OO第十四次作业说明文档

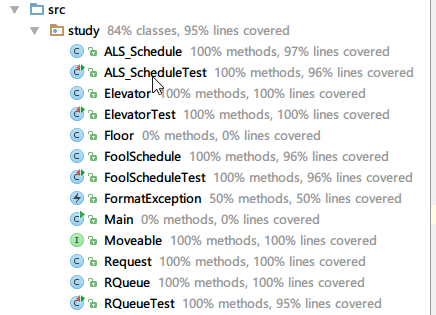
这次作业主要是对于上次作业中的注释进行了添加，并撰写了类实现正确性推理文档。代码改动并不多。但是还请测试者注意一下几点：

1. 上次作业并没有被测试出Bug。如果需要复现上次作业的测试结果，请建立工程并将junit和hamcrest包导入工程，这两个包我已经附加在文件夹中了。
2. 这次作业的任务核心是正确性推理，因此请不要在回溯性测试上大费周折。
3. 下附上次作业的说明文档，祝期末顺利~

OO第十三次作业——ALS电梯测试说明文档

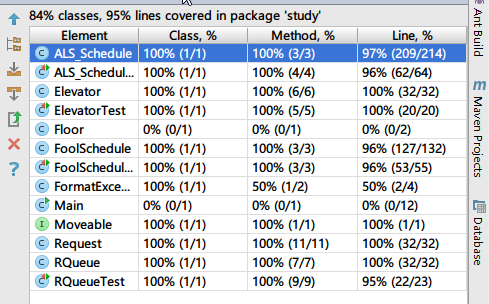
本次作业主要通过Junit4对第三次作业中的电梯类，调度类和请求队列类进行了测试，对这四个类的类规格和过程规格进行了补充。

先放文件目录：



此次测试仅针对ALS\_Schedule，Fool\_Schedule, Elevator和RQueue这四个类进行，分别使用三个测试单元构建测试用例来对应每个类，以使其覆盖率达到100%。

覆盖效果如图：



关于更详细的说明也可以参见软件生成的覆盖率报告，中文乱码啥的也请见谅ORZ

如要测试不变式，请手动将四个类中的RepOK方法取消注释。FoolSchedule和ALS\_Schedule有5行没有覆盖上，其中有1行是抛出异常，有4行应该已经执行了但是没有统计上，个人认为是IDEA软件的问题，还请同学相互谅解……

如果你使用的是Eclipse，进行测试之前请手动引入Junit4包，该包已附加在提交目录中。因为本人没有使用Eclipse测试过覆盖率，如果测试结果与报告中有差异，还请少侠手下留情。



下附没有卵用的第三次作业说明：

# 输入规范

输入应首先遵循基本法~

用户输入应为一系列按照请求产生时间排序的请求序列。

每个请求应独占一行，但是请求的总个数不应超过100000个，否则程序将会提示输入错误。

楼层请求的输入格式为：(FR, n, UP/DOWN, t)。

对于楼层请求来说，

如果括号中出现了除{'F','R','U','P','D','O','W','N','0'~'9', ',' }以外的任何字符，该指令视为无效指令。

当n=1且方向为DOWN或者n=10且方向为UP时，该指令视为无效指令。

当n不是1~10之间的整数时，该指令视为无效指令

当t不是非负整数或t大于100000000时，该指令视为无效指令

电梯请求的输入格式为：(ER, n, t)

对于电梯请求来说：

如果括号中出现了除{'E','R','0'~'9', ','}以外的任何字符，改指令视为无效指令。

当n不是1~10之间的整数时，该指令视为无效指令

当t不是非负整数或t大于100000000时，该指令视为无效指令

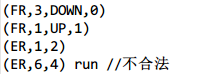
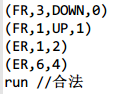
括号中可以过滤空格，但是每条请求都请在同一行中完成输入，如下图：



