武汉大学计算机学院
本科生实验报告

数据结构实验报告

简易图书管理系统大作业

“白泽书房”图书管理 APP

专 业 名 称	：计算机科学与技术
课 程 名 称	：数据结构
指 导 教 师	：安　　扬
学 生 学 号	：2017301500061
学 生 姓 名	：彭 思 翔
学 生 班 级	：计科二班

二〇一九 年 1 月

摘 要

实验标题：简易图书管理系统——“白泽书房”图书管理APP

实验目的：完成大作业，进一步熟悉面向对象的思想，进一步熟悉服务器、客户端模式的编程，熟悉各种数据结构的选择、构建和应用，熟悉部分算法，完成一个手机端图书管理应用。

实验内容：使用 Android Studio 编写 APP前端，使用JAVA语言编写后台。

实验结果：实现了用户的注册、登陆、更换密码，并对管理者用户和普通用户在权限上进行了区分。实现了书籍的（编号、书名、作者）查询、新增、删除、添加、减少以及借阅和归还功能，实现书籍的导入更新，实现了用户的查询和删除，实现了超时不能续借、借书不超过五本限制、查无此书等鲁棒性能，并且在书名查询之中实现了模糊匹配。此次书籍使用爬虫增加至了近三万本。我们还添加了聊天的功能。

关键词：Android; Application; C/S 模式; 数据结构; 查询算法

目 录

1 实验目的和意义

1.1 实验要求

1.1.1 问题描述

编写一个图书管理系统，能完成简单的图书管理业务。

1.1.2 基本要求

(1)新书入库：登记新书的编号、书名、著者和数量;

(2)书目信息维护：删除（旧书销毁，若该书还有读者未还，则不可销毁）、更新（新购进若干本相同书目）;

(3)读者信息维护：新增、删除读者（读者的注销必须在还清所有书目的情况下才可进行），还可根据实情添加补办读者证等功能;

(4)查询：书目查询需罗列出该书的基本信息及该书当前的借阅记录，读者查询需罗列出读者的基本信息及当前借阅的书目信息;

(5)借阅、归还功能：需登记借阅的图书号、读者证号、借阅日期和应归还日期。

1.1.3 扩展要求

(1)增加图书分类信息，实现图书分类查询;

(2)增加借阅限制，若有超期未还则不可再借，借阅数目不超过 5 本;

(3)可参考现实图书馆增加相应的功能，如续借、预订等

1.2 实验目的

本实验是在完成大作业的基础上，为了对外观界面以及内核实现的更高的追求，进行一些大胆的尝试和代码想法的实验，并适当进行扩展。但是这次的时间比较紧迫，有部分想法并没有来得及实现和完善，未来有时间会逐步改进的。

一是要实现基于安卓端的 APP 界面。我在之前已经熟悉过了 Android Studio 的使用以及 JAVA 和 XML 语法，所以在这次时间比较紧迫的情况下实现 APP 的界面比较方便美观。

二是服务器客户端模式的编程。服务器与客户端的通信依然用 Socket 传输，但是这次遇到了些许问题，比如信息的限制导致查询的书籍不能一次返回，在报告后面的部分会有介绍。

三是要懂得面向对象的编程的合作编程，对类的函数的布置有初步的了解和掌握，理解 Android 库中的抽象类。这次还尝试双人合作，一人编写面向对象的数据结构设计并提供接口，另一人进行接口的调用即可，大大减小了项目的耦合性。

四是要建立起强大而合适的数据结构和算法。我们的目的不是要做一个简单的图书管理示例程序，而是要实实在在地进行几万本甚至几百万本书籍的管理，这就需要选择合适的数据结构、设计合适的算法进行支撑。

五是完善的用户体验和鲁棒性能。我们希望用户的体验良好，支持注册登陆、查找书名时可以模糊匹配，有完善的操作反馈。并且在输入错误时增加检验使系统健壮。

1.3 实验意义

本次实验进一步熟悉了如何选择合适的数据结构，如何设计合适的算法，同时也进一步学习了计算机网络通信的知识，为后来的课程打下一定的基础。我们也深刻体会到了如何合作编程，降低项目的耦合性，在合作中收获编程的乐趣。

1.4 涉及知识点

本次实验设计到课堂的知识点有：树以及树的算法（Trie 树，Splay 树）、递归调用、搜索算法、链表、数组线性表等其他数据结构。

2 实验设计

2.1 实验概述

实验环境：

客户端环境：

Windows 10 家庭中文版 AMD FX-9830P 内存 8.00GB 64 位

Android 7.0 STF-AL10C00B201 内存 6.0GB

服务器环境：

腾讯云 1 核 2 GB 1 Mbps Windows Server 2012 R2 Standard Intel(R) Xeon(R) CPU E5-26xx v4 内存 2.00GB 64 位

实验工具：

Android Studio、Visual Studio Code、PyCharm

实验简述：

本次实验将分服务器端和客户端分别设计，主要使用 Android Studio 的 JAVA 实现 APP 前端，Visual Studio Code 实现后端 JAVA 编写，pyCharm 负责数据爬虫，下面将介绍 UI 思路和类的设计、数据结构和算法的设计等设计方面内容。

2.2 服务器实验方案

2.2.1 功能要求

服务器要求能够接入客户端并接收来自客户端的消息,对客户端的消息进行含义分析。能够提供书籍的（编号、书名、作者）查找、新增、删除、增加、减少、加载、更新、借阅、归还等服务，提供用户的新建、登陆、更换密码、查找、删除的服务，提供报错服务，并把信息返回给客户端。

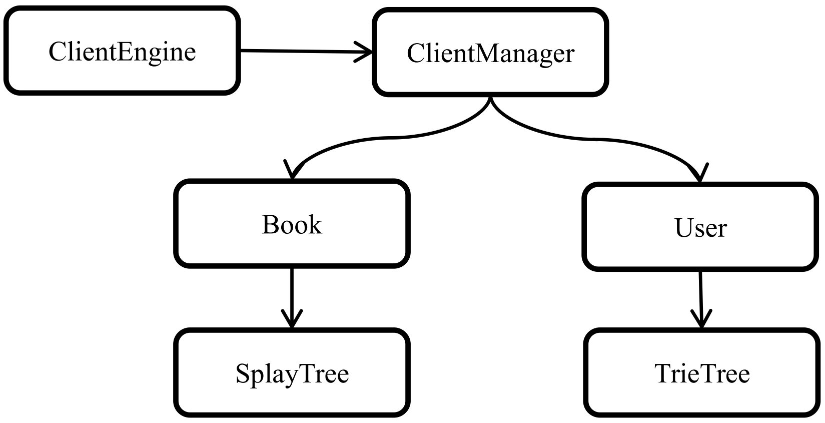
2.2.2 程序排布

目录和文件（未列入的为 IDE 自建文件，无需手动配置）		功能
com.creatures.siang.sever	ClientEngine.java	服务器启动
	ClientManager.java	为客户端提供服务器管理
	Book.java	书籍管理服务
	User.java	用户管理服务
	TrieTree.java	基础字典树类
	SplayTree.java	基础伸展树类

2.2.3 类之间的调用关系

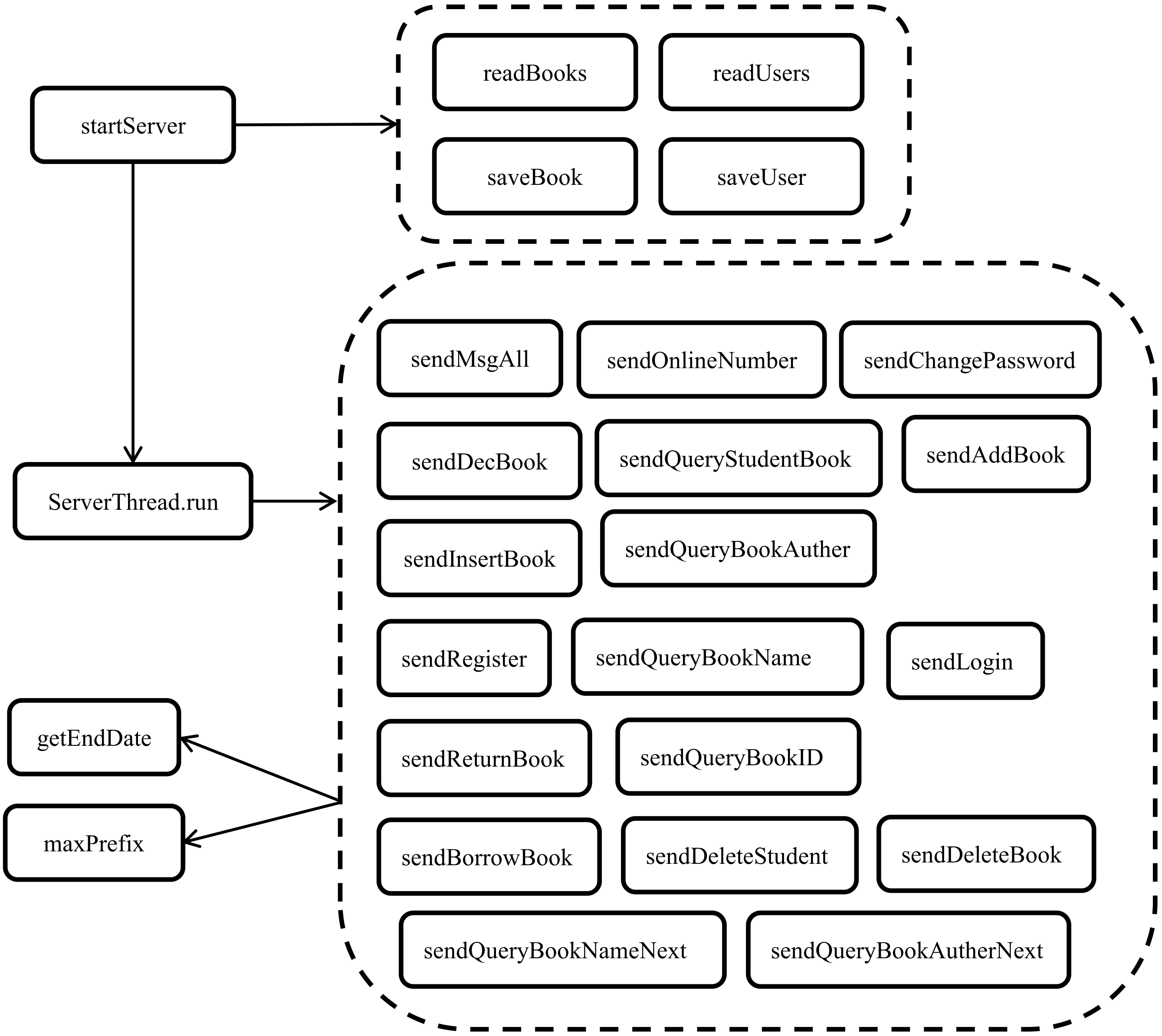
所有的类从 ClientEngine 开始启动，通过 clientManager 对书籍和用户进行管理，是服务器管理系统。其他类提供接口给服务器管理使用。Book 类书籍管理是建立在 Splay 树上的，User 类用户管理是建立在 Trie 树上的，稍后的数

据结构会进行解释说明。调用关系如图：



2.2.4 函数之间的调用关系

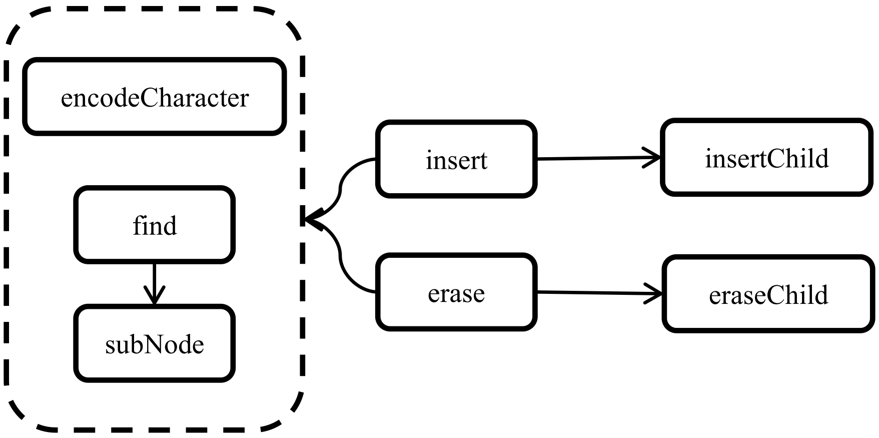
对于 ClientEngine 只是作为启动引擎只有主函数并没有什么函数之间的调用，但对于 ClientManager 却处理了所有来自客户端的功能请求，请求指令详见后面的算法及实现设计部分。



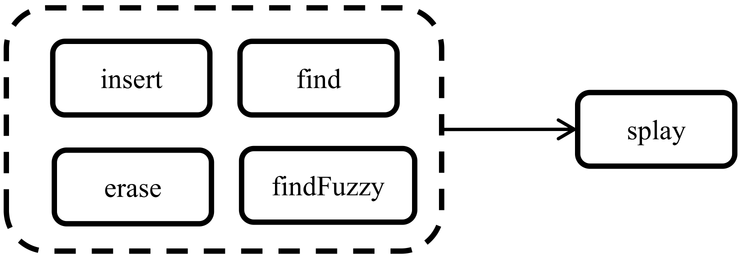
其中的 startServer 为启动服务器的接口，启动的同时调用 readBooks 和 readUsers 函数读入存入的数据，saveBooks 和 saveUsers 每隔一定时间保存数据。startServer 中调用线程子类 ServerThread.run 启动 Socket 连接，接收来自客户端的指令，根据指令执行对应的函数（函数功能根据名称对应指令功能，这里不再赘述函数功能）。在操作过程中会用到 getEndDate 获取当前时间以及 maxPrefix 求两个字符串的最长公共前缀，具体请见算法及实现设计。

至于 Book 和 User 管理类为服务器管理系统提供了诸多接口，内部的调用有但是不太重要，所以在此省略。

对于 Trie 树主要有 find、insert 和 erase 三种操作（操作解释详见数据结构设计），同时在构建 Trie 树的过程中提供了 subNode 查找子节点，insertChild 添加新的子节点，eraseChild 删除旧的子节点，encodeCharacter 的字符转码函数。



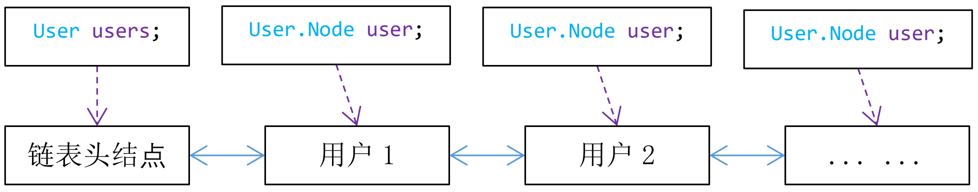
对于 Splay 树有也 find、findFuzzy、insert 和 erase 四种操作（操作解释详见数据结构设计），这四种操作由于传入参数的多少不同进行了重复定义，操作外部接口和内部函数也是重名的。但是思路是清晰的，都需要经典的 splay 旋转操作。对于 Splay 树由于需要构建的元素类型可能不同，所以元素定义为了泛型。



2.2.5 数据结构设计

User 类:为了能支持用户的查询、新增、删除等用户管理操作而设计的类。我们为储存用户信息和功能实现分别建立了两种数据结构——存储用户信息用的双链表而查询等功能实现则利用了字典树 TrieTree。

(1)储存用户信息：储存用户的数据结构是一个带头结点的双链表，每一个用户是一个结点，该结点包含该用户包括用户名、密码、借书信息在内的所有信息，结点相连形成双链表。其中链表的类型为 User，用户结点的类型为 User.Node（Node 是一个位于类 User 中的子类）。其构建方法如下图：



当程序开始运行时，先创建一个用户链表（只包含一个空的头结点），之后在程序运行过程中按照需要逐个添加或删除用户结点。

(2) 查询等功能实现：如果选择在刚刚的双链表暴力查找，然后执行插入和删除那么时间复杂度是很大的，假设有 N 个用户则时间复杂度达到了 O（N）。维护一棵 Trie 树，在 Trie 树的用户名字符串末尾节点添加指针指向对应双链表的位置，这样可以快速找到用户在双链表中的位置，时间复杂度为 O(Lenth)，其中 Lenth 为用户名字符串长度。说了这么久那么到底什么是 Trie 树呢？

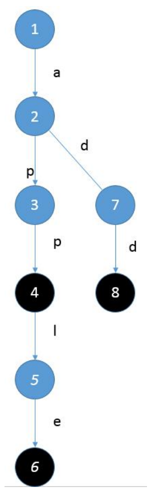
- 字典树：

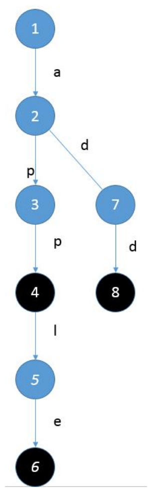
I. 简介：

又称单词查找树，Trie 树，是一种哈希树的变种。典型应用是用于统计，排序和保存大量的字符串（但不仅限于字符串），所以经常被搜索引擎系统用于文本词频统计。它的优点是：利用字符串的公共前缀来减少查询时间，最大限度地减少无谓的字符串比较，查询效率比哈希树高。

II. 原理：

假设我们需要根据用户名来匹配用户链表中有无此用户，可以将用户名创建为一棵 Trie 树，其过程如图 1 所示（假设我们已经添加了三个用户名：“app”、“apple”和“add”）：





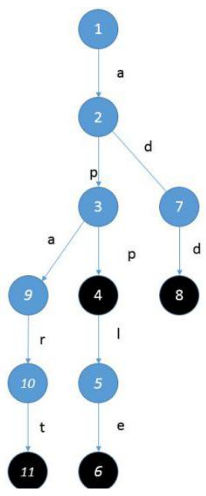


图 1　　　　　　　　　图 2　　　　　　　　　图 3

在上述 Trie 树中，每个节点都有一个有效位，当此结点为某个用户名的结尾时（如图 1 中的结点 4、6 和 8），则将有效位置为有效；否则有效位无效。由于用户名不允许重名，所以不存在同一个结点代表两个用户名的情况。但又由于不同用户名可能共用一个结点，所以需要一个计数位，记录此结点被用了几次。

因此，字典树的每一个结点需要储存以下数据：

①主键：字符类型，作为该结点的值，用于比较。（在本程序中，为简化操作，将用户名限制为只可使用 26 个小写字母 a~z 及 10 个阿拉伯数字 0~9，因此主键只能在这 36 种字符中取值）

②数据域：使用 Trie 树的目的是避免直接在用户链表中进行查找导致的高耗时,因此在 Trie 树中查找完毕后应将找到的数据返还给用户链表。（在本程序中，数据域为指向该用户名对应的用户在链表中的指针）。其关系如下图所示，注意图中的 trie 应是树状结构，这里为了方便将其画为链状。

③子结点数组：每一个结点最多有 36 个子结点（即①中所述的 36 种字符），只有当需要某子结点时才会创建，这样可以减少空间复杂度。

④有效位：当此结点为某个用户名的结尾时，则将有效位置为有效；否则有效位无效。

⑤计数位：由于不同用户名可能共用一个结点，所以需要一个计数位，记录此结点被用了几次。

III.实现：

a.插入：

如图 2 所示为在树中新加入一个用户名“apart”后。其步骤如下：

①从根结点（图 1 中的结点 1）开始一次搜索；

②取用户名的第一个字母 C，若当前结点的子树中无 C，则新建一个结点，值即为 C；若当前结点的子树中有 C，则转到该子树继续执行步骤②；

③当 C 为用户名的最后一个字母时，若当前结点的子树中无 C，则新建一个结点，值即为 C，并将此结点的有效位置为有效，将相关信息（如用户密码等）录入此结点中；若当前结点的子树中有 C 且有效位为无效，则将此结点的有效位置为有效，将相关信息录入此结点中；若当前结点的子树中有 C 且有效位为有效，说明此用户名已存在，则插入失败。

b.查找：

步骤和插入类似，只在当 C 为用户名的最后一个字母时，若当前结点的子树中有 C 且有效位为有效，说明此用户名存在，其他情况说明用户名不存在。

c.删除：

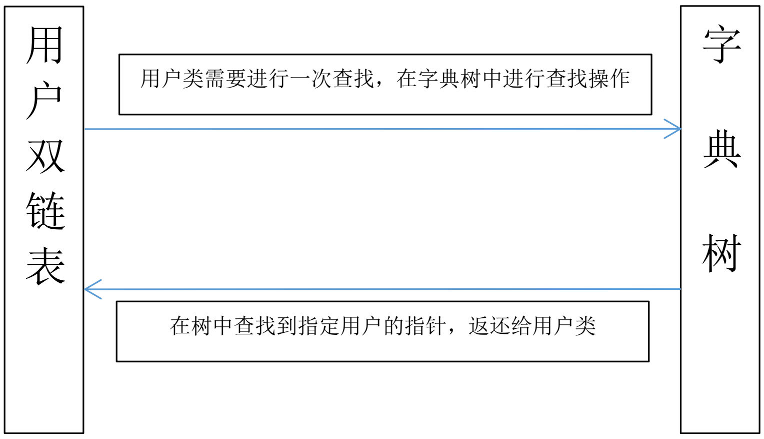
首先要先进行一次查找，以确保此用户名存在。

之后，步骤与查找类似，但在每一步中都将匹配到的结点的计数

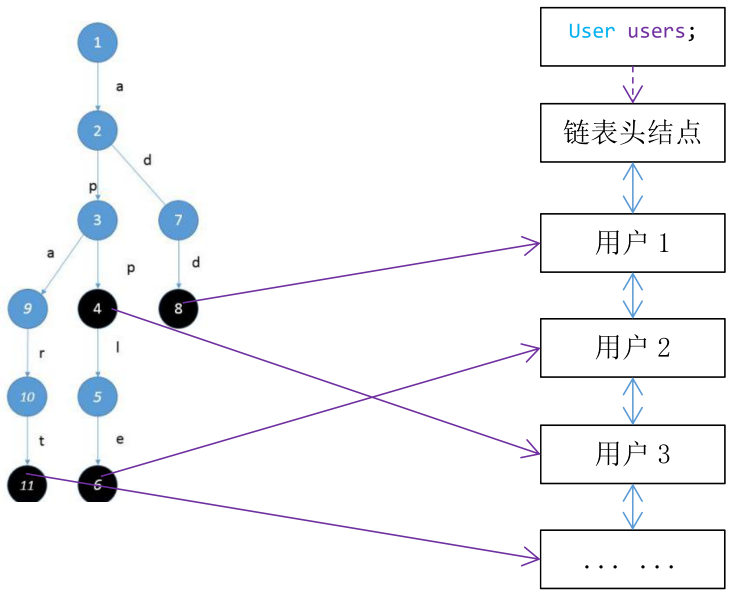
位减 1（若计数位减到 0 则删除这棵子树），当匹配到最后一位时，将有效位置为无效。

d. 与 User 类的联系：

Trie 树中的每一个结点都有一个数据域，当有效位为有效时，数据域存储的是指向该用户的指针（即 User 类型的指针），以便在查找到该用户后将操作权转交给 User 类。两者的关系如下：


用户双链表
字典树
用户类需要进行一次查找，在字典树中进行查找操作
在树中查找到指定用户的指针，返还给用户类

这样 Tire 树就实现了，只需把 Trie 树用户名字符串末尾节点（成为“有效节点”）的指针指向双链表对应位置就可以了。形式如下：



构建好了 Trie 树和双链表，这样功能就可以使用以下函数实现了：

a. 添加用户：

```
public boolean insert(String name, String password)
```

按照指定的用户名 name（不可重复）和密码 password，创建一个新的用户节点并用头插法加入到链表中。注意：

①用户名不可重复，重复时添加失败，返回 false；

②用户名必须为小写字母和数字的组合；

③采用头插法创建链表。

b. 删除用户：

```
    public boolean erase(String name)
```

按照指定的用户名 name，删除链表中的一个用户节点。注意：

①如果没有此用户，则删除失败；

②如果该用户存在，但其借书记录不为空，则也不可删除。

c. 查找用户：

```
    public Node find(String name)
```

按照指定的用户名 name，在链表中找到该用户并返回指向其结点的指针。注意：

①如果没有此用户，则返回空指针 null；

②查找算法在 Trie 树中实现，User 链表知识调用了其函数。

d. 匹配用户：

```
    public boolean match(String name, String password)
```

按照指定的用户名 name 和密码 password，在链表中找到该用户并检查其密码是否匹配。注意：

①如果没有此用户，则匹配失败；

②如果有此用户，但密码错我，则也匹配失败。

e. 更改密码：

```
    public boolean changePassword(String name, String newPassword)
```

按照指定的用户名 name，在链表中找到该用户并将其密码域修改为 newPassword。注意：如果没有此用户，则更改失败。

f. 借书：

```
    public boolean borrowBook(String name, String title)
```

按照指定的用户名 name，在链表中找到该用户并在其借书记录中按给定的书名 title 新增一个记录。注意：

①如果没有此用户，则借书失败；

②如果有此用户，但该书已在借书记录中，则也失败。

g. 还书：

```
    public boolean returnBook(String name, String title)
```

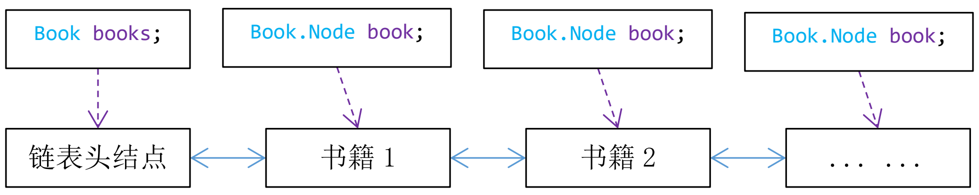
按照指定的用户名 name，在链表中找到该用户并在其借书记录中按给定的书名 title 删除一个记录。注意：

①如果没有此用户，则还书失败；

②如果有此用户，但该书已在借书记录中，则也失败。

Book 类:为了能支持书籍的查询、新增、更新、删除等书籍管理操作而设计的类。我们为储存书籍信息和功能实现分别建立了两种数据结构——存储书籍信息用的双链表而查询等功能实现则利用了伸展树 SplayTree。

(1)储存书籍信息：储存书籍的数据结构是一个带头结点的双链表，每一个书籍是一个结点，该结点包含该书籍包括名称、编号、作者在内的所有信息，结点相连形成双链表。其中链表的类型为 Book，用户结点的类型为 Book.Node。



(2)查询等功能实现：同样如果选择在双链表暴力查找，假设有 N 本书则时间复杂度达到了 O（N）。维护一棵 Splay 树，在树上按或是编号或是书名或是作者名称建立大小关系，每一个节点代表一本书，指针指向该书在双链表的位置，这样的查找是 O（logN）的。我们没有选择平衡树而选择伸展树是因为伸展树会将使用频率高的节点旋转到上面，这样更符合书籍查询的效率。那么什么是伸展树呢？

• 伸展树：

I. 简介：

伸展树（Splay Tree），也叫分裂树，是一种二叉排序树，它能在 O(log n)内完成插入、查找和删除操作。

在伸展树上的一般操作都基于伸展操作：假设想要对一个二叉查找树执行一系列的查找操作，为了使整个查找时间更小，被查频率高的那些条目就应当经常处于靠近树根的位置。于是想到设计一个简单方法，在每次查找之后对树进行重构，把被查找的条目搬移到离树根近一些的地方。伸展树应运而生。伸展树是一种自调整形式的二叉查找树，它会沿着从某个节点到树根之间的路径，通过一系列的旋转把这个节点搬移到树根去。

它的优势在于不需要记录用于平衡树的冗余信息。

II. 原理：

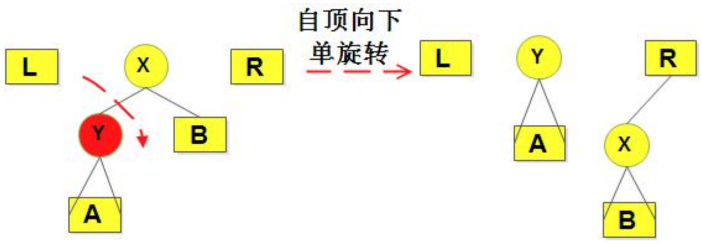
伸展树有两种实现方式：一种是自底向上伸展，即先找到目标结点，再将其一层一层旋转至根节点处；另一种是自顶向下伸展，本实验即采用这种实现，因为考虑到进行模糊查找时，可能找不到目标结点（如想找“app”，但树中只有“apple”），此时我们仍希望将其视为想要查找到的结果，并能将“apple”旋转至根节点（如果不这么做，那么每次查找都必须完整地输入书名，用户交互就会很不友好）。基于这种需求，使用自顶向下的方式更容易实现。

