");
38. *// 设置背景图片*
39. *//        setContentPane(new JLabel(new ImageIcon("C:\\Users\\Administrator\\eclipse-workspace\\sb\\src\\sb\\img\\background.png")));*
41. jp1 = new JPanel();
42. jtf = new JTextField(10);
43. jb1 = new JButton("查询");
44. jb1.addActionListener(this);
45. jl1 = new JLabel("请输入名字：");
46. jp1.add(jl1);
47. jp1.add(jtf);
48. jp1.add(jb1);
49. jb2 = new JButton("添加");
51. jb2.addActionListener(this);
52. jb3 = new JButton("修改");
53. jb3.setIcon(new ImageIcon("C:\\Users\\Administrator\\eclipse-workspace\\sb\\src\\sb\\img\\bug.png")); *// 使用绝对路径*
54. jb3.addActionListener(this);
55. jb4 = new JButton("删除");
57. jb4.addActionListener(this);
59. jb5 = new JButton("清除查询"); *// 创建“清除查询”按钮*
60. jb5.addActionListener(this);
61. jp2 = new JPanel();
62. jp2.add(jb2);
63. jp2.add(jb3);
64. jp2.add(jb4);
65. jp1.add(jb5); *// 添加清除查询按钮到面板jp1(上部面板)*
66. *// 创建模型对象*
67. sm = new StuModel();
68. *// 初始化*
69. jt = new JTable(sm);
70. *// 添加鼠标点击事件监听器*
71. jt.addMouseListener(new MouseAdapter() {
72. @Override
73. public void mouseClicked(MouseEvent e) {
74. *// 监听双击事件*
75. if (e.getClickCount() == 2) {
76. int row = jt.getSelectedRow();
77. if (row != -1) {
78. *// 获取选中行的学号*
79. String stuID = (String) sm.getValueAt(row, 0);
80. handleStudentScore(stuID);
81. }
82. }
83. }
84. });
85. jsp = new JScrollPane(jt);
86. *//        {*
87. *//            // 自定义paintComponent以设置背景*
88. *//            @Override*
89. *//            protected void paintComponent(Graphics g) {*
90. *//                super.paintComponent(g);*
91. *//                ImageIcon bgIcon = new ImageIcon("C:\\Users\\Administrator\\eclipse-workspace\\sb\\src\\sb\\img\\background.png");*
92. *//                Image bgImage = bgIcon.getImage();*
93. *//                g.drawImage(bgImage, 0, 0, getWidth(), getHeight(), this);*
94. *//            }*
95. *//        };*
96. *// 将jsp放入到jframe中*
97. this.add(jsp);
98. this.add(jp1, "North");
99. this.add(jp2, "South");
100. this.setSize(600, 400);
101. *// this.setLocation(300, 200);*
102. this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);
103. this.setVisible(true);
104. }
105. *// 处理学生成绩的方法*
106. private void handleStudentScore(String stuID) {
107. Connection ct = null;
108. PreparedStatement ps = null;
109. ResultSet rs = null;
110. try {
111. Class.forName("com.mysql.cj.jdbc.Driver");
112. String url = "jdbc:mysql://localhost:3306/stuinfo?useUnicode=true&characterEncoding=utf8&nullCatalogMeansCurrent=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC";
113. String user = "root";
114. String passwd = "123456";
115. ct = DriverManager.getConnection(url, user, passwd);
116. String sql = "SELECT \* FROM stuScore WHERE stuID = ?";
117. ps = ct.prepareStatement(sql);
118. ps.setString(1, stuID);
119. rs = ps.executeQuery();
120. if (rs.next()) {
121. *// 成绩信息存在，打开浏览窗口*
122. double stuSc = rs.getDouble("stuSc");
123. double stuScno = rs.getDouble("stuScno");
124. String stuNt = rs.getString("stuNt");
125. ScoreViewDialog scoreViewDialog = new ScoreViewDialog(this, stuID, stuSc, stuScno, stuNt);
126. scoreViewDialog.setVisible(true);
127. } else {
128. *// 没有成绩记录，询问是否添加*
129. int option = JOptionPane.showConfirmDialog(this, "该学生没有成绩记录，是否添加？", "添加成绩", JOptionPane.YES\_NO\_OPTION);
130. if (option == JOptionPane.YES\_OPTION) {
131. if (performAuthentication()) { *// 验证身份*
132. ScoreEditDiag scoreEditDiag = new ScoreEditDiag(this, "添加成绩", true, stuID);
133. scoreEditDiag.setVisible(true);
134. }
135. }
136. }
137. } catch (Exception e) {
138. e.printStackTrace();
139. JOptionPane.showMessageDialog(this, "查询成绩时出错", "错误", JOptionPane.ERROR\_MESSAGE);
140. } finally {
141. try {
142. if (rs != null) rs.close();
143. if (ps != null) ps.close();
144. if (ct != null) ct.close();
