20.图书系统项目（四）

本节当中主要完成业务逻辑框架的搭建，让整个项目的框架可以跑起来!

1. ProxyUser类完善业务逻辑框架

`ProxyUser` 类添加如下方法：

> - 每个业务需要判断该代理对象是否是有权限的
> - 调用对应用户当中的方法进行操作

```
1   public void addBook() {
2       //查看代理的对象是是不是管理员
3       checkRealUserWhetherAdminUser("普通用户没有权限上架图书");
4       ((AdminUser) realUser).addBook();
5   }
6
7   //更新书籍操作
8   public void updateBook() {
9   }
10
11  //移除图书
12  public void removeBook() {
13  }
14
15  //查看图书的借阅次数
16  public void borrowCount( ) {
17  }
18
19  //查看最受欢迎的前K本书
20  public void generateBook() {
21  }
22
23  //查看库存状态
24  public void checkInventoryStatus() {
25  }
26  //按照类别  统计图书
27  public void categorizeBooksByCategory() {
28  }
29
30  //按照作者 统计图书
31  public void categorizeBooksByAuthor() {
```

```
    }

    //移除上架超过1年的书籍
    public void checkAndRemoveOldBooks() {
    }

    //------------------------------普通相关方法------------------------------//
    //借阅图书
    public void borrowBook() {
    }

    //归还图书
    public void returnBook() {
    }


    //查看个人借阅情况
    public void viewBorrowHistory() {
    }
```

`handleOperation` 方法，根据选择进行对应操作

> 查找图书、显示图书、退出系统三种操作对于管理员和普通用户来说都是共用的，所以直接定义在Library当中

```
public void handleOperation(int choice) {
    if (realUser instanceof AdminUser) {
        // 管理员操作
        switch (choice) {
            case Constant.SEARCH_BOOK:
                library.searchBook();
                break;
            case Constant.DISPLAY_BOOK:
                library.displayBooks();
                break;
            case Constant.EXIT:
                library.exit();
                break;
            case Constant.ADD_BOOK:
                addBook();
                break;
            case Constant.UPDATE_BOOK:
                updateBook();
                break;
            case Constant.REMOVE_BOOK:
```

```
                removeBook();
                break;
            case Constant.BORROWED_BOOK_COUNT:
                borrowCount();
                break;
            case Constant.GENERATE_BOOK:
                generateBook();
                break;
            case Constant.CHECK_INVENTORY_STATUS:
                checkInventoryStatus();
                break;
            case Constant.CATEGORIZE_BOOK_BY_CATEGORY:
                categorizeBooksByCategory();
                break;
            case Constant.CATEGORIZE_BOOK_BY_AUTHOR:
                categorizeBooksByAuthor();
                break;
            case Constant.CHECK_AND_REMOVE_OLD_BOOK:
                checkAndRemoveOldBooks();
                break;
            default:
                System.out.println("无效的操作。");
        }
    }else if (realUser instanceof NormalUser) {
        // 普通用户操作
        switch (choice) {
            case Constant.SEARCH_BOOK:
                library.searchBook();
                break;
            case Constant.DISPLAY_BOOK:
                library.displayBooks();
                break;
            case Constant.EXIT:
                library.exit();
            case Constant.BORROWED_BOOK:
                borrowBook();
                break;
            case Constant.RETURN_BOOK:
                returnBook();
                break;
            case Constant.VIEW_BORROW_HISTORY_BOOK:
                viewBorrowHistory();
                break;
            default:
                System.out.println("无效的操作。");
        }
    }
```

```
68  }
```

- 这里并没有使用1 2 3 等常数作为case的判断条件，而是使用了常量来表示
- 1 2 3 这种也被称为 magic number（幻数），缺点在于如果只是使用数字，别的程序员可能并不能读懂甚至过一段时间，自己也可能看不懂。所以定义为常量可以增加代码的可读性。