自顶向下伸展：边查询边旋转，在访问路径上的一次遍历就可完成，效率上要比自低向上的方式高。要想在访问路径上一次遍历并完成伸展，需要开辟额外子树用于保存遍历过程中的信息。

在伸展操作的过程中，当前遍历结点 X 是树的中树，左树 L 保留小于 X 的结点，右树 R 保留大于 X 的结点。树开始遍历时候 X 是树的根 T，而 L 和 R 是空树。

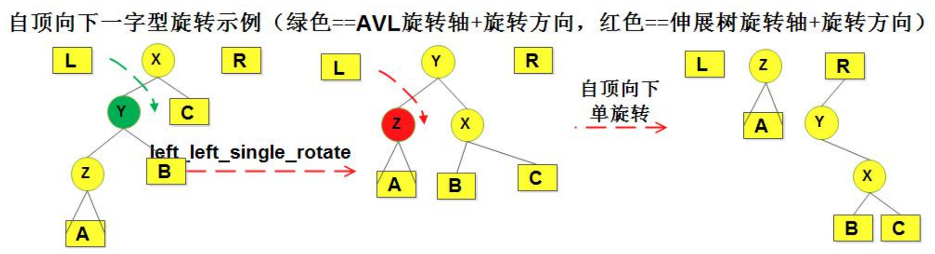
有三种旋转模式：

①单旋转（Zig）



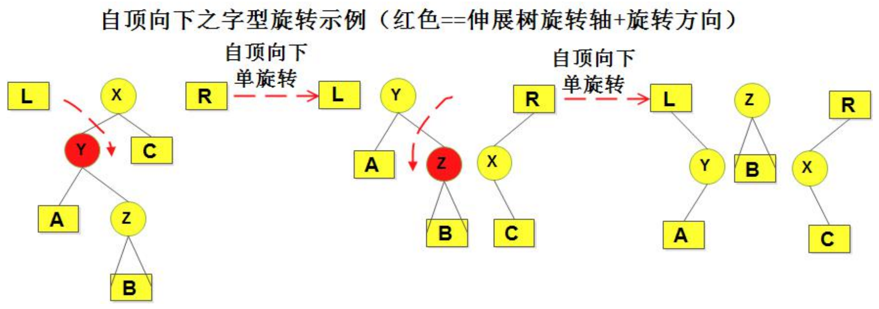
在这种情况下，要查找的结点比当前结点 X 小，但正好是 X 的子结点 Y。此时只需进行简单的单旋，将 Y 变成新的中树的树根，而 X 连接到右树上

②一字形旋转（Zig-Zig）


自顶向下一字型旋转示例（绿色==AVL旋转轴+旋转方向，红色==伸展树旋转轴+旋转方向）
L
X
R
Y
C
Z
B
A
left_left_single_rotate
自顶向下
单旋转

在这种情况下，要查找的结点比当前结点 X 小，而且比 X 的左子树 Y 还小。此时首先将 Y 绕 X 右旋，然后将 Z 变成新的中树根节点。将 Y 及其子树移到右树中。（因为这些结点都一定比目标结点要大）（此外需要注意，挂载到右树时只能挂载到 R 的右子树上，同理，挂载到左树时只能挂载到 L 的右子树上）

③之字形旋转（Zig-Zag）

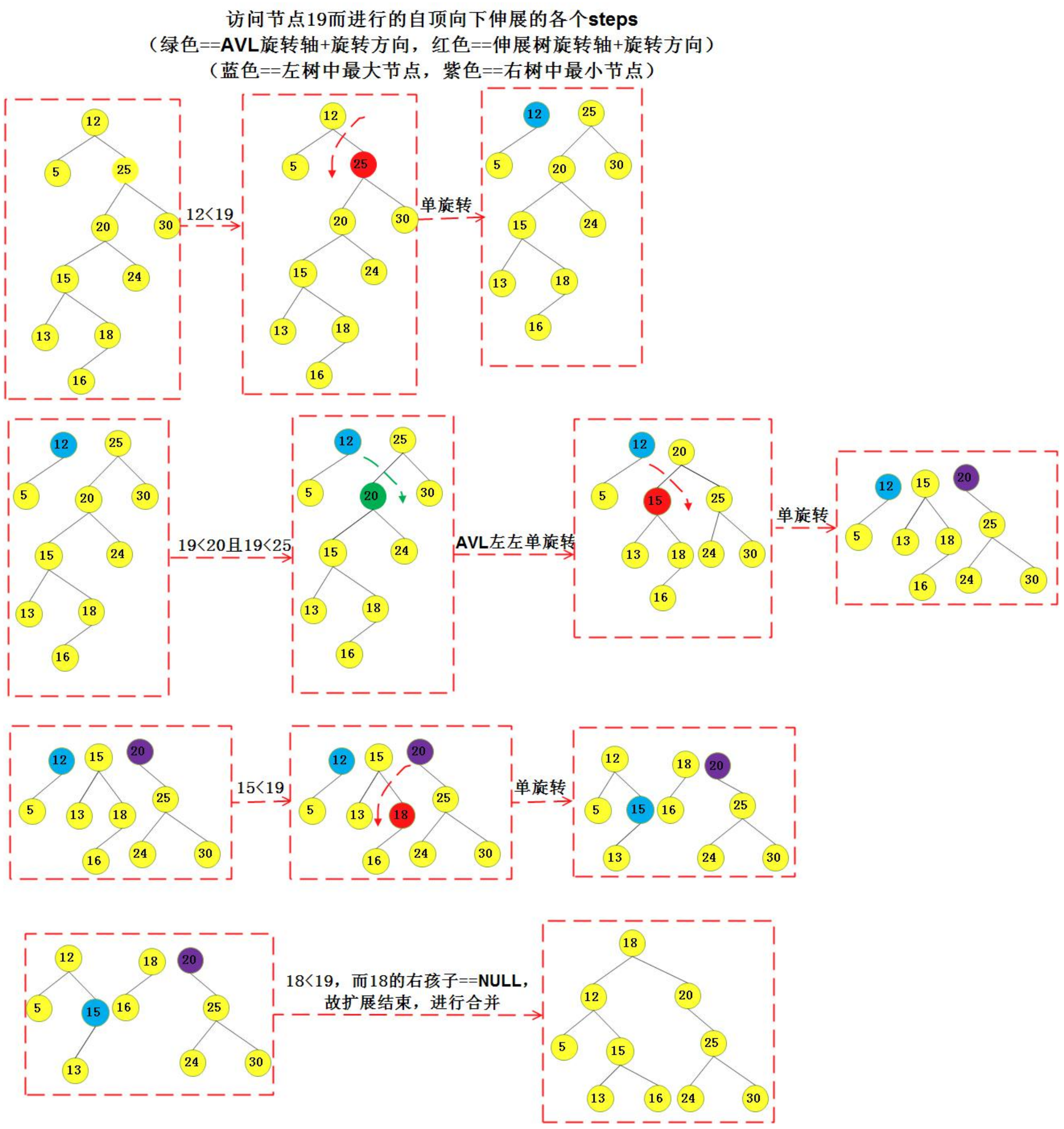


在这种情况下，要查找的结点比当前结点 X 小，但比 X 的左子树 Y 大。此时先将 Y 右旋到根，将 X 及其子树挂载到右树 R 上，变成上图中间的 Zag 情况，接下来对 Z 进行左旋变成 Zig 情况，将父结点 Y 链接到左树 L 上 。

④合并

所查找的结点 X 已经找到，需要将 X，L 和 R 合并，X 是合并新树的树根，L 是比 X 结点小的结点的集合，R 是比 X 大的结点的集合。如果 X 有左子树和右子树的话，则 L.right = X.left, R.left = X.right

下图是一个完整例子：


访问节点19而进行的自顶向下伸展的各个steps
（绿色==AVL旋转轴+旋转方向，红色==伸展树旋转轴+旋转方向）
（蓝色==左树中最大节点，紫色==右树中最小节点）
12<19
单旋转
19<20且19<25
AVL左左单旋转
单旋转
15<19
单旋转
18<19，而18的右孩子==NULL，
故扩展结束，进行合并

III.实现：

查找：按指定的主键在树中查找结点并返回其数据域，若找不到则返回空指针。

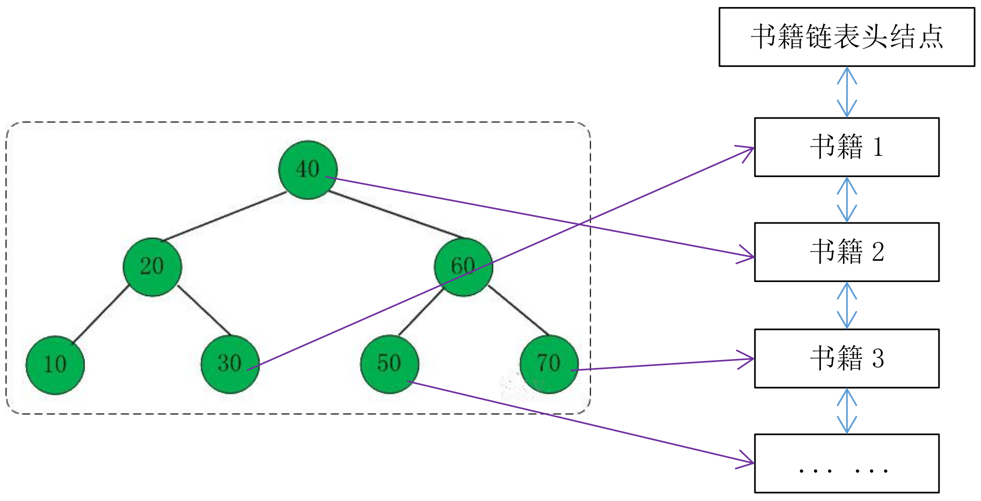
模糊查找：若精准查找失败则返回距离其最近的结点的数据域。

删除：先用查找算法将目标结点旋转到根节点，再进行删除。

插入：与查找算法相似，将新节点插入到最后查找失败处。

旋转：将指定结点旋转到根节点处，其算法已在原理一节中描述过。

类似 User 类，Splay 树上的节点也指向对应书籍双链表中的对应结点即可。假设是一棵编号构成的 Splay 树，则与书籍双链表的对应关系如下：



需要分别以编号、书名和作者为关键字构建三棵 splay 树，每棵 splay 树都像上面这样连接（其实作者的 Splay 树指向的是一个该作者写的书的链表，这个链表依次指向了对应的书籍双链表节点，其他的两棵树也是这样，但由于相同编号和书名不会有很多本，所以中间的“链表”只有一个节点，这个节点指向了对应编号或书名的书籍双链表节点）。构建好了 Splay 树和双链表，这样功能就可以使用以下函数实现了：

a. 添加书籍：

```
    public boolean insert(String title, String author, Integer category, int totalQuantiy)
```

按指定的书名、作者、类别、总量，新建一本书。注意：

①书名不可重复；

②作者、类别可以重复；

③书籍编号自动初始化为按当前链表大小；

④书籍借出量初始化为 0；

⑤按头插法创建链表。

b. 删除书籍：

```
    public boolean erase(String title)
```

注意：若该书的借出量不为 0，则无法删除。

c. 查找书籍：

```
    public SplayTree<Integer, Node>.Node.DataNode findById(Integer id)
    public SplayTree<String, Node>.Node.DataNode findByTitle(String title)
    public SplayTree<String, Node>.Node.DataNode findByTitleFuzzy(String title)
    public SplayTree<String, Node>.Node.DataNode findByAuthor(String author)
```

有四种查找书籍的方法

①按书名查找是模糊匹配，并且只能返回一本书；

②按编号查找是精准匹配，并且只能返回一本书；

③按作者查找是精准匹配，可以返回多本书；

④按类别查找是精准匹配，可以返回多本书。

d. 书籍借出：

```
    public boolean changeBorrowedQuantity(String title, int increment)
```

e. 书籍还入：

```
    public boolean changeBorrowedQuantity(String title, int increment)
```

f. 书籍增订：

```
    public boolean changeTotalQuantity(String title, int increment)
```

2.2.6 算法及实现设计

指令分析的实现：我们规定一条来自客户端指令的模式如下：

对话型：“[消息类型]//[来源端口]//[来源用户名]//[消息内容]”

任务型：“[操作类型]//[方式]//[内容]”

即不同涵义字段以双斜杠“//”进行分割，在实现的时候使用.split()对字符串进行分割即可。规定指令信息表如下：

对话型：

消息类型	来源端口	来源用户名	消息内容	功能
Message	来源端口	来源用户名	任意内容	发送消息给其他用户
OnlineNumber	来源端口	来源用户名	数字	查询在线人数

任务型：

操作	端口号	方式	内容字串 1	内容字串 2	内容字串 3	功能
Insert_B	端口号	0	书籍名称	书籍数目	0	增加相应数目书籍
	端口号	1	书籍名称	书籍数目	0	减少相应数目书籍
	端口号	2	书籍名称	作者	书籍数目//类别	新增书籍
Query_B	端口号	0	书籍编号	用户名	0	以编号查询书籍
	端口号	1	书名	用户名	0	以书名查询书籍
	端口号	2	作者	用户名	0	以作者查询书籍
	端口号	3	用户名	0	0	用户已借书籍返回
Delete_B	端口号	0	书籍名称	0	0	以编号删除书籍
Register_S	端口号	0	用户名	密码	0	用户注册
Delete_S	端口号	0	用户名	0	0	删除用户
Login_S	端口号	0	用户名	密码	0	查询用户密码
Borrow_S	端口号	0	用户名	书籍名称	0	用户借阅书籍
Return_S	端口号	0	用户名	书籍名称	0	用户归还书籍
Modify_S	端口号	0	用户名	原密码	新密码	更换密码
Query_B_next	端口号	1	书名	用户名	断开处书名	继续书名查询
	端口号	2	断开处书名	用户名	作者	继续作者查询

服务器收到客户端消息后进行相应功能实现。下面的实现中会介绍比较重点的功能实现，对于在上一节数据结构的设计中已经提供了直接的接口的实现不再赘述。

接入客户端的实现：接入客户端的实现用 Socket 套接字接口，服务器通过 Socket 进行多线程监听是否有客户端联接，如果联接成功，则将该客户端的地

址加入接入列表，并在代码中循环接收是否有消息流传入。客户端断开联接则将客户端地址从接入列表中删除。

```
// 进入等待环节
System.out.println("等待手机的连接... ... ");
final Socket socket = server.accept();
// 获取手机连接的地址及端口号
final String address = socket.getRemoteSocketAddress().toString();
System.out.println("连接成功，连接的手机为：" + address);
……
clientList.put(address,socket);
```

书籍和用户数据的读取和更新：书籍数据是利用爬虫在豆瓣上抓下来的（在后面会有介绍），用 csv 格式保存。在启动服务器管理的时候读取书籍和用户的 csv 文件（注意 csv 是以逗号作为分隔），逐一插入到 splay 树和 Trie 树中即可。下面的代码以读取书籍为例：

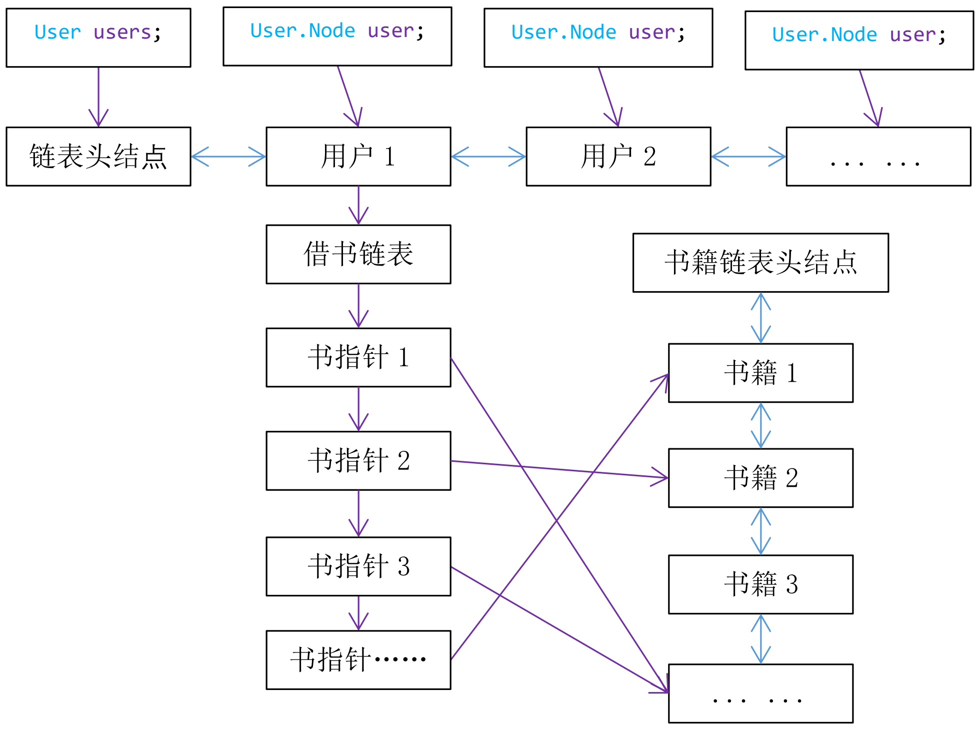
```
public static void readBooks() {
    BufferedReader br = null;
    try
    {
        File csv = new
File("C:\\Creatures\\Android\\lib\\src\\main\\java\\com\\creatures\\siang\\sever\\BookList.csv");  //
CSV 文件路径
        br = new BufferedReader(new FileReader(csv));
    } catch (FileNotFoundException e)
    {
        e.printStackTrace();
    }
    String line = "";
    String everyLine = "";
    try {
        int count = 0;
        while ((line = br.readLine()) != null)  //读取到的内容给 line 变量
        {
            everyLine = line;
            //System.out.println(count+ everyLine);
            String[] split = ((String) everyLine).split(",");
            boolean can = books.insert(split[0].trim().replace("\"", ""), split[1].trim().replace("\"",
"").replace("?", "·"), 0, Integer.parseInt(split[2]));
            if (can) {
                Book.Node theBook = books.findByTitle(split[0].trim().replace("\"", "")).next.datum;
                theBook.changeBorrowedQuantity(Integer.parseInt(split[3]));
                count ++;
            }
        }
        System.out.println("csv 表格中所有行数："+count);
    } catch (IOException e)
    {
        e.printStackTrace();
    }
}
```

在更新书籍和用户数据的时候，书籍每隔 5 分钟重写一次 csv 文件，用户每隔 6 分钟重写一次 csv 文件。注意存用户的时候要把所有借了的书籍都记录进去。下面以存储用户为例：

```
public static void saveUser() {
    final long timeInterval = 360000;// 6 分钟运行一次
    Runnable runnable = new Runnable() {
        public void run() {
            while (true) {
                try {
                    FileOutputStream outFile = new
FileOutputStream("C:\\Creatures\\Android\\lib\\src\\main\\java\\com\\creatures\\siang\\sever\\UserLis
t.csv");//写出的 CSV 文件
                    BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(outFile, "UTF-8"));
                    User.Node node = users.getHead().next;
                    while (node != null) {
                        String text = node.name + "," + node.password + ",";
                        String temp = "";
                        int count = 0;
                        SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
                        List<User.Node.Record> borrowRecord = users.getAllRecords(node.name);
                        if (borrowRecord != null) {
                            for (int i = 0; i < borrowRecord.size(); i++) {
                                temp = temp + ',' + borrowRecord.get(i).title + ',' +
sdf.format(borrowRecord.get(i).date);
                                count ++;
                            }
                        }
                        text = text + String.valueOf(count) + temp;
                        writer.write(text);
                        writer.newLine();
                        node = node.next;
                    }
                    writer.close();
                }  catch (IOException e)
                {
                    e.printStackTrace();
                }
                try {
                    Thread.sleep(timeInterval);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    };
    Thread thread = new Thread(runnable);
    thread.start();
}
```

这样便有效防止了因服务器崩溃或重启而导致的数据全部丢失。同时也方便了手动加入书籍到文件以后重新读取数据（比如爬虫到的书籍可以放入 csv 重启服务器读取）。

用户借阅书籍的查询：在用上面的 Trie 树找到双链表上的用户以后，事实上双链表中的用户结点中有一个链表元素，这个链表是一串指针链表，链表中的元素指向某本书籍，表示当前用户借了这本书。这样用户借阅的书籍就可以很快查询完了。借书还书只需在借书链表上添加和删除即可。



下面是针对查询我的书籍的代码：

```
public static void sendQueryStudentBook(String username, Socket socket){
    List<User.Node.Record> borrowRecord = users.getAllRecords(username);
    try {
        OutputStream outputStream = socket.getOutputStream();
        String text;
        if (borrowRecord == null){
           text = "Error_Student";
        }
        else {
            String temp = "";
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
            for (int i = 0; i < borrowRecord.size(); i++) {
                Book.Node theBook = books.findByTitle(borrowRecord.get(i).title).next.datum;
                temp = temp + "//" + String.valueOf(theBook.id) + "//" + theBook.title + "//" +
theBook.author + "//" + sdf.format(borrowRecord.get(i).date) + "//" +
sdf.format(getEndDate(borrowRecord.get(i).date));
            }
            text = "Book" + "//1//" + username  + "//" + String.valueOf(borrowRecord.size()) + temp;
        }
        outputStream.write(text.getBytes("utf-8"));
        outputStream.flush();
    }catch (Exception e){
        e.printStackTrace();
    }
}
```

编号书名和作者的查询：对于用编号书名和作者查询书籍，在下面客户端的实现设计中会介绍到需要返回用户的借阅书籍（因为显示标签不一样）。所以首先查询当前用户借阅的书籍。然后再在 splay 树上查找。在 Book 类中需要分别以编号、书名和作者为关键字构建三棵 splay 树，按要求在树上找到返回即可。

注意作者为关键字的时候，一个作者可能有很多本书，所以节点的数据并不是像上面所说的直接指向书籍双链表而是有一个链表记录了所有该作者的书的指针，这些指针指向了书的双链表，就像用户借阅书籍查询中构建的那样。下面的代码是按照作者的 splay 树找的：

```
public static void sendQueryBookAuthor(String author, String username, Socket socket){
    SplayTree<String, Book.Node>.Node.DataNode can = books.findByAuthor(author);
    List<User.Node.Record> borrowRecord = users.getAllRecords(username);
    ……
        SplayTree<String, Book.Node>.Node.DataNode theBook = can.next;";
            ……
                for (int i = 0; i < borrowRecord.size(); i++) {  //找用户借的书籍
                    Book.Node theUserBook = books.findByTitle(borrowRecord.get(i).title).next.datum;
                    tempu = tempu + "//" + String.valueOf(theUserBook.id) + "//" + theUserBook.title + "//"
+ theUserBook.author + "//" + sdf.format(borrowRecord.get(i).date) + "//" +
sdf.format(getEndDate(borrowRecord.get(i).date));
                }
            String text, temp = "";
            int count = 0;
            while (theBook != null) {          //找要查询的书籍
                if (count >= 20)
                    break;
                count += 1;
                temp = temp + "//" + String.valueOf(theBook.datum.id) + "//" + theBook.datum.title + "//"
+ theBook.datum.author + "//" + String.valueOf(theBook.datum.totalQuantity) + "//" +
String.valueOf(theBook.datum.borrowedQuantity);
                theBook = theBook.next;
            }
            text = "Book" + "//" + "0"  + tempu + "//" + String.valueOf(count) + temp;
           ……
    ……
}
```

书名的模糊匹配查询：如果像 Splay 树中那样找到目标返回目标书籍，找不到目标就返回无效，即必须输入完全正确的书名才可以找到书籍，这样未免太不智能。我们为模糊匹配设计了一套算法：

①首先 Splay 树提供模糊查找接口，即如果树上找不到就返回最相近的那个书籍，这个在刚刚的数据结构中已经提到了。

②在新增书籍的时候，不在书籍双链表上直接添加，而是通过 splay 树找到最相近的那本书的位置，比较判断后把新书插入到这本书的前面或后面。这样就维护了双链表按书名有序。

③当查询到最近似书籍以后，从该书籍的位置开始，向双链表该位置的前面和后面遍历，如果满足遍历到的书籍和要查询的书名的误差不超过一定范围指标σ，我们就把这本书加入到需要返回给客户端的书单，否则停止遍历，因为之后的书籍由于双链表是有序的所以偏差更大。

那么我们如何评判误差指标σ呢？我们采取了这样的公式。设查询的书名为 N，遍历到的书籍书名为 M，设两个字符串的最长公共前缀为 maxPrefix(N,M)，设字符串长度为 L(M)。我们的思路是只要书名的最长公共前缀超过两个书名中最长的长度的 1/3 或者 M，N 之间有一个是另一个的前缀，我们就认为这样的误差是允许的。即，

$$\mathrm{maxPrefix}(N,M) >= \frac{1}{3}Max\{L(N),L(M)\},$$
$$\mathrm{maxPrefix}(N,M) == N, \max\mathrm{Pr}\,efix(N,M) == M$$

满足这个条件就判断为可行，这样就可以模糊匹配了。算法代码实现如下：

```
public static void sendQueryBookName(String title, String username, Socket socket){
……
    String temp = "";
    int count = 0;
    if (theBook.title.compareTo(title) >= 0) {
        int prefix_or = maxPrefix(theBook.title, title);
        if (prefix_or > theBook.title.length() / 3 || prefix_or == title.length()) {
            temp = temp + "//" + String.valueOf(theBook.id) + "//" + theBook.title + "//" + theBook.author
    + "//" + String.valueOf(theBook.totalQuantity) + "//" + String.valueOf(theBook.borrowedQuantity);
            count++;
        }
    }
    Book.Node currentBook;
    currentBook = theBook.next;    //双链表向后查找
    while (true) {
        if (count >= 20) break;
        if (currentBook == null) {
            break;
        }
        int len = currentBook.title.length() / 3;
        int prefix = maxPrefix(currentBook.title, title);
        if (prefix > len || prefix == title.length()) {            //判断条件
            temp = temp + "//" + String.valueOf(currentBook.id) + "//" + currentBook.title + "//" +
    currentBook.author + "//" + String.valueOf(currentBook.totalQuantity) + "//" +
    String.valueOf(currentBook.borrowedQuantity);
            count++;
        }
        else {
            break;
        }
        currentBook = currentBook.next;
    }

    if (theBook.title.compareTo(title) < 0) {
        int prefix_or = maxPrefix(theBook.title, title);
        if (prefix_or > title.length() / 3 || prefix_or == theBook.title.length()) {
            temp = temp + "//" + String.valueOf(theBook.id) + "//" + theBook.title + "//" + theBook.author
    + "//" + String.valueOf(theBook.totalQuantity) + "//" + String.valueOf(theBook.borrowedQuantity);
            count++;
        }
    }
```

```
        currentBook = theBook.pre;              //双链表向前查找
        while (true) {
            if (count >= 20) break;
            if (currentBook.pre == null) {
                break;
            }
            int len = title.length() / 3;
            int prefix = maxPrefix(currentBook.title, title);
            if (prefix > len || prefix == currentBook.title.length()) {
                temp = temp + "//" + String.valueOf(currentBook.id) + "//" + currentBook.title + "//" +
        currentBook.author + "//" + String.valueOf(currentBook.totalQuantity) + "//" +
        String.valueOf(currentBook.borrowedQuantity);
                count++;
            }
            else {
                break;
            }
            currentBook = currentBook.pre;
        }
    ……
    }
```

继续书名和继续作者的查询：在客户端的设计中会介绍到，由于 Socket 一次发送信息的大小是有限制的，当查询得到的书籍过多时，书籍信息会从中间截断，如此会导致字符串的分割不全而错误，导致程序崩溃。

……编号//书名//作者//数目//借阅数目编号//书名//作者//（此处被截断）

为了解决这一问题我们选择从断开处多次查询。如何实现从断开处查找呢？

实现的方法很暴力，就是按照原来的书名或作者重新查找，但是只返回从断掉的地方以后的书籍给客户端。下面的代码是以书名为例：

```
    public static void sendQueryBookNameNext(String title, String username, String keyTitle, Socket socket){
        ……
        Book.Node currentBook;
        currentBook = theBook.next;
        while (true) {
            if (count >= 20) break;
            if (currentBook == null) {
                break;
            }
            int len = currentBook.title.length() / 3;
            int prefix = maxPrefix(currentBook.title, title);
            if (prefix > len || prefix == title.length()) {
                if (key == 1) {                         //必须要在断开书籍之后 key 才为 1
                    temp = temp + "//" + String.valueOf(currentBook.id) + "//" + currentBook.title + "//"
        + currentBook.author + "//" + String.valueOf(currentBook.totalQuantity) + "//" +
        String.valueOf(currentBook.borrowedQuantity);
                    count++;
                }
                if (currentBook.title.equals(keyTitle)) key = 1;//找到了断开书籍
            }
            else {
                break;
            }
            currentBook = currentBook.next;
        }
        ……
    }
```