输入完成后，另起一行

并输入run结束输入，

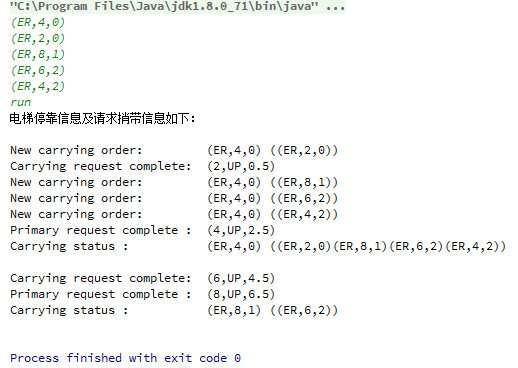
run之间请不要插入空格



因此，在第一行输入run并不会使程序结束。必须另起一行在第二行输入run才能使程序开始运行。

# 输入方式

本程序采取控制台(Console)输入，控制台输出。如下图：



# 实现方法

鉴于每个人实现的方法都不尽相同，我觉得还是有必要在这里将我程序里面的细节实现简要地概括一下，以方便您对于下一段输出格式的理解。

开始运行时我的程序接受输入，并创建一个请求队列（rQueue）用以保存所有输入的请求，这一点和第二次作业是完全相同的。在进行调度时，调度器几乎包揽了所有工作。本程序调度使用仿真的方式，即通过for循环模拟没0.5s时间段内发生的事。在调度的过程中会创建follow 队列用来保存主请求（Primary）和其捎带请求（carry）。当follow队列为空时，自动添加请求队列(rQueue)中最靠前的未实现请求为Primary request；当follow队列非空时，添加符合主请求运动条件的请求入队。Follow队列的第一个元素永远是主请求。捎带请求的判定函数见作业要求6. b) & c)

# 输出格式

1. **基本格式**

电梯运动状态在每条请求完成时输出，格式为(n, UP/DOWN/STAY, t)。

此处的t为电梯完成指令后（到达指定楼层），未包括开关门的时间。

方向表明自上一个请求运动到现在的请求电梯的运动过程。

如果有新请求加入follow队列成为捎带请求，那么将会输出：

New carrying order: 主请求 (新加入请求)

如果完成的指令为主请求的捎带请求，则输出：

carrying request complete: 电梯此时的运动状态

如果完成的指令为主请求，则输出：

Primary request complete：电梯此时的运动状态

在主请求完成之后，将会输出请求之间的主从关系，输出格式和作业要求相仿，为：

Carrying status : Primary request (carrying requests)

需要说明的是，如果一个主请求的执行过程中附带了多个捎带请求，那么这些请求将会根据加入follow队列的先后顺序同时出现在括号中。如下图：



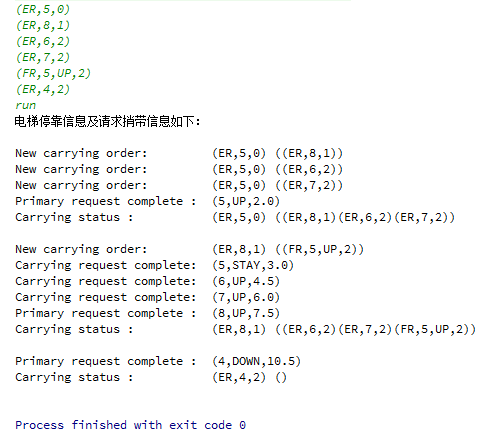
如果主请求完成时，还有捎带请求尚未完成，那么捎带队列中最靠前（最先加入）的捎带请求将会被设置为主请求。一次停靠可以满足所有“在开关门前接收并到达目标楼层”的请求，也就是说，同层的请求都将会在到达时完成。如上图，到达2层时将会完成捎带请求(ER,2,0)，到达4层后将会完成主请求(ER,4,0)和捎带请求(ER,4,2)。此时主请求完成，最先加入的未完成请求为(ER,8,1)，因此在下一段显示的是以(ER,8,1)为主请求执行的电梯的运动状况。

