

南开大学

JAVA 语言与应用实验报告

中文题目: 控制台版 CD 出租销售店实验报告

外文题目: Experiment Report: Console-based CD Rental and Sales Store

学号: 2210556

姓名: 廖望

年级: 2023 级

专业: 计算机科学与技术

系别: 计算机科学与技术

学院: 计算机学院

指导教师: 刘嘉欣

完成日期: 2024 年 11 月

摘 要

本实验设计并实现了一个基于控制台的会员制 CD 出租销售管理系统。系统采用 MVC（Model-View-Controller）架构，包含会员管理、CD 租赁与销售、库存管理和统计等功能。通过 Maven 构建项目，简化了依赖管理，提升了代码的可扩展性与复用性。系统主要功能包括增加和删除会员、CD 的租赁与销售、库存更新及状态汇报。控制台提供清晰的指令，模块化设计和 Java 面向对象编程思想提高了系统的可读性与维护性。本实验展示了 MVC 模式在 Java 控制台应用中的应用，并通过 Maven 提升了开发效率。

关键词：控制台应用；MVC 架构；会员管理系统

Abstract

This experiment involves the design and implementation of a console-based CD rental and sales management system for members. The system uses the MVC(Model-View-Controller) architecture, with functionality for member management, CD rental and sales, inventory management, and reporting. Built with Maven, the project benefits from simplified dependency management, enhancing code scalability and reusability. Key features include adding and deleting members, CD rental and sales, inventory updates, and status reporting. The console provides clear commands, while modular design and Java' s object-oriented principles improve readability and maintainability. This experiment demonstrates MVC application in a Java console environment, with Maven streamlining the development process.

Key Words: Console Application; MVC Architecture; Membership Management System

目 录

摘要	I
Abstract	II
目录	III
第一章 实验目的	1
第二章 需求分析	2
第三章 功能设计	3
第四章 代码实现	4
第一节 User 类	4
第二节 Disk 类	6
第三节 Map 类	7
第四节 View 类	9
第五节 Control 类	11
第五章 Maven 项目构建	15
第六章 代码特点	16
第七章 实验结果	17
第八章 心得体会	18

第一章 实验目的

本实验的目的是设计并实现一个控制台版的 CD 出租和销售管理系统，采用 MVC（Model-View-Controller）架构进行开发。通过将数据、业务逻辑和用户界面分离，实现了代码的模块化、可扩展性和可维护性。系统提供的主要功能包括会员管理、CD 管理、租赁与购买、库存报告和帮助功能。实验通过使用 Maven 进行构建和依赖管理，进一步提高了开发效率。本项目旨在巩固 Java 面向对象编程、MVC 架构和 Maven 构建管理的知识和应用。

第二章 需求分析

为了满足 CD 出租销售店的基本业务需求，本系统的功能需求可以划分为以下几部分：

1. **会员管理：**系统应支持添加和删除会员，记录会员的基本信息（姓名、ID、余额和已借出的 CD），以便进行身份识别和账户管理。
2. **CD 租赁和销售：**支持会员对 CD 的租赁和购买操作，系统需对每张 CD 的名称、价格和库存量进行管理，确保库存数据的准确性。
3. **库存管理：**系统支持新增和更新 CD 库存的功能，以保证有足够的库存满足租赁和销售需求。
4. **数据报告：**提供库存和会员报告功能，输出当前所有 CD 库存和会员信息，帮助管理者了解系统的使用情况。
5. **帮助信息：**提供用户操作说明，便于用户通过控制台进行有效操作。