145. } catch (Exception e) {
146. e.printStackTrace();
147. }
148. }
149. }
150. public void actionPerformed(ActionEvent arg0) {
151. *// 判断是哪个按钮被点击*
152. if (arg0.getSource() == jb1) {
153. System.out.println("用户希望查询");
154. *// 因为把对表的数据封装到StuModel中，可以比较简单的完成查询*
155. String name = this.jtf.getText().trim();
156. if (name.isEmpty()) { *// 如果输入框为空，提示用户*
157. JOptionPane.showMessageDialog(this, "请输入名字以进行查询！");
158. return;
159. }
160. *// 写一个sql语句*
161. String sql = "select \* from stu where stuName = '" + name + "' ";
162. *// 构建一个数据模型类，并更新*
163. sm = new StuModel(sql);
164. *// 更新jtable*
165. jt.setModel(sm);
166. } else if (arg0.getSource() == jb5) { *// 清除查询操作*
167. System.out.println("用户希望清除查询并显示所有记录");
168. *// 清空文本框*
169. jtf.setText("");
170. *// 重新加载所有数据*
171. sm = new StuModel();
172. jt.setModel(sm);
173. } else if (arg0.getSource() == jb2) {
174. *// 添加前进行身份验证*
175. if (performAuthentication()) {
176. System.out.println("添加...");
177. StuAddDiag sa = new StuAddDiag(this, "添加学生", true);
178. *// 重新再获得新的数据模型,*
179. sm = new StuModel();
180. jt.setModel(sm);
181. }
182. } else if (arg0.getSource() == jb4) {
183. *// 删除前进行身份验证*
184. if (performAuthentication()) {
185. *// 删除记录前进行确认*
186. int confirm = JOptionPane.showConfirmDialog(this, "此操作将永久删除该记录，是否继续？", "确认删除", JOptionPane.YES\_NO\_OPTION);
187. if (confirm == JOptionPane.YES\_OPTION) {
188. *// 执行删除操作*
189. int rowNum = this.jt.getSelectedRow();
190. if (rowNum == -1) {
191. JOptionPane.showMessageDialog(this, "请选中一行");
192. return;
193. }
194. *// 获取学生ID*
195. String stuId = (String) sm.getValueAt(rowNum, 0);
196. System.out.println("Id： " + stuId);
197. *// 删除学生记录和相应的成绩信息*
198. deleteStudentRecord(stuId);
199. *// 更新数据模型*
200. sm = new StuModel();
201. jt.setModel(sm);
202. }
203. }
204. } else if (arg0.getSource() == jb3) {
205. *// 修改前进行身份验证*
206. if (performAuthentication()) {
207. System.out.println("用户希望修改");
208. int rowNum = this.jt.getSelectedRow();
209. if (rowNum == -1) {
210. JOptionPane.showMessageDialog(this, "请选择一行");
211. return;
212. }
213. *// 显示修改对话框*
214. System.out.println("用户已经选中待修改行");
215. StuUpDiag su = new StuUpDiag(this, "学生信息修改", true, sm, rowNum);
216. *// 更新数据模型*
217. sm = new StuModel();
218. jt.setModel(sm);
219. }
220. }
221. }
222. *// 执行身份验证的方法*
223. private boolean performAuthentication() {
224. *// 创建身份验证对话框*
225. LoginDialog loginDialog = new LoginDialog(this);
226. loginDialog.setVisible(true);
227. return loginDialog.isAuthenticated();
228. }
229. *// 删除学生记录和成绩信息的方法*
230. private void deleteStudentRecord(String stuId) {
231. try {
232. Class.forName("com.mysql.cj.jdbc.Driver");
233. String url = "jdbc:mysql://localhost:3306/stuinfo?useUnicode=true&characterEncoding=utf8&nullCatalogMeansCurrent=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC";
234. String user = "root";
235. String passwd = "123456";
236. Connection ct = DriverManager.getConnection(url, user, passwd);
237. *// 删除stu表中的记录*
238. PreparedStatement ps1 = ct.prepareStatement("DELETE FROM stu WHERE stuId = ?");
239. ps1.setString(1, stuId);
240. ps1.executeUpdate();
241. *// 删除stuScore表中的记录*
242. PreparedStatement ps2 = ct.prepareStatement("DELETE FROM stuScore WHERE stuId = ?");
243. ps2.setString(1, stuId);
244. ps2.executeUpdate();
245. ps1.close();
246. ps2.close();
247. ct.close();
248. JOptionPane.showMessageDialog(this, "记录已删除", "成功", JOptionPane.INFORMATION\_MESSAGE);
249. } catch (Exception e) {
250. e.printStackTrace();
251. JOptionPane.showMessageDialog(this, "删除记录时出错", "错误", JOptionPane.ERROR\_MESSAGE);
252. }
253. }
254. }
255. 3.2 类StuModel  
     *//filename:StuModel*
256. package sb;
257. import java.sql.Connection;
258. import java.sql.DriverManager;
259. import java.sql.ResultSet;
