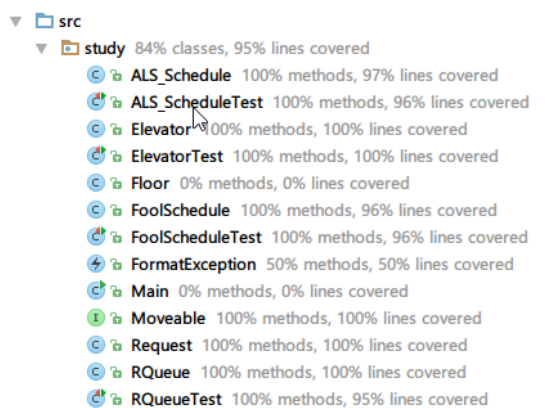


## OO 第十三次作业——ALS 电梯测试说明文档

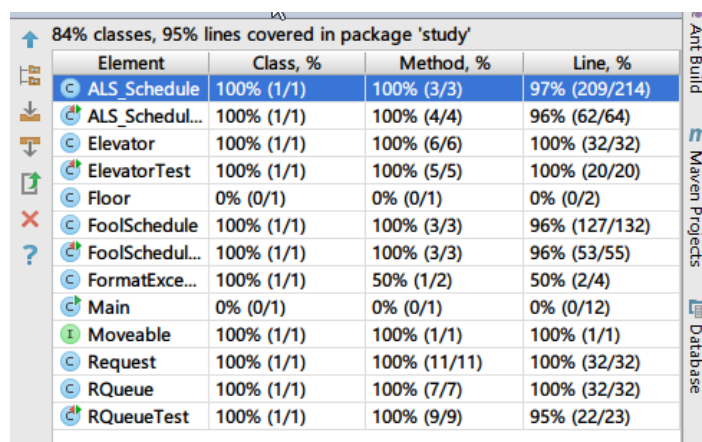
本次作业主要通过 Junit4 对第三次作业中的电梯类，调度类和请求队列类进行了测试，对这四个类的类规格和过程规格进行了补充。

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：



Element	Class, %	Method, %	Line, %
ALS_Schedule	100% (1/1)	100% (3/3)	97% (209/214)
ALS_ScheduleTest	100% (1/1)	100% (4/4)	96% (62/64)
Elevator	100% (1/1)	100% (6/6)	100% (32/32)
ElevatorTest	100% (1/1)	100% (5/5)	100% (20/20)
Floor	0% (0/1)	0% (0/1)	0% (0/2)
FoolSchedule	100% (1/1)	100% (3/3)	96% (127/132)
FoolScheduleTest	100% (1/1)	100% (3/3)	96% (53/55)
FormatException	100% (1/1)	50% (1/2)	50% (2/4)
Main	0% (0/1)	0% (0/1)	0% (0/12)
Moveable	100% (1/1)	100% (1/1)	100% (1/1)
Request	100% (1/1)	100% (11/11)	100% (32/32)
RQueue	100% (1/1)	100% (7/7)	100% (32/32)
RQueueTest	100% (1/1)	100% (9/9)	95% (22/23)

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：

## 一、 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：

```
( E R , 1 0 , 0 ) //合法
(FR,3,DO         //不合法
 |               WN,0)
```

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

```
(FR,3,DOWN,0)
(FR,1,UP,1)   (FR,3,DOWN,0)
(ER,1,2)      (FR,1,UP,1)
(ER,6,4)      (ER,1,2)
run //合法    (ER,6,4) run //不合法
```

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。

## 二、 输入方式

本程序采取控制台(Console)输入，控制台输出。如下图：

```
"C:\Program Files\Java\jdk1.8.0_71\bin\java" ...
```

```
(ER,4,0)
```

```
(ER,2,0)
```

```
(ER,8,1)
```

```
(ER,6,2)
```

```
(ER,4,2)
```

```
run
```

电梯停靠信息及请求捎带信息如下：

```
New carrying order:      (ER,4,0) ((ER,2,0))
```

```
Carrying request complete: (2,UP,0.5)
```

```
New carrying order:      (ER,4,0) ((ER,8,1))
```

```
New carrying order:      (ER,4,0) ((ER,6,2))
```

```
New carrying order:      (ER,4,0) ((ER,4,2))
```

```
Primary request complete : (4,UP,2.5)
```

```
Carrying status :        (ER,4,0) ((ER,2,0) (ER,8,1) (ER,6,2) (ER,4,2))
```

```
Carrying request complete: (6,UP,4.5)
```

```
Primary request complete : (8,UP,6.5)
```

```
Carrying status :        (ER,8,1) ((ER,6,2))
```

```
Process finished with exit code 0
```

### 三、 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)

### 四、 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：

```
New carrying order:      (ER,4,0) ((ER,4,2))
Primary request complete : (4,UP,2.5)
Carrying status :        (ER,4,0) ((ER,2,0) (ER,8,1) (ER,6,2) (ER,4,2))
```

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”()”

```
(ER,4,0)
run
电梯停靠信息及请求捎带信息如下：

Primary request complete : (4,UP,1.5)
Carrying status :         (ER,4,0) ()

Process finished with exit code 0
```

