# OO 第六次作业指导书

2018 版

## 1. 前言

- **作业目标**

本次作业的目标是训练同学们针对线程安全问题，如何平衡线程访问控制和共享对象之间的矛盾。

作业内容是实现一个监控程序，针对给定监控范围内的监控对象，以扫描方式探查监控对象相关属性的变化，从而触发规定的处理动作。所谓监控范围指计算机文件系统中的一棵目录树，监控对象则是位于监控范围内的具体文件（注：不包含目录文件）。在测试过程中，监控范围可以通过创建子目录来扩展，但不可以改变目录树中已经存在的任何目录名称。

- **作业背景知识**

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}*

其中 IF 和 THEN 为关键词，IF 后面的部分为触发器，THEN 后面部分为任务。为了避免用词混淆，称 IF***THEN***为一个**监控作业(monitoring job)**。

关于 IFTTT：（建议大家实验前先了解一下基本思想）

IFTTT_百科：http://baike.baidu.com/item/ifttt?fr=aladdin

## 2. 输入

通过**控制台**启动主程序后，输入形如“IF***THEN***”的监控作业。输入可为**一个或多个监控作业**，监控作业之间使用回车换行分隔。一次性输入全部监控作业后，监控线程开始启动。在监控过程中，不可添加新的监控作业。

监控作业的基本格式为：*IF 监视对象 触发器 THEN 任务*

触发器是一个逻辑条件，触发器用来检查监控对象的属性变化，一旦发生所关注的变化，则成功匹配触发器条件，从而“触发”THEN 后面的任务，即执行相应的任务。其中: *监视对象*由**测试者**设定，建议选择非系统盘的文件作为监控对象，以免造成系统破坏。***触发器***内容为下文触发器部分(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。触发器和任务的输入格式可自定义，但需在 readme 中准确说明。

可能的任务格式举例：

*IF /root/a.txt modified THEN record-summary*

*//含义为：对/root/a.txt 文件进行监控，一旦该文件被修改，则执行 record-summary。*

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

*//含义为：对D:\test\demo.cpp 文件进行监控，一旦被重命名，则执行recover。*

触发器中指明监控对象的完整路径信息，相应的，监控范围自动为该监控对象的所在位置对应的目录树。**以** *IF [D:\test\demo.cpp] renamed THEN recover* *为例*，监控范围自动为 D:\test。测试中，测试者可以在 *D:\test*下创建子目录，形成 D:\test\test1。注：不允许在 D:\层次下创建任何子目录。

**触发器**和**任务**的选择范围包括下文中给出的全部选项，每个输入命令只能选择其中之一。

程序针对输入的每个不同监控对象，创建相应的监视线程。要求同时支持的监控对象不多于 10 个。

对于同一个监控对象，可有设置多个**触发器**和**任务**组合。例如，对于输入的两个命令“IF x tr1 THEN ts1; IF x tr2 THEN ts2”，x 表示相同的监控对象，而 tr1 可以和 tr2 不同，ts1 可以和 ts2 不同。如果 tr1 和 tr2 相同，且 ts1 和 ts2 相同，则视为相同命令，实际只处理其中一个，无需反馈提示。

如果监控对象不存在，则输出错误提示信息即可。

## 3. 输出

根据第六节任务部分的要求产生相应输出。结果 summary 和 detail 输出到特定文件，但建议控制台即时输出相关信息。

## 4. 硬性规定

针对桌面操作系统的文件系统，要求如下：

1）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。提示：主要线程安全问题可能出现在 record-summary 和 record-detail 类型的任务中。

2) 要求文件访问类（线程安全类）提供文件修改（含创建文件、修改文件属性和删除文件）方法供测试使用

3) 使用 Java.io.File 类来实现文件的访问和相关操作(提示：Java.io.File is not thread safe!!!)。

4) 依据文件路径和文件全名来唯一识别一个监控对象（即文件），文件属性包括最后修改时间和文件大小。

5) 文件路径一律使用完整的**绝对路径**来表示。

6) 每个触发器对应一个独立的监视线程。

7) **不允许被测试程序在测试中打开被监控对象，此操作会阻塞测试动作的实施，一旦测试者查出被测程序代码中有这样的作弊行为，可直接举报无效作业。**

8) **只能通过测试线程来实施测试动作**，即在程序运行中不允许通过操作系统或文件管理系统对监控目录内目录和文件进行任何修改。

9) 为了能够正常访问包含中文字符的目录/文件，此次作业要求统一将编码格式设置为 UTF-8 编码。IDE 中的修改步骤见指导书末尾提示。

## 5. 触发器

要求支持的触发器包括:

- **“renamed”文件重命名触发器**

文件名称变化可定义为：在监控对象所在的同一层目录下，无法再找到监控对象文件，同时新增了一个文件，且新增文件跟缺失的监控对象有相同的最后修改时间和文件大小。

若在监控对象同一层次下新增了多个文件，且监控对象文件缺失了，且新增的所有文件都与监控对象有相同的文件大小和最后修改时间，则其中任意一个新增文件都可以认为是监控对象的重命名结果。举例如下：监控对象为//root/a.txt，经扫面发现/root/a.txt 不见了，但新增了 /root/c.txt 和 /root/d.txt，且这两个文件与/root/a.txt 有相同的规模和最后修改时间。此时，认为/root/a.txt 改名为/root/c.txt 或 /root/d.txt 皆正确。

- **“Modified”文件最后修改时间变化触发器**

文件修改时间变化的含义是，监控对象在两次扫描中被发现最后修改时间不同（不理会两个时间的先后关系）。

- **“path-changed”路径变化触发器**

文件路径变化可定义为：在监控范围内，监控对象的路径发生了变化。举例来说，设监控对象为 D:\test\a.txt，则对应的监控范围为 D:\test。若监控对象发生了路径变化，则意味着在监控范围内新增一个和原来文件名字相同、规模相同、最后修改时间相同、但目录层次不同的文件，且原来的文件消失。如果发现监控对象文件消失了，发现有新增的多个文件满足条件：文件名称相同、规模相同、最后修改时间相同，则随意指定其中一个为原来的文件皆正确。举例来说，D:\test\下有一个子目录 dir，且 D:\test\dir 下本来没有 a.txt 文件，文件 D:\test\a.txt 在 D:\test 下不见了，但出现在了 D:\test\dir 中，而且名称、文件大小和最后修改时间均相同，此时可判定 D:\test\a.txt 的路径属性发生了改变，由 D:\test 变为 D:\test\dir。

- **“size-changed”文件规模变化触发器**

针对给定的监视对象文件，只有当该文件在相应目录层次下仍然存在，且文件大小和最后修改时间均发生变化，则可触发 size-changed 触发器。

**注意：**

1. 若监视对象发生了文件重命名或路径移动后，要求程序可以继续按照规定的监控作业来监视该重命名或路径移动后的文件。
2. 限定路径移动和重命名仅针对普通文件而非文件夹。

## 6. 监控任务

支持的任务包括：

- **" *record-summary* " 记录 summary**

构造一个 summary 记录对象，保存相应触发器的触发次数信息。每当触发器触发，触发器向 summary 对象登记信息，后者按触发器类别统计触发次数；summary 对象每隔一段时间（自行设定）保存信息至监控范围之外的某个特定文件（具体位置应在 readme 中交代说明）。

- **"*record-detail*" 记录 detail**

构造一个 detail 记录对象，保存相应触发器触发时的相关详细信息，包括：

- 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，触发器向 detail 对象登记信息，由后者按照触发器类别记录整理；detail 对象每隔一段时间（自行设定）保存信息至监控范围之外的某个特定文件（具体位置应在 readme 中交代说明）。

- **"*recover*" 恢复文件路径**

仅可与当前作业中的重命名或路径改变触发器配合使用，其他情况应报错并忽略相应命令。效果为将重命名恢复原状以及将路径改变恢复原状。

## 7. 测试

测试者可借鉴提供的测试线程样例代码，使用被测试者提供的线程安全类构造测试线程，模拟用户对文件的修改，从而实施预期的测试。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，即 crash 类型的 bug。

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，除此以外不可以对被测程序的任何代码做实质性改变。

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。

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、移动文件以改变文件路径、新增及删除文件和文件夹的功能和文件写入功能。具体使用方法由被测试者在 readme 中说明。注：不允许被测试者在所提供的文件读写线程安全类中进行方法调用的监视，即获得测试者调用相应方法的参数。此行为视为作弊，一旦发现可被举报为无效。

对测试线程的说明和要求：

（1） 一个测试用例可以包括一个或多个对监控范围内的文件进行修改的动作，注意可以创建子目录，但是不允许对监控范围中已经存在的任何目录名进行修改。

（2） 在一个测试用例中，不允许对同一个文件实施两次或两次以上的变化或修改。如，不能对一个文件做两次文件改名动作。举例：对文件 D:\test\a.txt，不可在一个测试用例中首先更改文件名为 D:\test\a1.txt，再更改文件名为 D:\test\a2.txt。