260. import java.sql.Statement;
261. import java.util.Vector;
262. import javax.swing.table.AbstractTableModel;
263. public class StuModel extends AbstractTableModel {
264. *// 继承TableModel抽象类，实现学生模型*
265. *// 在rowData中存放行数据，在columnNames存放列名*
266. Vector rowData, columnNames;
267. *// 定义连接数据库的变量*
268. *// 进行初始化，置为空*
269. Statement stat = null;
270. Connection ct = null;
271. ResultSet rs = null;
272. *// 初始化*
273. public void init(String sql) {
274. if (sql.equals("")) {
275. *// 初始化时显示所有的存放在数据库中的数据*
276. sql = "select \* from stu";
277. }
278. *// 中间部分*
279. *// 设置列名*
280. columnNames = new Vector();*//创建向量*
281. columnNames.add("学号");
282. columnNames.add("名字");
283. columnNames.add("性别");
284. columnNames.add("年龄");
285. columnNames.add("专业");
286. columnNames.add("院系");
287. *// rowData存放多行*
288. rowData = new Vector();
289. try {
290. *// 1.加载驱动*
291. *// 8版本MySQL使用的是"com.mysql.cj.jdbc.Driver"，5版本使用的是"com.mysql.jdbc.Driver".*
292. Class.forName("com.mysql.cj.jdbc.Driver");
293. System.out.println("加载成功");*//控制台提示*
294. *// 2.连接数据库*
295. *// 定义几个常量*
296. *// 8版本需要加"?useUnicode=true&characterEncoding=utf8&nullCatalogMeansCurrent=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC"*
297. *// 其中需要设置nullCatalogMeansCurrent=true，出现这种问题的原因是mysql版本问题，mysql8.xxx以上驱动会出现这个问题，下图是我原mysql配置，是8.0.16的。*
298. *// 一开始因为使用的版本不一致导致总是连接不成功。后来驱动版本与数据库版本都统一之后连接成功。*
299. String url = "jdbc:mysql://localhost:3306/stuinfo?useUnicode=true&characterEncoding=utf8&nullCatalogMeansCurrent=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC";
300. String user = "root";  *// 账号*
301. String passwd = "123456";  *// 密码*
302. ct = DriverManager.getConnection(url, user, passwd);  *// connection*
303. stat = ct.createStatement(); *// 创建stat对象，prepare statement*
304. rs = stat.executeQuery(sql); *// 查询结果*
305. while (rs.next()) {
306. Vector hang = new Vector();
307. hang.add(rs.getString(1));
308. hang.add(rs.getString(2));
309. hang.add(rs.getString(3));
310. hang.add(rs.getInt(4));
311. hang.add(rs.getString(5));
312. hang.add(rs.getString(6));
313. *// 加入到rowData中*
314. rowData.add(hang);
315. }
316. } catch (Exception e) {
317. *// 捕获异常*
318. e.printStackTrace();
319. } finally {
320. try {
321. if (rs != null) {
322. rs.close();
323. rs = null;
324. }
325. if (stat != null) {
326. stat.close();
327. stat = null;
328. }
329. if (ct != null) {
330. ct.close();
331. ct = null;
332. }
333. } catch (Exception e) {
334. e.printStackTrace();
335. }
336. }
337. }
338. *// 增加学生函数*
339. public void addStu(String sql) {
340. *// 根据用户输入的sql语句，完成添加任务*
341. }
342. *// 构造函数，传递的sql语句获取数据模型*
343. public StuModel(String sql) {
344. this.init(sql);
345. }
346. *// 构造学生模型，并进行初始化*
347. public StuModel() {
348. this.init("");
349. }
350. *// 获取行数*
351. public int getRowCount() {
352. return this.rowData.size();
353. }
354. *// 获取列数*
355. public int getColumnCount() {
356. return this.columnNames.size();
357. }
358. *// 获取目标行及目标列的数据*
359. public Object getValueAt(int row, int column) {
360. return ((Vector) (this.rowData.get(row))).get(column);
361. }
362. *// 获取属性字段名字*
363. public String getColumnName(int column) {
364. return (String) this.columnNames.get(column);
365. }
366. @Override
367. public boolean isCellEditable(int rowIndex, int columnIndex) {
368. *// 如果需要让表格中的某些单元格可编辑，可以在此方法中返回true*
369. return false;
370. }
371. @Override
372. public void setValueAt(Object value, int row, int column) {
373. *// 更新rowData中指定单元格的数据*
374. ((Vector) rowData.get(row)).setElementAt(value, column);
376. *// 通知监听器单元格数据已更改*
377. fireTableCellUpdated(row, column);
378. }
379. }

### 3.3类StuAddDiag

1. package sb;
2. import javax.swing.\*;
3. import java.awt.\*;
4. import java.awt.event.ActionEvent;
5. import java.awt.event.ActionListener;
