第六次 OO 作业指导书 v3（周六晚上 9 点截至）

修改：

1. 添加触发器部分的说明
2. 添加硬性规定：文件修改线程与文件扫描线程之间不能存在通信，即只能通过扫描硬盘来获取文件的变动信息。

一、 前言

IFTTT 是互联网的一种应用形态，它支持以 IF X THEN Y 的方式来定义任务，并能够在后台自动执行任务，比如：

IF {news.163.com has new message} THEN {drag the message to my blog}

关于 IFTTT：（建议大家实验前先了解一下基本思想）

IFTTT_度娘百科

二、 输入

命令行方式或控制台输入任务。

输入可为一行或多行。但全部输入结束后，监控线程才开始启动。监控中，不可添加新的任务，即非并发输入。

自行设计触发器和任务的输入格式，务必细致和准确。

触发器中指明监视的目录/文件。

基本的输入格式为：

IF 监视的目录/文件 触发器 THEN 任务

其中: 监视的目录/文件 为测试者自定义；触发器 为下文触发器部分(第五节)中给出的选项；任务 为下文任务部分(第六节)中给出的选项。

可能的任务格式举例：

IF /root modified THEN record-summary

IF [D:\test\demo.cpp] renamed THEN recover

触发器和任务的选择范围包括下文中给出的全部选项，每个输入为其中之一。

程序针对输入的不同的监视的目录/文件，创建不同的监视线程，使用者可监控的目录/文件不小于 5 个，不大于 8 个（即最多同时支持监视 8 个工作区，工作区定义在触发器部分）。

对于同一个监视的目录/文件，可有多个触发器和任务组合的监控工作。

如果监控的是目录，则监控目录下的所有文件，包括子目录（无论递归多少层）。

如果监控的工作区根目录不存在/非法，则输出错误提示信息即可。

目录或文件名的非法判定都以测试者的系统为准。

三、 输出

最终由 summary 和 detail 输出到特定文件，但建议控制台即时输出相关信息。

四、 硬性规定

针对 PC 文件系统，要求支持如下任务：

1. 要求使用线程安全设计，设计线程安全的文件访问类和其他有可能被共享的类，使用多线程进行检测和处理，主要线程安全问题可能出现在 summary 和 detail。
2. 使用 FILE 类，实现文件的操作。
3. 通过判断的依据有文件名、最后修改时间、文件大小和路径来识别一个文件。
4. 路径使用绝对路径。对于不同系统区不区分大小写问题，可以说，测试者使用被测试者的提供的文件读写类，在 WINDOWS 下创建 D:\test\和 D:\TEST\将是同一个，而 Linux 等区分大小写的系统将是两个，这里设计者只需使用 java 提供的文件接口，并且对接口返回的 Exception 进行处理，一般情况下无需担心大小写问题。
5. 不同的监视线程之间相互独立。
6. 对于监视的主目录不允许有修改（删除，移动，重命名等导致工作区消失）的操作，对工作区的子目录，以上修改是允许的。
7. 为了能够正常访问中文目录，此次作业要求统一将编码格式改为 UTF-8 编码。各 IDE 的修改步骤见本文末尾。
8. 文件修改线程与文件扫描线程之间不能存在通信，即只能通过扫描来获取文件的变动信息。
9. 不能使用 JDK 提供的一些涉及文件监控的高级玩意，如 watchserve,md5 等，只能使用 JDK 提供的基本文件操作类，然后自己动手设计实现相应的线程安全类和线程；否则算做功能性缺失。
10. 工作区不允许被执行消失（被删除）等操作。即工作区始终在。

五、 触发器

需要在程序运行之初设定一个程序监控范围，即输入一个监视目录名称作为工作区，在工作区外的目录文件不进行监控；若输入的是文件，则仅监控文件，且工作区（最大监控范围）为该文件的父文件夹。

输入的触发器选项可能下列指令:

1. "renamed" 重命名触发器（仅对文件）

文件名称变化可简化定义为在同一个目录（非整个工作区）下，新增了一个文件，缺失了一个文件，新增的文件跟缺失的原来文件的最后修改时间相同，文

件大小一样，但名称不一样的文件，即可定义为文件名称变化。例子：/root/a.txt => /root/b.txt .特殊情况，缺失了文件但没有新增文件或新增了文件没有缺失文件，则不予理会；若新增了 x 个文件，缺失了 y 个文件且全部文件大小、最后修改时间均相同，仅名称不一样，则任意缺失到新增的映射均认为正确。如缺失/root/a.txt 和 /root/b.txt 新增 /root/c.txt 和 /root/d.txt 则 rename /root/a.txt 和 /root/b.txt 均映射到 /root/c.txt 视为正确。

