**基于Java Swing 开发之图书管理系统**

1 需求分析

1.1 开发环境

系统：Win11

IDE：IntelliJ IDEA 2022

JDK：1.8.0

数据库：MySQL 5.7

数据库工具：SQLyog Community

项目结构图如下：

├─.idea

│ └─libraries

└─src

├─com

│ └─book

│ ├─lib 存放jar包依赖

│ ├─pojo 存放实体类

│ └─utils 存放工具类

├─images 图片资源

└─window 窗口类

**1.2 项目背景**

近年来，随着社会、经济的发展，和人对书籍的需求的增长，许多图书馆的规模都在不断扩大，图书数量、有关图书的各种信息量、来图书馆的读者也成倍增加。巨大的信息量，使传统的人工式管理方法面临着极大的挑战，按照传统的方式管理相关信息，会导致图书馆管理上的混乱，人力与物力过多浪费，信息的准确性和安全性也禁不起推敲。图书馆混乱的管理方式，会使图书馆的负担过重，影响整个图书馆的运作和控制管理。

好在IT行业和Internet的飞速发展，为图书馆的管理方式带来了革新。一套合理、有效，规范和实用的图书管理系统，对图书资料和用户信息进行集中统一的管理。另一方面，图书馆的客人也可以通过该系统查询书籍，自助借书还书，节省了大量的人力物力，实现了信息的自动化处理，提高了处理的及时性和正确性。

**1.3 项目目标**

本文介绍了图书管理系统开发的整个过程，该系统通过使用MySQL数据库作为后台数据库，通过使用三层架构组织各个模块。实现管理员的登录、图书信息的增加、删除、修改，图书分类以图书馆分类方法分类和图书信息的显示，以及显示借阅书籍和借阅书籍的归还。本系统实现了图书的信息化管理，完成了图书管理系统的基本功能。该系统主要包括用户登录子系统、用户管理子系统、图书管理子系统、图书借阅信息与归还子系统三个部分，各部分主要功能如下： 用户登录子系统：用户登录。 用户管理子系统：用于用户信息的管理，包括用户信息的注册。 图书管理子系统：用于图书信息的管理，如图书信息查询、图书信息修改、图书信息删除、图书信息增加。 图书借阅与归还子系统：包括图书借阅信息显示以及归还图书。

同时对于管理员来说，还要实现更改密码功能。使用Java + MySQL设计，利用Java Swing设计窗体，通过表格组件与数据库的交互显示当前图书全部信息。通过文本框和按钮与表格和数据库的交互显示增加、删除、修改数据库数据后图书的信息。

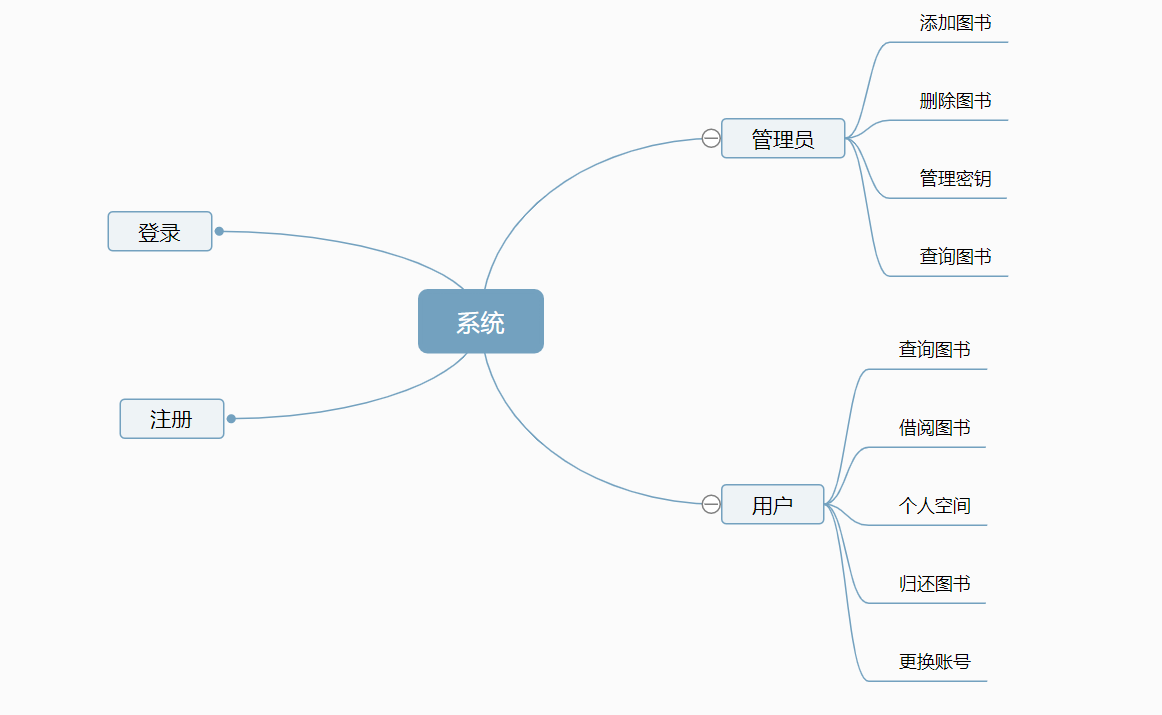
(1)系统能够提供友好的用户界面，使用户能够一目了然的使用本系统

(2)管理员能够更方便的查看、管理图书信息和用户信息

(3)系统能够起到提高图书管理效率的目的

**2 系统模块及功能**

项目流程图：



**2.1 注册模块**

在此模块中，用户可以设置自己的用户名、密码等信息，最后确认密码完成注册



代码实现：

package window;

import com.book.utils.JdbcUtils;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class RegisterFrame {

public static void main(String[] args) {

new RegisterFrame();

}

JTextField userText = null;

JPasswordField pwdText1 = null;

JPasswordField pwdText2 = null;

String userName = null;

String pwd1 = null;

String pwd2 = null;

JButton signBtn = null;

JButton returnBtn = null;

public RegisterFrame() {

JFrame frame = new JFrame("用户注册页面");

frame.setSize(600, 500);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(frame.EXIT\_ON\_CLOSE);

//frame.setDefaultCloseOperation(frame.DISPOSE\_ON\_CLOSE);

JPanel panel = new JPanel(null);

JLabel userLabel = new JLabel("输入用户名：");

userLabel.setBounds(160, 90, 80, 25);

panel.add(userLabel);

userText = new JTextField(22);

userText.setBounds(250, 90, 170, 35);

panel.add(userText);

JLabel pwdLabel1 = new JLabel("设置新密码：");

pwdLabel1.setBounds(160, 150, 80, 25);

panel.add(pwdLabel1);

pwdText1 = new JPasswordField(22);

pwdText1.setBounds(250, 150, 170, 35);

panel.add(pwdText1);

JLabel pwdLabel2 = new JLabel("确认新密码：");

pwdLabel2.setBounds(160, 210, 80, 25);

panel.add(pwdLabel2);

pwdText2 = new JPasswordField(22);

pwdText2.setBounds(250, 210, 170, 35);

panel.add(pwdText2);

signBtn = new JButton("注册");

signBtn.setBounds(200, 260, 80, 35);

panel.add(signBtn);

returnBtn = new JButton("重新登录");

returnBtn.setBounds(340, 260, 90, 35);

panel.add(returnBtn);

//注册按钮

signBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//获取注册信息

userName = userText.getText().trim();

pwd1 = pwdText1.getText().trim();

pwd2 = pwdText2.getText().trim();

//判断两次输入的密码是否一样

if (pwd1.equals(pwd2)) {

int i = addUser(); //如果两次密码一样，再调用添加用户方法，并用i借书返回值

if (i > 0) { //说明sql语句添加成功

JOptionPane.showMessageDialog(

panel,

"注册成功！",

"提示",

JOptionPane.INFORMATION\_MESSAGE);

clearText(); //清空注册信息

}

} else {

JOptionPane.showMessageDialog(

panel,

"两次输入的密码不一致，请重新输入！",

"注意",

JOptionPane.WARNING\_MESSAGE);

