**目录**

[1 绪论 2](#_Toc59657044)

[1.1课程设计的目的 2](#_Toc59657045)

[1.2课程设计的内容及要求 2](#_Toc59657046)

[2 系统分析 3](#_Toc59657047)

[2.1 技术可行性分析 3](#_Toc59657048)

[2.2用户需求分析 3](#_Toc59657049)

[3 系统设计 4](#_Toc59657050)

[3.1 系统功能模块 4](#_Toc59657051)

[3.2 系统架构 4](#_Toc59657052)

[3.1.2 实体层设计 5](#_Toc59657053)

[3.1.3数据应用层设计 5](#_Toc59657054)

[3.2.4业务逻辑层设计 5](#_Toc59657055)

[3.2.5UI层设计 6](#_Toc59657056)

[4 系统实现 7](#_Toc59657057)

[4.1 游客部分 7](#_Toc59657058)

[4.2账户部分 9](#_Toc59657059)

[4.3 用户部分 11](#_Toc59657060)

[4.4管理员部分 19](#_Toc59657061)

[5系统测试 24](#_Toc59657062)

[5.1测试目的 24](#_Toc59657063)

[5.2测试方法 24](#_Toc59657064)

[5.3测试内容 24](#_Toc59657065)

[5.4测试用例 24](#_Toc59657066)

[5.5 测试结果 34](#_Toc59657067)

[5.6 结果分析 35](#_Toc59657068)

[5.7 小结 35](#_Toc59657069)

[6设计总结和心得体会 36](#_Toc59657070)

[参考文献 37](#_Toc59657071)

# 1 绪论

“WeReadMall”线上图书商城有很多，首先线上图书商城中经营图书的数量之多、品种之全，是令许多传统书店望尘莫及的。另外，还为客户提供便捷的检索方式，线上图书商城提供了多种检索途径，用户可以按书名、作者、出版社等进行单独检索和组合检索，这些优点都是传统书店不能相比的。

其次，解决了图书采购的地域限制，采购人员可以进入线上网络中的的任何一家书店，也就是说无论是国内的线上商城还是国外的线上商城，只要输入它的网址，顾客就可以进入到，然后根据自己的需要采购图书。

然后线上图书商城可以使网络优势发挥到极致，多数的线上图书商城都为用户提供了各色各样的服务，如新书推荐、畅销书排行、读者评论、作者访谈、专家点评等，这样一来，用户不仅能全方位地把握图书信息，又可以进行在线讨论，交流心得。还有就是，线上图书商城能够提供跟踪服务，如果用户在某一线上图书商城有过一次购买行为后，它就会自动根据记录，然后在这位用户下一次再浏览该商城时，为其推荐相似的书单。

最后，在线上购买图书相对于线下来说会比较实惠，传统采购无论是从本地书店还是从异地书市采购，都必须派采购人员前往，大大的提高了商品的成本，而通过线上订购则是节约了不小的成本开支。

## 1.1课程设计的目的

本课程设计的目的就是要将我们所学习到的理论知识与实际应用相互结合，每个人充分应用所学的知识，通过课程设计的过程，不断的提高我们组织数据及编写大型程序的能力，并培养基本的、良好的程序设计技能以及小组成员之间的合作能力。

设计中要求综合运用所学知识，上机解决一些与实际应用结合紧密的、规模较大的问题，然后通过分析、设计、编码、调试等各环节的训练过程，让我们深刻理解数据结构的逻辑特性和物理表示以及数据结构的选择和应用、算法的设计及其实现等方面的知识，在加深对课程基本内容理解的同时，也对于程序设计方法以及上机操作等基本技能有一定的训练效果。

## 1.2课程设计的内容及要求

“WeReadMall“线上图书商城是由WeTeam小组全程设计并实现的线上图书商城系统。主要机制是线上浏览，线上支付。 在线上实现全部功能，系统分为三个入口：游客、顾客、管理员, 实现不同的功能，进行不同的管理。

通过JavaSE+MySQL技术在IDEA和SQLyog两个环境里进行实现预想的功能。 顺利实现并完成任务目标。

# **2 系统分析**

## 2.1 技术可行性分析

在近半年的学习中，本小组成员掌握了关于JavaSE，MySQL，JDBC等一系列的计算机软件课程，并加以灵活运用，能够在该设计课程上加以分析与讨论，适用于该课题的研究，同时也为进一步探索理论知识的奥秘。同时对此课题已潜心研究了很长时间，积累了大量有关材料，能够比较准确地，全方位地把握该课题的研究方向，使本课题的研究有了很好的理论基石[1]。

## 2.2用户需求分析

随着时代的发展与进步，越来越多的人开始了网购生活，无论是衣食住行，还是日常生活需要，网络已经越来越融入到我们的生活中，那么对于喜好看书的朋友来说，拥有一本书是很幸福的事情。但是去书店采购的话，就会使这个过程变得十分麻烦，甚至有可能在书店会找不到自己需要的书籍，也有可能是自己想要的书籍在书店出售的价格过于高昂，那样的话，就在这样的过程中消磨对于图书的兴趣，这样一来，是非常得不偿失的，而WeReadMall线上图书商城，就是为了用户而生，绝对能够满足大部分用户的需求，永远给客户提供最好的体验与最多的选择。

每个人都有需求，对于一名书友来说，能够买到自己想要的书一定是这个世界上最为快乐的事情之一，那么，能够快人一步的买到自己想要的书一定就会得到更大的满足感，同样，让客户更加方便快速的获得自己想要的书籍也是每个线上图书商城想要做到的事情，WeReadMall线上图书商城为了客户，划分了多个模块，清晰明确的显示各项客户需要的功能，就算有的书友觉得注册麻烦，我们也为他提供游客登陆方式，先浏览后注册，等找到你想要的书后或者觉得确实很好用之后，再进行注册也是完全可以的。

互助互惠是每个人都应该明白的道理，那么，线上图书商城也是如此，在享受客户带来的利益的同时，也应该为客户提供更多的优惠与实际奖励，毕竟具有人情味才能够让更多的用户喜欢上，那么我们的WeReadMall线上图书商城也是如此，不仅仅为了老用户提供了不同的优惠力度，而且每位新用户也都可以享受到属于自己的优惠，绝对让客户买的放心，用的舒心。

# 3 系统设计

## 3.1 系统功能模块

WeReadMall线上书城系统从实际需求方向分析可以分成以下几大模块：

（1）游客模块：浏览商品；账户注册；账户登录；退出系统。

（2）客户模块：退出登录；浏览商品；账户注册；账户登录；图书查找；加购；查看购物车：结算；修改商品数量；删除商品；查看订单：签收订单；退出系统。

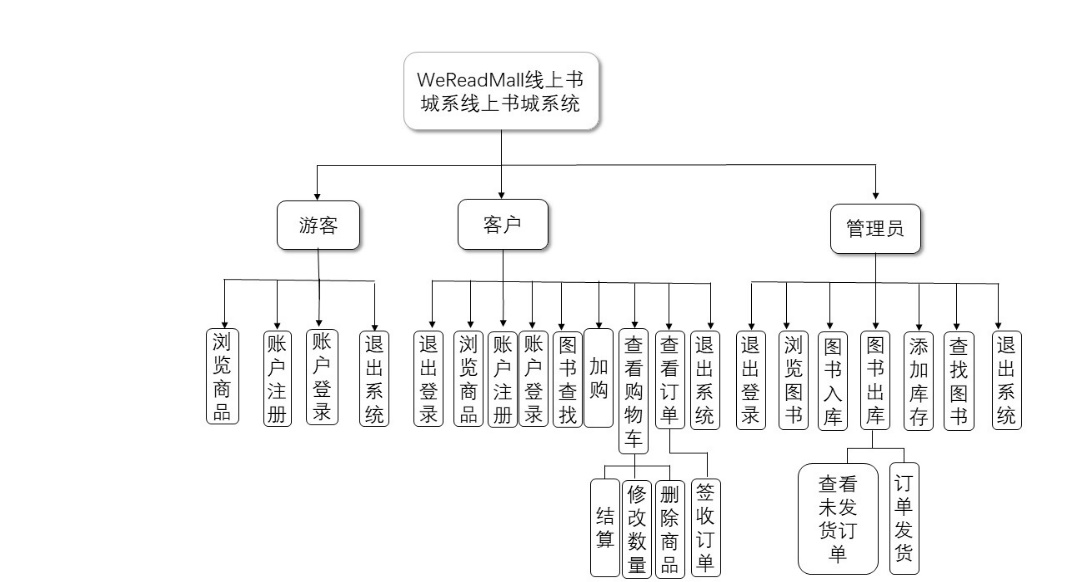
（3）管理员模块：退出登录；浏览图书；图书入库；图书出库：查看未发货订单；订单发货；添加库存；查找图书；退出系统。WeReadMall线上图书商城系统功能图如图3.1所示。

图3.1 系统功能图

## 3.2 系统架构

“WeReadMall”线上图书商城系统架构整体遵从分层设计，系统架构图如图3.2所示。系统通过实体层(modules04\_Entity)贯穿三层实现各种业务功能。

