系统设计

功能分析

图书馆管理系统是管理图书的一个重要工具，一个功能完善的图书馆管理系统需要提供6个功能模块，分别是图书类别管理模块、图书信息管理模块、读者信息管理模块、新书订购管理模块、图书借阅管理模块、操作员信息管理模块。其中各功能模块的具体说明如下：

1. 图书类别管理模块：该模块主要负责管理图书馆的图书种类信息，如图书种类的名称、可借天数、罚款数目等信息。
2. 读者信息管理模块：该模块主要负责管理图书馆的读者信息，如读者编号、读者信息、证件号码、最大借书量等信息。
3. 新书订购管理模块：该模块主要负责管理图书馆的新书订购信息，包括新书订购和验收新书两个子模块。
4. 图书信息管理模块：该模块主要负责管理图书馆的图书信息，如图书编号、图书名称、作者、出版社等信息。
5. 图书借阅管理模块：该模块主要负责管理图书馆的书籍借阅和归还信息，包括图书借阅、图书归还、图书搜索3个子模块
6. 操作员信息管理模块：该模块主要负责管理图书馆的操作员信息维护。

系统设计

系统用例图

图书馆管理系统是一个内部人员使用的系统，也就是说不是所有人都能够使用它，只有图书馆的工作人员才能使用。如图1-1所示。

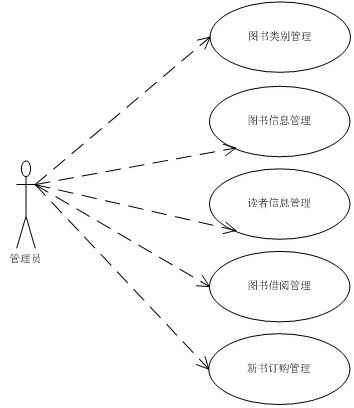


图1-1 管理员用例图

系统流程图

本系统首先需要对用户的身份进行识别，只有合法的用户才能进入系统;否则将无法进入系统。进入系统后，首先打开系统主窗体，在系统首页的菜单栏或者功能区可以选择各种导向来进行各种操作。如图1-2所示：



图1-2 系统流程图

数据库分析与设计

1.1数据库概要设计

本系统一共设计规划出6个实体，分别是图书类别信息实体、图书信息实体、读者信息实体、管理员实体、图书借阅信息实体以及图书订购信息实体。

图书类别有很多，例如有哲学、政治、法律、经济、自然科学、计算机等。因此可以建立一个图书类别信息表，专门用来保存图书的类别信息。同时因为每种类别的书籍阅读时间有所不同，所以需要在类别表中保存该类别可借天数。图书类别信息实体E-R图如图1-3所示。



图1-3 图书类别信息实体E-R图

对于图书馆来说最重要的就是要管理其下的书籍，所以需要建立一个图书信息表，用来保存图书的所有信息，图书信息实体E-R图如图1-4



图1-4 图书信息实体E-R图

要想在图书馆借书首先要进行登记并交会押金，领取读书卡后才能借书。所以需要建立一个读者信息表来保存图书馆的读者信息。读者信息实体E-R图如图1-5所示：



图1-5 读者信息实体E-R图

图书馆里一般有一个以上的系统操作员，需要建立一个操作员信息表，用来保存操作员的身份信息。操作员信息实体E-R图如图1-6所示：



图1-6 操作员信息实体E-R图

图书馆最大的功能就是能够借书，这时需要建立一个图书借阅信息表，用来保存读者的借书信息。图书借阅信息实体E-R图如图1-7所示：



图1-7 图书借阅信息实体E-R图

图书馆除了借书，还需要到出版社或其他代理商订购新书，这时就需要建立一个图订购信息表，用来保存所有的订购信息。订购信息实体E-R图如图1-8所示



图1-8 图书订购信息实体E-R图

1.2 数据库逻辑结构设计

根据设计好的各实体E-R图创建数据库的逻辑结构，数据库各表的结构如下。

（1）图书类别信息表用来储存所有的图书类别信息，包括图书类别编号、图书类别名称、可借天数以及迟还一天的罚款数目4个字段。该表的逻辑结构如表1-1所示：

表1-1

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **是否主键** | **描述** |
| id | 整数（int） | 是 | 图书类别编号 |
| typeName | 文本（varchar） | 否 | 图书类别名称 |
| days | 整数（int） | 否 | 可借天数 |
| fk | 浮点数 | 否 | 迟还一天的罚款数目 |

