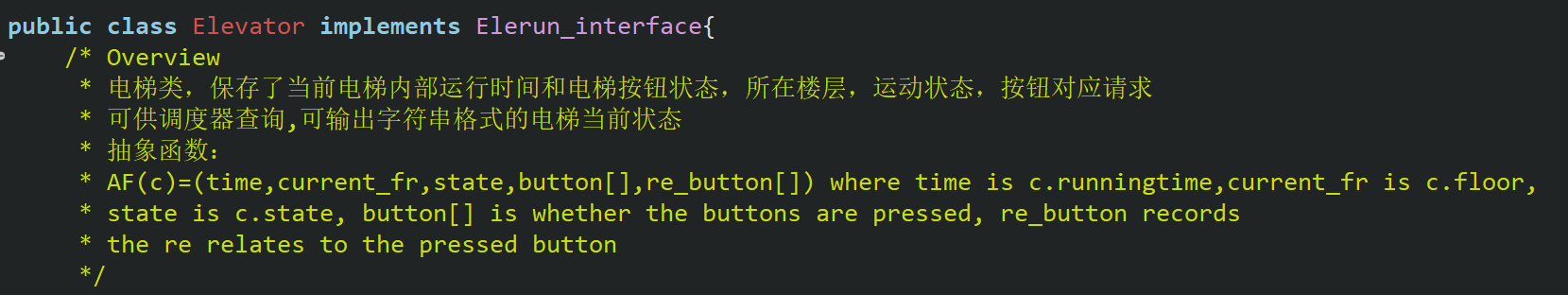
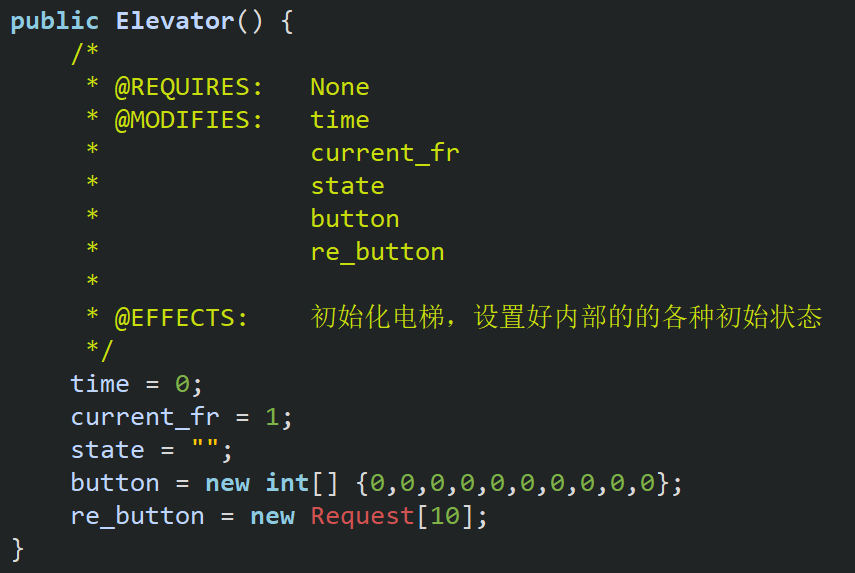
Elevator类实现正确性论证

1. 抽象对象有效实现论证



电梯类管理了电梯内部的各种状态。使用按钮数组和请求数组，状态字符串，楼层，时间（double）来记录电梯的各种状态，抽象函数中将它们映射到了对应的数据结构。

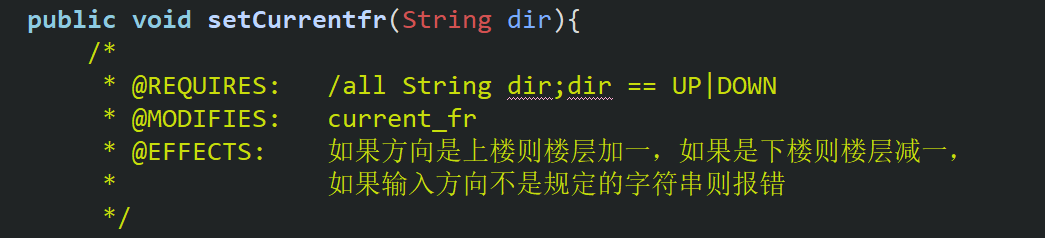
1. 对象有效性论证
   1. 针对构造方法，论证初始态对象的repOK为真。