第三章 功能设计

系统采用 MVC 架构，分为 Model、View 和 Controller 三个主要模块。每个模块各司其职，具体设计如下：

1. **Model 层**：数据模型层负责存储系统中的主要业务数据，包括会员信息和 CD 信息。Model 层包含以下核心类：
 - **User** 类：用于存储会员的基本信息，如姓名、ID、租赁状态和余额。
 - **Disk** 类：用于存储每张 CD 的详细信息，包括名称、价格和库存数量。
 - **storeUser** 类：继承自 **LinkedList<User>** 类，用于存储和管理所有会员信息。
 - **storeDisk** 类：继承自 **LinkedList<Disk>** 类，用于管理 CD 库存，提供进货、租赁和销售的操作方法。
 - **Map** 类：作为系统的业务逻辑核心，管理 **storeUser** 和 **storeDisk** 实例，执行 CD 和用户的增删查改操作以及租赁、购买逻辑。
2. **View 层**：展示层通过控制台与用户交互，输出系统的操作反馈。View 层包含提示和输出函数，将用户输入的信息传递至控制层进行处理，并展示操作结果。
 - **View** 类：向用户显示欢迎界面、帮助信息，并提供方法获取用户输入以便添加用户、删除用户、添加 CD、租赁和购买 CD。
3. **Controller 层**：
 - **Control** 类：作为控制器连接 View 层和 Model 层。控制器负责接收 View 层的用户指令，解析后调用相应的业务逻辑。控制器通过明确的事件处理流程，确保用户请求得到正确处理。

第四章 代码实现

第一节 User 类

User 类用于存储和管理会员的基本信息，包括会员的姓名、ID、余额和借入的 CD 信息。User 类提供了访问和修改这些属性的方法，并通过 equals 和 hashCode 方法保证会员信息的一致性。

```
1 package Bean;
2
3 import java.util.LinkedList;
4
5 public class User {
6     private String name;           // 用户姓名
7     private int id;                // 用户唯一ID
8     private double money;          // 用户余额
9     private LinkedList<Disk> borrowed; // 用户已借出的CD列表
10
11     public User(String name, int id, double money) {
12         this.name = name;
13         this.id = id;
14         this.money = money;
15         this.borrowed = new LinkedList<>();
16     }
17
18     public String getName() { return name; }
19     public int getId() { return id; }
20     public double getMoney() { return money; }
21     public LinkedList<Disk> getBorrowed() { return borrowed; }
22
23     public void setName(String name) { this.name = name; }
24     public void setId(int id) { this.id = id; }
25     public void setMoney(double money) { this.money = money; }
26     public void setBorrowed(Disk disk) { this.borrowed.add(disk); }
```



```
); }

27
28 @Override
29 public String toString() {
30     return "User [name=" + name + ", id=" + id + ", money="
31         + money + ", borrowed=" + borrowed + "]";
32 }
33
34 @Override
35 public int hashCode() {
36     final int prime = 31;
37     int result = 1;
38     result = prime * result + id;
39     result = prime * result + ((name == null) ? 0 : name.
40         hashCode());
41     long temp = Double.doubleToLongBits(money);
42     result = prime * result + (int) (temp ^ (temp >>> 32));
43     return result;
44 }
45
46 @Override
47 public boolean equals(Object obj) {
48     if (this == obj) return true;
49     if (obj == null) return false;
50     if (getClass() != obj.getClass()) return false;
51     User other = (User) obj;
52     if (id != other.id) return false;
53     if (name == null) {
54         if (other.name != null) return false;
55     } else if (!name.equals(other.name)) return false;
56     return true;
57 }
```

第二节 Disk 类

Disk 类用于存储 CD 的基本信息，包括名称、价格和库存量。通过提供 get 和 set 方法，可以访问和修改这些属性。该类还重写了 equals 和 hashCode 方法，以确保 CD 数据的一致性。

```
1 package Bean;
2
3
4 public class Disk {
5     private String name;        // CD 名称
6     private double money;       // CD 价格
7     private int num;           // CD 库存量
8
9     public String getName() { return name; }
10    public double getMoney() { return money; }
11    public int getNum() { return num; }
12
13    public void setName(String name) { this.name = name; }
14    public void setMoney(double money) { this.money = money; }
15    public void setNum(int num) { this.num = num; }
16
17    @Override
18    public String toString() {
19        return "Disk [name=" + name + ", money=" + money + ",
20            num=" + num + " ]";
21    }
22
23    @Override
24    public int hashCode() {
25        final int prime = 31;
26        int result = 1;
27        long temp;
28        temp = Double.doubleToLongBits(money);
29        result = prime * result + (int) (temp ^ (temp >>> 32));
30        result = prime * result + ((name == null) ? 0 : name.
```

```

        hashCode());
30    result = prime * result + num;
31    return result;
32    }
33
34    @Override
35    public boolean equals(Object obj) {
36        if (this == obj) return true;
37        if (obj == null) return false;
38        if (getClass() != obj.getClass()) return false;
39        Disk other = (Disk) obj;
40        if (name == null) {
41            if (other.name != null) return false;
42        } else if (!name.equals(other.name)) return false;
43        return true;
44    }
45    }