（2）图书信息表用来储存所有的图书信息，包括图书编号、类别编号、图书名称、作者、译者、出版社、出版日期、以及书籍价格8个字段.该表的逻辑结构如表1-2所示。

表1-2

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **是否主键** | **描述** |
| ISBN | 文本（varchar） | 是 | 图书编号 |
| typeId | 整数（int） | 否（外键） | 类别编号 |
| bookname | 文本（varchar） | 否 | 图书名称 |
| writer | 文本（varchar） | 否 | 作者 |
| translator | 文本（varchar） | 否 | 译者 |
| publisher | 文本（varchar） | 否 | 出版社 |
| date | 日期时间（datetime） | 否 | 出版日期 |
| price | 金钱（money） | 否 | 书籍价格 |

（3）读者信息表用来储存所有的读者信息，包括读者姓名、性别、年龄、证件号码、有效日期、最大借书量、电话号码、押金、证件类型、职业、读者编号以及办证日期12个字段。该表的逻辑结构如表1-3所示：

表1-3

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 是否主键 | 描述 |
| name | 文本（varchar） | 否 | 读者姓名 |
| set | 文本（varchar） | 否 | 读者性别 |
| age | 整数（int） | 否 | 读者年龄 |
| identityCard | 文本（varchar） | 否 | 证件号码 |
| date | 日期时间（datetime） | 否 | 会员证有效日期 |
| maxNum | 整数（int） | 否 | 最大借书量 |
| tel | 文本（varchar） | 否 | 电话号码 |
| keepMoney | 金钱（money） | 否 | 押金 |
| zj | 整数（int） | 否 | 证件类型 |
| zy | 文本（varchar） | 否 | 读者编号 |
| ISBN | 文本（varchar） | 是 | 读者编号 |
| bztime | 日期时间（datetime） | 否 | 办证日期 |

1. 操作员信息表用来保存操作员信息，该表的逻辑结构如表1-4所示：

表1-4

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **是否主键** | **描述** |
| id | 整数（int） | 是（自动递增） | 操作员编号 |
| name | 文本（varchar） | 否 | 用户名 |
| sex | 文本（varchar） | 否 | 性别 |
| age | 整数（int） | 否 | 年龄 |
| identityCard | 文本（varchar） | 否 | 证件号码 |
| workdate | 日期时间（datetime） | 否 | 工作时间 |
| tel | 文本（varchar） | 否 | 电话号码 |
| password | 文本（varchar） | 否 | 密码 |

（5）图书借阅信息表用来保存所有图书的借阅信息。该表的逻辑结构如表1-5所示：

表1-5

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **是否主键** | **描述** |
| id | 整数（int） | 是（自动递增） | 借阅编号 |
| bookISBN | 文本（varchar） | 否（外键） | 书籍编号 |
| operatorId | 整数（int） | 否（外键） | 操作员编号 |
| readerISBN | 文本（varchar） | 否（外键） | 读者编号 |
| isback | 整数（int） | 否 | 是否归还 |
| borrowDate | 日期时间（datetime） | 否 | 借书日期 |
| backdate | 日期时间（datetime） | 否 | 应还日期 |

（6）图书订购信息表用来保存图书馆的所有订购信息。该表的逻辑结构如表1-6所示：

表1-6

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **是否主键** | **描述** |
| ISBN | 文本（varchar） | 是 | 书籍编号 |
| date | 日期时间（datetime） | 否 | 订购日期 |
| number | 整数（int） | 否 | 订购数量 |
| operator | 文本（varchar） | 否 | 操作员 |
| checkAndAccept | 整数（int） | 否 | 是否验收 |
| zk | 浮点数 | 否 | 书籍折扣 |

