**ReadMe&测试覆盖率报告**

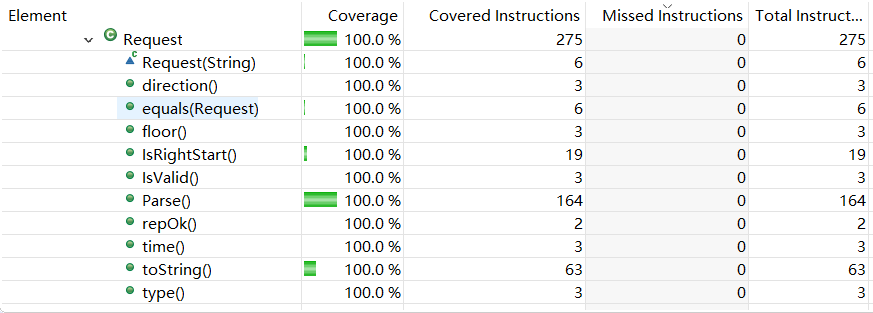
2018/6/11

1. **目录说明**
2. src文件夹下为源程序，包括添加类规格以后的源电梯程序。
3. tests文件夹下为单独分出来的测试程序，运行时加到源程序中。
4. Coverage为所有类（除了main）的单元测试ElcEmma代码覆盖率截图。
5. **测试程序说明**

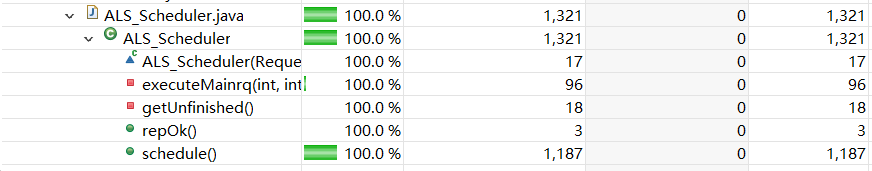
每个类的测试程序为xxxTest.java，其中main类不需测试。

***各类方法测试覆盖率如下，均达到100%：***

**Request类**



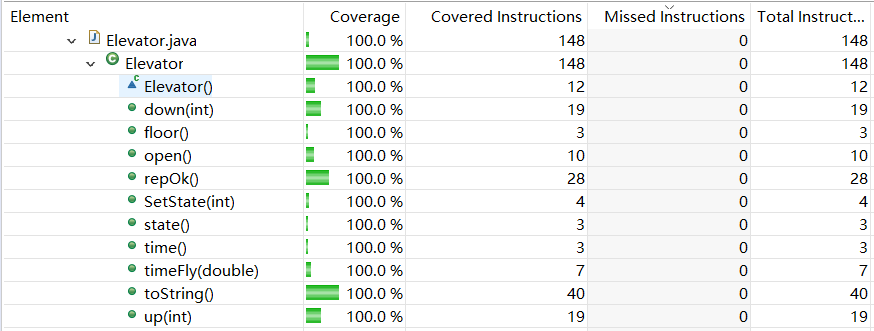
**ALS\_Scheduler类**



**Scheduler类**



**Elevator类**



**RequestQueue类**



1. **测试用例说明**

所有测试均通过，无BUG，因此**所复现的BUG数和新发现的BUG数为零**。覆盖率在上一点，均为100%。**测试用例数目看表格即可**。

电梯类测试：

|  |  |  |
| --- | --- | --- |
| 所测方法 | 测试用例 | 说明 |
| Elevator() | 创建初始的电梯类 | 初始电梯停留在1层，时间为0，状态为0 |
| up() | 上行2层 | 时间，状态，楼层号应与规则一致 |
| down() | 上行7层，下行2层 | 时间，状态，楼层号应与规则一致 |
| open() | 开关门一次 | 时间，状态，楼层号应与规则一致 |
| SetState() | 分别设置状态为0,1,2 | 对应状态一致 |
| TimeFly() | 时间分别流逝12.5和-10 | 获取对应时间一致 |
| time() | 时间流逝11.5 | 获取对应时间一致 |
| state() | 设置状态为0,1,2 | 对应状态一致 |
| floor() | 上行5层 | 对应楼层号一致 |
| toString() | 上行3层，开关门1次，下行1层 | 与预期电梯字符串表示一致 |
| repOk() | 设置电梯状态为0,1,2,3 | 前三状态有效，状态3无效为false |

请求类测试：

|  |  |  |
| --- | --- | --- |
| 所测方法 | 测试用例 | 说明 |
| Request() | 以字符串1122初始化 | repOk为true即可，因为还未解析 |
| Parse() | END==- | 长度不足8 |
| (FR,1,UP,2 | 缺括号 |
| (RR,1,UP,3) | 类型标识符错误 |
| (FR,1,UP,3,5) | 多项 |
| (FR,3,UPP,2) | 方向标识符错误 |
| (FR,3,UP,2) | 正确FR样例 |
| (FR,4,DOWN,5) | 正确FR样例 |
| (ER,4,3,5)" | 多项 |
| (ER,6,4) | 正确ER样例 |
| (ER,0,1) | 层数超下限 |
| (ER,15,1) | 层数超上限 |
| (FR,1,DOWN,1) | 一楼不能下 |
| (FR,10,UP,3) | 十楼不能上 |
| (ER,0,-1) | 时间不为负 |
| (ER,0,2147483649) | 时间超过INT最大值 |
| (ER,214748364956,5) | 楼层号超INT最大值 |
| IsValid() | (ER,4,3,5) | 无效 |
| (ER,7,3) | 有效 |
| type() | (ER,7,3) | ER请求 |
| (FR,7,UP,3) | FR请求 |
| direction() | (FR,7,UP,3) | UP请求 |
| (FR,7,DOWN,3) | DOWN请求 |
| floor() | (FR,7,UP,3) | 7层请求 |
| (ER,2,3) | 2层请求 |
| time() | (FR,7,UP,3) | 时间为3 |
| IsRightStart() | (FR,1,UP,0) | 是合法的第一条请求 |
| (FR,7,UP,3) | 不合法 |
| toString() | (FR,7,UP,3) | 其实就是把小括号变成中括号 |
| (FR,7,DOWN,3) | / |
| (ER,7,3) | / |
| equals() | (FR,1,UP,0)和(FR,1,UP,0) | 相等 |
| (FR,1,UP,0)和(FR,1,UP,2) | 不相等 |
| repOk() | (FR,1,UP,0) | 不论合不合法，repOk永为true |
| (FR,6,UP,10) | / |
| (ER,5,3) | / |

