浙江大学实验报告

课程名称：数据库系统原理 实验类型：JDBC 编程

实验项目名称：图书管理系统

学生姓名 1：王宇晗 专业 1：混合班 学号 1：3170106051

学生姓名 2：漆翔宇 专业 2：计算机科学与技术 学号 2：3170104557

指导老师：陈岭

实验地点：紫金港机房/寝室 等 实验日期：2019 年 4 月 28 日

一、 实验目的和要求

1. 实验目的

设计并实现一个精简的图书管理系统，要求具有图书入库、查询、借书、还书、借书证管理等功能。通过该图书馆系统的设计与实现，提高学生的系统编程能力，加深对数据库系统原理及应用的理解。

2. 实验要求

(1) 管理基本数据对象

(2) 实现基本功能模块

(3) 实现用户界面

完成一个交互式应用界面，可采用图形界面或字符界面。

(4) 数据库平台

SQL Server 或 MySQL

其中 MySql 详细信息请参见 http://www.mysql.com

MySql APIs:

1. MySql ODBC 3.51
2. MySql JDBC 5.0
3. MySql PHP APIs

有关 MySql 的安装，请参看有关参考书。

(5) 开发工具

任选（如 VC++, PHP, Java, Delphi, PowerBuilder 等）

二、 实验内容和原理

1. 数据对象管理

对象名称	包含属性
书	书号，类别，书名，出版社，年份，作者，价格，总藏书量，库存
借书证	卡号，姓名，单位，类别 (教师 学生等)
管理员	管理员 ID，密码，姓名，联系方式
借书记录	卡号，借书证号 ,借期，还期，经手人 (管理员 ID)

四个数据对象分别要在数据库端和应用程序端控制。在数据库端可以分别以 4 张表的形式存在，在应用程序端以四个类的形式组织。通过 JDBC 接口，可以让数据在前后端的两种数据结构间互相传输。

2. 登录系统与用户管理

(1) 登录系统

将帐号组织为管理员和普通借书证用户，把所有帐号信息存储在数据库端中。验证登录权限的时候，从数据库端中扒取账户信息，通过与前端输入的账户信息对比控制登录状态。

(2) 用户管理

在客户端中实现普通的借书证用户的添加和删除控制以及借书证用户信息查询功能。可以通过 JDBC 实现对数据库中用户表的一些基本查询和操作来完成。

3. 图书入库/图书查询/借书/还书

模块名称	功能描述
图书入库	1. 单本入库 2. 批量入库 (方便最后测试) 图书信息存放在文件中，每条图书信息为一行. 一行中的内容如下 (书号，类别，书名，出版社，年份，作者，价格，数量) Note: 其中 年份、数量是整数类型；价格是两位小数类型；其余为字符串类型 Sample: (book_no_1, Computer Science, Computer Architecture, xxx, 2004, xxx, 90.00, 2)
图书查询	要求可以对书的 类别，书名，出版社，年份(年份

	区间), 作者, 价格(区间) 进行查询. 每条图书信息包括以下内容: (书号, 类别, 书名, 出版社, 年份, 作者, 价格, 总藏书量, 库存) 可选要求: 可以按用户指定属性对图书信息进行排序. (默认是书名)
借书	1.输入借书证卡号 显示该借书证所有已借书籍 (返回, 格式同查询模块) 2.输入书号 如果该书还有库存，则借书成功，同时库存数减一。 否则输出该书无库存，且输出最近归还的时间。
还书	1.输入借书证卡号 显示该借书证所有已借书籍 (返回, 格式同查询模块) 2.输入书号 如果该书在已借书籍列表内, 则还书成功, 同时库存加一. 否则输出出错信息.

4. 用户界面

通过常用的流行 GUI 设计手段，构建一个图形化的交互界面。应用程序处理前端的交互逻辑并通过 JDBC 访问数据库，最后将结果反馈给前端界面，完成交互。

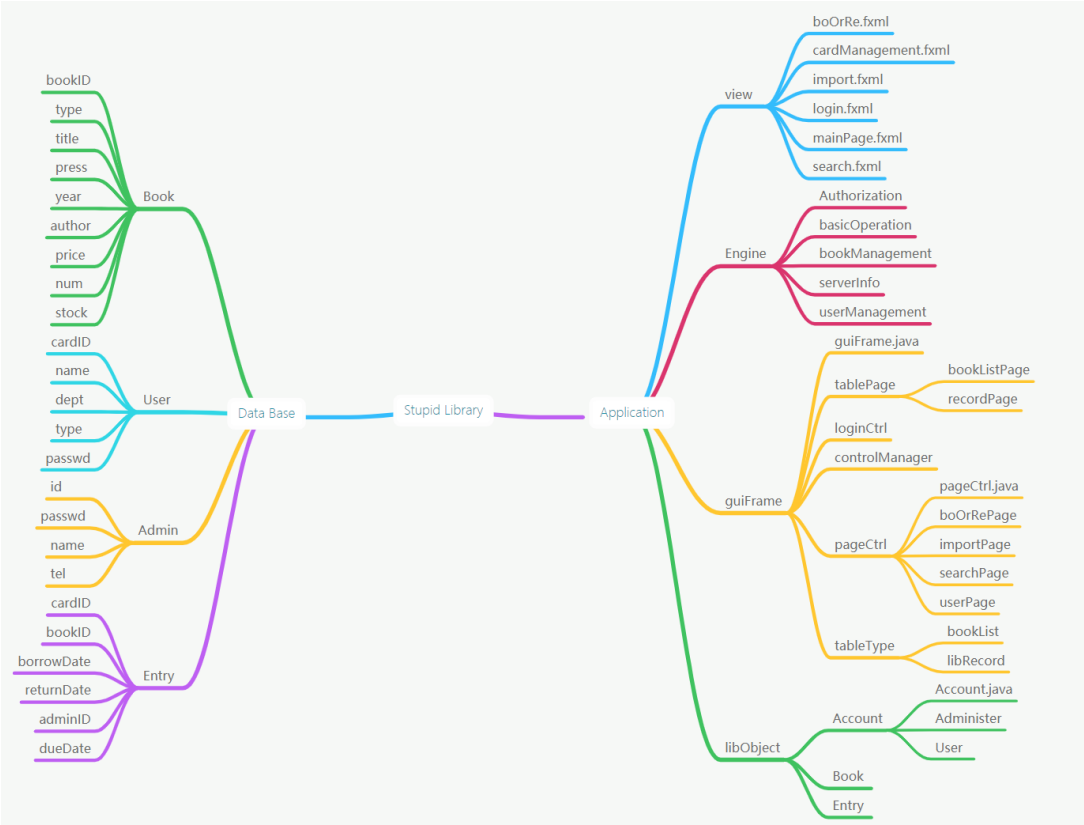
三、 主要仪器设备

MySQL 数据库系统
Java8 SE
JavaFX + JavaFX Scene Builder
MySQL-JDBC-8.0.15
Mac OS / Windows 10

四、 操作方法与实验步骤

1. 项目架构

(1)总览



如上所示，整个图书管理系统由数据库引擎后端与应用程序前端两部分组成。所有数据表都由 Oracle MySQL 数据库引擎托管，由 Java 构建的应用程序端可以通过 JDBC 向数据库引擎发送各种 SQL 命令并从数据库系统中获得各种类型的返回数据。最后用户再通过 GUI 界面和应用程序端交互完成各种图书管理功能。

(2)数据库端

数据库端采用了 Orcale MySQL 系统。由于整个图书管理系统只要求对四类数据对象进行管理。所以整个后端的数据库系统中只有四张表。

数据表定义如下：

```
create table admin
(
    id varchar(45),
    passwd varchar(45),
    name varchar(45),
    tel varchar(45),
    primary key(id)
)
```

```
create table Book
(
	bookID varchar(45),
	type varchar(45),
	title varchar(45),
	press varchar(45),
	year int(11),
	author varchar(45),
	price decimal(18,2),
	num int(11),
	stock int(11),
	primary key(bookID)
)

create table entry
(
	cardID varchar(45),
	bookID varchar(45),
	borrowDate varchar(45),
	returnDate varchar(45),
	adminID varchar(45),
	dueDate varchar(45),
	primary key(cardID,bookID,borrowDate)
)

create table user
(
	cardID varchar(45),
	name varchar(45),
	dept varchar(45),
	type varchar(45),
	primary key(cardID)
)
```