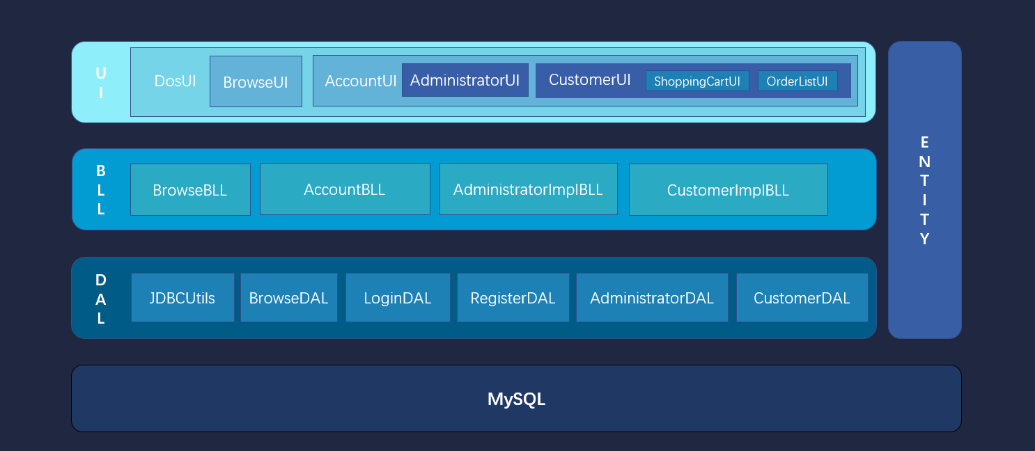
UI层(modules01\_UI)；业务逻辑层(modules02\_BLL)；数据应用层(modules03\_DAL)

图3.2系统架构图

### 3.1.2 实体层设计

(1)礼物工厂包(gifts\_factory)：礼物工厂包括帆布袋(Bookmark)，钢笔(Pen)，帆布袋(CanvasBag)三种，都继承了Gifts类；GiftsFactory类实现工厂功能：1是帆布袋(Bookmark)，2是钢笔(Pen)，3是帆布袋(CanvasBag)[2]。

(2)权限接口包(permissions\_entity)：管理员权限接口(Administrator)，顾客权限接口(Customer)；用户类(User)是数据库中Users表元素的Java类模型；角色类(Role)对两个接口和Users的id，name进行包装，进而实现各种功能。

(3)工具包(utils)：用户输入类(OptimizeScanner)：本类有效防止了用户的错误输入，避免了运行期异常的发生。

(4)图书类(Book)：Book类和Gifts类共同继承了顶级商品类Product。

(5)购物车类(ShoppingCart)：包含用户的userId(String)和商品列表productsLinkedList(LinkedList<Product>)，选型未链表的原因是购物车的常用操作是添加和删除，而链表增删快[3]。

(6)订单信息类(OrderList)：该类包含了用户的下单的所有信息。

### 3.1.3数据应用层设计

(1)数据层权限包(permissions\_dal)

分为两种类AdministratorDAL和CustomerDAL利用JDBC通过SQL语句实现对管理员和客户的业务数据的更新和保存。

(2)工具包(util)

JdbcUtils类实现对数据库连接对象驱动的注册和创建，连接资源的关闭和公共操作的实现。

(3)数据层浏览类(BrowseDAL)

本类通过MySQL的LIMIT"方言"，实现对商品浏览的分页。

(4)数据层登录和注册类(LoginDAL，RegisterDAL)

通过JDBC和SQL语句实现对用户登录，注册数据的检验，保存等功能[4]。

(5)JDBC配置文件(jdbc。properties)

url=jdbc:mysql:///we\_read\_mall

user=root

password=root

driver=com.Mysql.jdbc.Driver

### 3.2.4业务逻辑层设计

(1)业务逻辑层权限包(permissions\_bll)

管理员业务逻辑类(AdministratorImplBLL)和客户业务逻辑层(CustomerImplBLL)分别实现了实体类的Administrator和Customer接口，负责对UI层数据和数据应用层之间数据的传递[5]。

(2)业务逻辑层账户类(AccountBLL)

负责对UI层账户业务(登录，注册)和数据应用层之间的数据传递。

(3)业务逻辑层浏览类(BrowseBLL)

负责将数据应用层的商品数据传递给UI层。

### 3.2.5UI层设计

(1) 账户UI层(AccountUI)：展示功能选项“b.返回1.登录2.注册e.退出”,保存用户的输入,并把数据传递给业务逻辑层。

(2)管理员UI层(AdministratorUI)：展示功能选项“b.退出登录1.浏览2.入库3.出库4.添加5.查找e.退出”,让用户进行业务的选择,并把用户输入的数据传递给业务逻辑层。

(3)浏览UI层(BrowseUI)：展示功能选项“b.返回(1~n).查看页数e.退出”,通过用户的输入展示相应的商品。

(4)客户UI层(CustomerUI)：展示功能选项“b.退出登录1.浏览2.查找3.加购4.购物车5.查看订单e.退出”，让用户进行业务的选择,并把用户输入的数据传递给业务逻辑层。

(5)DosUI层(DosUI)：系统的唯一入口,展示功能选项"1.浏览2.购物车3.加购4.账户e.退出",只能浏览商品,若要进行更多操作只能通过4。账户选项登录系统

(6)订单UI层(OrderListUI)：负责对订单进行展示

(7)购物车UI层(ShoppingCartUI)：展示功能选项“b.返回主页1.结算2.修改数量3.删除商品e.退出”[6]。

# **4 系统实现**

系统实现是课程设计的最终目的，是课程设计结果的呈现。在设计过程中要根据之前的需求分析，完成系统要实现的各种需求及功能以及系统的呈现结构。本系统的实现如下

## 4.1 游客部分

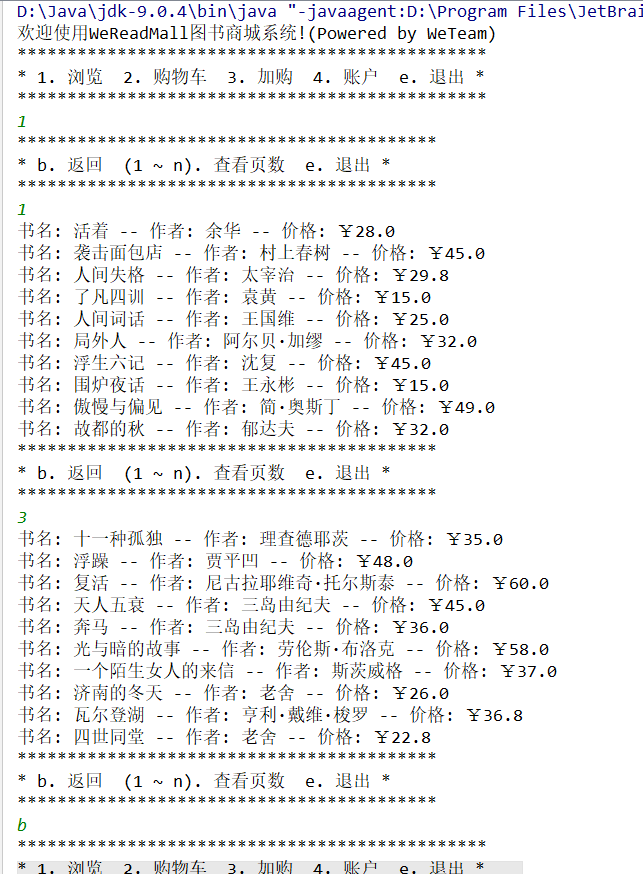
 游客部分，顾名思义就是用户的未登录状态，在不登录的情况下，游客是不支持太多的功能使用的，只能进行基础的页面浏览功能，可以对于商城中的图书目录进行较为浅显的了解。游客浏览界面如图4.1所示。

图4.1游客浏览界面

核心代码:

**public class** BrowseDAL {  
 **public static** LinkedHashMap<String, Book> getBookLinkedHashMap(**int** showNumber) {  
 String sql = **"SELECT** *\** **FROM books LIMIT ?, ?"**;  
 Connection connection = **null**;  
 PreparedStatement preparedStatement = **null**;  
 ResultSet resultSet = **null**;  
 LinkedHashMap<String, Book> bookLinkedHashMap = **new** LinkedHashMap<>();  
 **try** {  
 connection = JdbcUtils.*getConnection*();  
 preparedStatement = connection.prepareStatement(sql);  
 preparedStatement.setInt(1, showNumber);  
 preparedStatement.setInt(2, 10);  
 resultSet = preparedStatement.executeQuery();  
 **while** (resultSet.next()) {  
 Book book = **new** Book(  
 resultSet.getInt(**"id"**),  
 resultSet.getString(**"name"**),  
 resultSet.getString(**"author"**),  
 resultSet.getString(**"pubTime"**),  
 resultSet.getInt(**"inventory"**),  
 resultSet.getDouble(**"price"**));  
 bookLinkedHashMap.put(book.**name**, book);  
 }  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **finally** {  
 JdbcUtils.*close*(resultSet, preparedStatement, connection);  
 }

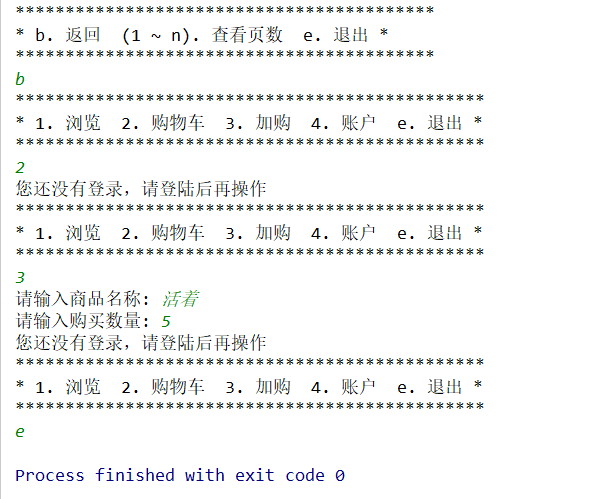
如果客户未进行登录又想要体验其他的功能时，就会在页面下方提示其未登录，会请求用户在登录后进一步使用，未登录提示图如图4.2 所示。

图4.2未登录提示图

核心代码:

**public boolean** isNotLogin() {  
 **boolean** boo = (**administrator** == **null** && **customer** == **null**);  
 **if** (boo) {  
 System.***out***.println(**"您还没有登录，请登陆后再操作"**);  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
}

## 4.2账户部分

(1) 登录

通过MySQL数据库users表中的isAdministrator列来判断是否是管理员登录：

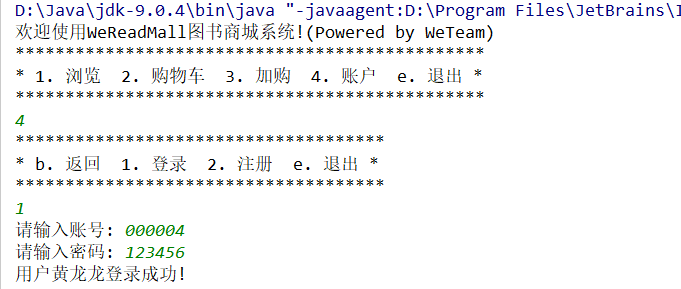
用户登录界面如图4.3所示。

图4.3用户登录界面

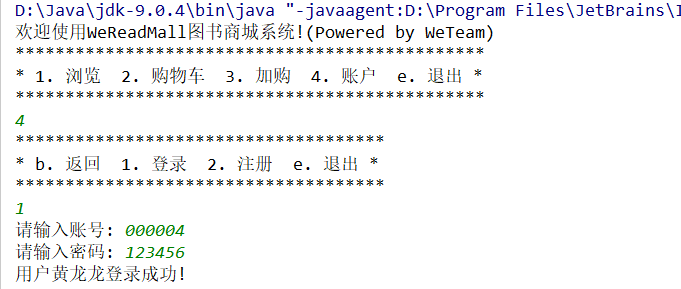
管理员登录界面如图4. 4所示。

图4.4管理员登录界面

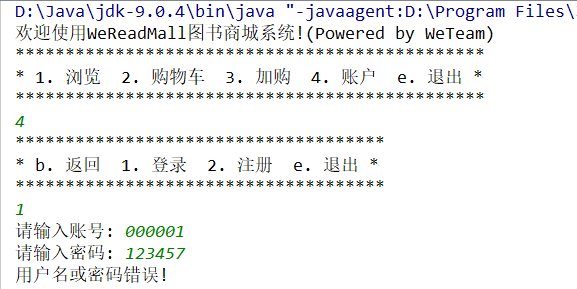
密码输入错误界面如图4.5所示。

图4.5密码输入错误界面

核心代码:

**public static** Role login() {  
 System.***out***.print(**"请输入账号: "**);  
 String username = OptimizeScanner.*getString*();  
 System.***out***.print(**"请输入密码: "**);  
 String password = OptimizeScanner.*getString*();  
 User user = AccountBLL.*getUser*(username, password);  
 **if** (user != **null**) {  
 **if** (user.getIsAdministrator() == 1) {  
 System.***out***.println(**"管理员"** + user.getUsername() + **"登录成功!"**);  
 **return new** Role(user.getUserId(), user.getUsername(), **new** AdministratorImplBLL());  
 } **else** {  
 System.***out***.println(**"用户"** + user.getUsername() + **"登录成功!"**);  
 **return new** Role(user.getUserId(), user.getUsername(), **new** CustomerImplBLL());  
 }  
 } **else** {  
 System.***out***.println(**"用户名或密码错误!"**);  
 **return null**;

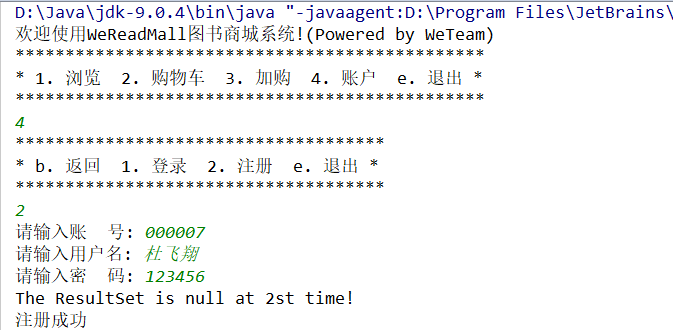
(2) 新用户注册如图4.6所示。

图4.6新用户注册图

核心代码:

**public static void** register() {  
 System.***out***.print(**"请输入账 号: "**);  
 String userId = OptimizeScanner.*getString*();  
 System.***out***.print(**"请输入用户名: "**);  
 String userName = OptimizeScanner.*getString*();  
 System.***out***.print(**"请输入密 码: "**);  
 String password = OptimizeScanner.*getString*();  
 System.***out***.println(AccountBLL.*isRegisterBLL*(**new** User(userId, userName, password, 0)));  
}

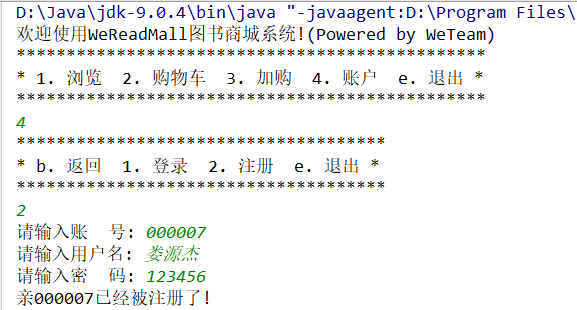
当用户注册已存在账号时, 系统将给出提示，如图4.7所示。

图4.7系统提示图

核心代码:

**public static** String isRegisterBLL(User user) {  
 **if** (RegisterDAL.*isExistDAL*(user.getUserId())) {  
 **return** (**"亲"** + user.getUserId() + **"已经被注册了!"**);  
 } **else** {  
 **if** (RegisterDAL.*isInsert*(user) > 0) {  
 **return "注册成功"**;  
 } **else** {  
 **return "注册失败"**;  
 }  
 }  
}

## 4.3 用户部分

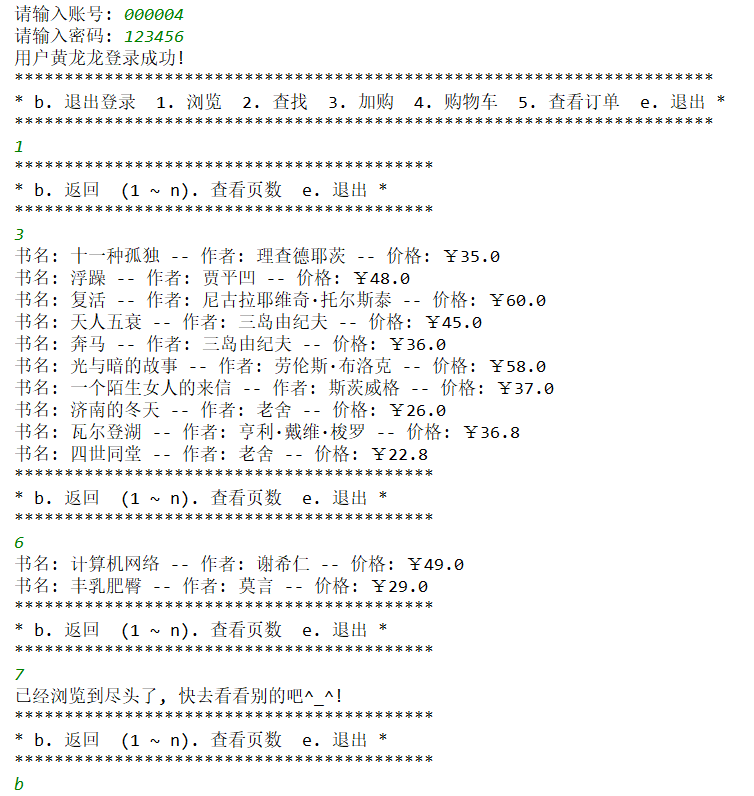
(1)客户登录后可以使用身份浏览，浏览界面每页只可展示10件商品，当所有的商品浏览结束之后，就会自动提示用户已经浏览到尽头，自动提示如图4.8所示。

图4.8自动提示图

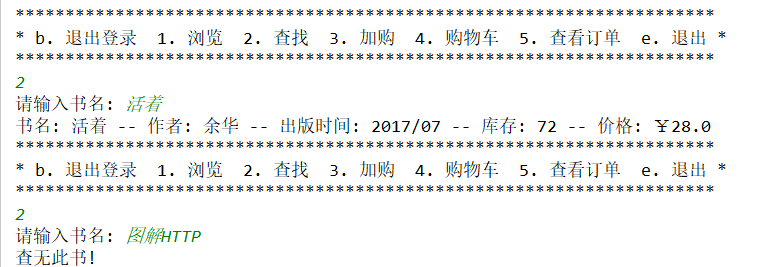
(2)每当用户输入书名进行搜索时，就会出现相关的图书信息，如果搜索的图书商城中没有的话就会提示用户查无此书，搜索图书如图4.9所示。

图4.9搜索图书

核心代码:

@Override  
**public** Book select(String name) {  
 **if** (CustomerDAL.*selectBookDAL*(name) != **null**) {  
 Book book = CustomerDAL.*selectBookDAL*(name);  
 **if** (book != **null**) {  
 System.***out***.println(  
 **"书名: "** + book.**name** + **" -- 作者: "** + book.getAuthor() + **" -- 出版时间: "** + book.getPubTime()  
 + **" -- 库存: "** + book.getInventory() + **" -- 价格: ￥"** + book.**price** );  
 }  
 **return** book;  
 } **else** {  
 System.***out***.println(**"查无此书!"**);  
 **return null**;

