**成绩：**

《高级面向对象程序设计实验》

课程大作业

**学 期： 2021-2022学年第二学期**

**学 院： 信息工程学院**

**年 级 及 专 业：）**

**班 级：2班**

**姓 名：**

# 图书管理系统

1. **项目简述：**

**项目分工：。。。（系统设计+主要代码实现）+。。。（数据库设计+其他功能模块+运行测试+文档资料整理）**

该项目是一个通过Java和swing实现的图书管理系统的小程序，该小程序具有图书类别添加、图书类别维护、图书添加和图书维护的功能。

**使用软件：**

* IntelliJ IDEA 2020.3（Ultimate Edition）：编写Java项目代码。
* Navicat Premium 15.0.30 ：数据库可视化操作软件。

**运行环境：**

* JDK8
* MySQL 8.0.26

**第三方JAR 包：**

* mysql-connector-java-5.1.7-bin.jar

**功能描述：**

* 用数据库建立 1 或 2 个学生信息表
* 图书类别添加
* 图书类别维护
* 图书添加
* 图书维护
* 其他（如关于软件等）

要实现这些功能需要熟练使用数据库和 Java 编程语言，并且能够实现数据库和 Java 的连接。本课题涉及数据库、Java 等领域。通过本项目的设计实践，力求熟练掌握 GUI 程序设计、Java 操作后台数据库等概念。

**2.系统设计**

**2.1 数据库设计**

根据系统功能描述，确定建立 2 张表，表结构描述如下所示。

表1 （tb\_book）记录图书相关信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否为空 | 主键/外键 | 说明 |
| bId | int | NO | PRI | 图书ID，自动增长 |
| bBookName | varchar(40) | NO |  | 图书名称 |
| bAuthor | varchar(20) | NO |  | 图书作者 |
| bSex | varchar(10) | NO |  | 图书作者性别 |
| bPrice | float | NO |  | 图书价格 |
| bBookDscription | varchar(1000) | NO |  | 图书描述 |
| btId | int | NO | MUL | 图书类别ID |

表2 （tb\_booktype）记录图书类别相关信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否为空 | 主键/外键 | 说明 |
| btId | int | NO | PRI |  |
| btName | varchar(40) | NO |  |  |
| btDescription | varchar(1000) | NO |  |  |

**2.2 功能模块设计**

**2.2.1 通用功能模块**

JDBCUtils类

功能实现：连接数据库

**2.2.2 登录模块功能**

1. 界面类设计

功能组件:2 个文本框，4 个标签，2 个按钮

功能实现:用户信息的校对，进入系统

2.监听类设计

事件源:按钮

事件过程:对按钮添加监控，实现按钮事件。当用户点击登录按钮时，获取文本框中的信息，监视器调用接口中的方法，对用户信息进行校对。用户 信息正确，进入系统。当用户点击重置按钮时，会重置清除输入的用户名和密码。

**2.2.3 主界面模块功能**

1. 界面类设计

功能组件:1 个菜单条，1 个内容面板

功能实现:菜单栏的菜单选项，面板显示

2.监听类设计

事件源:按钮

事件过程:对按钮添加监控，实现按钮事件。当用户点击菜单按钮时，获取信息，监视器调用接口中的方法，将主界面的内容面板替换成图书/图书类别面板，并显示各个面板内容。

**2.2.4 图书类别添加模块功能**

1. 界面类设计

功能组件: 2 个文本框，3 个标签，2 个按钮

功能实现:图书类别信息添加

2.监听类设计

事件源:按钮

事件过程:对按钮添加监控，实现按钮事件。当用户点击菜单按钮时，获取信息，监视器调用接口中的方法，获取用户输入的内容，然后将其插入到数据库中的图书表中即可，这就是添加功能的实现。

**2.2.5 图书类别维护模块功能**

1. 界面类设计

功能组件: 1 个文本框，5 个标签，3 个按钮，1个表格

功能实现:图书类别信息维护即查询、修改、删除功能

2.监听类设计

事件源:按钮

事件过程:对按钮添加监控，实现按钮事件。当用户点击查询按钮时，获取信息，监视器调用接口中的方法，判断图书类别输入框是否为空并做处理。当用户点击修改按钮时，获取各输入框的内容，执行修改操作并对执行结果进行处理。当用户点击删除按钮时，获取要删除的id，执行删除操作，对删除结果进行处理。

**2.2.6 图书添加模块功能**

1. 界面类设计

功能组件: 5 个文本框，7 个标签，4个按钮，1个下拉框

功能实现:图书信息添加

2.监听类设计

事件源:按钮

事件过程:对按钮添加监控，实现按钮事件。当用户点击男/女性别按钮时，判断获取选项并保存记录。当用户点击添加按钮，获取输入的信息，完成所有图书信息的添加。当用户点击重置按钮，则清空用户的选择和输入

**2.2.7 图书维护模块功能**

1. 界面类设计

功能组件: 7 个文本框，11 个标签，5个按钮，2个下拉框、1个表格

功能实现:图书信息维护

2.监听类设计

事件源:按钮 ，鼠标

事件过程:对按钮添加监控，实现按钮事件。当用户选中表格某行后，判断显示表格内容。当用户点击查询按钮，表格显示查询到的图书信息。当用户点击修改按钮，获取用户输入的修改信息，对信息进行修改并返回结果。当用户点击删除按钮，获取图书ID，执行删除所有信息的操作，返回处理结果。两个重置按钮，第一个点击后，清空用户的输入，第二个按钮点击后，清空用户输入并刷新表格。

**2.2.8 其他界面及功能实现**

1. 界面类设计

功能组件: 2 个标签，1个按钮

功能实现:退出、弹窗信息提示

2.监听类设计

事件源:按钮、鼠标

事件过程:对按钮添加监控，实现按钮事件。当用户点击关闭按钮，系统响应，退出系统。对鼠标添加监控，实现鼠标事件，当用户点击链接位置，自动跳转网页链接。

**3.系统实现**

**3.1 通用功能模块实现**

JDBCUtils类:

图书管理系统的重要功能的实现，是以数据库作为依托。本系统采用的数据库是 MYSQL 数据库。通过 Java 提供的 JDBC 来与 MYSQL 数据库建立连接。具体方 法是通过 JDBC-数据库连接器。本系统采用的数据库连接器是 mysql-connector-java-8.0.20jar。

连接数据库代码如下:

1. public class JDBCUtils {
2. /\*\*
3. \* 加载驱动，并建立数据库连接
4. \*
5. \* @return 返回数据库连接
6. \* @throws SQLException 抛出SQLException
7. \* @throws ClassNotFoundException 抛出ClassNotFoundException
8. \*/
9. public static Connection getConnection() throws SQLException, ClassNotFoundException {
10. // 设置数据库驱动
11. Class.forName("com.mysql.jdbc.Driver");
12. // 数据库链接URL及数据库名称
13. String url = "jdbc:mysql://localhost:3306/Library\_System";
14. // 登录账户名
15. String username = "root";
16. // 登录账户密码
17. String password = "duxiaobai";
18. // 创建连接
19. Connection conn = DriverManager.getConnection(url, username, password);
20. return conn;
21. }

**3.2 登录模块实现**

本系统运行开始，向用户展现的是登录界面。如图 3-1 所示，在文本框中输入用户信息，然后单击登录按钮。系统执行便登录成功，会弹出登录成功的提示，进入到系统界面。如图 3-2。

在本项目中虽然使用了数据库，但是没有在登录中使用数据库创建用户表，所以，登录成功的唯一用户是定死的（即用户名：duxiaobai；密码：123456），通过用户输入的用户名和密码对系统既定的用户名密码进行比较，如果二者都为true即账户正确登录成功，否则登录失败。重置功能即清空两个输入框的内容。

图3-1 登录界面



图3-2 登录成功

用户未输入信息或输入的信息有误。系统也会给出相应的提示来提醒用户。如图 3-3 所示。

图 3-3 错误提示

登录界面及功能设计代码:

*public class* LogupFrame *extends* JFrame *implements ActionListener* {  
 *private* SimpleTools simpleTools = *new* SimpleTools();  
 *private* ComponentTools componentTools = *new* ComponentTools();  
 *private* JPanel panel;  
 *private* Box totalVBox, systemLabelBox, usernameBox, passwordBox, buttonBox;  
 *private* JLabel systemLabel, usernameLabel, passwordLabel;  
 *private* JTextField usernameTextField;  
 *private* JPasswordField passwordField;  
 *private* JButton logupButton, resetButton;  
  
 *public* LogupFrame() {  
 *this*.setTitle("登录");  
 *this*.setBounds(400, 400, 600, 500);  
  
 *this*.setContentPane(*this*.createLogupPane());  
  
 *this*.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 *this*.setVisible(*false*);  
  
 *// 为按钮注册事件* logupButton.addActionListener(*this*);  
 resetButton.addActionListener(*this*);  
 }  
  
 *public* JPanel createLogupPane() {  
 panel = *new* JPanel();  
 totalVBox = Box.*createVerticalBox*();  
  
 systemLabelBox = Box.*createHorizontalBox*();  
 systemLabel = *new* JLabel("图书管理系统");  
 systemLabel.setFont(*new* Font("微软雅黑", Font.BOLD, 40));  
 systemLabelBox.add(systemLabel);  
  
 usernameBox = Box.*createHorizontalBox*();  
 usernameLabel = *new* JLabel("用户名：");  
 usernameTextField = *new* JTextField(20);  
 usernameBox.add(usernameLabel);  
 usernameBox.add(Box.*createHorizontalStrut*(20));  
 usernameBox.add(usernameTextField);  
  
 passwordBox = Box.*createHorizontalBox*();  
 passwordLabel = *new* JLabel("密 码：");  
 passwordField = *new* JPasswordField(20);  
 passwordBox.add(passwordLabel);  
 passwordBox.add(Box.*createHorizontalStrut*(20));  
 passwordBox.add(passwordField);  
  
 buttonBox = Box.*createHorizontalBox*();  
 logupButton = *new* JButton("登录");  
 resetButton = *new* JButton("重置");  
 buttonBox.add(logupButton);  
 buttonBox.add(Box.*createHorizontalStrut*(80));  
 buttonBox.add(resetButton);  
  
 totalVBox.add(systemLabelBox);  
 totalVBox.add(Box.*createVerticalStrut*(80));  
 totalVBox.add(usernameBox);  
 totalVBox.add(Box.*createVerticalStrut*(80));  
 totalVBox.add(passwordBox);  
 totalVBox.add(Box.*createVerticalStrut*(80));  
 totalVBox.add(buttonBox);  
  
 panel.add(totalVBox);  
  