6. import java.sql.Connection;
7. import java.sql.DriverManager;
8. import java.sql.PreparedStatement;
9. import java.sql.SQLIntegrityConstraintViolationException;
10. import java.sql.SQLException;
11. public class StuAddDiag extends JDialog implements ActionListener {
12. *// 定义需要的 Swing 组件*
13. JLabel jl1, jl2, jl3, jl4, jl5, jl6;
14. JTextField jf1, jf2, jf3, jf4, jf5, jf6;
15. JPanel jp1, jp2, jp3;
16. JButton jb1, jb2;
17. *// 构造函数，参数 owner 代表父窗口，title 是窗口的名字，modal 指定是模态窗口还是非模态窗口*
18. public StuAddDiag(Frame owner, String title, boolean modal) {
19. *// 调用父类构造函数*
20. super(owner, title, modal);
21. *// 初始化标签*
22. jl1 = new JLabel("学号"); *// jl = Java Label*
23. jl2 = new JLabel("名字");
24. jl3 = new JLabel("性别");
25. jl4 = new JLabel("年龄");
26. jl5 = new JLabel("专业");
27. jl6 = new JLabel("院系");
28. *// 初始化文本框*
29. jf1 = new JTextField(10);
30. jf2 = new JTextField(10);
31. jf3 = new JTextField(10);
32. jf4 = new JTextField(10);
33. jf5 = new JTextField(10);
34. jf6 = new JTextField(10);
35. *// 初始化按钮*
36. jb1 = new JButton("添加"); *// jb = Java Button*
37. jb1.addActionListener(this); *// 为 "添加" 按钮注册监听器*
38. jb2 = new JButton("取消");
39. jb2.addActionListener(this); *// 为 "取消" 按钮注册监听器*
40. *// 初始化面板*
41. jp1 = new JPanel();
42. jp2 = new JPanel();
43. jp3 = new JPanel();
44. *// 设置面板布局*
45. jp1.setLayout(new GridLayout(6, 1)); *// 6 行 1 列的网格布局*
46. jp2.setLayout(new GridLayout(6, 1));
47. *// 将组件添加到面板*
48. jp1.add(jl1);
49. jp1.add(jl2);
50. jp1.add(jl3);
51. jp1.add(jl4);
52. jp1.add(jl5);
53. jp1.add(jl6);
54. jp2.add(jf1);
55. jp2.add(jf2);
56. jp2.add(jf3);
57. jp2.add(jf4);
58. jp2.add(jf5);
59. jp2.add(jf6);
60. *// 将按钮添加到第三个面板*
61. jp3.add(jb1);
62. jp3.add(jb2);
63. *// 将面板添加到对话框*
64. this.add(jp1, BorderLayout.WEST);
65. this.add(jp2, BorderLayout.CENTER);
66. this.add(jp3, BorderLayout.SOUTH);
67. *// 设置对话框大小和可见性*
68. this.setSize(300, 200);
69. this.setLocationRelativeTo(owner); *// 居中显示*
70. this.setVisible(true);
71. }
72. @Override
73. public void actionPerformed(ActionEvent e) {
74. *// 判断哪个按钮触发了事件*
75. if (e.getSource() == jb1) { *// 如果是“添加”按钮*
76. *// 执行数据验证*
77. if (!validateInput()) {
78. return; *// 如果验证失败，停止执行*
79. }
80. *// 开始数据库操作*
81. Connection ct = null;
82. PreparedStatement pstmt = null;
83. try {
84. *// 加载驱动*
85. Class.forName("com.mysql.cj.jdbc.Driver");
86. System.out.println("驱动加载成功");
87. *// 连接数据库*
88. String url = "jdbc:mysql://localhost:3306/stuinfo?useUnicode=true&characterEncoding=utf8&nullCatalogMeansCurrent=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC";
89. String user = "root";
90. String passwd = "123456";
91. ct = DriverManager.getConnection(url, user, passwd);
92. *// 插入语句*
93. String strsql = "INSERT INTO stu (stuID, stuName, stuSex, stuAge, stuJG, stuDept) VALUES (?, ?, ?, ?, ?, ?)";
94. pstmt = ct.prepareStatement(strsql);
95. *// 设置参数*
96. pstmt.setString(1, jf1.getText().trim());
97. pstmt.setString(2, jf2.getText().trim());
98. pstmt.setString(3, jf3.getText().trim());
99. pstmt.setInt(4, Integer.parseInt(jf4.getText().trim())); *// 年龄转为整数*
100. pstmt.setString(5, jf5.getText().trim());
101. pstmt.setString(6, jf6.getText().trim());
102. *// 执行更新*
103. int result = pstmt.executeUpdate();
104. if (result > 0) {
105. JOptionPane.showMessageDialog(this, "添加成功！");
106. } else {
107. JOptionPane.showMessageDialog(this, "添加失败，请重试！");
108. }
109. this.dispose(); *// 关闭对话框*
110. } catch (SQLIntegrityConstraintViolationException ex) {
111. *// 捕获并自定义处理 SQLIntegrityConstraintViolationException 异常*
112. JOptionPane.showMessageDialog(this, "错误: 学号已存在，请使用不同的学号", "数据库错误", JOptionPane.ERROR\_MESSAGE);
113. } catch (SQLException ex) {
114. *// 捕获其他 SQL 异常*
