【实验7-2】 商城进货交易记录程序设计

### 【任务介绍】

**1.任务描述**

在日常工作中，经常会遇到批量操作系统文件的事情，通常情况下，只能手动重复的完成批量文件的操作，这样很是费时费力。本案例要求编写一个文件管理器，实现文件的批量操作。文件管理器具体功能要求如下：

1. 用户输入指令1，代表“指定关键字检索文件”，此时需要用户输入检索的目录和关键字，系统在用户指定的目录下检索出文件名中包含关键字的文件，并将其绝对路径展示出来。
2. 用户输入指令2，代表“指定后缀名检索文件”，此时需要用户输入检索的目录和后缀名（多个后缀名用逗号分隔），系统在用户指定的目录下检索出指定后缀名的文件，并将其绝对路径展示出来。
3. 用户输入指令3，代表“复制文件/目录”，此时需要用户输入源目录和目标目录，程序执行后会将源目录下的内容复制到目标目录下。
4. 用户输入指令4，代表“删除文件/目录”，此时需要用户输入需要删除掉的文件目录，程序执行后会将目录以及目录下的内容全部删除。
5. 用户输入指令5，代表“退出”，即退出该系统。

**2.运行结果**

任务运行结果如图7-1所示。

手机屏幕截图

描述已自动生成

图7-1 运行结果

运行结束后在本地生成一个“进货记录20191209.csv”文件，用Excel方式打开此文件，如图7-1所示。

手机屏幕截图

描述已自动生成

### 【任务目标】

* 学会分析"商城进货交易记录"程序的实现思路。
* 根据思路独立完成"商城进货交易记录"的源代码编写、编译及运行。
* 掌握字节流操作本地文件的方法。
* 掌握ArrayList和StringBuffer的使用，以及异常的处理
* 了解csv的文件格式。

### 【实现思路】

1. 为了方便保存商品的相关信息，可以将图书信息封装为一个实体类。商品进货过程中可能会打印图书相关信息，所以需要对该实体类的toString()方法进行重写，使其能更清晰地显示商品信息，商品每次进货后要修改库存数量，需要在实体类中编写一个操作库存数量的方法。
2. 对于一个超市，首先会有很多商品，商品需要不断进货。这里我们需要创建一个集合用于模拟超市仓库，然后向集合中添加有具体信息的商品对象，这样一个超市就有了商品。
3. 管理员进货是通过在控制台键盘输入商品编号和购买数量的方式进行的，如果商品编号正确，且购买数量也正常，则商品进货成功，并将此商品的进货信息保存到csv文件中，同时要将库存数量增加。
4. 查询商品信息时，可以通过Scanner类的nextInt()方法从控制台获取商品编号，之后根据这个编号到库存中查询此商品的信息，如果查到了商品的信息，从控制台获取进货的数量之后，将此商品的所有信息进行封装。
5. 将商品的销售信息写入到csv文件之前，需先拼凑好csv文件名，再判断本地是否已存在此文件，这里可通过输入流尝试获取此文件的字节流，如果获取成功，则证明这个文件已存在，那么就通过输出流向文件末尾追加销售信息，如果获取失败，即异常，说明之前并没有生成当日的销售信息，则需要新建此文件。
6. 将封装的信息写入csv文件中时，csv格式的文件以纯文本形式存储表格数据，写入文件时可以用图7-1的格式写入，当此类文件用Excel格式打开的时候，展现信息如图7-11。
7. 在拼凑csv文件名时，需要获取当日的日期。这里可以通过以下代码来获取并拼凑csv文件名：

DateFormat format = new SimpleDateFormat("yyyyMMdd");// 定义日期格式

String name = "进货记录" + format.format(date) + ".csv";// 拼接文件名

### 【实现代码】

（1）将商品信息封装成一个实体类Good，具体如文件7-1所示。

文件7-1 Good.java

1. package chapter0701;
2. public class Good {
3. int id;
4. String name; //商品的价格
5. double price; //商品的单价
6. int number; //进货的数量
7. double money; //总价
8. String people; //审批人
9. public Good(int id, String name, double price, int number, double
10. money, String people) {
11. this.id = id;
12. this.name = name;
13. this.price = price;
14. this.number = number;
15. this.money = money;
16. this.people = people;
17. }
18. @Override
19. public String toString() {
20. String message="进货记录编号："+id+"\n商品名称："+name+"\n联系
21. 人："+people+"\n单价："+price+"\n库存数量："+number+"\n";
22. return message;
23. }
24. public void setNumber(int number) {
25. this.number=number;
26. }
27. }

