1. **主要内容**：

应用所学知识，完成一个Java类应用程序或基础类库，结合Java语言基本知识如：字符串、集合、类、数据库操作等，设计开发一个Java类应用程序或基础类库。该软件能实现基本功能，涉及到所用Java库的主要函数。

主要学习内容包括：

（一）进一步学习并熟练掌握Java语言语法和面向对象编程思想。

（二）学习Java库中提供的函数，熟悉Netbeans、eclipse开发环境。

（三）学习软件的设计与开发过程中所需要思想和细节。

**二、基本要求：**

1.资料充分，结构完整，论述清晰。

2.符合我院课程设计报告撰写规范。

3.按时完成课程设计各阶段工作，不突击，不抄袭。

目 录

[1 概述 3](#_Toc28522735)

[1.1需求分析 3](#_Toc28522736)

[1.2 设计思想 3](#_Toc28522737)

[1.3 设计原则 3](#_Toc28522738)

[1.4 课程设计分工情况 4](#_Toc28522739)

[2 总体设计 5](#_Toc28522740)

[2.1 数据库设计 5](#_Toc28522741)

[2.2 系统功能设计 6](#_Toc28522742)

[2.3 系统实现思路 8](#_Toc28522743)

[3 详细设计 9](#_Toc28522744)

[3.1 系统具体实现 9](#_Toc28522745)

[3.2 登录、注册功能的实现 9](#_Toc28522746)

[3.3 主界面功能的实现 14](#_Toc28522747)

[3.4 录入功能的实现 17](#_Toc28522748)

[3.5 借书功能的实现 21](#_Toc28522749)

[4 运行结果与调试 25](#_Toc28522750)

[4.1运行结果 25](#_Toc28522751)

[4.2软件调试 27](#_Toc28522752)

[结束语 28](#_Toc28522753)

[参考资料 30](#_Toc28522754)

# **概述**

## 1.1需求分析

图书管理信息系统需实现的功能如下：

（1）用户管理：包括用户的注册于登录。

（2）图书管理：包括录入、查询、修改和删除图书信息。

（3）借书：包括借阅图书和查看借书记录。

（4）还书：包括还书和查看还书记录。

（5）为了保证系统安全，进入系统时，对用户登录的密码进行加密与解密。

## 1.2 设计思想

（1）该系统的设计分成一些相对独立的模块，这些模块都进行集中式管理。

（2）分层的模块化程序设计思想，而且整个系统采用Java库中自带的类，可以减少代码的量，提高代码的质，可读性强。

（3）本系统合理的设计了数据库，使得应用程序的数据显示为有效的信息。

（4）本系统结合了我们学校图书馆的实际情况，对应用功能进行了设计，这些功能符合学生要求。

（5）该系统中我们对用户的登录密码进行了加密与解密，提高了系统安全性。

## 1.3 设计原则

为了使本系统功能齐全完备，操作简便，最大限度的提高用户的使用的体验，从而满足用户的实际需要，在设计开发过程中遵循了如下原则：

（1）易操作原则：要求设计的系统功能齐全，界面友好，操作方便，必要的地方进行提示。

（2）面向对象思想原则： 为了更好的使得程序编写简单，具备跨平台、可移植性，应采用面向对象编程思想，而并不是面向过程编程思想。

（3）源程序可读性原则：为了便于其他设计，团队人员读懂代码或以后的代码修改，软件升级维护，尽可能做好代码注释工作。

## 1.4 课程设计分工情况

本组共有3人，具体分工情况如下：

……………………………………………………………………………..

# **2 总体设计**

## 2.1 数据库设计

根据图书管理系统的需求分析，设计数据库mybooks，包含4个表，图书信息表（books）、借书记录表（lendrecord）、还书记录表（returnrecord）和用户信息表（user）。表数据结构分别如1-1～1-4所示。

表1-1 图书信息表（books）结构

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | int | 图书顺序号，主键，自增 |
| isbn | varchar | ISBN号 |
| name | varchar | 书名 |
| author | varchar | 作者 |
| publisher | varchar | 出版社 |
| price | varchar | 价格 |
| pubDate | varchar | 出版日期 |
| deposit | varchar | 存放位置 |
| quantify | int | 数量 |
| lend | int | 借出数量 |

表1-2 借书记录表（lendrecord）结构

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | int | 借书记录顺序号，主键，自增 |
| bookId | int | 图书顺序号 |
| userId | bigint | 借书人学号 |
| borrowName | varchar | 借书人姓名 |
| borrowDate | varchar | 借书日期 |
| state | varchar | 还书状态，已还：true |

表1-3 还书记录表（returnrecord）结构

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | int | 还书记录顺序号，主键，自增 |
| bookId | int | 图书顺序号 |
| userId | bigint | 还书人学号 |
| returnName | varchar | 还书人姓名 |
| returnDate | varchar | 借书日期 |