}

}

});

//返回登录按钮事件

returnBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new LoginFrame();

frame.setVisible(false);

}

});

frame.add(panel);

frame.setVisible(true);

}

//获取注册信息,并注册用户

public int addUser() {

int i = JdbcUtils.update("insert into `user` values (?,?)", new Object[]{

userName, pwd1

});

return i;

}

//清空注册信息

public void clearText() {

userText.setText("");

pwdText1.setText("");

pwdText2.setText("");

}

}

**2.2 登录模块**

在这个模块中，用户通过输入姓名、密码无误后，即可登录图书管理系统用户界面



代码实现：

package window;

import com.book.pojo.User;

import com.book.utils.JdbcUtils;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class LoginFrame {

String name2 = null;

String pwd2 = null;

public LoginFrame() {

JFrame frame = new JFrame("图书管理系统");

frame.setSize(600, 500);

frame.setLocationRelativeTo(null);

// frame.setDefaultCloseOperation(frame.EXIT\_ON\_CLOSE);

// 把窗口关闭交给窗口监听器处理

frame.setDefaultCloseOperation(frame.DO\_NOTHING\_ON\_CLOSE);

JPanel panel = new JPanel(null);

JLabel userLabel = new JLabel("用户名：");

userLabel.setBounds(170, 140, 80, 25);

panel.add(userLabel);

JTextField userText = new JTextField(22);

userText.setBounds(240, 140, 170, 35);

panel.add(userText);

JLabel pwdLabel = new JLabel("密 码：");

pwdLabel.setBounds(170, 200, 80, 25);

panel.add(pwdLabel);

JPasswordField pwdText = new JPasswordField(22);

pwdText.setBounds(240, 200, 170, 35);

panel.add(pwdText);

JButton loginBtn = new JButton("登录");

loginBtn.setBounds(240, 270, 60, 35);

panel.add(loginBtn);

JButton registerBtn = new JButton("注册");

registerBtn.setBounds(340, 270, 60, 35);

panel.add(registerBtn);

// 登录按钮添加事件

loginBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// 处理事件

// 得到文本框中的内容，并去空格，转为字符串

String text1 = userText.getText().trim();

String text2 = pwdText.getText().trim();

// 调用isUserTable方法，并把输入框中的信息作为实参传过去，得到一个boolean值

boolean bool = isUserTable(text1, text2);

if (bool) { // 如果bool是true时

new UserFrame(); // 打开用户界面

System.out.println("登录成功..");

// 登录成功后关闭当前窗口

frame.setVisible(false);

} else if (text1.length() == 0 || text2.length() == 0) {

JOptionPane.showMessageDialog(

loginBtn,

"用户名或密码不能为空，请输入！",

"注意",

JOptionPane.WARNING\_MESSAGE);

} else {

JOptionPane.showMessageDialog(

loginBtn,

"你输入的用户名或密码有误，请重新输入！",

"提示",

JOptionPane.INFORMATION\_MESSAGE);

}

}

});

// 注册按钮添加事件

registerBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new RegisterFrame();

frame.setVisible(false); // 设置当前窗口不可见

}

});

// 窗口监听器，正在被关闭时，给出提示框

frame.addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

int result = JOptionPane.showConfirmDialog(

frame,

"确认退出",

"提示",

JOptionPane.YES\_NO\_OPTION);

if (result == JOptionPane.YES\_OPTION) {

System.exit(0);

}

}

});

frame.add(panel);

frame.setVisible(true);

}

// 得到数据库中所有用户的信息，并把每一个用户放到用户集合中，返回集合列表

public List<User> findAllUsers() {

ResultSet rs = JdbcUtils.getRs("select \* from `user`");

List<User> list = new ArrayList();

User users = null;

try {

while (rs.next()) {

users = new User();

users.setUserName(rs.getString("user\_name"));

users.setPassword(rs.getString("pwd"));

list.add(users);

}

return list;

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return list;

}

// 判断输入框中的姓名和密码等数据，是否为用户表中的信息，返回true或false

public boolean isUserTable(String userTxt, String pwd) {

List<User> list = findAllUsers();

boolean bools = false;

System.out.println("集合里有多少个用户：" + list.size());

User[] user = new User[list.size()];

for (int i = 0; i < list.size(); i++) {

// 循环遍历集合中的所有用户，放到用户数组里

user[i] = list.get(i);

// 判断数组中的每一个用户，只要有用户的姓名和密码信息与输入框中的数据相同，则返回true

if (userTxt.equals(user[i].getUserName()) && pwd.equals(user[i].getPassword())) {

bools = true; // 如果输入框中姓名、密码等信息与数据库中用户表的某一个用户信息相同

}

// System.out.println(user[i].getUserName() + " " + user[i].getPassword());

}

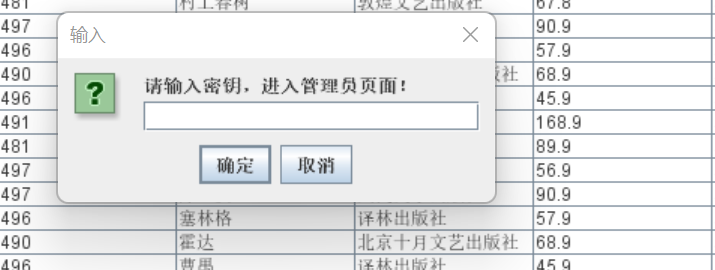
return bools;