2. "modified" 修改时间变化触发器

修改时间变化可简化定义为，扫描前后存在一个路径和名称相同的文件或者文件夹，最后一次修改时间扫描前后不一样。

3. "path-changed" 路径变化触发器（仅对文件）

不论监视对象为目录还是文件，文件访问路径变化可定义为：在工作区内，在不同路径下，新增一个和原来文件名字相同，大小一样的，修改时间相同的文件且原来的文件消失（类似重命名触发器）。

4. " size-changed " 文件规模变化可定义为

若监视对象为目录，工作区目录下（递归定义）的文件/文件夹新增（0->x bytes）、删除(x->0 byte)（修改文件名或路径视为一删一增）以及修改内容产生文件大小变化(x->y bytes)；若监视对象为文件，仅对该文件的删除和新增以及修改进行检测，对于该文件的新增即出现了一个和刚删除的文件路径、名字一样的文件。

注意：

- 若监视对象为目录，所有检测均需要支持子目录递归检测，即针对该目录下的所有文件及文件夹。
- 若监视对象为文件，当文件重命名或路径移动，应该继续监视该重命名或路径移动后的文件。
- 限定路径移动和重命名仅针对文件对象而非文件夹对象检测。
- 工作区：若监控对象是目录，则目录自身和递归定义的下层目录和文件组成工作区；若监控对象是文件，则文件的父目录和其和递归定义的下层目录和文件组成工作区。
- 监控范围：若监控对象是目录，则目录自身和递归定义的下层目录和文件组成监控范围，也就是工作区；若监控对象是文件，则监控范围就是文件自身。
- 所有监控范围内的文件和目录都应该纳入触发检测，超出工作区的一切操作不予理会。
- 文件大小定义为文件字节数，目录大小定义为直接下层文件（不递归）的大小总和。

六、 任务

输入的任务选项可以包括如下指令：

1. " record-summary " 记录 summary

构造一个 summary 记录对象，其中保存对应的各类触发器触发次数信息，每当触发器触发，触发器操作 summary 对象新增并保存次数信息；summary 对象

每隔一段时间保存信息至特定文件。

2. "record-detail" 记录 detail

构造一个 detail 记录对象，其中保存对应的各类触发器触发的详细信息：文件规模的前后变化、文件重命名的前后变化、文件路径的前后变化、文件修改时间的前后变化；每当触发器触发，触发器操作的 detail 对象新增并保存详细变化信息；detail 对象每隔一段时间保存信息至特定文件。

3. "recover" 恢复文件路径（仅对文件）

当对应的触发器为重命名或路径改变触发器触发可以使用；其他触发器不能够使用，若使用则无视该任务。效果为将重命名恢复原状以及将路径改变恢复原状。

注意：

若监控范围内的一个目录或者文件大小发生变化，就触发一次 size-changed，包括新增删除以及修改大小

若监控范围内的一个目录或者文件最后修改时间发生变化，就触发一次 modified，前后文件都存在

若监控范围内一个文件的名称发生变化但仍在同一个目录内，则触发一次 renamed

若监控范围内一个文件的路径发生变化但名称没有变化，则触发一次 path-changed

七、 测试

测试者可以使用以下方法修改工作区的目录和文件：

1. 使用编程者提供的线程安全类中的读写方法实现文件修改。

测试者使用编程者提供的线程安全类构造一个测试线程，模拟用户在资源管理器对文件的修改（与 IF 触发器匹配，仅需实现功能，界面忽略）--测试线程没有义务采用任何同步控制措施，由此导致的程序崩溃均为被测程序问题。

测试者在被测程序 main 方法的合适位置创建和启动测试线程。

测试者检查被测程序是否能够按照触发器正确检测到相应的变化，并按照任务要求进行处理。

测试者更新文件的操作频率要小于被测试者给定的频率，同时每个工作区的总目录和文件数不要大于 1000 个。

被测试者提供的文件读写线程安全类应该具备文件信息（名称、大小、修改时间）读取功能，同时可以重命名文件（对应重命名触发器）、移动文件以改变文件路径（对应路径改变触发器）、新增及删除文件和文件夹的功能和文件写入功能（用于造成文件大小和修改时间不同，对应规模变化触发器和修改时间触发器）。具体使用方法由被测试者在 readme 中说明。提供的文件读写类与扫描线程互相独立。

