【实验7-4】 升级版日记本

### 【任务介绍】

**1.任务描述**

本案例要求编写一个模拟日记本的程序，通过在控制台输入指令，实现在本地新建日记本、打开日记本和修改日记本等功能。

1. 用户输入指令1代表“新建日记本”，可以从控制台获取用户输入的日记内容。
2. 指令2代表“打开日记本”，读取指定路径的txt文件的内容并输出到控制台。
3. 指令3代表“修改日记本”，修改日记时，既可以修改新建日记本的内容，也可以修改已打开日记本的内容。
4. 指令4代表“保存”，如果是新建的日记本需要保存，则将日记本保存到用户输入的路径；如果是打开的日记本需要保存，则将原来内容覆盖；
5. 指令5代表“退出”，即退出本系统。

**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

运行过程中，本地D:\chapter07\task02路径下生成了note.txt文件，打开后如图7-2所示。

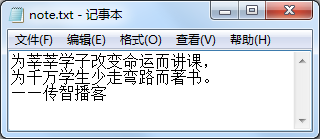


图7-2 生成的文件

运行结束后，本地D:\chapter07\task02路径下的note.txt文件已经修改，打开后如图7-3所示。

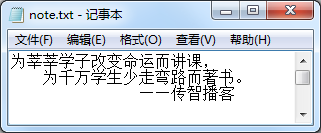


图7-3 修改后的文件

### 【任务目标】

* 学会分析“日记本”任务的实现思路。
* 能够根据思路独立完成“模拟记事本”程序的源代码编写、编译及运行。
* 掌握利用字符流操作本地文件的方法。

### 【实现思路】

（1）根据任务介绍和运行结果分析可知，此任务需要创建一个记事本类，在类中可以使用while循环实现控制台中操作指令的多次输入，使用switch语句根据控制台输入的操作指令来判断执行什么操作。

（2）输入指令1时进行新建文件操作，此时需要从控制台获取输入的内容并暂存到全局变量中，以便后期进行保存和修改内容时使用。可以使用StringBuffer来保存控制台中每次输入的内容，输入内容时可能会涉及换行，这里可以使用while循环来循环输入，每循环一次都将控制台获取的内容拼接换行符后，追加到文本内容中，当内容中输入“stop”字符串时，输入文本内容的操作结束，并将新建文本的内容暂存在全局变量中。

（3）执行指令2时，可以打开指定路径的文件，由于是模拟记事本程序，因此限定此功能只支持打开txt文件。此操作首先要获取输入的路径，判断此路径的文件是否是txt文件，如果是则通过字符流读取此文件，然后将此文件的内容打印至控制台，并且暂存在全局变量中，将文件的路径也保存至全局变量中。

（4）执行指令3时，可以对暂存在全局变量中的文本内容进行修改，修改后的内容也需要暂存到全局变量中。此操作首先要判断一下修改之前是否先经过新建文件或者打开文件操作，当确认经过上述操作后，可以将输入的字符串与文本内容中目标字符串进行替换，来完成修改内容功能。修改过程可通过while循环来进行多次的修改，当捕捉到“stop”字符串时修改结束，并将修改后的内容暂存到全局变量中。

（5）执行指令4时，将所有全局变量中的内容保存至本地，如果是新建的文件则保存至用户输入的路径，如果是打开后的文件则将原文件覆盖。可以通过文件路径的全局变量判断是哪种保存方式，当是打开文件的方式时，直接通过字符流写入；当是新建的方式时，先获取用户输入的路径，再用字符流写入。

（6）执行指令5，直接退出系统，可以通过代码“System.exit(0);”实现。

### 【实现代码】

（1）创建记事本类，在类中编写执行程序的main方法，如文件7-1所示。

1. Notepad.java
2. package cn.itcast.chapter07.task02;
3. import java.io.FileReader;
4. import java.io.FileWriter;
5. import java.io.IOException;
6. import java.util.Scanner;
7. public class Notepad {
8. private static String filePath;
9. private static String message = "";
10. public static void main(String[] args) throws Exception {
11. Scanner sc = new Scanner(System.in);
12. System.out.println("--1:新建文件 2:打开文件 3:修改文件 4:保存 5:退出--");
13. while (true) {
14. System.out.print("请输入操作指令：");
15. int command = sc.nextInt();
16. switch (command) {
17. case 1:
18. createFile();//1:新建文件
19. break;
20. case 2:
21. openFile();// 2:打开文件
22. break;
23. case 3:
24. editFile();//3:修改文件
25. break;
26. case 4:
27. saveFile();// 4:保存
28. break;
29. case 5:
30. exit();//5:退出
31. break;
32. default:
33. System.out.println("您输入的指令错误！");
34. break;
35. }
36. }
37. }
38. }

在文件7-1中，第7行和第8行分别创建了文件路径和内容的全局变量。第12~35行代码，是循环主体，用于循环接收用户的指令。当从第14行代码获取指令后，通过switch语句判断应该执行的操作。

（2）在Notepad中编写新建文件方法，如文件7-2所示。

1. createFile()
2. /\*\*
3. \* 新建文件 从控制台获取内容
4. \*/
5. private static void createFile() {
6. message = "";//新建文件时，暂存文件内容清空
7. Scanner sc = new Scanner(System.in);
8. System.out.println("请输入内容，停止编写请输入\"stop\":");//提示
9. StringBuffer stb = new StringBuffer();//用于后期输入内容的拼接
10. String inputMessage = "";
11. while (!inputMessage.equals("stop")) {//当输入“stop”时，停止输入
12. if (stb.length() > 0) {
13. stb.append("\r\n");//追加换行符
14. }
15. stb.append(inputMessage);//拼接输入信息
16. inputMessage = sc.nextLine();//获取输入信息
17. }
18. message = stb.toString();//将输入内容暂存
19. }