(3)应用程序端

整个应用程序端可以分为四个主要的大模块。分别是 guiFrame，view，libObject，Engine。

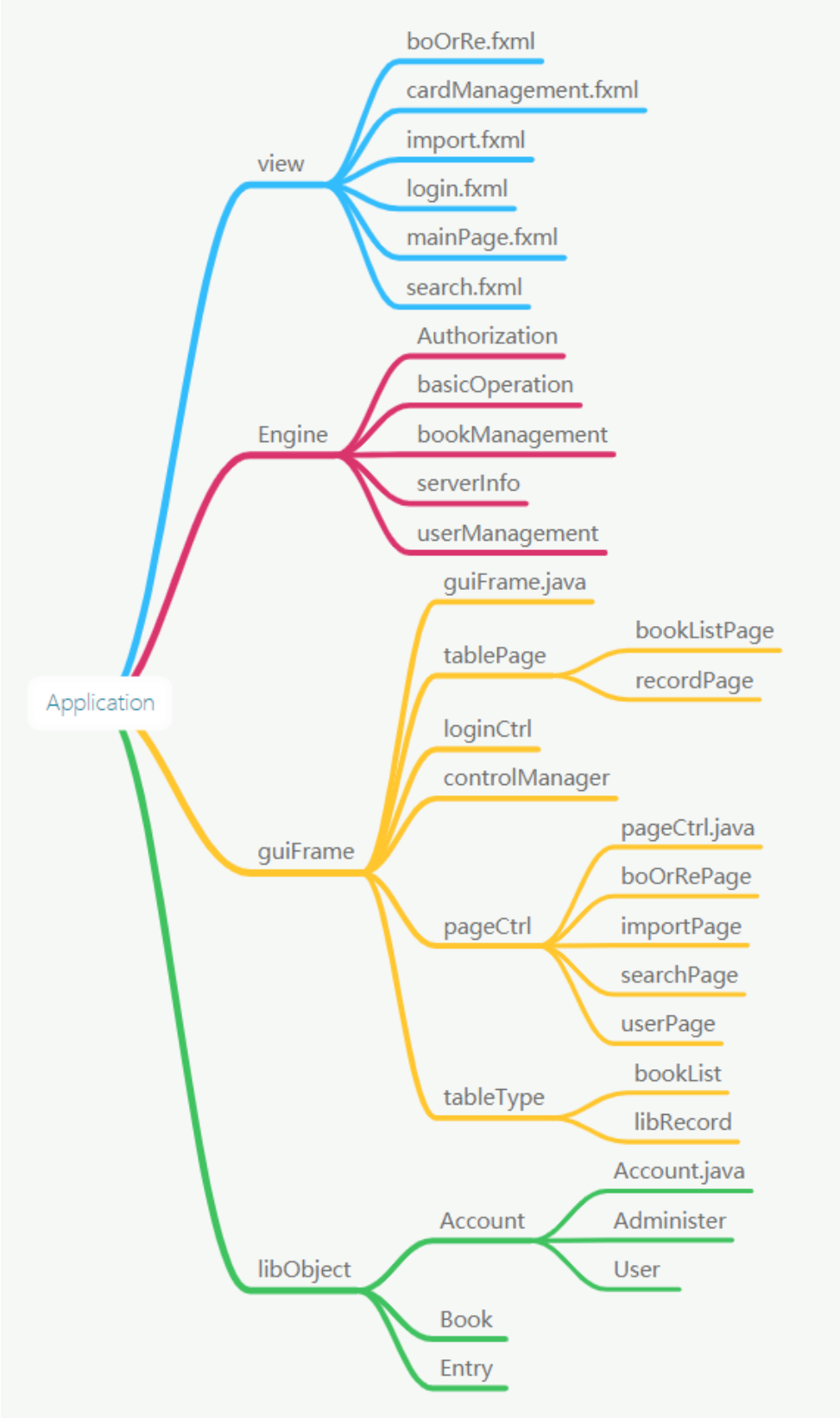
其中 view 主要是一些 GUI 的素材资源和前端布局的 fxml 文件，主要完成了 GUI 界面控件命名与布局，事件绑定等工作。整个应用程序主要做了 6 个静态的图形页面，分别用来管理登陆，主页欢迎页面，借还书，导入书籍，用户管理和书籍搜索。这部分内容都在 JavaFX Scene builder 中通过可视化编辑的手段完成了。

libObject 中定义了四个数据对象 Book, Entry, Administer, User。它们以类的形式组织起来，作为图书管理系统中四个基本对象在应用端的数据结构。

guiFrame 主要是前端各个页面交互事件的控制器。loginCtrl 用来控制登陆界面的逻辑，pageCtrl 用来控制各个功能界面的事务。controlManager 负责界面切换，权限控制逻辑。tableType 和 tablePage 主要用于将查询的结果表格进行呈现。

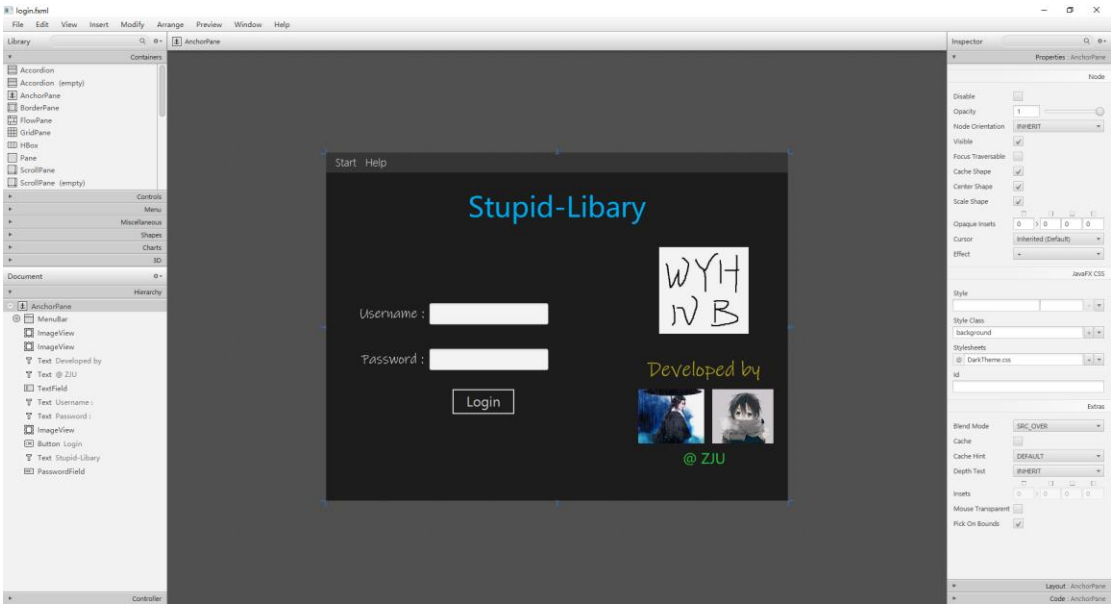
Engine 模块是整个图书管理系统应用层与数据库系统层的信息交流核心。basicOperation 中封装了基于 JDBC 的基本 SQL 命令操作。serverInfo 封装了本地数据库服务器的信息。Authorization 主要用来做权限控制，权限管理。userManagement 和 bookManagement 分别负责用户与书籍的数据管理和交互。

应用程序端的基本架构逻辑图如下：

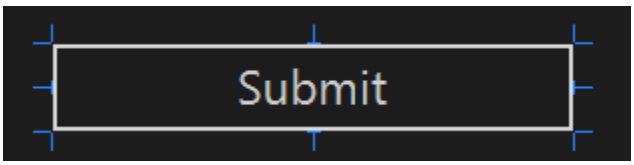


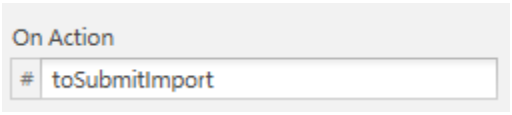
2. GUI 设计

本项目的 GUI 设计完全基于 JavaFX 技术。



前端的布局和风格可以通过在 JavaFX Scene Builder 中可视化编辑，并加载流行的 css 定义表控制。上图是加载了 VScode 界面样式 css 表的登陆界面。






```
fx:controller="guiFrame.pageCtrl.importPage.importPage"
```

```
42  public void toSubmitImport()
43  {
44      String message;
45      try
46      {···
70      {···
77      return;
78  }
```

如上所示，我们在 Scene Builder 中编辑了前端图形后，给每一个需要事件响应的控件再绑定对应的监听器/控件 ID 等属性，自动生成.fxml 文件。然后在.fxml 文件中再绑定具体的逻辑控制模块。最后在逻辑控制模块中用代码实现事件响应。这也就是 JavaFX 的基本交互模式。

显然，用户的所有请求都是通过 GUI 界面提供的按钮，输入框等控件送到后台的。如上所述，这些请求会最先送到 guiFrame 模块的各个 controller 中，对于涉及到修改和查询数据库信息的请求，controller 会调用 Engine 模块中实现的接口，Engine 模块再通过 JDBC 和数据库交互。最后从数据库返回的信息又会经由 Engine 传回 controller。一些需要反馈给用户的

信息，则通过前端的弹窗对话框或者一个弹出的表格界面再呈现给用户。

3. Engine 设计

(0.)服务器信息与服务器连接

我们将服务器信息存储在 serverInfo 模块，同时提供接口可以从外部文件中导入服务器信息。

```
public static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
public static   final String DB_URL="jdbc:mysql://localhost/StupidLibrary";///StupidLibrary";
public static   String USER = "root";
public static   String PASS = "123456";
private static String dbUseUnicode = "TRUE";
private static String dbEncoding = "UTF8";
private static String dbUseSSL = "TRUE";

public static void init(String filePath)
{
    try
    {
        BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(filePath),"UTF-8"));
        USER = in.readLine();
        PASS =in.readLine();
        System.out.println("User : " + USER);
        System.out.println("Pwd : "+PASS);
    }catch(Exception e)
    {
        System.out.println("ERROR : Failed to get DB account");
        System.exit(0);
    }
}
```