115. JOptionPane.showMessageDialog(this, "数据库连接错误: " + ex.getMessage(), "数据库错误", JOptionPane.ERROR\_MESSAGE);
116. } catch (Exception ex) {
117. *// 捕获所有其他异常*
118. JOptionPane.showMessageDialog(this, "发生未知错误: " + ex.getMessage(), "错误", JOptionPane.ERROR\_MESSAGE);
119. } finally {
120. *// 关闭资源*
121. try {
122. if (pstmt != null) {
123. pstmt.close();
124. }
125. if (ct != null) {
126. ct.close();
127. }
128. } catch (Exception ex) {
129. ex.printStackTrace();
130. }
131. }
132. } else if (e.getSource() == jb2) { *// 如果是“取消”按钮*
133. this.dispose(); *// 关闭对话框*
134. }
135. }
136. *// 输入验证方法*
137. private boolean validateInput() {
138. *// 检查是否有任何输入为空*
139. if (jf1.getText().trim().isEmpty() || jf2.getText().trim().isEmpty() || jf3.getText().trim().isEmpty() ||
140. jf4.getText().trim().isEmpty() || jf5.getText().trim().isEmpty() || jf6.getText().trim().isEmpty()) {
141. JOptionPane.showMessageDialog(this, "所有字段均不能为空！");
142. return false;
143. }
144. *// 检查学号是否为13位的阿拉伯数字*
145. String stuID = jf1.getText().trim();
146. if (!stuID.matches("\\d{13}")) {
147. JOptionPane.showMessageDialog(this, "学号必须是13位的数字！");
148. return false;
149. }
150. *// 检查性别是否合法*
151. String stuSex = jf3.getText().trim();
152. if (!stuSex.equals("男") && !stuSex.equals("女")) {
153. JOptionPane.showMessageDialog(this, "性别只能为'男'或'女'！");
154. return false;
155. }
156. *// 检查年龄是否合法*
157. try {
158. int stuAge = Integer.parseInt(jf4.getText().trim());
159. if (stuAge < 0 || stuAge > 150) {
160. JOptionPane.showMessageDialog(this, "年龄必须在0到150之间！");
161. return false;
162. }
163. } catch (NumberFormatException nfe) {
164. JOptionPane.showMessageDialog(this, "年龄必须是有效的整数！");
165. return false;
166. }
167. return true; *// 输入合法*
168. }
169. }

### 3.4 类StuUpDiag

1. *//filename: StuUpDiag.java*
2. package sb;
3. import java.awt.\*;
4. import java.awt.event.\*;
5. import javax.swing.\*;
6. import java.sql.\*;
7. public class StuUpDiag extends JDialog implements ActionListener {
8. *// 定义组件*
9. JLabel jl1, jl2, jl3, jl4, jl5, jl6;
10. JTextField jf1, jf2, jf3, jf4, jf5, jf6;
11. JButton jb1, jb2;
12. JPanel jp1, jp2, jp3;
13. *// 添加新成员变量*
14. private StuModel stuModel; *// 用于存储传入的 StuModel 实例*
15. private int selectedRow;   *// 用于存储传入的选中行号*
16. *// 构造函数定义，与 Main 类中的调用匹配*
17. public StuUpDiag(Main owner, String title, boolean modal, StuModel model, int row) {
18. *// 调用父类 JDialog 的构造函数，设置所有者、标题和模态选项*
19. super(owner, title, modal);
21. *// 初始化传入的参数*
22. this.stuModel = model;
23. this.selectedRow = row;
24. *// 初始化面板和标签*
25. jp1 = new JPanel();
26. jp2 = new JPanel();
27. jp3 = new JPanel();
28. jl1 = new JLabel("学号");
29. jl2 = new JLabel("名字");
30. jl3 = new JLabel("性别");
31. jl4 = new JLabel("年龄");
32. jl5 = new JLabel("籍贯");
33. jl6 = new JLabel("院系");
34. *// 初始化文本输入框*
35. jf1 = new JTextField(10);
36. jf2 = new JTextField(10);
37. jf3 = new JTextField(10);
38. jf4 = new JTextField(10);
39. jf5 = new JTextField(10);
40. jf6 = new JTextField(10);
41. *// 初始化按钮*
42. jb1 = new JButton("修改");
43. jb2 = new JButton("取消");
44. *// 初始化文本字段*
45. initializeFields();
46. *// 设置布局管理器*
47. jp1.setLayout(new GridLayout(6, 1));
48. jp2.setLayout(new GridLayout(6, 1));
49. *// 添加组件到面板*
50. jp1.add(jl1);
51. jp1.add(jl2);
52. jp1.add(jl3);
53. jp1.add(jl4);
54. jp1.add(jl5);
55. jp1.add(jl6);
56. jp2.add(jf1);
57. jp2.add(jf2);
58. jp2.add(jf3);
59. jp2.add(jf4);
60. jp2.add(jf5);
61. jp2.add(jf6);
62. *// 添加按钮到第三个面板*
63. jp3.add(jb1);
64. jp3.add(jb2);
65. *// 添加事件监听器*
66. jb1.addActionListener(this);
67. jb2.addActionListener(this);
68. *// 设置主窗口布局*
69. this.add(jp1, BorderLayout.WEST);
70. this.add(jp2, BorderLayout.CENTER);
71. this.add(jp3, BorderLayout.SOUTH);
72. *// 设置窗口大小和可见性*
73. this.setSize(300, 200);
74. this.setVisible(true);
75. }
76. *// 初始化文本字段*
77. private void initializeFields() {