在文件7-1中，第3~8行代码定义了用于标识商品的信息各种字段，，第9~17行代码定义了一个有参的构造方法，用于对象的创建和初始化，在第18~23行，重写了toString()方法，用于返回商品的详细信息。第24~26行的代码定义了一个setNumber()的方法，用于修改商品的库存量。

（2）定义RecordGoodOrder类来记录和操作商品信息，具体如文件7-2所示。

文件7-2 RecordGoodOrder.java

1. package chapter0701;
2. import java.util.ArrayList;
3. import java.util.Scanner;
4. public class RecordGoodOrder {
5. //创建商品库存
6. static ArrayList<Good> goodsList=new ArrayList<Good>();
7. public static void main(String[] args) {
8. init(); //初始化商品库存
9. //将书架上所以商品信息打印出来
10. for (int i = 0; i < goodsList.size(); i++) {
11. System.out.println(goodsList.get(i));
12. }
13. while(true) {
14. //获取控制台输入的信息
15. Scanner scan=new Scanner(System.in);
16. System.out.println("请输入商品编号");
17. int goodId=scan.nextInt();
18. Good stockGood=getGoodsById(goodId);
19. if (stockGood != null) {// 判断是否存在此商品
20. System.out.println("当前商品库存信息" + stockGood);
21. System.out.print("请输入进货数量：");
22. int bookNumber = scan.nextInt();
23. // 将输入信息封装成Books对象
24. Good good = new Good(stockGood.id, stockGood.name,
25. stockGood.price, bookNumber, stockGood.price
26. \* bookNumber, stockGood.people);
27. FileUtil.saveBooks(good);// 将本条数据保存至本地文件
28. // 修改库存
29. stockGood.setNumber(stockGood.number + bookNumber);
30. } else {
31. System.out.println("商品编号输入错误！");
32. }
33. }
34. }
35. /\*
36. \* 初始化商品库存的信息 将商品存入库存
37. \*/
38. private static void init() {
39. Good good1=new Good(1001,"百事可乐",4.5,100,450,"张三");
40. Good good2=new Good(1002,"可口可乐",4,100,400,"李四");
41. Good good3=new Good(1003,"百事雪碧",3.8,100,380,"张三");
42. goodsList.add(good1);
43. goodsList.add(good2);
44. goodsList.add(good3);
45. }
46. /\*
47. \* 根据输入的商品编号查找图书信息，循环遍历库存中商品信息，找到商品编号相等的取出
48. \*/
49. private static Good getGoodsById(int goodId) {
50. for (int i = 0; i < goodsList.size(); i++) {
51. Good thisGood=goodsList.get(i);
52. if (goodId==thisGood.id) {
53. return thisGood;
54. }
55. }
56. return null;
57. }
58. }

在文件7-8中，第6行代码创建了ArrayList类型的全局变量作为商品的仓库。第39~46代码，初始化了商品信息，向ArrayList中添加了3种商品的信息，并在第10~12行代码中，通过for循环进行展示。第13~34行代码使用while循环来获取和处理用户输入信息，每次循环先由第16~17行代码，从控制台获取商品编号的数据，再由第49~57行的代码，根据商品编号查询到商品信息，当获得的商品信息不为空时，可从第21~22代码获得购买的数量，可通过24~29的代码，将所有数据封装，再利用第27行代码，调用FileUtil类中的saveBooks()方法，将其保存至本地。最后在第29行代码中调用setNumber()方法，修改库存。