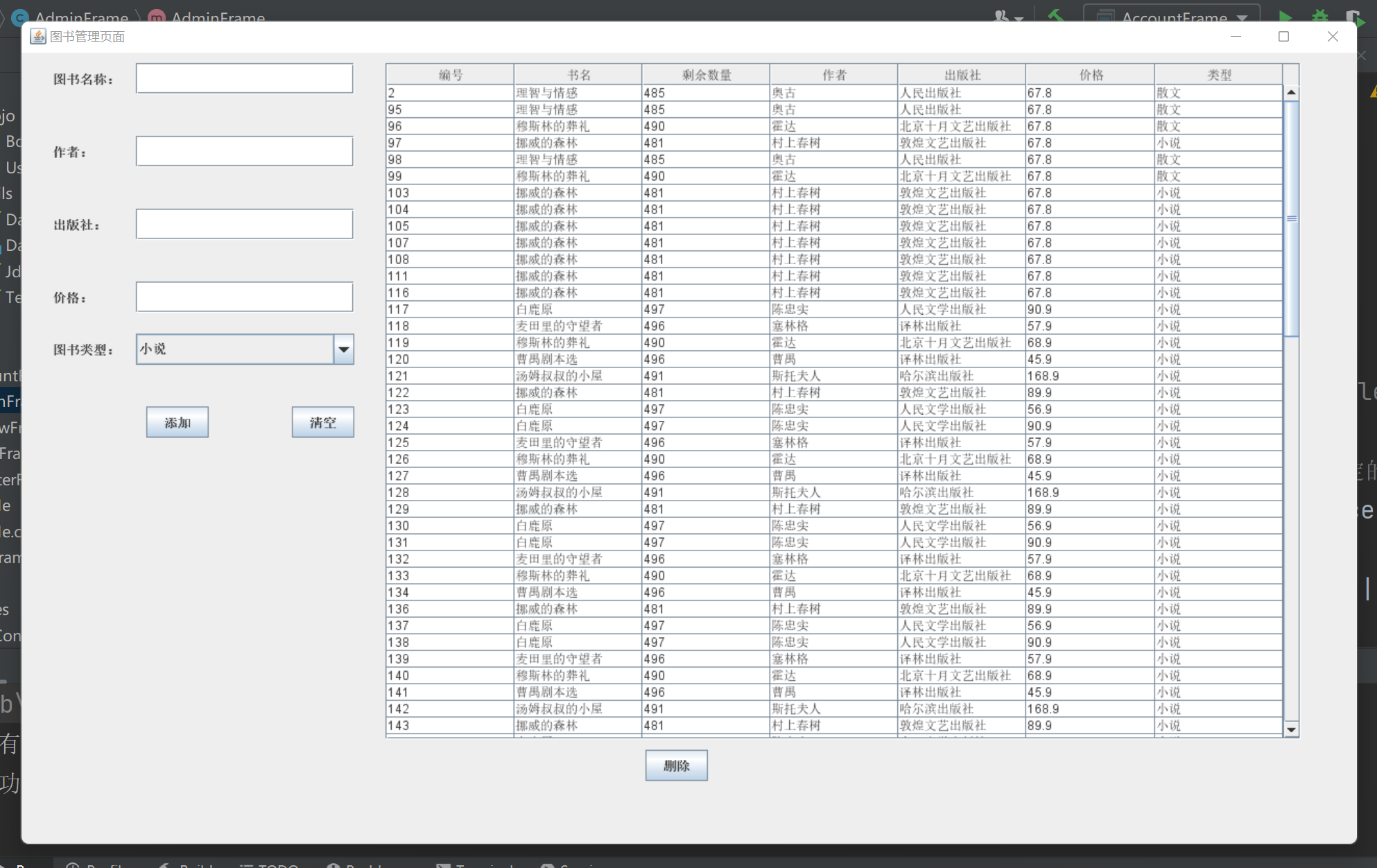
}

}

**2.3 管理员模块**

用户输入管理员密钥即可进入管理员界面





代码实现：

package window;

import com.book.pojo.Book;

import com.book.utils.JdbcUtils;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.event.\*;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class AdminFrame extends JFrame {

//声明组件 //图书名称 //作者 //出版社 //价格

public JLabel label1 = null; //图书名称

public JLabel label2 = null; //作者

public JLabel label3 = null; //出版社

public JLabel label4 = null; //价格

public JLabel label5 = null; //图书类型

public JTextField field1 = null; //图书名称

public JTextField field2 = null; //作者

public JTextField field3 = null; //出版社

public JTextField field4 = null; //价格

JComboBox box = null;

JButton button1 = null; //添加

JButton button2 = null; //清空

JButton button3 = null; //删除

static DefaultTableModel dtm = null; //行和列

JTable table = null;

JScrollPane scrollPane = null;

int id = 0; //定义图书编号

Object[] comboBoxValues = null; //定义下拉框里的所有值

Object itemValue = null; //定义下拉框中被选定的那一项值

JPopupMenu popupMenu = null; //弹出菜单

public AdminFrame() {

this.setTitle("图书管理页面");

this.setSize(1300, 800);

this.setLocationRelativeTo(null);

this.setLayout(null);

label1 = new JLabel("图书名称：");

label1.setBounds(30, 10, 70, 30);

field1 = new JTextField();

field1.setBounds(110, 10, 210, 30);

label2 = new JLabel("作者：");

label2.setBounds(30, 80, 70, 30);

field2 = new JTextField();

field2.setBounds(110, 80, 210, 30);

label3 = new JLabel("出版社：");

label3.setBounds(30, 150, 70, 30);

field3 = new JTextField();

field3.setBounds(110, 150, 210, 30);

label4 = new JLabel("价格：");

label4.setBounds(30, 220, 70, 30);

field4 = new JTextField();

field4.setBounds(110, 220, 210, 30);

label5 = new JLabel("图书类型：");

label5.setBounds(30, 270, 70, 30);

//调用getAllComboboxValue方法，用全局变量 Object[] comboBoxValues来接收所返回的下拉框里的所有项的值

comboBoxValues = getAllComboboxValue();

box = new JComboBox(comboBoxValues); //把所有项的值给到下拉框

box.addItem("诗歌");

box.addItem("白话文");

box.addItem("玄幻");

box.addItem("小说");

//下拉框事件

box.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

itemValue = box.getSelectedItem(); //得到下拉框被选择这一项的值

//System.out.println(itemValue);

}

});

box.setBounds(110, 270, 210, 30);

//按钮

button1 = new JButton("添加");

button2 = new JButton("清空");

button3 = new JButton("删除");

button1.setBounds(120, 340, 60, 30);

button2.setBounds(260, 340, 60, 30);

button3.setBounds(600, 670, 60, 30);

//表格模型

dtm = new DefaultTableModel();

//表头 String[]

String[] head = {"编号", "书名", "剩余数量", "作者", "出版社", "价格", "类型"};

for (int i = 0; i < head.length; i++) {

dtm.addColumn(head[i]);

}

//数据填充到表格模型里

fillTable();

//将表格模型交割table

table = new JTable(dtm);

//设置表格字体和颜色

// table.setFont(new Font("楷体", Font.PLAIN, 18));

// table.setForeground(Color.black);

//设置表头字体和颜色

// table.getTableHeader().setFont(new Font("楷体", Font.BOLD, 22));

// table.getTableHeader().setBackground(Color.pink);

//设置被选中行的背景

// table.setSelectionBackground(Color.lightGray);

//将表格放入滚动面板

scrollPane = new JScrollPane(table);

scrollPane.setBounds(350, 10, 880, 650);

//添加按钮事件

button1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//获取文本框和下拉框的值 //图书名称 作者 出版社 价格

String text1 = field1.getText(); //图书名称 去空格

String text2 = field2.getText(); //作者

String text3 = field3.getText(); //出版社

String text4 = field4.getText(); //价格

Object price = (text4); //把价格这个字符串转换成double类型

//新建一个Object[]类型的数组

//依次把 图书名称、作者、出版社、价格、以及在下拉框中被选定的值全部放到数组fieldTexts里

Object[] fieldTexts = {text1, text2, text3, price, itemValue};

if (text1.length() == 0 || text2.length() == 0 || text3.length() == 0 || text4.length() == 0) {

JOptionPane.showMessageDialog(

button1,

"请在所有文本框内输入完整信息！",

"注意",

JOptionPane.WARNING\_MESSAGE);

} else {

//调用该类中的 addBooks方法，并传入一个Object[]类型的实参数组fieldTexts

//用int类型i来接收，返回受影响的记录(元组)个数

int i = addBooks(fieldTexts);

//如果返回值大于0，说明sql语句执行成功，即已经向表中插入了数据，此时先清空原先表格，然后在填充新数据刷新

if (i > 0) {

JOptionPane.showMessageDialog(

button2,

"图书添加成功",

"提示",

JOptionPane.INFORMATION\_MESSAGE);

clearTable();

fillTable();

}

}

}

});

//清空按钮事件

button2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//点击清空按钮时，把文本框里的内容变成空字符串

field1.setText("");

field2.setText("");

field3.setText("");

field4.setText("");

}

});

//删除按钮事件

button3.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int result = JOptionPane.showConfirmDialog(

scrollPane,

"确认删除所选图书吗？",

"提示",

JOptionPane.YES\_NO\_OPTION);

if (result == JOptionPane.YES\_OPTION) {

//调用删除图书方法 deleteBooksById，当点击删除按钮时就执行方法

int i = deleteBooksById(id); //把选中的id传过去

//调用deleteBooksById方法后，用i来接收返回值，即有多少行受影响

if (i > 0) {

clearTable();

fillTable();

}

}

}

});

//表格添加一个鼠标监听事件，当用户用鼠标对表格进行操作时，执行mousePressed鼠标按压点击方法

table.addMouseListener(new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

//super.mousePressed(e);

//得到选中的行数

int row = table.getSelectedRow();

//表格的行和列都从下标0开始

//System.out.println("当前在表格中选中的是第"+ row + "行");

//调用getValueAt方法，得到第row行，第column列的值,

//Object valueAt = table.getValueAt(row, 0); //得到你选中的行的第1列的值

//System.out.println(valueAt);

if(row >= 0) { //当已经选中一行的时候才能获得图书ID，才可以右键弹出菜单

id = (int) table.getValueAt(row, 0); //id是全局变量，强制转换为int类型

}

}

});

this.add(label1);

this.add(field1);

this.add(label2);

this.add(field2);

this.add(label3);

this.add(field3);

this.add(label4);

this.add(field4);

this.add(label5);

this.add(box);

this.add(button1);

this.add(button2);

this.add(button3);

this.add(scrollPane);

this.setDefaultCloseOperation(this.EXIT\_ON\_CLOSE);

this.setVisible(true);