在应用进行中，为了获取与服务器的连接，每次都需要提供至少包含服务器 URL、登陆用户名、登陆密码的信息，我们将该过程封装，提供了一个获取 URL 的接口。

```
public static String getUrl()
{
        return DB_URL+
        "?user=" + USER +
        "&password=" + PASS +
        "&useUnicode=" + dbUseUnicode +
        "&characterEncoding=" + dbEncoding +
        "&useSSL=" + dbUseSSL+"&serverTimezone=GMT%2B8";
}
```

(1.)用户与认证模块

a) 启动应用时准备用户信息

为了方便用户认证和本地进行完整性约束，我们每次都在本地同步用户和管理员的信息。因此在每次启动应用时，都要先备份好用户信息。

准备用户信息分为两步，第一步是获取数据库的 root 连接后，从 user 和 admin 表拉取所有的用户信息存入一个 Account 列表中，Account 包含两个子类 User 和 Administer。核心代码如下。

```
private static void dragFromBackend() {
try {
        stmt = conn.createStatement();
        String sql;
        sql = "select * from user";
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()) {
            String cardID = rs.getString("cardID");
            String name = rs.getString("name");
            String dept = rs.getString("dept");
            String type = rs.getString("type");
            String passwd = rs.getString("passwd");
            if (existID(cardID) == true) continue;
            //public User(String id,String pwd,String name,String depart,String type)
            User tmp = new User(cardID, passwd, name, dept, type);
            account.add(tmp);
        }
        stmt.close();
    } catch (SQLException se) {
        se.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }

    try {
        stmt = conn.createStatement();
        String sql;
        sql = "select * from admin";
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()) {
            String id = rs.getString("id");
            String passwd = rs.getString("passwd");
            String name = rs.getString("name");
            String tel = rs.getString("tel");
            if (existID(id) == true) continue;
```

```
                Administer tmp = new Administer(id, passwd, name, tel);
                account.add(tmp);
            }
            stmt.close();
        } catch (SQLException se) {
            se.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
         }
}
```

第二步是从本地的默认用户列表中导入用户信息。这一步中导入的用户是必须存在的用户，因此有可能会与数据库中已存在的用户产生重复，我们通过判断避免了重复添加。我们将这一步抽象为从外部文件导入用户信息，这样在启动时和进行用户管理时都可以使用该接口。

```
public static boolean existID(String ID) {
    for(Account t:account) {
        if (t.getId().equals(ID)) return true;
    }
    return false;
}
public static void importFromList(String filePath)
{
    try
    {
        BufferedReader in = new BufferedReader(new InputStreamReader(new
FileInputStream(filePath),"UTF-8"));
        String temp=in.readLine();
        String buf[];
        Account tempAccount;
        while(temp!=null)
        {
            System.out.println(temp);
            buf=temp.split(",");
            if(buf[0].equals("A"))
            {
                tempAccount=new Administer(buf[1], buf[2], buf[3], buf[4]);
            }
            else
            {
                tempAccount=new User(buf[1], buf[2], buf[3], buf[4],buf[5],buf[6]);
            }
            if (existID(tempAccount.getId()) == false) {
                if (tempAccount.isAdmin())
                    insertAdmin((Administer)tempAccount);
```

```
                else
                    insertUser((User)tempAccount);
            }
            temp=in.readLine();
        }
        in.close();
    }catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

b） 用户登陆身份验证

遍历备份在本地的用户列表进行核验。在登陆成功后，会在 Authorization 类中生成一个连接供后续操作使用。而在该用户登出后，连接也会自动断开。

```
public static boolean checkAuthorization(String id,String pwd)
{
    for(Account t:account)
    {
        if(t.getId().equals(id)&&t.getPassword(pwd))return true;
    }
    return false;
}
public static void logIn(String id)
{
    for(Account t:account)
    {
        if(t.getId().equals(id)) {
            currentAccount=t;
            currentConnection = basicOperation.getConnection();
        }
    }
}
public static void logOut()
{
    currentAccount=null;
    try {
        currentConnection.close();
    } catch (SQLException se) {
        se.printStackTrace();
    }
}
```

c） 用户增删

由六个函数组成，前四个函数为 private 静态成员函数，在确认用户存在或不存在后进行相应的 SQL 更新。详细代码略。

```
private static boolean insertUser(User t) {
    if (existID(t.getId()))
        return false;
    else {
        account.add((Account)t);
        try {
            Connection conn = null;
            Class.forName(serverInfo.JDBC_DRIVER);
            conn = DriverManager.getConnection(serverInfo.getUrl());
            String sql = "insert into user (cardID, name, dept, type, passwd) values(?, ?, ?, ?, ?)";
            PreparedStatement pstmt;
            System.out.println("##### Inserting User " + t.getId());
            pstmt = (PreparedStatement)conn.prepareStatement(sql);
            pstmt.setString(1, t.getId());
            pstmt.setString(2, t.getName());
            pstmt.setString(3, t.getDepartment());
            pstmt.setString(4, t.getType());
            pstmt.setString(5, t.getPass());
            pstmt.executeUpdate();
            pstmt.close();
            conn.close();
        } catch (SQLException se) {
            se.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    }
}

private static boolean deleteUser(User t)
private static boolean insertAdmin(Administer t)
private static boolean deleteAdmin(Administer t)
```

后两个函数为 public static，是可供调用的接口。会根据账户类型自行调用私有成员函数，并判断经手人是否为管理员。

```
public static boolean insertAccount(Account worker, Account acct) {
    if (acct.isAdmin() == true || worker.isAdmin() == false) return false;
    return insertUser((User)acct);
}
```

```
public static boolean deleteAccount(Account worker, String id) {
    if (worker.isAdmin() == false) return false;
    for(Account t:account) {
        if (t.isAdmin()) continue;
        if (t.getId().equals(id))
            return deleteUser((User)t);
    }
    return false;
}
```

(2.)基本操作模块

d) 获取链接

使用 serverInfo 中的服务器信息获取一个数据库连接。

```
public static Connection getConnection() {
    Connection conn = null;
    try {
        Class.forName(serverInfo.JDBC_DRIVER);
        conn = (Connection)DriverManager.getConnection(serverInfo.getUrl());
    } catch (SQLException se) {
        se.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return conn;
}
```

e) 两种查询

第一种查询是为已经完成编辑的 sql 语句进行的，程序将通过获取到的连接直接执行这条 sql 语句，并返回查询得到的结果。

第二种查询是为一个 sql 模板和对应的参数列表进行的，程序在执行之前会先使用 prepareStatement 对这个 sql 模板进行填充，再执行查询。在实际使用中绝大部分 jdbc 操作都是通过这种查询完成的。

值得注意的是在执行这两个操作时都必须传入一个已经获取好的连接，因为如果连接在方法里获得，那么将面临两难的情况：

- 如果在方法中最后将连接关闭，则查询结果 ResultSet 无法送出，只能将内容取出再传出，但这就失去了黑箱封装的意义，这个基本方法应该不能获取 sql 查询的相关信息。
- 如果将 ResultSet 送出，则无法关闭这个连接。

```
public static ResultSet select(Connection conn, String sql) {
    Statement stmt = null;
    ResultSet rs = null;
    //System.out.println("====================");
    //System.out.println(sql);
```

```
        //System.out.println("====================");
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
        } catch (SQLException se) {
            se.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return rs;
    }

    public static ResultSet selectWithArgs(Connection conn, PreparedStatement pstmt, ArrayList<String>
    args) {
        ResultSet rs = null;
        try {
            //System.out.println(args);
            for(int i = 0; i < args.size(); i++)
            {
                pstmt.setString(i + 1, args.get(i));
            }
            rs = pstmt.executeQuery();
        } catch(SQLException se) {
            se.printStackTrace();
        } catch(Exception e) {
            e.printStackTrace();
        }
        return rs;
    }
```

f) 两种修改

方法与查询大同小异，只是返回值不同，这里只给出函数定义。

```
public static void update(Connection conn, String sql)
public static void updateWithArgs(Connection conn, PreparedStatement pstmt, ArrayList<String>
args)
```

(3.)图书管理模块

g) 图书导入

图书导入的逻辑为以下几步：

1) 查询被导入的书是否已经存在

2) 若不存在，则向 Book 表中插入一条记录

3) 若存在，则通过主码 bookID 执行 update，将该书的余量和总量修改。