2.3 客户端实验方案

2.3.1 功能要求

客户端要求可以向服务器发送注册、登陆、查询等请求。要求实现简约美观的页面，显示美观合理的书籍查询、用户查询、聊天水缸的操作和查看页面。实现权限用户和普通用户的分开。能将服务器的提示以 Toast 方式返回。实现管理书籍（允许查询并删除、更新特定书籍）、新增书籍（新建书籍）、查询书籍（允许查询并借阅或归还某本书籍）、查询用户（允许查询并删除特定用户）、我的书籍（允许查询我借的书籍）、聊天水缸（提供交流平台）共六大模块。

2.3.2 艺术设计

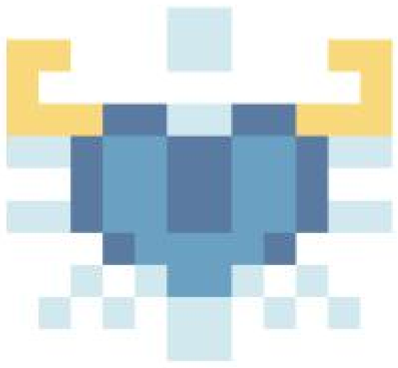


图 1 “白泽书房”APP 图标

这次的设计风格采用蓝色作为主调，并用马赛克风格进行设计。艺术的灵感来源源自古代神兽白泽。白泽的形象是根据《白泽图》的“羊有一角当顶上，龙也”抽象而来的。

据《云笈七签·轩辕本纪》记载，“帝巡狩，东至海，登桓山，于海滨得白泽神兽。能言，达于万物之情。因问天下鬼神之事，自古精气为物、游魂为变者凡万一千五百二十种。白泽言之，帝令以图写之，以示天下。帝乃作祝邪之文以祝之。”

我们 APP 的名字遍来源于此，白泽上知天文下知地理，可谓无所不知、博学多识，我们希望我们的图书馆的功能也像白泽一样能问知世间万物。设计软件是用的 dotpict 绘制。

2.3.3 资源排布

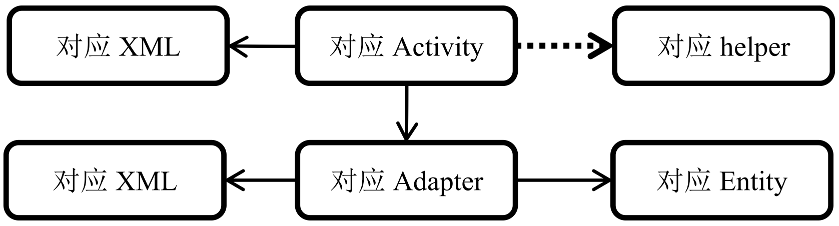
目录和文件（未列入的为 IDE 自建文件，无需手动配置）			功能	
res	drawable	bg_et.xml	定义输入框样式	图形资源
		bg_message.xml	定义消息样式	
		bg_sent.xml	定义发送按钮	
		ic_luacher.png	应用图标	
	layout	activity_add_book.xml	定义添加书籍页面样式	布局资源
		activity_admin.xml	定义管理者用户页面样式	
		activity_change_password.xml	定义更换密码页面样式	
		activity_chatroom.xml	定义聊天水缸页面样式	
		activity_login.xml	定义登陆页面样式	
		activity_manage_book.xml	定义管理书籍页面样式	
		activity_mybook.xml	定义我的书籍页面样式	
		activity_normal.xml	定义普通用户页面样式	
		activity_register.xml	定义注册页面样式	
		activity_search_book.xml	定义查询书籍页面样式	
		activity_search_student.xml	定义查询用户页面样式	
		book_list_item.xml	定义我的书籍页面样式	
		manager_book_list_item.xml	定义管理书籍列表样式	
		message_item.xml	定义消息显示样式	
		mybook_list_item.xml	定义我的书籍列表样式	
	menu	options_menu.xml	定义聊天水缸顶菜单栏样式	菜单资源
	values	color.xml	定义软件配色	取值资源
		styles.xml	定义主题风格	
		strings.xml	定义字段（英文）	
		strings.xml(zh-rCN)	定义字段（中文）	
	AndroidManifest.xml		定义应用程序	应用资源

2.3.4 程序排布

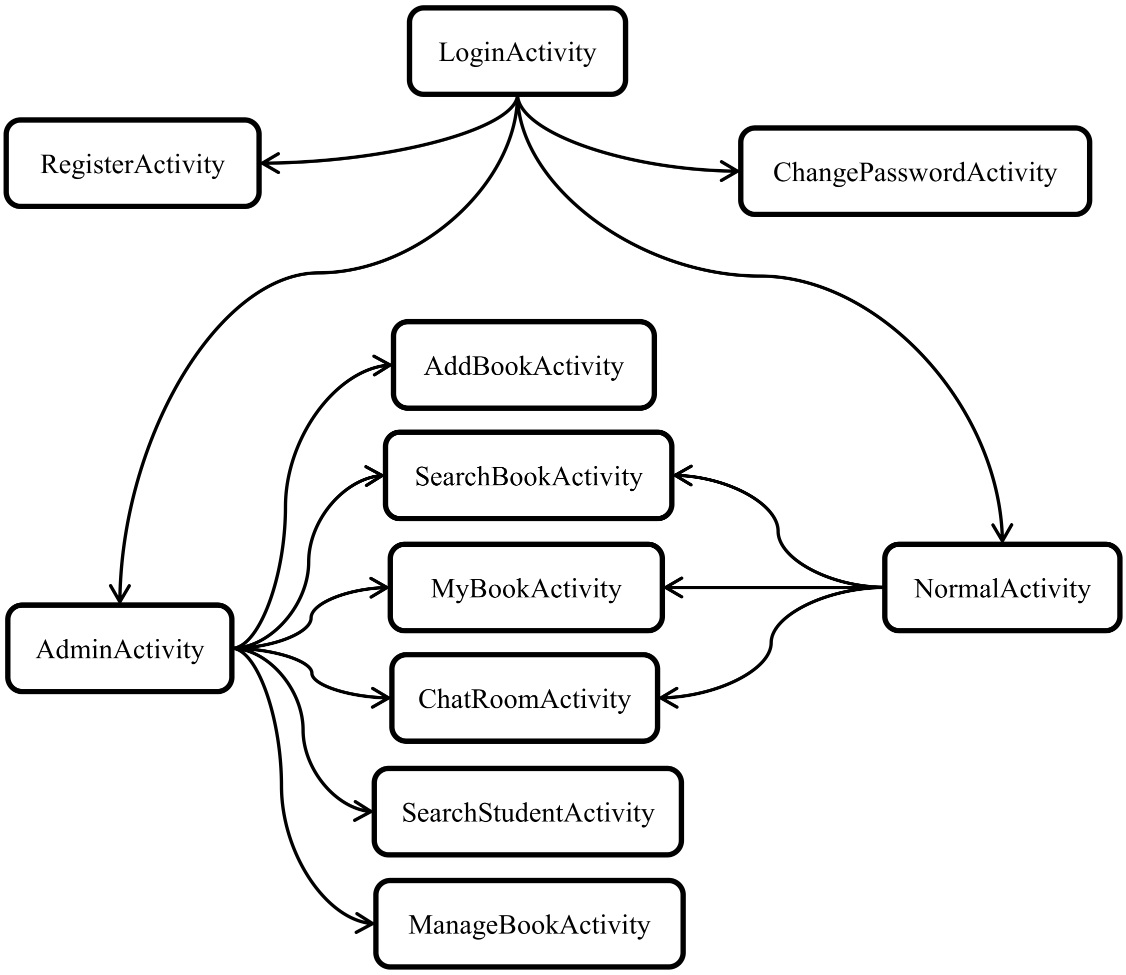
目录和文件（未列入的为 IDE 自建文件，无需手动配置）			功能	
com.siang.pc.librarysystem	activity	AddBookActivity.java	新增书籍界面实现	Activity 的实现
		AdminActivity.java	管理者界面实现	
		ChangePasswordActivity.java	更换密码界面实现	
		ChatRoomActivity.java	聊天水缸界面实现	
		LoginActivity.java	登陆界面实现	
		ManagerBookActivity.java	管理书籍界面实现	
		MyBookActivity.java	我的书籍界面实现	
		NormalActivity.java	普通用户界面实现	
		RegisterActivity.java	注册界面实现	
		SearchBookActivity.java	查询书籍界面实现	
		SearchStudentActivity.java	查询用户界面实现	
	adapter	BookListAdapter.java	书籍查询 RecyclerView 的适配器	Adapter 的实现
		ManagerBookListAdapter.java	管理书籍查询 RecyclerView 的适配器	
		MessageAdapter.java	聊天水缸 RecyclerView 的适配器	
		MyBookListAdapter.java	我的书籍查询 RecyclerView 的适配器	
	entity	BookInfo.java	书籍实体	实体定义
		ChatMessage.java	聊天消息实体	
		MyBookInfo.java	我的书籍实体	
	helper	ChatRecordSQLiteOpenHelper.java	记录聊天的 SQLite 连接助手	助手类

2.3.5 类之间的调用关系

对于 activity 与其他类型的类之间的调用，大致按照功能符合如下所示，对于每一个模块都是这样，由于关系较多不一一展示：



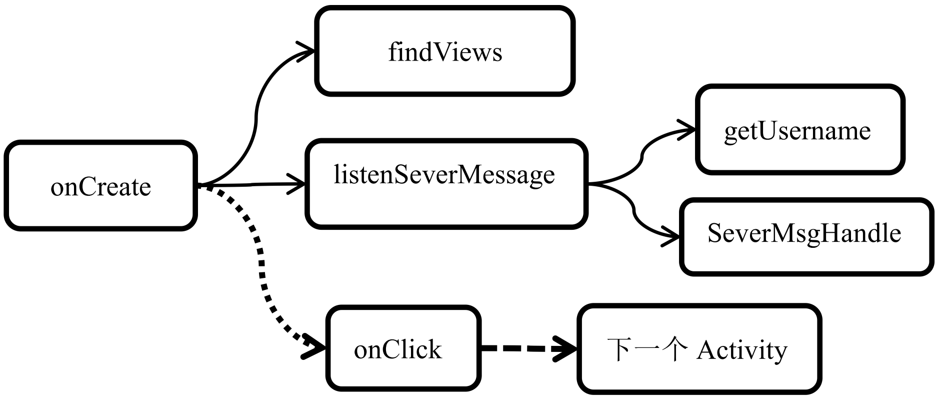
而对于 Activity 中类的调用关系则是按照程序流程来的，和用户体验思路一致，大致如下：



其中 APP 从 LoginActivity 开始，根据选择按钮触发的事件不同决定进入不同的 Activity。

2.3.6 函数之间的调用关系

对于非 Activity 的函数往往是做成接口供其他函数调用，而且过于庞杂，所以这里只简单地展示 Activity 中非生命周期函数之间的调用关系：



其 中 onCreate 为 Activity 的 生 命 周 期 ， findView 为 找 到 对 应 布 局 ， listenSeverMessage 用 于 监 听 服 务 器 的 消 息 ， getUsername 为 当 前 用 户 名 称 。 SeverMsgHandle 是一个子类，因为接受服务器的线程里的信息想要传到主线程里来必须构造这样的 Handler 子类负责传递。onClick 是点击事件触发，一个 Activity 中

可能有多个点击事件，它们的名称是不相同的，点击事件要么根据上面的流程图进入下一个 Activity，要么向服务器发送对应请求。

2.3.7 数据结构设计

BookInfo 类:为了能方便存储查询书籍的名称、编号、作者、库存、借阅量、当前可否借阅状态的信息而构建的类。这个类提供了上述信息查询和修改的接口。

```
public class BookInfo {
    private String Id;
    private String name;
    private String author;
    private String bookHave;
    private String bookBorrow;
    private int type;               //0 借阅 1 归还 2 不可操作

    public int getType() { return type;}
    public void setType(int type) { this.type = type;}
    ……
}
```

MyBookInfo 类:对于查询我的借阅书籍时，应存入书籍的名称、编号、作者、开始借阅时间、开始截至时间的信息，与上面的 BookInfo 类有所区别。

```
public class MyBookInfo {
    private String Id;
    private String name;
    private String author;
    private String startTime;
    private String endTime;
    ……
}
```

ChatMessage 类:在聊天水缸中对于一次对话需要知道用户名称、消息内容】消息来源，所以构造这个类。

```
public class ChatMessage {
    public static final int TYPE_RECEIVED = 0;  // 接收消息
    public static final int TYPE_SENT = 1;      // 发送消息

    private String username;                     // 消息用户
    private String content;                     // 消息内容
    private int type;                           // 消息来源
    ……
}
```

各种 adapter 类：对于各种 RecyclerView 的适配器而言，必须把信息用 List 类型保存，并且提供当前 Socket 号、用户名、查询方式和查询内容的修改和访问接口。以 BookListAdapter 为例：

```
public class BookListAdapter extends RecyclerView.Adapter<BookListAdapter.ViewHolder> {
    private List<BookInfo> bookList;        //view 的数据来源
    private Socket socket;
    private Context context;
    private String username;
    private String way = "2";
    private String search = "";
    ……
}
```

2.3.8 算法及实现设计

指令分析的实现：我们规定一条来自服务器指令的模式如下：

对话型：“[消息类型]//[来源端口]//[来源用户名]//[消息内容]”

任务型：“[操作类型]//[方式]//[内容]”

即不同涵义字段以双斜杠“//”进行分割，在实现的时候使用.split()对字符串进行分割即可。规定指令信息表如下：

对话型：

消息类型	来源端口	来源用户名	消息内容	功能
Message	来源端口	来源用户名	任意内容	显示消息，来自自己的在右侧，来自他人的在左侧
OnlineNumber	来源端口	来源用户名	数字	用 Toast 显示在线人数数字为消息内容

任务型：

操作	方式	内容字串 1	内容字串 2	内容字串 3	内容字串 4	功能
Password	0	密码	0	0	0	返回用户的密码
Book	0	借阅数目	编号//书名//作者//借阅时间//归还时间……（用户借阅书籍信息）	书本数目	编号//书名//作者//数目//借阅数目……（查询的书籍信息）	返回查询的书籍
	1	用户名	借阅数目	编号//书名//作者//借阅时间//归还时间……		查询用户信息
Error_Borrow	0					不可借阅超过 5 本
	1					未归还不可借阅
Error_Register						用户名相同不可注册
Error_DeleteS						学生未还书不可删除
Error_DeleteB						书未还不可删除
Error_Insert1						书未还不可减少
Info_Borrow	0	书籍名称				借阅成功
Info_Return	0	书籍名称				归还成功
Info_Insert	0	书籍名称	数目			增加成功
	1	书籍名称	数目			减少成功
	2					添加书籍成功
Info_Register	是否成功					注册成功
Info_DeleteB		书籍名称				删除书籍成功
Info_DeleteS						用户删除成功

Info_Login	是否成功					登录成功
Error_Query						查找书籍失败
Error_Student						查无此人
Error_Modify						更换失败
Info_Modify						更换成功

没有表示信息省略，客户端收到服务器消息后进行相应功能实现。下面的实现中会介绍比较重点的功能实现，比较简单的比如弹出 Toast 将不会赘述。

服务器通信的实现：我们使用 Socket 进行传输，JAVA 提供了 socket 类十分方便，只需提供端口号和公网 IP 地址就可以连接。为了防止连接失败，服务器连接是在子线程中的。子线程收到来自服务器的数据想要传到主线程需要借助 handler 类。我们的服务器公网 IP 是 140.143.209.173，端口号设置为 2222。

```
new Thread(new Runnable() {
    @Override
    public void run() {
        try {
            //socket 构造函数中会先尝试连接，未连接则阻塞
            System.out.println("尝试连接。。");
            socket = new Socket("140.143.209.173", 2222);
            System.out.println("连接成功！");
            InputStream inputStream = socket.getInputStream();
            byte[] buffer = new byte[1024];
            int len;
            //inputStream.read()文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时一直阻塞
在此
            while ((len = inputStream.read(buffer)) != -1) {
                String data = new String(buffer, 0, len);
                // 将收到的数据发到主线程中
                Message message = Message.obtain();
                message.what = 1;
                message.obj = data;
                handler.sendMessage(message);  //handler 传给主线程
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}).start();
```

所以需要给服务器发送信息时，只需对 socket 使用流输出即可。下面以发送登陆信息为例。

```
//向服务器端用输出流输出消息
OutputStream outputStream = socket.getOutputStream();
outputStream.write(("Login_S" + "//" + socket.getLocalPort() + "//0//" + username + "//" + password +
"//0").getBytes("utf-8"));
//输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
outputStream.flush();
```

查询显示的实现：在任务型指令信息表中可以看到，在查询书籍和用户信息的过程中，服务器返回了所有需要查询的书籍的全部信息。这样会出现一个问题：socket 一次发送信息的大小是有限制的，经过实测发现，当查询得到的书籍过多时，书籍信息会从中间截断，如此会导致字符串的分割不全而错误，导致程序崩溃。

……编号//书名//作者//数目//借阅数目编号//书名//作者//（此处被截断）

为了解决这一问题我们只选择书籍信息完整的录用进RecyclerView而舍弃被截断的书籍。这样虽然不会崩溃了，但是查询不完整不全面。我们在服务器端新增了继续查询指令，当RecyclerView滚动到最后一个的时候（即用户看到最后一条时），向服务器发送继续查询指令，将剩余书籍内容传输回来添加至RecyclerView。如果这次查询也被截断，则从截断处继续查询，直到书籍全部查找完毕。这样分次查询就绕过了传输信息大小的障碍。

RecyclerView录用书籍代码：

```
    if (split[0].equals("Book") && split[1].equals("0")) {

        ……

        for (int i = 1; i <= number_book && j + 4 < split.length; i++, j += 5) {   //书信息被截断则退
    出录用循环

            ……

            bookList.add(book);

        }

    }
```

当滚动到最后一个项目时向服务器继续查询书籍代码：

```
    public void loadMoreData(final String name, final String author) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Query_B_next" + "//" + socket.getLocalPort() + "//" + way + "//" +
    name + "//" + username + "//" + author).getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有flush清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
```

借阅归还显示的实现：在查询完书籍后，有些书籍是用户已经借阅的，那么应该显示按钮“归还”，有些书籍用户没有借阅，那么应该显示按钮“借阅”，如果库存里的书都被借走了，应该不提供借阅按钮。

如何实现呢？细心的朋友会发现在任务型指令中查询书籍指令返回了用户借阅的书籍。对了，就是暴力比较当前书是否被用户借了(在BookInfo里有type标签指示当前书是“借阅”状态标签还是“归还”状态标签）。因为用户借阅的书籍很少，所以不会消耗很多时间，最多增加5倍常数，而且用户眼睛看的速度跟不上信息处理速度，大可不必担心。

```
if (split[0].equals("Book") && split[1].equals("0")) {
        ……
        if (total == borrow) // 判断是否不可借
            type = 2;
        else
            type = 0;
        for (int k = 0; k < myBookList.size(); k++) // 判断是不是被用户借了，应显示“归还”按钮
            if (myBookList.get(k).getId().equals(split[j])) {
                type = 1;
                break;
            }
       ……
}
```

聊天界面的实现：向服务器输出流传输 Message 信息和 OnlineNumber 请求。每当从服务器收到消息的时候，按照上述对话型指令分析模式，若为 Message 则更新适配器，根据端口号区分是不是自己来在左边或者右边显示消息，用 notification 类在手机端显示通知提示。

Android 自带 SQLite 的连接库，SQLiteOpenHelper 类，直接利用此类在未新建时新建数据库，直接查询、插入、删除。在每次启动软件时都预处理连接，导入聊天记录，加入适配器，显示在聊天界面上，这样实现了聊天记录的保存。

下面是判断是否为自己消息的代码：

```
if (split[1].equals(localPort + "")) {
    ChatMessage chatmsg = new ChatMessage(split[2], split[3], ChatMessage.TYPE_SENT);
    System.out.println("收到自己消息");
    helper.insertChatMessage(chatmsg);
    msgList.add(chatmsg);
} else {
    ChatMessage chatmsg = new ChatMessage(split[2], split[3], ChatMessage.TYPE_RECEIVED);
    System.out.println("收到他人消息");
    helper.insertChatMessage(chatmsg);
    msgList.add(chatmsg);
    ……
}
```

2.4 数据爬虫实验方案

直接爬的豆瓣图书上的数据，用 Xpath 的方式进行的爬虫，爬虫后的数据存入了 csv。为了方便我们的项目，我们筛去了作者的国籍，把书名中的英文逗号改成了中文逗号，去除了作者中的乱码。豆瓣上有很多图书的分类，大致爬了计算机、历史、心理、中外文学、小说、散文、科幻、青春等大类，共 29241 本书。对于单独的一页爬虫关键代码如下：

```
def get_book_info(url):
    html = requests.get(url, headers=headers)
    selector = etree.HTML(html.text)
    try:
        infos= selector.xpath('//li[@class="subject-item"]')
        for info in infos:
            name = info.xpath('div[2]/h2/a/@title')[0]
            author = info.xpath('div[2]/div[1]/text()')[0].split('/')[0].split('、')[0].strip()
            author = re.sub(u"\\(.*?\\)|\\（.*?）|\\{.*?}|\\[.*?]|\\【.*?】", "", author)
            author = re.sub(u".*? 著", "", author)
            author = author.strip()
            #print(name, author)
            writer.writerow((name, author, round(random.uniform(1, 51)), 0))
    except IndexError:
        pass
```

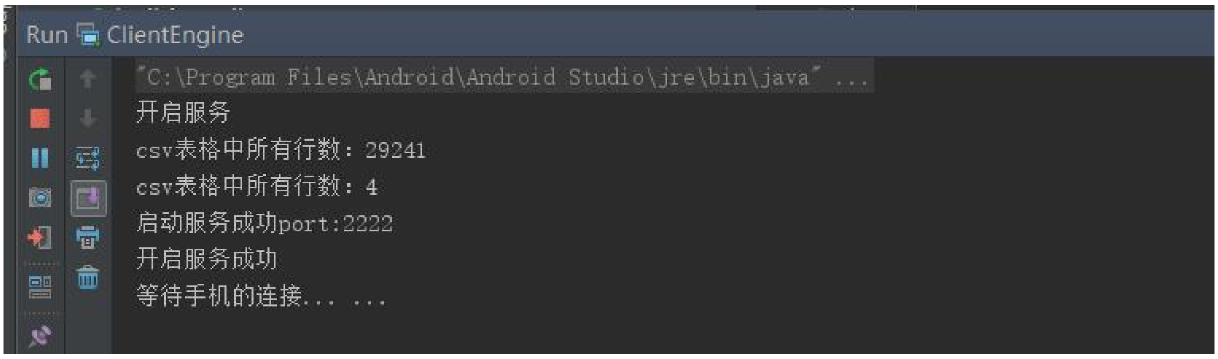
2.5 小组分工

彭思翔：负责数据爬虫，前端设计和编写，数据结构和算法设计，指令设计，服务器管理系统编写，Socket 通信编写，大部分报告的撰写。

喻家乐：负责基础 Trie 树和 Splay 树编写，后端书籍和用户管理服务编写，进行接口设计，提供数据接口服务。

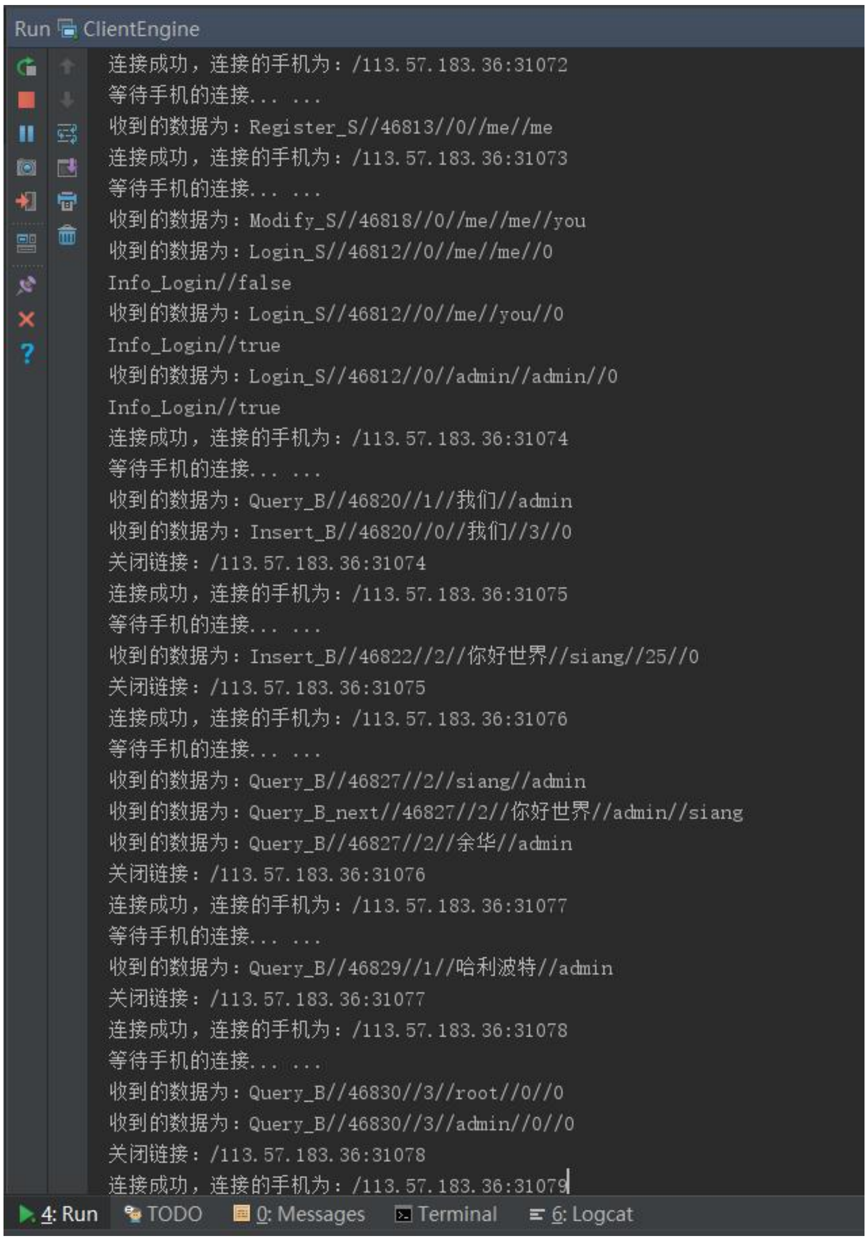
3 实验设计

3.1 服务器实验成果



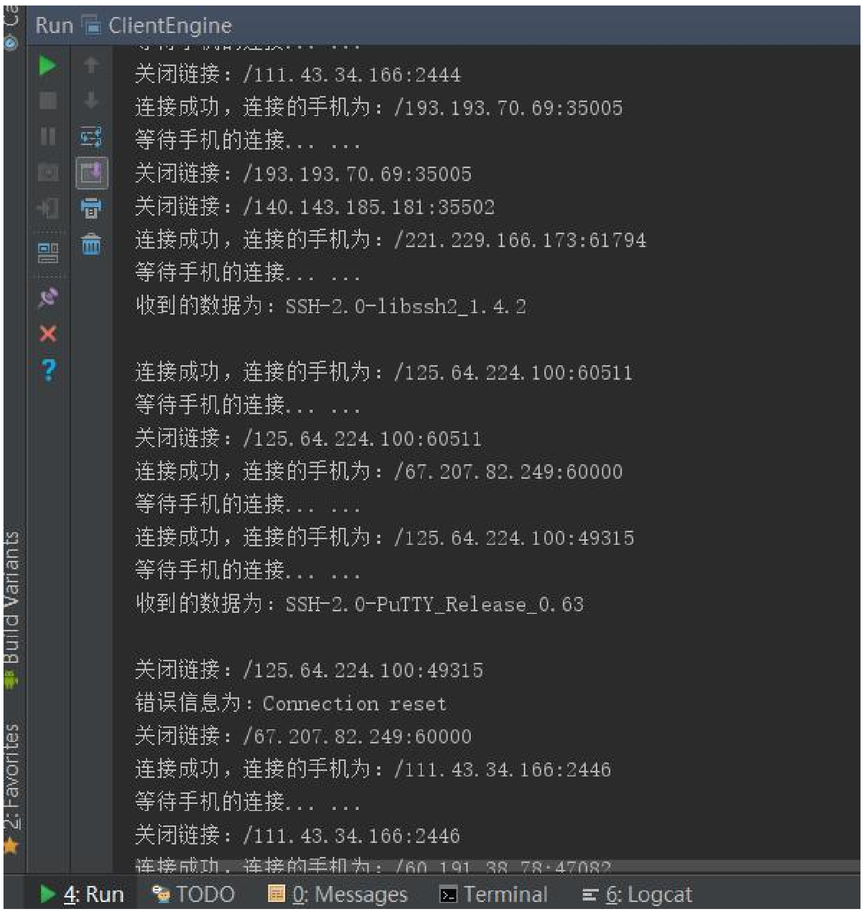
```
"C:\Program Files\Android\Android Studio\jre\bin\java" ...
开启服务
csv表格中所有行数：29241
csv表格中所有行数：4
启动服务成功port:2222
开启服务成功
等待手机的连接... ...
```

