面向对象第六次课程作业指导说明

一、程序功能

如指导书中所说，该程序可以实现对中小规模文件进行简单监控操作的功能。

二、程序运行所需环境和使用规范

运行环境：与课程提供的 JDK 和 Eclipse 相同即可。

运行步骤 :将全部.java 导入工程文件, 从属 fileScan 包, 该工程文件中需要有 JRE System Library，之后运行 run Begin 文件，即可在控制台输入 IF THEN 信息。

三、正确输入格式和使用操作说明

1. 控制台输入：

IF THEN 语句的使用，要求需满足如下格式，"IF XXX trigger_type THEN task_type"。

其中 trigger_type 有如下四种，renamed，modified，pathchanged，sizechanged，需要完全匹配；task_type 有如下三种，summary，detail，recover,同样需要完全匹配。

在上述格式中，每两个部分之间有且仅有一个空格，同时，限制文件或目录中不能出现空格字符，如果有会显示 Not Match 字样提示。

对于系统内要求的不能作为文件名的字符，输入中未作要求，但是由于使用 testCase 或者提前布置现场的方法均无法创建，所以实际上属于不会起作用的无效输入。同时由于不同操作系统对文件名长度限制有一定差异，限制文件名不超过 20 个字符。

2.TestCase 制造与使用

应指导书要求，除了提供 testThread 类可以供测试者进行修改，和 main 中 testThread.start()的位置可在一定范围内移动以外，其他程序部分按理说不能修改，因为修改后无法保证程序的正确性，所以不建议改动。

提供的 testThread 类有如下几种方法：

Boolean addFile(String file_name);

Boolean makeDirectory(String file_name);

Boolean rename(String from, String to);

Boolean deleteFile(String file_name);

Boolean move(String from, String to);

Boolean changeSize(String file_name);

Boolean changeTime(String file_name);

其中，addFile 的参数 file_name 为文件的绝对路径，如 addFile(“H:\\1\\1.txt”)，表示增加所在目录为 H:\1 的 1.txt 文件，由于 JAVA 转义字符的影响，字符串内应当以\\来表示文件的层级关系。

MakeDirectory（“H:\\1\\2”），表示新建一个所在目录为 H:\1 的文件夹 2.

Rename（”H:\\1.txt”, “H:\\2.txt”），表示将目录为 H:\的 1.txt 文件改名为 2.txt，使用时需注意保证前后两者的所在目录相同，以满足 rename 的含义。

DeleteFile（”H:\\1.txt”），表示将目录为 H:\的 1.txt 文件删除，文件删除的结果不会对监控产生作用。

Move（”H:\\1.txt”, “H:\\1\\1.txt”），表示将目录为 H:\的 1.txt 文件移动至目录为 H:\1 的地方，使用时需注意保证前后两者的名称相同，以满足 move 的含义。

ChangeSize（“H:\\1.txt”），表示修改目录为 H:\的 1.txt 文件大小和上一次修改时间。

ChangeTime（”H:\\1.txt”），表示修改目录为 H:\的 1.txt 文件上一次修改时间。

上述文件均可以自由组合，同时建议使用 Thread.Sleep()保证不同操作之间有一定的间隔，以防止扫描结束周期内多次操作导致的结果合并缺失，示例如下。

```
1.  public boolean testcase(){
2.      int i;
3.      addFile("H:\\1.txt");
4.      for(i= 0; i < 10; i++){
5.          try {
6.              Thread.sleep(300);
7.          } catch (InterruptedException e) {
8.              // TODO Auto-generated catch block
9.              e.printStackTrace();
10.         }
11.         if(!changeSize("H:\\3.txt"))
12.             return false;
13.     }
14.     if(!rename("H:\\1.txt", "H:\\2.txt"))
15.         return false;
16.     try {
17.         Thread.sleep(600);
18.     } catch (InterruptedException e) {
19.         // TODO Auto-generated catch block
20.         e.printStackTrace();
21.     }
```

```
    if(!deleteFile("H:\\2.txt"))
        return false;

    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return true;
}
```

一切测试请在 testcase（）中完成，具体使用方式与课上讲义相同。

注意，run（）方法请不要修改。

```
public void run(){
    if(!testcase()){
        System.out.println("File Operation Error.");
    }
    else{
        System.out.println("File Operation Over.");
    }
    file_operation_over = true;
}
```

当 testcase（）中有某一步操作失败后，程序将终止测试样例，可能的失败比如 add 一个已有的 File，delete 一个没有的文件等不合理的操作。

3.输出结果的说明

运行后会在程序所在的目录下生成两个文件，detail.txt 和 summary.txt，其中 detail 存放所有应该被记录的文件修改细节，包括大小、名称、路径和修改时间，summary 中存放某一类触发器被触发，且要求执行 summary 记录的次数，包括触发器类型和被要求记录的次数。

比如，

IF H:\1.txt modified THEN detail
IF H:\1.txt modified THEN summary
IF H:\ modified THEN detail
END

上述对 1.txt 进行 changeTime 操作，会被记录两次 detail 和一次 summary；

IF H:\1.txt modified THEN detail
IF H:\1.txt modified THEN summary
IF H:\ modified THEN summary
END

而上述对 1.txt 进行 changeTime 操作，会被记录一次 detail 和两次 summary。

四、特殊情况和边界可能

1.多个 recover 只会触发一次

比如：

IF H:\ renamed THEN recover

IF H:\1.txt renamed THEN recover

当对 1.txt 文件进行 rename 操作时，屏幕提示会出现两次，但是实际只会执行一次，保证 Rename 操作被取消。

2.对于多次 rename 操作

比如：

IF H:\1.txt renamed THEN summary

rename("H:\\1.txt", "H:\\2.txt")) -- A

rename("H:\\2.txt", "H:\\3.txt")) -- B

若 AB 操作之间的空隙不够，在两次扫描中完成了 AB 两个操作，那么输出将只有 1.txt -> 3.txt；若 A、B 操作之间的空隙足够长，跨过一次扫描，那么将有两个输出 1.txt -> 2.txt 与 2.txt -> 3.txt。该条说明对 Detail 和 Summary 均有效。

3.由于指导书说明极度不明确，截止到作业提交结束前，都未搞清楚监控对象不超过 10 个是什么意思，所以限制有效且不重复的 IF THEN 语句 120 个，超出范围会提示 Monitors Number Over 120!，剩下的怎么都好 TAT。

4.关于程序的扫描周期，小数据量的时候扫描周期为 1S，基本可以保证准确，但是当监控对象层次结果过深时，会出现扫描时间极长的可能，所以应当避免结构过深可能导致的栈溢出与等待时间延长。

5.若由于监控的文件过多，在扫描时间内未能完成对监控对象的遍历，而此时程序会认为该段时间内扫描范围中没有发生任何变化。若此时发生文件变动，很有可能不能被记录，比如监控自己堆满杂物的系统盘等。可以通过增加 testThread 的线程时间来解决，或者其实小一点的数据量就能看出来程序有没有问题了。

6.对于名称不同、其他均相同的文件，不应在同一个扫描周期内同时操作，否则对于 rename 与 changePath 的可能出现 AB 均变为 C，而 D 文件被忽略的情况。

7.对于 testThread.start()测试线程开始的位置，在 Begin 类中 main 函数里有一小块地方使用。

```
125         /*********
126          * Here
127          * 这里运行线程
128          * 运行时把注释删掉即可
129 
130         testThread.start();
131 
132         *********/
```