 *// 为标签设置图标  
 new* ComponentTools().setIcons(*new* JLabel[]{systemLabel, usernameLabel, passwordLabel},  
 *new* String[]{"src/bookManageSystem/images/logo.png", "src/bookManageSystem/images/userName.png",  
 "src" + "/bookManageSystem/images/password.png"});  
 *// 为按钮设置图标  
 new* ComponentTools().setIcons(*new* JButton[]{logupButton, resetButton}, *new* String[]{"src/bookManageSystem/images/login.png", "src/bookManageSystem/images/reset.png"});  
 *return* panel;  
 }  
  
 @Override  
 *public void* actionPerformed(ActionEvent e) {  
 *// 登录按钮的事件处理  
 if* (e.getSource() == logupButton) {  
 *// 判断输入框是否为空并做处理  
 if* (simpleTools.isEmpty(usernameTextField.getText()) && simpleTools.isEmpty(passwordField.getText())) {  
 *// 登录账户，这里没有调用数据库，而是使用默认唯一账户：（用户名：张三；密码：123456）  
 if* (usernameTextField.getText().equals("duxiaobai") && passwordField.getText().equals("123456")) {  
 *// 弹出提示框进行登录成功的提示  
 int* isOK = JOptionPane.*showConfirmDialog*(*null*, "恭喜您，登录成功！");  
 *// 如果用户点击了提示框中的确认按钮则进行下一步操作  
 if* (isOK == JOptionPane.OK\_OPTION) {  
 *// 重置输入框内容* componentTools.reset(usernameTextField, passwordField);  
 *// 当登录成功后，跳转到主界面  
 new* MainFrame().setVisible(*true*);  
 *// 登录成功后同时使登录界面不可见（即消失）  
 this*.setVisible(*false*);  
 } *else* {  
 *return*;  
 }  
 } *else* {  
 JOptionPane.*showMessageDialog*(*null*, "对不起，登录失败！");  
 }  
 } *else* {  
 JOptionPane.*showMessageDialog*(*null*, "不能为空！");  
 }  
 }  
 *// 重置按钮的事件处理  
 if* (e.getSource() == resetButton) {  
 *// 将输入框的内容置为空* componentTools.reset(usernameTextField, passwordField);  
 }  
 }  
  
}

**3.3 主界面模块实现**

主界面的中心面板是一张图片，而顶部是一个菜单条。如图3-4.



图 3-4 主界面

主界面实现及功能代码：

*public class* MainFrame *extends* JFrame *implements ActionListener* {  
 *private* ComponentTools componentTools = *new* ComponentTools();  
  
 *private* JMenuBar menuBar;  
 *private* JMenu bookTypeManageMenu, bookManageMenu, otherMenu;  
 *private* JMenuItem bookTypeAddMenuItem, bookTypeManageMenuItem, bookAddMenuItem, bookManageMenuItem, exitMenuItem,  
 aboutSoftMenuItem;  
  
 MainFrame() {  
 *this*.setTitle("图书管理系统");  
 *this*.setBounds(400, 400, 800, 700);  
  
 *// 为主界面添加菜单条  
 this*.setJMenuBar(*this*.createMenuBar());  
 *// 设置主界面的内容面板  
 this*.setContentPane(createMainPanel());  
 *this*.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 *// 设置默认不显示  
 this*.setVisible(*false*);  
  
 *// 为菜单条中的菜单项注册事件* bookTypeAddMenuItem.addActionListener(*this*);  
 bookTypeManageMenuItem.addActionListener(*this*);  
 bookAddMenuItem.addActionListener(*this*);  
 bookManageMenuItem.addActionListener(*this*);  
 exitMenuItem.addActionListener(*this*);  
 aboutSoftMenuItem.addActionListener(*this*);  
 }  
  
 */\*\*  
 \* 为主界面创建一个菜单条  
 \*  
 \** ***@return*** *返回菜单条  
 \*/  
 private* JMenuBar createMenuBar() {  
 *// 实例化一个菜单条对象* menuBar = *new* JMenuBar();  
  
 *// 【图书类别管理】菜单* bookTypeManageMenu = *new* JMenu("图书类别管理");  
 bookTypeAddMenuItem = *new* JMenuItem("图书类别添加");  
 bookTypeManageMenuItem = *new* JMenuItem("图书类别维护");  
 bookTypeManageMenu.add(bookTypeAddMenuItem);  
 bookTypeManageMenu.add(bookTypeManageMenuItem);  
 menuBar.add(bookTypeManageMenu);  
  
 *// 【图书管理】菜单* bookManageMenu = *new* JMenu("图书管理");  
 bookAddMenuItem = *new* JMenuItem("图书添加");  
 bookManageMenuItem = *new* JMenuItem("图书维护");  
 bookManageMenu.add(bookAddMenuItem);  
 bookManageMenu.add(bookManageMenuItem);  
 menuBar.add(bookManageMenu);  
  
 *// 【其他】菜单* otherMenu = *new* JMenu("其他");  
 exitMenuItem = *new* JMenuItem("退出");  
 aboutSoftMenuItem = *new* JMenuItem("关于软件");  
 otherMenu.add(exitMenuItem);  
 otherMenu.add(aboutSoftMenuItem);  
 menuBar.add(otherMenu);  
  
 *// 批量为菜单项设置图标* componentTools.setIcons(  
 *new* JMenuItem[]{bookTypeAddMenuItem, bookTypeManageMenuItem, bookAddMenuItem, bookManageMenuItem, exitMenuItem, aboutSoftMenuItem},  
 *new* String[]{"src/bookManageSystem/images/add.png", "src/bookManageSystem/images/edit.png", "src" +  
 "/bookManageSystem/images/add.png", "src/bookManageSystem/images/edit.png", "src" +  
 "/bookManageSystem/images/exit.png", "src/bookManageSystem/images/about.png"}  
 );  
  
 *return* menuBar;  
 }  
  
 */\*\*  
 \* 为主界面创建内容面板  
 \*  
 \** ***@return*** *返回一个面板  
 \*/  
 private* JPanel createMainPanel() {  
 *// 实例化一个面板* JPanel panel = *new* JPanel();  
 *// 设置面板中的布局方式* panel.setLayout(*new* BorderLayout());  
 JLabel label = *new* JLabel();  
 label.setHorizontalAlignment(*SwingConstants*.CENTER);  
 label.setVerticalAlignment(*SwingConstants*.CENTER);  
 ImageIcon ii = *new* ComponentTools().iconSize(*new* ImageIcon("src/bookManageSystem/images/bookmanagesystem.png"), 600, 400);  
 *// 设置一个图片标签* label.setIcon(ii);  
 *// 将标签添加到面板中* panel.add(label);  
 *return* panel;  
 }  
  
 @Override  
 *public void* actionPerformed(ActionEvent e) {  
 *// “图书类别添加”菜单项事件处理  
 if* (e.getSource() == bookTypeAddMenuItem) {  
 *// 将主界面的内容面板替换成图书类别添加面板  
 this*.setContentPane(*new* BookTypeAddPanel());  
 *// 并设置该面板显示  
 this*.setVisible(*true*);  
 }  
 *// “图书类别维护”菜单项事件处理  
 if* (e.getSource() == bookTypeManageMenuItem) {  
 *// 将主界面的内容面板替换成图书类别维护面板  
 this*.setContentPane(*new* BookTypeManagePanel());  
 *// 并设置该面板显示  
 this*.setVisible(*true*);  
 }  
 *// “图书添加”菜单项事件处理  
 if* (e.getSource() == bookAddMenuItem) {  
 *// 将主界面的内容面板替换成图书添加面板  
 this*.setContentPane(*new* BookAddPanel());  
 *// 并设置该面板显示  
 this*.setVisible(*true*);  
 }  
 *// “图书维护”菜单项事件处理  
 if* (e.getSource() == bookManageMenuItem) {  
 *// 将主界面的内容面板替换成图书维护面板  
 this*.setContentPane(*new* BookManagePanel());  
 *// 并设置该面板显示  
 this*.setVisible(*true*);  
 }  
 *// “退出”菜单项事件处理  
 if* (e.getSource() == exitMenuItem) {  
 *// 结束整个程序* System.*exit*(0);  
 }  
 *// “关于软件”菜单项事件处理  
 if* (e.getSource() == aboutSoftMenuItem) {  
 *// 设置关于软件提示框显示  
 new* AboutSoftDialog().setVisible(*true*);  
 }  
 }  
}