（3） 测试者可以编写多个测试用例，但是在一次测试（即启动被测程序执行）中只允许执行一个测试用例。

指导书末尾提供了一个可参考的测试线程

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。

## 8. 设计参考和补充说明

对于同一个修改，若满足多个触发器触发规则，则这些触发器要有响应。

recover 造成的文件移动或重命名不会再次触发当前线程的 renamed 或者 path-changed。

不关注文件具体内容，仅关注文件的属性：文件名称、文件所在完整路径、文件大小、文件最后修改时间。

建议借鉴 ppt 中的 safe wrapper 方法，基于 Java 类库的 File 类来构造线程安全的 SafeFile 类。

建议为监视范围和监控对象建立 snapshot，且使用线程安全类来管理 snapshot。专门设计线程去扫描文件夹或具体文件，建立 snapshot，同时设计线程去做 snapshot 对比分析。

注意锁的使用方式。注意通过 sleep，或者共享对象的 wait、notify、notifyAll 等方式让每个线程都有机会获得调度执行机会，否则可能会引起程序执行过程中发生难以预料的行为。

summary 和 detail 对象应使用线程安全的类来管理。

## 9. 测试线程样例代码

测试线程例子：

```
public class TestThread extends Thread{
    public boolean addFile(String file){
        …//使用被测者提供的 SafeFile 类来实现增加一个文件，如果成功则返回 true；否则返回 false（如参数有误，或者遇到了任何文件访问异常）。
```

```
    }
    public boolean rename(String from, String to){
        …//使用被测者提供的 SafeFile 类来实现文件重命名，如果成功则返回 true；否则返回 false（如参数有误，或者遇到了任何文件访问异常）。
    }
    public boolean delete(String src){
        …//使用被测者提供的 SafeFile 类来删除一个文件，如果成功则返回 true；否则返回 false（如参数有误，或者遇到了任何文件访问异常）。
    }
    public boolean move(String from, String to){
        …//使用被测者提供的 SafeFile 类来移动一个文件，如果成功则返回 true；否则返回 false（如参数有误，或者遇到了任何文件访问异常）。
    }
    public boolean changeSize(String file){
        …//使用被测者提供的 SafeFile 类来改变一个文件的规模和最后修改时间，如果成功则返回 true；否则返回 false（如参数有误，或者遇到了任何文件访问异常）。
    }
    public boolean changeTime(String file){
        …//使用被测者提供的 SafeFile 类来改变一个文件的最后修改时间，如果成功则返回 true；否则返回 false（如参数有误，或者遇到了任何文件访问异常）。
    }
    public boolean testcase(){
/* 该测试为：在 D 盘 OO 文件夹下，新建文件 a.txt，并将 b1.txt 更名为 b2.txt */
        if(!add("D:\\OO\\a.txt"))return false;
        if(!rename("D:\\OO\\b1.txt", "D:\\OO\\b2.txt"))return false;
        return true;
    }
    public void run(){
        if(!testcase())…;//输出相关提示信息，或者进行额外的必要处理。
    }
```

}

## 10. 修改编码格式的方法

请同学们修改字符集编码，否则读取的文件名和路径名等可能显示为乱码。这里仅列出常用的 Eclipse 和 IDEA。使用其他 IDE 或编辑工具的建议在搜索引擎中寻找解决方案。