继续完善 `Constant` 类：

```
1   public class Constant {
2
3       //内存中的书籍数组初识容量
4       public static final int CAPACITY = 5;
5
6       //存储所有图书的文件
7       public static final String ALL_BOOK_FILE_NAME = "allbook.txt";
8
9
10      //------------------管理员相关操作管理----------------
11      //查找图书
12      public static final int SEARCH_BOOK = 1;
13      //显示图书
14      public static final int DISPLAY_BOOK = 2;
15      //退出系统
16      public static final int EXIT = 3;
17      //上架图书
18      public static final int ADD_BOOK = 4;
19      //更新图书
20      public static final int UPDATE_BOOK = 5;
21      //删除图书
22      public static final int REMOVE_BOOK = 6;
23      //查看图书的借阅次数
24      public static final int BORROWED_BOOK_COUNT = 7;
25      //查看受欢迎的图书
26      public static final int GENERATE_BOOK = 8;
27      //查看库存状态
28      public static final int CHECK_INVENTORY_STATUS = 9;
29      //按照类别  统计图书
30      public static final int CATEGORIZE_BOOK_BY_CATEGORY = 10;
31      //按照作者 统计图书
32      public static final int CATEGORIZE_BOOK_BY_AUTHOR = 11;
33      //移除上架超过1年的书籍
34      public static final int CHECK_AND_REMOVE_OLD_BOOK = 12;
35
36      //------------------普通用户相关操作管理----------------
37
```

```
38      //借阅图书
39      public static final int BORROWED_BOOK = 4;
40
41      //归还图书
42      public static final int RETURN_BOOK = 5;
43
44      //查看个人借阅情况
45      public static final int VIEW_BORROW_HISTORY_BOOK = 6;
46
47  }
```

总结：

1. 完善了代理类中方法的定义和整合，根据 `handleOperation` 方法进行实际选择
2. ProxyUser类代理了 AdminUser类和NormalUser类中的方法

2. `AdminUser` 类完善业务逻辑框架

由上节课可知，该类当中需要做一些准备工作，比如添加图书的时候需要做数据准备工作！

```
1   public class AdminUser extends User{
2
3       private Scanner scanner = null;
4
5       public AdminUser(String name,int userID) {
6           super(name,userID,"管理员");
7           scanner = new Scanner(System.in);
8       }
9       @Override
10      public int display() {
11          System.out.println("管理员 " + name + " 的操作菜单:");
12          System.out.println("1. 查找图书");
13          System.out.println("2. 打印所有的图书");
14          System.out.println("3. 退出系统");
15          System.out.println("4. 上架图书");
16          System.out.println("5. 修改图书");
17          System.out.println("6. 下架图书");
18          System.out.println("7. 统计借阅次数");
19          System.out.println("8. 查看最后欢迎的前K本书");
20          System.out.println("9. 查看库存状态");
21          System.out.println("10. 按类别统计图书 ");
22          System.out.println("11. 按作者统计图书 ");
23          System.out.println("12. 检查超过一年未下架的图书");
24          System.out.println("请选择你的操作: ");
```

```
25            return scanner.nextInt();
26        }
27        //其他操作方法
28        //上架图书
29        public void addBook() {
30            scanner.nextLine();
31            System.out.println("请输入书名：");
32            String title = scanner.nextLine();  // 输入书名
33            System.out.println("请输入作者：");
34            String author = scanner.nextLine();  // 输入作者
35            System.out.println("请输入类别：");
36            String category = scanner.nextLine();  // 输入图书类别
37            System.out.println("请输入出版年份：");
38            int year = scanner.nextInt();  // 输入出版年份
39            scanner.nextLine();  // 吞掉换行符
40
41            LocalDate shelfDate = LocalDate.now();  // 当前时间作为上架时间
42            Book newBook = new Book(title, author, category, year, shelfDate);
    // 创建新书对象
43
44            //调用图书类 添加图书
45            library.addBook(newBook);
46        }
47
48        //图书修改 支持修改书名 作者 类别
49        public void updateBook() {
50
51        }
52
53        //删除书籍
54        public void removeBook() {
55
56        }
57
58        //统计每本书的借阅次数
59        public void borrowCount() {
60             //不需要准备其他工作，直接调用具体的业务实现
61            library.borrowCount();
62        }
63
64        //查询最受欢迎的前n本书 【借阅报告】
65        public void generateBook() {
66
67        }
68
69        //查看库存状态
70        public void checkInventoryStatus() {
```

```
71          }
72
73          //按照类别 统计图书
74          public void categorizeBooksByCategory() {
75
76          }
77
78          //按照作者统计图书
79          public void categorizeBooksByAuthor() {
80
81          }
82
83          //并移除上架超过一年的图书
84          public void checkAndRemoveOldBooks() {
85
86          }
87
88          public void exit() {
89              library.exit();
90          }
91
92      }
```