**3.4 图书类别添加模块实现**

获取用户输入的内容，然后将其插入到数据库中的图书表中即可，这就是添加功能的实现。如图3-5如果出错也会提示如3-6



图3-5 添加界面

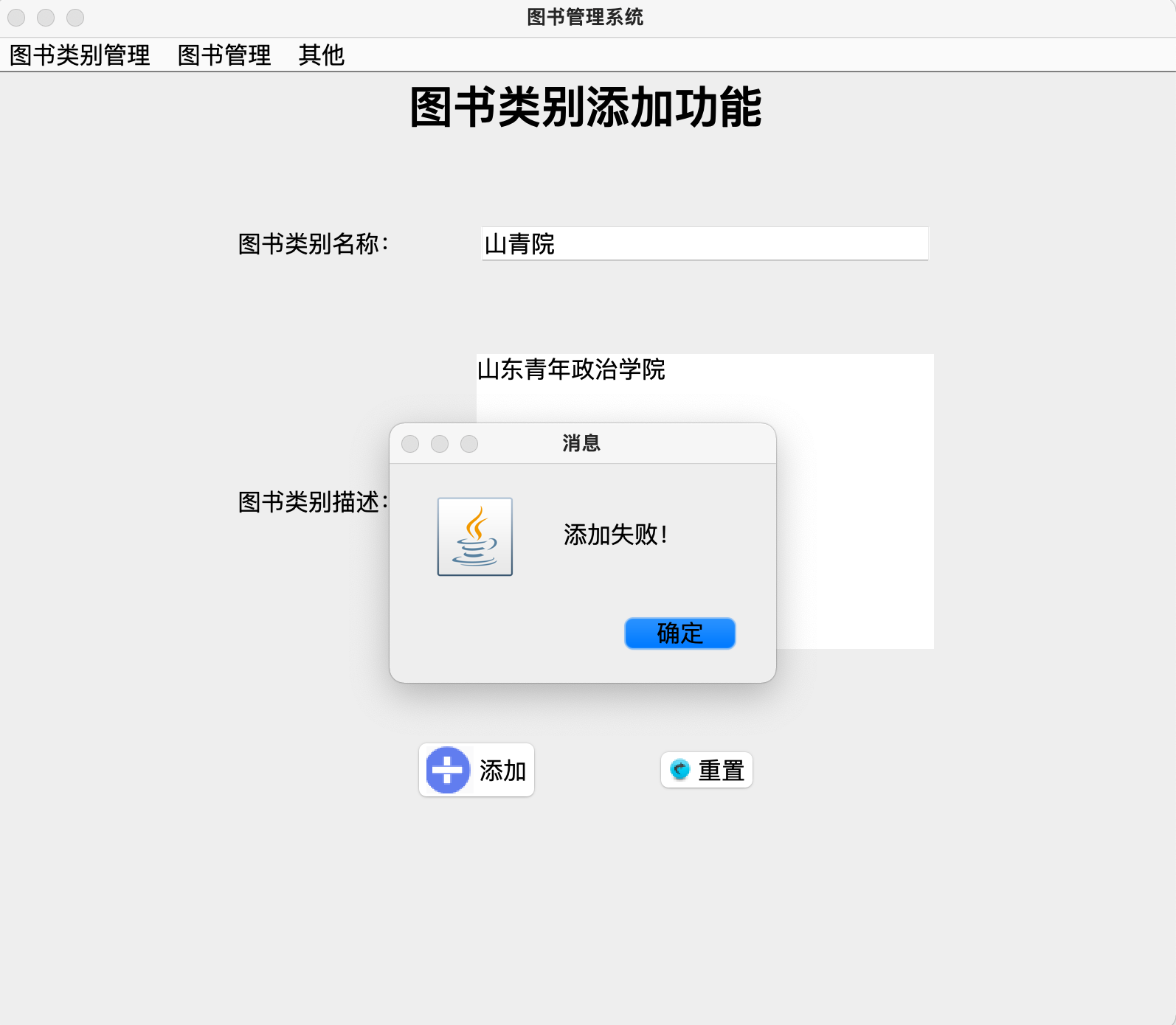


图 3-6 添加成功/失败

数据库操作实现代码：

*public class* BookTypeDao {  
  
 */\*\*  
 \* 操作结果：根据SQL语句执行数据库的增删改操作  
 \*  
 \** ***@param sql*** *SQL语句  
 \** ***@return*** *boolean 如果操作数据库成功返回true，否则返回false  
 \*/  
 public boolean* dataChange(String sql) {  
 *Connection* conn = *null*;  
 *Statement* stmt = *null*;  
 *try* {  
 *//获得数据的连接  
 new* JDBCUtils();  
 conn = JDBCUtils.*getConnection*();  
 *//获得Statement对象* stmt = conn.createStatement();  
 *//发送SQL语句并返回受影响行数  
 int* num = stmt.executeUpdate(sql);  
 *// 根据受影响行数判断是否操作成功  
 if* (num > 0){  
 *return true*;  
 }*else* {  
 *return false*;  
 }  
 }*catch*(Exception e)  
 { e.printStackTrace();  
 } *finally* {  
 JDBCUtils.*release*(stmt, conn);  
 }  
 *return false*;  
 }  
  
 */\*\*  
 \* 操作结果：根据参数sql获取数据库记录数据  
 \*  
 \** ***@param sql*** *SQL语句  
 \** ***@return*** *List 返回包含记录Records对象的集合  
 \*/  
 public List* getRecordsDataBySql(String sql) {  
 *Connection* conn = *null*;  
 *Statement* stmt = *null*;  
 *ResultSet* rs = *null*;  
 *List* list = *new* ArrayList();  
 *try* {  
 *//获得数据的连接* conn = JDBCUtils.*getConnection*();  
 *//获得Statement对象* stmt = conn.createStatement();  
 *//发送SQL语句并获取结果集* rs = stmt.executeQuery(sql);  
 *// 循环遍历结果集  
 while* (rs.next()) {  
 *// 实例化图书类别实体类并将结果集中的数据封装到实体类中* BookTypeBean bookTypeBean = *new* BookTypeBean();  
 bookTypeBean.setBookTypeId(rs.getInt(1));  
 bookTypeBean.setBookTypeName(rs.getString(2));  
 bookTypeBean.setBookTypeDescription(rs.getString(3));  
 *// 将实体类添加到集合中* list.add(bookTypeBean);  
 }  
 } *catch* (Exception e) {  
 e.printStackTrace();  
 } *finally* {  
 JDBCUtils.*release*(rs, stmt, conn);  
 }  
 *return* list;  
 }  
  
 */\*\*  
 \* 操作结果：将集合转换成数组  
 \*  
 \** ***@param list*** *集合  
 \** ***@return*** *String[][] 二维数组  
 \*/  
 public* String[][] ListToArray(*List*<BookTypeBean> list) {  
 String[][] array = *new* String[list.size()][3];  
 *for* (*int* i = 0; i < list.size(); i++) {  
 BookTypeBean l = list.get(i);  
 array[i][0] = String.*valueOf*(l.getBookTypeId());  
 array[i][1] = l.getBookTypeName();  
 array[i][2] = l.getBookTypeDescription();  
 }  
 *return* array;  
 }  
  
}

界面即功能实现代码：

*public class* BookTypeAddPanel *extends* JPanel *implements ActionListener* {  
 *private* ComponentTools componentTools = *new* ComponentTools();  
  
 *private* Box totalVBox, funcationBox, typeNameBox, typeDescriptionBox, buttonBox;  
 *private* JLabel typeNameLabel, bookTypeAddFuncationLabel, typeDescriptionLabel;  
 *private* JTextArea typeDescriptionTextArea;  
 *private* JTextField typeNameTextField;  
 *private* JButton addButton, resetButton;  
  
 BookTypeAddPanel() {  
 *// 为图书类型添加面板添加控件  
 this*.add(*this*.createBookTypeAddBox());  
 *// 设置按钮的图标* componentTools.setIcons(*new* JButton[]{addButton, resetButton},  
 *new* String[]{"src/bookManageSystem/images/add.png", "src/bookManageSystem/images/reset.png"});  
 *// 为按钮注册事件监听器* addButton.addActionListener(*this*);  
 resetButton.addActionListener(*this*);  
 }  
  
 */\*\*  
 \* 图书类型添加面板的控件内容  
 \*  
 \** ***@return*** *返回一个Box  
 \*/  
 private* Box createBookTypeAddBox() {  
 *// 创建一个垂直容器盒子* totalVBox = Box.*createVerticalBox*();  
 *// 创建一个水平容器盒子* funcationBox = Box.*createHorizontalBox*();  
 bookTypeAddFuncationLabel = *new* JLabel("图书类别添加功能");  
 bookTypeAddFuncationLabel.setFont(*new* Font("微软雅黑", Font.BOLD, 30));  
 funcationBox.add(bookTypeAddFuncationLabel);  
 totalVBox.add(funcationBox);  
 totalVBox.add(Box.*createVerticalStrut*(60));  
 typeNameBox = Box.*createHorizontalBox*();  
 typeNameLabel = *new* JLabel("图书类别名称：");  
 typeNameTextField = *new* JTextField(20);  
 typeNameBox.add(typeNameLabel);  
 typeNameBox.add(Box.*createHorizontalStrut*(50));  
 typeNameBox.add(typeNameTextField);  
 totalVBox.add(typeNameBox);  
 totalVBox.add(Box.*createVerticalStrut*(60));  
 typeDescriptionBox = Box.*createHorizontalBox*();  
 typeDescriptionLabel = *new* JLabel("图书类别描述：");  
 typeDescriptionTextArea = *new* JTextArea(10, 20);  
 typeDescriptionTextArea.setLineWrap(*true*);  
 typeDescriptionBox.add(typeDescriptionLabel);  
 typeDescriptionBox.add(Box.*createHorizontalStrut*(50));  
 typeDescriptionBox.add(typeDescriptionTextArea);  
 totalVBox.add(typeDescriptionBox);  
 totalVBox.add(Box.*createVerticalStrut*(60));  
 buttonBox = Box.*createHorizontalBox*();  
 addButton = *new* JButton("添加");  
 resetButton = *new* JButton("重置");  
 buttonBox.add(addButton);  
 buttonBox.add(Box.*createHorizontalStrut*(80));  
 buttonBox.add(resetButton);  
 totalVBox.add(buttonBox);  
  
 *return* totalVBox;  
 }  
  
 @Override  
 *public void* actionPerformed(ActionEvent e) {  
 *// 添加按钮的事件处理  
 if* (e.getSource() == addButton) {  
 *// 获取图书分类名称* String bookTypeName = typeNameTextField.getText();  
 *// 获取图书分类描述* String bookTypeDescription = typeDescriptionTextArea.getText();  
 *// 组装SQL语句* String sql = "insert into tb\_bookType (btName, btDescription) values ('" + bookTypeName + "','" + bookTypeDescription + "');";  
 *// 执行方法插入一条书籍  
 boolean* isOK = *new* BookTypeDao().dataChange(sql);  
 *// 判断是否操作成功并作出相应的处理  
 if* (isOK) {  
 JOptionPane.*showMessageDialog*(*null*, "添加成功！");  
 componentTools.reset(typeNameTextField, typeDescriptionTextArea);  
 } *else* {  
 JOptionPane.*showMessageDialog*(*null*, "添加失败！");  
 }  
 }  
  
 *// 重置按钮的事件处理  
 if* (e.getSource() == resetButton) {  
*// 即清空输入框的内容* componentTools.reset(typeNameTextField, typeDescriptionTextArea);  
 }  
  
  
 }  
}

**3.5 图书类别维护模块实现**

功能实现：

* + - 查询功能的实现是获取用户输入的图书类别名称，由于这里没有使用模糊查询，所以需要输入图书类别全称才能查询成功，通过用户输入的图书类别名称在数据库里查询数据然后将其显示在表格上。如图3-7
    - 当选中表格中的某一行后会将该行的数据显示在下面的各个文本框控件中，如图3-8
    - 可修改选中行显示在下面文本框中的内容，修改后，点击修改按钮，再次获取用户输入的内容然后使用更新SQL语句在数据库表中进行更新数据以完成修改功能，如图3-9
    - 删除按钮的功能就是删除该条记录，是通过获取编号文本框的编号来指定删除记录的，同时编号是不可编辑的，否则会删除失败。如图3-10