}

//查询图书数据,并获取

public List<Book> findAllBook() {

ResultSet rs = JdbcUtils.getRs("select \* from tb\_book");

List<Book> list = new ArrayList<>();

Book book = null;

try {

while (rs.next()) {

book = new Book();

book.setBookId(rs.getInt("bookId"));

book.setBookName(rs.getString("bookName"));

book.setBookNumber(rs.getInt("bookNumber"));

book.setBookAuthor(rs.getString("bookAuthor"));

book.setBookPublish(rs.getString("bookPublish"));

book.setBookPrice(rs.getDouble("bookPrice"));

book.setBookType(rs.getString("bookType"));

list.add(book);

}

return list;

} catch (SQLException e) {

e.printStackTrace();

}

return list; //null

}

//把获取到的图书信息填充到表格

public void fillTable() {

List<Book> books = findAllBook();

for (int i = 0; i < books.size(); i++) {

Book book = books.get(i);

dtm.addRow(new Object[]{

book.getBookId(),

book.getBookName(),

book.getBookNumber(),

book.getBookAuthor(),

book.getBookPublish(),

book.getBookPrice(),

book.getBookType()

});

}

}

//清空表格数据方法，并刷新

public void clearTable() {

dtm.getDataVector().clear(); //清空表格

dtm.fireTableDataChanged(); //刷新数据

}

//删除图书方法，根据图书编号id来删除所选的书籍信息 , 返回返回有多少行数据受影响，int值

public int deleteBooksById(int id) { //形参id，接收传入选中的id

//调用工具类的update方法来实现删除书籍，并得到返回值，返回的是数据库表执行sql语句后受影响的行数

int i = JdbcUtils.update("delete from tb\_book where bookId=?", new Object[]{id});

return i;

}

//获取下拉框里每项的所有值，并返回一个Object[]数组

public Object[] getAllComboboxValue() {

List<String> list = new ArrayList<>();

Object[] strs = null;

ResultSet rs = JdbcUtils.getRs("select distinct bookType from tb\_book");

try {

while (rs.next()) {

String type = rs.getString("bookType");

list.add(type);

strs = list.toArray();

}

return strs;

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return strs;

}

//添加图书方法,用 obj接收一个Object[]类型的实参数组，并返回受影响的行数int值

public int addBooks(Object[] obj) {

//调用工具类中的update增删改查方法，返回受影响的记录(元组)个数，并用int值 i 来接收

//如果返回值大于0，说明sql语句执行成功，即已经向表中插入了数据

int i = JdbcUtils.update(

"insert into tb\_book (bookName,bookAuthor,bookPublish,bookPrice,bookType) values (?,?,?,?,?)",

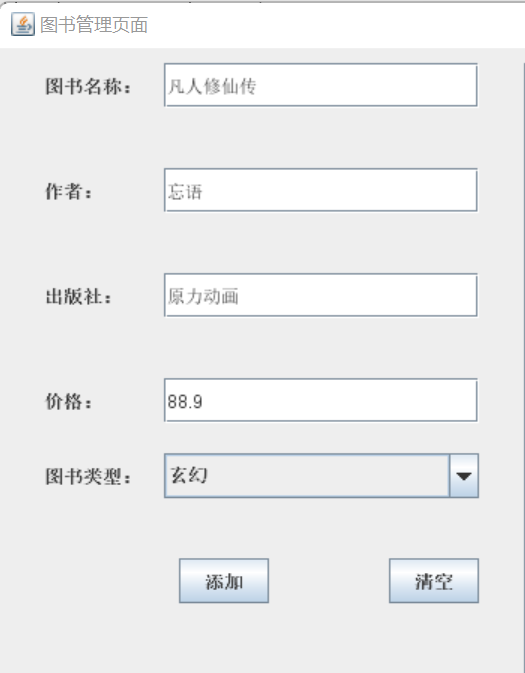
obj); //此处的 obj 为调用方法时参入的实参数组

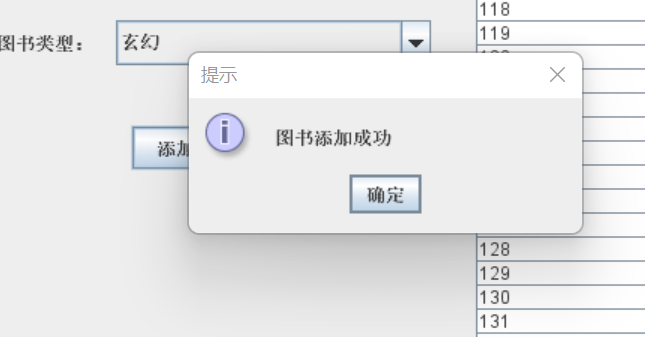
return i;

}

}

**2.3.1 添加图书**





具体实现：

//添加图书方法,用 obj接收一个Object[]类型的实参数组，并返回受影响的行数int值

public int addBooks(Object[] obj) {

//调用工具类中的update增删改查方法，返回受影响的记录(元组)个数，并用int值 i 来接收

//如果返回值大于0，说明sql语句执行成功，即已经向表中插入了数据

int i = JdbcUtils.update(

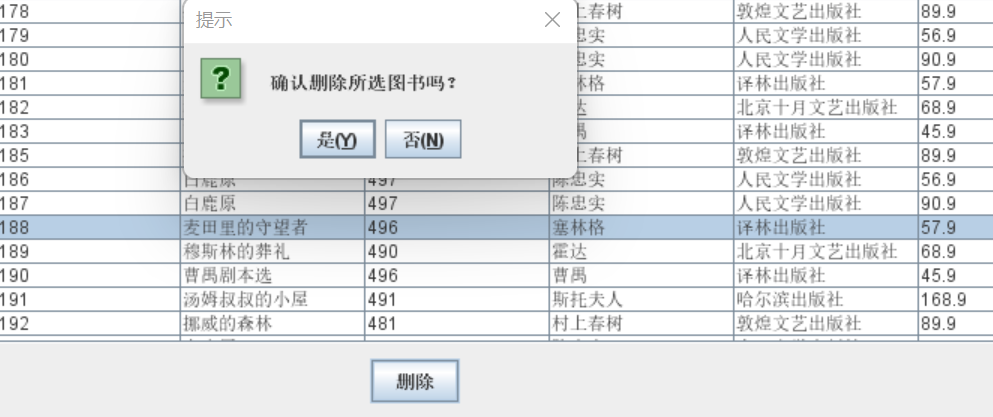
"insert into tb\_book (bookName,bookAuthor,bookPublish,bookPrice,bookType) values (?,?,?,?,?)",

obj); //此处的 obj 为调用方法时参入的实参数组

return i;

}

**2.3.2 删除图书**



具体实现：

//删除图书方法，根据图书编号id来删除所选的书籍信息 , 返回返回有多少行数据受影响，int值

public int deleteBooksById(int id) { //形参id，接收传入选中的id

//调用工具类的update方法来实现删除书籍，并得到返回值，返回的是数据库表执行sql语句后受影响的行数

int i = JdbcUtils.update("delete from tb\_book where bookId=?", new Object[]{id});

return i;

}

**2.3.3 查询图书**

代码实现：

//查询图书数据,并获取

public List<Book> findAllBook() {

ResultSet rs = JdbcUtils.getRs("select \* from tb\_book");

List<Book> list = new ArrayList<>();

Book book = null;

try {

while (rs.next()) {

book = new Book();

book.setBookId(rs.getInt("bookId"));

book.setBookName(rs.getString("bookName"));

book.setBookNumber(rs.getInt("bookNumber"));

book.setBookAuthor(rs.getString("bookAuthor"));

book.setBookPublish(rs.getString("bookPublish"));

book.setBookPrice(rs.getDouble("bookPrice"));

book.setBookType(rs.getString("bookType"));

list.add(book);