在文件7-2中，createFile()方法是Notepad类中的新建文件方法。第5行代码用于将文件内容清空。第8行代码定义了一个字符串缓冲区StringBuffer，用于存储每次控制台输入的内容。第10~16行代码，通过while循环，使控制台可以多次输入内容，每次输入结束后，就将输入字符串拼入字符串缓冲区中，并在末尾追加一个换行符，直至输入“stop”字符串时循环结束，输入停止。第17行代码，将StringBuffer中的数据转成String类型，并存入全局变量message中，以便修改和保存时使用。

（3）在Notepad中编写打开文件方法，如文件7-3所示。

1. openFile()
2. /\*\*
3. \* 打开文件
4. \*/
5. private static void openFile() throws Exception {
6. message = "";//打开文件时，将暂存内容清空
7. Scanner sc = new Scanner(System.in);
8. System.out.print("请输入打开文件的位置：");
9. filePath = sc.next();//获取打开文件的路径
10. //控制只能输入txt格式的文件路径
11. if (filePath != null && !filePath.endsWith(".txt")) {
12. System.out.print("请选择文本文件！");
13. return;
14. }
15. FileReader in = new FileReader(filePath);//实例化一个FileReader对象
16. char[] charArray = new char[1024];//缓冲数组
17. int len = 0;
18. StringBuffer sb = new StringBuffer();
19. // 循环读取，一次读取一个字符数组
20. while ((len = in.read(charArray)) != -1) {
21. sb.append(charArray);
22. }
23. message = sb.toString();//将打开文件内容暂存
24. System.out.println("打开文件内容：" + "\r\n" + message);
25. in.close();// 释放资源
26. }

在文件7-3中，openFile()方法是Notepad类中的打开文件方法。第5行代码实现将内容的message全局变量清空。第6~8行获取到控制台传入的文件打开的路径，并将其存储到filePath全局变量中。第10~13行代码，判断将要打开的文件的格式是否是文本文件。第14~21行代码，利用了字符输入流读取文件的内容。第22行代码，将StringBuffer中的内容转成String类型，并存入全局变量message中，以便修改和保存时使用。

（4）在Notepad中编写修改文件方法，如文件7-4所示。

1. editFile()
2. /\*\*
3. \* 修改文件内容 通过字符串替换的形式
4. \*/
5. private static void editFile() {
6. if(message == "" && filePath == null){
7. System.out.println("请先新建文件或者打开文件");
8. return;
9. }
10. Scanner sc = new Scanner(System.in);
11. System.out.println("请输入要修改的内容（以 \"修改的目标文字:修改之后的文字\" 格式）,"
12. + "停止修改请输入\"stop\":");
13. String inputMessage = "";
14. while (!inputMessage.equals("stop")) {//当输入stop时,停止修改
15. inputMessage = sc.nextLine();
16. if (inputMessage != null && inputMessage.length() > 0) {
17. // 将输入的文字根据“：”拆分成数组
18. String[] editMessage = inputMessage.split(":");
19. if (editMessage != null && editMessage.length > 1) {
20. //根据输入的信息将文件中内容替换
21. message=message.replace(editMessage[0], editMessage[1]);
22. }
23. }
24. }
25. System.out.println("修改后的内容:" + "\r\n" + message);
26. }

在文件7-4中，editFile()是Notepad类中的修改文件方法。第13~23行代码，使用while循环实现多次替换文件内容的功能，每次由第14行代码获取替换的指令，再由第17行代码将其根据“:”拆分成数组（:是英文格式的），并再将内容的全局变量message进行替换，此时注意判断数组长度，否则容易报数组下标越界的错误。

（5）编写保存文件方法，如文件7-5所示。

1. saveFile()
2. /\*\*
3. \* 保存 新建文件存在用户输入的路径 打开的文件将原文件覆盖
4. \*/
5. private static void saveFile() throws IOException {
6. Scanner sc = new Scanner(System.in);
7. FileWriter out = null;
8. if (filePath != null) {// 文件是由“打开”载入的
9. out = new FileWriter(filePath);//将原文件覆盖
10. } else {//新建的文件
11. System.out.print("请输入文件保存的绝对路径：");
12. String path = sc.next();//获取文件保存的路径
13. filePath = path;
14. //将输入路径中大写字母替换成小写字母后判断是不是文本格式
15. if (!filePath.toLowerCase().endsWith(".txt")) {
16. filePath += ".txt";
17. }
18. out = new FileWriter(filePath);// 构造输出流
19. }
20. out.write(message);//写入暂存的内容
21. out.close();//关闭输出流
22. message = "";// 修改文件前现将写入内容置空
23. filePath = null;//将文件路径至null
24. }

在文件7-5中，saveFile()方法是Notepad类中的保存文件的方法。第7行代码，根据路径的全局变量是否为空，来判断此时内容的全局变量是新建文件产生的还是打开文件产生的，如果是打开文件产生的，则用第8行代码，将此文件覆盖掉，如果是新建文件产生的，则执行第10~17行代码。第11行代码获得控制台输入的路径之后，执行第14到16行的代码判断其是否是文本格式，如果不是将其转换成文本格式。第19行代码将message中暂存的内容写入文本文件。最后第21和22行的代码将文件路径和文件内容的全局变量message置空，完成了一次对文本的操作。

（6）编写退出程序的方法，如文件7-6所示。

1. exit()
2. /\*\*
3. \* 退出
4. \*/
5. private static void exit() {
6. System.out.println("您已退出系统，谢谢使用！");
7. System.exit(0);
8. }

在文件7-6中，exit()方法是Notepad类中的退出的方法。第6行代码执行了退出的命令。