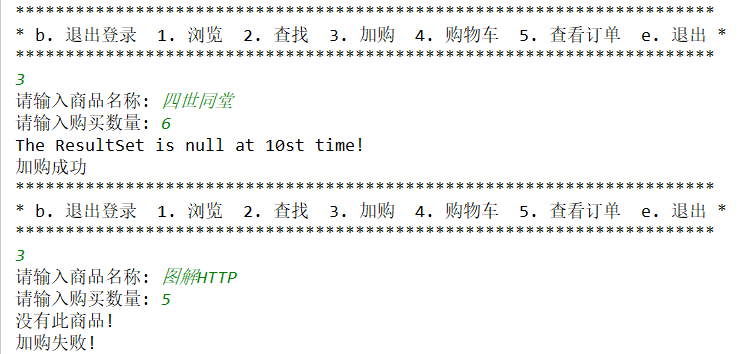
(3)选中喜欢的图书进行加购，若成功提示用户加购成功，若商品不存在则提示用户加购失败，如图4.10所示。

图4.10图书加购图

核心代码:

@Override  
**public boolean** insertShoppingCart(String userId, String name, **int** number) {  
 Product product = *getProductBLL*(name, number);  
 **return** product != **null** && CustomerDAL.*insertShoppingCartDAL*(userId, product);

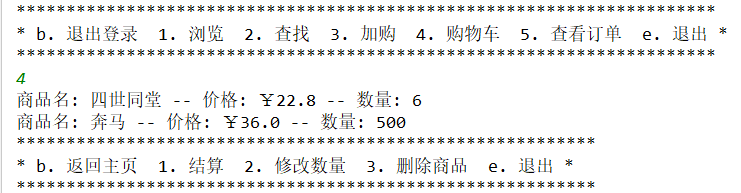
(4)用户可以选择查看购物车，然后在购物车中就可以查看所有用户自己已加购的图书，查看购物车详情如图4.11所示。

图4.11查看购物车

核心代码:

**private static void** show(ShoppingCart shoppingCart) {  
 **if** (shoppingCart != **null**) {  
 **for** (Product e : shoppingCart.getProductsLinkedList()  
 ) {  
 System.***out***.println(**"商品名: "** + e.**name** + **" -- 价格: ￥"** + e.**price** + **" -- 数量: "** + e.**buyNumber**);  
 }  
 }

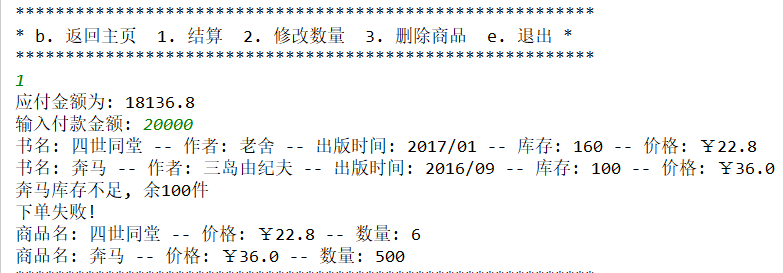
如果在用户下单时出现库存不足的情况，先提示库存不足，然后提示用户现有库存，最后提示用户下单失败，下单失败图如图4.12所示。

图4.12下单失败图

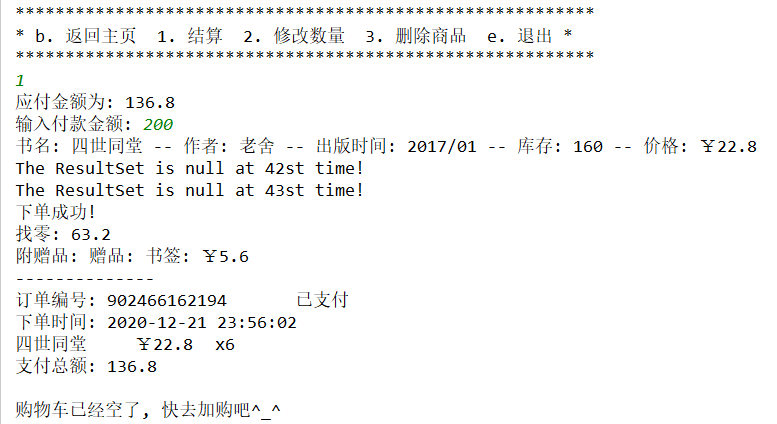
当图书库存充足时，就会接着进行付款步骤，提示用户输入付款金额，当用户支付相应的金额时就会为用户提供不同的赠品，并展示订单信息，告知顾客购物车已经空了，快去加购吧^\_^，付款成功图如图4.13所示。

图4.13付款成功图

核心代码:

**private static void** checkout(Role role) {  
 Product product = **null**;  
 ShoppingCart shoppingcart = (role.getShoppingCart(role.getId()));  
 **if** (!*shoppingCartIsEmpty*(role)) {  
 **double** price = shoppingcart.cost() \* 100;  
 System.***out***.println(**"应付金额为: "** + shoppingcart.cost());  
 **if** (price >= 150) {  
 product = GiftsFactory.*create*(3);  
 } **else if** (price >= 100) {  
 product = GiftsFactory.*create*(2);  
 } **else if** (price >= 50) {  
 product = GiftsFactory.*create*(1);  
 }  
 System.***out***.print(**"输入付款金额: "**);  
 **double** money = OptimizeScanner.*getDouble*() \* 100;  
 **while** (price > money) {  
 System.***out***.println(**"付款金额必须大于支付金额!"**);  
 System.***out***.print(**"输入付款金额: "**);  
 money = OptimizeScanner.*getDouble*() \* 100;  
 }  
 OrderList orderList = role.checkout(role.getId(), shoppingcart);  
 **if** (orderList != **null**) {  
 System.***out***.println(**"下单成功!"**);  
 System.***out***.println(**"找零: "** + (Math.*ceil*(money - price)) / 100.0);  
 **if** (product != **null**) {  
 product.cost();  
 }  
 OrderListUI.*show*(orderList);  
 } **else** {  
 System.***out***.println(**"下单失败!"**);  
 }  
 }  
}

@Override  
**public** OrderList checkout(String id, ShoppingCart shoppingCart) {  
 LinkedList<Product> productsLinkedList = shoppingCart.getProductsLinkedList();  
 **if** (isCanCheckout(productsLinkedList)){  
 *//生成订单号并添加订单列表* Date date = **new** Date();  
 String orderId = String.*valueOf*(date.getTime()).substring(5);  
 String number = **""**;  
 **for** (**int** i = 0; i < 4; i++) {  
 number += String.*valueOf*(**new** Random().nextInt(10));  
 }  
 orderId = number + orderId;  
 DateFormat dateFormat = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);  
 ArrayList<Product> productArrayList = **new** ArrayList<>();  
 productArrayList.addAll(productsLinkedList);  
 OrderList orderList = **new** OrderList(orderId, id, productArrayList, OrderList.***PAID***, dateFormat.format(date));  
 **boolean** boo = CustomerDAL.*insertOrderList*(orderList);  
 *//删除已结算过的购物车商品* **for** (Product e : productsLinkedList  
 ) {  
 CustomerDAL.*deleteProductDAL*(id, e.**id**);  
 }  
 **return** orderList;  
 }**else** {  
 **return null**;  
 }  
}

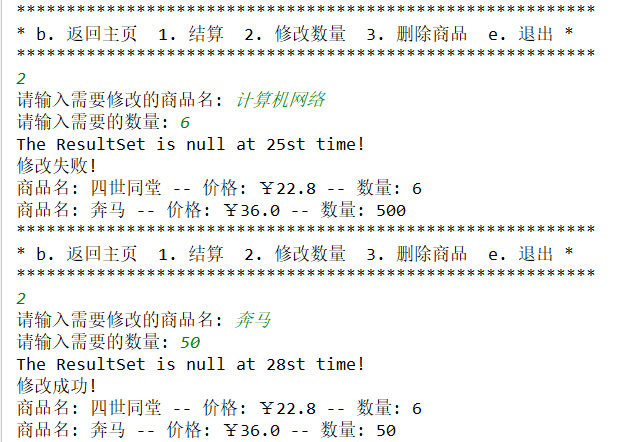
当用户在购物车中输入需要修改的商品名不存在时，就会提示用户修改失败，并展示用户购物车中的内容!当商品存在时，则会提示用户修改成功，并展示修改后的购物车，修改购物车内容如图4.14所示。

图4.14修改购物车内容

核心代码:

**private static void** updateNumber(ShoppingCart shoppingCart, Role role) {  
 **if** (!*shoppingCartIsEmpty*(role)) {  
 System.***out***.print(**"请输入需要修改的商品名: "**);  
 String name = OptimizeScanner.*getString*();  
 System.***out***.print(**"请输入需要的数量: "**);  
 **int** updateNumber = OptimizeScanner.*getInt*();  
 **if** (role.updateShoppingCart(role.getId(), name, updateNumber)) {  
 System.***out***.println(**"修改成功!"**);  
 } **else** {  
 System.***out***.println(**"修改失败!"**);  
 }  
 }  
}

当用户删除用户需要的商品时，会先展示商品信息，并提示用户删除成功!若商品不存在，展示商品信息，并提示用户删除失败!删除图书结果如图4.15所示。

图4.15删除图书

核心代码:

**private static void** delete(Role role) {  
 **if** (!*shoppingCartIsEmpty*(role)) {  
 System.***out***.print(**"请输入需要删除的商品名: "**);  
 String name = OptimizeScanner.*getString*();  
 Book book = role.select(name);  
 **if** (role.deleteProduct(role.getId(), book.**id**)) {  
 System.***out***.println(**"删除成功!"**);  
 } **else** {  
 System.***out***.println(**"删除失败!"**);  
 }  
 }  
}