表1-4 用户信息表（user）结构

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | int | 系统操作用户编号，主键自增 |
| userName | bigint | 用户学号，主键 |
| setPassword | varchar | 设置密码 |
| surePassword | varchar | 确认密码 |
| name | varchar | 用户姓名 |
| sex | varchar | 用户性别 |
| brithday | varchar | 出生日期 |

## 2.2 系统功能设计

根据图书管理系统需求分析，设计系统功能如下：

（1）注册：显示注册界面，用户输入自己的学号、密码、姓名、性别、出生日期，单击“注册”按钮后，系统为该用户进行注册，存入数据库中。

（2）登录：显示登录界面，用户输入用户名和密码，单击“登录”按钮后，系统验证用户名和密码是否正确，如正确则进入主界面。

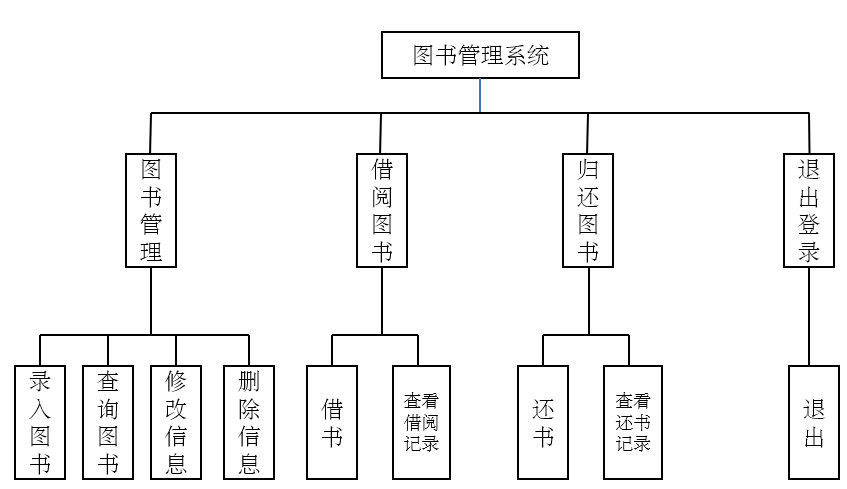
（3）主界面：显示菜单，菜单结构如图1所示

图1 图书管理系统主菜单结构

1. 图书管理：共4项子功能，分别说明如下。

录入图书：用户在主界面菜单中选择此功能时，进入录入界面。用户在录入界面中输入各项图书信息后，单击“提交”按钮，系统将采集界面中的图书信息，将其存入数据库的图书信息表中。

查询图书：用户在主界面菜单中选择此功能时，进入查询界面。系统为用户提供完全信息查询和模糊查询功能。 用户在查询界面中输入书名、作者和出版社信息（或任意的一部分信息，甚至没有输入任何信息）后，选择查询条件为“完全一致”或“模糊查询”，最后单击“查询”按钮，系统按要求查询图书信息表， 查询出图书信息显示于列表中。

修改图书：用户在主界面菜单中选择此功能时，进入修改界面1，首先根据用户输入信息完成图书信息的查询（详细请参见（4）查询图书），用户在列表中选择需修改的图书后，进入修改界面2。用户在修改界面2中完成图书信息的修改后，单击“修改”按钮，系统采集界面2中的信息后，修改数据库中图书信息表中的相应记录。

删除图书：用户在主界面菜单中选择此功能时，进入删除界面。首先根据用户输入信息完成图书信息的查询（详细请参见（4）查询图书），用户在列表中选择需删除的图书后，单击“删除”按钮，系统删除数据库中图书信息表中的相应记录。

1. 借书：共两项子功能,分别说明如下。

借书：用户在主界面菜单中选择此功能时，进入借阅图书界面。首先根据用户输入信息完成图书信息的查询（详细请参见（4）查询图书），用户在列表中选择需借阅的图书后，单击“借书”按钮，系统修改图书信息表中的相应记录（将借出数量加一，如借出数量已经大于该图书的数量，则不加修改，且提示用户该图书已经全部借出），并在借书记录表中登记一条记录（如借书不成功则不予登记）。

查询借书记录：用户在主界面菜单中选择此功能时，进入借书记录界面。系统根据用户输人的信息完成借书记录的查询（查询涉及三个表：借书记录表、图书信息表、用户表），如用户在查出的借书记录列表中选择一条借书记录，单击“查看详细信息”按钮，系统为用户查出所借图书的详细信息。

1. 还书：共两项子功能，分别说明如下。

还书：用户在主界面菜单中选择此功能时，进入还书界面。首先根据用户输入信息完成借书记录的查询（详细请参见（5）查询借书记录），用户在列表中选择需还的图书后，单击“还书”按钮，系统修改图书信息表中的相应记录（将借出数量减1），且修改借书记录表中相应记录的“还书状态”为true，并在还书记录表中登一-条记录。