根据构造方法代码可知执行构造方法之后repOK明显成立

* 1. 逐个论证每个对象状态更改方法的执行都不会导致repOK的返回值为false。

这个类一共提供了3个会改变当前对象状态的方法，分别是



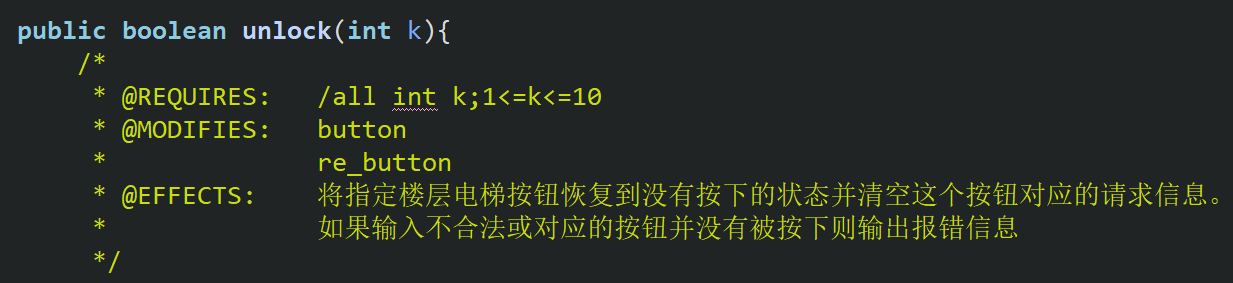
假设setCurrenfr开始执行时repOK为真。

因为这个方法会对输入的状态字符串进行检查，如果不符合要求就会报错而不会更新到run\_dir上，因此运行后仍可以保证repOK为真。



假设pressButton方法开始执行的时候repOK为真。

这个方法将输入的电梯请求对应的请求信息记录到对应的按钮上，在更新前会对输入的请求做是否为空的判断，一旦为空就会报错而不会更新，因此执行之后仍然可以满足repOK

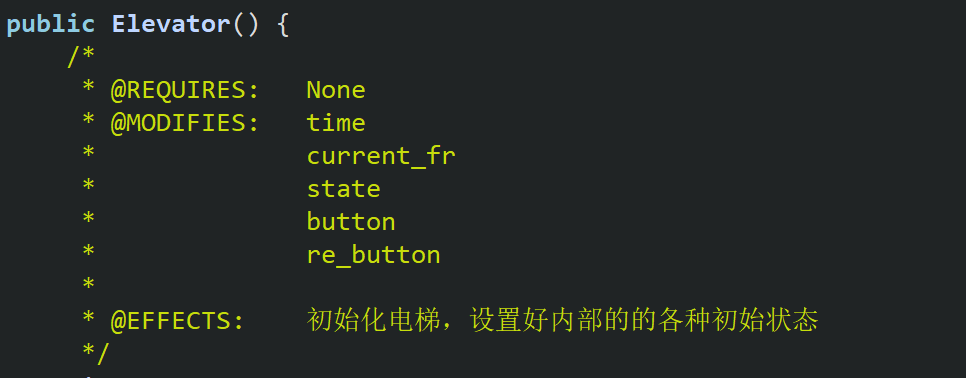


假设unlock方法在执行之前repOK为真。

这个方法会将输入的数字所对应的电梯按钮回复到没有被按下的状态病情Kona个对应的按钮请求。在清空之前会对输入的数字是否符合楼层范围以及该按钮是否已经被按下，若不满足以上条件就会报错而不会解锁改变对象状态，因此执行后repOK仍然成立为真。

* 1. 该类的其他几个方法的执行皆不改变对象状态，因此这些方法执行前和执行后的repOK都为true。
  2. 综上，对该类任意对象的任意调用都不会改变其repOK为true的特性。因此该类任意对象始终保持对象有效性。

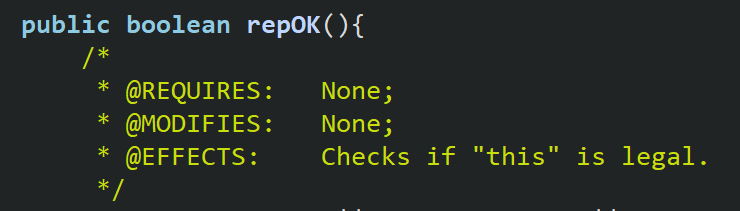
1. 方法实现正确性论证



根据以上规格获得以下划分：

<initialize all the var> with <nothing>

* 该方法是电梯类的构造函数，会将电梯的各种状态初始化，包括运行时间置0，设置当前楼层为1楼，运行状态设置为静止，初始化两个按钮记录数组，故而满足了<initialize all the var>

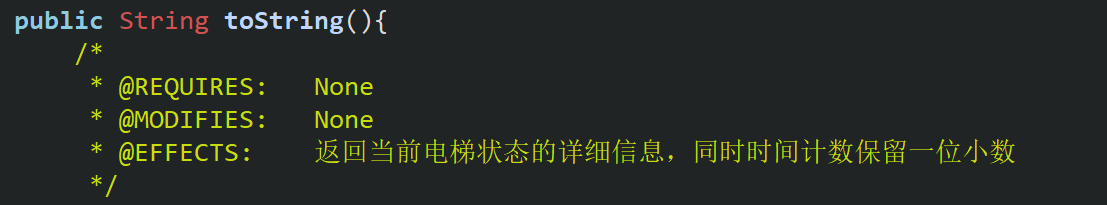


根据以上规格获得以下划分：

<if this is legal, return true>with<nothing>

<if this is not legal, return false>with<nothing>

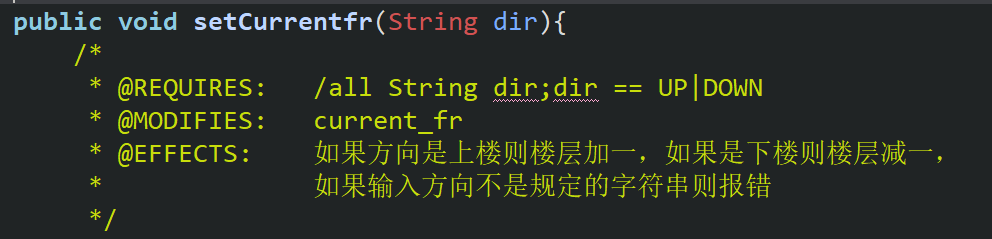
* 该方法是检验对象有效性的时候被调用的。它会检查电梯运行工程中需要使用的各种状态属性以及当前楼层是否合法，因此该方法的确实现了。