1.2 数据库表之间的关系

各表的关系图，如图1-9：

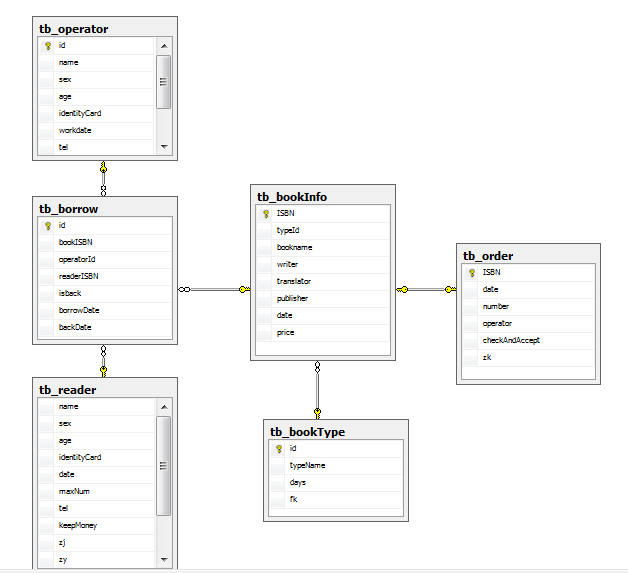


图1-9 表之间的关系图

系统演示

## 数据库访问操作

（1）变量定义代码：

**protected** **static** String *dbClassName* =

"com.microsoft.sqlserver.jdbc.SQLServerDriver"; //数据库连接驱动类

**protected** **static** String *dbUrl* = "jdbc:sqlserver://localhost:55766;"

+ "DatabaseName=librarysys;"; //数据库连接URL

**protected** **static** String *dbUser* = "sa"; //数据库用户名

**protected** **static** String *dbPwd* = "sa"; //数据库密码

**private** **static** Connection *conn* = **null**; //数据库连接对象

（2）数据库连接代码：

**try** {

**if** (*conn* == **null**) { //如果连接对象为空

Class.*forName*(*dbClassName*); //加载驱动类

*conn* = DriverManager.*getConnection*(*dbUrl*, *dbUser*, *dbPwd*); //获得连接对象

}

} **catch** (Exception ee) {

ee.printStackTrace();

}

（3）关闭连接代码：

**try** {

*conn*.close(); //关闭连接对象

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

*conn* = **null**; //设置连接对象为null值

}

（4）查询方法代码：

**private** **static** ResultSet executeQuery(String sql) { //查询方法

**try** {

**if**(*conn*==**null**) **new** Dao(); //如果连接对象为空，则重新调用构造方法

**return** *conn*.createStatement(ResultSet.*TYPE\_SCROLL\_SENSITIVE*,

ResultSet.*CONCUR\_UPDATABLE*).executeQuery(sql); //执行查询

} **catch** (SQLException e) {

e.printStackTrace();

**return** **null**; //返回null值

} **finally** {

}

}

（5）更新方法代码：

**private** **static** **int** executeUpdate(String sql) { //更新方法

**try** {

**if**(*conn*==**null**) **new** Dao(); //如果连接对象为空，则重新调用构造方法

**return** *conn*.createStatement().executeUpdate(sql); //执行更新

} **catch** (SQLException e) {

e.printStackTrace();

**return** -1;

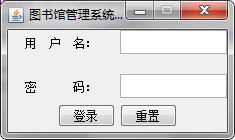
} **finally** {

}

}

## 管理员登录演示

（1）登录界面：



（2）核心代码：

**public** **static** Operater checkOperater(String name, String password) {

Operater operater=**new** Operater(); //操作员信息对象

String sql = "select \* from tb\_operator where name='" + name

+ "' and password='" + password + "'and admin=1"; //查询字符串

ResultSet rs = Dao.*executeQuery*(sql); //执行查询

**try** {

**while** (rs.next()) { //如果查询到了记录

operater.setId(rs.getString("id")); //设置操作员编号

operater.setName(rs.getString("name")); //设置操作员用户名

operater.setGrade(rs.getString("admin")); //设置操作员等级

operater.setPassword(rs.getString("password")); //设置管理员密码

}

} **catch** (Exception e) {

e.printStackTrace();

