试验总结及电梯设计思路分析

一、 实验总结

1. 第一次作业

类图：

Bug 分析：

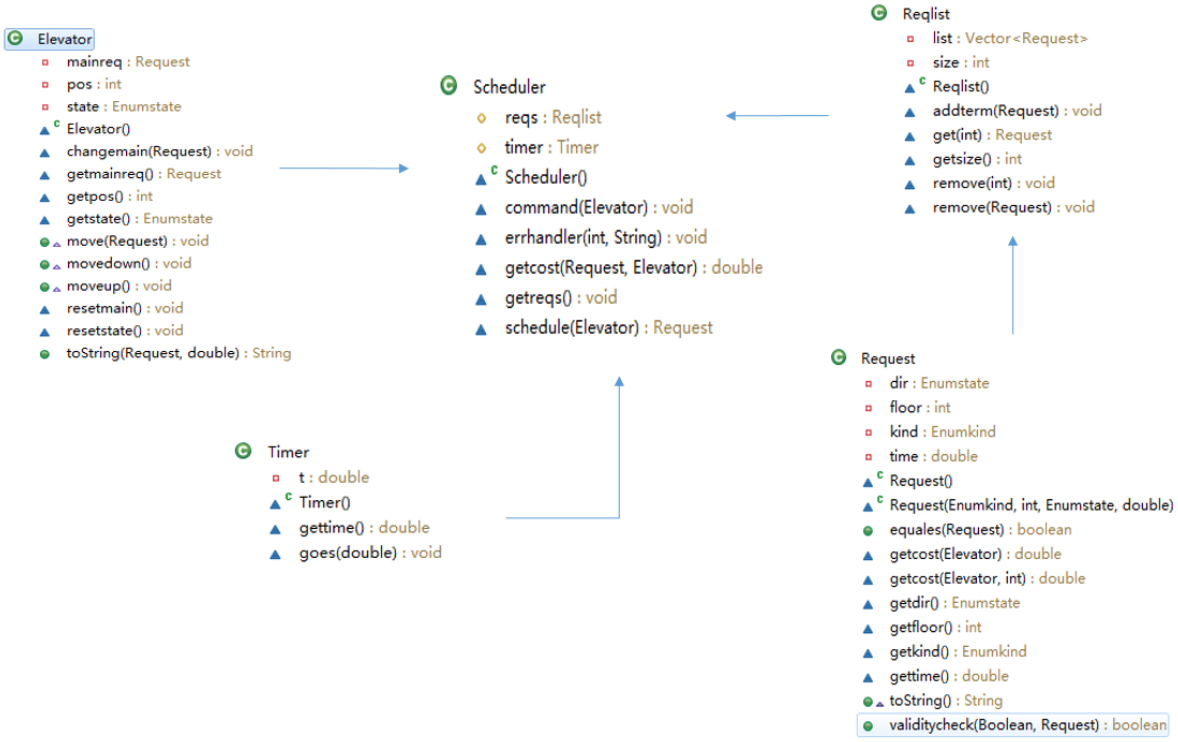
- 不提示错误的容错处理
- 只有一项时输出格式不正确：输出时考虑不周全，Poly 类的 print 方法设计不完善。

测试方法：

- 对输入情况进行分类：合法输入、不合法输入
- 合法输入包括普通输入、边界测试、负数、前导零等情况，分别测试。
- 不合法输入包括括号不合法、逗号不合法、正负号不合法、指数不合法、出现其他字符、数字过大、数量过多等情况，分别测试。