调度器测试：

|  |  |  |
| --- | --- | --- |
| 所测方法 | 测试用例 | 说明 |
| ALS\_Scheduler() | 空请求数组和初始电梯 | repOk()为true即可 |
| schedule() | (FR,1,UP,0)  (ER,4,0)  (ER,4,1)  (FR,2,UP,1)  (FR,2,UP,2)  (ER,7,3)  (ER,9,4)  (FR,3,DOWN,5) | 包含同质请求，向上的捎带，以及简单的下行请求，正确执行结果如下：  [FR,1,UP,0]/(1,STILL,1.0)  #SAME [(ER,4,1)]  #SAME [(FR,2,UP,2)]  [FR,2,UP,1]/(2,UP,1.5)  [ER,4,0]/(4,UP,3.5)  [ER,7,3]/(7,UP,6.0)  [ER,9,4]/(9,UP,8.0)  [FR,3,DOWN,5]/(3,DOWN,12.0) |
| (FR,1,UP,0)  (ER,10,1)  (FR,2,UP,8)  (ER,5,9)  (FR,4,DOWN,11)  (ER,2,12)  (ER,1,13) | 增加了向下的捎带和同层请求：  [FR,1,UP,0]/(1,STILL,1.0)  [ER,10,1]/(10,UP,5.5)  [ER,5,9]/(5,DOWN,10.5)  [FR,4,DOWN,11]/(4,DOWN,12.0)  [FR,2,UP,8]/(2,DOWN,14.0)  [ER,2,12]/(2,DOWN,14.0)  [ER,1,13]/(1,DOWN,15.5) |
| (FR,1,UP,0)  (ER,4,1)  (FR,4,UP,2)  (FR,2,DOWN,9)  (ER,2,9)  (ER,4,13)  (FR,4,UP,13) | [FR,1,UP,0]/(1,STILL,1.0)  [ER,3,2]/(3,UP,3.0)  [ER,4,2]/(4,UP,4.5)  [FR,4,UP,3]/(4,UP,4.5)  [FR,2,UP,3]/(2,DOWN,6.5)  [ER,7,5]/(7,UP,10.0) |
| (FR,1,UP,0)  (ER,4,1)  (FR,7,UP,2)  (ER,7,2)  (ER,5,3)  (ER,6,3)  (ER,10,4)  (ER,5,12)  (FR,2,DOWN,13)  (ER,2,13) | 增加了未完成的捎带请求成为主请求后，捎带请求发起时间在前的同层请求：  [FR,1,UP,0]/(1,STILL,1.0)  [ER,4,1]/(4,UP,2.5)  [FR,4,UP,2]/(4,UP,2.5)  [FR,2,DOWN,9]/(2,DOWN,10.0)  [ER,2,9]/(2,DOWN,10.0)  [ER,4,13]/(4,UP,14.0)  [FR,4,UP,13]/(4,UP,14.0) |
| repOk() | (FR,1,UP,0)  (ER,4,0)  (ER,4,1)  (FR,2,UP,1)  (FR,2,UP,2)  (ER,7,3)  (ER,9,4)  (FR,3,DOWN,5) | true，之前的测试均用到了repOk，因此只测一组用例。 |

请求队列类：

|  |  |  |
| --- | --- | --- |
| 所测方法 | 测试用例 | 说明 |
| RequestQueue() | 新建实例 | 判断repOk为true即可 |
| ParseInput() | 模拟输入比特流进行输入：  (FR,2,UP,1)  (FR,1,UP,0)  (FR,12,UP,0)  (ER,4,0)  (ER,4,1)  (FR,2,UP,1)  (FR,2,UP,2)  (ER,9,4)  (ER,7,3)  (ER,9,4)  (FR,3,DOWN,5)  RUN | 除去无效请求，有效请求数目为8条 |

傻瓜调度类：

|  |  |  |
| --- | --- | --- |
| 所测方法 | 测试用例 | 说明 |
| Scheduler() | 新建实例 | repOk()为真即可 |
| schedule() | (FR,1,UP,0)  (FR,12,UP,0)  (ER,4,0)  (ER,4,1)  (FR,2,UP,1)  (FR,2,UP,1)  (ER,7,3)  (ER,9,4)  (FR,3,DOWN,25)  RUN | 运行结果为最终在3层，电梯时间为29  [ER,4,0]/(4,UP,2.5)  #SAME [(ER,4,1)]  [FR,2,UP,1]/(2,DOWN,4.5)  #SAME [(FR,2,UP,1)]  #SAME [(FR,2,UP,1)]  [ER,7,3]/(7,UP,8.0)  [ER,9,4]/(9,UP,10.0)  [FR,3,DOWN,25]/(3,DOWN,28.0) |