}

Dao.*close*(); //关闭连接对象

**return** operater; //返回操作员信息对象

}

## 三、系统主界面

（1）主界面：



主界面上有一个菜单栏，下方是一张图片。菜单栏上有读者信息管理、图书类别管理、图书信息管理、新书订购管理、借阅管理和修改密码六个子功能模块的入口。

（2）菜单栏创建代码：

**private** JMenuBar createMenu() { //创建菜单栏的方法

JMenuBar menuBar = **new** JMenuBar(); //创建工具栏

/\*读者信息管理\*/

JMenu readerManagerMItem = **new** JMenu("读者信息管理"); //新增读者信息管理子菜单

readerManagerMItem.add(MenuActions.*READER\_ADD*); //添加读者信息添加菜单项

readerManagerMItem.add(MenuActions.*READER\_MODIFY*); //添加读者信息添加菜单项

/\*图书类别信息管理\*/

JMenu bookTypeManageMItem = **new** JMenu("图书类别管理"); //新增图书类别管理子菜单

bookTypeManageMItem.add(MenuActions.*BOOKTYPE\_ADD*); //添加图书类型添加菜单项

bookTypeManageMItem.add(MenuActions.*BOOKTYPE\_MODIFY*); //添加图书类型修改菜单项

/\*图书信息管理\*/

JMenu menu = **new** JMenu("图书信息管理"); //新增图书信息管理子菜单

menu.add(MenuActions.*BOOK\_ADD*); //添加图书信息添加菜单项

menu.add(MenuActions.*BOOK\_MODIFY*); //添加图书信息修改菜单项

/\*新书订购管理\*/

JMenu bookOrderMenu = **new** JMenu("新书订购管理"); //初始化新书订购管理菜单

bookOrderMenu.add(MenuActions.*NEWBOOK\_ORDER*); //添加新书定购菜单项

bookOrderMenu.add(MenuActions.*NEWBOOK\_CHECK\_ACCEPT*); //添加验收新书菜单项

/\*借阅管理\*/

JMenu borrowManageMenu = **new** JMenu("借阅管理"); //借阅管理

borrowManageMenu.add(MenuActions.*BORROW*); //添加借阅菜单项

borrowManageMenu.add(MenuActions.*GIVE\_BACK*); //添加归还菜单项

borrowManageMenu.add(MenuActions.*BOOK\_SEARCH*); //添加搜索菜单项

/\*修改密码\*/

JMenu sysManageMenu = **new** JMenu("修改密码"); //操作员管理

sysManageMenu.add(MenuActions.*MODIFY\_PASSWORD*); //添加更改菜单项

/\*在工具栏上添加菜单\*/

menuBar.add(readerManagerMItem); //添加读者信息菜单到菜单栏

menuBar.add(bookTypeManageMItem); //添加基础数据维护菜单到菜单栏

menuBar.add(menu); //添加基础数据维护菜单到菜单栏

menuBar.add(bookOrderMenu); //添加新书订购管理菜单到菜单栏

menuBar.add(borrowManageMenu); //添加借阅管理菜单到菜单栏

menuBar.add(sysManageMenu); //添加系统维护菜单到菜单栏

**return** menuBar;

}

## 四、读者信息管理功能演示

a）、 （1）读者信息添加界面



输入需要添加的读者的基本信息，并检查各文本框内的数据是否合法，然后插入到数据库tb\_reader表当中。

（2）核心代码：

**class** ButtonAddListener **implements** ActionListener {

**private** **final** JRadioButton button1;

ButtonAddListener(JRadioButton button1) {

**this**.button1 = button1;

}

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

**if**(readername.getText().length()==0){

JOptionPane.*showMessageDialog*(**null**, "读者姓名文本框不可为空");

**return**;

}

**if**(age.getText().length()==0){

JOptionPane.*showMessageDialog*(**null**, "读者年龄文本框不可为空");

**return**;

}

**if**(zjnumber.getText().length()==0){

JOptionPane.*showMessageDialog*(**null**, "证件号码文本框不可为空");

**return**;

}

**if**(zjnumber.getText().length()!=13){

JOptionPane.*showMessageDialog*(**null**, "证件号码位数为13");

**return**;

}

**if**(keepmoney.getText().length()==0){

JOptionPane.*showMessageDialog*(**null**, "押金文本框不可为空");

**return**;

}

**if**(zy.getText().length()==0){

JOptionPane.*showMessageDialog*(**null**, "职业文本框不可为空");

**return**;

}

**if**(zy.getText().length()>20){

JOptionPane.*showMessageDialog*(**null**, "职业文本框位数为20");

**return**;

}

**if**(ISBN.getText().length()==0){

JOptionPane.*showMessageDialog*(**null**, "读者条形码文本框不可为空");

**return**;

}

**if**(ISBN.getText().length()!=13){

JOptionPane.*showMessageDialog*(**null**, "读者条形码文本框为13位");

**return**;

}

**if**(tel.getText().length()==0){

JOptionPane.*showMessageDialog*(**null**, "电话号码文本框不可为空");

**return**;

}

**if**(tel.getText().length()>11||tel.getText().length()<0){

JOptionPane.*showMessageDialog*(**null**, "电话号码位数小于11位");

**return**;

}

**if**(maxnumber.getText().length()==0){

JOptionPane.*showMessageDialog*(**null**, "最大借书量文本框不可为空");

**return**;

}

**if**(maxnumber.getText().length()>2||tel.getText().length()<0){

JOptionPane.*showMessageDialog*(**null**, "最大借书量为两位数字");

**return**;

}

**if**(bztime.getText().isEmpty()||date.getText().isEmpty()){

JOptionPane.*showMessageDialog*(**null**, "时间格式请使用\"2007-05-10\"格式");

**return**;

}

String sex="1";

**if**(!button1.isSelected()){

sex="2";}

String zj=String.*valueOf*(comboBox.getSelectedIndex());

**int** i=Dao.*InsertReader*(readername.getText().trim(), sex.trim(), age.getText().trim(),zjnumber.getText().trim(), Date.*valueOf*(date.getText().trim()), maxnumber.getText().trim(),tel.getText().trim(), Double.*valueOf*(keepmoney.getText().trim()),zj,zy.getText().trim(),Date.*valueOf*(bztime.getText().trim()),ISBN.getText().trim());

