Readme

tags：面向对象课程文档 Java

程序运行所需环境和运行指令规范

```
Java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
Eclipse IDE for Java Developers Version: Neon Release (4.6.0)
程序输入由Eclipse控制台输入
```

程序输入说明，包括标准输入格式、输入限制和遇见输入错误时的响应信息

1. 标准输入格式

- 正确的输入格式为 `IF [监视的文件或目录] 触发器 THEN 任务`
- 几点说明：1）监视的文件或目录必须以 `[]` 包围；2）各个部分之间以空格分隔，不允许有多余空格，文件或目录内允许有空格；
- 触发器和任务组合与指导书上相同，其中 `recover` 任务只能与 `renamed` 与 `path-changed` 触发器组合
- 不同任务分行输入，最后输入 `start` 开始执行监视，复制粘贴多行是不符合本程序输入规范的

2. 输入限制

- 不符合正确输入格式的都被视为错误
- 完全相同的任务将会被忽略
- 监视文件或目录如果不存在将会被视为错误
- 输入的文件或目录的正确格式应该是操作系统下（windows10）的正确的完整绝对路径，例如目录间以多个'\'分隔将会被视为错误，路径的结尾如果有多余的'\'也会被视为错误；示例 正确路径：`f:\movie\test`；错误路径：`f:\\movie\\test` 或 `f:\movie\`

3. 遇见输入错误时的响应信息

- 被视为错误的任务将会输出 `INVALID 原始请求` 并不会执行

程序计算结果的输出规格，以及可预见的运行错误响应信息

- 对于任务 `record-summary` 每当触发器被触发，将在控制台输出被触发的触发器类型和已触发次数；每隔10s会向文件 summary.txt（在项目目录下）中写入四种触发器和对应触发次数，写入是覆盖的

- 对于任务 `record-detail` 每当触发器被触发，将在控制台输出对应信息，格式大体为 监视文件/目录+触发类型+变化前的路径（如果变化前没有文件即新增文件，则用null代替）+变化前的信息+变化后的路径（如果变化后没有文件即删除文件，则用null代替）+变化后的信息；每隔10s会向文件detail.txt（在项目目录下）中写入新增的触发信息，写入是追加的
- 对于任务 `recover` 不会输出信息，但是可以观察到变化

提供测试的文件访问类使用说明

- 工程中提供了一个 `filedemo` 线程，里面将所有测试者使用到的关于文件属性读取以及重命名与移动文件等操作的方法都列举出来并提供了示例；测试者需要用到两个类中的若干方法
- 最为重要的一点是取得文件的方法只能是 `fileshell (FileList类)` 的 `get_shell(String path)` 方法，该方法会返回一个 `safe_file` 对象；所以如果测试者想要自己构建测试线程，需要在线程的构造函数中传入main函数中已经构造好的 `FileList` 对象（可以参考 `filedemo`）
- 可以通过 `safe_file` 直接获取文件信息，如果系统中对应路径下的文件或目录不存在，依然可以获取指导书要求的名称、大小、修改时间信息
- 可以通过 `safe_file` 的 `createNew(String type)` 来新增文件或目录，只有当系统中对应路径下的文件或目录不存在且路径正确时才会新建；如果对应文件或目录已经存在，将输出提示信息并不会新建；如果对应文件或目录的路径上存在系统中没有的目录，则是路径错误，也不会新建；type 为 File 或 Directory
- 可以通过`safe_file` 的 `delete()` 方法来删除对应的文件或文件夹，同样只有当系统中对应路径下的文件或目录存在时才会执行删除
- 文件的写入，重命名，移动文件都需要调用 `fileshell` 的方法；
  - 1）文件的写入需要调用 `fileshell` 的 `write(safe_file file,String content,boolean f)` 方法，其中 file 为对应的 `safe_file` 对象，content 为写入的内容，f 如果为 `true` 则表示在文件末尾追加内容，文件夹是不可写的；
  - 2）文件的重命名需要调用 `fileshell` 的 `rename(safe_file file,String newname)`，需要注意的是该函数会返回一个safe_file对象，使用时需要以 `file = fileshell.rename(file,newname)` 的形式，文件夹是不提供重命名的；
  - 3）文件的移动需要调用 `fileshell` 的 `moveTo(safe_file file,String derectory)`，需要注意的是该函数同样会返回一个safe_file对象，使用时需要以 `file = fileshell.moveTo(file,directory)` 的形式，文件夹是不提供移动的；
- 以上方法可以满足指导书中提出的所有测试要求，两个类的其他方法是程序其他部分使用的，不提供给测试者使用
- 关于`filedemo`：filedemo已经写好，在main函数中也被调用但被注释；filedemo中的文件路径可以自行修改，测试者可以直接在filedemo修改其他内容，也可以自己新建线程，但要注意FileList对象的传入，不可以自己新建FileList对象，方法使用规范已在Readme中说明
- 需要注意的是，监视线程的扫描间隔是 1s，如果在filedemo或者自己建立的线程的自动化测试中监视线程没有触发输出结果，有可能是对应的两次操作如删除又新建等太快了，正好在两次扫描中间，根据指导书触发器不会触发，建议在两次操作中间休眠一段时间或者手动测试，不是程序的问题