```

第三节 Map 类

Map 类是系统的业务逻辑核心类，负责管理会员和 CD 数据。它包含增加 CD、增加用户、删除用户、租赁和购买操作的实现，确保数据操作符合业务需求。

```

1
2 package Mapper;
3
4 import Bean.Disk;
5 import Bean.User;
6 import Bean.storeDisk;
7 import Bean.storeUser;
8
9 public class Map {
10     private storeDisk sDisk;
11     private storeUser sUser;

```

```
12
13     public Map() {
14         sDisk = new storeDisk();
15         sUser = new storeUser();
16     }
17
18     public void addDisk(Disk d) {
19         for (Disk disk : sDisk) {
20             if (d.equals(disk)) {
21                 disk.setMoney(d.getMoney());
22                 disk.setNum(d.getNum());
23                 return;
24             }
25         }
26         sDisk.add(d);
27     }
28
29     public void addUser(User u) {
30         for (User user : sUser) {
31             if (u.equals(user)) {
32                 System.out.println("User Duplicate");
33                 return;
34             }
35         }
36         sUser.add(u);
37     }
38
39     public void delUser(User u) {
40         if (!sUser.remove(u)) {
41             System.out.println("Delete Failed");
42         }
43     }
44
45     public void reportStore() {
```

```
46         System.out.println(sDisk);
47     }
48
49     public void reportUser() {
50         System.out.println(sUser);
51     }
52
53     public boolean rent(User rent) {
54         // 用户租赁CD逻辑
55         return false;
56     }
57
58     public boolean buy(User buy) {
59         // 用户购买CD逻辑
60         return false;
61     }
62 }
```

第四节 View 类

View 类负责与用户进行交互，展示系统提示信息并接收用户的输入。通过控制台，用户可以输入指令来执行各种操作，例如添加会员、添加 CD、删除会员和进行 CD 租赁或购买。

```
1
2 package Viewer;
3
4 import java.util.Scanner;
5 import Bean.Disk;
6 import Bean.User;
7
8 public class View {
9     private Scanner inScanner = new Scanner(System.in);
10
11     public void welcome() {
```

```
12         System.out.println("Welcome to CD Store! Enter 'h' for  
13             help.");  
14     }  
15     public String getMessage() {  
16         return inScanner.next();  
17     }  
18  
19     public User addUser() {  
20         System.out.println("Enter user name:");  
21         String name = inScanner.next();  
22         System.out.println("Enter user ID:");  
23         int id = inScanner.nextInt();  
24         System.out.println("Enter initial balance:");  
25         double balance = inScanner.nextDouble();  
26         return new User(name, id, balance);  
27     }  
28  
29     public User delUser() {  
30         System.out.println("Enter user ID to delete:");  
31         int id = inScanner.nextInt();  
32         return new User("", id, 0);  
33     }  
34  
35     public Disk addDisk() {  
36         System.out.println("Enter CD name:");  
37         String name = inScanner.next();  
38         System.out.println("Enter CD price:");  
39         double price = inScanner.nextDouble();  
40         System.out.println("Enter stock quantity:");  
41         int quantity = inScanner.nextInt();  
42         return new Disk(name, price, quantity);  
43     }  
44
```

```
45     public User rent() {
46         System.out.println("Enter user ID to rent CD:");
47         int id = inScanner.nextInt();
48         return new User("", id, 0);
49     }
50
51     public User buy() {
52         System.out.println("Enter user ID to buy CD:");
53         int id = inScanner.nextInt();
54         return new User("", id, 0);
55     }
56 }
```