查询还书记录：用户在主界面菜单中选择此功能时，进入还书记录界面。系统根据用户输入信息完成还书记录的查询（涉及三个表：还书记录表、图书信息表、用户表），如用户在查出的还书记录列表中选择一条还书记录，单击“查看详细信息”按钮，系统为用户查出所还图书的详细信息。

（7）退出登录：用户在主界面菜单中选择此功能时，单击“退出”按钮后，系统会结束主菜单界面，退回到初始时的登录界面。

## 2.3 系统实现思路

为避免图形用户界面中对按钮事件处理的方法中代码堆砌太多，造成程序的结构性太差，系统实现时采用清晰的分层结构模型如图2所示。

数据操作层

数据库

用户

界面层

业务层

图—2本系统采用的三层结构模型

用户通过界面输人信息或单击按钮，界面层通过事件处理控制转向业务层的不同处理方法进行处理，当需要对数据库进行操作时，业务层调用数据操作层的相应数据操作方法完成；之后数据操作层将结果返回业务层继续处理，业务层处理完毕，则再将结果返回到界面层，界面层显示结果，用户由此可进一步在界面上进行操作。

# **3 详细设计**

## 3.1 系统具体实现

在图书管理信息系统中，根据以上设计和实现思路，共写了14个类。其中界面层11个类：LoginManager类、BooksManager类、UpdateFrame类、UpdateFrame1类、SelectFrame类、InsertFrame类、DeleteFrame类、LendFrame类、LendRecordFrame类、ReturnFrame类和ReturnRecordFrame类；业务层1个类：Service类；数据库操作层1个类：DataOperator 类；另外还有一个附加类：MD5类，该类用于添加用户时以及登录时对密码进行加密、解密处理。

在界面层中，LoginManager类用于登录、注册界面； BooksManager 类是主界面；UpdateFrame类用于“查询”图书功能所需界面，以便于修改图书；UpdateFrame1类用于“修改”图书功能所需界面；SelectFrame类用于“检索”图书界面，InsertFrame类用于“添加”图书功能所需界面；DeleteFrame类用于“删除”图书功能所需界面；LendFrmae类用于“借阅”图书功能所需界面；LendRecordFrame类用于查询借书记录所需界面；ReturnFrame类用于还书功能所需界面；ReturnRecordFrame类用于查询还书记录所需界面；

在业务层中,Service类封装了界面层中除主界面之外,各个界面中按钮事件处理时所需调用的业务处理方法,这些方法都是静态的。

在数据库操作层中,DataOperator对象封装了所有对数据库进行操作的方法，这些方法为业务层中的方法调用。

## 3.2 登录、注册功能的实现

1. 效果图如图3-1、3-2所示



图3-1 “登录”功能运行效果 图3-2 “注册”功能运行效果

2. UML图，如图4所示。

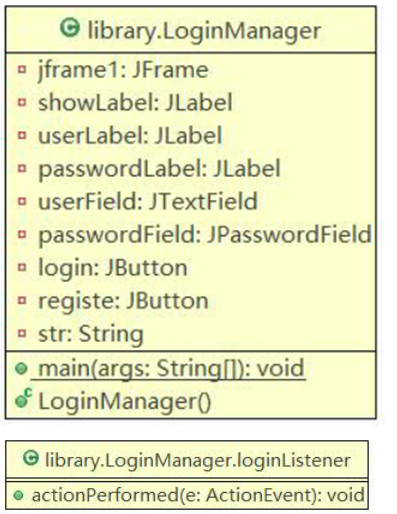


图4 LoginManager类的UML图

封装在LoginManager类中的main(String)方法时系统运行的入口方法。在该方法中，创建LoginManager窗口对象，并为LoginManager窗口注册监听器（使用匿名内部类对象），最后显示loginManager窗口。LoginManager()是构造方法，负责完成登录窗口的初始化，并为login按钮注册动作事件监听器。actionPerformed(ActionEvent)是ActionListener 接口定义的方法。在LoginManager窗口中，当用户输入了用户名和密码，单击“登录”按钮时，执行该方法。在该方法中,，取得用户在userField和passwordField输入的用户名与密码，对密码进行加密后，调用Service类的login(String,string)方法，完成登录操作。如接收login(String, string)方法返回值大于0（返回值为该用户在users表中对应存储的顺序号或标识号），登录成功，则创建主界面BooksManager窗口对象（同时将用户标识号传递给主界面对象）,并为主界面窗口注册窗口监听器(使用匿名类对象)，最后显示主界面。

3. 核心代码如下：

// 重写登录按钮动作事件监听方法

**public** **class** loginListener **implements** ActionListener {

@Override

**public** **void** actionPerformed(ActionEvent e) {

String userName = userField.getText().trim(); // 获取文本字段里的内容（账号）

String password = MD5.*GetMD5Code*(passwordField.~~getText~~().trim());// 获取口令字段里的内容进行加密（密码）

**long** id = Service.*login*(userName, password); // 作为用户标识号，核对登录信息

**if** (id > 0) {// 在数据库中可以匹配

BooksManager app = **new** BooksManager(id); // 创建主界面，返回用户标识号

jframe1.dispose();