如果主请求完成时，其所有捎带请求也都被完成了，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：

```
(ER,5,0)
(ER,8,1)
(ER,6,2)
(ER,7,2)
(FR,5,UP,2)
(ER,4,2)
run
电梯停靠信息及请求捎带信息如下：

New carrying order:      (ER,5,0) ((ER,8,1))
New carrying order:      (ER,5,0) ((ER,6,2))
New carrying order:      (ER,5,0) ((ER,7,2))
Primary request complete : (5,UP,2.0)
Carrying status :        (ER,5,0) ((ER,8,1) (ER,6,2) (ER,7,2))

New carrying order:      (ER,8,1) ((FR,5,UP,2))
Carrying request complete: (5,STAY,3.0)
Carrying request complete: (6,UP,4.5)
Carrying request complete: (7,UP,6.0)
Primary request complete : (8,UP,7.5)
Carrying status :        (ER,8,1) ((ER,6,2) (ER,7,2) (FR,5,UP,2))

Primary request complete : (4,DOWN,10.5)
Carrying status :         (ER,4,2) ()

Process finished with exit code 0
```

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。

## 2. 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。

```
"C:\Program Files\Java\jdk1.8.0_71\bin\java" ...  
(FR,3,DOWN,1)  
(FR,1,UP,1)  
(ER,1,2)|  
(ER,6,4)  
run  
study.FormatException: 第一个请求的时间要求为0!  
格式错误!  
  
Process finished with exit code 0
```

## 3. 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。

```
"C:\Program Files\Java\jdk1.8.0_71\bin\java" ...  
  
run  
电梯停靠信息及请求捎带信息如下:  
  
Process finished with exit code 0
```

# 五、 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。

<pre>"C:\Program Files\Java\jdk1.8.0_71\bin\java" ... (FR,3,DOWN,1) (FR,1,UP,1) (ER,1,2)  (ER,6,4) run study.FormatException: 第一个请求的时间要求为0! 格式错误!  Process finished with exit code 0</pre>	<pre>"C:\Program Files\Java\jdk1.8.0_71\bin\java" ... (FR,3,DOWN,0) (FR,1,UP,1) (ER,1,5) (ER,6,4) run study.FormatException: 请输入按时间排序的请求序列! 格式错误!  Process finished with exit code 0</pre>
--	--

2. 当请求中出现了非法字符，该请求被自动忽略。

```
"C:\Program Files\Java\jdk1.8.0_71\bin\java" ...  
(FR,3,DOWN,0)  
(FR,1,UP,1.0)  
(ER,1,3)  
(ER,6,4)  
run  
调度结果如下:  
(3, UP, 2.0)  
(1, DOWN, 5.0)  
(6, UP, 8.5)  
  
Process finished with exit code 0
```

3. 当请求外出现非法字符，尽量采取容错以增强鲁棒性。

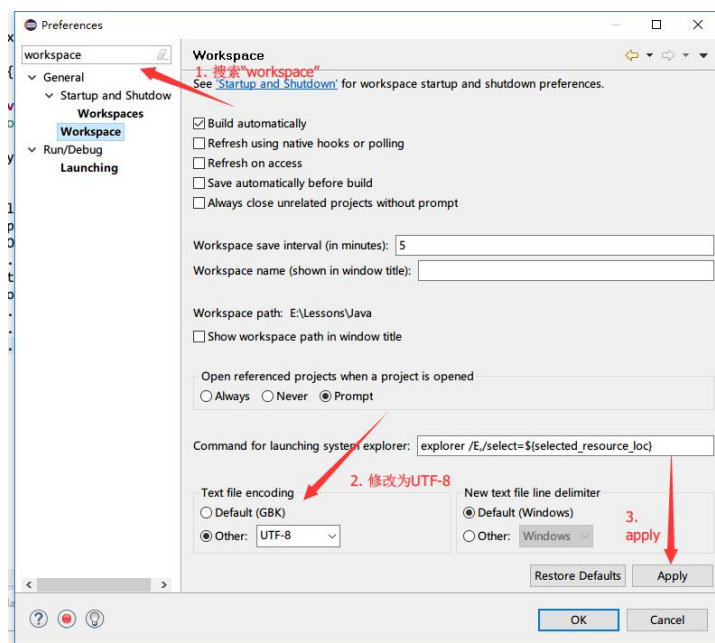
```
(FR,3,DOWN,0)adfhjasdf
adshkashd(FR,1,UP,1)
(ER,1,2)我#####
##(ER,6,4)##
##run
run
调度结果如下:
(3, UP, 2.0)
(1, DOWN, 4.0)
(1, DOWN, 5.0)
(6, UP, 8.5)

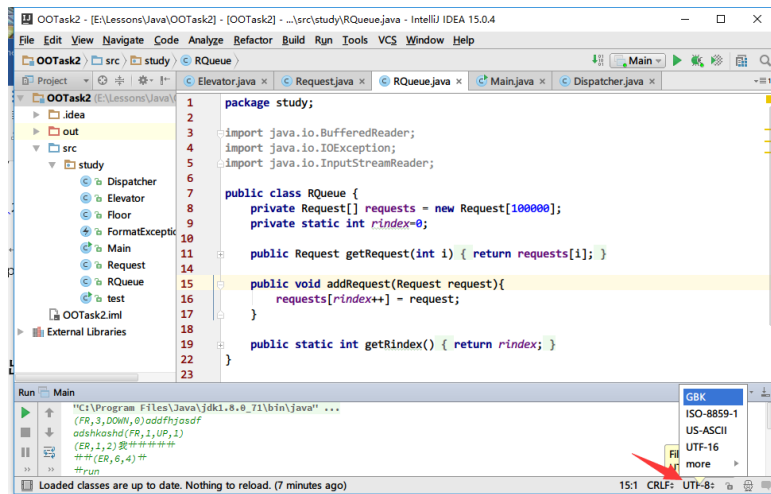
Process finished with exit code 0
```

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！

## 六、 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！