根据以上规格获得以下划分：

<return the current ele\_state in string which the time cuts down >with<nothing>

* 该方法将当前电梯状态按照字符串的形式返回，其中返回的时间格式是保留到小数点后一位的。因此该方法满足了<return the current ele\_state in string which the time cuts down >



根据规格获得如下划分：

<currentfr+=1>with<dir==UP>

<currentfr-=1>with<dir == DOWN>

<output the error message>with<dir is illegal>

* 该方法对于输入的字符串代表的当前运动方向来判断应该从所在楼层电梯应该向上或者下。在判断输入方向合法之后，对当前楼层进行更新。因此满足< output the error message>with<dir is illegal>
* <currentfr-=1>with<dir == DOWN>检查过当前方向合法性后，如果dir向上，就给当前电梯所在楼层加一，反之减一。故而满足了<currentfr+=1>with<dir==UP>和<currentfr-=1>with<dir == DOWN>

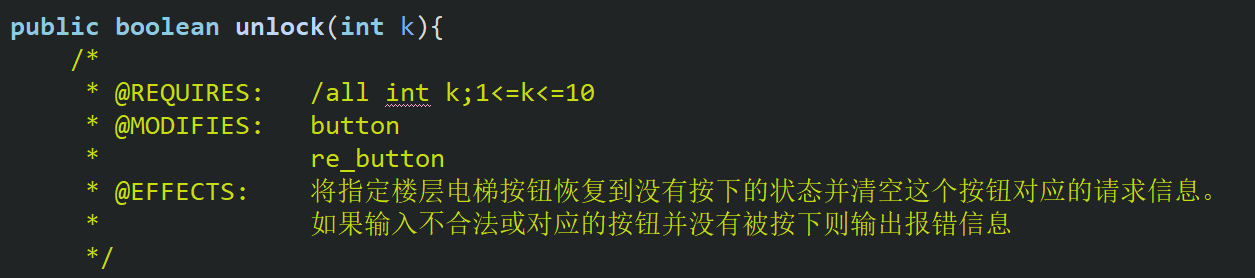


根据规格获得如下划分：

<i=button\_request.floor, button[i] = 1, re\_button[i]=button\_request>with<input is legal>

<output the error message>with<input is illegal>

* <output the error message>with<input is illegal>方法首先检查请求是否有效，然后将这个请求对应的楼层按钮点亮并记录这个请求，所以满足了<i=button\_request.floor, button[i] = 1，re\_button[i]=button\_request>with<input is legal>
* 方法若在开始检查输入请求有效性的时候就发现了问题，则将输出报错信息并不对当前电梯状态进行更新。因此满足了<output the error message>with<input is illegal>



根据规格获得如下划分：

<clear the pressed button>with<k is legal>

<output the error message>with<k is not in range>

<output the error message>with<the k button is not pressed>

* 根据条件分支，当请求的楼层数字合法并且按钮已经被按下的时候，说明可以被恢复到按下之前的状态，因此可以对按钮进行更新，清空记录的请求。

所以实现了<clear the pressed button>with<k is legal>

* 由条件分支可以看到如果请求中的k不符合楼层要求范围或者请求的按钮并没有被按下，就会输出报错信息并不会更新按钮状态。

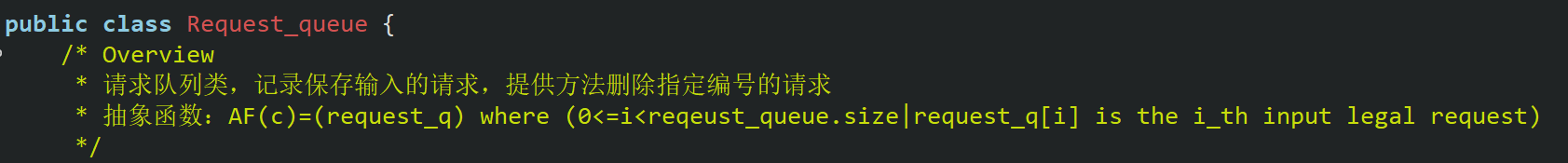
因此实现了<output the error message>with<k is not in range>

和<output the error message>with<the k button is not pressed>

综上所述，所有方法的实现都满足规格。从而可以推断，Elevator的实现是正确的，即满足其规格要求。

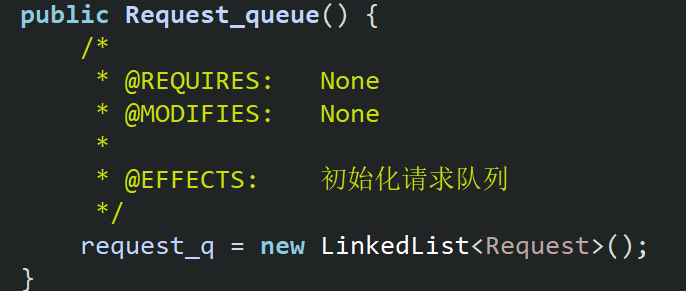
Request\_Queue类实现正确性论证

1. 抽象对象有效实现论证



该请求队列类只作为一个存储请求的容器，封装了基础的链表同时加了一个寻找删除指定编号请求的方法，抽象函数中明确要求了该队列中存储的每一个请求都是有效的某一次输入请求。