; // 释放由此 Window、其子组件及其拥有的所有子组件所使用的所有本机屏幕资源。

} **else** **if** (id == 0) {// 与数据库没有完全匹配

JOptionPane.*showMessageDialog*(jframe1, "您输入的用户名或密码不正确");

} **else** {// 与数据库完全不匹配

JOptionPane.*showMessageDialog*(jframe1, "查询用户表出错！");

}

}

}

// 处理登录界面里的注册按钮监听事件

@Override

**public** **void** actionPerformed(ActionEvent e) {

jframe2.setResizable(**false**); // 设置注册界面大小不可改变

jframe2.getContentPane().setBackground(Color.***PINK***);

jframe2.setVisible(**true**); // 当点击登录界面的注册按钮时设置注册界面可见

jframe1.dispose();

; // 释放由此 Window、其子组件及其拥有的所有子组件所使用的所有本机屏幕资源。

// 清空登录界面各组件内容

userField.setText("");

passwordField.setText("");

Service.*windowClose*(jframe2);// 注册界面的窗口监听事件

jframe2.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***); // 释放注册界面窗口

regisiteButton.addActionListener(**new** regisiteListener2()); // 注册界面里的注册按钮动作监听事件

}

// 处理注册界面里的注册按钮动作监听事件

**public** **class** regisiteListener2 **implements** ActionListener {

@Override

**public** **void** actionPerformed(ActionEvent e) {

**if** (text1.getText().trim().equals("")) {

JOptionPane.*showMessageDialog*(jframe2, "学号项为空，请您重新填写！");

} **else** **if** (text2.~~getText~~().trim().equals("")) {

JOptionPane.*showMessageDialog*(jframe2, "设置密码项为空，请您重新填写！");

} **else** **if** (text3.~~getText~~().trim().equals("")) {

JOptionPane.*showMessageDialog*(jframe2, "确认密码项为空，请您重写填写！");

} **else** **if** (text4.getText().trim().equals("")) {

JOptionPane.*showMessageDialog*(jframe2, "姓名项为空，请您重写填写！");

} // else if (text5.getText().trim().equals("")) {

**else** **if** (str.equals("")) {

JOptionPane.*showMessageDialog*(jframe2, "性别项为空，请您重写填写！");

} **else** **if** (text6.getText().trim().equals("")) {

JOptionPane.*showMessageDialog*(jframe2, "出生日期项为空，请您重写填写！");

} **else** {

// 注册用户信息录入数据库

Service.*regisiteUser*(text1.getText().trim(), text2.~~getText~~().trim(), text3.~~getText~~().trim(),

text4.getText().trim(), /\* text5.getText().trim() \*/str, text6.getText().trim());