System.*out*.println(i);

**if**(i==1){

JOptionPane.*showMessageDialog*(**null**, "添加成功！");

doDefaultCloseAction();

}

}

}

b）、（1）读者信息修改界面



读者信息修改界面可对读者信息进行修改或者删除读者信息。点击表格时读者信息会在相应的文本框写入相应的信息

（2）核心代码：

表格点击时事件代码：

**class** TableListener **extends** MouseAdapter {

**public** **void** mouseClicked(**final** MouseEvent e) {

**int** selRow = table.getSelectedRow();

readername.setText(table.getValueAt(selRow, 0).toString().trim());

**if**(table.getValueAt(selRow, 1).toString().trim().equals("男"))

JRadioButton1.setSelected(**true**);

**else**

JRadioButton2.setSelected(**true**);

age.setText(table.getValueAt(selRow, 2).toString().trim());

zjnumber.setText(table.getValueAt(selRow, 3).toString().trim());

date.setText(table.getValueAt(selRow, 4).toString().trim());

maxnumber.setText(table.getValueAt(selRow, 5).toString().trim());

tel.setText(table.getValueAt(selRow, 6).toString().trim());

keepmoney.setText(table.getValueAt(selRow, 7).toString().trim());

comboBox.setSelectedItem(table.getValueAt(selRow, 8).toString().trim());

zy.setText(table.getValueAt(selRow, 9).toString().trim());

ISBN.setText(table.getValueAt(selRow, 10).toString().trim());

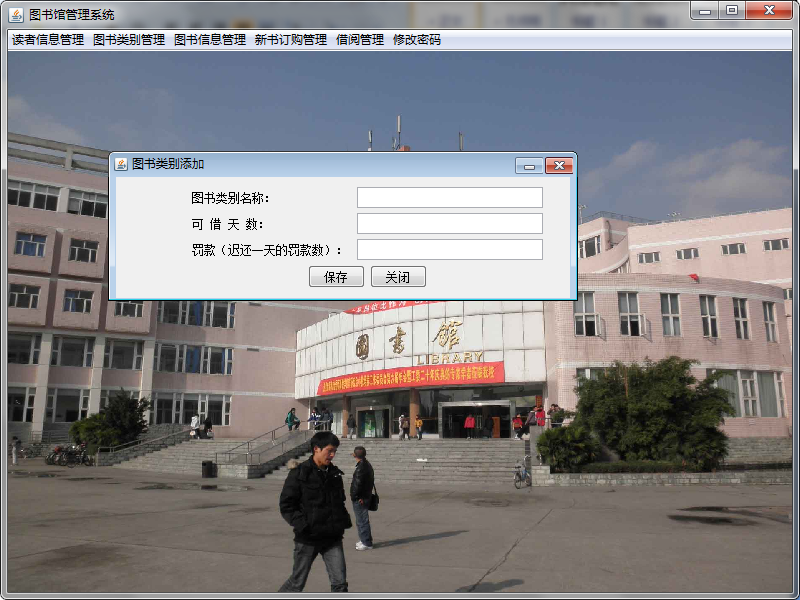
bztime.setText(table.getValueAt(selRow, 11).toString().trim());

}

}

## 五、图书类别管理功能

a）、（1）图书类别添加功能界面：



在相应的文本框输入信息，点击保存，保存信息到数据库tb\_bookType表中。

（2）保存信息的代码：

**public** **static** **int** InsertBookType(String bookTypeName,String days,Double fk){

**int** i=0;

**try**{

String sql="insert into tb\_bookType(typeName,days,fk) values('"+bookTypeName+"','"+days+"',"+fk+")";

i=Dao.*executeUpdate*(sql);

}**catch**(Exception e){

e.printStackTrace();

}

**return** i;

}

b)、（1）图书类别信息修改界面：



点击表格时，表中的内容会读入相应的文本框。点击修改时修改的内容被更新到数据库相应的表中。

（2）修改代码：

**public** **static** **int** UpdatebookType(String id,String typeName,String days,String fk){

**int** i=0;

**try**{

String sql="update tb\_bookType set typeName='"+typeName+"',days='"+days+"',fk='"+fk+"' where id='"+id+"'";

i=Dao.*executeUpdate*(sql);

}**catch**(Exception e){

e.printStackTrace();

}

Dao.*close*();