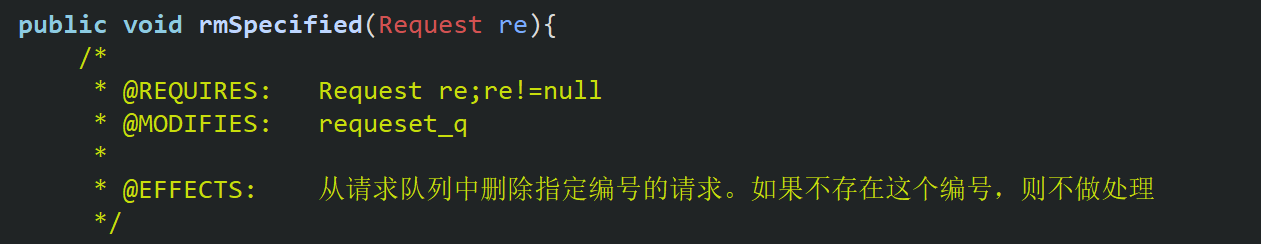
1. 对象有效性论证
   1. 针对构造方法，论证初始态对象的repOK为真。



根据构造方法代码可知执行构造方法之后repOK明显成立

* 1. 逐个论证每个对象状态更改方法的执行都不会导致repOK的返回值为false。

这个类一共提供了1个会改变当前对象状态的方法，是

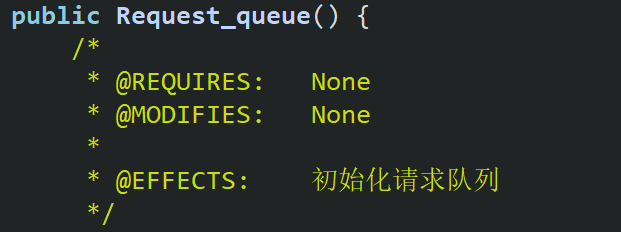


假设rmSpecified开始执行时repOK为真。

该方法通过遍历队列，找到队列中和输入的请求同样编码的第一个请求删除掉，然后返回。只会改变队列的size但不会让队列变成null野对象名。故而执行之后仍然成立repOK。

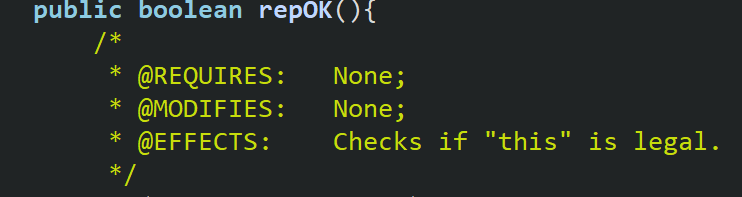
* 1. 该类的其他方法的执行皆不改变对象状态，因此这些方法执行前和执行后的repOK都为true。
  2. 综上，对该类任意对象的任意调用都不会改变其repOK为true的特性。因此该类任意对象始终保持对象有效性。

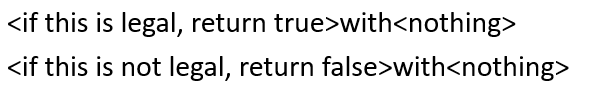
1. 方法实现正确性论证



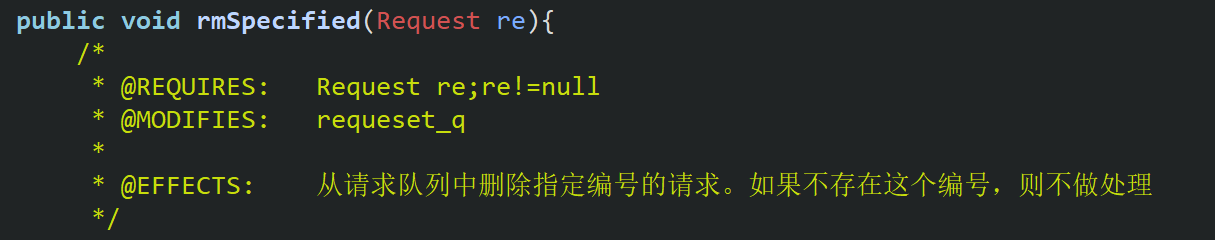
<initialize all the var> with <nothing>

* 该方法是请求队列类的构造函数，会初始化一个空链表。故而满足了<initialize all the var>





* 该方法是检验对象有效性的时候被调用的。它会检查请求队列是否还存在，不为NULL，因此该方法的确实现了 <check if this is legal>with<nothing>