第五节 Control 类

Control 类作为系统的控制器，通过 parser 方法解析用户输入的命令，并调用 Map 类的业务逻辑方法来执行用户请求的操作。

```
1
2 package Controller;
3
4 import java.util.Scanner;
5 import Viewer.View;
6
7 public class Control {
8     private Mapper.Map st;
9     private View vwView;
10    private Scanner cScanner = new Scanner(System.in);
11
12    public Control() {
13        st = new Mapper.Map();
14        vwView = new View();
15    }
16
17    public static void main(String[] args) {
```

```
18         Control boss = new Control();
19         boss.starthere();
20     }
21
22     private void starthere() {
23         vwView.welcome();
24         while (parser(vwView.getMessage()));
25     }
26
27     private void addUser() { st.addUser(vwView.addUser()); }
28     private void delUser() { st.delUser(vwView.delUser()); }
29     private void addCD() { st.addDisk(vwView.addDisk()); }
30     private void rentCD() { st.rent(vwView.rent()); }
31     private void buyCD() { st.buy(vwView.buy()); }
32
33     private boolean parser(String s) {
34         switch (s) {
35             case "h":
36             case "help":
37                 help();
38                 break;
39             case "b":
40             case "buy":
41                 buyCD();
42                 break;
43             case "r":
44             case "rent":
45                 rentCD();
46                 break;
47             case "ru":
48             case "reportUser":
49                 reportUser();
50                 break;
51             case "rd":
```



```
52         case "reportDisk":
53             reportStore();
54             break;
55         case "ac":
56         case "addCD":
57             addCD();
58             break;
59         case "au":
60         case "addUser":
61             addUser();
62             break;
63         case "du":
64         case "deleteUser":
65             delUser();
66             break;
67         case "exit":
68             return false;
69         default:
70             System.out.println("Unknown Command");
71             break;
72     }
73     return true;
74 }
75
76 private void help() {
77     System.out.println("This is Help: "
78         + "buy disks:buy/b "
79         + "rent disks:rent/r "
80         + "add user:addUser/au "
81         + "add disk:addCD/ac "
82         + "delete user:deleteUser/du "
83         + "check all users:reportUser/ru "
84         + "check all disks:reportDisk/rd "
85         + "exit:exit");
```

86	}
87	}

第五章 Maven 项目构建

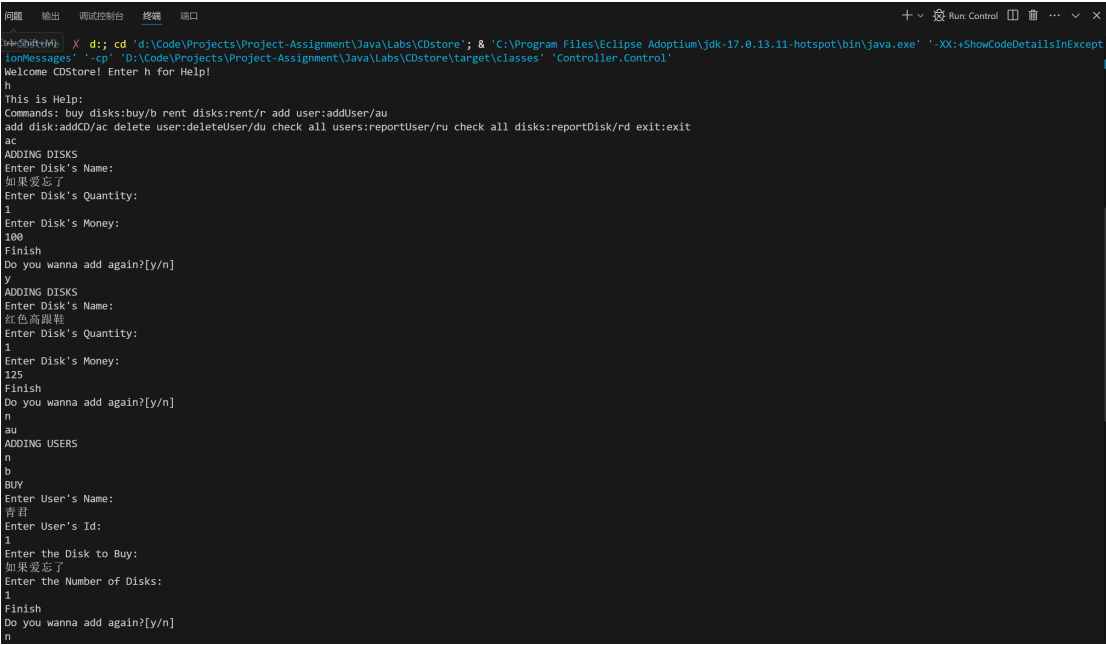
项目使用 **Maven** 进行构建和依赖管理，**Maven** 提供了自动化的依赖处理和构建流程，极大提升了开发和部署效率。