**return** i;

}

## 六、图书信息管理功能：

a）、（1）图书信息添加界面：



在文本框中输入相应的信息，点击添加，将信息保存到数据库相应的表中。

（2）添加代码：

**public** **static** **int** Insertbook(String ISBN,String typeId,String bookname,

String writer,String translator,String publisher,Date date,Double price){

**int** i=0;

**try**{

String sql="insert into tb\_bookInfo(ISBN,typeId,bookname,writer,translator," +

"publisher,date,price) values('"+ISBN+"','"+typeId+"','"+bookname+"'," +

"'"+writer+"','"+translator+"','"+publisher+"','"+date+"',"+price+")";

i=Dao.*executeUpdate*(sql);

}**catch**(Exception e){

System.*out*.println(e.getMessage());

}

Dao.*close*(); //关闭连接对象

**return** i; //返回更新记录数

}

b）、（1）图书信息修改界面：



点击表时，相应数据读入文本框中，点击修改按钮，将修改信息写到数据库表中。

（2）修改代码：

**public** **static** **int** Updatebook(String ISBN,String typeId,String bookname,

String writer,String translator,String publisher,Date date,Double price){

**int** i=0;//更新记录数

**try**{

String sql="update tb\_bookInfo set ISBN='"+ISBN+"',typeId='"+typeId+"',bookname='"

+bookname+"',writer='"+writer+"',translator='"+translator+"',publisher='"

+publisher+"',date='"+date+"',price="+price+" where ISBN='"+ISBN+"'";

i=Dao.*executeUpdate*(sql);//执行更新

}**catch**(Exception e){

e.printStackTrace();

}

Dao.*close*();//关闭连接对象

**return** i;//返回更新记录数

}

## 七、新书订购管理功能

a）、（1）新书订购界面：



添加新书订购信息，如果已验收，表明新书已收到。

（2）代码：

**public** **static** **int** InsertBookOrder(String ISBN,Date date,String number,String operator,String checkAndAccept,Double zk){

**int** i=0;

**try**{

String sql="insert into tb\_order(ISBN,date,number,operator,checkAndAccept,zk) values('"+ISBN+"','"+date+"','"+number+"','"+operator+"',"+checkAndAccept+",'"+zk+"')";

i=Dao.*executeUpdate*(sql);

}**catch**(Exception e){

e.printStackTrace();

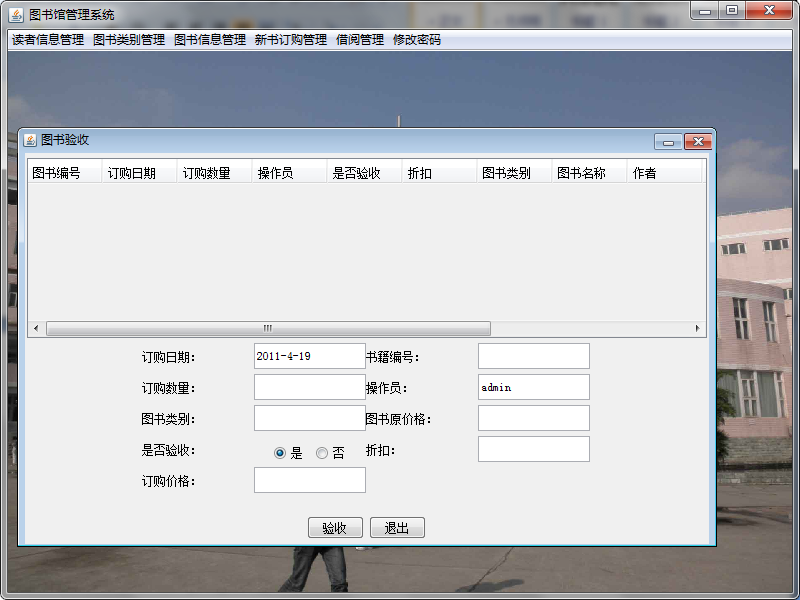
}

Dao.*close*();

**return** i;

}

b）、（1）新书验收界面：



对已到的新书进行验收操作

（2）代码：

**public** **static** **int** UpdateCheckBookOrder(String ISBN){

**int** i=0;

**try**{

String sql="update tb\_order set checkAndAccept=0 where ISBN='"+ISBN+"'";

i=Dao.*executeUpdate*(sql);

}**catch**(Exception e){

e.printStackTrace();

}

Dao.*close*();

**return** i;

}

## 八、书籍借阅

a）、（1）图书借阅界面：



在左边框中输入需要借书的读者编号，并且在右边输入要借书籍的编号，回车，会在下面表格中显示当前可借书籍信息，点击“借出当前图书”按钮。

（2）、得到借书信息的代码：

**class** bookISBNListenerlostFocus **extends** KeyAdapter {

**public** **void** keyTyped(KeyEvent e) {

**if** (e.getKeyChar() == '\n') { // 判断在文本框是否输入回车。

**if** (readerISBN.getText().trim().length()!=0

&& bookISBN.getText().trim().length()!=0) {//判断书籍编号和读者编号是否为空

String ISBNs = bookISBN.getText().trim();