图 3-7 类别维护主界面



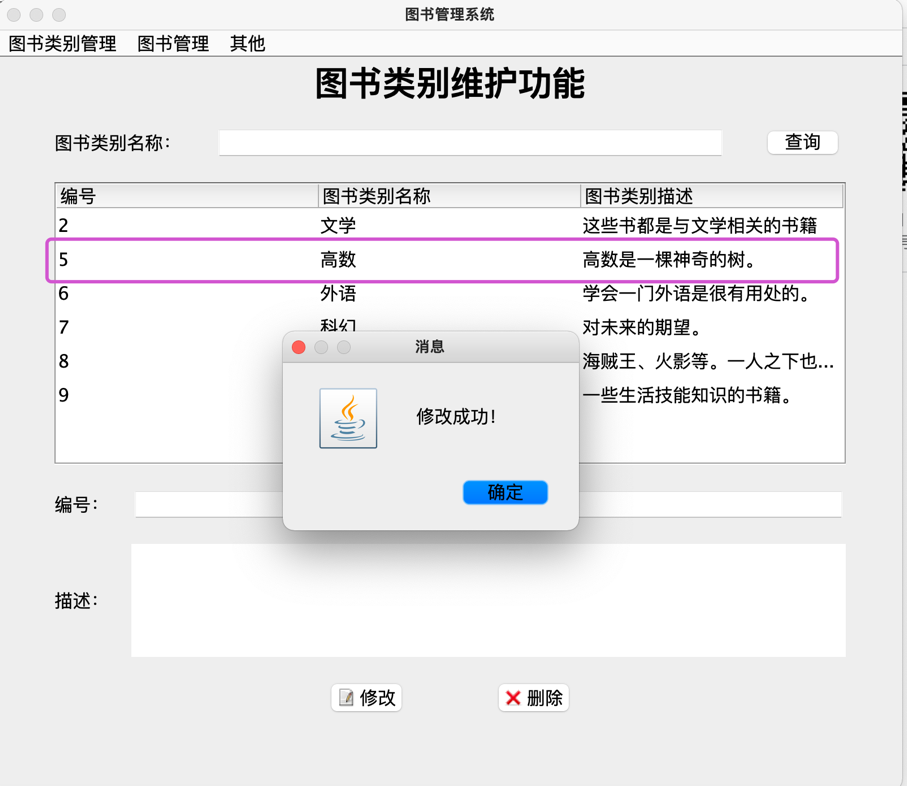
图 3-8 选中表格显示

图 3-9 修改图书类别信息



图 3-10 删除图书类别及信息

功能实现代码：

创建表格JTable控件：

1. /\*\*
2. \* 根据SQL查询得到的数据填充一个表格控件
3. \*
4. \* @param sql SQL语句
5. \* @return 返回一个JTable控件
6. \*/
7. private JTable createTable(String sql) {
8. // 将表格数据转换成一个二维数组
9. String[][] rowdatas = new BookTypeDao().ListToArray(new BookTypeDao().getRecordsDataBySql(sql));
10. // 表头内容
11. String[] headers = {"编号", "图书类别名称", "图书类别描述"};
12. // 实例化一个表格
13. table = new JTable();
14. // 设置行高
15. table.setRowHeight(30);
16. // 将表头数据和表格内容数据填充到默认表格模型中，使用表格模型方便获取表格数据内容和刷新表格
17. tableModel = new DefaultTableModel(rowdatas, headers);
18. // 设置表格的模型
19. table.setModel(tableModel);
20. // 返回表格控件
21. return table;
22. }

表格选中行显示数据

1. @Override
2. public void valueChanged(ListSelectionEvent e) {
3. // 获取表格被选中行
4. int getSelectedRowIndex = table.getSelectedRow();
5. // 判断是否未选中行
6. if (getSelectedRowIndex == -1) {
7. refreshTable();
8. } else {
9. // 将选中的表格行数据依次填充到下面的控件中
10. idTextField.setText((String) table.getValueAt(getSelectedRowIndex, 0));
11. bookTypeNameTextField2.setText((String) table.getValueAt(getSelectedRowIndex, 1));
12. descriptionTextArea.setText((String) table.getValueAt(getSelectedRowIndex, 2));
13. }
14. }

“查询”按钮的事件处理

1. // “查询”按钮的事件处理
2. if (e.getSource() == checkButton) {
3. String sql = "";
4. // 判断图书类别输入框是否为空并做处理
5. if (!new SimpleTools().isEmpty(bookTypeNameTextField1.getText())) {
6. // 为空即查询所有图书类别
7. sql = "select \* from tb\_booktype;";
8. } else {
9. // 不为空即按条件查询图书类别
10. sql = "select \* from tb\_booktype where btName='" + bookTypeNameTextField1.getText() + "';";
11. }
12. // 查询结果转换成二维数组
13. String[][] rowdatas = new BookTypeDao().ListToArray(new BookTypeDao().getRecordsDataBySql(sql));
14. // 表头数据
15. String[] headers = {"编号", "图书类别名称", "图书类别描述"};
16. // 填充表格模型，刷新表格
17. tableModel.setDataVector(rowdatas, headers);
18. }

“修改”按钮的事件处理：

1. // “修改”按钮的事件处理
2. if (e.getSource() == alterButton) {
3. // 获取各输入框的内容
4. String id = idTextField.getText();
5. String name = bookTypeNameTextField2.getText();
6. String description = descriptionTextArea.getText();
7. // 组装修改的SQL语句
8. String alterSQL = "update tb\_booktype set btName='" + name + "',btDescription='" + description + "' where" +
9. " btId=" + id + ";";
10. // 执行修改操作并返回结果
11. boolean isOK = new BookTypeDao().dataChange(alterSQL);
12. // 对修改结果进行判定
13. if (isOK) {
14. // 修改成功则刷新表格
15. refreshTable();
16. // 重置各输入框
17. new ComponentTools().reset(idTextField, bookTypeNameTextField2, descriptionTextArea);
18. // 弹出提示框
19. JOptionPane.showMessageDialog(null, "修改成功！");
20. } else {
21. // 修改失败也弹出提示框
22. JOptionPane.showMessageDialog(null, "修改失败！");
23. }
24. }

“修改”按钮的事件处理：

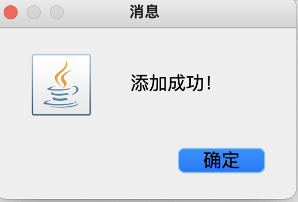
1. // “删除”按钮的事件处理
2. if (e.getSource() == deleteButton) {
3. // 获取要删除的id
4. String id = idTextField.getText();
5. // 组装删除SQL并且删除后对主键进行重置
6. String sql1 = "set FOREIGN\_KEY\_CHECKS=0;";
7. String deleteSQL = "delete from tb\_booktype where btId=" + id + ";";
8. String sql2 = "set FOREIGN\_KEY\_CHECKS=1;";
9. // 弹出确认框
10. int isOK = JOptionPane.showConfirmDialog(null, "是否确认删除？");
11. // 点击确定按钮则继续操作
12. if (isOK == JOptionPane.OK\_OPTION) {
13. // 执行SQL语句
14. new BookTypeDao().dataChange(sql1);
15. boolean is = new BookTypeDao().dataChange(deleteSQL);
16. new BookTypeDao().dataChange(sql2);
17. // 对删除结果进行判定
18. if (is) {
19. // 删除成功则删除表格并弹出提示框
20. refreshTable();
21. new ComponentTools().reset(idTextField, bookTypeNameTextField2, descriptionTextArea);
22. } else {
23. // 删除失败也弹出提示框
24. JOptionPane.showMessageDialog(null, "删除失败！");
25. }
26. } else {
27. return;
28. }
29. }

**3.6 图书添加模块实现**

添加功能是获取用户输入的数据然后插入到数据库表中。重置按钮即清空用户输入的内容。添加主页面如图3-11，实现添加图书功能如图3-12



图 3-11 图书添加页面

图 3-12

添加记录到数据库实现代码：

package BookManageSystem.dao;

import java.sql.Connection;

import java.sql.Statement;

public class BookDao {

/\*\*

\* 操作结果：根据SQL语句执行数据库的增删改操作

\*

\* @param sql SQL语句

\* @return boolean 如果操作数据库成功返回true，否则返回false

\*/

public boolean dataChange(String sql) {

Connection conn = null;

Statement stmt = null;

try {

//获得数据的连接

conn = JDBCUtils.getConnection();

//获得Statement对象

stmt = conn.createStatement();

//发送SQL语句

int num = stmt.executeUpdate(sql);

return num > 0;

} catch (Exception e) {

e.printStackTrace();

} finally {

JDBCUtils.release(stmt, conn);

}

return false;

}

}

界面设计及功能实现代码：

*public class* BookAddPanel *extends* JPanel *implements ActionListener* {  
 *private* ComponentTools componentTools = *new* ComponentTools();  
 *private* Box totalVBox, funcationHBox, nameAndAuthorHBox, sexAndPriceHBox, typeHBox, descriptionHBox, buttonHBox;  
 *private* JLabel bookAddFuncationLabel, bookNameLabel, bookAuthorLabel, authorSexLabel, bookTypeLabel, descriptionLabel, bookPriceLabel;  
 *private* JTextField bookNameTextField, bookAuthorTextField,bookPriceTextField;  
 *private* JTextArea descriptionTextArea;  
 *private* JRadioButton femaleRadioButton, maleRadioButton;  
 *private* ButtonGroup radioButtonGroup;  
 *private* JComboBox bookTypeComboBox;  
 *private* JButton addButton, resetButton;  
 BookAddPanel() {  
 *// 为图书添加面板添加控件内容  
 this*.add(createBookAddBox());  
 *// 批量为按钮设置图标* componentTools.setIcons(*new* JButton[]{addButton, resetButton},  
 *new* String[]{"src/bookManageSystem/images/add" +  
 ".png", "src/bookManageSystem/images/reset.png"});  
 *// 图书类别查询SQL* String getBookTypeSQL = "select \* from tb\_booktype";  
 *// 获取所有的图书类别数据  
 List* bookTypeList = *new* BookTypeDao().getRecordsDataBySql(getBookTypeSQL);  
 *// 获取所有的图书类别名称* String[] typeNames = *new* String[bookTypeList.size()];  
 *for* (*int* i = 0; i < bookTypeList.size();i++) {  
 BookTypeBean bookTypeBean = (BookTypeBean) bookTypeList.get(i);  
 typeNames[i] = bookTypeBean.getBookTypeName();  
 }  
 *// 初始化下拉列表框中的选项* componentTools.addComboBoxItems(bookTypeComboBox, typeNames);  
 *// 为按钮注册事件监听器* addButton.addActionListener(*this*);  
 resetButton.addActionListener(*this*);  
 }  
  