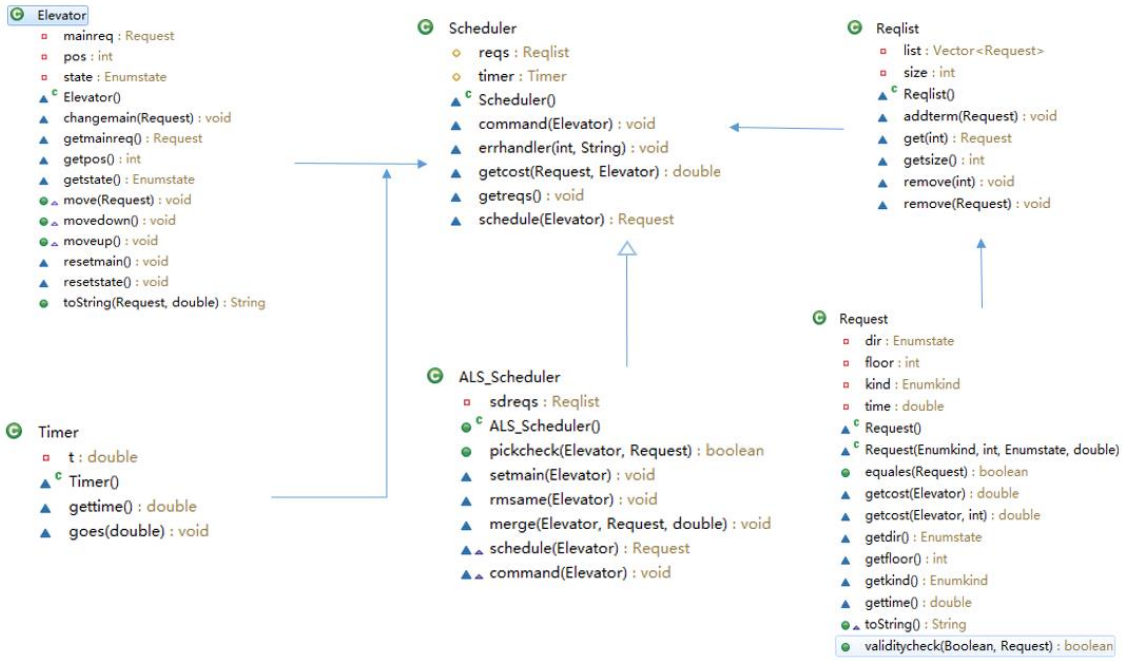
2. 第二次作业

类图：



3. 第三次作业

类图



Bug 分析：

- 捎带请求判断错误：在设计时没有确定好是以楼层判断捎带还是以时间判断捎带，导致有些情况捎带请求判断错误，导致电梯调度错误。
- 同质请求判断不全面。

测试方法：

- 分不同情况测试。
- 随机生成一组数量较大的测试集，多人运行后比较结果，然后对结果不同的测试样例单独分析。

二、 电梯设计思路分析

在后两次试验中，出现了两种主流的电梯设计思路。一种是是电梯一次运动就完成一个请求，而另一种是电梯一次只运动一层楼经多次运动后完成一个请求。我把前一思路称为“**请求型**“，后一种思路称为“**楼层型**”。事实上，还有以 0.5s 或 1s 为单位运行的，但这与“楼层型”电梯本质上是相同的，不再单独分类。

这两种设计思路的优缺点都和明显。“指令型”的优点在于电梯运行过程简单，易于控制，而带来的问题则是难以判断电梯在某时刻的状态。第二次试验我采用的就是这种设计思路，因为不需要捎带也就不需要知道电梯的状态，只要按指令执行即可。“楼层型”则恰好相反，电梯运行过程复杂，不易控制，但却可以明确的知道电梯在某时刻的状态，这对于捎带请求的判断是非常方便的；此外，电梯的运动过程由整化零使得“灵活性”也有了增加，合并请求也更加方便。因此，我在第三次实验前重构时，选择了“楼层型”设计思路。显然，真实的电梯使用的必然是“楼层型”设计思路，因为在后续请求未知时“指令型”电梯显然无法完成捎带任务。

那么，在现在的要求下——所有指令已知并且要求捎带，“指令型”电梯是否可以胜任呢？当然是可以的。

相比于“楼层型”电梯，“指令型”电梯不适用于捎带情况的判断主要原因是电梯在某时刻的状态不易预测，也就使得捎带的判断更加困难。但是，如果只判断在当前运动状态下某时刻的电梯状态则是比较容易的。因此，利用这一特点我们可以一次只寻找一条“最先”被执行的捎带指令，“最先”就是最早被执行的请求，也就是说，在电梯上行的时候，选择可捎带请求中楼层最低的，在电梯下行的时候，选择可捎带指令中楼层最高的。当本条指令被执行完时，再寻找下一条“最先”指令。这里涉及到可捎带的判断问题。我认为一种可行的方法是：在检测某一请求是否可捎带时，可以用电梯在当前状态下继续运行到达这一请求的目标楼层的时间，将预测得出的时间与请求发出时间比较，如果请求时间更早则说明可以捎带。这样的捎带判断局限性也很明显，即判断可靠的前提是电梯目前的运动状态不会被改变，一旦发生了捎带则可能使此后本不能捎带的请求变得可以捎带了，而对于原本就判断为可捎带的请求则没有影响。但是仔细思考可以发现，这样的局限性对我们每次找出“最先”可执行请求是没有影响的，因为某一请求的捎带只会影响执行时间(指令完成时间)比它晚的请求的可捎带性，而我们所寻找的是比它“更早的”请求。

相比之下，“楼层型”电梯要简单很多，我第三次实验采用的也是这种方式，在此不再细谈。