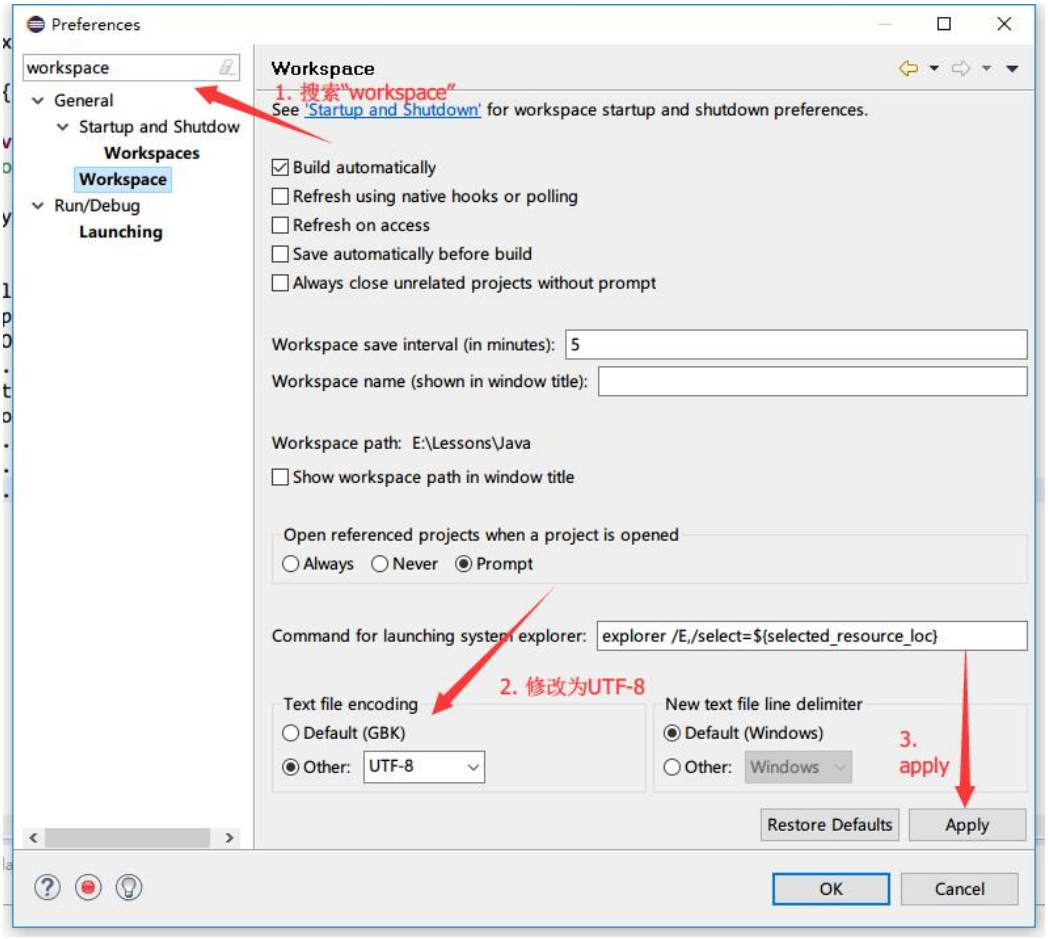


图 1：Eclipse IDE 中的字符集设定

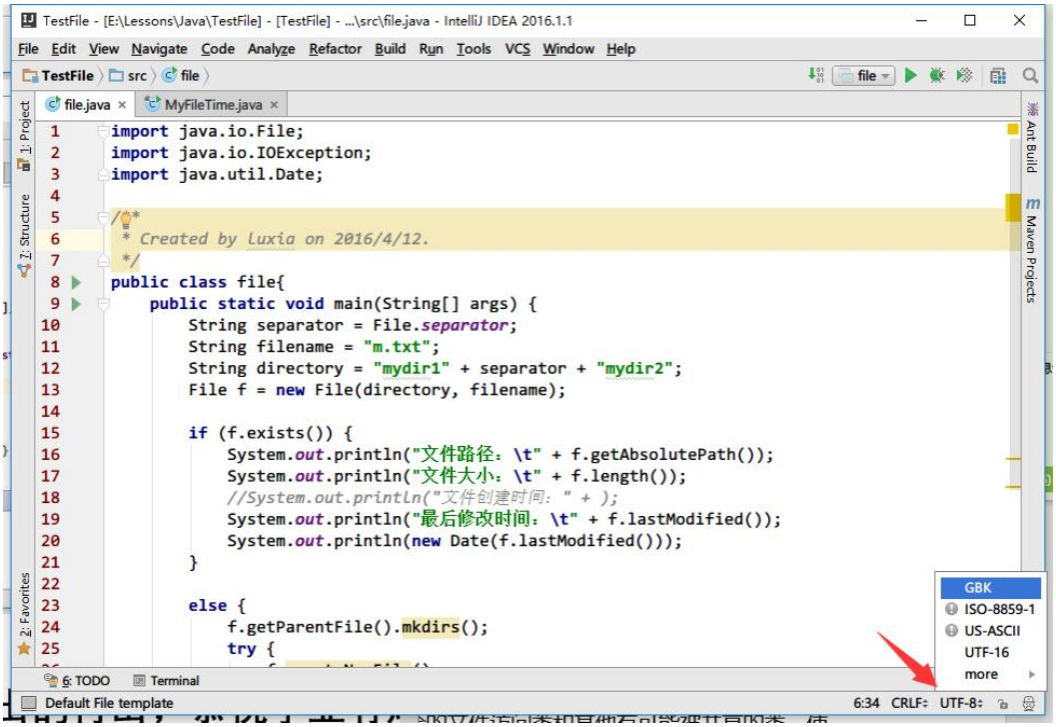


图 2： IDEA 中的字符集设定