List list = Dao.*selectBookInfo*(ISBNs);

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

BookInfo book = (BookInfo) list.get(i);

bookName.setText(book.getBookname());

bookType.setText(String.*valueOf*(map.get(book

.getTypeid())));

price.setText(String.*valueOf*(book.getPrice()));

}

String days = "0";

List list2 = Dao.*selectBookCategory*(bookType.getText()

.trim());

**for** (**int** j = 0; j < list2.size(); j++) {

BookType type = (BookType) list2.get(j);

days = type.getDays();

}

String readerISBNs = readerISBN.getText().trim();

List list5 = Dao.*selectReader*(readerISBNs); // 此读者是否在tb\_reader表中

List list4 = Dao.*selectBookInfo*(ISBNs); // 此书是否在tb\_bookInfo表中

**if** (!readerISBNs.isEmpty() && list5.isEmpty()) {

JOptionPane.*showMessageDialog*(**null**,

"此读者编号没有注册，查询输入读者编号是否有误！");

**return**;

}

**if** (list4.isEmpty() && !ISBNs.isEmpty()) {

JOptionPane.*showMessageDialog*(**null**,

"本图书馆没有此书，查询输入图书编号是否有误！");

**return**;

}

**for** (**int** i = 0; i < list5.size(); i++) {

Reader reader = (Reader) list5.get(i);

readerName.setText(reader.getName());

number.setText(reader.getMaxNum());

keepMoney.setText(reader.getKeepMoney() + "");

}

**if** (Integer.*parseInt*(number.getText().trim()) <= 0) {

JOptionPane.*showMessageDialog*(**null**, "借书量已经超过最大借书量！");

**return**;

}

add();

number.setText(String.*valueOf*(Integer.*parseInt*(number

.getText().trim()) - 1));

}

**else**

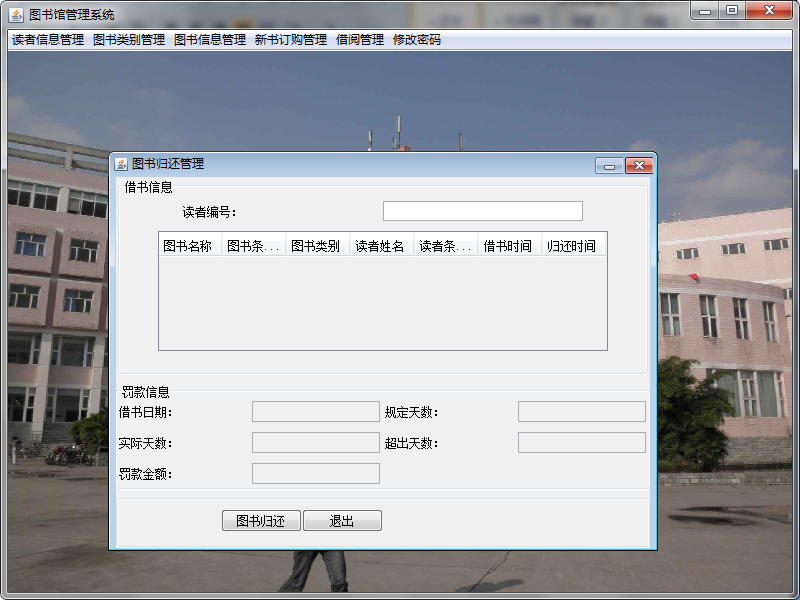
JOptionPane.*showMessageDialog*(**null**, "请输入读者条形码！或书籍条形码");

}

}

}

B）、（1）图书归还界面：



输入读者编号，则在表格中出该读者的借书信息，点击要还的图书会显示一些相关信息，比如是否超过可借天数等。

（2）图书归还代码：

**class** BookBackActionListener **implements** ActionListener{

**private** **final** DefaultTableModel model;

BookBackActionListener(DefaultTableModel model) {

**this**.model = model;

}

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

**if**(readerISBNs==**null**){

JOptionPane.*showMessageDialog*(**null**, "请输入读者编号！");

**return**;

}

System.*out*.println(bookISBNs==**null**);

**if**(table.getSelectedRow()==-1){

JOptionPane.*showMessageDialog*(**null**, "请选择所要归还的图书！");

**return**;

}

**int** i=Dao.*UpdateBookBack*(bookISBNs, readerISBNs,id);

System.*out*.print(i);

**if**(i==1){

**int** selectedRow = table.getSelectedRow();

model.removeRow(selectedRow);