根据规格划分可以得到：

<do nothing>with<re==null>

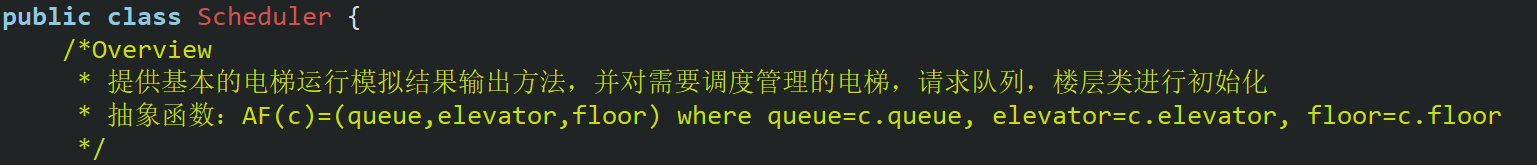
<do nothing>with<re.NO is not in the list>

<remove>with<re!=null&&re.NO in the list>

* 由代码中的条件分支可以看到一旦re为null则会直接返回，因此实现了<do nothing>with<re==null>
* 而在进入循环之后，遍历链表元素，比较每一个链表中记录的请求的编号和输入的请求编号是否相同，如果相同则删除这个请求，并退出循环，否则不作任何变化。因此如果链表中不存在这个请求就不会发生任何事情。也即实现了<do nothing>with<re.NO is not in the list>
* 同上分析，如果找到了编号相同的，则删除这个请求并返回即可。（存储的编号唯一，由readin类保证，该类不在此次分析范围内）。故而也实现了<remove>with<re!=null&&re.NO in the list>

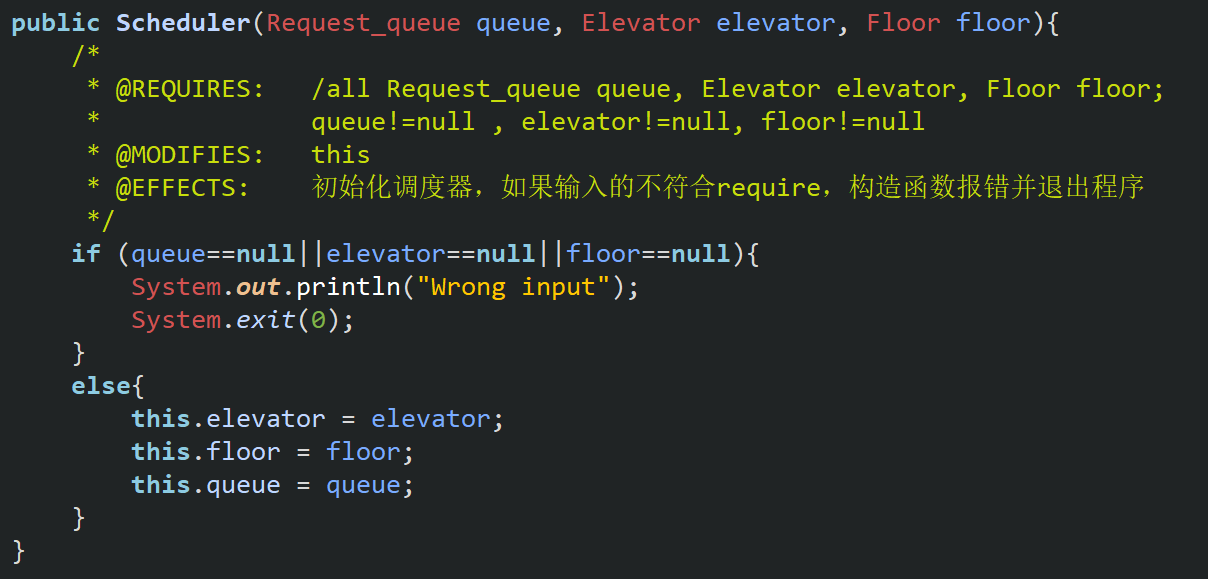
Scheduler类实现正确性论证

1. 抽象对象有效实现论证



这个类作为电梯调度器的父类，只存储需要管理的电梯，请求队列和楼层类，同时提供模拟结果的输出方法。抽象函数将电梯，请求队列和楼层分别映射到调度器中声名的自定义类。

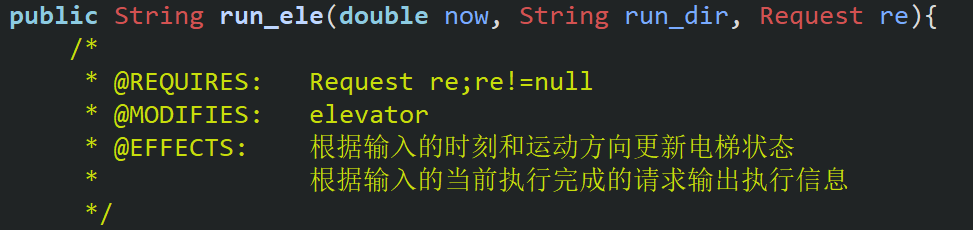
1. 对象有效性论证
   1. 针对构造方法，论证初始态对象的repOK为真。