第六章 代码特点

1. **MVC 架构**: 分离 Model、View 和 Controller 层，实现模块化设计，提高了代码的清晰度和可维护性。
2. **数据一致性保障**: Dist 类中的 equals 和 hashCode 方法确保了数据的一致性。
3. **控制器解析**: Control 类通过 parser 方法解析用户输入，映射到具体的业务逻辑。
4. **面向对象设计**: 利用封装、继承和多态增强代码复用性。

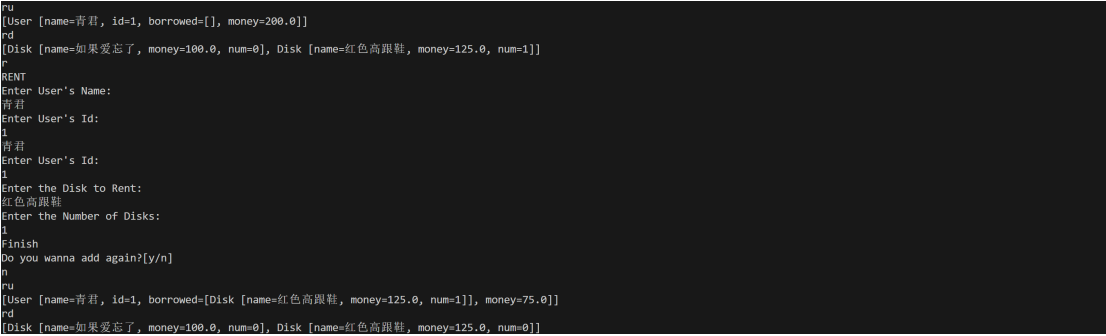
第七章 实验结果

以下为程序的典型运行流程，包括增加会员、租赁 CD、进货和库存汇报操作的截图。



```
问题 输出 调试控制台 终端 窗口
+ v Run: Control 11 100 ... v X
jw@Dell-M7: X d:; cd 'd:\Code\Projects\Project-Assignment\Java\Labs\CDstore'; & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.13-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Code\Projects\Project-Assignment\Java\Labs\CDstore\target\classes' 'Controller.Control'
Welcome CDStore! Enter h for Help!
h
This is Help:
Commands: buy disks:buy/b rent disks:rent/r add user:addUser/au
add disk:addCD/ac delete user:deleteUser/du check all users:reportUser/ru check all disks:reportDisk/rd exit:exit
ac
ADDING DISKS
Enter Disk's Name:
如果爱忘了
Enter Disk's Quantity:
1
Enter Disk's Money:
100
Finish
Do you wanna add again?[y/n]
y
ADDING DISKS
Enter Disk's Name:
红色高跟鞋
Enter Disk's Quantity:
1
Enter Disk's Money:
125
Finish
Do you wanna add again?[y/n]
n
au
ADDING USERS
n
b
BUY
Enter User's Name:
青君
Enter User's Id:
1
Enter the Disk to Buy:
如果爱忘了
Enter the Number of Disks:
1
Finish
Do you wanna add again?[y/n]
n
```

图 7.1 程序运行截图



```
ru
[User [name=青君, id=1, borrowed=[], money=200.0]]
rd
[Disk [name=如果爱忘了, money=100.0, num=0], Disk [name=红色高跟鞋, money=125.0, num=1]]
r
RENT
Enter User's Name:
青君
Enter User's Id:
1
青君
Enter User's Id:
1
Enter the Disk to Rent:
红色高跟鞋
Enter the Number of Disks:
1
Finish
Do you wanna add again?[y/n]
n
ru
[User [name=青君, id=1, borrowed=[Disk [name=红色高跟鞋, money=125.0, num=1]], money=75.0]]
rd
[Disk [name=如果爱忘了, money=100.0, num=0], Disk [name=红色高跟鞋, money=125.0, num=0]]
```