}

}

}

## 3.3 主界面功能的实现

1. 效果图，如图5所示。

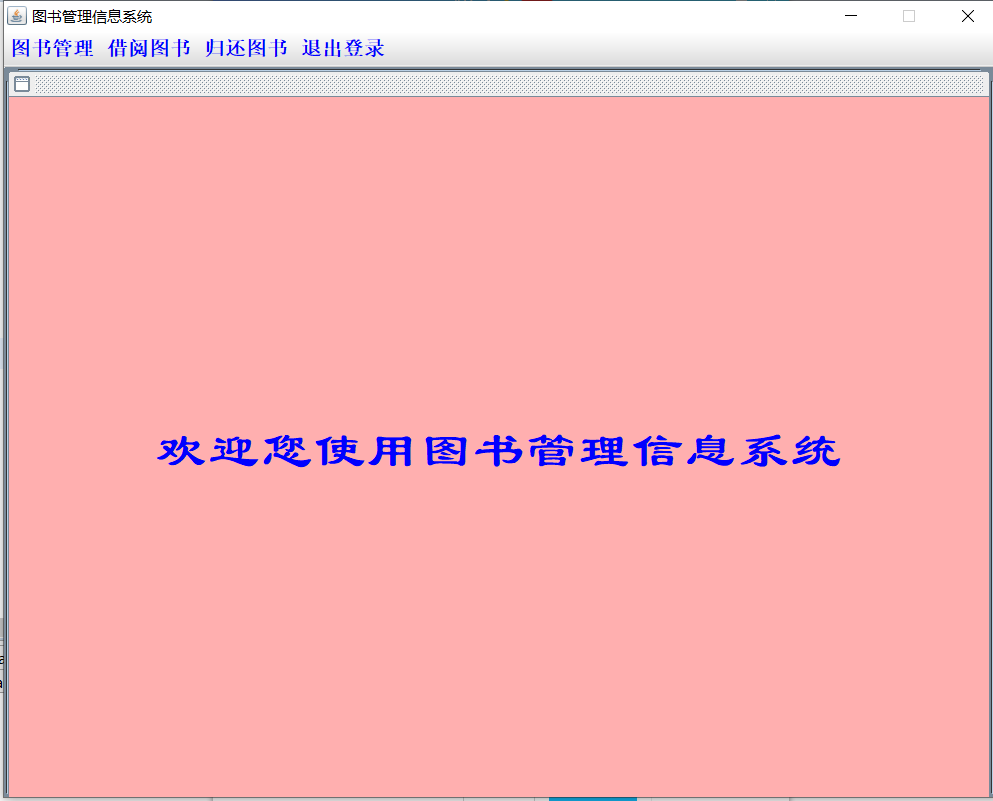


图5 主界面运行效果

2. UML图，如图6所示。

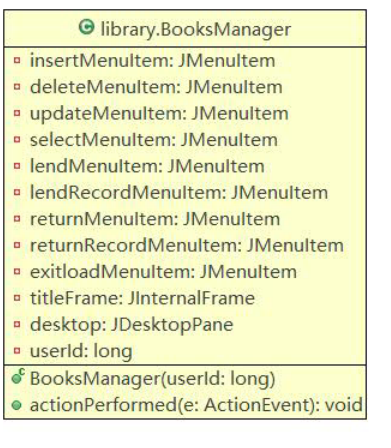


图6 BooksManager类的UML图

userId接收从LoginManager窗口对象传递进来的参数，表示当前用户的标识号。insertMenuItem，selectMenuItem，updateMenuItem，deleteMenuItem，lendMenultem，lendRecordMenuItem，returnMenultem，returnRecordMenultem，exitloadMenuItem，为定义的9个菜单项。desktop为定义的桌面容器。当用户选择一个菜单项时，会创建一个内部窗口框架（JInternalFrame）放在desktop 桌面容器上。桌面容器上放的第一个内部窗口框架是titleFrame显 示标题的内部窗口框架，如图5所示。BooksManager(long)是构造方法，负责完成主界面窗口的初始化，并为各菜单项注册动作监听器。其中lomg类型的参数是从登录窗口对象传递进来的用户标识号(userId)。actionPerformed(ActionEvent)方法是ActionListener接口定义的方法。当用户在主界面中选中不同的菜单项时，执行该方法。在该方法中，根据不同的事件源（用户选中的菜单项），相应对应的事件。

3. 核心代码如下：

@Override

**public** **void** actionPerformed(ActionEvent e) {

Object source = e.getSource(); // 最初发生的Event对象，返回Object

**if** (source == insertMenuItem) {// 如果点击的是录入图书菜单项

InsertFrame insertFrame = **new** InsertFrame(userId);// 创建录入图书界面

**this**.dispose();// 释放由此 Window、其子组件及其拥有的所有子组件所使用的所有本机屏幕资源。

}

**if** (source == deleteMenuItem) {// 如果点击的是删除图书菜单项

DeleteFrame deleteFrame = **new** DeleteFrame(userId); // 删除图书

**this**.dispose();

// 添加删除图书界面关闭监听事件

}

**if** (source == updateMenuItem) {// 如果点击的是修改图书信息菜单项

UpdateFrame1 updateFrame = **new** UpdateFrame1(userId); // 修改图书信息

**this**.dispose(); // 释放由此 Window、其子组件及其拥有的所有子组件所使用的所有本机屏幕资源。

// 添加修改图书信息界面关闭监听事件

}

**if** (source == selectMenuItem) {// 如果点击的是检索图书菜单项

SelectFrame selectFrame = **new** SelectFrame(userId);// 查询图书

**this**.dispose();

// 添加检索图书界面关闭监听事件

}

**if** (source == lendMenuItem) {// 如果点击的是开始借书菜单项

LendFrame lendFrame = **new** LendFrame(userId);// 开始借书

**this**.dispose();

}

**if** (source == lendRecordMenuItem) {// 如果点击的是借书记录菜单项

LendRecordFrame lendRecordFrame = **new** LendRecordFrame(userId); // 借书记录

**this**.dispose();

// 添加借书记录界面关闭监听事件

}

**if** (source == returnMenuItem) {// 如果点击的开始还书菜单项

ReturnFrame returnFrame = **new** ReturnFrame(userId); // 开始还书

**this**.dispose();

// 添加开始还书界面关闭监听事件

}

**if** (source == returnRecordMenuItem) {// 如果点击的是还书记录菜单项

ReturnRecordFrame returnRecordFrame = **new** ReturnRecordFrame(userId);

// 还书记录

**this**.dispose();

// 添加还书记录界面关闭监听事件

}

**if**(source == exitloadMenuItem) {

LoginManager lm = **new** LoginManager();