根据构造方法代码可知执行构造方法之后repOK明显成立

* 1. 逐个论证每个对象状态更改方法的执行都不会导致repOK的返回值为false。

这个类一共提供了1个会改变当前对象状态的方法，是

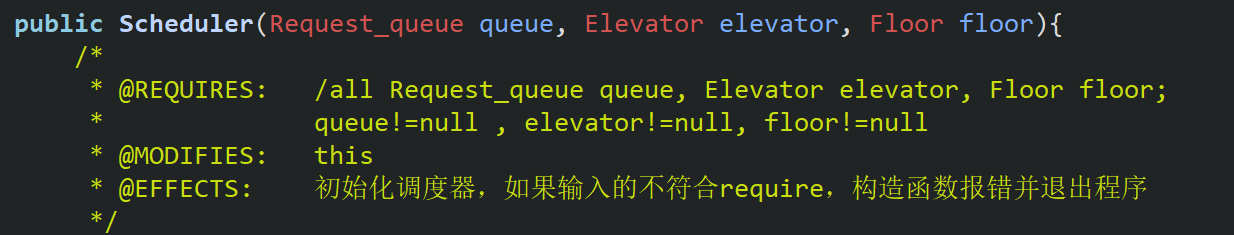


假设run\_ele开始执行时repOK为真。

该方法除了输出请求和电梯信息之外，只会更新电梯当前运行时间和运动方向，由于输入的时间和方向并不会改变管理的对象的存在性，不会把被管理的对象变成null，故而执行之后仍然成立repOK。

* 1. 该类的其他方法的执行皆不改变对象状态，因此这些方法执行前和执行后的repOK都为true。
  2. 综上，对该类任意对象的任意调用都不会改变其repOK为true的特性。因此该类任意对象始终保持对象有效性。

1. 方法实现正确性论证



根据规格划分可以得到

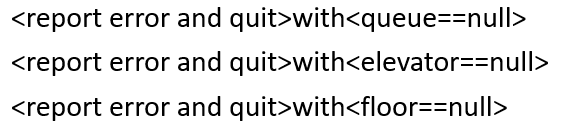
<report error and quit>with<queue==null>

<report error and quit>with<elevator==null>

<report error and quit>with<floor==null>

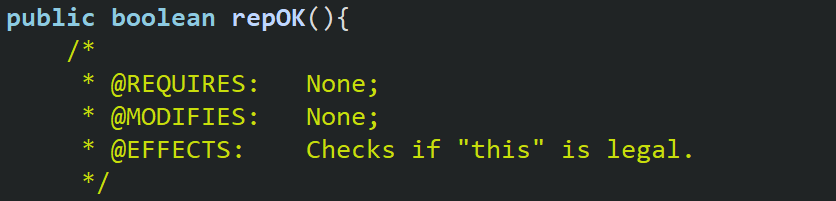
<initialize>with<inputs are not null>

* 可以看到构造函数在进入的时候就对三个输入进行了检查，如果任一输入为null就会报错并退出，从而实现了

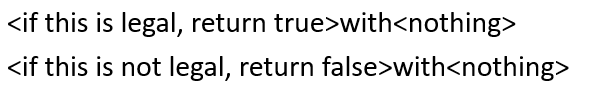


* 进一步地，如果通过了if条件的检查，调度器就会初始化为这三个输入，因此也实现了

<initialize>with<inputs are not null>

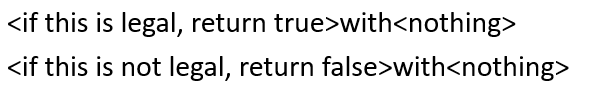


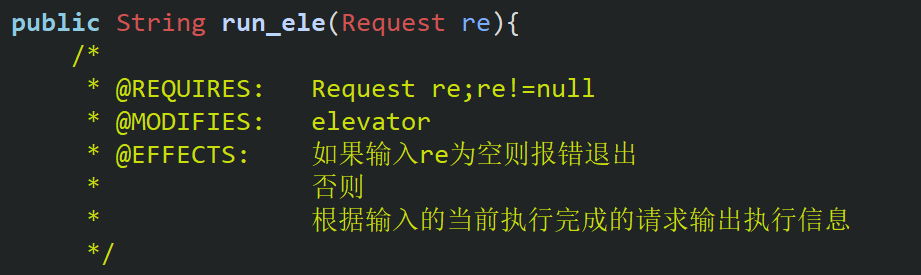
根据规格划分这个方法可以得到



* 这个方法会检查调度器有效性，在经过初始化之后，这个调度器的属性都不会为空，检查可以通过，而在构造函数初始化之前或者初始化失败的时候，调度器属性为null就会返回错误。

因此实现了





根据规格划分这个方法可以得到

<report and quit>with<re==null>

<output the run info>with<re==null>

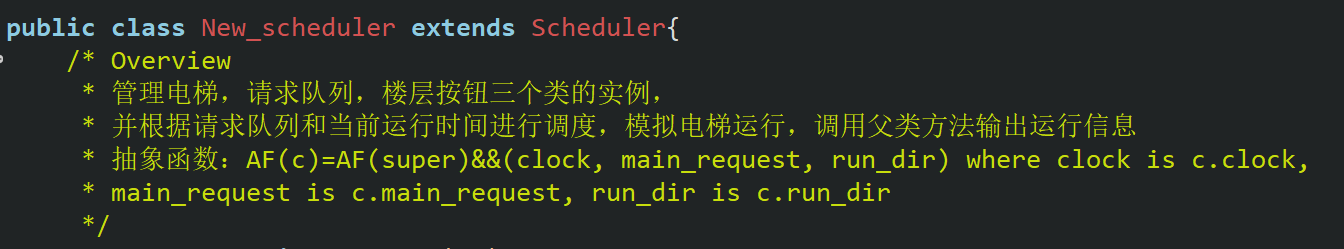
* 该方法在一开始就对输入的re进行了检查，如果不对就会报错并退出。如果re符合要求就可以输出字符串形式的请求信息，因此，实现了