- 上述方法中有的需要做数据准备，比如添加图书等，但是有些不需要比如**统计每本书的借阅次数**等。我们就直接调用 `Libary` 中的方法即可
- 我们观察到对于 `Scanner` 对象来说，不仅仅在 `AdminUser` 当中会使用到，而且将来也可能在其他地方使用到，此时我们可以把 `Scanner` 这个类设计为单例模式。保证整个项目只有这一个实例，大家都可以使用。

2.1 单例模式设计 `Scanner` 类和 `Libary` 类

在utils下创建2个类： `ScannerSingleton` 和 `LibrarySingleton`

2.1.1 单例模式设计 `Scanner`

```
1   package utils;
2
3   import java.util.Scanner;
4
5   public class ScannerSingleton {
6       private static Scanner scanner;
7
8       private ScannerSingleton() {
9
```

```
    }
    public static  Scanner getInstance() {
        if (scanner == null) {
            scanner = new Scanner(System.in);
        }
        return scanner;
    }

}
```

2.1.2 单例模式设计 `Libary`

```
package utils;

import book.Library;

public class LibrarySingleton {
    private static Library library;

    private LibrarySingleton() {

    }
    public static  Library getInstance() {
        if (library == null) {
            library = new Library();
        }
        return library;
    }
}
```

我们在使用到 `Scanner` 和 `Libary` 的地方，进行更改

```
public class AdminUser extends User{
    private Scanner scanner = null;
    private Library library = null;

    public AdminUser(String name,int userID) {
        super(name,userID,"管理员");
        library = LibrarySingleton.getInstance();
        scanner = ScannerSingleton.getInstance();
    }
}
```

```java
public class ProxyUser {
    //被代理的真实用户
    private User realUser;
    private Library library = null;

    public ProxyUser(User user) {
        //对被代理的对象进行赋值
        realUser = user;
        library = LibrarySingleton.getInstance();
    }
}
```

3. `NormalUser` 类完善业务逻辑框架

在设计该类的时候，我们需要想清楚一个逻辑。

- 用户借书怎么设计?
- 多个用户借书怎么设计?
- 借书之后存储到文件当中的格式是什么?

3.1 `PairOfUidAndBookId` 类设计

我们的设计是这样的:

- 文件当中存储借阅书籍信息为：用户ID,书籍ID的形式，每一行是一个借阅信息

> 1,101 :表示ID为1的用户借阅了ID为101的书籍
>
> 2,111
>
> 1,301

所以我们就涉及到了读取文件后，内存当中需要存储这些借阅信息。我们定义类 `PairOfUidAndBookId` 来表示一条借阅信息，多条信息使用数组即可。

> 在book包当中定义

```java
public class PairOfUidAndBookId {
    private int userId;

    private int bookId;

    public PairOfUidAndBookId() {
```

```
7
8       }
9
10      public PairOfUidAndBookId(int userId, int bookId) {
11          this.userId = userId;
12          this.bookId = bookId;
13      }
14
15      public int getUserId() {
16          return userId;
17      }
18
19      public void setUserId(int userId) {
20          this.userId = userId;
21      }
22
23      public int getBookId() {
24          return bookId;
25      }
26
27      public void setBookId(int bookId) {
28          this.bookId = bookId;
29      }
30      //把对象序列化为JSON字符串的形式
31      public String toJson() {
32          StringBuilder json = new StringBuilder();
33          json.append(userId).append(",");
34          json.append(bookId);
35          return json.toString();
36      }
37  }
```