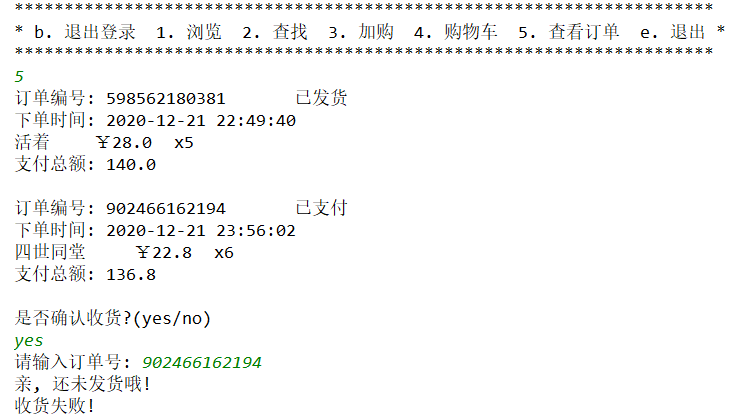
(5)展示历史所有订单信息，并询问顾客是否需要确认收货，若确认收货的订单状态为已完成或已支付提示用户亲，还未发货哦!收货失败!订单信息如图4.16所示。

图4.16订单信息

核心代码:

**private static void** showOrderList(Role role) {  
 ArrayList<OrderList> orderListArrayList = CustomerImplBLL.*selectOrderList*(role);  
 **if** (orderListArrayList.isEmpty()) {  
 System.***out***.println(**"暂无订单信息哦, 快去下单吧^\_^"**);  
 } **else** {  
 **for** (OrderList e : orderListArrayList  
 ) {  
 OrderListUI.*show*(e);  
 }  
 System.***out***.println(**"是否确认收货?(yes/no)"**);  
 String flag = OptimizeScanner.*getString*();  
 **if** (flag.equalsIgnoreCase(**"yes"**)) {  
 System.***out***.print(**"请输入订单号: "**);  
 String orderId = OptimizeScanner.*getString*();  
 **if** (role.signOrder(orderId)) {  
 System.***out***.println(**"收货成功!"**);  
 } **else** {  
 System.***out***.println(**"收货失败!"**);  
 }  
 }  
 }  
}

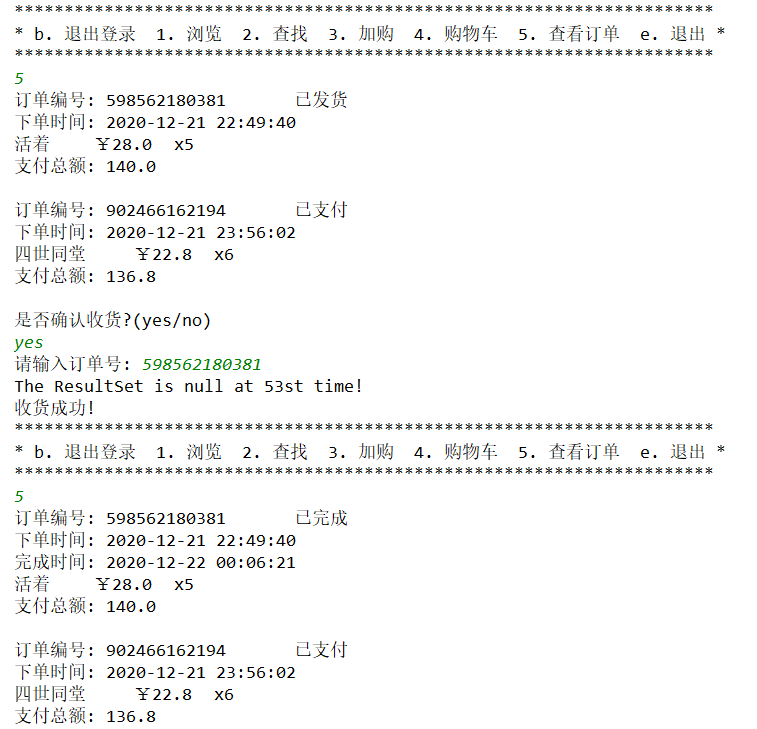
否则，提示用户收货成功!并将订单状态改为已完成。订单状态如图4.17所示。

图4.17订单状态图

核心代码:

@Override  
**public boolean** signOrder(String orderListId) {  
 **if** (!CustomerDAL.*getOrderState*(orderListId).equals(OrderList.***SHIPPED***)){  
 System.***out***.println(**"亲, 还未发货哦!"**);  
 **return false**;  
 }**else**{  
 Date date = **new** Date();  
 DateFormat dateFormat = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);  
 **return** CustomerDAL.*updateOrderList*(orderListId, dateFormat.format(date));  
 }  
}

## 4.4管理员部分

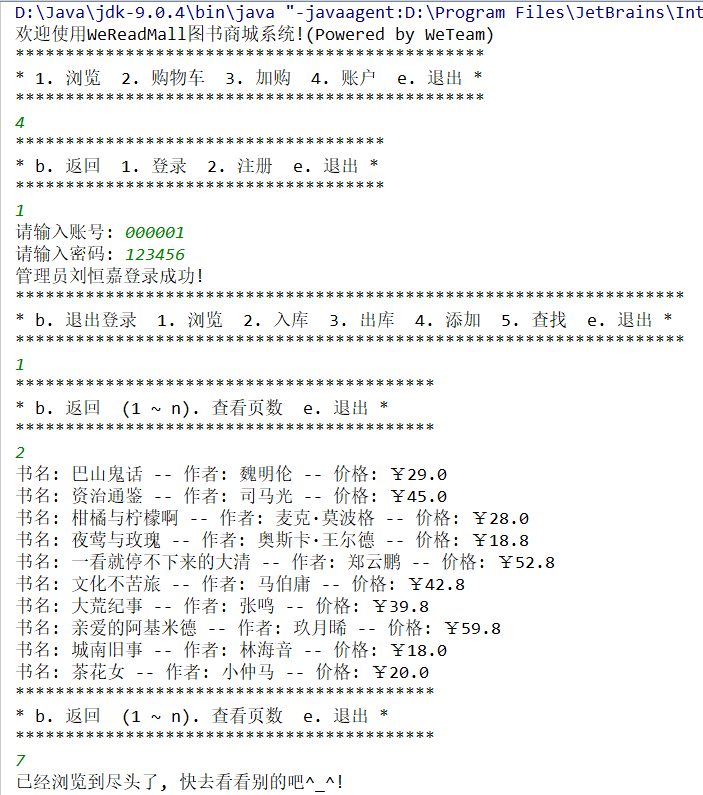
(1)浏览功能依旧采用MySQL的LIMIT分页技术，并实现了代码的复用。分页界面如图4.18所示。

图4.18分页界面图

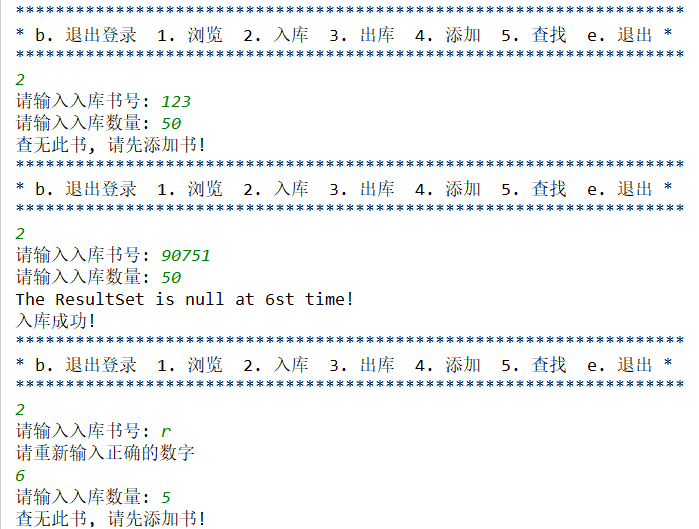
(2)本功能是让管理员输入书号和入库数量，若书号不存在则提示用户查无此书，请先添加书!若书号正确则提示管理员入库成功!若用户输入非法数据，系统自动识别并让用户重新输入。图书入库过程如图4.19所示。

图4.19图书入库过程图

核心代码:

@Override  
**public** String storage(**int** id, **int** number) {  
 **if** (AdministratorDAL.*isHasDAL*(id)) {  
 **if** (AdministratorDAL.*updateNumberUpDAL*(number, id)) {  
 **return "入库成功!"**;  
 } **else** {  
 **return "入库失败!"**;  
 }  
 } **else** {  
 **return "查无此书, 请先添加书!"**;  
 }  
}

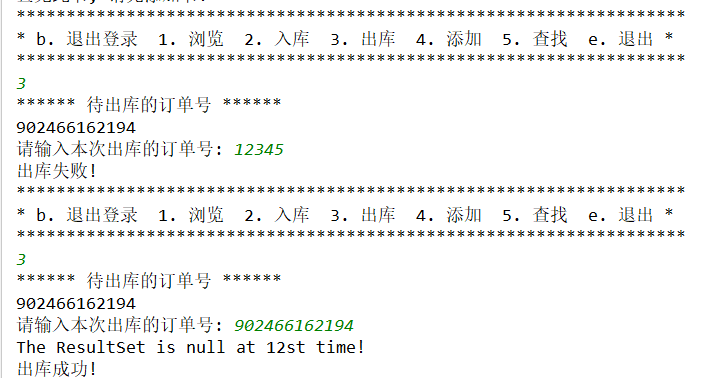
(3)系统自动识别未出库的订单并展示，若管理员输入错误单号自动提示出库失败!若输入正确则将每个商品的库存减去，并提示管理员出库成功!图书出库模拟如图4.20所示。