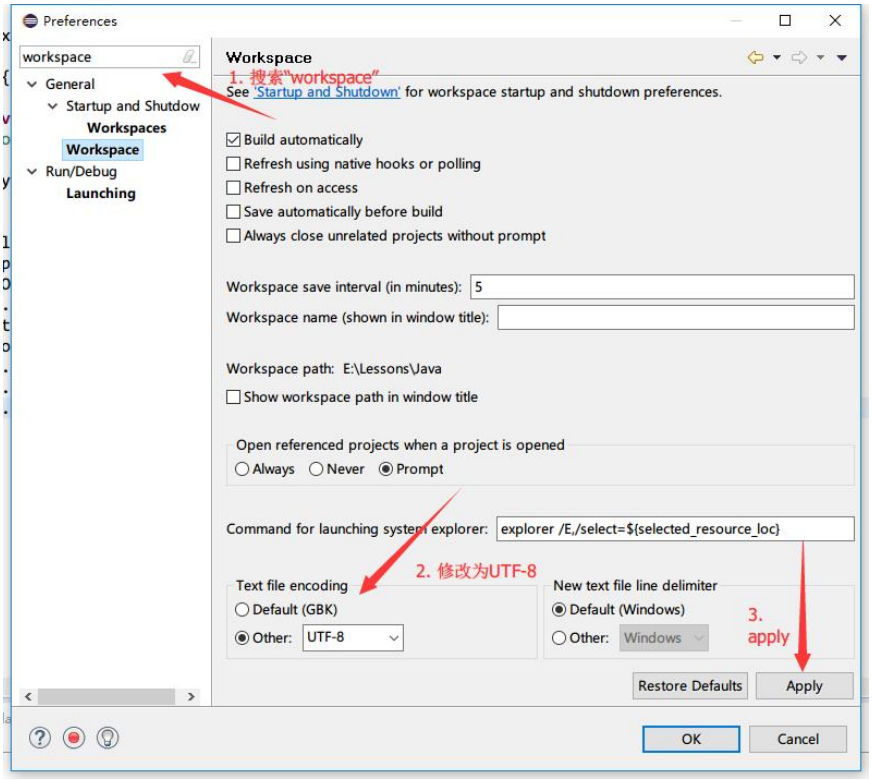
被测试者能够提供扫描的频率，写在 readme 中。

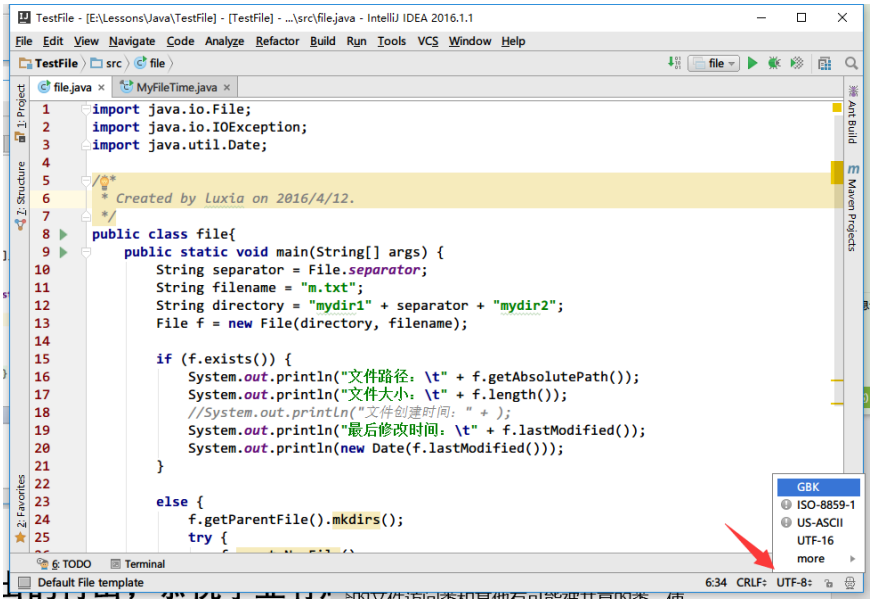
无论测试者和被测试者，如果出现乱码问题，请自行研究解决。

（老师刚刚发话，不允许资源管理器修改）

八、 附：Eclipse 和 IDEA 修改编码格式的方法

一图流，上为 Eclipse，下为 IDEA





拿命令行编译的同学，javac 可以制定代码的编码格式，具体用法为：

javac –encoding utf8 demo.java

其中，demo.java 是需要被编译的 java 文件，并且此时 javac 会以 utf8 格式读取并编译 demo.java 文件。

由于时间仓促，此文档可能尚有疏漏。另如有其他问题，鼓励大家在讨论区中提问。你的问题可能也是大家的问题，老师们和课代表也好统一解决，谢谢大家。

2016/4/12