图 2 服务器启动成功



```
连接成功，连接的手机为：/113.57.183.36:31072
等待手机的连接... ...
收到的数据为：Register_S//46813//0//me//me
连接成功，连接的手机为：/113.57.183.36:31073
等待手机的连接... ...
收到的数据为：Modify_S//46818//0//me//me//you
收到的数据为：Login_S//46812//0//me//me//0
Info_Login//false
收到的数据为：Login_S//46812//0//me//you//0
Info_Login//true
收到的数据为：Login_S//46812//0//admin//admin//0
Info_Login//true
连接成功，连接的手机为：/113.57.183.36:31074
等待手机的连接... ...
收到的数据为：Query_B//46820//1//我们//admin
收到的数据为：Insert_B//46820//0//我们//3//0
关闭链接：/113.57.183.36:31074
连接成功，连接的手机为：/113.57.183.36:31075
等待手机的连接... ...
收到的数据为：Insert_B//46822//2//你好世界//siang//25//0
关闭链接：/113.57.183.36:31075
连接成功，连接的手机为：/113.57.183.36:31076
等待手机的连接... ...
收到的数据为：Query_B//46827//2//siang//admin
收到的数据为：Query_B_next//46827//2//你好世界//admin//siang
收到的数据为：Query_B//46827//2//余华//admin
关闭链接：/113.57.183.36:31076
连接成功，连接的手机为：/113.57.183.36:31077
等待手机的连接... ...
收到的数据为：Query_B//46829//1//哈利波特//admin
关闭链接：/113.57.183.36:31077
连接成功，连接的手机为：/113.57.183.36:31078
等待手机的连接... ...
收到的数据为：Query_B//46830//3//root//0//0
收到的数据为：Query_B//46830//3//admin//0//0
关闭链接：/113.57.183.36:31078
连接成功，连接的手机为：/113.57.183.36:31079
```

图 3 服务器实现指令功能

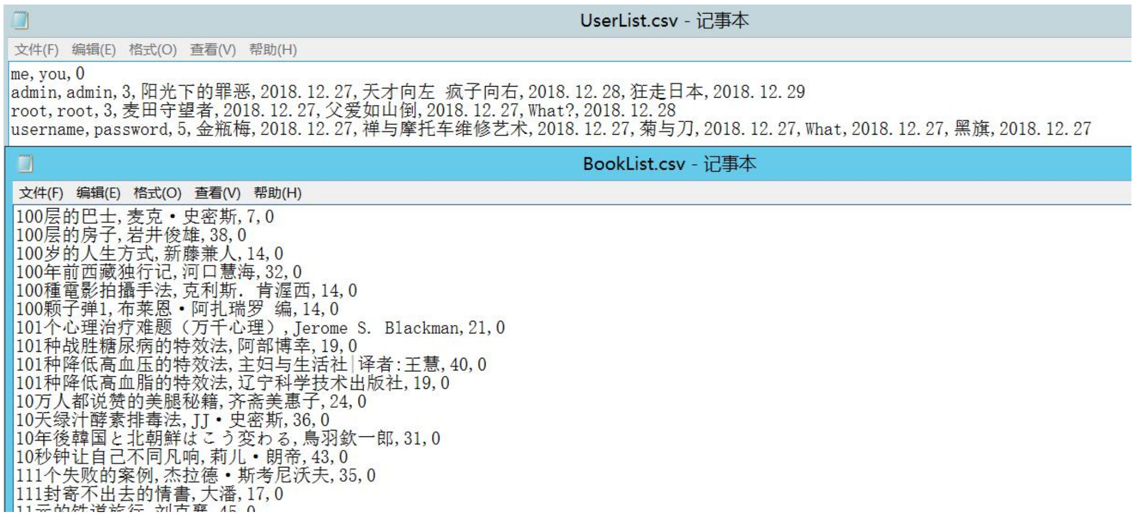


```
关闭链接：/111.43.34.166:2444
连接成功，连接的手机为：/193.193.70.69:35005
等待手机的连接... ...
关闭链接：/193.193.70.69:35005
关闭链接：/140.143.185.181:35502
连接成功，连接的手机为：/221.229.166.173:61794
等待手机的连接... ...
收到的数据为：SSH-2.0-libssh2_1.4.2

连接成功，连接的手机为：/125.64.224.100:60511
等待手机的连接... ...
关闭链接：/125.64.224.100:60511
连接成功，连接的手机为：/67.207.82.249:60000
等待手机的连接... ...
连接成功，连接的手机为：/125.64.224.100:49315
等待手机的连接... ...
收到的数据为：SSH-2.0-PuTTY_Release_0.63

关闭链接：/125.64.224.100:49315
错误信息为：Connection reset
关闭链接：/67.207.82.249:60000
连接成功，连接的手机为：/111.43.34.166:2446
等待手机的连接... ...
关闭链接：/111.43.34.166:2446
连接成功，连接的手机为：/60.191.38.78:47082
```

图 4 服务器受到攻击


UserList.csv - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
BookList.csv - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
me, you, 0
admin, admin, 3, 阳光下的罪恶, 2018. 12. 27, 天才向左 疯子向右, 2018. 12. 28, 狂走日本, 2018. 12. 29
root, root, 3, 麦田守望者, 2018. 12. 27, 父爱如山倒, 2018. 12. 27, What?, 2018. 12. 28
username, password, 5, 金瓶梅, 2018. 12. 27, 禅与摩托车维修艺术, 2018. 12. 27, 菊与刀, 2018. 12. 27, What, 2018. 12. 27, 黑旗, 2018. 12. 27
```

```
100层的巴士, 麦克・史密斯, 7, 0
100层的房子, 岩井俊雄, 38, 0
100岁的人生方式, 新藤兼人, 14, 0
100年前西藏独行记, 河口慧海, 32, 0
100種電影拍攝手法, 克利斯. 肯渥西, 14, 0
100颗子弹1, 布莱恩・阿扎瑞罗 编, 14, 0
101个心理治疗难题（万千心理）, Jerome S. Blackman, 21, 0
101种战胜糖尿病的特效法, 阿部博幸, 19, 0
101种降低高血压的特效法, 主妇与生活社 译者:王慧, 40, 0
101种降低高血脂的特效法, 辽宁科学技术出版社, 19, 0
10万人都说赞的美腿秘籍, 齐斋美惠子, 24, 0
10天绿汁酵素排毒法, JJ・史密斯, 36, 0
10年後韓国と北朝鮮はこう変わる, 鳥羽欽一郎, 31, 0
10秒钟让自己不同凡响, 莉儿・朗帝, 43, 0
111个失败的案例, 杰拉德・斯考尼沃夫, 35, 0
111封寄不出去的情書, 大潘, 17, 0
```

图 5 服务器数据储存

实现了服务器客户端模式的编程，实现了程序的交互，实现了服务器的功能，实现了书籍和用户和数据的读取（图 2），实现了服务器数据的储存（图 5），能够接入客户端并接收来自客户端的指令，对客户端的指令进行分析，并作出相应操作（图 3）。另外服务器每天都遭受着成百上千的攻击，最常见的是针对网页的攻击和 SSH 攻击（图 4），攻击来源的 IP 地址来自世界各地。由于服务器的指令是直接的字符串分析，所以攻击损失不大，但是大量的攻击涌入会导致服务器报错子线程崩溃，手机端的感受就是服务器连接莫名其妙断开。

3.2 客户端实验成果

图 6 “白泽书房”应用图标

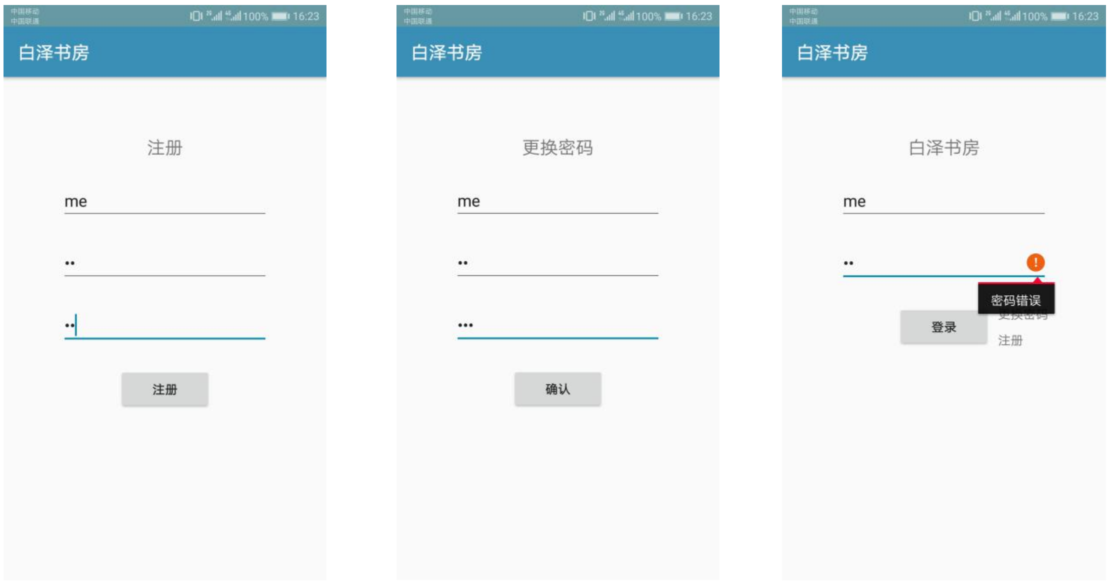


图 7 登陆、注册和更换密码界面

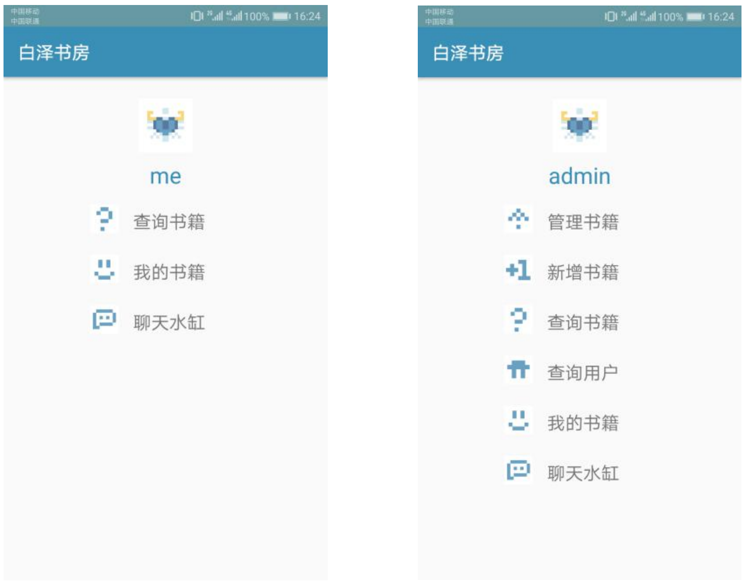
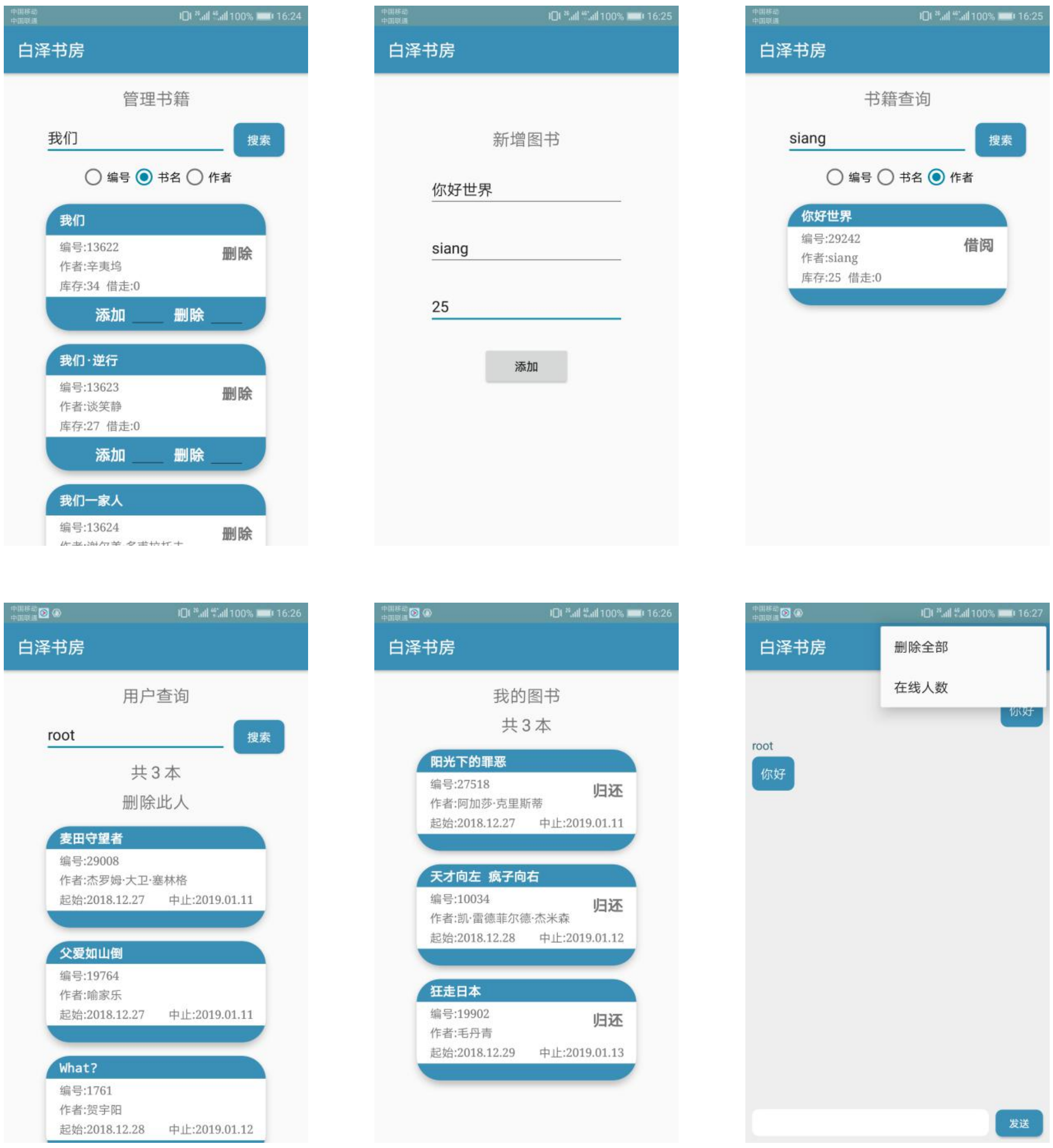


图 8 普通用户和管理者界面


白泽书房

管理书籍

我们 搜索

编号 [✓ 书名] 作者

我们
编号:13622 删除
作者:辛夷坞
库存:34 借走:0
添加 删除

我们·逆行
编号:13623 删除
作者:谈笑静
库存:27 借走:0
添加 删除

我们一家人
编号:13624 删除

白泽书房

新增图书

你好世界

siang

25

添加

白泽书房

书籍查询

siang 搜索

编号 书名 [✓ 作者]

你好世界
编号:29242 借阅
作者:siang
库存:25 借走:0

白泽书房

用户查询

root 搜索

共 3 本

删除此人

麦田守望者
编号:29008
作者:杰罗姆·大卫·塞林格
起始:2018.12.27 中止:2019.01.11

父爱如山倒
编号:19764
作者:喻家乐
起始:2018.12.27 中止:2019.01.11

What?
编号:1761
作者:贺宇阳
起始:2018.12.28 中止:2019.01.12

白泽书房

我的图书

共 3 本

阳光下的罪恶
编号:27518 归还
作者:阿加莎·克里斯蒂
起始:2018.12.27 中止:2019.01.11

天才向左 疯子向右
编号:10034 归还
作者:凯·雷德菲尔德·杰米森
起始:2018.12.28 中止:2019.01.12

狂走日本
编号:19902 归还
作者:毛丹青
起始:2018.12.29 中止:2019.01.13

白泽书房

删除全部

在线人数

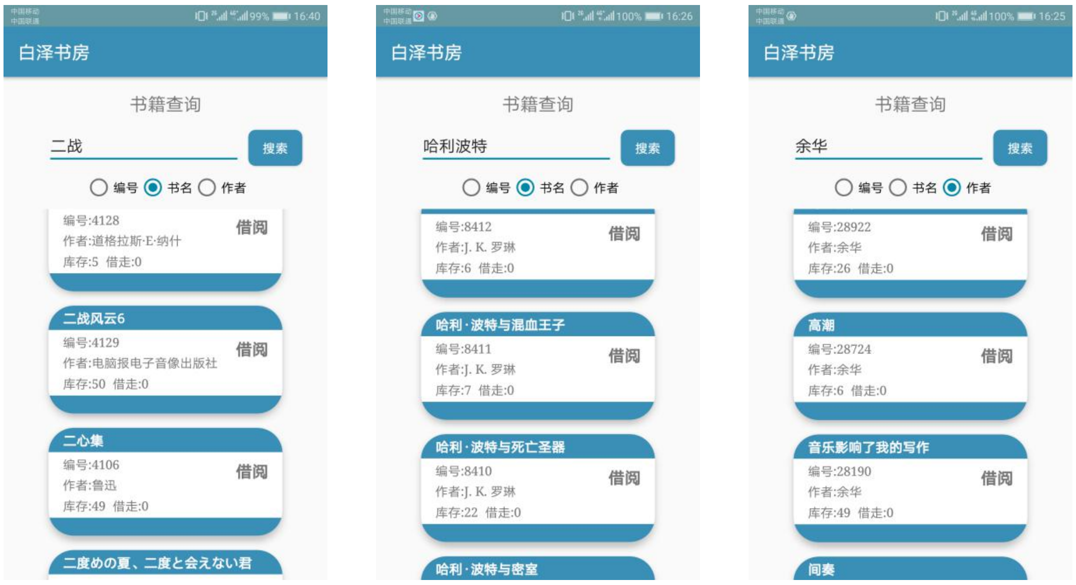
你好

root

你好

发送

图 9 六大功能板块


白泽书房

书籍查询

二战 搜索

编号 [✓ 书名] 作者

编号:4128
作者:道格拉斯·E·纳什
库存:5 借走:0
借阅

二战风云6
编号:4129
作者:电脑报电子音像出版社
库存:50 借走:0
借阅

二心集
编号:4106
作者:鲁迅
库存:49 借走:0
借阅

二度めの夏、二度と会えない君

白泽书房

书籍查询

哈利波特 搜索

编号 [✓ 书名] 作者

编号:8412
作者:J. K. 罗琳
库存:6 借走:0
借阅

哈利·波特与混血王子
编号:8411
作者:J. K. 罗琳
库存:7 借走:0
借阅

哈利·波特与死亡圣器
编号:8410
作者:J. K. 罗琳
库存:22 借走:0
借阅

哈利·波特与密室

白泽书房

书籍查询

余华 搜索

编号 书名 [✓ 作者]

编号:28922
作者:余华
库存:26 借走:0
借阅

高潮
编号:28724
作者:余华
库存:6 借走:0
借阅

音乐影响了我的写作
编号:28190
作者:余华
库存:49 借走:0
借阅

间奏

图 10 模糊匹配和继续查询

实现了客户端用户的登陆、注册和密码更改，并且有良好的反馈（图 7），实现了普通用户和管理者的区分（图 8），实现了书籍的新增、修改、借阅、归还，用户的查询等六大功能板块（图 9）。模糊匹配和继续查询效果很好（图 10）。鲁棒性效果很好。

3.3 其他问题

一是鲁棒性可以进一步完善，虽然已经判断了许多不合法输入的数据，但有些鲁棒性反馈是不到位或者反馈错误，还没有加上或更改，并且客户端很容易跟服务器断开，这里应进一步改进。

二是编码问题，服务器和本机电脑的数据由于编码不同导致乱码，此处已经通过编码统一改进了。

三是 Socket 传输大小限制的问题，这个已经在刚刚用上述继续查询的方式解决了，不过在未来可以尝试用 JSON 和其他方式传输。

四是虽然我们设计的算法在时间上达到了 O(N) 和 O(logM)，其中 N 为用户数量，M 为书籍数量，理论上在 1s 内支持 2^(1000000) 本书的查找更新，但是，我们的数据在操作上都存在了内存中，也就是说会受到内存的限制，服务器内存也就 2G，数据大小由此也就被限制了。在算法上应该改进为基于存储器的外部查找算法，避免浪费太多的内存空间。

3.4 实验总结

这次的项目实验，是我完成的第二个 C/S 结构的程序，并实现了一个较为使用的图书管理应用。

在技术上，我们了解到了面向对象上的一些程序思路，深深地感受到了

包括接口在内的程序设计模式的重要性，我们还重新回顾了 JAVA、XML 的语法和部分库的使用，Android Studio 平台的使用，回顾了 SQLite 数据库的使用方法和 Socket 通讯的原理。

同时我们学习了更多数据结构的知识，学习了 Trie 树以及 Splay 树的构建和使用，深刻比较理解了不同数据结构的适用范围，熟悉了链表的各种情况的用法，加深了对指针的理解。我们学习了根据不同的适用情况设计算法，合理布置通信指令，也学习了更多的排序方法和搜索方法，我们还熟悉了爬虫的应用。我们这些学习也为高年级的数据库原理、计算机网络、算法等课程做了铺垫，有了初步的认识和了解。

我们团结协作，降低项目耦合性的同时共同解决了许多问题。虽然这个项目仍然有一些问题，但是考虑到时间短和自学内容多，基本完成度已经非常较高了。我们都为我们的作品感到骄傲！

4 源代码

4.1 服务器源代码

ClientEngine.java

```
package com.creatures.siang.sever;

import com.creatures.siang.sever.ClientManager;

public class ClientEngine {

    public static void main(String[]args){
        // 开启服务器
        ClientManager.startServer();
    }
}
```

TrieTree.java

```
package com.creatures.siang.sever;

public class TrieTree {
    private Node root;

    public class Node {
        public char         key;
        public User.Node  data;
        public Node[] children;
        public boolean   isEnd;
        public int      counter;

        public Node(char key, User.Node data) {
            this.key      =   key;
            this.data     =  data;
            this.isEnd    = false;
            this.counter =     0;

            this.children = new Node[36];
            for (int i = 0; i < 36; i++) {
                children[i] = null;
            }
        }

        public Node subNode(char theKey) {
            for (int i = 0; i < 36; i++) {
                if (children[i] != null && children[i].key == theKey) {
                    return children[i];
                }
            }

            return null;
        }

        public void insertChild(int index, Node theNode) {
            if (this.children[index] == null) {
                this.children[index] = theNode;
            }
        }



        public void eraseChild(int index) {
            if (this.children[index] != null) {
                this.children[index] = null;
            }
        }
    }

    public TrieTree() {
        this.root = new Node('#', null);
    }

    public User.Node find(String word) {
        Node currentNode = this.root;

        for (int i = 0; i < word.length(); i++) {
            currentNode = currentNode.subNode(word.charAt(i));
            if (currentNode == null) {
                return null;
            }
        }

        if (currentNode.isEnd) {
            return currentNode.data;
        } else {
            return null;
        }
    }

    public boolean insert(String word, User.Node data) {
        if (this.find(word) != null) {
            return false;
        }

        Node currentNode = this.root;
        for (int i = 0; i < word.length(); i++) {
            char c = word.charAt(i);
            Node nextNode = currentNode.subNode(c);
            if (nextNode != null) {
                currentNode = nextNode;
            } else {
                Node newNode = new Node(c, null);
                currentNode.insertChild(encodeCharacter(c), newNode);

                currentNode = currentNode.subNode(c);
            }
            currentNode.counter++;
        }
        currentNode.data  = data;
        currentNode.isEnd = true;
        return true;
    }


    public void erase(String word) {
        if (this.find(word) == null) {
            return;
        }

        Node currentNode = this.root;
        for (int i = 0; i < word.length(); i++) {
            Node nextNode = currentNode.subNode(word.charAt(i));
            if (nextNode.counter == 1) {
                currentNode.eraseChild(encodeCharacter(word.charAt(i)));
                return;
            } else {
                nextNode.counter--;
                currentNode = nextNode;
            }
        }
        currentNode.data  =  null;
        currentNode.isEnd = false;
    }

    private int encodeCharacter(char c) {
        if (c >= 'a' && c <= 'z') {
            return (int)(c - 'a');
        } else if (c >= '0' && c <= '9') {
            return (int)(c - '0') + 26;
        } else {
            return -1;
        }
    }
}
```

SplayTree.java

```
package com.creatures.siang.sever;



public class SplayTree<KeyType extends Comparable<KeyType>, DataType> {

    private Node root;




    public class Node {
        KeyType     key;
        DataNode   data;
        Node  leftChild;
        Node rightChild;

        public class DataNode {
            DataType datum;
            DataNode  next;

            DataNode() {
                this.datum = null;
                this. next = null;
            }

            DataNode(DataType datum) {
                this.datum = datum;
                this.next   =  null;
            }

            void insert(DataType datum) {
                DataNode node  = new DataNode(datum);
                node.next      =          this.next;
                this.next      =               node;
            }
        }

        Node() {
            this.data       = null;
            this.leftChild  = null;
            this.rightChild = null;
        }

        Node(KeyType key, DataType datum) {
            this.key        =  key;
            this.leftChild  = null;
            this.rightChild = null;

            this.data = new DataNode();
            this.insertData(datum);
        }

        public void insertData(DataType datum) {
            DataNode node  = new DataNode(datum);
            node.next      =      this.data.next;
            this.data.next =                node;
        }
    }


    public SplayTree() {
        root = null;
    }




    private Node.DataNode find(Node tree, KeyType key) {
        if (tree == null) {
            return null;
        }

        Node  currentNode = tree;
        int compareResult =    0;

        while (currentNode != null) {
            compareResult = key.compareTo(currentNode.key);

            if (compareResult < 0) {
                currentNode = currentNode.leftChild;
            } else if (compareResult > 0) {
                currentNode = currentNode.rightChild;
            } else {
                return currentNode.data;
            }
        }

        return null;
    }

    public Node.DataNode find(KeyType key) {
        return find(this.root, key);
    }



    private Node.DataNode findFuzzy(Node tree, KeyType key) {
        if (tree == null) {
            return null;
        }

        Node      preNode = tree;
        Node  currentNode = tree;
        int compareResult =    0;

        while (currentNode != null) {
            preNode = currentNode;
            compareResult = key.compareTo(currentNode.key);

            if (compareResult < 0) {
                currentNode = currentNode.leftChild;
            } else if (compareResult > 0) {
                currentNode = currentNode.rightChild;
            } else {
                return currentNode.data;
            }
        }

        return preNode.data;
    }

    public Node.DataNode findFuzzy(KeyType key) {
        return findFuzzy(this.root, key);
```

```
}


private Node max(Node tree) {
    if (tree == null)
        return null;

    while(tree.rightChild != null)
        tree = tree.rightChild;
    return tree;
}

public KeyType max() {
    Node p = max(root);
    if (p != null) {
        return p.key;
    }

    return null;
}

private Node splay(Node tree, KeyType key) {
    if (tree == null) {
        return null;
    }