图4.20图书出库模拟图

核心代码:

**private static void** outOfLibrary(Role role) {  
 System.***out***.println(**"\*\*\*\*\*\* 待出库的订单号 \*\*\*\*\*\*"**);  
 **for** (String e : AdministratorImplBLL.*getOrderListIds*()  
 ) {  
 System.***out***.println(e);  
 }  
 System.***out***.print(**"请输入本次出库的订单号: "**);  
 String orderListId = OptimizeScanner.*getString*();  
 **if** (role.outOfLibrary(orderListId)) {  
 System.***out***.println(**"出库成功!"**);  
 } **else** {  
 System.***out***.println(**"出库失败!"**);  
 }  
}

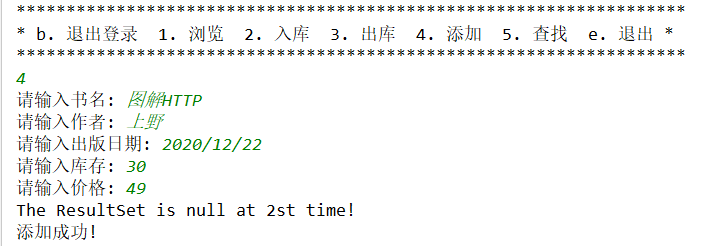
(4)该功能是添加商城没有的图书，由管理员进行信息录入，并告知管理员添加成功!添加新书如图4.21所示。

图4.21添加新书

核心代码:

**public static boolean** insertDAL(Book book) {  
 Connection connection = **null**;  
 PreparedStatement preparedStatement = **null**;  
 **int** count = 0;  
 **if**(!*isHaving*(book.**name**)){  
 String sql = **"INSERT INTO books(name, author, pubTime, inventory, price) VALUES (?, ?, ?, ?, ?)"**;  
 **try** {  
 connection = JdbcUtils.*getConnection*();  
 preparedStatement = connection.prepareStatement(sql);  
 preparedStatement.setString(1, book.**name**);  
 preparedStatement.setString(2, book.getAuthor());  
 preparedStatement.setString(3, book.getPubTime());  
 preparedStatement.setInt(4, book.getInventory());  
 preparedStatement.setDouble(5, book.**price**);  
 count = preparedStatement.executeUpdate();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **finally** {  
 JdbcUtils.*close*(preparedStatement, connection);  
 }  
 }**else**{  
 String sql = **"UPDATE books SET inventory = inventory + ? WHERE name = ?"**;  
 **try** {  
 connection = JdbcUtils.*getConnection*();  
 preparedStatement = connection.prepareStatement(sql);  
 preparedStatement.setInt(1, book.getInventory());  
 preparedStatement.setString(2, book.**name**);  
 count = preparedStatement.executeUpdate();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **finally** {  
 JdbcUtils.*close*(preparedStatement, connection);  
 }  
 }  
 **return** count > 0;  
}

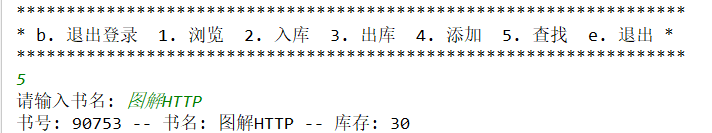
(5)该查找是管理员查找展示书号，书名，库存。查找书籍过程如图4.22所示。

图4.22查找书籍过程图

核心代码:

@Override  
**public** Book select(String name) {  
 **if** (AdministratorDAL.*selectBookDAL*(name) != **null**) {  
 Book book = AdministratorDAL.*selectBookDAL*(name);  
 **if** (book != **null**) {  
 System.***out***.println(  
 **"书号: "** + book.**id** + **" -- 书名: "** + book.**name** + **" -- 库存: "** + book.getInventory()  
 );  
 }  
 **return** book;  
 } **else** {  
 System.***out***.println(**"查无此书!"**);  
 **return null**;  
 }  
}

# **5系统测试**

测试是程序的一种执行过程，是软件生命周期中一项非常重要且非常复杂的工作，对软件可靠性保证具有极其重要的意义。在目前形式化方法和程序正确性证明技术还无望成为实用性方法的情况下，软件测试在将来相当一段时间内仍然是软件可靠性保证的有效方法。

## 5.1测试目的

测试的目的是尽可能发现并改正被测试软件中的错误，提高软件的可靠性，尽可能发现更多缺陷，提高开发效率，降低开发成本，提高用户体验[7]。

## 5.2测试方法

在用例设计环节中，测试人员为了尽可能实现高覆盖率的测试，测试用例设计涉及多种方法和技术，以保证更多的覆盖系统的每一行代码、每一个逻辑判断的运行，每一个可操作的功能分支都得以覆盖，相当于“穷举测试”。测试的最终目的，是尽可能在有限的时间内发现更多有效的，容易被用户认知的，复现几率较大的缺陷。

## 5.3测试内容

测试内容为检测系统的各项功能是否能在操作中正确的执行，通过输入各种信息，观测运行结果，与预期输出项相比较，查看各个功能是否能够正确的进行增删改查等操作，以及观测是否出现循环有误，数据是否正确等现象[8]。

## 5.4测试用例

以下是部分模块的测试用例，测试条件，与测试结果：

（1）游客测试用例如表5.1所示。

表5.1游客功能测试用例表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 设计人 | | 刘恒嘉 | | 测试人 | 姚舜宇 | 功能编号 | | 1 |
| 功能组 | | 登录模块 | | 功能点 | 登录账号 | 测试日期 | | 2020-12-22 |
| 测 试 前 提 | | | | | | | | |
| 测试条件 | 已添加顾客用户 用户：000004,密码: 123456，权限：用户 | | | | | | | |
| 测 试 项 目 及 内 容 | | | | | | | | |
| 测试步骤 | | 测试环境 | 用例描述 | | 输入项 | | | 预期输出项 |
| 1 | | 顾客功能选项主界面 | 未登录，使用购物车 | | 2 | | | 提示“您还没有登录，请登录后再操作”  显示出顾客功能选项主界面 |
| 2 | | 顾客功能选项主界面 | 未登录，使用加购功能 | | 边城/6 | | | 提示“您还没有登录，请登录后再操作”  显示出顾客功能选项主界面 |
| 3 | | 顾客账号，密码界面 | 浏览 | | 1 | | | 显示出浏览功能选项界面 |
| 测试结论 | | 游客功能可正常使用 | | | 总体结论 | | 功能正确，通过 | |

（2）顾客账号登录测试用例如表5.2所示。

表5.2顾客账号登录测试用例表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | | 测试人 | 姚舜宇 | 功能编号 | | 2 |
| 功能组 | 登录模块 | | 功能点 | 登录账号 | 测试日期 | | 2020-12-22 |
| 测 试 前 提 | | | | | | | |
| 测试条件 | 已添加顾客用户 用户：000004,密码: 123456，权限：用户 | | | | | | |
| 测 试 项 目 及 内 容 | | | | | | | |
| 测试步骤 | 测试环境 | 用例描述 | | 输入项 | | | 预期输出项 |
| 1 | 顾客账号，密码界面 | 账号正确，密码正确 | | 000004/123456 | | | 提示“登录成功”  进入顾客功能选项主界面 |
| 2 | 顾客账号，密码界面 | 账号，密码错误 | | 111111/12345678 | | | 提示“用户名或密码错误” |
| 3 | 顾客账号，密码界面 | 账号错误 | | 111111 | | | 提示“用户名或密码错误” |
| 4 | 顾客账号，密码界面 | 账号正确，密码错误 | | 111111/000000 | | | 提示“用户名或密码错误” |
| 测试结论 | 顾客账号登录功能可正常使用 | | | 总体结论 | | 功能正确，通过 | |

（3）游客注册测试用例如表5.3所示。

表5.3新用户注册测试用例表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 设计人 | | 刘恒嘉 | | 测试人 | 姚舜宇 | 功能编号 | | 3 |
| 功能组 | | 顾客模块 | | 功能点 | 用户注册 | 测试日期 | | 2020-12-22 |
| 测 试 前 提 | | | | | | | | |
| 测试条件 | 权限：用户 | | | | | | | |
| 测 试 项 目 及 内 容 | | | | | | | | |
| 测试步骤 | | 测试环境 | 用例描述 | | 输入项 | | | 预期输出项 |
| 1 | | 新用户注册界面 | 输入账号,输入用户名，输入密码 | | 000006/钱淼淼/123456 | | | 提示“注册成功” |
| 测试结论 | | 游客注册功能可正常使用 | | | 总体结论 | | 功能正确，通过 | |

（4）顾客浏览测试用例如表5.4所示。

表5.4顾客浏览测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | | | | 测试人 | 姚舜宇 | 功能编号 | 4 |
| 功能组 | 顾客模块 | | | | 功能点 | 浏览 | 测试日期 | 2020-12-22 |
| 测 试 前 提 | | | | | | | | |
| 测试条件 | | | 已添加顾客用户 用户：000006，密码: 123456，权限：用户 | | | | | |
| 测 试 项 目 及 内 容 | | | | | | | | |
| 测试步骤 | | 测试环境 | | 用例描述 | | 输入项 | | 预期输出项 |
| 1 | | 顾客主页面界面 | | 序号正确 | | 1 | | 显示出浏览功能选项界面 |
| 2 | | 顾客主页面界面 | | 序号错误 | | 6 | | 提示“输入正确菜单项”，显示出顾客功能选项主页面界面 |
| 3 | | 浏览选项界面 | | 输入要查看的页数（1-n） | | 1 | | 显示出第一页所有书籍信息（书名，作者，价格），显示浏览功能选项界面 |
| 4 | | 浏览选项界面 | | 输入未添加书籍的页数 | | 10 | | 提示“已经浏览到尽头，快去看看别的吧”，显示浏览功能选项界面 |
| 5 | | 浏览选项界面 | | 返回 | | b | | 显示出顾客功能选项主页面界面 |
| 测试结论 | | 顾客浏览书籍功能可正常使用 | | | | 总体结论 | | 功能正确，通过 |