  
 */\*\*  
 \* 图书添加面板的内容控件  
 \*  
 \** ***@return*** *返回一个Box  
 \*/  
 private* Box createBookAddBox() {  
 totalVBox = Box.*createVerticalBox*();  
 funcationHBox = Box.*createHorizontalBox*();  
 bookAddFuncationLabel = *new* JLabel("图书添加功能");  
 bookAddFuncationLabel.setFont(*new* Font("微软雅黑", Font.BOLD, 30));  
 funcationHBox.add(bookAddFuncationLabel);  
 totalVBox.add(funcationHBox);  
 totalVBox.add(Box.*createVerticalStrut*(30));  
 nameAndAuthorHBox = Box.*createHorizontalBox*();  
 bookNameLabel = *new* JLabel("图书名称：");  
 bookNameTextField = *new* JTextField(10);  
 bookAuthorLabel = *new* JLabel("图书作者：");  
 bookAuthorTextField = *new* JTextField(10);  
 nameAndAuthorHBox.add(bookNameLabel);  
 nameAndAuthorHBox.add(Box.*createHorizontalStrut*(40));  
 nameAndAuthorHBox.add(bookNameTextField);  
 nameAndAuthorHBox.add(Box.*createHorizontalStrut*(40));  
 nameAndAuthorHBox.add(bookAuthorLabel);  
 nameAndAuthorHBox.add(Box.*createHorizontalStrut*(40));  
 nameAndAuthorHBox.add(bookAuthorTextField);  
 totalVBox.add(nameAndAuthorHBox);  
 totalVBox.add(Box.*createVerticalStrut*(30));  
 sexAndPriceHBox = Box.*createHorizontalBox*();  
 authorSexLabel = *new* JLabel("作者性别：");  
 Box sexHBox = Box.*createHorizontalBox*();  
 femaleRadioButton = *new* JRadioButton("女");  
 maleRadioButton = *new* JRadioButton("男");  
 sexHBox.add(femaleRadioButton);  
 sexHBox.add(maleRadioButton);  
 radioButtonGroup = *new* ButtonGroup();  
 radioButtonGroup.add(femaleRadioButton);  
 radioButtonGroup.add(maleRadioButton);  
 bookPriceLabel = *new* JLabel("图书价格：");  
 bookPriceTextField = *new* JTextField(5);  
 sexAndPriceHBox.add(authorSexLabel);  
 nameAndAuthorHBox.add(Box.*createHorizontalStrut*(50));  
 sexAndPriceHBox.add(sexHBox);  
 nameAndAuthorHBox.add(Box.*createHorizontalStrut*(100));  
 sexAndPriceHBox.add(bookPriceLabel);  
 nameAndAuthorHBox.add(Box.*createHorizontalStrut*(10));  
 sexAndPriceHBox.add(bookPriceTextField);  
 totalVBox.add(sexAndPriceHBox);  
 totalVBox.add(Box.*createVerticalStrut*(30));  
 typeHBox = Box.*createHorizontalBox*();  
 bookTypeLabel = *new* JLabel("图书类别：");  
 *// 实例化下拉列表框控件* bookTypeComboBox = *new* JComboBox();  
 typeHBox.add(bookTypeLabel);  
 typeHBox.add(Box.*createHorizontalStrut*(40));  
 typeHBox.add(bookTypeComboBox);  
 totalVBox.add(typeHBox);  
 totalVBox.add(Box.*createVerticalStrut*(30));  
 descriptionHBox = Box.*createHorizontalBox*();  
 descriptionLabel = *new* JLabel("图书描述：");  
 descriptionTextArea = *new* JTextArea(10, 40);  
 descriptionHBox.add(descriptionLabel);  
 descriptionHBox.add(Box.*createHorizontalStrut*(40));  
 descriptionHBox.add(descriptionTextArea);  
 totalVBox.add(descriptionHBox);  
 totalVBox.add(Box.*createVerticalStrut*(30));  
 buttonHBox = Box.*createHorizontalBox*();  
 addButton = *new* JButton("添加");  
 resetButton = *new* JButton("重置");  
 buttonHBox.add(addButton);  
 buttonHBox.add(Box.*createHorizontalStrut*(80));  
 buttonHBox.add(resetButton);  
 totalVBox.add(buttonHBox);  
 *return* totalVBox;  
 }  
  
 @Override  
 *public void* actionPerformed(ActionEvent e) {  
 *// “添加”按钮的事件处理  
 if* (e.getSource() == addButton) {  
 *// 获取用户输入的图书名称* String name = bookNameTextField.getText();  
 *// 获取用户输入的图书作者名字* String author = bookAuthorTextField.getText();  
 *// 获取用户选择的图书作者的性别* String sex = "";  
 *if* (maleRadioButton.isSelected()) {  
 sex = maleRadioButton.getText();  
 } *else if* (femaleRadioButton.isSelected()) {  
 sex = femaleRadioButton.getText();  
 }  
 *// 获取用户输入的图书价格* String price = bookPriceTextField.getText();  
 *// 获取用户选择的图书类别* String type = (String) bookTypeComboBox.getModel().getSelectedItem();  
 *// 获取用户输入的图书描述* String description = descriptionTextArea.getText();  
 *// 组装图书类别SQL* String bookTypeSQL = "select \* from tb\_booktype where btName='" + type + "';";  
 *// 根据用户选择的图书类别进行查询  
 List* bookTypeList = *new* BookTypeDao().getRecordsDataBySql(bookTypeSQL);  
 *// 获取查询结果* BookTypeBean bookTypeBean = (BookTypeBean) bookTypeList.get(0);  
 *// 获取用户选择的图书类别的id号  
 int* bookTypeId = bookTypeBean.getBookTypeId();  
 *// 组装新增SQL语句* String sql = "insert into tb\_book(bBookName, bAuthor, bSex, bPrice, bBookDescription, btId) " +  
 "values('" + name + "','" + author + "','" + sex + "'," + price + ",'" + description + "'," + bookTypeId + ");" ;  
 *// 执行SQL并获取操作结果  
 boolean* isOK = *new* BookDao().dataChange(sql);  
 *// 对添加结果进行判断  
 if* (isOK) {  
 *// 添加成功则重置用户输入，并弹出提示框* componentTools.reset(bookNameTextField, bookAuthorTextField, bookPriceTextField, descriptionTextArea);  
 componentTools.reset(radioButtonGroup);  
 componentTools.reset(bookTypeComboBox);  
 JOptionPane.*showMessageDialog*(*null*, "添加成功！");  
 } *else* {  
 *// 添加失败也弹出提示框* JOptionPane.*showMessageDialog*(*null*, "添加失败！");  
  
 }  
 }  
 *// “重置”按钮的事件处理  
 if*(e.getSource() == resetButton) {  
 *// 重置用户输入和选择* componentTools.reset(bookNameTextField, bookAuthorTextField, bookPriceTextField, descriptionTextArea);  
 componentTools.reset(radioButtonGroup);  
 componentTools.reset(bookTypeComboBox);  
 }  
 }  
  
}

**3.7 图书维护模块实现**

实现思路：

* 查询按钮：获取用户输入的图书名称、图书作者、图书类别然后在数据库表中进行多条件查询，将查询结果转换成可以显示在表格中的数据，最后显示在表格中。如图 3-13
* 重置按钮：清空所有用户的输入。
* 用户选中表格行并将数据显示下面的各个控件中,如图3-14
* 修改按钮：获取用户输入的各项内容，然后使用修改SQL在数据库表中修改内容，再将修改成功的内容完整显示在表格中，如图3-15
* 删除按钮：获取编号文本框的内容，然后根据编号在数据库表中删除该条记录，如图3-16