```
args.add(b.getId());
```

```
sql = "select * from Book where bookID = ?";
pstmt = ***;
if (rs.next() == false) {
    sql = "insert into Book (bookID, type, title, press, year, author, price, num, stock)
values(?,?,?,?,?,?,?,?,?)";
    pstmt = ***;
    pstmt.setString(1, b.getId());
    ***
    pstmt.executeUpdate();
    pstmt.close();
} else {
    sql = "update Book set num = num + ?, stock = stock + ? where bookID = ?";
    pstmt = ***;
    pstmt.setInt(1, b.getNum());
    pstmt.setInt(2, b.getStock());
    pstmt.setString(3, b.getId());
    pstmt.executeUpdate();
    pstmt.close();
}
pstmt.close();
```

h) 图书借阅

图书借阅的逻辑分以下几步：

1) 首先查询该借书证是否有尚未归还的同一本书，若有则不能重复借阅。
2) 然后检查 Book 表中是否存在该书，若不存在则返回出错信息。
3) 然后检查该书是否有余量（和第二步共用 select 结果）
4) 若没有余量，则查询获得所有该书的未归还记录，选取其中最近的到期时间返回；
5) 若有余量，则向 entry 中插入记录并修改 Book 中的余量。

```
try {
    sql = "select * from entry where cardID = ? and bookID = ? and returnDate is null";
    pstmt = ***;
    pstmt.setString(1, cardID);
    pstmt.setString(2, bookID);
    rs = pstmt.executeQuery();
    if (rs.next() != false) {
        return "cannot borrow more than one certain book";
    } else {
        sql = "select * from Book where bookID = ?";
        pstmt = ***;
        pstmt.setString(1, bookID);
        rs = pstmt.executeQuery();
        if (rs.next() == false) {
            return "no such book";
```

```
        } else if (rs.getInt("stock") == 0) {
            sql = "select dueDate from entry where bookID = ? and returnDate is null";
            pstmt = ***;
            pstmt.setString(1, bookID);
            rs = pstmt.executeQuery();
            ArrayList<String> dueDates = new ArrayList<String>();
            while (rs.next()) {
                String date = rs.getString("dueDate");
                dueDates.add(date);
            }
            Collections.sort(dueDates);
            return "no book left, the earlist due date is " + dueDates.get(0);
        } else {
            sql = "insert into entry (cardID, bookID, borrowDate, dueDate, returnDate, adminID)
values(?, ?, ?, ?, ?, ?)";
            pstmt = ***;
            pstmt.setString(1, cardID);
            pstmt.setString(2, bookID);
            pstmt.setString(3, borrowDate);
            pstmt.setString(4, dueDate);
            pstmt.setString(5, null);
            pstmt.setString(6, Authorization.currentAccount.getId());
            pstmt.executeUpdate();

            sql = "update book set stock = stock - 1 where bookID = ?";
            pstmt = ***;
            pstmt.setString(1, bookID);
            pstmt.executeUpdate();

            return "succeed";
        }
    }
}
```

i） 图书归还

图书归还较为简单，若存在未归还记录，更新 entry 和 Book 表即可，详细代码略。

j） 图书查询

图书查询是模糊查询，由于查询参数包括字符串和数值，我们在后端作如下约定。

- 对于字符串，若不作限制则传入空串；
- 对于数值，若不作限制，下限传入一个最小值，上限传入一个最大值。

在构造 sql 语句时，使用模版构造。先添加四个数值约束进入 sql 模板，并将四个数值放入参数列表，这样在 sql 查询中对“and”的格式化会方便很多。

之后对于字符串的约束，若不为空，就添加该约束，并将该参数放入参数列表。
最后使用模板查询工具进行查询。
获得查询结果后，将每一条查询得到的记录导出，生成若干个 Book 对象，放入列表，返回前端。

```
public static ArrayList<Book> searchBook(String bookID, String type, String name,
                                         String press, int minYear, int maxYear,
                                         String author, float minPrice, float maxPrice) {
    String sql = "select * from Book where";
    ArrayList<String> args = new ArrayList<String>();
    sql += " year >= ?"; args.add(Integer.toString(minYear));
    sql += " and year <= ?"; args.add(Integer.toString(maxYear));
    sql += " and price >= ?"; args.add(Float.toString(minPrice));
    sql += " and price <= ?"; args.add(Float.toString(maxPrice));
    if (bookID.equals("") == false) {
        sql += " and bookID = ?";
        args.add(bookID);
    }
    if (type.equals("") == false) {
        sql += " and type = ?";
        args.add(type);
    }
    if (name.equals("") == false) {
        sql += " and title = ?";
        args.add(name);
    }
    if (press.equals("") == false) {
        sql += " and press = ?";
        args.add(press);
    }
    if (author.equals("") == false) {
        sql += " and author = ?";
        args.add(author);
    }

    try {
        Connection conn = basicOperation.getConnection();
        PreparedStatement pstmt =
(PreparedStatement)Authorization.currentConnection.prepareStatement(sql);
        ResultSet rs = basicOperation.selectWithArgs(conn, pstmt, args);
        ArrayList<Book> ret = new ArrayList<Book>();
        while (rs.next()) {
            Book tmp = new Book(rs.getString("bookID"),
                                rs.getString("type"),
                                rs.getString("title"),
```

```
                                    rs.getString("press"),
                                    rs.getInt("year"),
                                    rs.getString("author"),
                                    rs.getFloat("price"),
                                    rs.getInt("num"),
                                    rs.getInt("stock"));
                ret.add(tmp);
            }
            return ret;
        } catch (SQLException se) {
            se.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

(4.)用户管理模块

k) 添加用户

调用 Authorization 类中的 insertAccount 方法。

l) 删除用户

1) 先判断该用户是否有书未归还，若有则删除失败。
2) 然后尝试调用 Authorization 类中的 deleteAccount 方法。
3) 若删除成功，则同时销去 entry 中所有该 ID 的借书记录；
4) 否则删除失败。

```
public static boolean deleteUser(String id)
{
    try {
        String sql = "select * from entry where cardID = ? and returnDate is null";
        PreparedStatement pstmt =
(PreparedStatement)Authorization.currentConnection.prepareStatement(sql);
        pstmt.setString(1, id);
        ResultSet rs = pstmt.executeQuery();
        if (rs.next()) {
            return false;
        } else if (Authorization.deleteAccount(Authorization.currentAccount, id)) {
            sql = "delete from entry where cardID = ?";
            pstmt = (PreparedStatement)Authorization.currentConnection.prepareStatement(sql);
            pstmt.setString(1, id);
            pstmt.executeUpdate();
            return true;
```

```
            } else {
                return false;
            }
        } catch (SQLException se) {
            se.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return false;
}
```

m) 查询用户

通过用户 ID 直接查询即可。

n) 查询用户借阅记录

通过用户 ID 在 entry 表中查阅即可。

注意由于类似该功能的诸多查询需要，使用该应用须关闭 mysql 的安全模式。

~~~~~

下一步改进

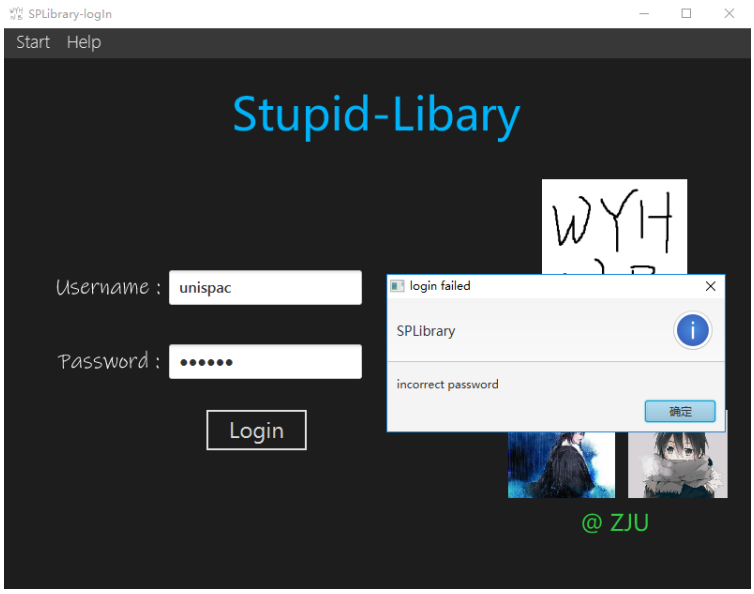
a. 使用 basicOperation 对 Authorization 类进行重构。由于 Authorization 类在 basicOperation 之前开发，因此存在一定的代码累赘，可以进行进一步改善以方便下一步开发。
b. 查询用户记录时只在 entry 表中进行检索，可以添加一个 join 使得可以显示该书的书名和余量。

五、 实验结果与分析

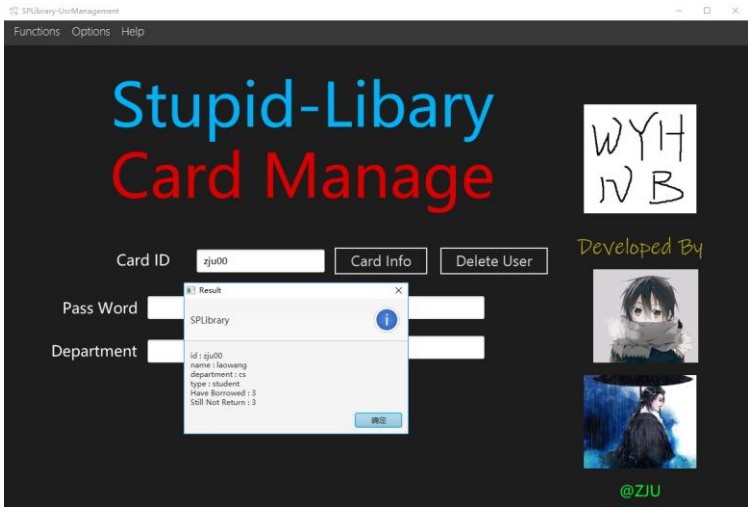
1. 账号与权限

我们用两个账号分别来测试。

一个是管理员账号（ID：unispac，Passwd:996ICU）。另一个是普通的借书证账号（ID：zju00，Passwd:123456）。
~~~~~