}

return list;

} catch (SQLException e) {

e.printStackTrace();

}

return list; //null

}

//把获取到的图书信息填充到表格

public void fillTable() {

List<Book> books = findAllBook();

for (int i = 0; i < books.size(); i++) {

Book book = books.get(i);

dtm.addRow(new Object[]{

book.getBookId(),

book.getBookName(),

book.getBookNumber(),

book.getBookAuthor(),

book.getBookPublish(),

book.getBookPrice(),

book.getBookType()

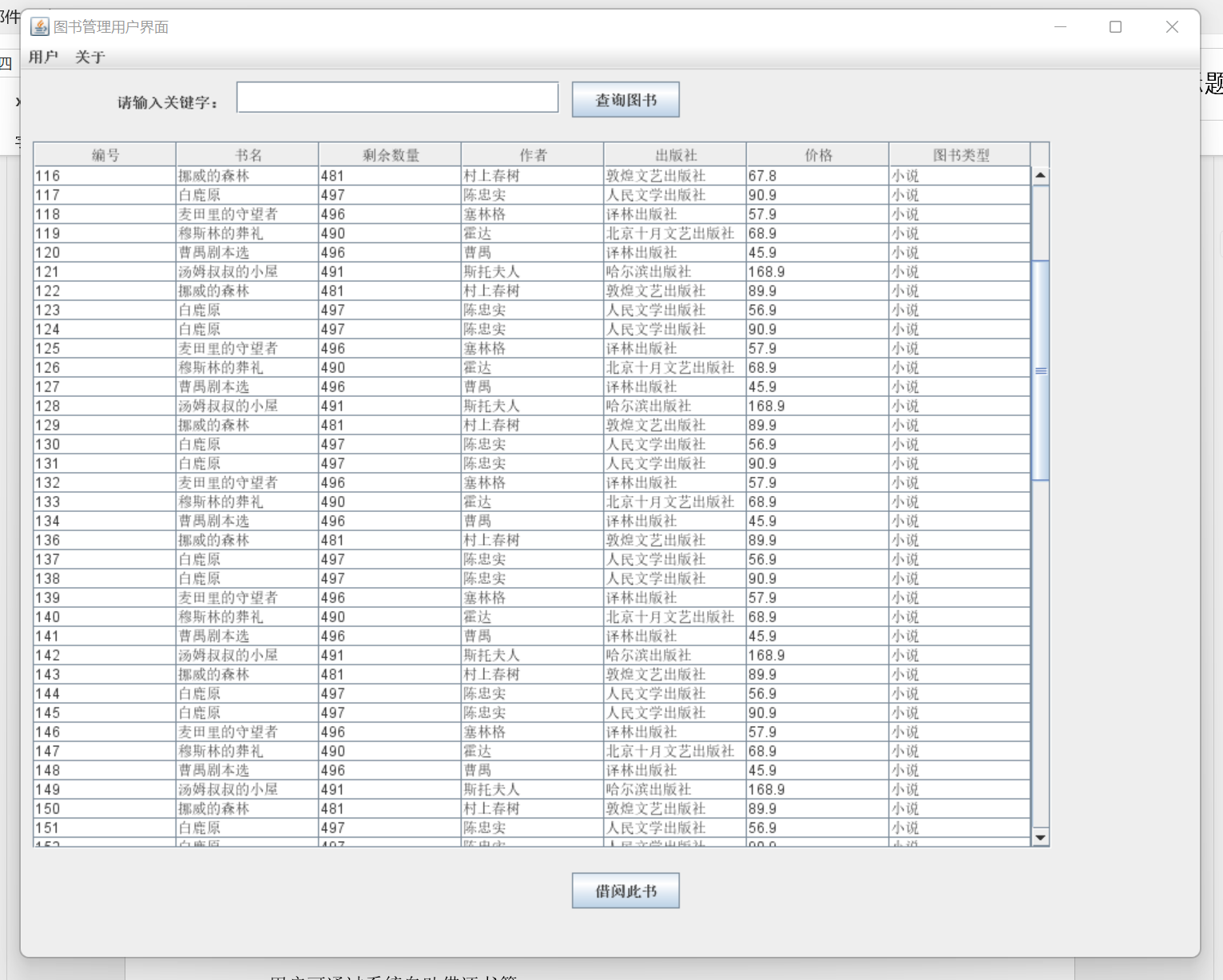
});

}

}

**2.4 用户模块**

用户登录后，可在此页面中查看到数据库中所有的图书信息，界面如下：



代码实现：

package window;

import com.book.pojo.Book;

import com.book.utils.JdbcUtils;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class UserFrame extends JFrame {

public JMenuBar menuBar = null;

public JMenu menu1 = null;

public JMenu menu2 = null;

public JMenuItem menuItem1 = null;

public JMenuItem menuItem2 = null;

public JMenuItem menuItem3 = null;

public JMenuItem menuItem4 = null;

public JMenuItem menuItem5 = null;

public JLabel label1 = null;

public JTextField field = null;

JButton button1 = null;

JButton button2 = null;

DefaultTableModel dtm = null;

JTable table = null;

JScrollPane scrollPane = null;

Object name = null; //根据被选择的图书名来借阅

public UserFrame() {

this.setSize(1000, 800);

this.setLayout(null);

this.setLocationRelativeTo(null);

this.setTitle("图书管理用户界面");

this.setDefaultCloseOperation(this.EXIT\_ON\_CLOSE);

//菜单栏

menuBar = new JMenuBar();

menuBar.setSize(1000, 20);

menu1 = new JMenu("用户");

menu2 = new JMenu("关于");

menuItem1 = new JMenuItem("我的借书空间");

menuItem2 = new JMenuItem("更换账号");

menuItem3 = new JMenuItem("退出");

menuItem4 = new JMenuItem("管理员密钥");

menuItem5 = new JMenuItem("帮助");

menu1.add(menuItem1);

menu1.add(menuItem2);

menu1.add(menuItem3);

menu2.add(menuItem4);

menu2.add(menuItem5);

menuBar.add(menu1);

menuBar.add(menu2);

this.add(menuBar);

label1 = new JLabel("请输入关键字：");

label1.setBounds(80, 30, 130, 35);

field = new JTextField();

field.setBounds(180, 30, 270, 27);

button1 = new JButton("查询图书");

//button1.setBorderPainted(false);

button1.setBounds(460, 30, 90, 30);

button2 = new JButton("借阅此书");

button2.setBounds(460, 690, 90, 30);

this.add(label1);

this.add(field);

this.add(button1);

this.add(button2);

dtm = new DefaultTableModel();

String[] head = {"编号","书名", "剩余数量", "作者","出版社","价格","图书类型"};

for (int i = 0; i < head.length; i++) {

dtm.addColumn(head[i]);

}

fillTable();

table = new JTable(dtm);

scrollPane = new JScrollPane(table);

scrollPane.setBounds(10, 80, 850, 590);

this.add(scrollPane);

table.addMouseListener(new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

int i = table.getSelectedRow();

//只有选中行的时候

name = table.getValueAt(i, 1);

//得到书名之后，根据书名来借阅

//System.out.println(name);

}

});

//查询按钮添加事件

button1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//System.out.println(field.getText());

clearTable();

fillTable2(); //把根据输入文本框内容来查询到的图书信息，填充到表格

}

});

//借阅按钮事件

button2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//，且表格中的行处于被选中状态时，才会让数量减一，并给出提示借阅成功

if(table.isRowSelected(table.getSelectedRow())) {

int i = borrowBooks(name); //根据图书名来借阅，用i来接收受影响的行数据

reduceBookNum(); //调用方法，让表格中剩余数量减一

//清空表格，填充数据

clearTable();

fillTable();

if (i > 0) { //说明sql语句执行成功，即借书成功，只有i > 0

//弹出提示

JOptionPane.showMessageDialog(

scrollPane,

"图书借阅成功",

"提示",

JOptionPane.INFORMATION\_MESSAGE);