    Node treeRoot  = new Node();
    Node leftTree  =   treeRoot;
    Node rightTree =   treeRoot;
    Node tempNode  =       null;
    int compareResult =       0;

    while (true) {
        compareResult = key.compareTo(tree.key);

        if (compareResult == 0) {
            break;
        } else if (compareResult < 0) {
            if (tree.leftChild == null) {
                break;
            } else {
                if (key.compareTo(tree.leftChild.key) < 0) {
                    tempNode            =      tree.leftChild;
                    tree.leftChild      = tempNode.rightChild;
                    tempNode.rightChild =                tree;

                    tree = tempNode;

                    if (tree.leftChild == null) {
                        break;
                    }
                }
            }

            rightTree.leftChild = tree;
            rightTree           = tree;

            tree = tree.leftChild;
        } else {
            if (tree.rightChild == null) {
                break;
            } else {
                if (key.compareTo(tree.rightChild.key) > 0) {
                    tempNode          =    tree.rightChild;
                    tree.rightChild    = tempNode.leftChild;
                    tempNode.leftChild =               tree;

                    tree = tempNode;

                    if (tree.rightChild == null) {
                        break;
                    }
                }
            }

            leftTree.rightChild = tree;
            leftTree            = tree;

            tree = tree.rightChild;
        }
    }

    leftTree .rightChild =     tree. leftChild;
    rightTree. leftChild =     tree.rightChild;
    tree     . leftChild = treeRoot.rightChild;
    tree     .rightChild = treeRoot. leftChild;

    return tree;
```

```
}

public void splay(KeyType key) {
    this.root = splay(this.root, key);
}



private Node insert(Node tree, KeyType key, DataType datum) {
    if (tree == null) {
        Node newNode = new Node(key, datum);
        return newNode;
    }

    Node  currentNode =        tree;
    Node   parentNode = currentNode;
    int compareResult =           0;

    while (currentNode != null) {
        parentNode = currentNode;
        compareResult = key.compareTo(currentNode.key);

        if (compareResult < 0) {
            currentNode = currentNode.leftChild;
        } else if (compareResult > 0) {
            currentNode = currentNode.rightChild;
        } else {
            currentNode.insertData(datum);
            return tree;
        }
    }



    Node newNode = new Node(key, datum);
    compareResult = newNode.key.compareTo(parentNode.key);
    if (compareResult < 0) {
        parentNode.leftChild  = newNode;
    } else {
        parentNode.rightChild = newNode;
    }
    return tree;
}

public void insert(KeyType key, DataType data) {
    this.root = this.insert(this.root, key, data);
    this.root = this.splay (this.root, key      );
}


private Node erase(Node tree, KeyType key) {
    if (tree == null) {
        return null;
    }
    if (this.find(tree, key) == null) {
        return tree;
    }

    tree = splay(tree, key);

    Node newRoot;
    if (tree.leftChild == null) {
        newRoot = tree.rightChild;
    } else if (tree.rightChild == null) {
        newRoot = tree.leftChild;
    } else {
        newRoot = splay(tree.leftChild, max(tree.leftChild).key);
        newRoot.rightChild = tree.rightChild;
    }

    return newRoot;
}

public void erase(KeyType key) {
    this.root = this.erase(this.root, key);
}

private Node erase(Node tree, KeyType key, DataType datum) {
    if (tree == null) {
        return null;
    }

    Node.DataNode deleteNode = find(tree, key);
    if (deleteNode != null) {
        Node.DataNode preNode = deleteNode;
        deleteNode = preNode.next;

        while (deleteNode != null) {
            if (deleteNode.datum == datum) {
                preNode.next = deleteNode.next;
                break;
            }

            preNode = deleteNode;
            deleteNode = deleteNode.next;
        }
    }

    return tree;
}

public void erase(KeyType key, DataType datum) {
    this.root = this.erase(this.root, key, datum);
    Node.DataNode deleteNode = find(this.root, key);

    if (deleteNode.next == null) {
        this.root = this.erase(this.root, key);
    }
}
}
```

Book.java

```
package com.creatures.siang.sever;


public class Book {

    private Node head;
    private int  size;
    private SplayTree<Integer, Node>       splayId;
    private SplayTree<String , Node>    splayTitle;
    private SplayTree<String , Node>   splayAuthor;
    private SplayTree<Integer, Node> splayCategory;

    public class Node {
        Integer           id;
        String         title;
        String        author;
        Integer     category;
        int    totalQuantity;
        int borrowedQuantity;

        Node pre;
        Node next;
        Node() {
            this.pre = this.next = null;
        }

        Node(Integer id, String title, String author, Integer category, int totalQuantiy) {
            this.id            =          id;
            this.title         =       title;
            this.author        =      author;
            this.category      =    category;
            this.totalQuantity = totalQuantiy;

            this.borrowedQuantity   = 0;
            this.pre = this.next = null;
        }

       boolean changeTotalQuantity(int increment) {
            int newQuantity = this.totalQuantity + increment;
            if (newQuantity >= 0 && newQuantity >= this.borrowedQuantity) {
                this.totalQuantity = newQuantity;
                return true;
            } else {
                return false;
            }
        }

        boolean changeBorrowedQuantity(int increment) {
            int newQuantity = this.borrowedQuantity + increment;
            if (newQuantity >= 0 && newQuantity <= this.totalQuantity) {
                this.borrowedQuantity = newQuantity;
                return true;
            } else {
                return false;
            }
        }
    }

    public Book() {
        this.head = new Node();
        this.size =          0;

        this.splayId       = new SplayTree<Integer, Node>();
        this.splayTitle    = new SplayTree<String , Node>();
        this.splayAuthor   = new SplayTree<String , Node>();
        this.splayCategory = new SplayTree<Integer, Node>();
    }

    public Node getHead() {
        return this.head;
    }



    public boolean insert(String title, String author, Integer category, int totalQuantiy) {
        if (this.splayTitle.find(title) != null) {
            return false;
        }
        this.size++;
        Node newBook = new Node(this.size, title, author, category, totalQuantiy);
        if (this.head.next == null) {
            this.head.next = newBook;
            newBook.pre = this.head;
        } else {
            Node preBook = this.splayTitle.findFuzzy(title).next.datum;

            if (newBook.title.compareTo(preBook.title) < 0) {
                newBook.pre = preBook.pre;
                if (preBook.pre != null) {
                    preBook.pre.next = newBook;
                }
                newBook.next = preBook;
                preBook.pre = newBook;

            } else {
                newBook.next = preBook.next;
                newBook.pre  =      preBook;
                preBook.next =      newBook;
                if (newBook.next != null) {
                    newBook.next.pre = newBook;
                }
            }
        }

        this.splayId.       insert(newBook.id, newBook);
        this.splayTitle .   insert(title     , newBook);
        this.splayAuthor.   insert(author    , newBook);
        this.splayCategory.insert(category   , newBook);
        return true;
```

```
}



public boolean changeTotalQuantity(String title, int increment) {
    Node node = this.splayTitle.find(title).next.datum;

    return (node != null && node.changeTotalQuantity(increment));
}



public boolean changeBorrowedQuantity(String title, int increment) {
    Node node = this.splayTitle.find(title).next.datum;

    return (node != null && node.changeBorrowedQuantity(increment));
}



public boolean erase(String title) {
    Node deleteNode = this.splayTitle.find(title).next.datum;

    if (deleteNode == null || deleteNode.borrowedQuantity != 0) {
        return false;
    } else {
        deleteNode.pre.next = deleteNode.next;
        if (deleteNode.next != null) {
            deleteNode.next.pre = deleteNode.pre;
        }

        this.splayId      .erase(deleteNode.id                  );
        this.splayTitle   .erase(deleteNode.title               );
        this.splayAuthor  .erase(deleteNode.author  , deleteNode);
        this.splayCategory.erase(deleteNode.category, deleteNode);

        return true;
    }
}

public SplayTree<Integer, Node>.Node.DataNode findById(Integer id) {
    return (this.splayId.find(id));
}
```

```java
    public SplayTree<String, Node>.Node.DataNode findByTitle(String title) {
        return (this.splayTitle.find(title));
    }

    public SplayTree<String, Node>.Node.DataNode findByTitleFuzzy(String title) {
        return (this.splayTitle.findFuzzy(title));
    }



    private int compareString(String a, String b) {
        int length = Math.min(a.length(), b.length());
        int i = 0;
        for (i = 0; i < length; i++) {
            if (a.charAt(i) != b.charAt(i)) {
                break;
            }
        }

        return i;
    }

    public SplayTree<String, Node>.Node.DataNode findByAuthor(String author) {
        return (this.splayAuthor.find(author));
    }

    public SplayTree<Integer, Node>.Node.DataNode findByCategory(Integer category) {
        return (this.splayCategory.find(category));
    }
}
```

User.java

```java
package com.creatures.siang.sever;


import java.util.ArrayList;


public class User {


    private Node     head;
    private TrieTree trie;


    protected class Node {
        protected String     name;
        protected String password;
        protected java.util.List<Record> records;

        protected Node pre ;
        protected Node next;

        protected class Record {
            String      title;
            java.util.Date date;

            Record(String title, java.util.Date date) {
                this.title = title;
                this.date  =  date;
            }
        }

        protected Node() {
            this.records = new ArrayList<Record>();
            this.pre = this.next = null;
        }

        protected Node(String name, String password) {
            this.name     =     name;
            this.password = password;

            this.records = new ArrayList<Record>();
            this.pre = this.next = null;
        }

        protected boolean borrowBook(String title) {
            java.util.Date date = new java.util.Date();
            for (int i = 0; i < this.records.size(); i++) {
                if (this.records.get(i).title.equals(title)) {
                    return false;
                }
            }

            Record newRecord = new Record(title, date);
            this.records.add(newRecord);
            return true;
        }

        protected boolean borrowBook(String title, java.util.Date date) {
            for (int i = 0; i < this.records.size(); i++) {
                if (this.records.get(i).title.equals(title)) {
                    return false;
                }
            }

            Record newRecord = new Record(title, date);
            this.records.add(newRecord);
            return true;
        }

        protected boolean returnBook(String title) {
            for (int i = 0; i < this.records.size(); i++) {
                if (this.records.get(i).title.equals(title)) {
                    this.records.remove(i);
                    return true;
                }
            }

            return false;
        }

        protected boolean findBook(String title) {
            if (this.records.size() == 0) {
                return false;
            } else {
                for (int i = 0; i < this.records.size(); i++) {
                    if (this.records.get(i).title.equals(title)) {
                        return true;
                    }
                }

                return false;
            }
        }

        protected java.util.List<Record> getAllRecords() {
            return this.records;
        }
    }


    public User() {
        this.head = new     Node();
        this.trie = new TrieTree();
    }
```

```
public Node getHead() {
    return this.head;
}


public boolean insert(String name, String password) {
    if (this.trie.find(name) != null) {
        return false;
    }

    Node newUser = new Node(name, password);

    newUser.pre    =      this.head;
    newUser.next   = this.head.next;
    this.head.next =        newUser;

    if (newUser.next != null) {
        newUser.next.pre = newUser;
    }

    this.trie.insert(name, newUser);

    return true;
}


public boolean erase(String name) {
    Node deleteNode = this.trie.find(name);

    if (deleteNode == null || deleteNode.records.size() != 0) {
        return false;
    } else {
        deleteNode.pre.next = deleteNode.next;
        if (deleteNode.next != null) {
            deleteNode.next.pre =  deleteNode.pre;
        }

        this.trie.erase(name);

        return true;
    }
}
```

```
public boolean changePassword(String name, String newPassword) {
    Node theUser = this.find(name);

    if (theUser != null) {
        theUser.password = newPassword;
        return true;
    }
    return false;
}


public boolean borrowBook(String name, String title) {
    Node theUser = this.find(name);

    return (theUser != null && theUser.borrowBook(title));
}

public boolean borrowBook(String name, String title, java.util.Date date) {
    Node theUser = this.find(name);

    return (theUser != null && theUser.borrowBook(title, date));
}



public boolean returnBook(String name, String title) {
    Node theUser = this.find(name);

    return (theUser != null && theUser.returnBook(title));
}


public java.util.List<Node.Record> getAllRecords(String name) {
    Node theUser = this.find(name);

    if (theUser != null) {
        return theUser.getAllRecords();
    } else {
        return null;
    }
}

public Node find(String name) {
        return this.trie.find(name);
    }

    public boolean match(String name, String password) {
        Node n = this.find(name);

        return (n != null && n.password.equals(password));
    }
}
```

ClientManager.java

```
package com.creatures.siang.sever;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import com.creatures.siang.sever.*;

import com.creatures.siang.sever.MysqlManager;

/**
 * Created by siang on 2018/5/29.
```

```java
 */

public class ClientManager {

    private static Map<String,Socket> clientList = new HashMap<>();
    private static ServerThread serverThread = null;
    public static int openRobot = 0;
    private static User users = new User();
    private static Book books = new Book();

    private static class ServerThread implements Runnable {

        private int port = 2222;
        private boolean isExit = false;
        private ServerSocket server;

        public ServerThread() {
            try {
                server = new ServerSocket(port);
                System.out.println("启动服务成功" + "port:" + port);
            } catch (IOException e) {
                System.out.println("启动 server 失败，错误原因：" + e.getMessage());
            }
        }

        @Override
        public void run() {
            try {
                while (!isExit) {
                    // 进入等待环节
                    System.out.println("等待手机的连接... ... ");
                    final Socket socket = server.accept();
                    // 获取手机连接的地址及端口号
                    final String address = socket.getRemoteSocketAddress().toString();
                    System.out.println("连接成功，连接的手机为：" + address);

                    new Thread(new Runnable() {
                        //匿名内部类的方式来创建一个线程
                        @Override
                        public void run() {
                            try {
                                // 单线程索锁
                                synchronized (this){
                                    // 放进到 Map 中保存
    clientList.put(address,socket);
}
// 定义输入流
InputStream inputStream = socket.getInputStream();
byte[] buffer = new byte[1024];
int len;
while ((len = inputStream.read(buffer)) != -1){
    String text = new String(buffer,0,len, "UTF-8");
    System.out.println("收到的数据为：" + text);

    String[] split = ((String) text).split("//");
    if (split[0].equals("Message")) {
        // 在这里群发消息
        if (split[3].equals("打开机器人")) {
            openRobot = 1;
        }
        if (split[3].equals("关闭机器人")) {
            openRobot = 0;
        }
        sendMsgAll(text);
    }
    else if (split[0].equals("OnlineNumber")) {
        sendOnlineNumber(socket);
    }
    else if (split[0].equals("Insert_B")) {
        if (split[2].equals("0")) {
            sendAddBook(split[3], split[4], socket);
        }
        else if (split[2].equals("1")) {
            sendDecBook(split[3], split[4], socket);
        }
        else if (split[2].equals("2")) {
            sendInsertBook(split[3], split[4], split[5], split[6],
socket);

        }
    }
    else if (split[0].equals("Query_B")) {
        if (split[2].equals("3")) {
            sendQueryStudentBook(split[3], socket);
        }
        else if(split[2].equals("2")) {
            sendQueryBookAuthor(split[3], split[4], socket);
        }
        else if(split[2].equals("1")) {
```

```
                    sendQueryBookName(split[3], split[4], socket);
                }
                else if(split[2].equals("0")) {
                    sendQueryBookID(split[3], split[4], socket);
                }
            }
            else if (split[0].equals("Query_B_next")) {
                if(split[2].equals("2")) {
                    sendQueryBookAuthorNext(split[3], split[4], split[5], socket);
                }
                else if(split[2].equals("1")) {
                    sendQueryBookNameNext(split[3], split[4], split[5], socket);
                }
            }
            else if (split[0].equals("Borrow_S")) {
                sendBorrowBook(text, socket);
            }
            else if (split[0].equals("Return_S")) {
                sendReturnBook(text, socket);
            }
            else if (split[0].equals("Register_S")) {
                sendRegister(split[3], split[4], socket);
            }
            else if (split[0].equals("Login_S")) {
                sendLogin(split[3], split[4], socket);
            }
            else if (split[0].equals("Delete_B")) {
                sendDeleteBook(split[3], socket);
            }
            else if (split[0].equals("Delete_S")) {
                sendDeleteStudent(split[3], socket);
            }
            else if (split[0].equals("Modify_S")) {
                sendChangePassword(split[3], split[4], split[5], socket);
            }
        }

    }catch (Exception e){
        System.out.println("错误信息为：" + e.getMessage());
    }finally {
        synchronized (this){
            System.out.println("关闭链接：" + address);
            clientList.remove(address);
        }
```

```
                        }
                    }
                }).start();

            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void Stop(){
        isExit = true;
        if (server != null){
            try {
                server.close();
                System.out.println("已关闭 server");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

public static ServerThread startServer(){
    System.out.println("开启服务");
    if (serverThread != null){
        showDown();
    }
    //启动 MySQL
    //MysqlManager.startMysql();
    readBooks();
    readUsers();
    saveBook();
    saveUser();
    serverThread = new ServerThread();
    new Thread(serverThread).start();
    System.out.println("开启服务成功");
    return serverThread;
}

public static void readBooks() {
    BufferedReader br = null;
    try
    {
            File csv = new
File("C:\\Creatures\\Android\\lib\\src\\main\\java\\com\\creatures\\siang\\sever\\BookList.csv");  //
CSV 文件路径
            br = new BufferedReader(new FileReader(csv));
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        String line = "";
        String everyLine = "";
        try {
            int count = 0;
            while ((line = br.readLine()) != null)  //读取到的内容给 line 变量
            {
                everyLine = line;
                //System.out.println(count+ everyLine);
                String[] split = ((String) everyLine).split(",");
                boolean can = books.insert(split[0].trim().replace("\"", ""), split[1].trim().replace("\"",
"").replace("?", "·"), 0, Integer.parseInt(split[2]));
                if (can) {
                    Book.Node theBook = books.findByTitle(split[0].trim().replace("\"", "")).next.datum;
                    theBook.changeBorrowedQuantity(Integer.parseInt(split[3]));
                    count ++;
                }
            }
            System.out.println("csv 表格中所有行数: "+count);
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }


    public static void readUsers() {
        BufferedReader br = null;
        try
        {
            File csv = new
File("C:\\Creatures\\Android\\lib\\src\\main\\java\\com\\creatures\\siang\\sever\\UserList.csv");  //
CSV 文件路径
            br = new BufferedReader(new FileReader(csv));
        } catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        String line = "";
        String everyLine = "";
        try {
            List<String> allString = new ArrayList<>();
            DateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
            while ((line = br.readLine()) != null)  //读取到的内容给 line 变量
            {
                everyLine = line;
                String[] split = ((String) everyLine).split(",");
                users.insert(split[0].trim(), split[1].trim());
                int count = Integer.parseInt(split[2].trim());
                int j = 3;
                Date date = new Date();
                for (int i = 1; i <= count; i++, j += 2) {
                    try {
                            date = sdf.parse(split[j+1].trim());
                        } catch (ParseException e)
                        {
                            e.printStackTrace();
                        }
                    users.borrowBook(split[0].trim(), split[j].trim(), date);
                }
                allString.add(everyLine);
            }
            System.out.println("csv 表格中所有行数: "+allString.size());
        } catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    public static void saveBook() {
        final long timeInterval = 300000;// 5 分钟运行一次
        Runnable runnable = new Runnable() {
            public void run() {
                while (true) {
                    try {
                        FileOutputStream outFile = new
FileOutputStream("C:\\Creatures\\Android\\lib\\src\\main\\java\\com\\creatures\\siang\\sever\\BookList.
csv");//写出的 CSV 文件
                        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(outFile,
"UTF-8"));
                        Book.Node node = books.getHead().next;
                        while (node != null) {
                            String text = node.title + "," + node.author + "," +
String.valueOf(node.totalQuantity) + "," + String.valueOf(node.borrowedQuantity);
                            writer.write(text);
                            writer.newLine();
                            node = node.next;
                        }
                        writer.close();
                    }  catch (IOException e)
                    {
                        e.printStackTrace();
                    }
                    try {
                        Thread.sleep(timeInterval);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        };
        Thread thread = new Thread(runnable);
        thread.start();
    }


    public static void saveUser() {
        final long timeInterval = 360000;// 6 分钟运行一次
        Runnable runnable = new Runnable() {
            public void run() {
                while (true) {
                    try {
                        FileOutputStream outFile = new
FileOutputStream("C:\\Creatures\\Android\\lib\\src\\main\\java\\com\\creatures\\siang\\sever\\UserList.
csv");//写出的 CSV 文件
                        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(outFile,
"UTF-8"));
                        User.Node node = users.getHead().next;
                        while (node != null) {
                            String text = node.name + "," + node.password + ",";
                            String temp = "";
                            int count = 0;
                            SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
                            List<User.Node.Record> borrowRecord = users.getAllRecords(node.name);
                            if (borrowRecord != null) {
                                for (int i = 0; i < borrowRecord.size(); i++) {
                                    temp = temp + ',' + borrowRecord.get(i).title + ',' +
sdf.format(borrowRecord.get(i).date);
                                    count ++;
                                }
                            }
                            text = text + String.valueOf(count) + temp;
                            writer.write(text);
                            writer.newLine();
                            node = node.next;
                        }
                        writer.close();
                    }  catch (IOException e)
                    {
                        e.printStackTrace();
                    }
                    try {
                        Thread.sleep(timeInterval);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        };
        Thread thread = new Thread(runnable);
        thread.start();
    }

    // 关闭所有 server socket 和 清空 Map
    public static void showDown(){
        MysqlManager.endMysql();
        for (Socket socket : clientList.values()) {
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        serverThread.Stop();
        clientList.clear();
    }

    // 群发的方法
    public static boolean sendMsgAll(String msg){
```

```
    try {
        String[] split = ((String) msg).split("//");
        String baize = "Message//0000//白泽//" + MysqlManager.queryMySQL(split[3]); //白泽回复
        for (Socket socket : clientList.values()) {
            OutputStream outputStream = socket.getOutputStream();
            outputStream.write(msg.getBytes("utf-8"));
            outputStream.flush();
            if (openRobot == 1) {
                outputStream.write(baize.getBytes("utf-8"));
                outputStream.flush();
            }
        }
        return true;
    }catch (Exception e){
        e.printStackTrace();
    }
    return false;
}

public static void sendOnlineNumber(Socket socket){
    try {
        OutputStream outputStream = socket.getOutputStream();
        String text = "OnlineNumber" + "//" + String.valueOf(clientList.size());
        outputStream.write(text.getBytes("utf-8"));
        outputStream.flush();
    }catch (Exception e){
        e.printStackTrace();
    }
}

public static void sendAddBook(String name, String num, Socket socket){
    boolean can = books.changeTotalQuantity(name, Integer.parseInt(num));

    try {
        OutputStream outputStream = socket.getOutputStream();
        String text;
        text = "Info_Insert//0//" + name + "//" + num;
        outputStream.write(text.getBytes("utf-8"));
        outputStream.flush();
    }catch (Exception e){
        e.printStackTrace();
    }
}
```

```
    public static void sendDecBook(String name, String num, Socket socket){
        boolean can = books.changeTotalQuantity(name, -Integer.parseInt(num));

        try {
            OutputStream outputStream = socket.getOutputStream();
            String text = "Error_Insert1";
            if (can)
                text = "Info_Insert//1//" + name + "//" + num;
            outputStream.write(text.getBytes("utf-8"));
            outputStream.flush();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    public static void sendInsertBook(String title, String author, String number, String category, Socket
socket){
        boolean can = books.insert(title, author, Integer.parseInt(category), Integer.parseInt(number));

        try {
            OutputStream outputStream = socket.getOutputStream();
            String text;
            if (can)
                text = "Info_Insert//2";
            else
                text = "Error_Insert";
            outputStream.write(text.getBytes("utf-8"));
            outputStream.flush();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    public static void sendQueryBookNameNext(String title, String username, String keyTitle, Socket socket){
        SplayTree<String, Book.Node>.Node.DataNode can = books.findByTitleFuzzy(title);
        List<User.Node.Record> borrowRecord = users.getAllRecords(username);
        if (can == null) {
            try {
                OutputStream outputStream = socket.getOutputStream();
                String text;
                text = "Error_Query";
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
        else {
            Book.Node theBook = can.next.datum;
            try {
                OutputStream outputStream = socket.getOutputStream();
                String tempu = "";
                if (borrowRecord == null) {
                    tempu = tempu + "//0";
                }
                else {
                    tempu = tempu + "//" + String.valueOf(borrowRecord.size());
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
                    for (int i = 0; i < borrowRecord.size(); i++) {
                        Book.Node theUserBook = books.findByTitle(borrowRecord.get(i).title).next.datum;
                        tempu = tempu + "//" + String.valueOf(theUserBook.id) + "//" + theUserBook.title
+ "//" + theUserBook.author + "//" + sdf.format(borrowRecord.get(i).date) + "//" +
sdf.format(getEndDate(borrowRecord.get(i).date));
                    }
                }

                String temp = "";
                int count = 0, key = 0;
                if (theBook.title.compareTo(title) >= 0) {
                    int prefix_or = maxPrefix(theBook.title, title);
                    if (prefix_or > theBook.title.length() / 3 || prefix_or == title.length()) {
                        if (key == 1) {
                            temp = temp + "//" + String.valueOf(theBook.id) + "//" + theBook.title + "//"
+ theBook.author + "//" + String.valueOf(theBook.totalQuantity) + "//" +
String.valueOf(theBook.borrowedQuantity);
                            count++;
                        }
                        if (theBook.title.equals(keyTitle)) key = 1;
                    }
                }
                Book.Node currentBook;
                currentBook = theBook.next;
                while (true) {
                    if (count >= 20) break;
                    if (currentBook == null) {
                        break;
                    }
                    int len = currentBook.title.length() / 3;
                    int prefix = maxPrefix(currentBook.title, title);
                    if (prefix > len || prefix == title.length()) {
                        if (key == 1) {
                            temp = temp + "//" + String.valueOf(currentBook.id) + "//" + currentBook.title
+ "//" + currentBook.author + "//" + String.valueOf(currentBook.totalQuantity) + "//" +
String.valueOf(currentBook.borrowedQuantity);
                            count++;
                        }
                        if (currentBook.title.equals(keyTitle)) key = 1;
                    }
                    else {
                        break;
                    }
                    currentBook = currentBook.next;
                }

                if (theBook.title.compareTo(title) < 0) {
                    int prefix_or = maxPrefix(theBook.title, title);
                    if (prefix_or > title.length() / 3 || prefix_or == theBook.title.length()) {
                        if (key == 1) {
                            temp = temp + "//" + String.valueOf(theBook.id) + "//" + theBook.title + "//"
+ theBook.author + "//" + String.valueOf(theBook.totalQuantity) + "//" +
String.valueOf(theBook.borrowedQuantity);
                            count++;
                        }
                        if (theBook.title.equals(keyTitle)) key = 1;
                    }
                }
                currentBook = theBook.pre;
                while (true) {
                    if (count >= 20) break;
                    if (currentBook.pre == null) {
                        break;
                    }
                    int len = title.length() / 3;
                    int prefix = maxPrefix(currentBook.title, title);
                    if (prefix > len || prefix == currentBook.title.length()) {
                        if (key == 1) {
                            temp = temp + "//" + String.valueOf(currentBook.id) + "//" + currentBook.title
+ "//" + currentBook.author + "//" + String.valueOf(currentBook.totalQuantity) + "//" +
String.valueOf(currentBook.borrowedQuantity);
                            count++;
                        }
                        if (currentBook.title.equals(keyTitle)) key = 1;
                    }
                    else {
                        break;
                    }
                    currentBook = currentBook.pre;
                }

                String text;
                if (count > 0)
                    text = "Book_next" + "//" + "0"  + tempu + "//" + String.valueOf(count) + temp;
                else
                    text = "Error_Query_next";
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    public static void sendQueryBookAuthorNext(String title, String username, String author, Socket socket){
        SplayTree<String, Book.Node>.Node.DataNode can = books.findByAuthor(author);
        List<User.Node.Record> borrowRecord = users.getAllRecords(username);
        if (can == null) {
            try {
                OutputStream outputStream = socket.getOutputStream();
                String text;
                text = "Error_Query";
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
        else {
            SplayTree<String, Book.Node>.Node.DataNode theBook = can.next;
            try {
                OutputStream outputStream = socket.getOutputStream();
                String tempu = "";
                if (borrowRecord == null) {
                    tempu = tempu + "//0";
                }
                else {
                    tempu = tempu + "//" + String.valueOf(borrowRecord.size());
```

```
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
                    for (int i = 0; i < borrowRecord.size(); i++) {
                        Book.Node theUserBook = books.findByTitle(borrowRecord.get(i).title).next.datum;
                        tempu = tempu + "//" + String.valueOf(theUserBook.id) + "//" + theUserBook.title
+ "//" + theUserBook.author + "//" + sdf.format(borrowRecord.get(i).date) + "//" +
sdf.format(getEndDate(borrowRecord.get(i).date));
                    }
                }
                String text, temp = "";
                int count = 0, key = 0;
                while (theBook != null) {
                    if (count >= 20)
                        break;
                    if (key == 1) {
                        count += 1;
                        temp = temp + "//" + String.valueOf(theBook.datum.id) + "//" + theBook.datum.title
+ "//" + theBook.datum.author + "//" + String.valueOf(theBook.datum.totalQuantity) + "//" +
String.valueOf(theBook.datum.borrowedQuantity);
                    }
                    if (theBook.datum.title.equals(title)) key = 1;
                    theBook = theBook.next;
                }
                if (count > 0)
                    text = "Book_next" + "//" + "0"  + tempu + "//" + String.valueOf(count) + temp;
                else
                    text = "Error_Query_next";