3.2 `AnalyzingBorrowedBook` 类的设计

> util包中定义

该设计方案和 `AnalyzingBook` 设计是类似的

```
1   public class AnalyzingBorrowedBook {
2
3       public PairOfUidAndBookId[] loadObject(String filename) throws
    IOException {
4           //从文件读取数据
5           String content = FileUtils.readFile(filename);
6           if (content == null || content.isEmpty()) {
```

```
            System.out.println("已借阅书籍列表无数据，表示没有用户借阅过书籍");
            return null;
        }

        String[] JsonStrings = content.split("\n");

        PairOfUidAndBookId[] pairOfUidAndBookIds
                                    = new
PairOfUidAndBookId[JsonStrings.length];


        for (int i = 0; i < JsonStrings.length; i++) {
            PairOfUidAndBookId pairOfUidAndBookId = new PairOfUidAndBookId();
            String[] uidAndBookIds = JsonStrings[i].split(",");
            pairOfUidAndBookId.setUserId(Integer.parseInt(uidAndBookIds[0]));
            pairOfUidAndBookId.setBookId(Integer.parseInt(uidAndBookIds[1]));

            pairOfUidAndBookIds[i] = pairOfUidAndBookId;
        }

        return pairOfUidAndBookIds;
    }

    public void storeObject(PairOfUidAndBookId[] pairOfUidAndBookIds, String
filename) throws IOException {

        //先遍历pairOfUidAndBookIds数组当中不为空的数据多少个?
       int booksUseLen = 0;
        for (int i = 0; i < pairOfUidAndBookIds.length; i++) {
            if(pairOfUidAndBookIds[i] != null) {
                booksUseLen++;
            }
        }

        StringBuilder jsonArray = new StringBuilder();

        for (int i = 0; i < booksUseLen; i++) {
            if(pairOfUidAndBookIds[i] != null) {
                jsonArray.append(pairOfUidAndBookIds[i].toJson());
                if (i != booksUseLen-1) {
                    jsonArray.append("\n");
                }
            }
        }

        FileUtils.writeFile(jsonArray.toString(),filename);/* */
    }
```

```
52  }
```