}

}

}

});

//我的借书空间菜单项添加事件

menuItem1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

new BorrowFrame();

}

});

//管理员密钥菜单项添加事件

menuItem4.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String password = JOptionPane.showInputDialog(scrollPane,

"请输入密钥，进入管理员页面！"

);

if("twb".equals(password)) {

new AdminFrame(); //打开管理员窗口页面

setVisible(false); //设置当前窗口不可见

}

}

});

setVisible(true);

}

//查询所有图书信息

public List<Book> findAllBooks() {

String sql2 = "select \* from tb\_book";

ResultSet rs = JdbcUtils.getRs(sql2);

List<Book> list = new ArrayList<>();

Book book = null;

try {

while (rs.next()) {

book = new Book();

book.setBookId(rs.getInt("bookId"));

book.setBookName(rs.getString("bookName"));

book.setBookNumber(rs.getInt("bookNumber"));

book.setBookAuthor(rs.getString("bookAuthor"));

book.setBookPublish(rs.getString("bookPublish"));

book.setBookPrice(rs.getDouble("bookPrice"));

book.setBookType(rs.getString("bookType"));

list.add(book);

}

return list;

} catch (SQLException e) {

e.printStackTrace();

}

return list;

}

//查询到的所有图书信息填充到表格

public void fillTable() {

List<Book> list = findAllBooks();

for (int i = 0; i < list.size(); i++) {

Book book = list.get(i);

dtm.addRow(new Object[] {

book.getBookId(),

book.getBookName(),

book.getBookNumber(),

book.getBookAuthor(),

book.getBookPublish(),

book.getBookPrice(),

book.getBookType()

});

}

}

//根据输入文本框内容来查询图书，模糊查询

public List<Book> searchAllBooks() {

ResultSet rs = JdbcUtils.getRs(

"select \* from tb\_book " +

"where bookName like '%" + field.getText() + "%'"

+ " or bookAuthor like '%" + field.getText() + "%'"

+ " or bookType like '%" + field.getText() + "%'"

);

// "select \* from tb\_book where bookName like \"%\" ? \"%\" ",new Object[]{field.getText()}

List<Book> list = new ArrayList<>();

Book book = null;

try {

while (rs.next()) {

book = new Book();

book.setBookId(rs.getInt("bookId"));

book.setBookName(rs.getString("bookName"));

book.setBookNumber(rs.getInt("bookNumber"));

book.setBookAuthor(rs.getString("bookAuthor"));

book.setBookPublish(rs.getString("bookPublish"));

book.setBookPrice(rs.getDouble("bookPrice"));

book.setBookType(rs.getString("bookType"));

list.add(book);

}

return list;

} catch (SQLException e) {

e.printStackTrace();

}

return list;

}

//把根据输入文本框内容来查询到的图书信息，填充到表格

public void fillTable2() {

List<Book> list = searchAllBooks();

for (int i = 0; i < list.size(); i++) {

Book book = list.get(i);

dtm.addRow(new Object[] {

book.getBookId(),

book.getBookName(),

book.getBookNumber(),

book.getBookAuthor(),

book.getBookPublish(),

book.getBookPrice(),

book.getBookType()

});

}

}

//借阅图书

public int borrowBooks(Object name) {

//返回一个int,表示有多少行数据受影响

int i = JdbcUtils.update(

"INSERT INTO tb\_borrow (bookName,bookAuthor,bookType)" +

" SELECT DISTINCT bookName,bookAuthor,bookType" +

" FROM tb\_book WHERE bookName='" + name + "'"

);

return i;

}

//当用户点击借阅后，图书剩余数量减一，根据书名

public int reduceBookNum() {

int i = JdbcUtils.update("update tb\_book" +

" set bookNumber = bookNumber - 1" +

" where bookName = '" + name + "'");

return i; //返回有多少受影响的数据行

}

//清空表格数据方法，并刷新

public void clearTable() {

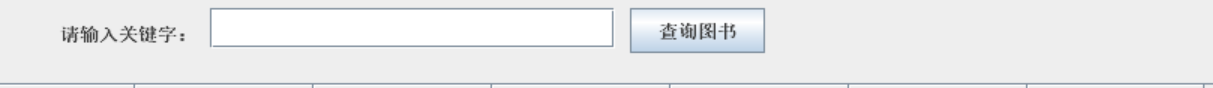
dtm.getDataVector().clear(); //清空表格

dtm.fireTableDataChanged(); //刷新数据

}

}

**2.4.1 查询图书**



用户输入书名、图书类型、作者即可查询相关信息

具体实现：

//根据输入文本框内容来查询图书，模糊查询

public List<Book> searchAllBooks() {

ResultSet rs = JdbcUtils.getRs(

"select \* from tb\_book " +

"where bookName like '%" + field.getText() + "%'"

+ " or bookAuthor like '%" + field.getText() + "%'"

+ " or bookType like '%" + field.getText() + "%'"

);

// "select \* from tb\_book where bookName like \"%\" ? \"%\" ",new Object[]{field.getText()}

List<Book> list = new ArrayList<>();

Book book = null;

try {

while (rs.next()) {

book = new Book();

book.setBookId(rs.getInt("bookId"));

book.setBookName(rs.getString("bookName"));

book.setBookNumber(rs.getInt("bookNumber"));

book.setBookAuthor(rs.getString("bookAuthor"));

book.setBookPublish(rs.getString("bookPublish"));

book.setBookPrice(rs.getDouble("bookPrice"));

book.setBookType(rs.getString("bookType"));

list.add(book);

}

return list;

} catch (SQLException e) {

e.printStackTrace();

}

return list;

}

**2.4.2 借阅图书**

用户选中表格中的某一本书，点击按钮即可借阅图书，具体实现：

//借阅图书

public int borrowBooks(Object name) {

//返回一个int,表示有多少行数据受影响

int i = JdbcUtils.update(

"INSERT INTO tb\_borrow (bookName,bookAuthor,bookType)" +

" SELECT DISTINCT bookName,bookAuthor,bookType" +

" FROM tb\_book WHERE bookName='" + name + "'"

);

return i;

}

//当用户点击借阅后，图书剩余数量减一，根据书名

public int reduceBookNum() {

int i = JdbcUtils.update("update tb\_book" +

" set bookNumber = bookNumber - 1" +

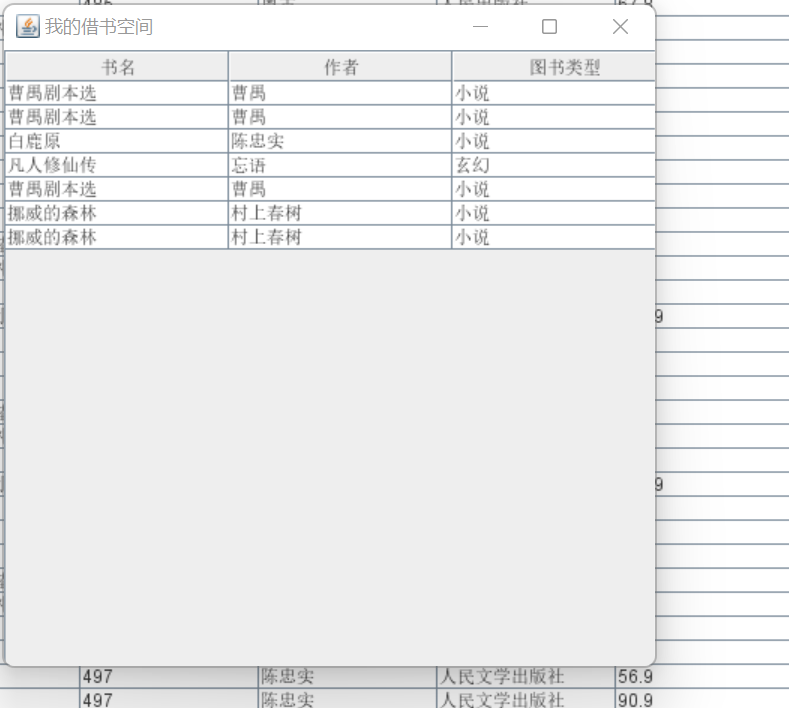
" where bookName = '" + name + "'");

return i; //返回有多少受影响的数据行

}

**2.4.3 个人空间**

用户从图书库中借过来的书放在此模块中，并可以进行退还图书



代码实现：

package window;

import com.book.pojo.Book;

import com.book.utils.JdbcUtils;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class BorrowFrame extends JFrame {