78. *// 从 StuModel 中获取数据，并填充到文本字段*
79. jf1.setText(stuModel.getValueAt(selectedRow, 0).toString()); *// 学号*
80. jf2.setText(stuModel.getValueAt(selectedRow, 1).toString()); *// 名字*
81. jf3.setText(stuModel.getValueAt(selectedRow, 2).toString()); *// 性别*
82. jf4.setText(stuModel.getValueAt(selectedRow, 3).toString()); *// 年龄*
83. jf5.setText(stuModel.getValueAt(selectedRow, 4).toString()); *// 籍贯*
84. jf6.setText(stuModel.getValueAt(selectedRow, 5).toString()); *// 院系*
85. }
86. *// 实现 ActionListener 接口的方法*
87. @Override
88. public void actionPerformed(ActionEvent e) {
89. *// 如果点击了修改按钮*
90. if (e.getSource() == jb1) {
91. Connection ct = null;
92. PreparedStatement pstmt = null;
93. *// 获取用户输入的数据*
94. String stuID = jf1.getText().trim();
95. String stuName = jf2.getText().trim();
96. String stuSex = jf3.getText().trim();
97. String stuAgeStr = jf4.getText().trim();
98. String stuJG = jf5.getText().trim();
99. String stuDept = jf6.getText().trim();
100. *// 输入验证*
101. if (stuID.isEmpty() || stuName.isEmpty() || stuSex.isEmpty() || stuAgeStr.isEmpty() || stuJG.isEmpty() || stuDept.isEmpty()) {
102. JOptionPane.showMessageDialog(this, "所有字段均不能为空！");
103. return;
104. }
105. *// 性别验证*
106. if (!stuSex.equals("男") && !stuSex.equals("女")) {
107. JOptionPane.showMessageDialog(this, "性别只能为'男'或'女'！");
108. return;
109. }
110. *// 年龄验证*
111. int stuAge = 0;
112. try {
113. stuAge = Integer.parseInt(stuAgeStr);
114. if (stuAge < 0 || stuAge > 150) {
115. JOptionPane.showMessageDialog(this, "年龄必须在0到150之间！");
116. return;
117. }
118. } catch (NumberFormatException nfe) {
119. JOptionPane.showMessageDialog(this, "年龄必须是有效的整数！");
120. return;
121. }
122. try {
123. *// 加载驱动*
124. Class.forName("com.mysql.cj.jdbc.Driver");
125. System.out.println("加载成功");
126. *// 连接数据库*
127. String url = "jdbc:mysql://localhost:3306/stuinfo?useUnicode=true&characterEncoding=utf8&nullCatalogMeansCurrent=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC";
128. String user = "root";
129. String passwd = "123456";
130. ct = DriverManager.getConnection(url, user, passwd);
131. *// 创建 SQL 更新语句*
132. String strsql = "UPDATE stu SET stuName = ?, stuSex = ?, stuAge = ?, stuJG = ?, stuDept = ? WHERE stuID = ?";
133. pstmt = ct.prepareStatement(strsql);
134. *// 设置参数*
135. pstmt.setString(1, stuName); *// 设置名字*
136. pstmt.setString(2, stuSex); *// 设置性别*
137. pstmt.setInt(3, stuAge); *// 设置年龄*
138. pstmt.setString(4, stuJG); *// 设置籍贯*
139. pstmt.setString(5, stuDept); *// 设置院系*
140. pstmt.setString(6, stuID); *// 设置学号*
141. *// 执行更新*
142. int updateCount = pstmt.executeUpdate();
143. if (updateCount > 0) {
144. JOptionPane.showMessageDialog(this, "更新成功！");
145. } else {
146. JOptionPane.showMessageDialog(this, "更新失败：学号未找到！");
147. }
148. *// 更新表格中的数据*
149. stuModel.setValueAt(stuName, selectedRow, 1);
150. stuModel.setValueAt(stuSex, selectedRow, 2);
151. stuModel.setValueAt(stuAge, selectedRow, 3);
152. stuModel.setValueAt(stuJG, selectedRow, 4);
153. stuModel.setValueAt(stuDept, selectedRow, 5);
154. *// 关闭学生对话框*
155. this.dispose();
156. } catch (Exception arg1) {
157. arg1.printStackTrace();
158. } finally {
159. try {
160. if (pstmt != null) {
161. pstmt.close();
162. pstmt = null;
163. }
164. if (ct != null) {
165. ct.close();
166. ct = null;
167. }
168. } catch (Exception arg2) {
169. arg2.printStackTrace();
170. }
171. }
172. } else if (e.getSource() == jb2) {
173. *// 如果点击了取消按钮，关闭对话框*
174. this.dispose();
175. }
176. }
177. }
178. *//package SchoolProject.src;*
179. */\**
180. \* 修改学生
181. \*/

### 3.5 类StuScoreEditDiag

1. *// filename : ScoreEditDiag.java*
2. package sb;
3. import javax.swing.\*;
4. import java.awt.\*;
5. import java.awt.event.ActionEvent;
6. import java.awt.event.ActionListener;