图 3-13 图书维护界面

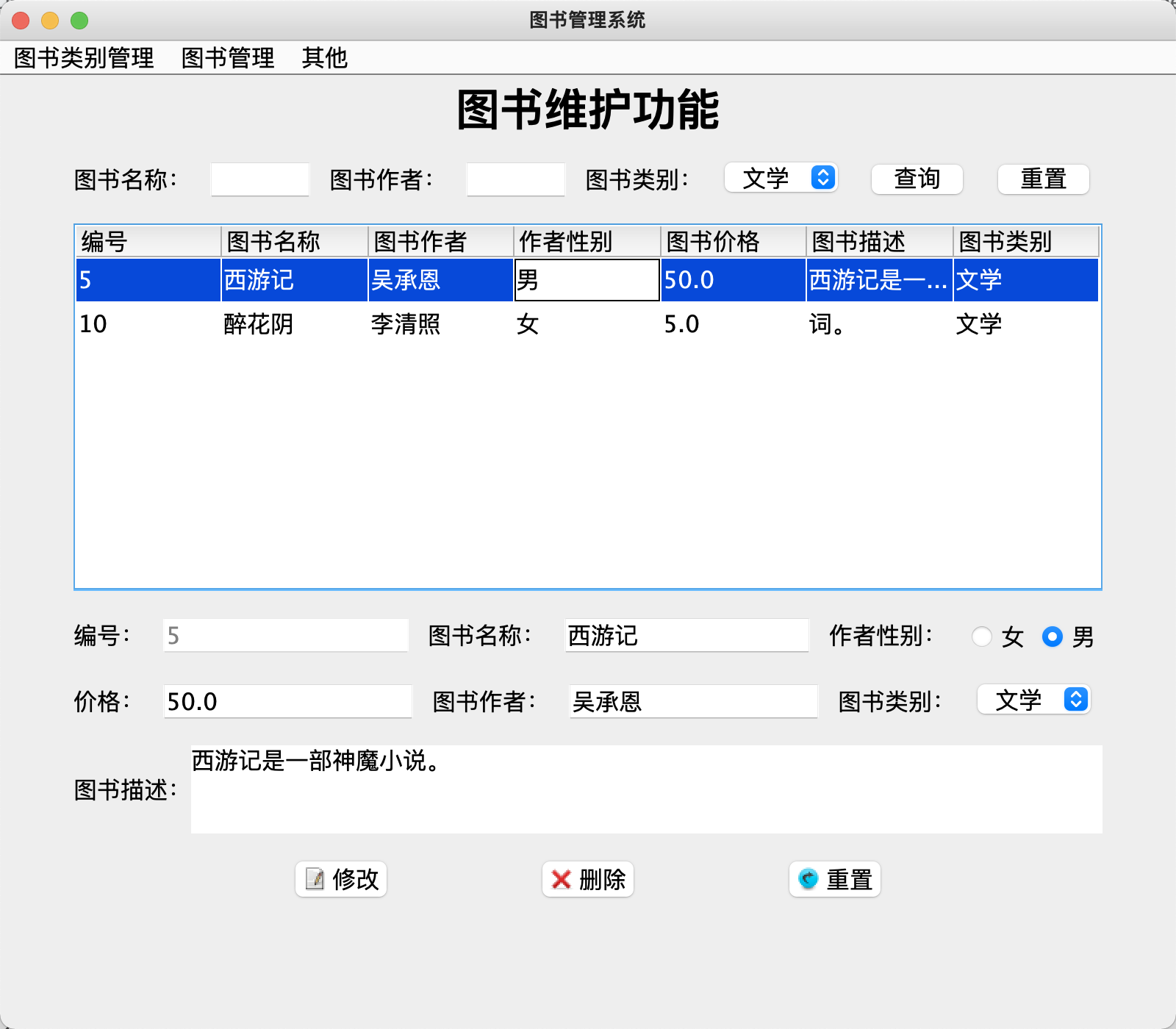


图 3-14 选中表格行显示图书信息

图3-15 图书信息修改



图3-16 删除图书信息

“查询”按钮事件处理实现代码：

1. // “查询”按钮的事件处理
2. if (e.getSource() == checkButton) {
3. // 查询SQL
4. String sql =
5. "select bId,bBookName,bAuthor,bSex,bPrice,bBookDescription,btName from tb\_book,tb\_booktype where "
6. + "tb\_book.btId=tb\_booktype.btId ";
7. // 判断输入是否为空
8. if (simpleTools.isEmpty(bookNameTextField.getText()) && simpleTools.isEmpty(bookAuthorTextField.getText()) && bookTypeComboBox.getSelectedIndex() < 0) {
9. sql += " and 1=1;";
10. } else {
11. // 用户输入的图书名称
12. if (bookNameTextField.getText().length() > 0) {
13. sql += " and bBookName like '%" + bookNameTextField.getText() + "%'";
14. }
15. // 用户输入的图书作者
16. if (bookAuthorTextField.getText().length() > 0) {
17. sql += " and bAuthor like '%" + bookAuthorTextField.getText() + "%'";
18. }
19. // 用户选择的图书类别
20. String booktype = (String) bookTypeComboBox.getModel().getSelectedItem();
21. if (simpleTools.isEmpty(booktype)) {
22. sql += " and btName='" + booktype + "';";
23. }
24. }
25. // 根据组装的SQL获取表格数据并转换成二维数组
26. String[][] rowdatas = new BookDao().ListToArray(new BookDao().getRecordsDataBySql(sql));
27. // 表头
28. String[] headers = {"编号", "图书名称", "图书作者", "作者性别", "图书价格", "图书描述", "图书类别"};
29. // 重新填充表格数据模型，刷新表格
30. tableModel.setDataVector(rowdatas, headers);
31. }

表格选中行显示数据实现代码：

1. @Override
2. public void valueChanged(ListSelectionEvent e) {
3. // 获取表格所选中行的索引
4. int getSelectedRowIndex = table.getSelectedRow();
5. // 判断是用户否选中表格中的某一行
6. if (getSelectedRowIndex == -1) {
7. refreshTable();
8. } else {
9. // 如果选中表格中的某一行，则将选中行的数据填充到下面的控件中
10. idTextField.setText((String) table.getValueAt(getSelectedRowIndex, 0));
11. bookNameTextField2.setText((String) table.getValueAt(getSelectedRowIndex, 1));
12. boolean b = table.getValueAt(getSelectedRowIndex, 3).equals("男");
13. if (b) {
14. maleRadioButton.setSelected(true);
15. } else {
16. femaleRadioButton.setSelected(true);
17. }
18. priceTextField.setText((String) table.getValueAt(getSelectedRowIndex, 4));
19. bookAuthorTextField2.setText((String) table.getValueAt(getSelectedRowIndex, 2));
20. bookTypeComboBox2.setSelectedItem(table.getValueAt(getSelectedRowIndex, 6));
21. bookDescriptionTextArea.setText((String) table.getValueAt(getSelectedRowIndex, 5));
22. }
23. }

“修改”按钮事件处理实现代码：

1. // “修改”按钮的事件处理
2. if (e.getSource() == alterButton) {
3. String id = idTextField.getText();
4. String bookName = bookNameTextField2.getText();
5. String sex = "";
6. if (femaleRadioButton.isSelected()) {
7. sex = femaleRadioButton.getText();
8. } else {
9. sex = maleRadioButton.getText();
10. }
11. Float price = Float.parseFloat(priceTextField.getText());
12. String bookAuthor = bookAuthorTextField2.getText();
13. String bookType = (String) bookTypeComboBox2.getModel().getSelectedItem();
14. String bookDescription = bookDescriptionTextArea.getText();
15. // 组装修改SQL
16. String alterSQL =
17. "update tb\_booktype,tb\_book set bBookName='" + bookName + "',bAuthor='" + bookAuthor + "',bSex='" + sex + "'," + "bPrice" + "=" + price + "," + "tb\_booktype.btName='" + bookType + "',bBookDescription='" + bookDescription + "' where " + "tb\_book.btId=tb\_booktype" + ".btId " + "and" + " bId=" + id + ";";
18. // 执行修改操作并返回操作结果
19. boolean isOK = new BookTypeDao().dataChange(alterSQL);
20. // 对修改结果进行处理
21. if (isOK) {
22. // 修改成功则刷新表格、重置控件内容、弹出提示框
23. refreshTable();
24. componentTools.reset(idTextField, bookNameTextField2, priceTextField, bookAuthorTextField2,
25. bookDescriptionTextArea);
26. componentTools.reset(sexButtonGroup);
27. componentTools.reset(bookTypeComboBox2);
28. JOptionPane.showMessageDialog(null, "修改成功！");
29. } else {
30. // 修改失败则也弹出提示框
31. JOptionPane.showMessageDialog(null, "修改失败！");
32. }
33. }

“删除”按钮事件处理代码：

1. // “删除”按钮的事件处理
2. if (e.getSource() == deleteButton) {
3. // 获取要删除的图书id
4. String id = idTextField.getText();
5. // 删除SQL并重置tb\_book的主键
6. String sql1 = "set FOREIGN\_KEY\_CHECKS=0;";
7. String deleteSQL = "delete from tb\_book where bId=" + id + ";";
8. String sql2 = "set FOREIGN\_KEY\_CHECKS=1;";
9. // 弹出确认框来确认用户是否要删除
10. int isOK = JOptionPane.showConfirmDialog(null, "是否确认删除？");
11. // 如果用户点击了确定按钮
12. if (isOK == JOptionPane.OK\_OPTION) {
13. // 执行删除SQL
14. new BookDao().dataChange(sql1);
15. boolean is = new BookDao().dataChange(deleteSQL);
16. new BookDao().dataChange(sql2);
17. // 对删除操作结果进行判断
18. if (is) {
19. // 删除成功则刷新表格、重置控件内容
20. refreshTable();
21. componentTools.reset(idTextField, bookNameTextField2, priceTextField, bookAuthorTextField2,
22. bookDescriptionTextArea);
23. componentTools.reset(sexButtonGroup);
24. componentTools.reset(bookTypeComboBox2);
25. } else {
26. // 删除失败也弹出提示框
27. JOptionPane.showMessageDialog(null, "删除失败！");
28. }
29. } else {
30. return;
31. }
32. }

第二个“重置”按钮事件处理：

1. // 第二个“重置”按钮的事件处理
2. if (e.getSource() == resetButton2) {
3. // 重置控件内容
4. componentTools.reset(idTextField, bookNameTextField2, priceTextField, bookAuthorTextField2,
5. bookDescriptionTextArea);
6. componentTools.reset(sexButtonGroup);
7. componentTools.reset(bookTypeComboBox2);
8. }

界面及功能实现代码：

*public class* BookManagePanel *extends* JPanel *implements ActionListener*,*ListSelectionListener*{  
 *private* ComponentTools componentTools = *new* ComponentTools();  
 *private* SimpleTools simpleTools = *new* SimpleTools();  
 *private* Box totalVBox, funcationHBox, checkHBox, tableHBox, sexRadioButtonHBox, descriptionHBox, buttonHBox, showContentHBox1, showContentHBox2;  
 *private* JComboBox bookTypeComboBox, bookTypeComboBox2;  
 *private* JLabel bookManageFuncationLabel, bookNameLabel, bookTypeLabel, idLabel, priceLabel, bookNameLabel2, bookAuthorLabel, bookTypeLabel2, bookAuthorLabel2, authorSexLabel, bookDescriptionLabel;  
 *private* JTextField bookNameTextField, bookNameTextField2, bookAuthorTextField, bookAuthorTextField2, idTextField, priceTextField;  
 *private* JTextArea bookDescriptionTextArea;  
 *private* JButton checkButton, resetButton, resetButton2, alterButton, deleteButton;  
 *private* ButtonGroup sexButtonGroup;  
 *private* JScrollPane tableScrollPanel;  
 *private* JRadioButton femaleRadioButton, maleRadioButton;  
 *private* JTable table;  
 *private* DefaultTableModel tableModel;  
  
 BookManagePanel() {  
 *// 为图书维护面板添加控件内容  
 this*.add(*this*.createBookManageBox());  
 *// 图书类别查询SQL* String getBookTypeSQL = "select \* from tb\_booktype;";  
 *// 获取所有的图书类别数据  
 List* bookTypeList = *new* BookTypeDao().getRecordsDataBySql(getBookTypeSQL);  
 *// 提取所有的图书类别名称信息* String[] typeNames = *new* String[bookTypeList.size()];  
 *for* (*int* i = 0; i < bookTypeList.size(); i++) {  
 BookTypeBean bookTypeBean = (BookTypeBean) bookTypeList.get(i);  
 typeNames[i] = bookTypeBean.getBookTypeName();  
 }  
 *// 初始化下拉列表框中的图书类别信息* componentTools.addComboBoxItems(bookTypeComboBox, typeNames);  
 componentTools.addComboBoxItems(bookTypeComboBox2, typeNames);  
  
  
 *// 为按钮注册事件监听器* checkButton.addActionListener(*this*);  
 resetButton.addActionListener(*this*);  
 alterButton.addActionListener(*this*);  
 deleteButton.addActionListener(*this*);  
 resetButton2.addActionListener(*this*);  
  