                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    public static void sendQueryBookName(String title, String username, Socket socket){
        SplayTree<String, Book.Node>.Node.DataNode can = books.findByTitleFuzzy(title);
        List<User.Node.Record> borrowRecord = users.getAllRecords(username);
        if (can == null) {
            try {
                OutputStream outputStream = socket.getOutputStream();
                String text;
                text = "Error_Query";
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
        else {
            Book.Node theBook = can.next.datum;
            try {
                OutputStream outputStream = socket.getOutputStream();
                String tempu = "";
                if (borrowRecord == null) {
                    tempu = tempu + "//0";
                }
                else {
                    tempu = tempu + "//" + String.valueOf(borrowRecord.size());
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
                    for (int i = 0; i < borrowRecord.size(); i++) {
                        Book.Node theUserBook = books.findByTitle(borrowRecord.get(i).title).next.datum;
                        tempu = tempu + "//" + String.valueOf(theUserBook.id) + "//" + theUserBook.title
+ "//" + theUserBook.author + "//" + sdf.format(borrowRecord.get(i).date) + "//" +
sdf.format(getEndDate(borrowRecord.get(i).date));
                    }
                }

                String temp = "";
                int count = 0;
                if (theBook.title.compareTo(title) >= 0) {
                    int prefix_or = maxPrefix(theBook.title, title);
                    if (prefix_or > theBook.title.length() / 3 || prefix_or == title.length()) {
                        temp = temp + "//" + String.valueOf(theBook.id) + "//" + theBook.title + "//" +
theBook.author + "//" + String.valueOf(theBook.totalQuantity) + "//" +
String.valueOf(theBook.borrowedQuantity);
                        count++;
                    }
                }
                Book.Node currentBook;
                currentBook = theBook.next;
                while (true) {
                    if (count >= 20) break;
                    if (currentBook == null) {
                        break;
                    }
                    int len = currentBook.title.length() / 3;
                    int prefix = maxPrefix(currentBook.title, title);
                    if (prefix > len || prefix == title.length()) {
                        temp = temp + "//" + String.valueOf(currentBook.id) + "//" + currentBook.title +
"//" + currentBook.author + "//" + String.valueOf(currentBook.totalQuantity) + "//" +
String.valueOf(currentBook.borrowedQuantity);
                        count++;
                    }
                    else {
                        break;
                    }
                    currentBook = currentBook.next;
                }

                if (theBook.title.compareTo(title) < 0) {
                    int prefix_or = maxPrefix(theBook.title, title);
                    if (prefix_or > title.length() / 3 || prefix_or == theBook.title.length()) {
                        temp = temp + "//" + String.valueOf(theBook.id) + "//" + theBook.title + "//" +
theBook.author + "//" + String.valueOf(theBook.totalQuantity) + "//" +
String.valueOf(theBook.borrowedQuantity);
                        count++;
                    }
                }
                currentBook = theBook.pre;
                while (true) {
                    if (count >= 20) break;
                    if (currentBook.pre == null) {
                        break;
                    }
                    int len = title.length() / 3;
                    int prefix = maxPrefix(currentBook.title, title);
                    if (prefix > len || prefix == currentBook.title.length()) {
                        temp = temp + "//" + String.valueOf(currentBook.id) + "//" + currentBook.title +
"//" + currentBook.author + "//" + String.valueOf(currentBook.totalQuantity) + "//" +
String.valueOf(currentBook.borrowedQuantity);
                        count++;
                    }
                    else {
                        break;
                    }
                    currentBook = currentBook.pre;
                }

                String text;
                if (count > 0)
                    text = "Book" + "//" + "0"  + tempu + "//" + String.valueOf(count) + temp;
                else
                    text = "Error_Query";
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    public static void sendQueryBookAuthor(String author, String username, Socket socket){
        SplayTree<String, Book.Node>.Node.DataNode can = books.findByAuthor(author);
        List<User.Node.Record> borrowRecord = users.getAllRecords(username);
        if (can == null) {
            try {
                OutputStream outputStream = socket.getOutputStream();
                String text;
                text = "Error_Query";
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
        else {
            SplayTree<String, Book.Node>.Node.DataNode theBook = can.next;
            try {
                OutputStream outputStream = socket.getOutputStream();
                String tempu = "";
                if (borrowRecord == null) {
                    tempu = tempu + "//0";
                }
                else {
                    tempu = tempu + "//" + String.valueOf(borrowRecord.size());
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
                    for (int i = 0; i < borrowRecord.size(); i++) {
                        Book.Node theUserBook = books.findByTitle(borrowRecord.get(i).title).next.datum;
                        tempu = tempu + "//" + String.valueOf(theUserBook.id) + "//" + theUserBook.title
+ "//" + theUserBook.author + "//" + sdf.format(borrowRecord.get(i).date) + "//" +
sdf.format(getEndDate(borrowRecord.get(i).date));
                    }
                }
                String text, temp = "";
                int count = 0;
                while (theBook != null) {
                    if (count >= 20)
                        break;
                    count += 1;
                    temp = temp + "//" + String.valueOf(theBook.datum.id) + "//" + theBook.datum.title +
"//" + theBook.datum.author + "//" + String.valueOf(theBook.datum.totalQuantity) + "//" +
String.valueOf(theBook.datum.borrowedQuantity);
                    theBook = theBook.next;
                }
                text = "Book" + "//" + "0"  + tempu + "//" + String.valueOf(count) + temp;
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    public static void sendQueryBookID(String ID, String username, Socket socket){
        SplayTree<Integer, Book.Node>.Node.DataNode can = books.findById(Integer.parseInt(ID));
        List<User.Node.Record> borrowRecord = users.getAllRecords(username);
        if (can == null) {
            try {
                OutputStream outputStream = socket.getOutputStream();
                String text;
                text = "Error_Query";
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
        else {
            Book.Node theBook = can.next.datum;
            try {
                OutputStream outputStream = socket.getOutputStream();
                String tempu = "";
                if (borrowRecord == null) {
                    tempu = tempu + "//0";
                }
                else {
                    tempu = tempu + "//" + String.valueOf(borrowRecord.size());
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
                    for (int i = 0; i < borrowRecord.size(); i++) {
                        Book.Node theUserBook = books.findByTitle(borrowRecord.get(i).title).next.datum;
                        tempu = tempu + "//" + String.valueOf(theUserBook.id) + "//" + theUserBook.title
+ "//" + theUserBook.author + "//" + sdf.format(borrowRecord.get(i).date) + "//" +
sdf.format(getEndDate(borrowRecord.get(i).date));
                    }
                }
                String text;
                text = "Book" + "//" + "0"  + tempu + "//1//" + String.valueOf(theBook.id) + "//" +
theBook.title + "//" + theBook.author + "//" + String.valueOf(theBook.totalQuantity) + "//" +
String.valueOf(theBook.borrowedQuantity);
                outputStream.write(text.getBytes("utf-8"));
                outputStream.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    public static void sendDeleteBook(String name, Socket socket){
        boolean can = books.erase(name);

        try {
            OutputStream outputStream = socket.getOutputStream();
            String text = "Error_DeleteB";
            if (can)
                text = "Info_DeleteB//0//" + name;
            outputStream.write(text.getBytes("utf-8"));
            outputStream.flush();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    public static void sendDeleteStudent(String name, Socket socket){
        boolean can = users.erase(name);

        try {
            OutputStream outputStream = socket.getOutputStream();
            String text = "Error_DeleteS";
            if (can)
                text = "Info_DeleteS";
            outputStream.write(text.getBytes("utf-8"));
            outputStream.flush();
        }catch (Exception e){
```

```
        e.printStackTrace();
    }
}

public static void sendRegister(String name, String password, Socket socket){
    boolean can = users.insert(name, password);
    try {
        OutputStream outputStream = socket.getOutputStream();
        String text = "Info_Register//" + String.valueOf(can);
        outputStream.write(text.getBytes("utf-8"));
        outputStream.flush();
    }catch (Exception e){
        e.printStackTrace();
    }
}

public static void sendChangePassword(String name, String password, String newPassword, Socket socket){
    boolean can = users.match(name, password);
    try {
        OutputStream outputStream = socket.getOutputStream();
        if (can) {
            users.changePassword(name, newPassword);
        }
        String text = "Info_Modify//" + String.valueOf(can);
        outputStream.write(text.getBytes("utf-8"));
        outputStream.flush();
    }catch (Exception e){
        e.printStackTrace();
    }
}

public static void sendLogin(String name, String password, Socket socket){
    boolean can = users.match(name, password);

    try {
        OutputStream outputStream = socket.getOutputStream();
        String text = "Info_Login//" + String.valueOf(can);
        System.out.println(text);
        outputStream.write(text.getBytes("utf-8"));
        outputStream.flush();
    }catch (Exception e){
        e.printStackTrace();
    }
}
```

```
    public static void sendQueryStudentBook(String username, Socket socket){
        List<User.Node.Record> borrowRecord = users.getAllRecords(username);

        try {
            OutputStream outputStream = socket.getOutputStream();
            String text;
            if (borrowRecord == null){
               text = "Error_Student";
            }
            else {
                String temp = "";
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
                for (int i = 0; i < borrowRecord.size(); i++) {
                    Book.Node theBook = books.findByTitle(borrowRecord.get(i).title).next.datum;
                    temp = temp + "//" + String.valueOf(theBook.id) + "//" + theBook.title + "//" +
theBook.author + "//" + sdf.format(borrowRecord.get(i).date) + "//" +
sdf.format(getEndDate(borrowRecord.get(i).date));
                }
                text = "Book" + "//1//" + username  + "//" + String.valueOf(borrowRecord.size()) + temp;
            }
            outputStream.write(text.getBytes("utf-8"));
            outputStream.flush();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    public static void sendBorrowBook(String msg, Socket socket){
        String[] split = msg.split("//");
        String name = split[4], user = split[3];
        Book.Node theBook = books.findByTitle(name).next.datum;
        List<User.Node.Record> borrowRecord = users.getAllRecords(user);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
        Date today = new java.util.Date();
        int unDue = 0;
        for (int i = 0; i < borrowRecord.size(); i++)
            if (getEndDate(borrowRecord.get(i).date).compareTo(today) < 0) {
                unDue = 1;
                break;
            }

        try {
            OutputStream outputStream = socket.getOutputStream();
            String text = "Error_Borrow";
            if (unDue == 1) {
                text = text + "//1";
            }
            else {
                text = text + "//0";
                if (theBook.totalQuantity > theBook.borrowedQuantity && borrowRecord.size() < 5) {
                    theBook.changeBorrowedQuantity(1);
                    users.borrowBook(user, name);
                    text = "Info_Borrow" + "//" + "0" + "//" + split[4];
                }
            }
            outputStream.write(text.getBytes("utf-8"));
            outputStream.flush();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    public static void sendReturnBook(String msg, Socket socket){
        String[] split = msg.split("//");
        String name = split[4], user = split[3];
        Book.Node theBook = books.findByTitle(name).next.datum;

        try {
            OutputStream outputStream = socket.getOutputStream();
            theBook.changeBorrowedQuantity(-1);
            users.returnBook(user, name);
            String text = "Info_Return" + "//" + "0" + "//" + split[4];
            outputStream.write(text.getBytes("utf-8"));
            outputStream.flush();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    public static int maxPrefix(String a, String b) {
        int len = Math.min (a.length(), b.length());

        for (int i = 0; i < len; i++) {
            if (a.charAt(i) != b.charAt(i)) {
                return i;
            }
        }
        return len;
    }

    public static Date getEndDate(Date cur) {
        Calendar c = Calendar.getInstance();
        c.setTime(cur);   //设置时间
        c.add(Calendar.DATE, 15); //日期分钟加1,Calendar.DATE(天),Calendar.HOUR(小时)
        Date date = c.getTime(); //结果
        return date;
    }
}
```

4.2 客户端源代码

(由于代码量很大，这里只展示 JAVA 的代码，XML 的代码忽略)

AddBookActivity.java

```
package com.siang.pc.librarysystem.activity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.entity.BookInfo;
import com.siang.pc.librarysystem.entity.MyBookInfo;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;


public class AddBookActivity extends AppCompatActivity {
    private EditText etName, etAuthor, etNumber;
    private Socket socket;
    public AddBookActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_book);
        findViews();

        listenSeverMessage();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        etName = (EditText) findViewById(R.id.etName);
        etAuthor = (EditText) findViewById(R.id.etAuthor);
        etNumber = (EditText) findViewById(R.id.etNumber);
    }

    //单击添加
    public void onAdd(View view) {
        final String name = etName.getText().toString();
        final String author = etAuthor.getText().toString();
        final String number = etNumber.getText().toString();

        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Insert_B" + "//" + socket.getLocalPort() + "//2//" + name + "//"
+ author + "//" + number + "//0").getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //连接并监听服务器
    public void listenSeverMessage() {
        final Handler handler = new SeverMsgHandler();   //创建 Handler 类传递数据给主线程
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //socket 构造函数中会先尝试连接，未连接则阻塞
                    System.out.println("尝试连接。。");
                    socket = new Socket("140.143.209.173", 2222);
                    System.out.println("连接成功！");
                    InputStream inputStream = socket.getInputStream();
                    byte[] buffer = new byte[1024];
                    int len;
                    //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                    while ((len = inputStream.read(buffer)) != -1) {
                        String data = new String(buffer, 0, len);
                        // 将收到的数据发到主线程中
                        Message message = Message.obtain();
                        message.what = 1;
                        message.obj = data;
                        handler.sendMessage(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }


    protected  void onStart() {
        super.onStart();
    }

    protected void onDestroy() {
        super.onDestroy();
        if (socket != null) {
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    /**
     * 构建 handler 子类，将数据传递给主线程 UI 使用
     * Messege 类中，what 为自定义类型，obj 为传递的数据对象
     */
    private class SeverMsgHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            String data = ((String) msg.obj);
            String[] split = data.split("//");
            if (split[0].equals("Info_Insert") && split[1].equals("2")) {
                String text = getResources().getString(R.string.Info_Insert2);
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
            }
            else if (split[0].equals("Error_Insert")) {
                etName.setError(getResources().getString(R.string.booknameError));
            }
        }
    }
}
```

AdminActivity.java

```
package com.siang.pc.librarysystem.activity;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.TextView;

import com.siang.pc.librarysystem.R;


public class AdminActivity extends AppCompatActivity {
    TextView tvGreet;

    public AdminActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin);
        findViews();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        tvGreet = (TextView) findViewById(R.id.tvGreet2);
        tvGreet.setText(getUsername());
    }

    //单击管理书籍
    public void onControlBook(View view) {
        Intent intent = new Intent(this, ManagerBookActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("username",getUsername());          //传递用户名
        intent.putExtras(bundle);
        startActivity(intent);
    }

    //单击新增书籍
    public void onInsertBook(View view) {
        Intent intent = new Intent(this, AddBookActivity.class);
        startActivity(intent);
    }

    //单击查询书籍
    public void onQueryBook(View view) {
        Intent intent = new Intent(this, SearchBookActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("username",getUsername());          //传递用户名
        intent.putExtras(bundle);
        startActivity(intent);
    }

    //单击查询学生
    public void onQueryStudent(View view) {
        Intent intent = new Intent(this, SearchStudentActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("username",getUsername());          //传递用户名
        intent.putExtras(bundle);
        startActivity(intent);
    }

    //单击我的书籍
```

```
    public void onMyBook(View view) {
        Intent intent = new Intent(this, MyBookActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("username",getUsername());          //传递用户名
        intent.putExtras(bundle);
        startActivity(intent);
    }

    //单击聊天水缸
    public void onChatRoom(View view) {
        Intent intent = new Intent(this, ChatRoomActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("username",getUsername());          //传递用户名
        intent.putExtras(bundle);
        startActivity(intent);
    }

    //获取用户名
    private String getUsername() {
        Bundle bundle = getIntent().getExtras();
        return bundle.getString("username","");
    }
}
```

ChangePasswordActivity.java

```
package com.siang.pc.librarysystem.activity;


import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;


public class ChangePasswordActivity extends AppCompatActivity {
    private EditText etUsername, etPassword, etChangePassword;
    Socket socket;

    public ChangePasswordActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_change_password);
        findViews();
        listenSeverMessage();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        etUsername = (EditText) findViewById(R.id.etUsername);
        etPassword = (EditText) findViewById(R.id.etPassword);
        etChangePassword = (EditText) findViewById(R.id.etChangePassword);
    }

    //单击注册
    public void onConfirm(View view) {
        final String username = etUsername.getText().toString();
        final String password = etPassword.getText().toString();
        final String change_password = etChangePassword.getText().toString();
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Modify_S" + "//" + socket.getLocalPort() + "//0//" + username +
"//" + password + "//" + change_password).getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //连接并监听服务器
```

```
public void listenSeverMessage() {
    final Handler handler = new SeverMsgHandler();   //创建 Handler 类传递数据给主线程
    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                //socket 构造函数中会先尝试连接，未连接则阻塞
                System.out.println("尝试连接。。");
                socket = new Socket("140.143.209.173", 2222);
                System.out.println("连接成功！");

                InputStream inputStream = socket.getInputStream();
                byte[] buffer = new byte[1024];
                int len;
                //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                while ((len = inputStream.read(buffer)) != -1) {
                    String data = new String(buffer, 0, len);
                    // 将收到的数据发到主线程中
                    Message message = Message.obtain();
                    message.what = 1;
                    message.obj = data;
                    handler.sendMessage(message);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }).start();
}

/**
 * 构建 handler 子类，将数据传递给主线程 UI 使用
 * Messege 类中，what 为自定义类型，obj 为传递的数据对象
 */
private class SeverMsgHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        String data = ((String) msg.obj);
        String[] split = data.split("//");
        System.out.println(data);
        if (split[0].equals("Info_Modify")) {
            if (split[1].equals("true")) {
```

```java
                    String text = getResources().getString(R.string.Info_Modify);
                    Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                    toast.show();
                }
                else {
                    etPassword.setError(getResources().getString(R.string.passwordError));
                }
            }
        }
    }
}
```

ChatRoomActivity.java

```java
package com.siang.pc.librarysystem.activity;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.support.v4.app.NotificationCompat;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.siang.pc.librarysystem.adapter.MessageAdapter;
import com.siang.pc.librarysystem.entity.ChatMessage;
import com.siang.pc.librarysystem.helper.ChatRecordSQLiteOpenHelper;
import com.siang.pc.librarysystem.R;

import java.io.InputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.List;
import java.net.Socket;
```

```
/**
 * Created by siang on 2018/5/21.
 */

public class ChatRoomActivity extends AppCompatActivity {
    private List<ChatMessage> msgList;                                  // 消息列表
    private EditText etMessage;                                         // 输入框
    private RecyclerView msgRecyclerView;                              // recyclerview 块，滚动显示消息
    private MessageAdapter adapter;                                     // msgRcyclerview 的 adapter，设
置内容
    private Socket socket;                                              //Socket 类
    private String username;                                            //当前用户名称
    private final static int NOTIFICATION_ID = 0;                     //通知信息 ID
    private NotificationManager notificationManager;                   //通知信息管理器
    private ChatRecordSQLiteOpenHelper helper;                          //聊天记录 SQLite 辅助器

    public static final int ChatMessage_From_Server = 1;      // Message 类收到消息的 what

    public ChatRoomActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chatroom);

        if (helper == null) {
            helper = new ChatRecordSQLiteOpenHelper(ChatRoomActivity.this);
        }
        msgList = helper.getAllChatMessage();
        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        username = getUsername();

        findViews();
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        msgRecyclerView.setLayoutManager(layoutManager); // 将 msgRecyclerView 赋予 Listview 样式
        adapter = new MessageAdapter(msgList);
        msgRecyclerView.setAdapter(adapter);     // 为 msgRecyclerView 设置一个 adapter


        listenSeverMessage();
        // 更新列表
        adapter.notifyDataSetChanged();
        // 将 RecyclerView 定位到最后一行
        msgRecyclerView.scrollToPosition(msgList.size() - 1);

    }

    protected  void onStart() {
        super.onStart();
    }

    protected void onDestroy() {
        super.onDestroy();
        if (helper != null) {
            helper.close();
        }
        if (socket != null) {
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        etMessage = (EditText) findViewById(R.id.etMessage);
        msgRecyclerView = (RecyclerView) findViewById(R.id.messageRecycleView);
    }

    //获取用户名
    private String getUsername() {
        Bundle bundle = getIntent().getExtras();
        return bundle.getString("username","");
    }

    //连接并监听服务器
    public void listenSeverMessage() {
        final Handler handler = new ChatMsgHandler();   //创建 Handler 类传递数据给主线程
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //socket 构造函数中会先尝试连接，未连接则阻塞
                    System.out.println("尝试连接。。");
                    socket = new Socket("140.143.209.173", 2222);
```

```
                    System.out.println("连接成功！");
                    InputStream inputStream = socket.getInputStream();
                    byte[] buffer = new byte[1024];
                    int len;
                    //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                    while ((len = inputStream.read(buffer)) != -1) {
                        String data = new String(buffer, 0, len);
                        // 将收到的数据发到主线程中
                        Message message = Message.obtain();
                        message.what = 1;
                        message.obj = data;
                        handler.sendMessage(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //点击发送按钮触发
    public void onSentMessage(View view) {
        final String data = etMessage.getText().toString().trim();
        // 清空输入框中的内容
        etMessage.setText(null);
        if (data.isEmpty()) {
            //若输入为空则 Toast 出不能为空的提示
            Toast toast = Toast.makeText(getApplicationContext(), R.string.inputEmpty, Toast.LENGTH_SHORT);
            toast.show();
            return;
        } else {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        //向服务器端用输出流输出消息
                        OutputStream outputStream = socket.getOutputStream();
                        outputStream.write(("Message" + "//" + socket.getLocalPort() + "//" + username +
"//" + data).getBytes("utf-8"));
                        //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                        outputStream.flush();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }).start();
        }
    }

    @Override
    //创建标题栏菜单
    public boolean onCreateOptionsMenu(Menu menu) {
        //inflater 使用 options_menu 的 item 项目
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.options_menu, menu);
        return true;
    }

    @Override
    //设置标题栏菜单选项
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            // 如果点击了删除全部
            case R.id.deleteMessage:
                //清除消息列表中所有元素
                msgList.clear();
                //更新 RecyclerView
                adapter.notifyDataSetChanged();
                helper.deleteAllChatMessage();;
                break;
            case R.id.showOnlineNumber:
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            //向服务器端用输出流输出消息
                            OutputStream outputStream = socket.getOutputStream();
                            outputStream.write(("OnlineNumber" + "//" +
socket.getLocalPort()).getBytes("utf-8"));
                            //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出
去
                            outputStream.flush();
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                }).start();
                break;
            default:
                return super.onContextItemSelected(item);
        }
        return true;
    }

    /**
     * 构建 handler 子类，将数据传递给主线程 UI 使用
     * Messege 类中，what 为自定义类型，obj 为传递的数据对象
    */
    private class ChatMsgHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            if (msg.what == ChatMessage_From_Server) {
                int localPort = socket.getLocalPort();
                String[] split = ((String) msg.obj).split("//");
                //判断消息来源是否为本身
                //0---消息类型  1---端口号  2---用户名   3--消息内容
                if (split[0].equals("Message")) {
                    if (split[1].equals(localPort + "")) {
                        ChatMessage chatmsg = new ChatMessage(split[2], split[3], ChatMessage.TYPE_SENT);
                        System.out.println("收到自己消息");
                        helper.insertChatMessage(chatmsg);
                        msgList.add(chatmsg);
                    } else {
                        ChatMessage chatmsg = new ChatMessage(split[2], split[3],
ChatMessage.TYPE_RECEIVED);
                        System.out.println("收到他人消息");
                        helper.insertChatMessage(chatmsg);
                        msgList.add(chatmsg);

                        //设置通知信
                        // 消息样式
                        Intent intent = new Intent(ChatRoomActivity.this, ChatRoomActivity.class);
                        PendingIntent pendingIntent = PendingIntent.getActivity(ChatRoomActivity.this, 0,
intent, PendingIntent.FLAG_UPDATE_CURRENT);
                        Notification notification = new
NotificationCompat.Builder( ChatRoomActivity.this,"default")
                                .setContentTitle(split[2])
                                .setContentText(split[3])
                                .setSmallIcon(R.mipmap.ic_launcher)
                                .setAutoCancel(true)
```

```java
                        .setContentIntent(pendingIntent)
                        .build();
                    //显示通知信息
                    notificationManager.notify(NOTIFICATION_ID, notification);
                }

                System.out.println("消息数目："+msgList.size());
                // 当有新消息时，更新列表最后的位置上的数据可以调用
                adapter.notifyItemInserted(msgList.size() - 1);
                // 将 RecyclerView 定位到最后一行
                msgRecyclerView.scrollToPosition(msgList.size() - 1);
            }
            else if (split[0].equals("OnlineNumber")) {
                String text = "当前人数为" + split[1] + "人";
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
            }
        }
    }
  }
}
```

LoginActivity.java

```java
package com.siang.pc.librarysystem.activity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;


public class LoginActivity extends AppCompatActivity {
    private EditText etUsername, etPassword;
    Socket socket;
    Intent intentVIP;
    Intent intent;

    public LoginActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        findViews();
        intentVIP = new Intent(this, AdminActivity.class);
        intent = new Intent(this, NormalActivity.class);
        listenSeverMessage();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        etUsername = (EditText) findViewById(R.id.etUsername);
        etPassword = (EditText) findViewById(R.id.etPassword);
    }

    //单击登录
    public void onLogin(View view) {
        final String username = etUsername.getText().toString();
        final String password = etPassword.getText().toString();
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Login_S" + "//" + socket.getLocalPort() + "//0//" + username + "//"
+ password + "//0").getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
    //单击注册
    public void onRegister(View view) {
        Intent intent = new Intent(this, RegisterActivity.class);
        startActivity(intent);
    }

    //单击重置密码
    public void onChangePassword(View view) {
        Intent intent = new Intent(this, ChangePasswordActivity.class);
        startActivity(intent);
    }

    //连接并监听服务器
    public void listenSeverMessage() {
        final Handler handler = new SeverMsgHandler();   //创建 Handler 类传递数据给主线程
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //socket 构造函数中会先尝试连接，未连接则阻塞
                    System.out.println("尝试连接。。");
                    socket = new Socket("140.143.209.173", 2222);
                    System.out.println("连接成功！");
                    InputStream inputStream = socket.getInputStream();
                    byte[] buffer = new byte[1024];
                    int len;
                    //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                    while ((len = inputStream.read(buffer)) != -1) {
                        String data = new String(buffer, 0, len);
                        // 将收到的数据发到主线程中
                        Message message = Message.obtain();
                        message.what = 1;
                        message.obj = data;
                        handler.sendMessage(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    /**
```

```
     * 构建 handler 子类，将数据传递给主线程 UI 使用
     * Messege 类中，what 为自定义类型，obj 为传递的数据对象
     */
    private class SeverMsgHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            final String username = etUsername.getText().toString();
            String data = ((String) msg.obj);
            String[] split = data.split("//");
            if (split[0].equals("Info_Login")) {
                //密码正确跳转至聊天页面
                if (split[1].matches("true")) {
                    if (username.equals("root") || username.equals("admin")) {
                        Bundle bundle = new Bundle();
                        bundle.putString("username",username);          //传递用户名
                        intentVIP.putExtras(bundle);
                        startActivity(intentVIP);
                    }
                    else{
                        Bundle bundle = new Bundle();
                        bundle.putString("username",username);          //传递用户名
                        intent.putExtras(bundle);
                        startActivity(intent);
                    }
                }
                else {
                    etPassword.setError(getResources().getString(R.string.passwordError));
                    return;
                }
            }
        }
    }
}
```

ManagerBookActivity.java