（3）定义工具类FileUtil保存图书信息，具体如文件7-3所示。

文件7-3 FileUtil.java

1. package chapter0701;
2. import java.io.BufferedOutputStream;
3. import java.io.FileInputStream;
4. import java.io.FileNotFoundException;
5. import java.io.FileOutputStream;
6. import java.io.IOException;
7. import java.io.InputStream;
8. import java.text.DateFormat;
9. import java.text.SimpleDateFormat;
10. import java.util.Date;
11. /\*\*
12. \* 工具类
13. \*/
14. public class FileUtil {
15. public static final String SEPARATE\_FIELD = ",";// 字段分隔 英文逗号
16. public static final String SEPARATE\_LINE = "\r\n";// 行分隔
17. /\*\*
18. \* 保存商品信息
19. \*/
20. public static void saveBooks(Good good) {
21. // 判断本地是否存在此文件
22. Date date = new Date();
23. // 定义日期格式
24. DateFormat format = new SimpleDateFormat("yyyyMMdd");
25. // 拼接文件名
26. String name = "进货记录" + format.format(date) + ".csv";
27. InputStream in = null;
28. try {
29. in = new FileInputStream(name);// 判断本地是否存在此文件
30. if (in != null) {
31. in.close();// 关闭输入流
32. // 可获取输入流，则存在文件，采取修改文件方式
33. createFile(name, true, good);
34. }
35. } catch (FileNotFoundException e) {
36. // 输入流获取失败，则不存在文件，采取新建新文件方式
37. createFile(name, false, good);
38. } catch (IOException e) {
39. e.printStackTrace();
40. }
41. }
42. /\*\*
43. \* 将进货记录的信息保存到本地，可通过label标识来判断是修改文件还是新建文件
44. \* @param name 文件名
45. \* @param label 文件已存在的标识 true：已存在则修改； false：不存在则新建
46. \* @param good 商品信息
47. \*/
48. public static void createFile(String name,boolean label,Good good) {
49. BufferedOutputStream out = null;
50. StringBuffer sbf = new StringBuffer();// 拼接内容
51. try {
52. if (label) {// 当已存在当天的文件，则在文件内容后追加
53. // 创建输出流，用于追加文件
54. out = new BufferedOutputStream(new FileOutputStream(name, true));
55. } else {// 不存在当天文件，则新建文件
56. // 创建输出流，用于保存文件
57. out = new BufferedOutputStream(new FileOutputStream(name));
58. String[] fieldSort = new String[] { "商品编号", "商品名称", "购买数量","
59. 单价", "总价", "联系人" };// 创建表头
60. for (String fieldKye : fieldSort) {
61. // 新建时，将表头存入本地文件
62. sbf.append(fieldKye).append(SEPARATE\_FIELD);
63. }
64. }
65. sbf.append(SEPARATE\_LINE);// 追加换行符号
66. sbf.append(good.id).append(SEPARATE\_FIELD);
67. sbf.append(good.name).append(SEPARATE\_FIELD);
68. sbf.append(good.number).append(SEPARATE\_FIELD);
69. sbf.append((double) good.price).append(SEPARATE\_FIELD);
70. sbf.append((double) good.money).append(SEPARATE\_FIELD);
71. sbf.append(good.people).append(SEPARATE\_FIELD);
72. String str = sbf.toString();
73. byte[] b = str.getBytes();
74. for (int i = 0; i < b.length; i++) {
75. out.write(b[i]);// 将内容写入本地文件
76. }
77. } catch (Exception e) {
78. e.printStackTrace();
79. } finally {
80. try {
81. if (out != null)
82. out.close();// 关闭输出流
83. } catch (Exception e2) {
84. e2.printStackTrace();
85. }
86. }
87. }
88. }

在文件7-3中，当saveBooks()方法被调用时，获取当前日期并格式化后，拼出了文件名，再通过第29行代码尝试获取此文件的字节输入流。当能够获取输入流时，可通过第30~34行代码，先关闭输入流，再在文件末尾追加信息。当不能获取输入流时则抛出异常，在异常处理中调用第37行代码的createFile()方法，可以通过此方法中的label参数来区分，是新建文件还是在已有文件中追加内容，如果label值是true则追加内容，如果label值是false则新建文件，并写入表头，其中进行追加还是新建操作，由构造函数的append参数来定义。然后利用65~72行代码拼出一行数据，且在每次拼接之前都要加上换行符“\r\n”，每个字段之间通过“,”分隔字段，再利用第73~76行的代码写入文件。最后80~86行代码关闭了输出流。