 *// 为表格注册事件监听器* table.getSelectionModel().addListSelectionListener(*this*);  
 }  
  
  
 */\*\*  
 \* 创建图书维护面板的控件内容  
 \*  
 \** ***@return*** *返回一个Box  
 \*/  
 private* Box createBookManageBox() {  
 totalVBox = Box.*createVerticalBox*();  
 funcationHBox = Box.*createHorizontalBox*();  
 bookManageFuncationLabel = *new* JLabel("图书维护功能");  
 bookManageFuncationLabel.setFont(*new* Font("微软雅黑", Font.BOLD, 30));  
 funcationHBox.add(bookManageFuncationLabel);  
 totalVBox.add(funcationHBox);  
 totalVBox.add(Box.*createVerticalStrut*(15));  
 checkHBox = Box.*createHorizontalBox*();  
 bookNameLabel = *new* JLabel("图书名称：");  
 bookNameTextField = *new* JTextField();  
 bookAuthorLabel = *new* JLabel("图书作者：");  
 bookAuthorTextField = *new* JTextField();  
 bookTypeLabel = *new* JLabel("图书类别：");  
 bookTypeComboBox = *new* JComboBox();  
 checkButton = *new* JButton("查询");  
 resetButton = *new* JButton("重置");  
 checkHBox.add(bookNameLabel);  
 checkHBox.add(Box.*createHorizontalStrut*(10));  
 checkHBox.add(bookNameTextField);  
 checkHBox.add(Box.*createHorizontalStrut*(10));  
 checkHBox.add(bookAuthorLabel);  
 checkHBox.add(Box.*createHorizontalStrut*(10));  
 checkHBox.add(bookAuthorTextField);  
 checkHBox.add(Box.*createHorizontalStrut*(10));  
 checkHBox.add(bookTypeLabel);  
 checkHBox.add(Box.*createHorizontalStrut*(10));  
 checkHBox.add(bookTypeComboBox);  
 checkHBox.add(Box.*createHorizontalStrut*(10));  
 checkHBox.add(checkButton);  
 checkHBox.add(Box.*createHorizontalStrut*(10));  
 checkHBox.add(resetButton);  
 totalVBox.add(checkHBox);  
 totalVBox.add(Box.*createVerticalStrut*(15));  
 tableHBox = Box.*createHorizontalBox*();  
 *// 实例化一个滚动面板* tableScrollPanel = *new* JScrollPane();  
 *// 将表格添加到滚动面板中* tableScrollPanel.setViewportView(*this*.createTable("select bId,bBookName," +  
 "bAuthor,bSex,bPrice,bBookDescription,btName from tb\_book,tb\_booktype where tb\_book.btId=tb\_booktype.btId;" ));  
 *// 设置预定义大小* tableScrollPanel.setPreferredSize(*new* Dimension(700, 250));  
 tableHBox.add(tableScrollPanel);  
 totalVBox.add(tableHBox);  
 totalVBox.add(Box.*createVerticalStrut*(15));  
 showContentHBox1 = Box.*createHorizontalBox*();  
 idLabel = *new* JLabel("编号：");  
 idTextField = *new* JTextField(5);  
 idTextField.setEnabled(*false*);  
 bookNameLabel2 = *new* JLabel("图书名称：");  
 bookNameTextField2 = *new* JTextField(5);  
 authorSexLabel = *new* JLabel("作者性别：");  
 sexRadioButtonHBox = Box.*createHorizontalBox*();  
 femaleRadioButton = *new* JRadioButton("女");  
 maleRadioButton = *new* JRadioButton("男");  
 sexButtonGroup = *new* ButtonGroup();  
 sexButtonGroup.add(femaleRadioButton);  
 sexButtonGroup.add(maleRadioButton);  
 sexRadioButtonHBox.add(femaleRadioButton);  
 sexRadioButtonHBox.add(maleRadioButton);  
 showContentHBox1.add(idLabel);  
 showContentHBox1.add(Box.*createHorizontalStrut*(10));  
 showContentHBox1.add(idTextField);  
 showContentHBox1.add(Box.*createHorizontalStrut*(10));  
 showContentHBox1.add(bookNameLabel2);  
 showContentHBox1.add(Box.*createHorizontalStrut*(10));  
 showContentHBox1.add(bookNameTextField2);  
 showContentHBox1.add(Box.*createHorizontalStrut*(10));  
 showContentHBox1.add(authorSexLabel);  
 showContentHBox1.add(Box.*createHorizontalStrut*(10));  
 showContentHBox1.add(sexRadioButtonHBox);  
 totalVBox.add(showContentHBox1);  
 totalVBox.add(Box.*createVerticalStrut*(15));  
 showContentHBox2 = Box.*createHorizontalBox*();  
 priceLabel = *new* JLabel("价格：");  
 priceTextField = *new* JTextField(5);  
 bookAuthorLabel2 = *new* JLabel("图书作者：");  
 bookAuthorTextField2 = *new* JTextField(5);  
 bookTypeLabel2 = *new* JLabel("图书类别：");  
 bookTypeComboBox2 = *new* JComboBox();  
 showContentHBox2.add(priceLabel);  
 showContentHBox2.add(Box.*createHorizontalStrut*(10));  
 showContentHBox2.add(priceTextField);  
 showContentHBox2.add(Box.*createHorizontalStrut*(10));  
 showContentHBox2.add(bookAuthorLabel2);  
 showContentHBox2.add(Box.*createHorizontalStrut*(10));  
 showContentHBox2.add(bookAuthorTextField2);  
 showContentHBox2.add(Box.*createHorizontalStrut*(10));  
 showContentHBox2.add(bookTypeLabel2);  
 showContentHBox2.add(Box.*createHorizontalStrut*(10));  
 showContentHBox2.add(bookTypeComboBox2);  
 totalVBox.add(showContentHBox2);  
 totalVBox.add(Box.*createVerticalStrut*(15));  
 descriptionHBox = Box.*createHorizontalBox*();  
 bookDescriptionLabel = *new* JLabel("图书描述：");  
 bookDescriptionTextArea = *new* JTextArea(3, 10);  
 descriptionHBox.add(bookDescriptionLabel);  
 descriptionHBox.add(bookDescriptionTextArea);  
 totalVBox.add(descriptionHBox);  
 totalVBox.add(Box.*createVerticalStrut*(15));  
 buttonHBox = Box.*createHorizontalBox*();  
 alterButton = *new* JButton("修改");  
 deleteButton = *new* JButton("删除");  
 resetButton2 = *new* JButton("重置");  
 buttonHBox.add(alterButton);  
 buttonHBox.add(Box.*createHorizontalStrut*(100));  
 buttonHBox.add(deleteButton);  
 buttonHBox.add(Box.*createHorizontalStrut*(100));  
 buttonHBox.add(resetButton2);  
 totalVBox.add(buttonHBox);  
 *// 批量为各个按钮设置图标* componentTools.setIcons(*new* JButton[]{alterButton, deleteButton, resetButton2},  
 *new* String[]{"src/bookManageSystem/images/edit.png" , "src/bookManageSystem/images/delete.png" ,  
 "src/bookManageSystem/images/reset.png" });  
 *return* totalVBox;  
 }  
  
  
 */\*\*  
 \* 根据SQL查询得到的表格数据填充创建表格  
 \*  
 \** ***@param sql*** *SQL语句  
 \** ***@return*** *返回填充完成的JTable控件  
 \*/  
 private* JTable createTable(String sql) {  
 *// 获取所有的图书记录并将结果转换成一个二维数组* String[][] rowdatas = *new* BookDao().ListToArray(*new* BookDao().getRecordsDataBySql(sql));  
 *// 表头* String[] headers = {"编号", "图书名称", "图书作者", "作者性别", "图书价格", "图书描述", "图书类别"};  
 *// 实例化表格控件* table = *new* JTable();  
 *// 设置行高* table.setRowHeight(30);  
 *// 将表头和表格内容数据填充到表格数据模型中* tableModel = *new* DefaultTableModel(rowdatas, headers);  
 *//为表格设置数据模型* table.setModel(tableModel);  
 *// 返回填充完成的表格控件  
 return* table;  
 }  
  
 @Override  
 *public void* actionPerformed(ActionEvent e) {  
 *// “查询”按钮的事件处理  
 if* (e.getSource() == checkButton) {  
 *// 查询SQL* String sql = "select bId,bBookName,bAuthor,bSex,bPrice,bBookDescription,btName from tb\_book,tb\_booktype where tb\_book.btId=tb\_booktype.btId " ;  
 *// 判断输入是否为空  
 if* (simpleTools.isEmpty(bookNameTextField.getText())  
 && simpleTools.isEmpty(bookAuthorTextField.getText())  
 && bookTypeComboBox.getSelectedIndex() < 0) {  
 sql += " and 1=1;";  
 } *else* {  
 *// 用户输入的图书名称  
 if* (bookNameTextField.getText().length() > 0) {  
 sql += " and bBookName like '%" + bookNameTextField.getText() + "%'"; }  
 *// 用户输入的图书作者  
 if* (bookAuthorTextField.getText().length() > 0) {  
 sql += " and bAuthor like '%" + bookAuthorTextField.getText() + "%'";  
 }  
 *// 用户选择的图书类别* String booktype = (String) bookTypeComboBox.getModel().getSelectedItem();  
 *if* (simpleTools.isEmpty(booktype)) {  
 sql += " and btName='" + booktype + "';";  
 }  
 }  
  
 *// 根据组装的SQL获取表格数据并转换成二维数组* String[][] rowdatas = *new* BookDao().ListToArray(*new* BookDao().getRecordsDataBySql(sql));  
 *// 表头* String[] headers = {"编号", "图书名称", "图书作者", "作者性别", "图书价格", "图书描述", "图书类别"};  
 *// 重新填充表格数据模型，刷新表格* tableModel.setDataVector(rowdatas, headers);  
 }  
  
 *// “重置”按钮的事件处理  
 if* (e.getSource() == resetButton) {  
 *// 查询SQL* String sql = "select bId,bBookName,bAuthor,bSex,bPrice,bBookDescription,btName from tb\_book,tb\_booktype where tb\_book.btId=tb\_booktype.btId " ;  
 String[][] rowdatas = *new* BookDao().ListToArray(*new* BookDao().getRecordsDataBySql(sql));  
 String[] headers = {"编号", "图书名称", "图书作者", "作者性别", "图书价格", "图书描述", "图书类别"};  
 tableModel.setDataVector(rowdatas, headers);  
 componentTools.reset(bookNameTextField, bookAuthorTextField);  
 componentTools.reset(bookTypeComboBox);  
 }  
  