主请求即便没有捎带指令也会在捎带指令的位置打印”()”



如果主请求完成时，其所有捎带请求也都已经被完成了，则选择原始请求队列(rQueue)中最早输入且未完成的请求设为主请求。

再次举例具体说明：



这个例子包含了上述所说的所有规则：

首先电梯以输入的第一条合法输入为主指令，通过判定函数先后将(ER,8,1)、(ER,6,2)、(ER,7,2)加入follow的请求队列等待执行。

在2s时，电梯到达5楼，输出到站信息和捎带信息。此时虽然来了(FR,5,UP,2)这条指令，但是由于是在开关门期间输入的，因此由作业要求6 a)的规定并不加入follow队列算作已响应。此时主指令完成，最先加入的未完成指令(ER,8,1)成为主旨令。

在开关门完成后（3s），(FR,5,UP,2)被判定为捎带并首先完成。实际生活中，在电梯开关门时按楼层按钮，门也是会再打开一次，所以这个设定是符合现实的。

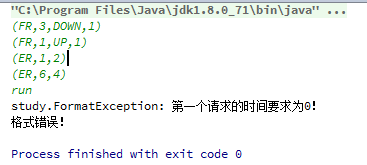
之后依次完成捎带指令，并在主旨令完成后输出捎带信息。

最后的(ER,4,2)虽然是主旨令，但是由于没有捎带请求，因此后面加空括号()

至此电梯调度结束。

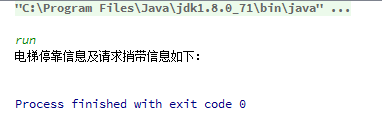
1. 异常时间序列输入

如果输入的序列不满足时间要求，即第一个请求时间不为0，或者输入了未经排序的时间序列，那么程序将中断并抛出错误原因。



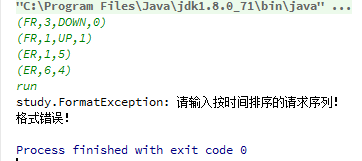
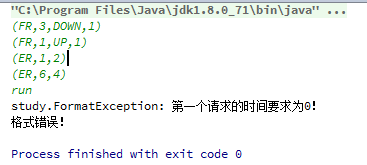
1. 空请求输入

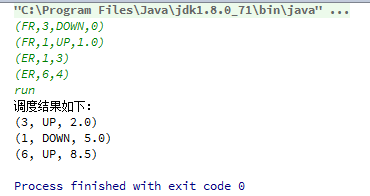
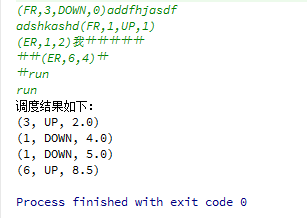
如果输入的请求全部不合法，或者未输入任何请求即输出run，那么将不会输出任何状态信息。