图 7.2 程序运行截图（续）

第八章 心得体会

通过本次实验的开发，我对 Java 编程中的 MVC 架构、面向对象编程（OOP）思想、Maven 构建工具的使用有了更深入的理解和体会。在设计和实现一个控制台版的 CD 出租销售管理系统的过程中，我遇到了许多值得思考的问题，也积累了宝贵的开发经验。

1. **深入理解了 MVC 架构的重要性：** MVC（Model-View-Controller）是一种非常有效的架构模式，通过将应用程序的逻辑划分为模型层、视图层和控制器层，能够大大提升代码的模块化和可维护性。此次实验中，我将数据操作、业务逻辑和用户交互分离到 Model、View 和 Controller 层，使得代码结构清晰，便于管理和扩展。具体来说，Model 层负责用户和 CD 数据的存储与管理，View 层处理用户的输入和输出，Controller 层则协调业务逻辑的执行。这样的设计让我更加直观地认识到 MVC 模式在复杂系统开发中的优势，同时也增强了我在设计代码结构时的思维方式。
2. **面向对象编程思想的深入应用：** 在项目开发过程中，面向对象编程的核心思想——封装、继承和多态得到了充分的应用。为了实现 CD 和用户的管理，我定义了 Disk 和 User 类，封装了每个对象的属性和行为。这种封装的设计模式不仅保护了数据的完整性，还提高了代码的复用性和可扩展性。继承在 storeUser 类中得到了体现，通过继承 LinkedList，能够直接利用链表的特性管理用户数据。在 Map 类中实现了 CD 租赁和购买逻辑，这个过程让我进一步理解了对对象之间关系的处理，也加深了对 Java 中接口、继承等概念的理解。
3. **Maven 构建工具的使用：** 实验中引入了 Maven 作为构建工具，使得项目的依赖管理和构建过程更加高效。Maven 能够自动下载和管理依赖库，简化了手动配置的工作。在团队协作或大型项目中，Maven 的自动化构建功能尤为重要，因为它能够确保项目的依赖一致性，简化了项目的打包和发布流程。通过本次实验，我不仅熟悉了 Maven 的基本配置，还了解了如何使用 Maven 构建更复杂的项目。
4. **数据一致性与逻辑完整性：** 在实现 Disk 和 User 类时，我重写了 equals 和 hashCode 方法，以确保在集合操作时能够正确比较对象，从而保证数据的唯一性和一致性。特别是在租赁和购买功能中，系统需实时更新

CD 的库存数量 and 用户余额，这样的操作需要仔细处理对象的状态，以确保逻辑的正确性。这让我体会到在开发数据密集型系统时，数据一致性和业务逻辑的完整性是非常关键的，稍有不慎就会引发数据错误。

5. **错误处理和用户体验：**在 Control 类的设计中，我特别注意了错误处理和用户提示信息的设计，例如在执行操作时检查输入的有效性、余额或库存是否足够等。在交互过程中通过适时的提示引导用户，比如确认操作、输出帮助信息、反馈执行结果等。这让我意识到，用户体验的好坏往往取决于细节处理，好的提示信息和完善的错误处理不仅可以减少用户的困惑，还可以提升整个系统的可用性。
6. **实践中遇到的挑战与解决：**在实际开发中，我遇到了一些挑战，比如在 rent 和 buy 功能中，如何确保库存的实时更新、用户余额的扣减和 CD 的借出管理等。解决这些问题的过程中，我学会了分步调试，分析代码的执行路径，逐步解决各个逻辑错误和漏洞。这一过程让我体会到开发中的问题解决能力尤为重要，同时也强化了我调试和分析代码的能力。
7. **未来的改进方向：**通过本次实验，我认识到系统还有一些可以优化的地方，例如引入日志记录功能来跟踪用户的操作记录，或进一步优化控制台的指令解析方式以支持更多复杂命令。此外，未来可以考虑在系统中加入数据库，将用户和 CD 数据永久存储，进一步提升系统的实用性和扩展性。

总结

本次实验不仅让我熟练掌握了 Java 编程的基础知识，还通过实战加深了对 MVC 架构、面向对象编程思想和 Maven 构建工具的理解。在开发过程中，遇到的问题让我学会了如何分析和解决实际编程中的挑战，从而提升了我的代码设计和逻辑思维能力。本实验为后续的复杂项目开发打下了坚实的基础，也让我更加自信地面对未来的编程挑战。