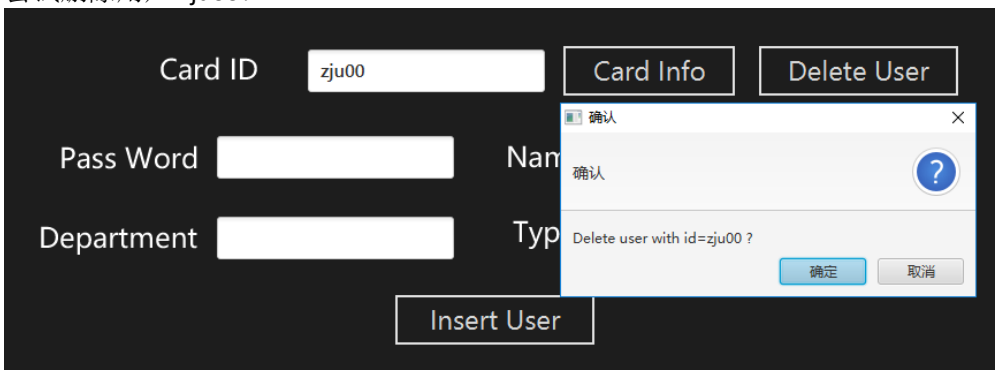


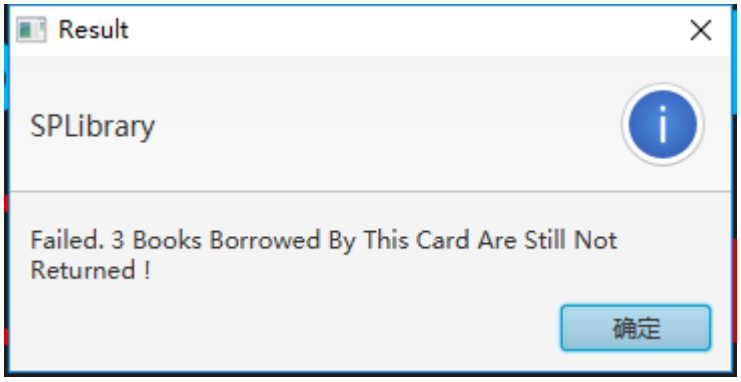
当输入的密码与数据库中的信息不吻合时，登陆就会被阻止。
管理员 unispac 账号信息成功认证后，就能登陆进入系统，并且可以访问所有功能界面。



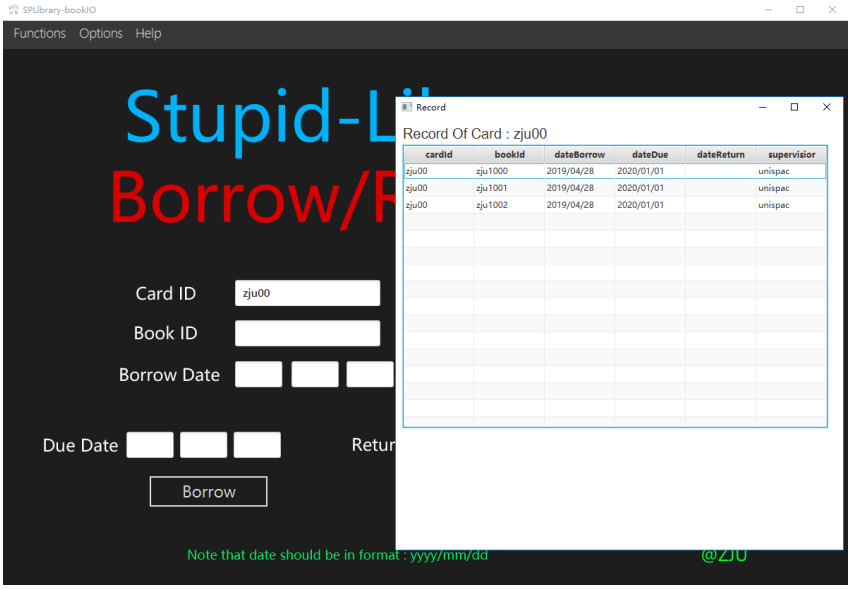
如图，unispac 可以查询 zju00 账号的信息。弹出对话框，显示账户的所有信息，并且可以看到账户借书记录总共有多少条，是否还有书没有归还。可以看到 zju00 还有三本书没有归还。

尝试删除用户 zju00：



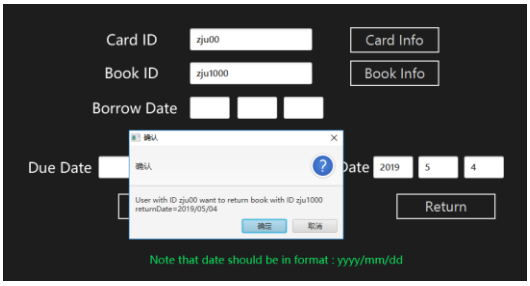


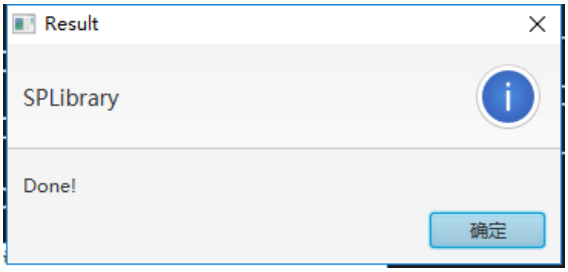
提示删除失败，因为用户还有书没有归还。

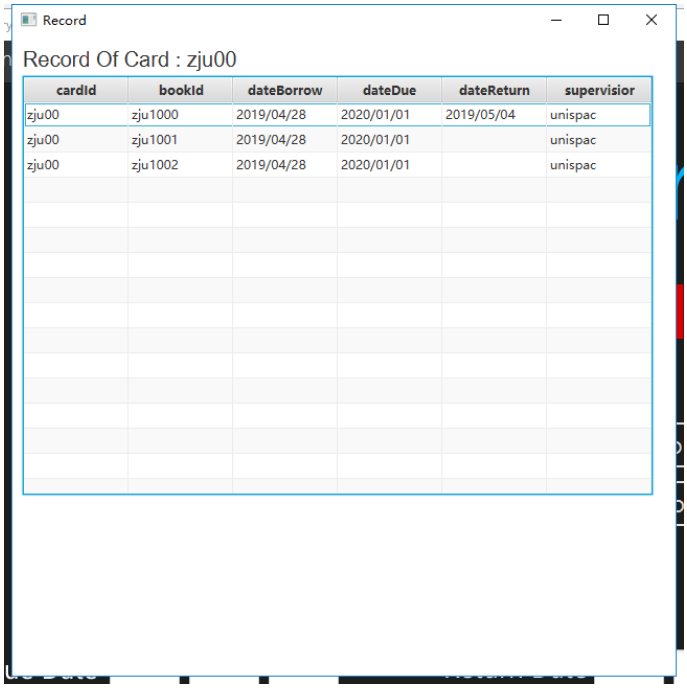


去借书页面查看用户 zju00 的借书记录，可以在表格中看到用户的三条借书记录。dateReturn 属性还是空的，表明没还。可以看到都是在管理员 unispac 的操作下，借出的。