7. import java.sql.Connection;
8. import java.sql.DriverManager;
9. import java.sql.PreparedStatement;
10. public class ScoreEditDiag extends JDialog {
11. private JTextField tfStuID, tfStuSc, tfStuScno, tfStuNt;
12. private JButton btnSave, btnCancel;
13. private String stuID;
14. public ScoreEditDiag(JFrame parent, String title, boolean modal, String stuID) {
15. super(parent, title, modal);
16. this.stuID = stuID;
17. *// 创建UI组件*
18. JPanel panel = new JPanel(new GridLayout(5, 2));
19. panel.add(new JLabel("学生ID："));
20. tfStuID = new JTextField(stuID);
21. tfStuID.setEditable(false);
22. panel.add(tfStuID);
23. panel.add(new JLabel("已修学分："));
24. tfStuSc = new JTextField();
25. panel.add(tfStuSc);
26. panel.add(new JLabel("未通过学分："));
27. tfStuScno = new JTextField();
28. panel.add(tfStuScno);
29. panel.add(new JLabel("备注："));
30. tfStuNt = new JTextField();
31. panel.add(tfStuNt);
32. btnSave = new JButton("保存");
33. btnCancel = new JButton("取消");
34. panel.add(btnSave);
35. panel.add(btnCancel);
36. getContentPane().add(panel, BorderLayout.CENTER);
37. btnSave.addActionListener(new ActionListener() {
38. public void actionPerformed(ActionEvent e) {
39. saveScore();
40. }
41. });
42. btnCancel.addActionListener(new ActionListener() {
43. public void actionPerformed(ActionEvent e) {
44. dispose();
45. }
46. });
47. setSize(300, 250);
48. setLocationRelativeTo(parent);
49. }
50. private void saveScore() {
51. try {
52. double stuSc = Double.parseDouble(tfStuSc.getText().trim());
53. double stuScno = Double.parseDouble(tfStuScno.getText().trim());
54. String stuNt = tfStuNt.getText().trim();
55. Class.forName("com.mysql.cj.jdbc.Driver");
56. String url = "jdbc:mysql://localhost:3306/stuinfo?useUnicode=true&characterEncoding=utf8&nullCatalogMeansCurrent=true&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC";
57. String user = "root";
58. String passwd = "123456";
59. Connection ct = DriverManager.getConnection(url, user, passwd);
60. PreparedStatement ps = ct.prepareStatement("REPLACE INTO stuScore (stuID, stuSc, stuScno, stuNt) VALUES (?, ?, ?, ?)");
61. ps.setString(1, stuID);
62. ps.setDouble(2, stuSc);
63. ps.setDouble(3, stuScno);
64. ps.setString(4, stuNt);
65. ps.executeUpdate();
66. ps.close();
67. ct.close();
68. JOptionPane.showMessageDialog(this, "成绩已保存", "成功", JOptionPane.INFORMATION\_MESSAGE);
69. dispose();
70. } catch (NumberFormatException e) {
71. JOptionPane.showMessageDialog(this, "请输入有效的数字", "输入错误", JOptionPane.ERROR\_MESSAGE);
72. } catch (Exception e) {
73. e.printStackTrace();
74. JOptionPane.showMessageDialog(this, "保存成绩时出错", "错误", JOptionPane.ERROR\_MESSAGE);
75. }
76. }
77. }

### 3.6类 StuScoreViewDiag

1. *// filename : ScoreViewDialog.java*
2. package sb;
3. import javax.swing.\*;
4. import java.awt.\*;
5. import java.awt.event.ActionEvent;
6. import java.awt.event.ActionListener;
7. import java.sql.\*;
8. public class ScoreViewDialog extends JDialog {
9. private JLabel lblStuID, lblStuSc, lblStuScno, lblStuNt;
10. private JButton btnEdit, btnClose;
11. private String stuID;
12. private double stuSc;
13. private double stuScno;
14. private String stuNt;
15. public ScoreViewDialog(JFrame parent, String stuID, double stuSc, double stuScno, String stuNt) {
16. super(parent, "成绩浏览", true);
17. this.stuID = stuID;
18. this.stuSc = stuSc;
19. this.stuScno = stuScno;
20. this.stuNt = stuNt;
21. *// 创建UI组件*
22. JPanel panel = new JPanel(new GridLayout(5, 2));
23. panel.add(new JLabel("学生ID："));
24. lblStuID = new JLabel(stuID);
25. panel.add(lblStuID);
26. panel.add(new JLabel("已修学分："));
27. lblStuSc = new JLabel(String.valueOf(stuSc));
28. panel.add(lblStuSc);
29. panel.add(new JLabel("未通过学分："));
30. lblStuScno = new JLabel(String.valueOf(stuScno));