JFrame borrowFrame = null;

Object name = null; //根据被选择的图书名来借阅

DefaultTableModel dtm = null;

JTable table = null;

JScrollPane scrollPane = null;

JPopupMenu popupMenu = null;

JMenuItem Item1 = null;

JMenuItem Item2 = null;

public BorrowFrame() {

borrowFrame = new JFrame("我的借书空间");

borrowFrame.setSize(450, 450);

borrowFrame.setLayout(null);

borrowFrame.setLocationRelativeTo(null);

borrowFrame.setDefaultCloseOperation(borrowFrame.DISPOSE\_ON\_CLOSE);

dtm = new DefaultTableModel();

String[] head = {"书名", "作者", "图书类型"};

for (int i = 0; i < head.length; i++) {

dtm.addColumn(head[i]);

}

fillTable();

table = new JTable(dtm);

scrollPane = new JScrollPane(table);

scrollPane.setBounds(0, 0, borrowFrame.getWidth(), borrowFrame.getHeight());

borrowFrame.add(scrollPane);

popupMenu = new JPopupMenu();

table.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

if(row >= 0) {

name = table.getValueAt(row, 0);

//只有当选中一行的时候，才可以得到图书名，才可以右键弹出菜单

if (e.isMetaDown()) { //判断是否为右键

popupMenu = new JPopupMenu();

//添加子菜单

Item1 = new JMenuItem("退还图书");

Item2 = new JMenuItem("刷新");

popupMenu.add(Item1);

popupMenu.add(Item2);

popupMenu.setPopupSize(70, 60);

popupMenu.show(e.getComponent(), e.getX() + 60, e.getY());

//退还图书菜单项添加事件

Item1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int num = JdbcUtils.update("delete from tb\_borrow where bookName='" + name + "'");

if(num > 0) { //说图书退还成功

//调用方法，增加原来表格图书剩余数量

increaseBookNum(name); //根据书名

JOptionPane.showMessageDialog(scrollPane,

"图书退还成功",

"提示",

JOptionPane.INFORMATION\_MESSAGE);

clearTable();

fillTable();

}

}

});

//刷新菜单项添加事件

Item2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

clearTable();

fillTable();

}

});

}

}

}

});

borrowFrame.setVisible(true);

}

//查询借书表的图书信息

public List<Book> findAllBooks() {

List<Book> list = new ArrayList();

Book books = null;

ResultSet rs = JdbcUtils.getRs("select \* from tb\_borrow");

try {

while (rs.next()) {

books = new Book();

books.setBookName(rs.getString("bookName"));

books.setBookAuthor(rs.getString("bookAuthor"));

books.setBookType(rs.getString("bookType"));

list.add(books);

}

return list;

} catch (

SQLException throwables) {

throwables.printStackTrace();

}

return list;

}

//填充表格

public void fillTable() {

List<Book> list = findAllBooks();

for (int i = 0; i < list.size(); i++) {

Book book = list.get(i);

dtm.addRow(new Object[]{

book.getBookName(),

book.getBookAuthor(),

book.getBookType()

});

}

}

//当你在我的借书空间点击退还图书时，让表格中图书剩余数量加一，根据书名

public int increaseBookNum(Object name) {

int i = JdbcUtils.update("update tb\_book" +

" set bookNumber = bookNumber + 1" +

" where bookName = '" + name + "'");

return i;

}

public void clearTable() {

dtm.getDataVector().clear(); //清空表格

dtm.fireTableDataChanged(); //刷新数据

}

}

**2.4.4 归还图书**

用户在个人空间里，通过右键点击弹出菜单进行操作，具体实现：

if (e.isMetaDown()) { //判断是否为右键

popupMenu = new JPopupMenu();

//添加子菜单

Item1 = new JMenuItem("退还图书");

Item2 = new JMenuItem("刷新");

popupMenu.add(Item1);

popupMenu.add(Item2);

popupMenu.setPopupSize(70, 60);

popupMenu.show(e.getComponent(), e.getX() + 60, e.getY());

//退还图书菜单项添加事件

Item1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int num = JdbcUtils.update("delete from tb\_borrow where bookName='" + name + "'");

if(num > 0) { //说图书退还成功

//调用方法，增加原来表格图书剩余数量

increaseBookNum(name); //根据书名

JOptionPane.showMessageDialog(scrollPane,

"图书退还成功",

"提示",

JOptionPane.INFORMATION\_MESSAGE);

clearTable();

fillTable();

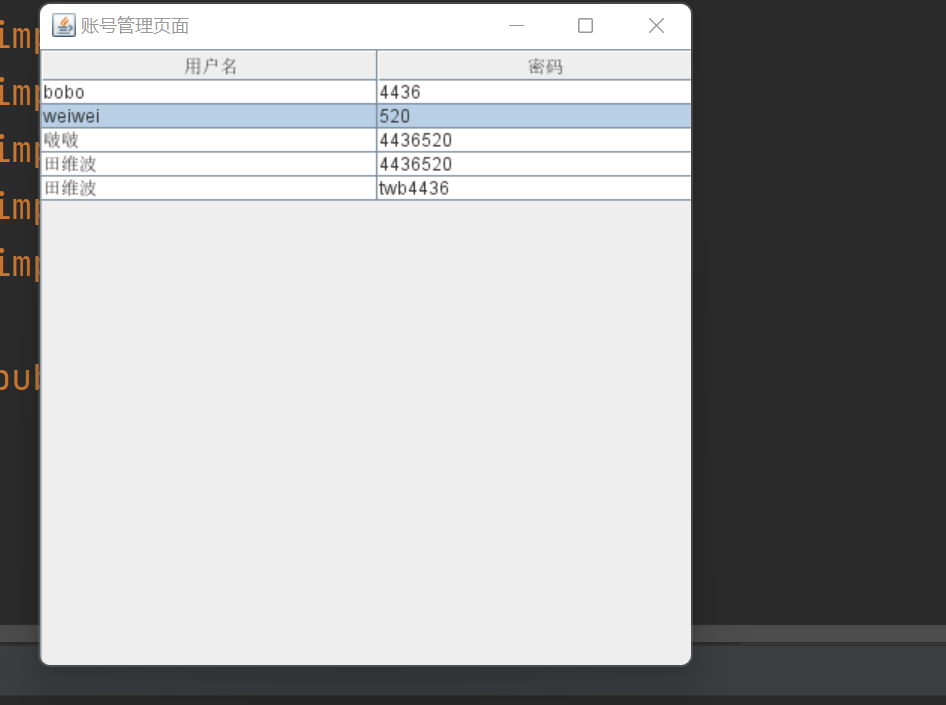
}

}

});

**2.4.5 更换账号**

在此模块中用户可以随便选择一个账号进行登录



代码实现：

package window;

import com.book.pojo.User;

import com.book.utils.JdbcUtils;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class AccountFrame {

JFrame accountFrame = null;

JPopupMenu popupMenu = null;

JMenuItem Item1 = null;

JMenuItem Item2 = null;

Object name = null; //根据被选择的用户名来更换账号

DefaultTableModel dtm = null;

JTable table = null;

JScrollPane scrollPane = null;

public static void main(String[] args) {

new AccountFrame();