<report and quit>with<re==null>

<output the run info>with<re==null>

New\_Scheduler类实现正确性论证

1. 抽象对象有效实现论证



新调度器继承了父类调度器对于基本数据对象的管理，包括电梯，楼层和请求队列，因此抽象函数中也包含了父类的抽象函数，此外，子类调度器还可以根据请求队列模拟电梯的运行，更新电梯运行时间并调用父类的方法将电梯的运行信息输出。抽象函数使用了自定义对象request和(double)时间，(String)运动方向来模拟映射到调度器管理的电梯状态，运行时间和主请求。

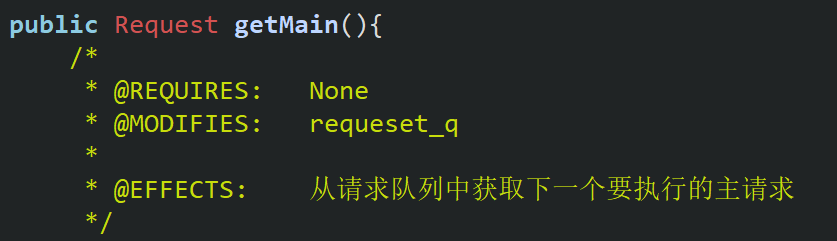
1. 对象有效性论证
   1. 针对构造方法，论证初始态对象的repOK为真。



根据构造方法代码以及父类已经证明的有效性可知，执行构造方法之后repOK明显成立

* 1. 逐个论证每个对象状态更改方法的执行都不会导致repOK的返回值为false。

这个类一共提供了2个会改变当前对象状态的方法，是

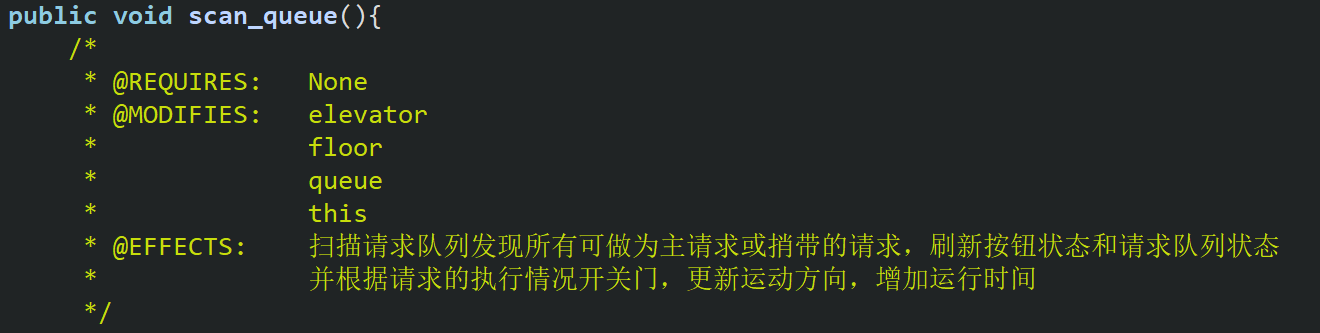


假设getMain开始执行时repOK为真。

这个方法会从请求队列中寻找带有稍带标记的请求并选第一个找到的请求为新的主请求，如果找不到就用队列第一个为主请求。并将刚才找到的主请求移出队列。

这个方法只会改变队列的size但不会改变队列的存在性。

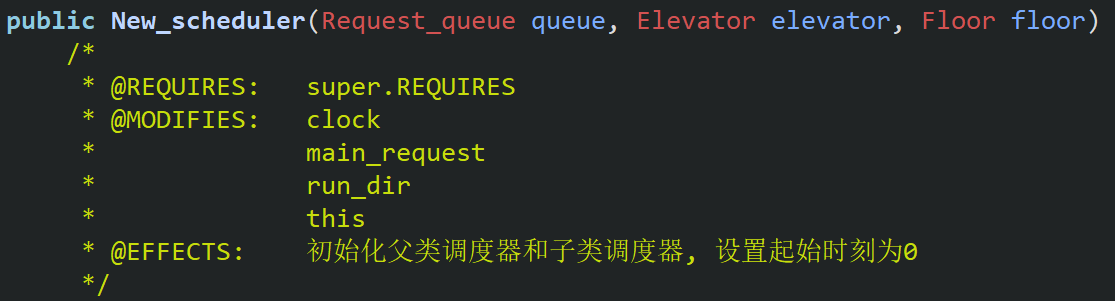
故而执行之后仍然成立repOK。



这个方法通过扫描请求队列模拟运行电梯，并调用外部类的方法更新按钮状态，更新电梯运行时间，最终调用父类方法输出电梯模拟运行的结果字符串。运行过程中对于请求队列，按钮数组，电梯状态都只是改变数值或存储的数据但是不改变变量或者对象本身的存在性，故而调用之后仍然满足repOK