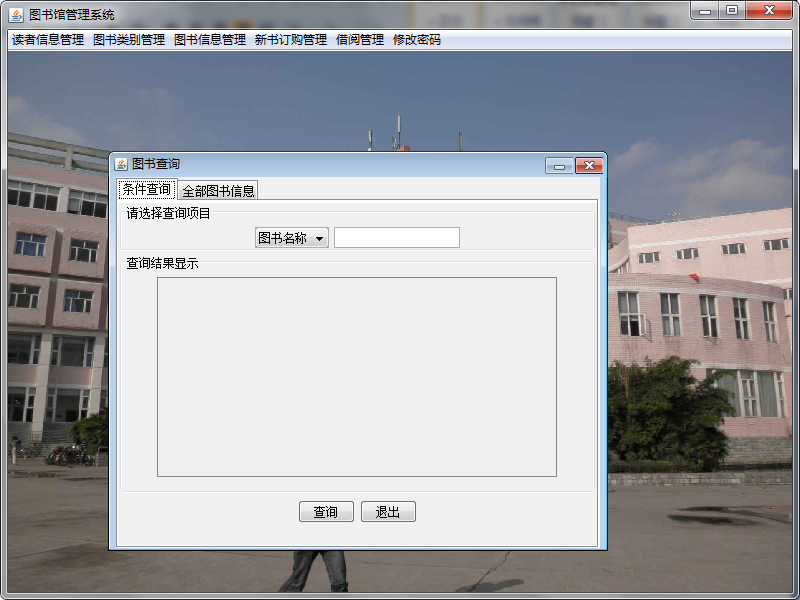
JOptionPane.*showMessageDialog*(**null**, "还书操作完成！");

}

}

}

c)、（1）图书搜索界面：



可按要求查询图书信息。

（2）查询代码：

**public** **static** List selectbooksearch() {

List list=**new** ArrayList();

String sql = "select \* from tb\_bookInfo";

ResultSet s = Dao.*executeQuery*(sql);

**try** {

**while** (s.next()) {

BookInfo bookinfo=**new** BookInfo();

bookinfo.setISBN(s.getString(1));

bookinfo.setTypeid(s.getString(2));

bookinfo.setBookname(s.getString(3));

bookinfo.setWriter(s.getString(4));

bookinfo.setTranslator(s.getString(5));

bookinfo.setPublisher(s.getString(6));

bookinfo.setDate(s.getDate(7));

bookinfo.setPrice(s.getDouble(8));

list.add(bookinfo);

}

} **catch** (Exception e) {

e.printStackTrace();

}

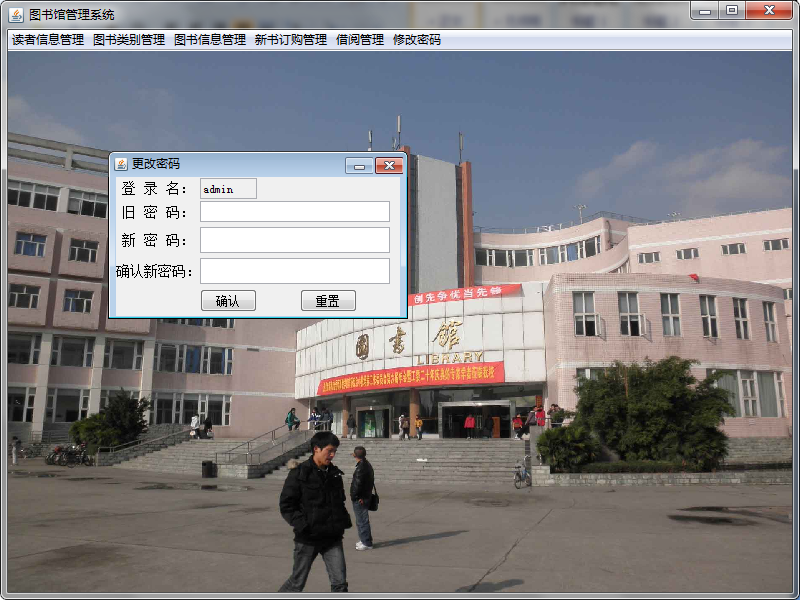
Dao.*close*();

**return** list;

}

## 九、管理员更改密码

（1）更改密码界面：



管理员输入相应信息可更改自己的密码。

（2）修改密码代码：

**public** **static** **int** Updatepass(String password,String name){

**int** i=0;

**try**{

String sql="update tb\_operator set password='"+password+"' where name='"+name+"'";

i=Dao.*executeUpdate*(sql);

}**catch**(Exception e){

e.printStackTrace();

}

Dao.*close*();

**return** i;

}