}

public AccountFrame() {

accountFrame = new JFrame("账号管理页面");

accountFrame.setSize(450, 450);

accountFrame.setLayout(null);

accountFrame.setLocationRelativeTo(null);

accountFrame.setDefaultCloseOperation(accountFrame.EXIT\_ON\_CLOSE);

dtm = new DefaultTableModel();

String[] head = {"用户名", "密码"};

for (int i = 0; i < head.length; i++) {

dtm.addColumn(head[i]);

}

table = new JTable(dtm);

fillTable();

scrollPane = new JScrollPane(table);

scrollPane.setBounds(0, 0, accountFrame.getWidth(), accountFrame.getHeight());

accountFrame.add(scrollPane);

accountFrame.setVisible(true);

table.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

//得到被选择行的用户名

name = table.getValueAt(row, 0);

if (e.isMetaDown()) {

popupMenu = new JPopupMenu();

Item1 = new JMenuItem("用此账号登录");

Item2 = new JMenuItem("刷新");

popupMenu.add(Item1);

popupMenu.add(Item2);

popupMenu.setPopupSize(120, 65);

popupMenu.show(scrollPane, e.getX() + 50, e.getY());

}

}

});

}

//得到用户信息

public List<User> findAllUsers() {

ResultSet rs = JdbcUtils.getRs("select \* from `user`");

List<User> list = new ArrayList();

User users = null;

try {

while(rs.next()) {

users = new User();

users.setUserName(rs.getString("user\_name"));

users.setPassword(rs.getString("pwd"));

list.add(users);

}

return list;

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return list;

}

//把用户填充到表格

public void fillTable() {

List<User> list = findAllUsers();

for (int i = 0; i < list.size(); i++) {

User user = list.get(i);

dtm.addRow(new Object[] {

user.getUserName(),

user.getPassword()

});

}

}

//清空表格

public void clearTable() {

dtm.getDataVector().clear(); //清空表格

dtm.fireTableDataChanged(); //刷新数据

}

}

**3 JDBC工具类**

JDBC（Java DataBase Connectivity）是Java和数据库之间的一个桥梁，是一个规范而不是一个实现，能够执行SQL语句。它由一组用Java语言编写的类和接口组成。各种不同类型的数据库都有相应的实现，本文中的代码都是针对MySQL数据库实现的。

JDBC的作用：可以通过java代码操作数据库

package com.book.utils;

import java.sql.\*;

//JDBC工具类

public class JdbcUtils {

static String driver = "com.mysql.cj.jdbc.Driver";

static String url = "jdbc:mysql://localhost:3306/library";

static Connection conn = null;

static PreparedStatement pstat = null;

static ResultSet rs = null;

//专门用来获取连接对象，只有有了连接对象，才可能会有 PreparedStatement，ResultSet

public static Connection getConn() {

try {

//注册驱动

Class.forName(driver);

conn = DriverManager.getConnection(url, "root", "twb");

return conn;

} catch (Exception e) {

e.printStackTrace();

}

return conn; //null

}

//查询所有 不带参数 select \* from student;

public static ResultSet getRs(String sql) {

try {

conn = getConn();

pstat = conn.prepareStatement(sql);

rs = pstat.executeQuery();

return rs;

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return rs;

}

public static ResultSet getRs(String sql,Object[] params) {

try {

conn = getConn();

//select \* from tb\_book where bookId=?

pstat = conn.prepareStatement(sql);

//给问号赋值

for (int i = 0; i < params.length; i++) {

pstat.setObject(i+1, params[i]);

}

rs = pstat.executeQuery();

return rs;

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return rs;

}

//增删改的方法 调用该方法时，通过传入的sql语句执行，返回有多少行数据受影响，int值

public static int update(String sql,Object[] params) {

try {

conn = getConn();

//delete from student where bookId=?

pstat = conn.prepareStatement(sql);

//给问号赋值

for (int i = 0; i < params.length; i++) {

pstat.setObject(i+1, params[i]);

}

int i = pstat.executeUpdate(); //返回受影响数据的行数，int值

return i;

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return 0;

}

public static int update(String sql) {

try {

conn = getConn();

pstat = conn.prepareStatement(sql);

int i = pstat.executeUpdate(); //返回受影响数据的行数，int值

return i;

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return 0;

}

public static void close(ResultSet rs, PreparedStatement pstat, Connection conn) {

if(rs != null) {

try {

rs.close();

} catch (SQLException throwables) {

throwables.printStackTrace();

}

}

if(pstat != null) {

try {

pstat.close();

} catch (SQLException throwables) {

throwables.printStackTrace();

}

}

if(conn != null) {

try {

conn.close();

} catch (SQLException throwables) {

throwables.printStackTrace();

}

}

}

public static void main(String[] args) {

conn=getConn();

System.out.println(conn);

}

}

**4 对象实体类**

对java实体类的理解：就是属性类，通常定义在model层里面 ，一般的实体类对应一个数据表，其中的属性对应数据表中的字段。

好处：

1.对对象实体的封装，体现OO思想。

2.属性可以对字段定义和状态进行判断和过滤

3.把相关信息用一个实体类封装后，我们在程序中可以把实体类作为参数传递，更加方便。

说白了就是为了让程序员在对数据库操作的时候不用写SQL语句，就是一个数据库表生成一个类，这样做对数据库操作起来方便编写代码较少提高效率可以使程序员专注逻辑关系 ，实体类就是把对某一个表的操作全写在一个类中。在Java开发中经常要定义一些实体类，这些类的定义的好坏会直接影响，编写代码的质量和难易程度。

**4.1 图书类**

package com.book.pojo;

public class Book {

private int bookId;

private String bookName;

private int bookNumber;

private String bookAuthor;

private String bookPublish;

private double bookPrice;

private String bookType;

public int getBookId() {

return bookId;

}

public void setBookId(int bookId) {

this.bookId = bookId;

}

public String getBookName() {

return bookName;

}

public void setBookName(String bookName) {

this.bookName = bookName;

}

public int getBookNumber() {

return bookNumber;

}

public void setBookNumber(int bookNumber) {

this.bookNumber = bookNumber;

}

public String getBookAuthor() {

return bookAuthor;

}

public void setBookAuthor(String bookAuthor) {

this.bookAuthor = bookAuthor;

}

public String getBookPublish() {

return bookPublish;

}

public void setBookPublish(String bookPublish) {

this.bookPublish = bookPublish;

}

public double getBookPrice() {

return bookPrice;

}

public void setBookPrice(double bookPrice) {

this.bookPrice = bookPrice;

}

public String getBookType() {

return bookType;

}

public void setBookType(String bookType) {

this.bookType = bookType;

}

}

**4.2 用户类**

package com.book.pojo;

public class User {

private String userName = null;

private String password = null;

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**5 数据库表结构的设计**

**5.1 图书信息表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键 | 非空? | 字符集 | 描述 |
| Bookid | Int | 是 | 是 |  | 编号 |
| Bookname | Varchar(50) |  | 是 | Utf8 | 书名 |
| Booknumber | Bigint |  |  |  | 数量 |
| Bookauthor | Varchar(30) |  |  | Utf8 | 作者 |
| Bookpublish | Varchar(30) |  |  | Utf8 | 出版社 |
| Bookprice | Decimal(10,2) |  |  |  | 价格 |
| Booktype | Varchar(10) |  |  | Utf8 | 类型 |

**5.2 借书信息表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键 | 非空? | 字符集 | 描述 |
| Bookname | Varchar(50) |  | 是 | Utf8 | 书名 |
| Bookauthor | Varchar(30) |  |  | Utf8 | 作者 |
| Booktype | Varchar(10) |  |  | Utf8 | 类型 |

**5.3 用户信息表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键 | 非空? | 字符集 | 描述 |
| User\_Name | Varchar(50) |  | 是 | Utf8 | 用户名 |
| Pwd | Varchar(20) |  | 是 | Utf8 | 密码 |

**6 系统特色**

1. 界面友好，操作简单，速度快，效率高
2. 用户可通过系统自助借还书籍
3. 图书信息一目了然，存档及查询管理方便
4. 书籍及用户的相关信息管理规范化、科学化