31. panel.add(lblStuScno);
32. panel.add(new JLabel("备注："));
33. lblStuNt = new JLabel(stuNt);
34. panel.add(lblStuNt);
35. btnEdit = new JButton("编辑");
36. btnClose = new JButton("关闭");
37. panel.add(btnEdit);
38. panel.add(btnClose);
39. getContentPane().add(panel, BorderLayout.CENTER);
40. btnEdit.addActionListener(new ActionListener() {
41. public void actionPerformed(ActionEvent e) {
42. if (performAuthentication(parent)) { *// 使用parent作为参数进行身份验证*
43. ScoreEditDiag scoreEditDiag = new ScoreEditDiag(parent, "编辑成绩", true, stuID);
44. scoreEditDiag.setVisible(true);
45. dispose();
46. }
47. }
48. });
49. btnClose.addActionListener(new ActionListener() {
50. public void actionPerformed(ActionEvent e) {
51. dispose();
52. }
53. });
54. setSize(300, 250);
55. setLocationRelativeTo(parent);
56. }
57. private boolean performAuthentication(JFrame parent) {
58. LoginDialog loginDialog = new LoginDialog(parent); *// 使用正确的构造函数*
59. loginDialog.setVisible(true);
60. return loginDialog.isAuthenticated();
61. }
62. }

### 3.7类 管理员验证 LoginDiag

1. *// filename : LoginDialog.java*
2. package sb;
3. import javax.swing.\*;
4. import java.awt.\*;
5. import java.awt.event.ActionEvent;
6. import java.awt.event.ActionListener;
7. import java.sql.Connection;
8. import java.sql.DriverManager;
9. import java.sql.PreparedStatement;
10. import java.sql.ResultSet;
11. public class LoginDialog extends JDialog {
12. private JTextField tfUsername;
13. private JPasswordField pfPassword;
14. private JButton btnLogin, btnCancel;
15. private boolean authenticated = false;
16. public LoginDialog(JFrame parent) {
17. super(parent, "身份验证", true);
18. *// 创建UI组件*
19. JPanel panel = new JPanel(new GridLayout(3, 2));
20. panel.add(new JLabel("用户名："));
21. tfUsername = new JTextField(20);
22. panel.add(tfUsername);
24. panel.add(new JLabel("密码："));
25. pfPassword = new JPasswordField(20);
26. panel.add(pfPassword);
27. btnLogin = new JButton("登录");
28. btnCancel = new JButton("取消");
29. panel.add(btnLogin);
30. panel.add(btnCancel);
31. getContentPane().add(panel, BorderLayout.CENTER);
32. btnLogin.addActionListener(new ActionListener() {
33. public void actionPerformed(ActionEvent e) {
34. String username = tfUsername.getText().trim();
35. String password = new String(pfPassword.getPassword()).trim();
36. if (authenticate(username, password)) {
37. authenticated = true;
38. dispose();
39. } else {
40. JOptionPane.showMessageDialog(LoginDialog.this, "用户名或密码错误", "身份验证失败", JOptionPane.ERROR\_MESSAGE);
41. }
42. }
43. });
44. btnCancel.addActionListener(new ActionListener() {
45. public void actionPerformed(ActionEvent e) {
46. dispose();
47. }
48. });
49. setSize(300, 150);
50. setLocationRelativeTo(parent);
51. }
52. public boolean isAuthenticated() {
53. return authenticated;
54. }
55. *// 执行数据库验证*
56. private boolean authenticate(String username, String password) {
57. try {
58. Class.forName("com.mysql.cj.jdbc.Driver");
59. String url = "jdbc:mysql://localhost:3306/users?useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC";
60. String user = "root";
61. String passwd = "123456";
62. Connection ct = DriverManager.getConnection(url, user, passwd);
63. PreparedStatement ps = ct.prepareStatement("SELECT \* FROM admin WHERE username = ? AND password = ?");
64. ps.setString(1, username);
65. ps.setString(2, password);
66. ResultSet rs = ps.executeQuery();
67. if (rs.next()) {
68. rs.close();
69. ps.close();
70. ct.close();
71. return true;
72. } else {
73. rs.close();
74. ps.close();
75. ct.close();
76. return false;
77. }
78. } catch (Exception e) {
79. e.printStackTrace();
80. JOptionPane.showMessageDialog(this, "数据库连接错误", "错误", JOptionPane.ERROR\_MESSAGE);
81. return false;
82. }
83. }
84. }