```
package com.siang.pc.librarysystem.activity;


import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.adapter.BookListAdapter;
import com.siang.pc.librarysystem.adapter.ManagerBookListAdapter;
import com.siang.pc.librarysystem.entity.BookInfo;
import com.siang.pc.librarysystem.entity.MyBookInfo;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class ManagerBookActivity extends AppCompatActivity {
    private List<BookInfo> bookList;                                  // 消息列表
    private EditText etSearch;
    private Socket socket;
    private ManagerBookListAdapter adapter;                                  // msgRcyclerview 的
adapter，设置内容
    private RecyclerView msgRecyclerView;                             // recyclerview 块，滚动显示消息
    private RadioGroup rgWay;
    private String check;
    private String username;
    private String way = "2";
    private String searchString = "";

    public ManagerBookActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_manage_book);
        check = getResources().getString(R.string.AuthorofBook);
        username = getUsername();
        findViews();

        bookList = new ArrayList<>();
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        msgRecyclerView.setLayoutManager(layoutManager); // 将 msgRecyclerView 赋予 Listview 样式
        adapter = new ManagerBookListAdapter(bookList, this);
        msgRecyclerView.setAdapter(adapter);    // 为 msgRecyclerView 设置一个 adapter

        listenSeverMessage();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        etSearch = (EditText) findViewById(R.id.etSearch);
        msgRecyclerView = (RecyclerView) findViewById(R.id.bookRecycleView);
        rgWay = (RadioGroup) findViewById(R.id.radioGroup);

        rgWay.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup radioGroup, int i) {
                RadioButton radioButton = (RadioButton) radioGroup.findViewById(i);
                check = radioButton.getText().toString();
            }
        });
    }

    //单击搜索
    public void onSearch(View view) {
        final String search = etSearch.getText().toString();
        if (search.equals("")) {
            String text = getResources().getString(R.string.Error_Empty);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
        }
        else {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        if (check.equals(getResources().getString(R.string.IDofBook))) {
                            way = "0";
                        }
                        else if (check.equals(getResources().getString(R.string.NameofBook))) {
                            way = "1";
                        }
                        else way = "2";
                        searchString = search;
                        //向服务器端用输出流输出消息
                        OutputStream outputStream = socket.getOutputStream();
                        outputStream.write(("Query_B" + "//" + socket.getLocalPort() + "//" + way + "//"
+ etSearch.getText().toString() + "//" + username).getBytes("utf-8"));
                        //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                        outputStream.flush();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }).start();
        }
    }

    //连接并监听服务器
    public void listenSeverMessage() {
        final Handler handler = new SeverMsgHandler();   //创建 Handler 类传递数据给主线程
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //socket 构造函数中会先尝试连接，未连接则阻塞
                    System.out.println("尝试连接。。");
                    socket = new Socket("140.143.209.173", 2222);
                    System.out.println("连接成功！");
                    adapter.setSocket(socket);
                    adapter.setUsername(getUsername());
                    InputStream inputStream = socket.getInputStream();
                    byte[] buffer = new byte[1024];
                    int len;
                    //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                    while ((len = inputStream.read(buffer)) != -1) {
                        String data = new String(buffer, 0, len);
                        // 将收到的数据发到主线程中
                        Message message = Message.obtain();
                        message.what = 1;
                        message.obj = data;
                        handler.sendMessage(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    protected  void onStart() {
        super.onStart();
    }

    protected void onDestroy() {
        super.onDestroy();
        if (socket != null) {
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * 构建 handler 子类，将数据传递给主线程 UI 使用
     * Messege 类中，what 为自定义类型，obj 为传递的数据对象
     */
    private class SeverMsgHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            String data = ((String) msg.obj);
            String[] split = data.split("//");
            if (split[0].equals("Book") && split[1].equals("0")) {
                adapter.notifyItemRangeRemoved(0, bookList.size());
                bookList.clear();
                adapter.setWay(way);
                adapter.setSearch(searchString);
                List<MyBookInfo> myBookList =  new ArrayList<>();
                int number_myborrow = Integer.parseInt(split[2]), j = 3;
                for (int i = 1; i <= number_myborrow; i++, j += 5) {
                    MyBookInfo book = new MyBookInfo(split[j], split[j + 1], split[j + 2], split[j + 3],
split[j + 4]);
                    myBookList.add(book);
                }
                int number_book = Integer.parseInt(split[j]);
                j += 1;
                for (int i = 1; i <= number_book && j + 4 < split.length; i++, j += 5) {
                    int type, total, borrow;
                    total = Integer.parseInt(split[j + 3].replace("/",""));
                    borrow = Integer.parseInt(split[j + 4].replace("/",""));
                    if (total == borrow)
                        type = 2;
                    else
                        type = 0;
                    for (int k = 0; k < myBookList.size(); k++)
                        if (myBookList.get(k).getId().equals(split[j])) {
                            type = 1;
                            break;
                        }
                    BookInfo book = new BookInfo(split[j], split[j + 1], split[j + 2], split[j +
3].replace("/",""), split[j + 4].replace("/",""), type);
                    bookList.add(book);
                }
                // 当有新消息时，更新列表最后的位置上的数据可以调用
                adapter.notifyItemInserted(bookList.size() - 1);
            }
            else if (split[0].equals("Book_next") && split[1].equals("0")) {
                System.out.println(data);
                List<MyBookInfo> myBookList =  new ArrayList<>();
                int number_myborrow = Integer.parseInt(split[2]), j = 3;
                for (int i = 1; i <= number_myborrow; i++, j += 5) {
                    MyBookInfo book = new MyBookInfo(split[j], split[j + 1], split[j + 2], split[j + 3],
split[j + 4]);
                    myBookList.add(book);
                }
                int number_book = Integer.parseInt(split[j]);
                j += 1;
                for (int i = 1; i <= number_book && j + 4 < split.length; i++, j += 5) {
                    int type, total, borrow;
                    total = Integer.parseInt(split[j + 3].replace("/",""));
                    borrow = Integer.parseInt(split[j + 4].replace("/",""));
                    if (total == borrow)
                        type = 2;
                    else
                        type = 0;
                    for (int k = 0; k < myBookList.size(); k++)
                        if (myBookList.get(k).getId().equals(split[j])) {
                            type = 1;
                            break;
                        }
                    BookInfo book = new BookInfo(split[j], split[j + 1], split[j + 2], split[j +
3].replace("/",""), split[j + 4].replace("/",""), type);
```

```
                bookList.add(book);
            }
            // 当有新消息时，更新列表最后的位置上的数据可以调用
            adapter.notifyItemInserted(bookList.size() - 1);
        }
        else if(split[0].equals("Info_Insert") && split[1].equals("0")) {
            String text = getResources().getString(R.string.Info_Insert0);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
            String bookName = split[2];
            int pos = -1;
            for(int i = 0; i < bookList.size(); i++)
                if (bookList.get(i).getName().equals(bookName)) {
                    pos = i;
                    break;
                }
            int have_num = Integer.parseInt(bookList.get(pos).getBookHave()) +
Integer.parseInt(split[3]);
            bookList.get(pos).setBookHave(String.valueOf(have_num));
            adapter.notifyItemChanged(pos);
        }
        else if (split[0].equals("Error_Query")) {
            adapter.notifyItemRangeRemoved(0, bookList.size());
            bookList.clear();
            String text = getResources().getString(R.string.Error_Query);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
        }
        else if (split[0].equals("Error_Query_next")) {
            String text = getResources().getString(R.string.Error_Query_next);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
        }
        else if(split[0].equals("Info_Insert") && split[1].equals("1")) {
            String text = getResources().getString(R.string.Info_Insert1);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
            String bookName = split[2];
            int pos = -1;
            for(int i = 0; i < bookList.size(); i++)
                if (bookList.get(i).getName().equals(bookName)) {
                    pos = i;
                    break;
                }
            int have_num = Integer.parseInt(bookList.get(pos).getBookHave()) -
Integer.parseInt(split[3]);
            bookList.get(pos).setBookHave(String.valueOf(have_num));
            adapter.notifyItemChanged(pos);
        }
        else if (split[0].equals("Error_Insert1")) {
            String text = getResources().getString(R.string.Error_Insert1);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
        }
        else if (split[0].equals("Error_DeleteB")) {
            String text = getResources().getString(R.string.Error_DeleteB);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
        }
        else if (split[0].equals("Info_DeleteB")) {
            String text = getResources().getString(R.string.Info_DeleteB);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
            String bookName = split[2];
            int pos = -1;
            for(int i = 0; i < bookList.size(); i++)
                if (bookList.get(i).getName().equals(bookName)) {
                    pos = i;
                    break;
                }
            bookList.remove(pos);
            adapter.notifyItemRemoved(pos);
        }
    }
}

//获取用户名
private String getUsername() {
    Bundle bundle = getIntent().getExtras();
    return bundle.getString("username","");
}
}
```

MyBookActivity.java

```
package com.siang.pc.librarysystem.activity;


import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.adapter.MyBookListAdapter;
import com.siang.pc.librarysystem.entity.MyBookInfo;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class MyBookActivity extends AppCompatActivity {
    private List<MyBookInfo> mybookList;                              // 消息列表
    private TextView tvNumber;
    private Socket socket;
    private String username;
    private MyBookListAdapter adapter;                                  // msgRcyclerview 的 adapter，
设置内容
    private RecyclerView msgRecyclerView;                         // recyclerview 块，滚动显示消息

    public MyBookActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mybook);
        username = getUsername();
        findViews();

        mybookList = new ArrayList<>();
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        msgRecyclerView.setLayoutManager(layoutManager); // 将 msgRecyclerView 赋予 Listview 样式
        adapter = new MyBookListAdapter(mybookList, this);
        msgRecyclerView.setAdapter(adapter);    // 为 msgRecyclerView 设置一个 adapter
        listenSeverMessage();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        msgRecyclerView = (RecyclerView) findViewById(R.id.bookRecycleView);
        tvNumber = (TextView) findViewById(R.id.number);
    }

    public void searchMyBook() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Query_B" + "//" + socket.getLocalPort() + "//3//" + username +
"//0//0").getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //连接并监听服务器
    public void listenSeverMessage() {
        final Handler handler = new SeverMsgHandler();   //创建 Handler 类传递数据给主线程
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //socket 构造函数中会先尝试连接，未连接则阻塞
                    System.out.println("尝试连接。。");
                    socket = new Socket("140.143.209.173", 2222);
                    System.out.println("连接成功！");
                    adapter.setSocket(socket);
                    adapter.setUsername(username);
                    searchMyBook();
                    InputStream inputStream = socket.getInputStream();
                    byte[] buffer = new byte[1024];
                    int len;
                    //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                    while ((len = inputStream.read(buffer)) != -1) {
                        String data = new String(buffer, 0, len);
                        // 将收到的数据发到主线程中
                        Message message = Message.obtain();
                        message.what = 1;
                        message.obj = data;
                        handler.sendMessage(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //获取用户名
    private String getUsername() {
        Bundle bundle = getIntent().getExtras();
        return bundle.getString("username","");
    }

    protected  void onStart() {
        super.onStart();
    }

    protected void onDestroy() {
        super.onDestroy();
        if (socket != null) {
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * 构建 handler 子类，将数据传递给主线程 UI 使用
     * Messege 类中，what 为自定义类型，obj 为传递的数据对象
     */
    private class SeverMsgHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            String data = ((String) msg.obj);
            String[] split = data.split("//");
            if (split[0].equals("Book") && split[1].equals("1")) {
                adapter.notifyItemRangeRemoved(0, mybookList.size());
                mybookList.clear();
                if (split[2].equals(username))
                    adapter.setSelf(0);                //0 为是自己  1 为是他人
                else
                    adapter.setSelf(1);
                int number = Integer.parseInt(split[3]), j = 4;
                tvNumber.setText(split[3]);
                for (int i = 1; i <= number; i++, j += 5) {
                    MyBookInfo book = new MyBookInfo(split[j], split[j + 1], split[j + 2], split[j + 3],
split[j + 4]);
                    mybookList.add(book);
                }
                // 当有新消息时，更新列表最后的位置上的数据可以调用
                adapter.notifyItemInserted(mybookList.size() - 1);
            }
            else if(split[0].equals("Info_Return")) {
                String text = getResources().getString(R.string.Info_Return);
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
                String bookName = split[2];
                int pos = -1;
                for(int i = 0; i < mybookList.size(); i++)
                    if (mybookList.get(i).getName().equals(bookName)) {
                        pos = i;
                        break;
                    }
                mybookList.remove(pos);
                tvNumber.setText(String.valueOf(Integer.parseInt((String)tvNumber.getText()) - 1));
                adapter.notifyItemRemoved(pos);
            }
        }
    }
}
```

NormalActivity.java

```
package com.siang.pc.librarysystem.activity;
```

```java
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.TextView;

import com.siang.pc.librarysystem.R;


public class NormalActivity extends AppCompatActivity {
    TextView tvGreet;

    public NormalActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_normal);
        findViews();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        tvGreet = (TextView) findViewById(R.id.tvGreet2);
        tvGreet.setText(getUsername());
    }

    //单击查询书籍
    public void onQueryBook(View view) {
        Intent intent = new Intent(this, SearchBookActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("username",getUsername());          //传递用户名
        intent.putExtras(bundle);
        startActivity(intent);
    }

    //单击我的书籍
    public void onMyBook(View view) {
        Intent intent = new Intent(this, MyBookActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("username",getUsername());          //传递用户名
        intent.putExtras(bundle);
        startActivity(intent);
    }

    //单击聊天水缸
    public void onChatRoom(View view) {
        Intent intent = new Intent(this, ChatRoomActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("username",getUsername());          //传递用户名
        intent.putExtras(bundle);
        startActivity(intent);
    }

    //获取用户名
    private String getUsername() {
        Bundle bundle = getIntent().getExtras();
        return bundle.getString("username","");
    }
}
```

RegisterActivity.java

```
package com.siang.pc.librarysystem.activity;

import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.entity.BookInfo;
import com.siang.pc.librarysystem.entity.MyBookInfo;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;


public class RegisterActivity extends AppCompatActivity {
    private EditText etUsername, etPassword, etPasswordConfirm;
    Socket socket;
    public RegisterActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        findViews();
        listenSeverMessage();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        etUsername = (EditText) findViewById(R.id.etUsername);
        etPassword = (EditText) findViewById(R.id.etPassword);
        etPasswordConfirm = (EditText) findViewById(R.id.etConfirmPassword);
    }

    //单击注册
    public void onRegister(View view) {
        final String username = etUsername.getText().toString();
        final String password = etPassword.getText().toString();
        String password_confirm = etPasswordConfirm.getText().toString();
        //两次密码输入一致
        if (password.equals(password_confirm)) {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        //向服务器端用输出流输出消息
                        OutputStream outputStream = socket.getOutputStream();
                        outputStream.write(("Register_S" + "//" + socket.getLocalPort() + "//0//" +
username + "//" + password).getBytes("utf-8"));
                        //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                        outputStream.flush();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }).start();
        }
        else {
            etPasswordConfirm.setError(getResources().getString(R.string.passwordConfirmError));
```

```
                return;
            }
        }

        //连接并监听服务器
        public void listenSeverMessage() {
            final Handler handler = new SeverMsgHandler();   //创建 Handler 类传递数据给主线程
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        //socket 构造函数中会先尝试连接，未连接则阻塞
                        System.out.println("尝试连接。。");
                        socket = new Socket("140.143.209.173", 2222);
                        System.out.println("连接成功！");

                        InputStream inputStream = socket.getInputStream();
                        byte[] buffer = new byte[1024];
                        int len;
                        //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                        while ((len = inputStream.read(buffer)) != -1) {
                            String data = new String(buffer, 0, len);
                            // 将收到的数据发到主线程中
                            Message message = Message.obtain();
                            message.what = 1;
                            message.obj = data;
                            handler.sendMessage(message);
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }).start();
        }

        /**
         * 构建 handler 子类，将数据传递给主线程 UI 使用
         * Messege 类中，what 为自定义类型，obj 为传递的数据对象
         */
        private class SeverMsgHandler extends Handler {
            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                String data = ((String) msg.obj);
                String[] split = data.split("//");
                if (split[0].equals("Info_Register")) {
                    if (split[1].equals("true")) {
                        String text = getResources().getString(R.string.Info_Register);
                        Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                        toast.show();
                    }
                    else {
                        etUsername.setError(getResources().getString(R.string.usernameError));
                    }
                }
            }
        }
}
```

SearchBookActivity.java

```
package com.siang.pc.librarysystem.activity;

import android.content.Intent;
import android.os.Handler;
import android.os.Message;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.adapter.BookListAdapter;
import com.siang.pc.librarysystem.adapter.MessageAdapter;
import com.siang.pc.librarysystem.entity.BookInfo;
import com.siang.pc.librarysystem.entity.MyBookInfo;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class SearchBookActivity extends AppCompatActivity {
    private List<BookInfo> bookList;                                    // 消息列表
    private EditText etSearch;
    private Socket socket;
    private BookListAdapter adapter;                                       // msgRcyclerview 的 adapter,
设置内容
    private RecyclerView msgRecyclerView;                                // recyclerview 块，滚动显示消息
    private RadioGroup rgWay;
    private String check;
    private String username;
    private String way = "2";
    private String searchString = "";

    public SearchBookActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search_book);
        check = getResources().getString(R.string.AuthorofBook);
        username = getUsername();
        findViews();

        bookList = new ArrayList<>();
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        msgRecyclerView.setLayoutManager(layoutManager); // 将 msgRecyclerView 赋予 Listview 样式
        adapter = new BookListAdapter(bookList, this);
        msgRecyclerView.setAdapter(adapter);    // 为 msgRecyclerView 设置一个 adapter

        listenSeverMessage();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        etSearch = (EditText) findViewById(R.id.etSearch);
        msgRecyclerView = (RecyclerView) findViewById(R.id.bookRecycleView);
        rgWay = (RadioGroup) findViewById(R.id.radioGroup);

        rgWay.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup radioGroup, int i) {
                RadioButton radioButton = (RadioButton) radioGroup.findViewById(i);
                check = radioButton.getText().toString();
            }
        });
    }

    //单击搜索
    public void onSearch(View view) {
        final String search = etSearch.getText().toString();
        if (search.equals("")) {
            String text = getResources().getString(R.string.Error_Empty);
            Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
            toast.show();
        }
        else {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        if (check.equals(getResources().getString(R.string.IDofBook))) {
                            way = "0";
                        }
                        else if (check.equals(getResources().getString(R.string.NameofBook))) {
                            way = "1";
                        }
                        else way = "2";
                        searchString = search;
                        //向服务器端用输出流输出消息
                        OutputStream outputStream = socket.getOutputStream();
                        outputStream.write(("Query_B" + "//" + socket.getLocalPort() + "//" + way + "//"
+ etSearch.getText().toString() + "//" + username).getBytes("utf-8"));
                        //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                        outputStream.flush();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }).start();
        }
    }

    //连接并监听服务器
```

```
public void listenSeverMessage() {
    final Handler handler = new SeverMsgHandler();   //创建 Handler 类传递数据给主线程
    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                //socket 构造函数中会先尝试连接，未连接则阻塞
                System.out.println("尝试连接。。");
                socket = new Socket("140.143.209.173", 2222);
                System.out.println("连接成功！");
                adapter.setSocket(socket);
                adapter.setUsername(getUsername());
                InputStream inputStream = socket.getInputStream();
                byte[] buffer = new byte[1024];
                int len;
                //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                while ((len = inputStream.read(buffer)) != -1) {
                    String data = new String(buffer, 0, len);
                    // 将收到的数据发到主线程中
                    Message message = Message.obtain();
                    message.what = 1;
                    message.obj = data;
                    handler.sendMessage(message);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }).start();
}

protected  void onStart() {
    super.onStart();
}

protected void onDestroy() {
    super.onDestroy();
    if (socket != null) {
        try {
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

/**
 * 构建 handler 子类，将数据传递给主线程 UI 使用
 * Messege 类中，what 为自定义类型，obj 为传递的数据对象
 */
private class SeverMsgHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        String data = ((String) msg.obj);
        String[] split = data.split("//");
        if (split[0].equals("Book") && split[1].equals("0")) {
            System.out.println(data);
            adapter.notifyItemRangeRemoved(0, bookList.size());
            bookList.clear();
            adapter.setWay(way);
            adapter.setSearch(searchString);
            List<MyBookInfo> myBookList =  new ArrayList<>();
            int number_myborrow = Integer.parseInt(split[2]), j = 3;
            for (int i = 1; i <= number_myborrow; i++, j += 5) {
                MyBookInfo book = new MyBookInfo(split[j], split[j + 1], split[j + 2], split[j + 3],
split[j + 4]);
                myBookList.add(book);
            }
            int number_book = Integer.parseInt(split[j]);
            j += 1;
            for (int i = 1; i <= number_book && j + 4 < split.length; i++, j += 5) {
                //System.out.println(split[j + 1]);
                int type, total, borrow;
                total = Integer.parseInt(split[j + 3].replace("/",""));
                borrow = Integer.parseInt(split[j + 4].replace("/",""));
                if (total == borrow)
                    type = 2;
                else
                    type = 0;
                for (int k = 0; k < myBookList.size(); k++)
                    if (myBookList.get(k).getId().equals(split[j])) {
                        type = 1;
                        break;
                    }
                BookInfo book = new BookInfo(split[j], split[j + 1], split[j + 2], split[j +
3].replace("/",""), split[j + 4].replace("/",""), type);
```

```
                    bookList.add(book);
                }
                // 当有新消息时，更新列表最后的位置上的数据可以调用
                adapter.notifyItemInserted(bookList.size() - 1);
            }
            else if (split[0].equals("Book_next") && split[1].equals("0")) {
                System.out.println(data);
                List<MyBookInfo> myBookList =  new ArrayList<>();
                int number_myborrow = Integer.parseInt(split[2]), j = 3;
                for (int i = 1; i <= number_myborrow; i++, j += 5) {
                    MyBookInfo book = new MyBookInfo(split[j], split[j + 1], split[j + 2], split[j + 3],
split[j + 4]);
                    myBookList.add(book);
                }
                int number_book = Integer.parseInt(split[j]);
                j += 1;
                for (int i = 1; i <= number_book && j + 4 < split.length; i++, j += 5) {
                    //System.out.println(split[j + 1]);
                    int type, total, borrow;
                    total = Integer.parseInt(split[j + 3].replace("/",""));
                    borrow = Integer.parseInt(split[j + 4].replace("/",""));
                    if (total == borrow)
                        type = 2;
                    else
                        type = 0;
                    for (int k = 0; k < myBookList.size(); k++)
                        if (myBookList.get(k).getId().equals(split[j])) {
                            type = 1;
                            break;
                        }
                    BookInfo book = new BookInfo(split[j], split[j + 1], split[j + 2], split[j +
3].replace("/",""), split[j + 4].replace("/",""), type);
                    bookList.add(book);
                }
                // 当有新消息时，更新列表最后的位置上的数据可以调用
                adapter.notifyItemInserted(bookList.size() - 1);
            }
            else if(split[0].equals("Info_Borrow")) {
                String text = getResources().getString(R.string.Info_Borrow);
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
                String bookName = split[2];
                int pos = -1;
                for(int i = 0; i < bookList.size(); i++)
```

```
        if (bookList.get(i).getName().equals(bookName)) {
            pos = i;
            break;
        }
    int borrow_num = Integer.parseInt(bookList.get(pos).getBookBorrow()) + 1;
    bookList.get(pos).setBookBorrow(String.valueOf(borrow_num));
    bookList.get(pos).setType(1);                    //0 为借阅, 1 为归还
    adapter.notifyItemChanged(pos);
}
else if(split[0].equals("Info_Return")) {
    String text = getResources().getString(R.string.Info_Return);
    Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
    toast.show();
    String bookName = split[2];
    int pos = -1;
    for(int i = 0; i < bookList.size(); i++)
        if (bookList.get(i).getName().equals(bookName)) {
            pos = i;
            break;
        }
    int borrow_num = Integer.parseInt(bookList.get(pos).getBookBorrow()) - 1;
    bookList.get(pos).setBookBorrow(String.valueOf(borrow_num));
    bookList.get(pos).setType(0);                    //0 为借阅, 1 为归还
    adapter.notifyItemChanged(pos);
}
else if (split[0].equals("Error_Query")) {
    adapter.notifyItemRangeRemoved(0, bookList.size());
    bookList.clear();
    String text = getResources().getString(R.string.Error_Query);
    Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
    toast.show();
}
else if (split[0].equals("Error_Query")) {
    adapter.notifyItemRangeRemoved(0, bookList.size());
    bookList.clear();
    String text = getResources().getString(R.string.Error_Query);
    Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
    toast.show();
}
else if (split[0].equals("Error_Query_next")) {
    String text = getResources().getString(R.string.Error_Query_next);
    Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
    toast.show();
}
```

```
            else if (split[0].equals("Error_Borrow")) {
                String text;
                if (split[1].equals("0")) {
                    text = getResources().getString(R.string.Error_Borrow0);
                }
                else {
                    text = getResources().getString(R.string.Error_Borrow1);
                }
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
            }
        }
    }

    //获取用户名
    private String getUsername() {
        Bundle bundle = getIntent().getExtras();
        return bundle.getString("username","");
    }
}
```

SearchStudentActivity.java

```
package com.siang.pc.librarysystem.activity;

import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.adapter.MyBookListAdapter;
import com.siang.pc.librarysystem.entity.BookInfo;
import com.siang.pc.librarysystem.entity.MyBookInfo;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class SearchStudentActivity extends AppCompatActivity {
    private List<MyBookInfo> mybookList;                                   // 消息列表
    private EditText etSearch;
    private TextView tvNumber;
    private Socket socket;
    private String username;
    private MyBookListAdapter adapter;                                      // msgRcyclerview 的 adapter，
设置内容
    private RecyclerView msgRecyclerView;                                 // recyclerview 块，滚动显示消息
    private String searchUser;
    private TextView btDelete;

    public SearchStudentActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search_student);
        username = getUsername();
        findViews();

        mybookList = new ArrayList<>();
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        msgRecyclerView.setLayoutManager(layoutManager); // 将 msgRecyclerView 赋予 Listview 样式
        adapter = new MyBookListAdapter(mybookList, this);
        msgRecyclerView.setAdapter(adapter);     // 为 msgRecyclerView 设置一个 adapter

        listenSeverMessage();
    }

    //通过 findViewById 将变量指向对应布局
    private void findViews() {
        etSearch = (EditText) findViewById(R.id.etSearch);
        btDelete = (TextView) findViewById(R.id.btDelete);
        btDelete.setVisibility(View.INVISIBLE);
        msgRecyclerView = (RecyclerView) findViewById(R.id.bookRecycleView);
        tvNumber = (TextView) findViewById(R.id.number);
    }
    //单击删除
    public void onDelete(View view) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Delete_S" + "//" + socket.getLocalPort() + "//" + 0 + "//" +
searchUser).getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //单击搜索
    public void onSearch(View view) {
        final String search = etSearch.getText().toString();

        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Query_B" + "//" + socket.getLocalPort() + "//3//" + search +
"//0//0").getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //连接并监听服务器
    public void listenSeverMessage() {
        final Handler handler = new SeverMsgHandler();    //创建 Handler 类传递数据给主线程
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //socket 构造函数中会先尝试连接，未连接则阻塞
                    System.out.println("尝试连接。。");
                    socket = new Socket("140.143.209.173", 2222);
                    System.out.println("连接成功！");
                    adapter.setSocket(socket);
                    adapter.setUsername(username);
                    InputStream inputStream = socket.getInputStream();
                    byte[] buffer = new byte[1024];
                    int len;
                    //inputStream.read()在文件结束返回-1，未收到消息为阻塞状态，所以当未收到服务器消息时
一直阻塞在此
                    while ((len = inputStream.read(buffer)) != -1) {
                        String data = new String(buffer, 0, len);
                        // 将收到的数据发到主线程中
                        Message message = Message.obtain();
                        message.what = 1;
                        message.obj = data;
                        handler.sendMessage(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //获取用户名
    private String getUsername() {
        Bundle bundle = getIntent().getExtras();
        return bundle.getString("username","");
    }

    protected  void onStart() {
        super.onStart();
    }

    protected void onDestroy() {
        super.onDestroy();
        if (socket != null) {
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * 构建 handler 子类，将数据传递给主线程 UI 使用
     * Messege 类中，what 为自定义类型，obj 为传递的数据对象
     */
    private class SeverMsgHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            String data = ((String) msg.obj);
            String[] split = data.split("//");
            if (split[0].equals("Book") && split[1].equals("1")) {
                adapter.notifyItemRangeRemoved(0, mybookList.size());
                mybookList.clear();
                searchUser = etSearch.getText().toString();
                if (searchUser.equals(username))
                    btDelete.setVisibility(View.INVISIBLE);
                else
                    btDelete.setVisibility(View.VISIBLE);
                if (split[2].equals(username))
                    adapter.setSelf(0);                //0 为是自己  1 为是他人
                else
                    adapter.setSelf(1);
                int number = Integer.parseInt(split[3]), j = 4;
                tvNumber.setText(split[3]);
                for (int i = 1; i <= number; i++, j += 5) {
                    MyBookInfo book = new MyBookInfo(split[j], split[j + 1], split[j + 2], split[j + 3],
split[j + 4]);
                    mybookList.add(book);
                }
                // 当有新消息时，更新列表最后的位置上的数据可以调用
                adapter.notifyItemInserted(mybookList.size() - 1);
            }
            else if(split[0].equals("Info_Return")) {
                String text = getResources().getString(R.string.Info_Return);
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
                String bookName = split[2];
                int pos = -1;
                for(int i = 0; i < mybookList.size(); i++)
                    if (mybookList.get(i).getName().equals(bookName)) {
                        pos = i;
                        break;
                    }
                mybookList.remove(pos);
                tvNumber.setText(String.valueOf(Integer.parseInt((String)tvNumber.getText()) - 1));
                adapter.notifyItemRemoved(pos);
            }
            else if(split[0].equals("Error_Student")) {
                adapter.notifyItemRangeRemoved(0, mybookList.size());
                mybookList.clear();
                btDelete.setVisibility(View.INVISIBLE);
                String text = getResources().getString(R.string.Error_Student);
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
            }
            else if(split[0].equals("Error_DeleteS")) {
                String text = getResources().getString(R.string.Error_DeleteS);
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
            }
            else if(split[0].equals("Info_DeleteS")) {
                adapter.notifyItemRangeRemoved(0, mybookList.size());
                mybookList.clear();
                btDelete.setVisibility(View.INVISIBLE);
                String text = getResources().getString(R.string.Info_DeleteS);
                Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
                toast.show();
            }
        }
    }
}
```

BookListAdapter.java