**this**.dispose();

// 添加退出登录界面,关闭监听事件

}

}

## 3.4 录入功能的实现

1. 效果图，如图6所示，删除、修改、查询功能就不一一列举了，写法类似。



图6 “录入图书”功能运行效果

2. UML图，如图7所示

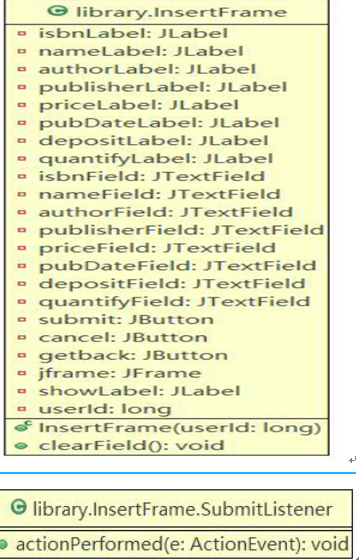


图7 InsertFrame类的UML图

  InsertFrame(long)是构造方法，负责完成录入图书界面的初始化，并为按钮submit和按钮cancel注册动作监听器。其中long类型的参数是从主界面窗口对象传递进来的操作标记，actionPerformed(ActionEvent)方法是ActionListener 接口定义的方法。当用户在界面上的文本框中输入了图书信息，单击“提交”(submit)按钮时，执行该方法。在该方法中，取得用户在文本框中输入的图书信息，调用Service类的addBook( Vector <String> )方法，完成在数据库中添加图书的操作。如接收addBook(Vector<String> )返回值等于0，则操作成功，否则操作失败。

3. 核心代码如下：

// 添加'提交'动作监听事件

**public** **class** SubmitListener **implements** ActionListener {

@Override

**public** **void** actionPerformed(ActionEvent e) {

// 获取界面中的录入文本内容

String isbn = isbnField.getText().trim();

String name = nameField.getText().trim();

String author = authorField.getText().trim();

String publisher = publisherField.getText().trim();

String price = priceField.getText().trim();

String pubDate = pubDateField.getText().trim();

String deposit = depositField.getText().trim();

String quantify = quantifyField.getText().trim();

**if** (isbn.equals("")) {

JOptionPane.*showMessageDialog*(jframe, "'书号'项为空，请您重新填写！");

} **else** **if**(name.equals("")) {

JOptionPane.*showMessageDialog*(jframe, "'书名'项为空，请您重新填写！");

} **else** **if**(author.equals("")) {

JOptionPane.*showMessageDialog*(jframe, "'作者;项为空，请您重新填写！");

} **else** **if**(publisher.equals("")) {

JOptionPane.*showMessageDialog*(jframe, "'出版社'项为空，请您重新填写！");

} **else** **if**(price.equals("")) {

JOptionPane.*showMessageDialog*(jframe, "'价格'项为空，请您重新填写！");

} **else** **if**(pubDate.equals("")) {

JOptionPane.*showMessageDialog*(jframe, "'出版时间'项为空，请您重新填写！");

} **else** **if**(deposit.equals("")) {

JOptionPane.*showMessageDialog*(jframe, "'存放位置'为空，请您重新填写！");

} **else** **if**(quantify.equals("")||quantify.trim().equals("0")) {

JOptionPane.*showMessageDialog*(jframe, "'数量'项为空(0)，请您重新填写！");

} **else** {//文本框内容都不为空时

Vector<String> bookInfo = **new** Vector<String>(); // Vector类可以实现可增长的对象数组。

// 将获得得各文本框的内容依次存放到bookInfo字符串数组中

bookInfo.add(isbn);

bookInfo.add(name);

bookInfo.add(author);

bookInfo.add(publisher);

bookInfo.add(price);

bookInfo.add(pubDate);

bookInfo.add(deposit);

bookInfo.add(quantify);

**if** (Service.*addBook*(bookInfo) == 0) {// 当录入信息成功存储到数据库中

clearField(); // 清空界面文本框内容

JOptionPane.*showMessageDialog*(jframe, "添加图书成功！");

} **else** { // 录入信息添加到数据库中失败

JOptionPane.*showMessageDialog*(jframe, "添加图书失败，请重新确认信息是否填写正确！");