（5）顾客查找测试用例如表5.5所示。

表5.5顾客查找测试用例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | | | 测试人 | 姚舜宇 | 功能编号 | 5 |
| 功能组 | 顾客模块 | | | 功能点 | 查找 | 测试日期 | 2020-12-22 |
| 测 试 前 提 | | | | | | | |
| 测试条件 | | 已添加顾客用户 用户：000006，密码: 123456，权限：用户 | | | | | |
| 测 试 项 目 及 内 容 | | | | | | | |
| 测试步骤 | | 测试环境 | 用例描述 | | 输入项 | | 预期输出项 |
| 1 | | 顾客主界面 | 序号正确 | | 2 | | 提示“请输入书名” |
| 2 | | 顾客主界面 | 序号不正确 | | 6 | | 提示“请输入正确菜单项”，显示顾客功能选项主页面 |
| 3 | | 查找书籍界面 | 书库中存在书籍 | | 活着 | | 显示书籍信息（书名，作者，出版时间，库存，价格），显示出顾客功能选项主界面 |
| 4 | | 查找书籍界面 | 书库中不存在书籍 | | 周易 | | 提示“查无此书”，显示出顾客功能选项主界面 |
| 测试结论 | | 顾客查找功能可正常使用 | | | 总体结论 | | 功能正确，通过 |

（6）顾客加购测试用例如表5.6所示。

表5.6顾客加购测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | | | | 测试人 | 姚舜宇 | 功能编号 | 6 |
| 功能组 | 顾客模块 | | | | 功能点 | 加购 | 测试日期 | 2020-12-22 |
| 测 试 前 提 | | | | | | | | |
| 测试条件 | | | 已添加顾客用户 用户：000006，密码: 123456，权限：用户 | | | | | |
| 测 试 项 目 及 内 容 | | | | | | | | |
| 测试步骤 | | 测试环境 | | 用例描述 | | 输入项 | | 预期输出项 |
| 1 | | 顾客主界面 | | 序号正确 | | 3 | | 提示“请输入商品名称，请输入购买数量” |
| 2 | | 顾客主界面 | | 序号不正确 | | 6 | | 提示“请输入正确菜单项”，显示顾客功能选项主页面 |
| 3 | | 加购界面 | | 加购书库存在书籍 | | 四世同堂/6 | | 提示“加购成功”, 显示顾客功能选项主页面 |
| 4 | | 加购界面 | | 加购书库不存在书籍 | | 周易/5 | | 提示“没有此商品，加购失败”，显示顾客功能选项主界面 |
| 5 | | 加购界面 | | 购买书籍数量大于库存量 | | 活着/200 | | 提示“加购成功” |
| 测试结论 | | 顾客加购功能可正常使用 | | | | 总体结论 | | 功能正确，通过 |

（7）顾客购物车测试用例如表5.7所示。

表5.7顾客购物车测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 设计人 |  | | 测试人 |  | 功能编号 | 7 |
| 功能组 | 顾客模块 | | 功能点 | 购物车 | 测试日期 | 2020-12-22 |
| 测 试 前 提 | | | | | | |
| 测试条件 | | 已添加顾客用户 用户：000006，密码: 123456，权限：用户  购物车内商品：四世同堂，活着 价格：22.8，28.0 数量：6，200 | | | | |
| 测 试 项 目 及 内 容 | | | | | | |
| 测试步骤 | 测试环境 | | 用例描述 | 输入项 | | 预期输出项 |
| 1 | 顾客主界面 | | 序号正确 | 4 | | 显示出购物车界面（商品名，价格，数量），购物车选择功能界面 |
| 2 | 顾客主界面 | | 序号不正确 | 6 | | 提示“请输入正确菜单项”，显示顾客功能选项主页面 |
| 3 | 结算 | | 输入付款金额大于商品总价值，购买商品数量未超过库存数 | 20000 | | 显示应付金额，商品信息（书名，作者，出版时间，库存，价格）提示“下单成功”，显示找零，附赠品，赠品，显示此次交易订单信息（订单编号，下单时间，下单详情，支付金额） |
| 4 | 结算 | | 输入付款金额小于商品总价值，购买商品数量未超过库存数 | 10 | | 提示“付款金额必须大于支付金额，输入付款金额” |
| 5 | 结算 | | 输入付款金额大于商品总价值，购买商品数量超过库存数 | 60000 | | 提示“活着库存不足，余72件，下单失败”，显示购物车添加商品信息 |
| 6 | 修改数量 | | 输入未加入购物车要修改的商品名和需要数量 | 计算机网络/6 | | 提示“修改失败”，显示出购物车中商品信息，显示购物车功能选择界面 |
| 7 | 修改数量 | | 输入已加入购物车中要修改的商品名和需要数量 | 活着/50 | | 提示“修改成功”，显示出购物车中更改后的全部商品信息（商品名，价格，数量），显示购物车功能选择界面 |
| 8 | 删除商品 | | 删除购物车中已存在的商品 | 活着 | | 显示活着信息（书名，作者，出版时间，库存，价格），提示“删除成功”，显示出购物车更新后商品信息（商品名，价格，数量），显示购物车功能选择界面 |
| 9 | 删除商品 | | 删除购物车中不存在的商品 | 计算机网络 | | 显示计算机网络信息（书名，作者，出版时间，库存，价格）提示“删除失败”，显示购物车中商品信息（商品名，价格，数量） |
| 10 | 返回主页 | | 返回主页 | b | | 显示顾客功能选项主页面 |
| 测试结论 | 用户购物车全部子功能可正常使用 | | | 总体结论 | | 功能正确，通过 |

（8）顾客查看订单测试用例如表5.8所示。

表5.8顾客查看订单商品测试用例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | | | 测试人 | 姚舜宇 | 功能编号 | 8 |
| 功能组 | 顾客模块 | | | 功能点 | 查看订单 | 测试日期 | 2020-12-22 |
| 测 试 前 提 | | | | | | | |
| 测试条件 | | | 已添加商家用户 用户：000006，密码: 123456，权限：用户  订单编号：880623850193下单时间：2020-12-22 15：57：30支付金额：416.8商品状态：已支付 | | | | |
| 测 试 项 目 及 内 容 | | | | | | | |
| 测试步骤 | | 测试环境 | | 用例描述 | 输入项 | | 预期输出项 |
| 1 | | 顾客主界面 | | 序号正确 | 5 | | 显示全部订单信息（订单编号，下单时间，商品信息，支付金额，商品状态），是否确认收货（yes/no） |
| 2 | | 顾客主界面 | | 序号不正确 | 6 | | 显示顾客功能选项主页面 |
| 3 | | 订单界面 | | 确认收货，订单编号错误，管理员未发货 | Yes/2344445645 | | 提示“亲，还未发货哦，收货失败”， 显示顾客功能选择界面 |
| 4 | | 订单界面 | | 确认收货，订单编号正确，管理员已发货 | Yes/880623850193 | | 提示“收货成功”显示顾客功能选择界面 |
| 5 | | 订单界面 | | 确认收货，订单编号错误 | Yes/2344445645 | | 提示“收货失败”， 显示顾客功能选择界面 |
| 6 | | 订单界面 | | 不收货 | No | | 显示顾客功能选择界面 |
| 测试结论 | | 顾客查看订单和子功能可正常使用 | | | 总体结论 | | 功能正确，通过 |

（9）管理员登陆测试如表5.9所示。

表5.9管理员登陆测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | 测试人 | 刘宝袁 | 功能测试 | 4-1 |
| 功能组 | 登录模块 | 功能点 | 登陆账号 | 测试日期 | 2020-12-22 |
| 测试前提 | | | | | |
| 测试条件 | 已添加管理员用户 用户名：000001 密码：123456 权限：管理员 | | | | |
| 测试项目及内容 | | | | | |
| 测试步骤 | 测试环境 | 用例描述 | 输入项 | 预期输出项 | |
| 1 | 请输入账户  请输入密码 | 账户密码正确 | 000001/  123456 | 管理员刘恒嘉登陆成功 | |
| 2 | 请输入账户  请输入密码 | 账户或密码错误 | 000001/  124563 | 用户名或密码错误 | |
| 3 | 请输入账户  请输入密码 | 账户或密码错误 | 154614/  123456 | 用户名或密码错误 | |
| 测试结论 | 管理员账户可以正常使用 | | 总体结论 | 功能正确，通过 | |

（10）管理员浏览用例如表5.10所示

表5.10管理员浏览用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | 测试人 | 刘宝袁 | 功能编号 | 4-1-1 |
| 功能组 | 管理员模块 | 功能点 | 浏览 | 测试日期 | 2020-12-22 |
| 测试前提 | | | | | |
| 测试条件 | 已添加管理员用户 用户名：000001 密码：123456 权限：管理员 | | | | |
| 测试项目及内容 | | | | | |
| 测试步骤 | 测试环境 | 用例描述 | 输入项 | 预期输出项 | |
| 1 | 浏览页码 | 书库第一页 | 1 | 输出书库第一页的各书籍书名作者价格共十条 | |
| 2 | 浏览页码 | 书库最后一页 | 6 | 输出书库最后一页的各书籍书名作则价格共三条 | |
| 3 | 浏览页码 | 书库不存在页 | 7 | 输出提示：  已经浏览到尽头了，快去看看别的吧^\_^! | |
| 4 | 浏览页码 | 退出浏览页 | b | 返回至上一级菜单 | |
| 测试结论 | 管理员账户能正常浏览书库 | | 总体结论 | 功能正确，通过 | |