尝试还书：



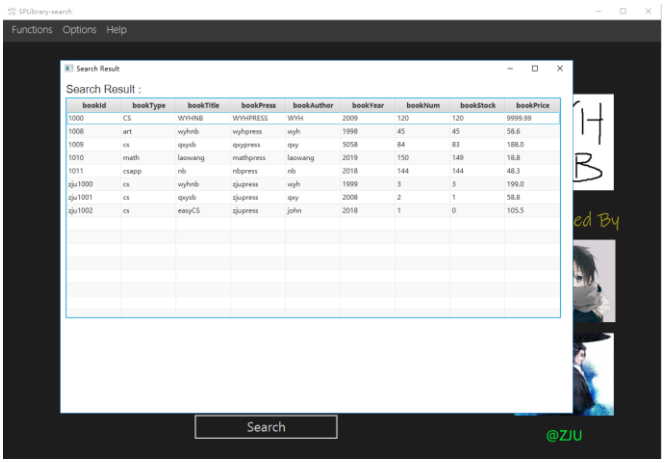




再次查询，可以看到这本书已经还上了。

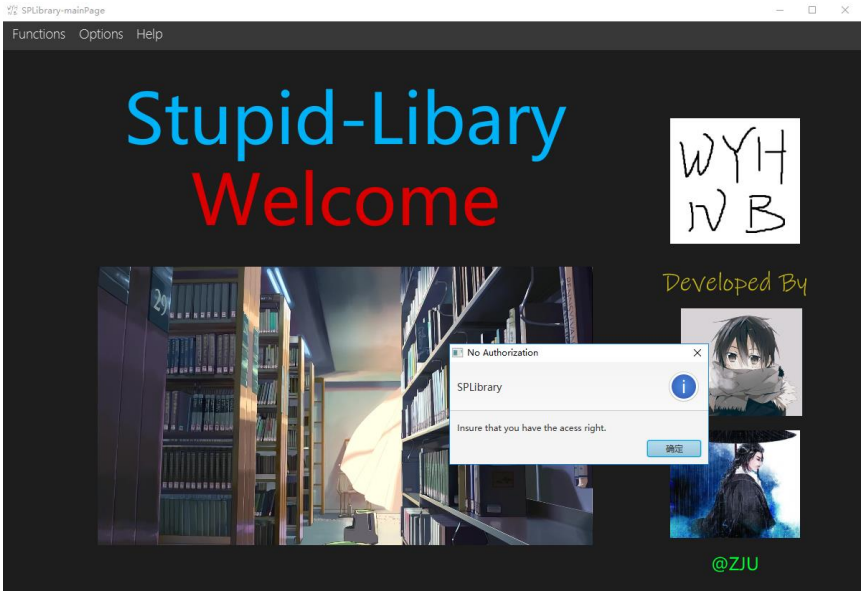
现在登陆 zju00 账号，尝试普通借书账号的权限。

进入 Search 页面，尝试搜索图书：



成功，书籍列表弹出。

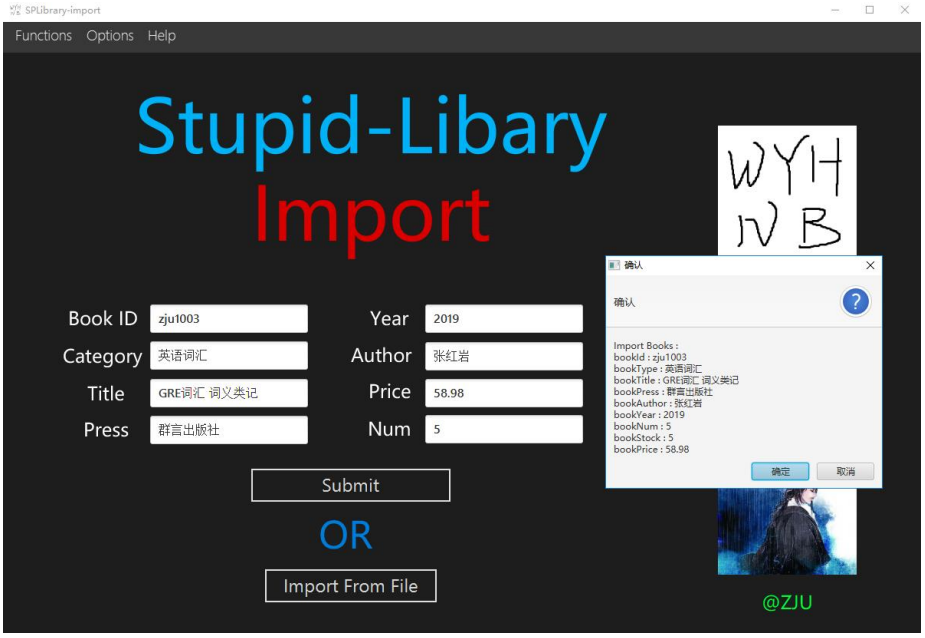
尝试进入其它管理界面：

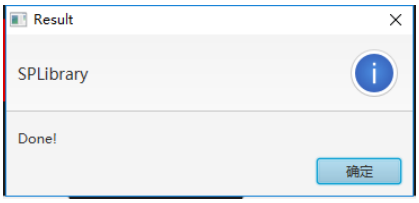


权限被拒。
完全符合实验预期。

2. 书籍导入

单本导入：



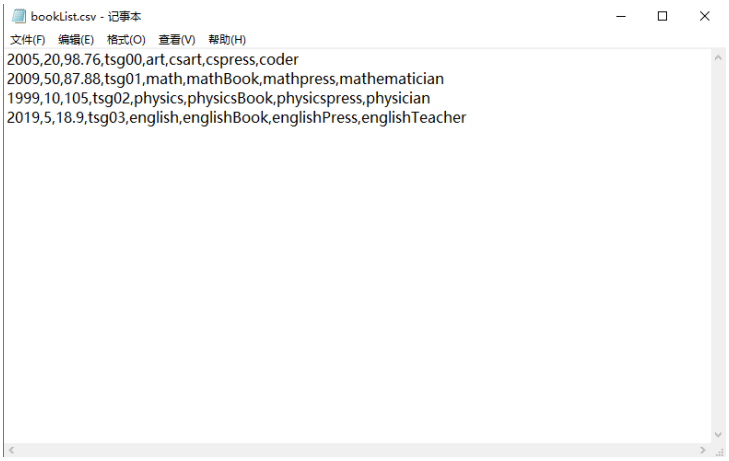


批量导入：

系统支持批量导入.csv 格式的书单。

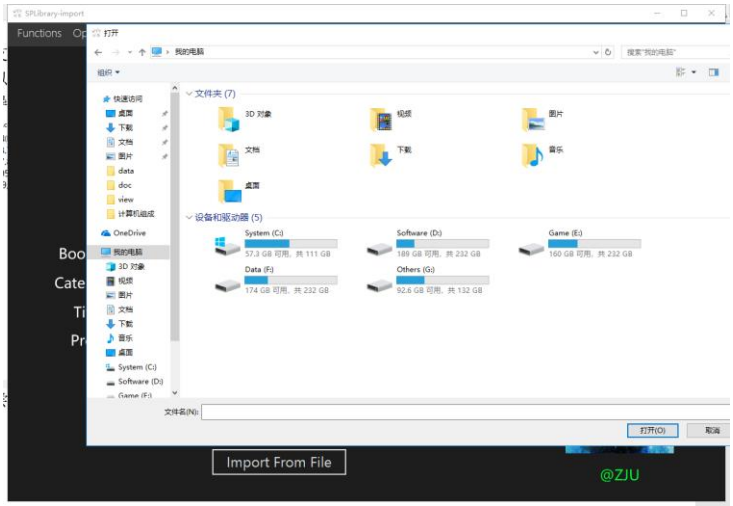
必须以（year,num,price,id,category,title,press,author）的格式一行一行的列出。

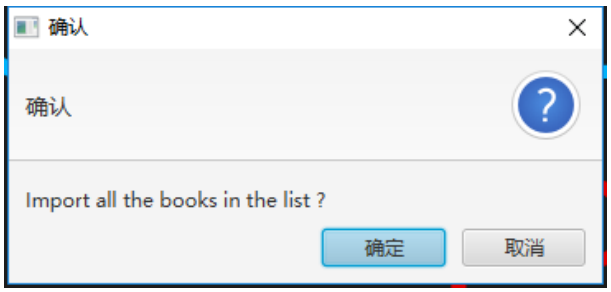
整个导入过程满足原子性，如果中途格式不符合要求，整个导入都会被拒绝。

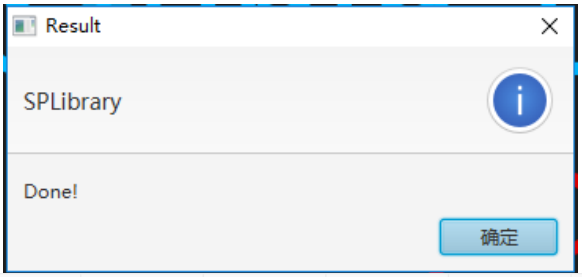


现在尝试导入上图清单。

点击 Import From File 弹出文件选择框 ：



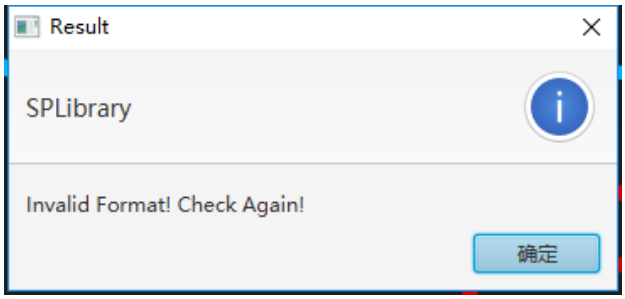




tsg00	art	csart	cspress	coder	2005	20	20	98.76
tsg01	math	mathBook	mathpress	mathematician	2009	50	50	87.88
tsg02	physics	physicsBook	physicspress	physician	1999	10	10	105.0
tsg03	english	englishBook	englishPress	englishTeacher	2019	5	5	18.9