}

}

}

}

// 添加'取消'按钮动作监听事件

**public** **class** CancelListener **implements** ActionListener {

@Override

**public** **void** actionPerformed(ActionEvent e) {

clearField(); // 当点击取消时，清空界面文本框内容

}

}

// 添加'返回'按钮动作监听事件

**public** **class** GetbackListener **implements** ActionListener {

@Override

**public** **void** actionPerformed(ActionEvent e) {

JFrame app = **new** BooksManager(userId); // 创建主界面

jframe.dispose(); // 关闭录入图书界面

}

}

## 3.5 借书功能的实现

1. 效果图，如图8所示，查询借阅历史、还书、查询还书历史功能就不再一一写了，写法类似



图8 “借书”功能运行效果

2. UML图，如图9所示。

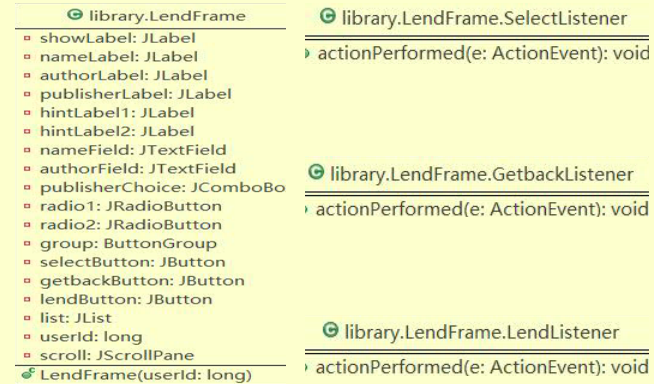


图9 LendFrame类的UML图

图9的UML图标识出LendFrame类中完成“借书”操作的主要成员。LendFrame窗体对象从主界面BooksManager对象接收到用户标识号参数后，初始化借书界面；当用户单击“查询”按钮后，actionPerformed(ActionEvent)方法获取用户在界面给出的条件，调用Service类的seek方法，查询出符合条件的图书信息，显示于列表中；当用户在列表中选择一本书，单击“借书”按钮后，actionPerformed( ActionEvent)方法获取用户在列表中选择的一本图书的信息，调用Service的lendBook方法，完成借书操作。

3. 核心代码如下：

**public** **class** LendListener **implements** ActionListener {

@Override

**public** **void** actionPerformed(ActionEvent e) {

String bookValue = (String) list.getSelectedValue();

**if** (bookValue == **null**) {

JOptionPane.*showMessageDialog*(**null**, "请先进行检索，然后再选一本图书！");

**return**;

}

Vector<String> result = **new** Vector<String>();

String str = bookValue;

**int** index = str.indexOf(':');

result.add(str.substring(2, index));

**int** index1 = 0;

str = str.substring(index + 1);

**for** (**int** i = 1; i <= 9; i++) {

index = str.indexOf(':');

index1 = str.indexOf(';');

result.add(str.substring(index + 1, index1).trim());

str = str.substring(index1 + 1);

}

String book[] = **new** String[10];

**for** (**int** i = 0; i < 10; i++) {

book[i] = result.elementAt(i).trim();

}

// 在馆数目

**int** remainder = Integer.*parseInt*(book[8]) - Integer.*parseInt*(book[9]);

//System.out.println(remainder);

**if** (remainder > 0) { // 还可以进行借此书

**int** confirm = JOptionPane.*showConfirmDialog*(**null**,

"您决定要借阅的一本书的信息如下: \n书号: " + book[1] + "\n书名: " + book[2] + "\n作者:" + book[3] + "\n出版社: "

+ book[4] + "\n价格: " + book[5] + "\n出版时间: " + book[6] + "\n存放位置: " + book[7]

+ "\n在馆数量: " + remainder + "\n" + "确定借阅吗?");

**if** (confirm > 0) {

JOptionPane.*showMessageDialog*(**null**, "您已取消了这次借书！");

**return**;

}

**int** id = Integer.*parseInt*(book[0]);

**int** lendQuantify = Integer.*parseInt*(book[9]) + 1;

// id:数的序号，userId:用户名（学号）lendQuantify：已借走此书的数量

**if** (Service.*lendBook*(id, userId, lendQuantify) == -1) {

JOptionPane.*showMessageDialog*(**null**, "此次操作失败，请重新操作！");

} **else** {

JOptionPane.*showMessageDialog*(**null**, "已成功借走此书！");

}

} **else** {

JOptionPane.*showMessageDialog*(**null**, "此书目前在馆数量为0，请重新选择！");