```
package com.siang.pc.librarysystem.adapter;

import android.content.Context;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
```

```java
import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.entity.BookInfo;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.util.List;

/**
 * Created by siang on 2018/5/20.
 * 聊天窗口显示信息的 Adapter，对象为 Message
 */

public class BookListAdapter extends RecyclerView.Adapter<BookListAdapter.ViewHolder> {

    private List<BookInfo> bookList;       //view 的数据来源
    private Socket socket;
    private Context context;
    private String username;
    private String way = "2";
    private String search = "";

    // 创建 viewholder 子类，用于暂存 recyclerview 选项的 view，以便重复利用
    static class ViewHolder extends RecyclerView.ViewHolder {

        TextView bookName;
        TextView bookAuthor;
        TextView bookId;
        TextView bookHave;
        TextView bookBorrow;
        TextView btBorrow;

        // 构造 viewHolder,其中 view 表示父类的布局，用其获取子项元素
        ViewHolder(View view) {
            super(view);
            bookName = (TextView) view.findViewById(R.id.book_name);        //R 为指向 res 下 layout 的 java
            bookAuthor = (TextView) view.findViewById(R.id.book_author);
            bookId = (TextView) view.findViewById(R.id.book_id);
            bookHave = (TextView) view.findViewById(R.id.book_have);
            bookBorrow = (TextView) view.findViewById(R.id.book_borrow);
            btBorrow = (TextView) view.findViewById(R.id.bt_borrow);
        }
    }
    // 构造 MessageAdapter，传入列表
    public BookListAdapter(List<BookInfo> listItem, Context ct) {
        bookList = listItem;
        context = ct;
    }

    public void setSocket(Socket soc) {
        socket = soc;
    }

    public void setUsername(String user) {
        username = user;
    }

    public void setWay(String w) {
        way = w;
    }

    public void setSearch(String s) {
        search = s;
    }

    /**
     * 创建 ViewHolder 通过 LayoutInflater 加载 RecycleView 子项的布局
     * 其中 parent 为 RecyclerView,即需要把 view 放置处的父元素
     * viewtype 为 View 的类型，可以根据这个类型判断去创建不同 item 的 ViewHolder
     * 构建一个 inflate，inflate 的作用是加载 XML 文件到 view 中来操作，view 并不会显示
     * 按照第一个参数的样式填充 view,第二个参数为父对象,第三个参数为是否挂到父对象上,即 add,因为 adapter
会自动调用挂上去，所以这里不用挂
     *
     * @param parent
     * @param viewType
     * @return
     */
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.book_list_item, parent,
false);
        return new ViewHolder(view);
    }

    /**
```

```
     * 当要显示 recyclerview 时会调用此方法
     * position 为要显示的数据的位置，根据 position 提供 message 对象给 viewholder
     *
     * @param holder
     * @param position
     */
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        final BookInfo book = bookList.get(position);
        holder.bookName.setText(book.getName());
        holder.bookAuthor.setText(book.getAuthor());
        holder.bookId.setText(book.getId());
        holder.bookHave.setText(book.getBookHave());
        holder.bookBorrow.setText(book.getBookBorrow());
        if (book.getType() == 0)
            holder.btBorrow.setText(context.getString(R.string.btBorrow));
        else if(book.getType() == 1)
            holder.btBorrow.setText(context.getString(R.string.btReturn));
        else
            holder.btBorrow.setVisibility(View.INVISIBLE);
        if (book.getType() != 2) {
            holder.btBorrow.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (book.getType() == 0) {
                        new Thread(new Runnable() {
                            @Override
                            public void run() {
                                try {
                                    //向服务器端用输出流输出消息
                                    OutputStream outputStream = socket.getOutputStream();
                                    outputStream.write(("Borrow_S" + "//" + socket.getLocalPort() + "//"
+ "0" + "//" + username + "//" + book.getName() + "//" + "0").getBytes("utf-8"));
                                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强
制发送出去
                                    outputStream.flush();
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                            }
                        }).start();
                    }
                    else {
                        new Thread(new Runnable() {
                            @Override
                            public void run() {
                                try {
                                    //向服务器端用输出流输出消息
                                    OutputStream outputStream = socket.getOutputStream();
                                    outputStream.write(("Return_S" + "//" + socket.getLocalPort() + "//"
+ "0" + "//" + username + "//" + book.getName() + "//" + "0").getBytes("utf-8"));
                                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强
制发送出去
                                    outputStream.flush();
                                } catch (IOException e) {
                                    e.printStackTrace();
                                }
                            }
                        }).start();
                    }
                }
            });
        }
        if(position == getItemCount()-1 && !way.equals("0")){//已经到达列表的底部
            if (way.equals("1"))
                loadMoreData(search, book.getName());
            else
                loadMoreData(book.getName(), book.getAuthor());

        }
    }

    public void loadMoreData(final String name, final String author) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Query_B_next" + "//" + socket.getLocalPort() + "//" + way + "//"
+ name + "//" + username + "//" + author).getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
```

```java
    }

    //提供 reclclerview 选项总数
    @Override
    public int getItemCount() {
        return bookList != null ? bookList.size() : 0;
    }
}
```

ManagerBookListAdapter.java

```java
package com.siang.pc.librarysystem.adapter;

import android.content.Context;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.entity.BookInfo;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.util.List;

/**
 * Created by siang on 2018/5/20.
 * 聊天窗口显示信息的 Adapter，对象为 Message
 */

public class ManagerBookListAdapter extends RecyclerView.Adapter<ManagerBookListAdapter.ViewHolder> {

    private List<BookInfo> bookList;       //view 的数据来源
    private Socket socket;
    private Context context;
    private String username;
    private String way = "2";
private String search = "";

// 创建 viewholder 子类，用于暂存 recyclerview 选项的 view，以便重复利用
static class ViewHolder extends RecyclerView.ViewHolder {

    TextView bookName;
    TextView bookAuthor;
    TextView bookId;
    TextView bookHave;
    TextView bookBorrow;
    TextView btAdd;
    TextView btDec;
    EditText etAdd;
    EditText etDec;
    TextView btDelete;

    // 构造 viewHolder,其中 view 表示父类的布局，用其获取子项元素
    ViewHolder(View view) {
        super(view);
        bookName = (TextView) view.findViewById(R.id.book_name);        //R 为指向 res 下 layout 的 java
        bookAuthor = (TextView) view.findViewById(R.id.book_author);
        bookId = (TextView) view.findViewById(R.id.book_id);
        bookHave = (TextView) view.findViewById(R.id.book_have);
        bookBorrow = (TextView) view.findViewById(R.id.book_borrow);
        btAdd = (TextView) view.findViewById(R.id.bt_add);
        btDec = (TextView) view.findViewById(R.id.bt_dec);
        btDelete = (TextView) view.findViewById(R.id.bt_delete);
        etAdd = (EditText) view.findViewById(R.id.ed_add);
        etDec = (EditText) view.findViewById(R.id.ed_dec);
    }
}

// 构造 MessageAdapter，传入列表
public ManagerBookListAdapter(List<BookInfo> listItem, Context ct) {
    bookList = listItem;
    context = ct;
}

public void setSocket(Socket soc) {
    socket = soc;
}

public void setUsername(String user) {
    username = user;
    }

    public void setWay(String w) {
        way = w;
    }

    public void setSearch(String s) {
        search = s;
    }

    /**
     * 创建 ViewHolder 通过 LayoutInflater 加载 RecycleView 子项的布局
     * 其中 parent 为 RecyclerView,即需要把 view 放置处的父元素
     * viewtype 为 View 的类型，可以根据这个类型判断去创建不同 item 的 ViewHolder
     * 构建一个 inflate，inflate 的作用是加载 XML 文件到 view 中来操作，view 并不会显示
     * 按照第一个参数的样式填充 view,第二个参数为父对象,第三个参数为是否挂到父对象上,即 add,因为 adapter
会自动调用挂上去，所以这里不用挂
     *
     * @param parent
     * @param viewType
     * @return
     */
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.manager_book_list_item,
parent, false);
        return new ViewHolder(view);
    }

    /**
     * 当要显示 recyclerview 时会调用此方法
     * position 为要显示的数据的位置，根据 position 提供 message 对象给 viewholder
     *
     * @param holder
     * @param position
     */
    @Override
    public void onBindViewHolder(@NonNull final ViewHolder holder, final int position) {
         if (holder.etAdd.getTag() instanceof TextWatcher){
            holder.etAdd.removeTextChangedListener ((TextWatcher) holder.etAdd.getTag());
        }
        if (holder.etDec.getTag() instanceof TextWatcher){
            holder.etDec.removeTextChangedListener ((TextWatcher) holder.etDec.getTag());
```

```
}
holder.etAdd.setText("");
holder.etDec.setText("");
TextWatcher watcherAdd = new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {

    }

    @Override
    public void afterTextChanged(Editable s) {
    }
};
TextWatcher watcherDec = new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {

    }

    @Override
    public void afterTextChanged(Editable s) {
    }
};
holder.etAdd.addTextChangedListener(watcherAdd);
holder.etAdd.setTag(watcherAdd);
holder.etDec.addTextChangedListener(watcherDec);
holder.etDec.setTag(watcherDec);
final BookInfo book = bookList.get(position);
holder.bookName.setText(book.getName());
holder.bookAuthor.setText(book.getAuthor());
holder.bookId.setText(book.getId());
holder.bookHave.setText(book.getBookHave());
holder.bookBorrow.setText(book.getBookBorrow());
holder.btDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            //向服务器端用输出流输出消息
                            OutputStream outputStream = socket.getOutputStream();
                            outputStream.write(("Delete_B" + "//" + socket.getLocalPort() + "//" + "0" +
"//" + book.getName()).getBytes("utf-8"));
                            //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出
去
                            outputStream.flush();
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                }).start();
            }
        });
        holder.btAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (holder.etAdd.getText().toString().equals("")) {
                    String text = context.getResources().getString(R.string.Error_Empty);
                    Toast toast = Toast.makeText(context, text, Toast.LENGTH_SHORT);
                    toast.show();
                }
                else {
                    new Thread(new Runnable() {
                        @Override
                        public void run() {
                            try {
                                //向服务器端用输出流输出消息
                                OutputStream outputStream = socket.getOutputStream();
                                outputStream.write(("Insert_B" + "//" + socket.getLocalPort() + "//" + "0"
+ "//" + book.getName() + "//" + holder.etAdd.getText().toString() + "//" + "0").getBytes("utf-8"));
                                //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发
送出去
                                outputStream.flush();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                        }
                    }).start();
                }
            }
        });
        holder.btDec.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (holder.etDec.getText().toString().equals("")) {
                    String text = context.getResources().getString(R.string.Error_Empty);
                    Toast toast = Toast.makeText(context, text, Toast.LENGTH_SHORT);
                    toast.show();
                }
                else {
                    new Thread(new Runnable() {
                        @Override
                        public void run() {
                            try {
                                //向服务器端用输出流输出消息
                                OutputStream outputStream = socket.getOutputStream();
                                outputStream.write(("Insert_B" + "//" + socket.getLocalPort() + "//" + "1"
+ "//" + book.getName() + "//" + holder.etDec.getText().toString() + "//" + "0").getBytes("utf-8"));
                                //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发
送出去
                                outputStream.flush();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                        }
                    }).start();
                }
            }
        });
        if(position == getItemCount()-1 && !way.equals("0")){//已经到达列表的底部
            if (way.equals("1"))
                loadMoreData(search, book.getName());
            else
                loadMoreData(book.getName(), book.getAuthor());

        }
    }

    public void loadMoreData(final String name, final String author) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
```

```
                    //向服务器端用输出流输出消息
                    OutputStream outputStream = socket.getOutputStream();
                    outputStream.write(("Query_B_next" + "//" + socket.getLocalPort() + "//" + way + "//"
+ name + "//" + username + "//" + author).getBytes("utf-8"));
                    //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发送出去
                    outputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }

    //提供 reclclerview 选项总数
    @Override
    public int getItemCount() {
        return bookList != null ? bookList.size() : 0;
    }
}
```

MessageAdapter.java

```
package com.siang.pc.librarysystem.adapter;

import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.view.ViewGroup;
import android.view.View;
import java.util.List;

import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.entity.ChatMessage;

/**
 * Created by siang on 2018/5/20.
 * 聊天窗口显示信息的 Adapter，对象为 Message
 */

public class MessageAdapter extends RecyclerView.Adapter<MessageAdapter.ViewHolder> {

    private List<ChatMessage> chatMessageList;       //view 的数据来源，Message 中定义了消息内容和来源

    // 创建 viewholder 子类，用于暂存 recyclerview 选项的 view，以便重复利用
    static class ViewHolder extends RecyclerView.ViewHolder {

        LinearLayout leftLayout;
        LinearLayout rightLayout;
        TextView leftMsg;
        TextView rightMsg;
        TextView leftUsername;
        TextView rightUsername;

        // 构造 viewHolder,其中 view 表示父类的布局，用其获取子项元素
        public ViewHolder(View view) {
            super(view);
            leftLayout = (LinearLayout) view.findViewById(R.id.left_layout);        //R为指向 res 下 layout
的 java
            rightLayout = (LinearLayout) view.findViewById(R.id.right_layout);
            leftMsg = (TextView) view.findViewById(R.id.left_message);
            rightMsg = (TextView) view.findViewById(R.id.right_message);
            leftUsername = (TextView) view.findViewById(R.id.left_username);
            rightUsername = (TextView) view.findViewById(R.id.right_username);
        }
    }

    // 构造 MessageAdapter，传入消息列表
    public MessageAdapter(List<ChatMessage> listItem) {
        chatMessageList = listItem;
    }

    /**
     * 创建 ViewHolder 通过 LayoutInflater 加载 RecycleView 子项的布局
     * 其中 parent 为 RecyclerView,即需要把 view 放置处的父元素
     * viewtype 为 View 的类型，可以根据这个类型判断去创建不同 item 的 ViewHolder
     * 构建一个 inflate，inflate 的作用是加载 XML 文件到 view 中来操作，view 并不会显示
     * 按照第一个参数的样式填充 view,第二个参数为父对象,第三个参数为是否挂到父对象上,即 add,因为 adapter
会自动调用挂上去，所以这里不用挂
     *
     * @param parent
     * @param viewType
     * @return
     */
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.message_item, parent, false);
        return new ViewHolder(view);
    }
```

```java
    /**
     * 当要显示 recyclerview 时会调用此方法
     * position 为要显示的数据的位置，根据 position 提供 message 对象给 viewholder
     *
     * @param holder
     * @param position
     */
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        ChatMessage chatMessage = chatMessageList.get(position);
        if (chatMessage.getType() == ChatMessage.TYPE_RECEIVED) {
            // 如果是收到的消息，则显示左边的消息布局，将右边的消息布局隐藏
            holder.leftLayout.setVisibility(View.VISIBLE);
            holder.rightLayout.setVisibility(View.GONE);
            holder.leftMsg.setText(chatMessage.getContent());
            holder.leftUsername.setText(chatMessage.getUsername());
        } else if (chatMessage.getType() == ChatMessage.TYPE_SENT) {
            // 如果是发出的消息，则显示右边的消息布局，将左边的消息布局隐藏
            holder.rightLayout.setVisibility(View.VISIBLE);
            holder.leftLayout.setVisibility(View.GONE);
            holder.rightMsg.setText(chatMessage.getContent());
            holder.rightUsername.setText(chatMessage.getUsername());
        }
    }

    //提供 reclclerview 选项总数
    @Override
    public int getItemCount() {
        return chatMessageList != null ? chatMessageList.size() : 0;
    }
}
```

MyBookListAdapter.java

```java
package com.siang.pc.librarysystem.adapter;


import android.content.Context;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.siang.pc.librarysystem.R;
import com.siang.pc.librarysystem.entity.MyBookInfo;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.util.List;

/**
 * Created by siang on 2018/5/20.
 * 聊天窗口显示信息的 Adapter，对象为 Message
 */

public class MyBookListAdapter extends RecyclerView.Adapter<MyBookListAdapter.ViewHolder> {

    private List<MyBookInfo> bookList;       //view 的数据来源
    private Socket socket;
    private Context context;
    private int self = 0;                    // 0 为查询自己，1 为查询他人
    private String username;

    // 创建 viewholder 子类，用于暂存 recyclerview 选项的 view，以便重复利用
    static class ViewHolder extends RecyclerView.ViewHolder {

        TextView bookName;
        TextView bookAuthor;
        TextView bookId;
        TextView bookStart;
        TextView bookEnd;
        TextView btBorrow;

        // 构造 viewHolder,其中 view 表示父类的布局，用其获取子项元素
        ViewHolder(View view) {
            super(view);
            bookName = (TextView) view.findViewById(R.id.book_name);        //R 为指向 res 下 layout 的 java
            bookAuthor = (TextView) view.findViewById(R.id.book_author);
            bookId = (TextView) view.findViewById(R.id.book_id);
            bookStart = (TextView) view.findViewById(R.id.book_start);
            bookEnd = (TextView) view.findViewById(R.id.book_end);
            btBorrow = (TextView) view.findViewById(R.id.bt_borrow);
        }
    }

    // 构造 MessageAdapter，传入列表
    public MyBookListAdapter(List<MyBookInfo> listItem, Context ct) {
        bookList = listItem;
        context = ct;
    }

    public void setSocket(Socket soc) {
        socket = soc;
    }

    public void setSelf(int sf) {
        self = sf;
    }
    /**
     * 创建 ViewHolder 通过 LayoutInflater 加载 RecycleView 子项的布局
     * 其中 parent 为 RecyclerView,即需要把 view 放置处的父元素
     * viewtype 为 View 的类型，可以根据这个类型判断去创建不同 item 的 ViewHolder
     * 构建一个 inflate，inflate 的作用是加载 XML 文件到 view 中来操作，view 并不会显示
     * 按照第一个参数的样式填充 view,第二个参数为父对象,第三个参数为是否挂到父对象上,即 add,因为 adapter
会自动调用挂上去，所以这里不用挂
     *
     * @param parent
     * @param viewType
     * @return
     */
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.mybook_list_item, parent,
false);
        return new ViewHolder(view);
    }


    public void setUsername(String user) {
        username = user;
    }

    /**
     * 当要显示 recyclerview 时会调用此方法
     * position 为要显示的数据的位置，根据 position 提供 message 对象给 viewholder
     *
     * @param holder
     * @param position
     */
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        final MyBookInfo book = bookList.get(position);
        holder.bookName.setText(book.getName());
        holder.bookAuthor.setText(book.getAuthor());
        holder.bookId.setText(book.getId());
        holder.bookStart.setText(book.getStartTime());
        holder.bookEnd.setText(book.getEndTime());
        holder.btBorrow.setText(context.getString(R.string.btReturn));
        if (self == 0) {
            holder.btBorrow.setVisibility(View.VISIBLE);
            holder.btBorrow.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    new Thread(new Runnable() {
                        @Override
                        public void run() {
                            try {
                                //向服务器端用输出流输出消息
                                OutputStream outputStream = socket.getOutputStream();
                                outputStream.write(("Return_S" + "//" + socket.getLocalPort() + "//" + "0"
+ "//" + username + "//" + book.getName() + "//" + "0").getBytes("utf-8"));
                                //输出流的消息在客户端存在缓冲区等待缓冲区满，只有 flush 清除缓冲区强制发
送出去
                                outputStream.flush();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                        }
                    }).start();
                }
            });
        }
        else
            holder.btBorrow.setVisibility(View.INVISIBLE);
    }

    //提供 reclclerview 选项总数
    @Override
    public int getItemCount() {
        return bookList != null ? bookList.size() : 0;
    }
}
```

BookInfo.Java
```
package com.siang.pc.librarysystem.entity;

/**
 * Created by siang on 2018/5/20.
 * 消息类：表示发送或接受的消息
 */

public class BookInfo {

    private String Id;
    private String name;
    private String author;
    private String bookHave;
    private String bookBorrow;
    private int type;               //0 借阅 1 归还 2 不可操作

    public BookInfo(String Id, String name, String author, String bookHave, String bookBorrow, int type)
{     // 构造函数
        this.Id = Id;
        this.name = name;
        this.author = author;
        this.bookHave = bookHave;
        this.bookBorrow = bookBorrow;
        this.type = type;
    }

    public int getType() {
        return type;
    }

    public void setType(int type) {
        this.type = type;
    }

    public String getId() {
        return Id;
    }

    public void setId(String Id) {
        this.Id = Id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getBookHave() {
        return bookHave;
    }

    public void setBookHave(String bookHave) {
        this.bookHave = bookHave;
    }

    public String getBookBorrow() {
        return bookBorrow;
    }

    public void setBookBorrow(String bookBorrow) {
        this.bookBorrow = bookBorrow;
    }
}
```

MyBookInfo.java

```
package com.siang.pc.librarysystem.entity;

/**
 * Created by siang on 2018/5/20.
 * 消息类：表示发送或接受的消息
 */

public class MyBookInfo {
    private String Id;
    private String name;
    private String author;
    private String startTime;
    private String endTime;

    public MyBookInfo(String Id, String name, String author, String startTime, String endTime) {     //构
造函数
        this.Id = Id;
        this.name = name;
        this.author = author;
        this.startTime = startTime;
        this.endTime = endTime;
    }

    public String getId() {
        return Id;
    }

    public void setId(String Id) {
        this.Id = Id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getStartTime() {
        return startTime;
    }
    public void setStartTime(String startTime) {
        this.startTime = startTime;
    }

    public String getEndTime() {
        return endTime;
    }

    public void setEndTime(String endTime) {
        this.endTime = endTime;
    }
}
```

ChatMessage.java

```
package com.siang.pc.librarysystem.entity;

/**
 * Created by siang on 2018/5/20.
 * 消息类：表示发送或接受的消息
 */

public class ChatMessage {
    public static final int TYPE_RECEIVED = 0;  // 接收消息
    public static final int TYPE_SENT = 1;      // 发送消息

    private String username;                       // 消息用户
    private String content;                       // 消息内容
    private int type;                             // 消息来源

    public ChatMessage(String username, String content, int type) {     // 构造函数
        this.username = username;
        this.content = content;
        this.type = type;
    }

    public static int getTypeReceived() {
        return TYPE_RECEIVED;
    }

    public static int getTypeSent() {
        return TYPE_SENT;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public int getType() {
        return type;
    }

    public void setType(int type) {
        this.type = type;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

ChatRecordSQLiteOpenHelper.java

```
package com.siang.pc.librarysystem.helper;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.siang.pc.librarysystem.entity.ChatMessage;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by siang on 2018/5/31.
 * 这个类是用于操作保存聊天记录的 SQLite 的辅助器
 */

public class ChatRecordSQLiteOpenHelper extends SQLiteOpenHelper {
```

```
private static final String DB_NAME = "Chat";
private static final int DB_VERSION = 1;
private static final String TABLE_NAME = "MessageRecord";
private static final String COL_USERNAME = "username";
private static final String COL_CONTENT = "content";
private static final String COL_TYPE = "type";

private static final String TABLE_CREAT =
        "CREATE TABLE " + TABLE_NAME + " ( " +
                COL_USERNAME + " TEXT, " +
                COL_CONTENT + " TEXT, " +
                COL_TYPE + " INTEGER ); ";

public ChatRecordSQLiteOpenHelper(Context context) {
    super(context, DB_NAME, null, DB_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(TABLE_CREAT);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
}

public List<ChatMessage> getAllChatMessage() {
    SQLiteDatabase db = getReadableDatabase();
    String[] columns = {
            COL_USERNAME, COL_CONTENT, COL_TYPE
    };
    Cursor cursor = db.query(TABLE_NAME, columns, null, null, null, null, null);
    List<ChatMessage> chatMessageList = new ArrayList<>();
    while (cursor.moveToNext()) {
        String username = cursor.getString(0);
        String content = cursor.getString(1);
        int type = cursor.getInt(2);
        ChatMessage chatMessage = new ChatMessage(username, content, type);
        chatMessageList.add(chatMessage);
    }
    cursor.close();
    return  chatMessageList;
}
```

```
    public long insertChatMessage(ChatMessage chatMessage) {
        SQLiteDatabase db = getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COL_USERNAME, chatMessage.getUsername());
        values.put(COL_CONTENT, chatMessage.getContent());
        values.put(COL_TYPE, chatMessage.getType());
        return db.insert(TABLE_NAME, null, values);
    }

    public void deleteAllChatMessage() {
        SQLiteDatabase db = getWritableDatabase();
        db.execSQL("DELETE FROM " + TABLE_NAME);
    }
}
```

4.3 爬虫源代码

DouBanBook.py

```
import time
import requests
import re
import csv
import random
from lxml import etree

headers = {
    'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36'
}

fp = open('./DouBan.csv', 'a', newline='',encoding='utf-8')
writer = csv.writer(fp)

def get_book_info(url):
    html = requests.get(url, headers=headers)
    selector = etree.HTML(html.text)
    try:
        infos= selector.xpath('//li[@class="subject-item"]')
        for info in infos:
            name = info.xpath('div[2]/h2/a/@title')[0]
            author = info.xpath('div[2]/div[1]/text()')[0].split('/')[0].split('、')[0].strip()
            author = re.sub(u"\\(.*?\\)|\\（.*?）|\\{.*?}|\\[.*?]|\\【.*?】", "", author)
            author = re.sub(u".*? 著", "", author)
            author = author.strip()
            #print(name, author)
            writer.writerow((name, author, round(random.uniform(1, 51)), 0))
    except IndexError:
        pass

if __name__ == '__main__':
    urls = ['https://book.douban.com/tag/摄影?start={}&type=T'.format(number) for number in range(0, 1000,
20)]
    page = 1
    for url in urls:
        get_book_info(url)
        print('已完成：' + str(page) + "-" + str(page+19))
        page = page + 20
        time.sleep(0.2)

fp.close()
```

Trans.py

```
import csv
import random

fp = open('./DouBan.csv', 'r',encoding='gbk')
fp2 = open('./BookList.csv', 'w',newline='',encoding='UTF-8')
writer = csv.writer(fp2)
count = 0

for line in csv.reader(fp):
    if count == 0:
        count = 1
        continue
    print(line)
    writer.writerow((line[0].replace(",",", "), line[1].split(',')[0].replace("?","•"), line[2], 0))



fp.close()
fp2.close()
```

参考文献

Android 开发

《Android 5.X App 开发实战》 黄彬华 清华大学出版社 2016 年 04 日
《第一行代码 Android 第 2 版》郭霖 人民邮电出版社 2016 年 12 月

JAVA 学习

http://www.runoob.com/java/java-tutorial.html

Socket 通信

https://blog.csdn.net/chengliang0315/article/details/54214380

https://blog.csdn.net/yehui928186846/article/details/52572157

https://blog.csdn.net/u012975705/article/details/48752377

Trie 树

https://blog.csdn.net/qq_37337268/article/details/79843491

https://baike.baidu.com/item/%E5%AD%97%E5%85%B8%E6%A0%91/9825209?fr=aladdin

https://www.cnblogs.com/xujian2014/p/5614724.html

Splay 树

https://www.tuicool.com/articles/Er6RnqI

https://www.cnblogs.com/skywang12345/p/3604238.html

https://baike.baidu.com/item/%E4%BC%B8%E5%B1%95%E6%A0%91/7003945?fr=aladdin

聊天界面

https://blog.csdn.net/bskfnvjtlyzmv867/article/details/71308343
https://blog.csdn.net/qq_16131393/article/details/51308006
https://www.cnblogs.com/cxq1126/p/7193228.html

教师评语评分

评语：

评分：______________

评阅人：

年　　　月　　　日