# 错误处理原则

1. 如果输入的序列不满足时间要求，即第一个请求时间不为0，或者输入了未经排序的时间序列，那么程序将中断并抛出错误原因。



1. 当请求中出现了非法字符，该请求被自动忽略。  
   
2. 当请求外出现非法字符，尽量采取容错以增强鲁棒性。  
     
   需要注意的是，输入’(’后就不要输入非法字符啦，否则将会判定为无效请求！

# 其他说明

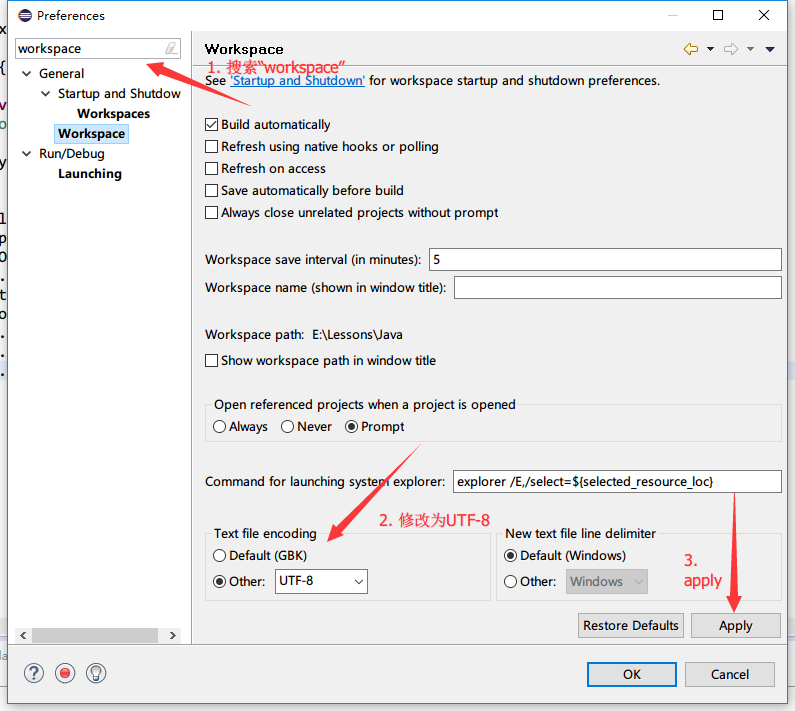
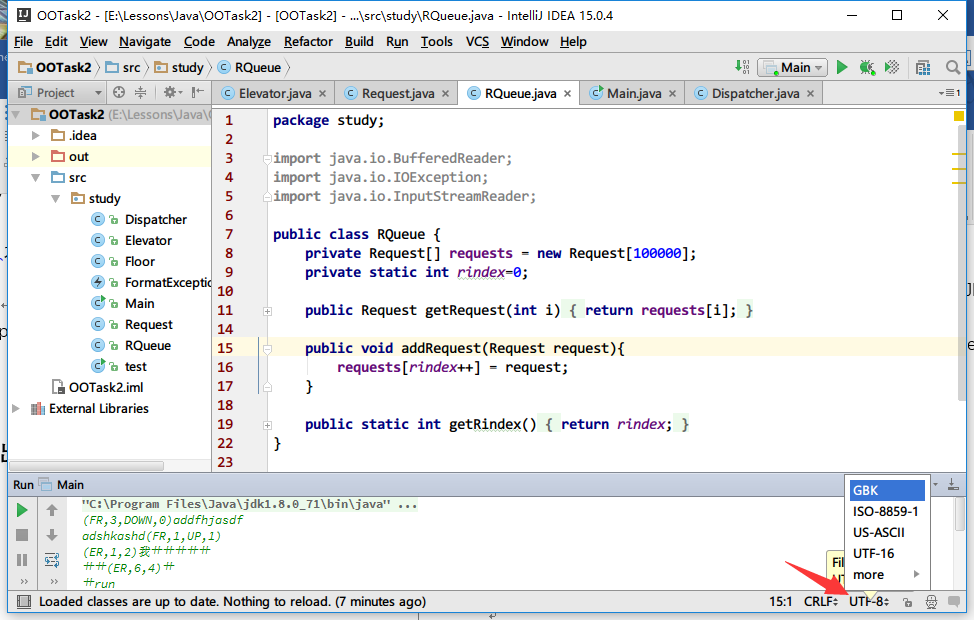
本程序的编码格式为UTF-8格式，所用IDE为Intellij IDEA，JDK版本为1.8。

IDEA导入程序只需将src整个文件夹拖入工程即可，

Eclipse导入工程需要新建一个名为study的包，并import所有.java文件即可。

如果你的JDK版本低于1.8，建议更新JAVA版本。

遇到中文乱码的情况，请调整编码设置~如下图，上为eclipse，下为IDEA

**最后、感谢你对我的程序所做出的付出，恭祝学业有成！**