}

}

# **运行结果与调试**

## 4.1运行结果

（1）运行该系统，最先出来的界面为登录界面，若用户名或者密码与数据库中数据不匹配，则会出现如图10所示情况。



图10 登录失败效果图

（2）若要注册一个图书管理系统用户，则点击“注册”按钮，若注册失败，则会出现图11所示情况。



图11 注册失败效果图

（3）登录成功后，会进入到主界面，主界面里可以进行基本的功能操作，例如添加图书功能，若添加图书失败，则会出现图12所示情况。



图12 录入图书失败效果图

（4）选择“借阅图书”菜单栏中的“开始借书”菜单项进行借书操作，若选择借书，则会出现如图13所示情况。

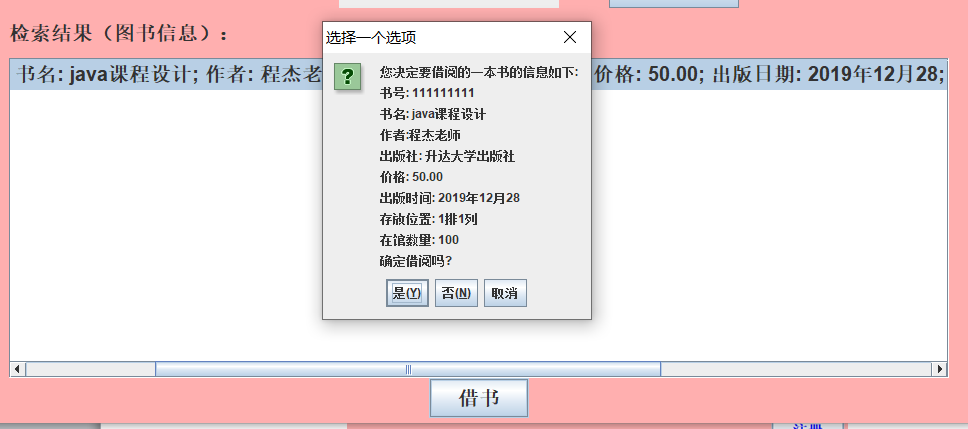


图13 选择借阅图书效果图

（5）选择“归还图书”菜单栏中的“开始还书”菜单项进行还书操作，若操作成功，会出现如图14所示情况。

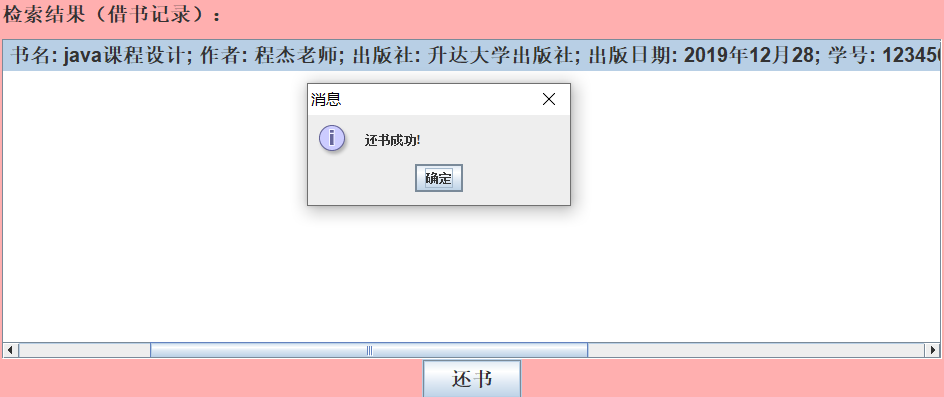


图14 还书成功效果图

## 4.2软件调试

在Eclipse环境下，创建一个新的工程，在工程中加载外部包mysql-connector-java-5.1.40-bin.jar（加载方法：将光标移至工程名，右击，在弹出的菜单中选择Bulid Path->Add ExternalcArchives，在弹出的打开文件对话框中选择mysql-connector-java-5.1.40-bin.jar文件即可）。在该工程中用来存放源代码的路径src的缺省包下，创建3.1系统的具体实现中讲到的14个类。然后运行主类LoginManager即可。其效果图为图16所示。

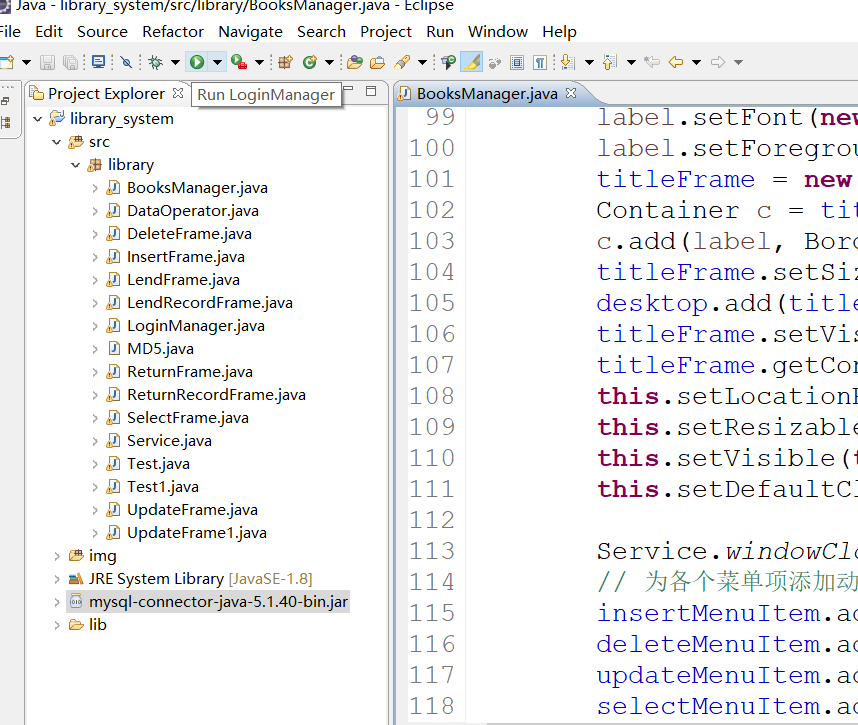


图16 系统调试效果图

# **结束语**

# **参考资料**

[1] 蔚哲明.《基于Java的综合课程设计》[M].清华大学出版社,2015年8月.

[2] 耿祥义，张跃平.《Java 2实用教程》[M].清华大学出版社,2018年8月.

[3] 王珊，萨师煊.《数据库系统概论》[M].高等教育出版社,2014年9月.