（11）管理员入库书籍如表5.11所示

表5.11管理员入库书籍

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | 测试人 | 刘宝袁 | 功能编号 | 4-1-2 |
| 功能组 | 管理员模块 | 功能点 | 入库 | 测试日期 | 2020-12-22 |
| 测试条件 | 已添加管理员用户 用户名：000001 密码：123456 权限：管理员 | | | | |
| 测试项目及内容 | | | | | |
| 测试步骤 | 测试环境 | 用例描述 | 输入项 | 预期输出项 | |
| 1 | 请输入入库书号  请输入入库数量 | 入库已有记录的书籍 | 9751/  50 | 入库成功 | |
| 2 | 请输入入库书号  请输入入库数量 | 入库未有记录的书籍 | 123/  50 | 查无此书，请先添加书！ | |
| 3 | 请输入入库书号  请输入入库数量 | 非法输入 | r | 请重新输入正确的数字 | |
| 测试结论 | 管理员账户能正常入库书籍 | | 总体结论 | 功能正确，通过 | |

4管理员出库书籍

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | 测试人 | 刘宝袁 | 功能编号 | 4-1-3 |
| 功能组 | 管理员模块 | 功能点 | 出库 | 测试日期 | 2020-12-22 |
| 测试条件 | 已添加管理员用户 用户名：000001 密码：123456 权限：管理员 | | | | |
| 测试项目及内容 | | | | | |
| 测试步骤 | 测试环境 | 用例描述 | 输入项 | 预期输出项 | |
| 1 | 出库界面 | 出库待出库书籍 | 订单号一致序列号 | 出库成功 | |
| 2 | 出库界面 | 出库待出库书籍 | 订单号不一致序列号 | 出库失败 | |
| 3 | 出库界面 | 无待出库书籍 | 12345678 | 出库失败 | |
| 测试结论 | 管理员账户能正常出库书籍 | | 总体结论 | 功能正确，通过 | |

（12）管理员添加书籍如表5.12所示

表5.12管理员添加书籍

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | 测试人 | 刘宝袁 | 功能编号 | 4-1-4 |
| 功能组 | 管理员模块 | 功能点 | 添加 | 测试日期 | 2020-12-22 |
| 测试条件 | 已添加管理员用户 用户名：000001 密码：123456 权限：管理员 | | | | |
| 测试项目及内容 | | | | | |
| 测试步骤 | 测试环境 | 用例描述 | 输入项 | 预期输出项 | |
| 1 | 添加界面 | 正确添加新书籍 | 图解HTTP/  上野/  2020/12/22/  30/  49 | 添加成功 | |
| 2 | 添加界面 | 添加已有书籍 | 图解HTTP/  上野/  2020/12/22/  30/  29 | 添加成功 | |
| 3 | 添加界面 | 非法输入书名 | 微软体育的风格和健康的风格和健康的法国海军风格化风格化就 | 提示：输入字符串过长，请输入13个字符以内的字符串！ | |
| 4 | 添加界面 | 非法输入库存 | java基础/  陈国军/  2020/12/22/  axb | 提示：请输入正确的数字 | |
| 测试结论 | 管理员账户能正常添加书籍 | | 总体结论 | 功能正确，通过 | |

（13）管理员查找书籍如表5.13所示

表5.13管理员查找书籍

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 设计人 | 刘恒嘉 | 测试人 | 刘宝袁 | 功能编号 | 4-1-5 |
| 功能组 | 管理员模块 | 功能点 | 查找 | 测试日期 | 2020-12-22 |
| 测试条件 | 已添加管理员用户 用户名：000001 密码：123456 权限：管理员 | | | | |
| 测试项目及内容 | | | | | |
| 测试步骤 | 测试环境 | 用例描述 | 输入项 | 预期输出项 | |
| 1 | 查找界面 | 查找已有书籍 | 图解HTTP | 书号，书名，库存 | |
| 2 | 查找界面 | 查找未收录书籍 | hakgkhgda | 查无此书 | |
| 测试结论 | 管理员账户能正常查找书籍 | | 总体结论 | 功能正确，通过 | |

测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 功能点 | 质量保证标准 | 测试结果 |
| 管理员模块 | 登录 | 功能运行正常 | 正常 |
| 浏览 | 功能运行正常 | 正常 |
| 入库 | 功能运行正常 | 正常 |
| 出库 | 功能运行正常 | 正常 |
| 添加 | 功能运行正常 | 正常 |
| 查找 | 功能运行正常 | 正常 |

## 5.5 测试结果

本系统的所有模块测试结果如下表：

表5.9系统功能测试结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 功能点 | 质量保证标准 | 测试结果 |
| 顾客模块 | 新用户注册 | 功能运行正确 | 正常 |
| 登录 | 功能运行正确 | 正常 |
| 浏览 | 功能运行正确 | 正常 |
| 查找 | 功能运行正确 | 正常 |
| 加购 | 功能运行正确 | 正常 |
| 购物车 | 功能运行正确 | 正常 |
| 结算 | 功能运行正确 | 正常 |
| 修改数量 | 功能运行正确 | 正常 |
| 删除商品 | 功能运行正确 | 正常 |
| 查看订单 | 功能运行正确 | 正常 |
| 返回主页 | 功能运行正确 | 正常 |
| 退出登录 | 功能运行正确 | 正常 |
| 退出 | 功能运行正确 | 正常 |
| 返回 | 功能运行正确 | 正常 |

## 5.6 结果分析

功能测试过程属于软件开发过程中的重要一环，许多在开发中没有注意到的小问题、小细节都要靠在测试中来一步步进行发现。只有经过测试后功能完整且没有问题的软件才可以放心地交给用户进行使用。本系统经过以上部分测试系统功能基本齐全，实现了预想的功能，各个功能未发现重大错误，全部功能以检验、测试完毕，可以投入使用[9]。

## 5.7 小结

本章介绍了软件测试的意义及目的，分别介绍了顾客/管理员登录部分功能的测试，顾客购书，查看信息，查找功能的测试，管理员商品管理，书籍出库，书籍入库功能的测试，以及所有模块测试的结果，本章的测试过程中产生过一些小功能显示异常，执行信息与预期输出项不符合等问题，但经过最后的修改，所有功能基本上都可以正常使用[10]。

# 6设计总结和心得体会

此次的课程设计使我们更加扎实的掌握了有关java方面的知识，虽然在设计过程中会遇到了一些问题，但是通过不断的思考与改进以及全方面的检查,也是找到了问题的所在，并且解决了很多的问题，在这样的过程中，即使暴露出我们经验的不足与应用知识能力的匮乏，但是实践出真知，只有通过自己动手制作，才能够使我们掌握的知识不再是纸上谈兵，就算是会产生很多的错误，但是只要在以后能够不断改过，毕竟过而能改，善莫大焉。

当然在设计线上图书商城时，使用Java程序实现的过程中，我们可以更深刻的体会到在本学期对于java知识的认识和应用，从现在的角度来看，Java的前景还很大，从完成作业，本次课程设计让大家都学到了很多,不仅仅是小组的合理分工，还有从系统分析，到代码的编辑最后到测试，小组成员的配合尤为关键，尤其是在最后的检测调试环节，这个环节本身就是在践行“过而能改，善莫大焉”的知行观，就更加需要全组成员的共同努力。

这次课程设计终于顺利完成了，在设计中遇到了很多问题，最后在老师的指导下，终于也是游逆而解，通过此次的课程设计，我们懂得了理论与实际相结合是很重要的，学习理论知识才仅仅是千里之行的第一步，最重要的是，如何去将所学的知识应用到实践操作中去，只有把所学的理论知识与实践相结合起来，从理论中得出结论，才能真正为社会服务，从而提高自己的实际动手能力和独立思考的能力。

在设计的过程中遇到问题，可以说得是困难重重，但值得庆幸的是最后我们对于问题都得到了解决。此外，在实验过程中，对于团队精神也进行了进一步的考察，让我们在未来的合作中更加的默契，明白了团队的重要性以及团结就是力量这句话的真正含义，在以后的日子里也会越来越明白团队的重要性。

参考文献

1. 齐敏菊.Java程序设计“互联网+”教学模式与方法探索[J].计算机时代,2020(12):66-69.
2. 王养廷,郭慧,任宏德.Java程序设计能力培养的探索与实践[J].工业和信息化教育,2020(11):89-92.
3. 陈素清.Java程序设计课程实验教学研究[J].福建电脑,2020,36(11):176-177.
4. 金刚波,康鲜菜.基于“互联网+教学”的《Java程序设计》课程建设[J].数码世界,2020(11):186-187.
5. 金刚波,康鲜菜.基于“互联网+教学”的《Java程序设计》课程建设[J].数码世界,2020(11):186-187.
6. 孙洪迪,杨民峰,贾民政,方园.“Java程序设计”课程混合式教学设计[J].北京工业职业技术学院学报,2020,19(04):86-90.
7. 周燕琴,吕绪洋.软件工程专业面向对象程序设计课程教学改革[J].电脑知识与技术,2020,16(17):165-166.
8. 方阿丽.Java程序设计教学方法改革的研究与实践[J].福建电脑,2020,36(09):178-179.
9. 林跃,杨倩,盖志静,张志坤.论Java编程语言的特点[J].湖北农机化,2020(06):158.
10. MathiasEllmann.Fernlehrenund Fernlernen von Objektorientierter Programmierung (OOP). 2020, :1-7.