3.3 NormalUser 类完善

```
1   public class NormalUser extends User {
2
3       // 用户已借阅的图书相关信息
4       private PairOfUidAndBookId[] pairOfUidAndBookIds;
5       // 当前书籍的借阅量
6       private int borrowedCount;
7       //最多借阅的图书数量
8       private static final int BORROW_BOOK_MAX_NUM = 5;
9       private final AnalyzingBorrowedBook analyzingBorrowedBook = new
    AnalyzingBorrowedBook();
10
11
12      private Scanner scanner = null;
13      private Library library = null;
14
15      //如果是普通用户，这里写死
16      public NormalUser(String name, int userID) {
17          super(name, userID, "普通用户");
18          loadBorrowedBook();
19          library = LibrarySingleton.getInstance();
20          scanner = ScannerSingleton.getInstance();
21      }
22
23      private void loadBorrowedBook() {
24          PairOfUidAndBookId[] allBorrowedBook;
25          try {
26              //1.先加载文件当中的借阅信息
27              allBorrowedBook=
    analyzingBorrowedBook.loadObject(Constant.BORROWED_BOOK_FILE_NAME);
28              //2. 默认已借阅的图书数组大小为BORROW_BOOK_MAX_NUM，这里也可以定义到常量
    类
29              pairOfUidAndBookIds = new PairOfUidAndBookId[BORROW_BOOK_MAX_NUM];
30              //3.没有读取到已借阅的图书信息
31              if (allBorrowedBook== null) {
32                  borrowedCount = 0;
33              } else {
34              //4. 查看实际读取到的数组长度是多少?
35                  int allBorrowedBookLen= allBorrowedBook.length;
36                  //5. 如果读取到了10本书被借阅 但是当前borrowedBooks数组长度小于10
37                  if (allBorrowedBookLen> pairOfUidAndBookIds.length) {
38                      //6. 按照实际情况进行分配数组内存
```

```
39                      pairOfUidAndBookIds = new
    PairOfUidAndBookId[allBorrowedBookLen];
40                  }
41                  //7.把数据拷贝回到 已借阅图书信息的数组当中
42                  for (int i = 0; i < allBorrowedBookLen; i++) {
43                      pairOfUidAndBookIds[i] = allBorrowedBook[i];
44                  }
45                  //8.更新当前实际借阅书籍的书籍数量
46                  borrowedCount = allBorrowedBookLen;
47              }
48          } catch (IOException e) {
49              throw new RuntimeException(e);
50          }
51
52      }
53
54      private void storeBorrowedBook() {
55          try {
56              analyzingBorrowedBook.storeObject(pairOfUidAndBookIds,
    Constant.BORROWED_BOOK_FILE_NAME);
57          } catch (IOException e) {
58              throw new RuntimeException(e);
59          }
60      }
61
62      @Override
63      public int display() {
64          System.out.println("普通用户 " + name + " 的操作菜单:");
65          System.out.println("1. 查找图书");
66          System.out.println("2. 打印所有的图书");
67          System.out.println("3. 退出系统");
68          System.out.println("4. 借阅图书");
69          System.out.println("5. 归还图书");
70          System.out.println("6. 查看当前个人借阅情况");
71          System.out.println("请选择你的操作: ");
72          return scanner.nextInt();
73      }
74
75      //借阅图书
76      public void borrowBook() {
77          scanner.nextLine();
78          System.out.println("请输入你要借阅图书的id: ");
79          int bookId = scanner.nextInt();
80          scanner.nextLine();
81
82          //如果书架没有书 不能借阅
83          if (library.getBookCount() == 0) {
```

```
84              System.out.println("书架没有书籍暂且不能借阅");
85              return;
86          }
87
88          loadBorrowedBook();
89
90          //判断要借阅的书 是否是已经被借阅过了
91          Book book = library.searchById(bookId);
92          if(book == null) {
93              System.out.println("没有该id的相关书籍: "+bookId);
94              return;
95          }
96          for (int i = 0; i < borrowedCount; i++) {
97              PairOfUidAndBookId pairOfUidAndBookId = pairOfUidAndBookIds[i];
98              if (pairOfUidAndBookId.getBookId() == book.getBookId()) {
99                  if (getUserID() == pairOfUidAndBookId.getUserId()) {
100                     System.out.println("该书已经被你借阅过了，你的ID是: " +
    getUserID());
101                     return;
102                 } else {
103                     System.out.println("该书已经被其他人借阅过了，他的ID是: " +
    pairOfUidAndBookId.getUserId());
104                     return;
105                 }
106             }
107         }
108
109         library.borrowBook(bookId);
110
111         //封装对象写到 借阅表当中
112         PairOfUidAndBookId pairOfUidAndBookId = new
    PairOfUidAndBookId(userID, book.getBookId());
113         pairOfUidAndBookIds[borrowedCount] = pairOfUidAndBookId;
114         borrowedCount++;
115
116         //存储借阅图书
117         storeBorrowedBook();
118
119         System.out.println("借阅成功! ");
120     }
121
122
123     //归还图书
124     public void returnBook() {
125
126     }
127
```

```
128         // 查看个人借阅情况
129         public void viewBorrowBooks() {
130 
131         }
132 
133     }
```

完