可以看到，已经成功导入到清单中了，现在将第三条内容破坏一下，检测原子性：

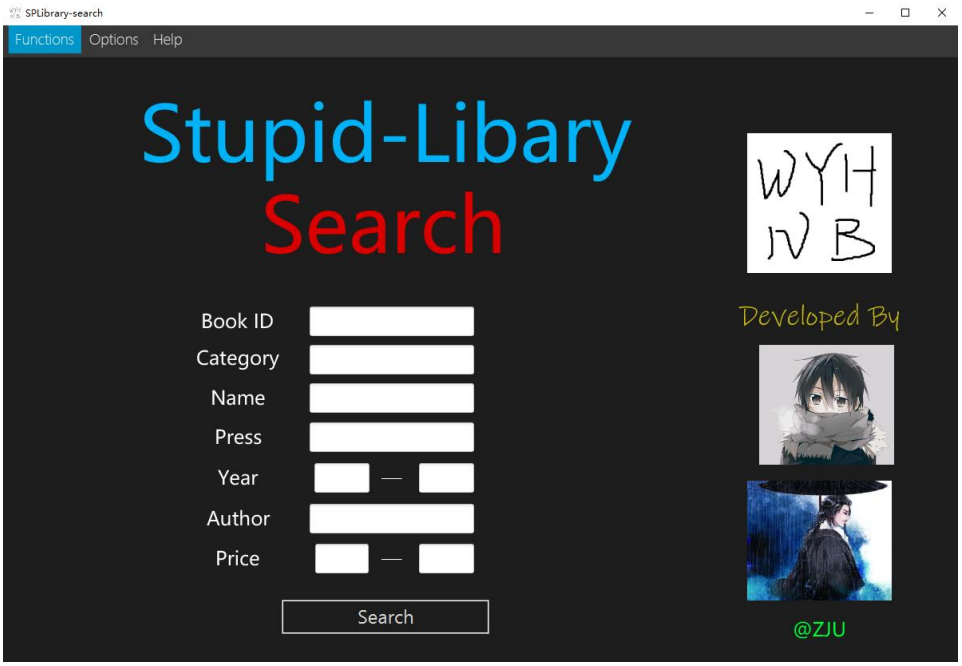
```
2005,20,98.76,tsg00,art,csart,cspress,coder
2009,50,87.88,tsg01,math,mathBook,mathpress,mathematician
1999,,105,tsg02,physics,physicsBook,physicspress,
2019,5,18.9,tsg03,english,englishBook,englishPress,englishTeacher
```



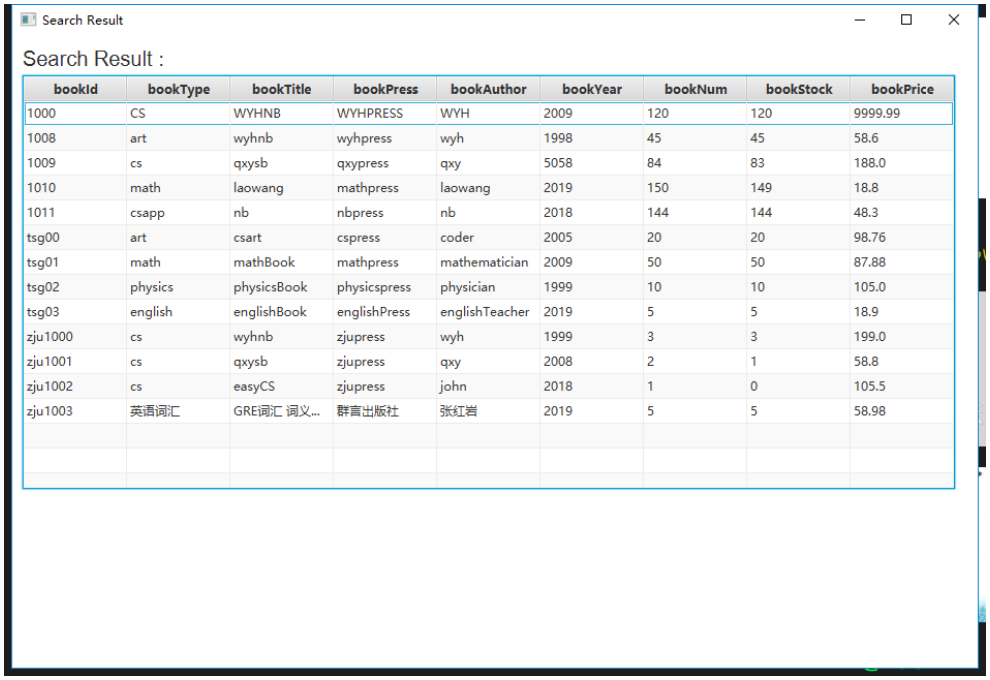
tsg00	art	csart	cspress	coder	2005	20	20	98.76
tsg01	math	mathBook	mathpress	mathematician	2009	50	50	87.88
tsg02	physics	physicsBook	physicspress	physician	1999	10	10	105.0
tsg03	english	englishBook	englishPress	englishTeacher	2019	5	5	18.9

书籍量全都没变，说明刚才虽然前两条合法，但是却没有被执行导入。

3. 书籍复合查询



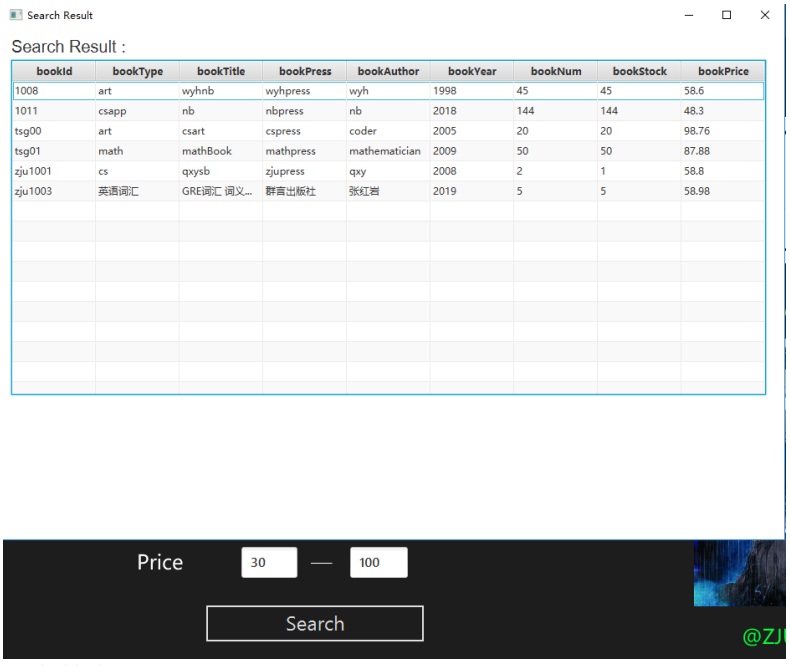
查询界面如上，可以选填一部分内容完成复合查询。
不填，直接查询：



bookId	bookType	bookTitle	bookPress	bookAuthor	bookYear	bookNum	bookStock	bookPrice
1000	CS	WYHNB	WYHPRESS	WYH	2009	120	120	9999.99
1008	art	wyhnb	wyhpress	wyh	1998	45	45	58.6
1009	cs	qxysb	qxypress	qxy	5058	84	83	188.0
1010	math	laowang	mathpress	laowang	2019	150	149	18.8
1011	csapp	nb	nbpress	nb	2018	144	144	48.3
tsg00	art	csart	cspress	coder	2005	20	20	98.76
tsg01	math	mathBook	mathpress	mathematician	2009	50	50	87.88
tsg02	physics	physicsBook	physicspress	physician	1999	10	10	105.0
tsg03	english	englishBook	englishPress	englishTeacher	2019	5	5	18.9
zju1000	cs	wyhnb	zjupress	wyh	1999	3	3	199.0
zju1001	cs	qxysb	zjupress	qxy	2008	2	1	58.8
zju1002	cs	easyCS	zjupress	john	2018	1	0	105.5
zju1003	英语词汇	GRE词汇 词义...	群言出版社	张红岩	2019	5	5	58.98

所有书籍都被列出来了。

现在查找价格在 30 到 100 以内的书：



bookId	bookType	bookTitle	bookPress	bookAuthor	bookYear	bookNum	bookStock	bookPrice
1008	art	wyhnb	wyhpress	wyh	1998	45	45	58.6
1011	csapp	nb	nbpress	nb	2018	144	144	48.3
tsg00	art	csart	cspress	coder	2005	20	20	98.76
tsg01	math	mathBook	mathpress	mathematician	2009	50	50	87.88
zju1001	cs	qxysb	zjupress	qxy	2008	2	1	58.8
zju1003	英语词汇	GRE词汇 词义...	群言出版社	张红岩	2019	5	5	58.98