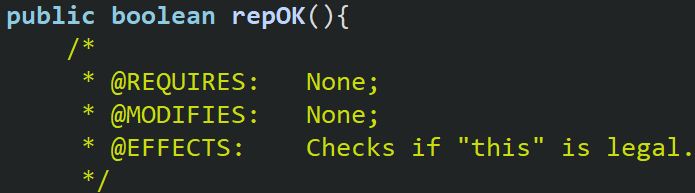
* 1. 该类的其他方法的执行皆不改变对象状态，因此这些方法执行前和执行后的repOK都为true。
  2. 综上，对该类任意对象的任意调用都不会改变其repOK为true的特性。因此该类任意对象始终保持对象有效性。

1. 方法实现正确性论证

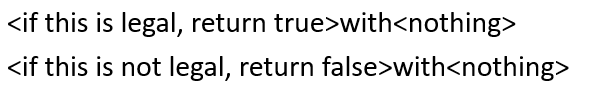


* 由于子类的要求和父类相同而父类正确性已经论证，因此父类部分不再重复。

子类部分规格可以划分为<initialize>with<super.Requires>因此一旦父类构造函数正确，子类即可正确初始化，也即实现了<initialize>with<super.Requires>

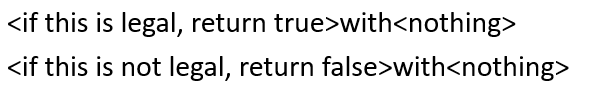


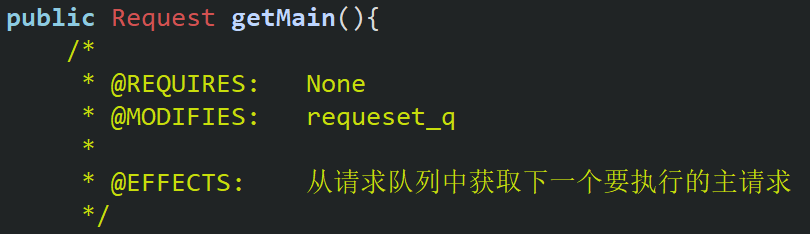
根据规格划分可以得到



* 这个方法检查子类的有效性，只要父类repOK满足且子类rundir有效，即可通过，父类的部分实现正确性也在前文已经得到证明。如果任一条件不满足，都会返回false。

因此实现了

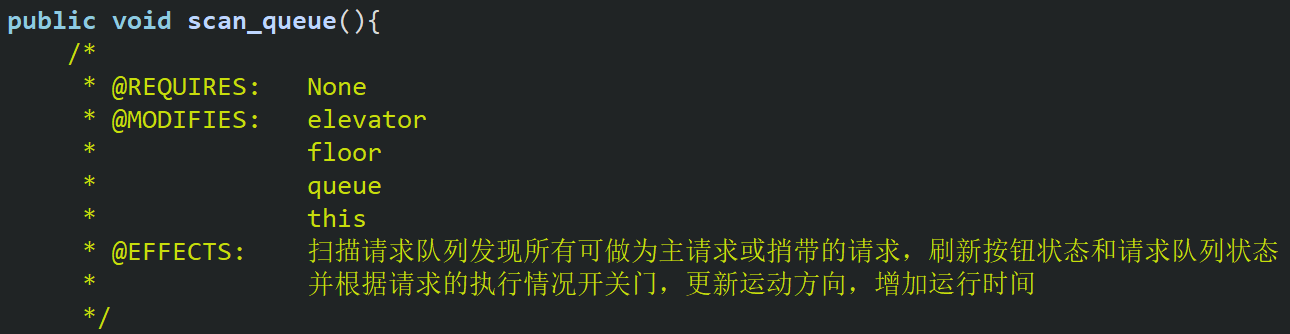




根据规格划分可以得到

<get a new mian\_re from the request\_list>with<nothing>

* 这个方法会遍历请求队列，直到找到第一个带有稍带标记的请求，这个请求就会成为新的主请求，如果遍历完全部队列仍没有找到带有稍带标记的请求，那么就取队列队首请求作为新的主请求。找到主请求后，将这个请求从请求队列中删除，并返回这个请求给调用者。因此实现了规格<get a new mian\_re from the request\_list>with<nothing>



根据规格划分可得：

<scan the queue&&run ele simulation&&refresh time,button,state&&output>with<nothing>

这个方法主要负责扫描请求队列，并根据请求队列扫描结果模拟运行电梯，更新相关状态并输出电梯的模拟运行信息。首先根据当前时间判断是否有按钮被按下，之后判断主请求和请求队列是否为空，只有当主请求为空且请求队列也空的时候才说明执行完了，然后选取主请求，更新运动状态，判断运动状态到达的逻辑，到达新的一层楼之后是否有请求被执行完了，是否需要更新电梯的相关状态并输出，在这个过程中时钟完全模拟电梯的运行速度，时间增加，电梯状态也会按照逻辑随着时间改变。因此实现了

<scan the queue&&run ele simulation&&refresh time,button,state&&output>with<nothing>