 *// “修改”按钮的事件处理  
 if* (e.getSource() == alterButton) {  
 String id = idTextField.getText();  
 String bookName = bookNameTextField2.getText();  
 String sex = "";  
 *if* (femaleRadioButton.isSelected()) {  
 sex = femaleRadioButton.getText();  
 } *else* {  
 sex = maleRadioButton.getText();  
 }  
 Float price = Float.*parseFloat*(priceTextField.getText());  
 String bookAuthor = bookAuthorTextField2.getText();  
 String bookType = (String) bookTypeComboBox2.getModel().getSelectedItem();  
 String bookDescription = bookDescriptionTextArea.getText();  
 *// 组装修改SQL* String alterSQL = "update tb\_booktype,tb\_book set bBookName='" + bookName + "',bAuthor='" + bookAuthor +  
  
 "',bSex='" + sex + "',bPrice=" + price + ",tb\_booktype.btName='" + bookType + "'," +  
 "bBookDescription='" + bookDescription + "' where tb\_book.btId=tb\_booktype.btId and bId=" + id + ";" ;  
 *// 执行修改操作并返回操作结果  
 boolean* isOK = *new* BookTypeDao().dataChange(alterSQL);  
 *// 对修改结果进行处理  
 if* (isOK) {  
 *// 修改成功则刷新表格、重置控件内容、弹出提示框* refreshTable();  
 componentTools.reset(idTextField, bookNameTextField2, priceTextField, bookAuthorTextField2, bookDescriptionTextArea);  
 componentTools.reset(sexButtonGroup);  
 componentTools.reset(bookTypeComboBox2);  
 JOptionPane.*showMessageDialog*(*null*, "修改成功！");  
 } *else* {  
 *// 修改失败则也弹出提示框* JOptionPane.*showMessageDialog*(*null*, "修改失败！");  
 }  
 }  
  
 *// “删除”按钮的事件处理  
 if* (e.getSource() == deleteButton) {  
 *// 获取要删除的图书id* String id = idTextField.getText();  
 *// 删除SQL并重置tb\_book的主键* String sql1 = "set FOREIGN\_KEY\_CHECKS=0;";  
 String deleteSQL = "delete from tb\_book where bId=" + id + ";";  
 String sql2 = "set FOREIGN\_KEY\_CHECKS=1;";  
 *// 弹出确认框来确认用户是否要删除  
 int* isOK = JOptionPane.*showConfirmDialog*(*null*, "是否确认删除？");  
 *// 如果用户点击了确定按钮  
 if* (isOK == JOptionPane.OK\_OPTION) {  
 *// 执行删除SQL  
 new* BookDao().dataChange(sql1);  
 *boolean* is = *new* BookDao().dataChange(deleteSQL);  
 *new* BookDao().dataChange(sql2);  
 *// 对删除操作结果进行判断  
 if* (is) {  
 *// 删除成功则刷新表格、重置控件内容* refreshTable();  
 componentTools.reset(idTextField, bookNameTextField2, priceTextField, bookAuthorTextField2, bookDescriptionTextArea);  
 componentTools.reset(sexButtonGroup);  
 componentTools.reset(bookTypeComboBox2);  
 } *else* {  
 *// 删除失败也弹出提示框* JOptionPane.*showMessageDialog*(*null*, "删除失败！");  
 }  
 } *else* {  
 *return*;  
 }  
 }  
  
 *// 第二个“重置”按钮的事件处理  
 if* (e.getSource() == resetButton2) {  
 *// 重置控件内容* componentTools.reset(idTextField, bookNameTextField2, priceTextField, bookAuthorTextField2, bookDescriptionTextArea);  
 componentTools.reset(sexButtonGroup);  
 componentTools.reset(bookTypeComboBox2);  
 }  
  
 }  
  
 @Override  
 *public void* valueChanged(ListSelectionEvent e) {  
 *// 获取表格所选中行的索引  
 int* getSelectedRowIndex = table.getSelectedRow();  
 *// 判断是用户否选中表格中的某一行  
 if* (getSelectedRowIndex == -1) {  
 refreshTable();  
 } *else* {  
 *// 如果选中表格中的某一行，则将选中行的数据填充到下面的控件中* idTextField.setText((String) table.getValueAt(getSelectedRowIndex, 0));  
 bookNameTextField2.setText((String) table.getValueAt(getSelectedRowIndex, 1));  
 *boolean* b = table.getValueAt(getSelectedRowIndex, 3).equals("男");  
 *if* (b) {  
 maleRadioButton.setSelected(*true*);  
  
 } *else* {  
 femaleRadioButton.setSelected(*true*);  
 }  
 priceTextField.setText((String) table.getValueAt(getSelectedRowIndex, 4));  
 bookAuthorTextField2.setText((String) table.getValueAt(getSelectedRowIndex, 2));  
 bookTypeComboBox2.setSelectedItem(table.getValueAt(getSelectedRowIndex, 6));  
 bookDescriptionTextArea.setText((String) table.getValueAt(getSelectedRowIndex, 5));  
 }  
 }  
  
 *private void* refreshTable() {  
 *// 条件查询SQL* String sql = "select bId,bBookName,bAuthor,bSex,bPrice,bBookDescription," +  
 "btName from tb\_book,tb\_booktype where tb\_book.btId=tb\_booktype.btId;" ;  
 String[][] rowdatas = *new* BookDao().ListToArray(*new* BookDao().getRecordsDataBySql(sql));  
 String[] headers = {"编号", "图书名称", "图书作者", "作者性别", "图书价格", "图书描述", "图书类别"};  
 *// 重新填充表格数据，刷新表格* tableModel.setDataVector(rowdatas, headers);  
 }  
  
}

**3.8 其他界面及功能实现**

退出功能：退出整个系统，如图3-17

关于软件：点击”关于软件“菜单项，然后弹出该dialog，并且上面的“开发者”可以作为链接点击打开，如图3-18



图 3-17 退出功能



图 3-18 关于软件

界面及功能实现代码：

*public class* AboutSoftDialog *extends* JDialog *implements ActionListener*,*MouseListener*{  
  
 *private* ComponentTools componentTools = *new* ComponentTools();  
 *private* JPanel aboutSoftPanel;  
 *private* Box totalHBox, leftHBox, rightVBox;  
 *private* JLabel iconLabel, systemLabel, editionLabel, hyperlinkLabel;  
 *private* JButton closeButton;  
  
 AboutSoftDialog() {  
 *// 设置Dialog的相关属性  
 this*.setTitle("关于软件");  
 *this*.setBounds(400, 400, 500, 300);  
 *this*.setContentPane(*this*.createAboutSoftPanel());  
 *this*.setVisible(*false*);  
 *// 为按钮注册事件监听器* closeButton.addActionListener(*this*);  
 *// 为标签注册鼠标事件监听器* hyperlinkLabel.addMouseListener(*this*);  
 }  
  
  
 */\*\*  
 \* 创建Dialog的内容面板  
 \*  
 \** ***@return*** *返回一个JPanel  
 \*/  
 private* JPanel createAboutSoftPanel() {  
 aboutSoftPanel = *new* JPanel();  
 aboutSoftPanel.setLayout(*new* BorderLayout());  
 totalHBox = Box.*createHorizontalBox*();  
 leftHBox = Box.*createHorizontalBox*();  
 iconLabel = *new* JLabel();  
 iconLabel.setIcon(componentTools.iconSize(*new* ImageIcon("src/bookManageSystem/images/panda.png"),  
 160, 160));  
 leftHBox.add(iconLabel);  
 totalHBox.add(leftHBox);  
 rightVBox = Box.*createVerticalBox*();  
 systemLabel = *new* JLabel("图书管理系统");  
 systemLabel.setFont(*new* Font("微软雅黑", Font.PLAIN, 30));  
 editionLabel = *new* JLabel("版本 2.0");  
 editionLabel.setFont(*new* Font("微软雅黑", Font.PLAIN, 30));  
 hyperlinkLabel = *new* JLabel("<html><u>开发者:\n\t\t杜小白\n\t\t孙文hao</u></html>");  
 hyperlinkLabel.setForeground(*new* Color(0, 149, 200));  
 hyperlinkLabel.setFont(*new* Font("微软雅黑", Font.PLAIN, 20));  
 rightVBox.add(systemLabel);  
 rightVBox.add(Box.*createVerticalStrut*(50));  
 rightVBox.add(editionLabel);  
 rightVBox.add(Box.*createVerticalStrut*(50));  
 rightVBox.add(hyperlinkLabel);  
 totalHBox.add(Box.*createHorizontalStrut*(20));  
 totalHBox.add(rightVBox);  
 aboutSoftPanel.add(totalHBox, BorderLayout.NORTH);  
 closeButton = *new* JButton("关闭");  
  
 Box buttonHBox = Box.*createHorizontalBox*();  
 buttonHBox.add(closeButton);  
 aboutSoftPanel.add(buttonHBox, BorderLayout.EAST);  
  
 *return* aboutSoftPanel;  
 }  
  
  
 @Override  
 *public void* actionPerformed(ActionEvent e) {  
 *// “关闭”按钮的事件处理  
 if* (e.getSource() == closeButton) {  
 *// 设置该Dialog不显示即可  
 this*.setVisible(*false*); }  
 }  
  
  
 @Override  
 *public void* mouseClicked(MouseEvent e) {  
 *// 鼠标点击事件  
 // 通过电脑本地打开默认浏览器然后再打开URI所指向的位置* Desktop desktop = Desktop.*getDesktop*();  
 *try* {  
 desktop.browse(*new* URI("https://www.icourse163.org/" )); }  
 *catch* (IOException | URISyntaxException e1) {  
 e1.printStackTrace();  
 }  
 }  
  
  
 @Override  
 *public void* mousePressed(MouseEvent e) {  
 *// 鼠标按下事件* hyperlinkLabel.setCursor(*new* Cursor(Cursor.HAND\_CURSOR));  
 hyperlinkLabel.setForeground(*new* Color(0, 0, 0));  
 }  
  
}

**3.项目总结**

本系统能够很好的完成对图书信息的增删改查等功能。在设计在采用了null的布局方式，使得界面更加的灵活。在进行系统的设计时充分的利用了处理事件，组件等 Java Swing的知识，并且与MySQL数据库相关联，保证了系统功能的实现。通过本项目的设 计实践，掌握了 GUI 程序设计、Java 操作后台数据库等概念。