满足要求的书籍全部出现了。

查询出版时间在 2010 年之后，价格在 30 到 100 以内的书：

bookId	bookType	bookTitle	bookPress	bookAuthor	bookYear	bookNum	bookStock	bookPrice
1011	csapp	nb	nbpress	nb	2018	144	144	48.3
zju1003	英语词汇	GRE词汇 词义...	群言出版社	张红岩	2019	5	5	58.98

查询结果和预期相符合。

4. 多功能排序

由于 JavaFX 提供的表格视图自带灵活排序。所以我们可以在界面上随意的选择排序关键字，并且可以要求顺序和倒序。

bookId ▲	bookType	bookTitle	bookPress	bookAuthor	bookYear	bookNum	bookStock	bookPrice
1000	CS	WYHNB	WYHPRESS	WYH	2009	120	120	9999.99
1008	art	wyhnb	wyhpress	wyh	1998	45	45	58.6
1009	cs	qxysb	qxypress	qxy	5058	84	83	188.0
1010	math	laowang	mathpress	laowang	2019	150	149	18.8
1011	csapp	nb	nbpress	nb	2018	144	144	48.3
tsg00	art	csart	cspress	coder	2005	20	20	98.76
tsg01	math	mathBook	mathpress	mathematician	2009	50	50	87.88
tsg02	physics	physicsBook	physicspress	physician	1999	10	10	105.0
tsg03	english	englishBook	englishPress	englishTeacher	2019	5	5	18.9
zju1000	cs	wyhnb	zjupress	wyh	1999	3	3	199.0
zju1001	cs	qxysb	zjupress	qxy	2008	2	1	58.8
zju1002	cs	easyCS	zjupress	john	2018	1	0	105.5
zju1003	英语词汇	GRE词汇 词义...	群言出版社	张红岩	2019	5	5	58.98

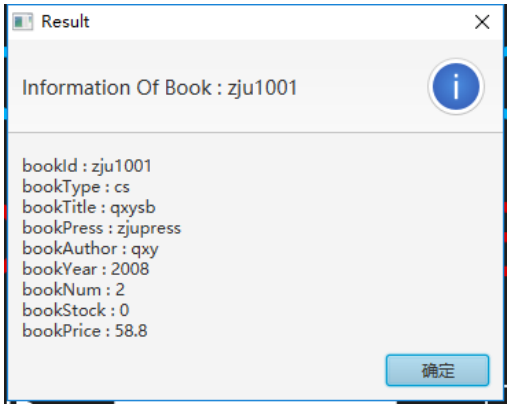
bookId ▼	bookType	bookTitle	bookPress	bookAuthor	bookYear	bookNum	bookStock	bookPrice
zju1003	英语词汇	GRE词汇 词义...	群言出版社	张红岩	2019	5	5	58.98
zju1002	cs	easyCS	zjupress	john	2018	1	0	105.5
zju1001	cs	qxysb	zjupress	qxy	2008	2	1	58.8
zju1000	cs	wyhnb	zjupress	wyh	1999	3	3	199.0
tsg03	english	englishBook	englishPress	englishTeacher	2019	5	5	18.9
tsg02	physics	physicsBook	physicspress	physician	1999	10	10	105.0
tsg01	math	mathBook	mathpress	mathematician	2009	50	50	87.88
tsg00	art	csart	cspress	coder	2005	20	20	98.76
1011	csapp	nb	nbpress	nb	2018	144	144	48.3
1010	math	laowang	mathpress	laowang	2019	150	149	18.8
1009	cs	qxysb	qxypress	qxy	5058	84	83	188.0
1008	art	wyhnb	wyhpress	wyh	1998	45	45	58.6
1000	CS	WYHNB	WYHPRESS	WYH	2009	120	120	9999.99

bookId	bookType	bookTitle	bookPress	bookAuthor	bookYear ▲	bookNum	bookStock	bookPrice
1008	art	wyhnb	wyhpress	wyh	1998	45	45	58.6
zju1000	cs	wyhnb	zjupress	wyh	1999	3	3	199.0
tsg02	physics	physicsBook	physicspress	physician	1999	10	10	105.0
tsg00	art	csart	cspress	coder	2005	20	20	98.76
zju1001	cs	qxysb	zjupress	qxy	2008	2	1	58.8
tsg01	math	mathBook	mathpress	mathematician	2009	50	50	87.88
1000	CS	WYHNB	WYHPRESS	WYH	2009	120	120	9999.99
zju1002	cs	easyCS	zjupress	john	2018	1	0	105.5
1011	csapp	nb	nbpress	nb	2018	144	144	48.3
zju1003	英语词汇	GRE词汇 词义...	群言出版社	张红岩	2019	5	5	58.98
tsg03	english	englishBook	englishPress	englishTeacher	2019	5	5	18.9
1010	math	laowang	mathpress	laowang	2019	150	149	18.8
1009	cs	qxysb	qxypress	qxy	5058	84	83	188.0

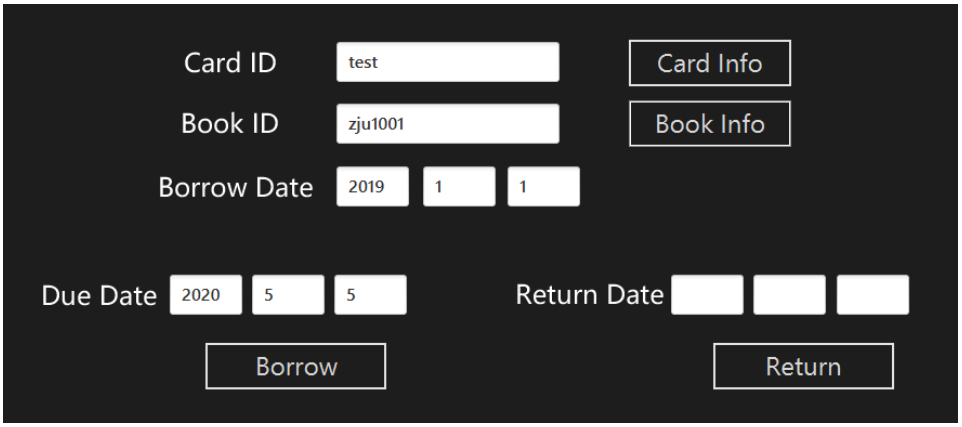
5. 检查书籍存量是否会负

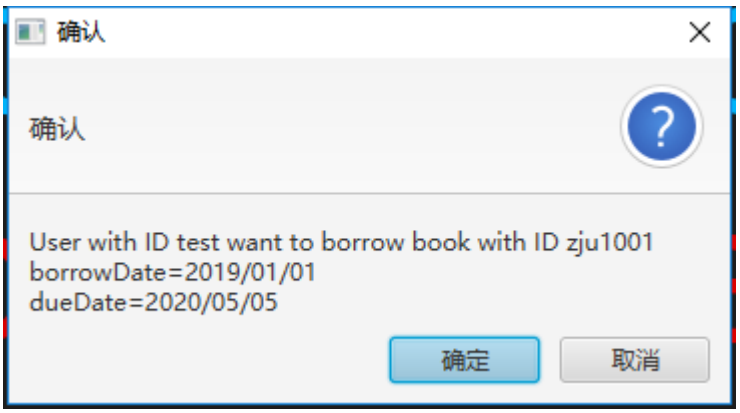
验收时总共有 4 个检测点，前面测试过程中，其中三个已经做过了。最后我们测试一下书籍已经被借完时的情况。

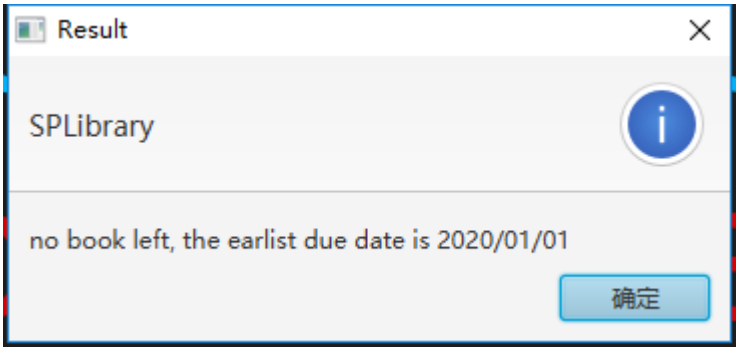
现在以 zju1001 这本书为例：



库存已经耗尽。
尝试借出：







弹出错误信息，并提示最早还书的截止日期。符合预期。

六、 讨论、心得

本次实验结合了 SQL 语句的综合使用、JDBC 的特性学习和 Java 开发技巧的学习掌握，比前四次实验要有挑战的多。结合面向对象程序设计课程设计的经验，我们两位同学避免了不少在合作开发上会碰到的问题。漆翔宇同学有相对丰富的 Java 开发经验，因此前端开发的重任主要交给了他，王宇晗同学以学习并使用前端接口为主。后端开发两人按照要求都写了一定的 JDBC 相关的代码，王宇晗同学在 Engine 包中付出了非常多的心血，保证了应用功能的顺利实现。这次实验对我们两位同学有着深刻的意义，全方面提升了我们的个人开发经验和合作开发能力。