正确性论证文档

一、ALS\_Schedule调度类实现正确性的论证

1. 抽象对象得到了有效实现论证

论证类的overview明确了抽象对象，且类的rep能够通过抽象函数映射到相应的抽象对象。在有继承层次的情况下，注意区分子类抽象函数与父类抽象函数的关系，并能够有效重用父类的论证结果。

overview：通过继承机制实现增加顺路捎带的电梯调度

rep：请求队列AR、上行up、下行down、当前楼层loc\_n、系统相对时间t

不变式：1=<loc\_n<=10、t>=0

1. 对象有效性论证

论证在任意时刻，一个类的任意对象都能保证repOK为真，即任意一个方法的执行都不会导致rep为假。该类的属性声明为public，无法进行有效论证。在有继承层次的情况下，如果父类所有的rep都通过private进行保护，则无需进行额外论证，否则必须针对每个对父类rep状态进行修改的方法进行论证，证明不会改变父类对象的repOK。注意首先假设repOK的实现是正确的。在方法论证部分对repOK的正确性进行论证。

1. 针对构造方法，论证初始态对象的repOK为真。

ALS\_Schedule提供了一个构造方法ALS\_Schedule()，它初始化全部的rep，repOK的运行结果显然返回true。

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。

ALS\_Schedule提供了三个状态更新方法：classifyRQ()、guideRunning()、moveConsole()，下面逐个进行论证。

* 假设classifyRQ()方法开始执行时，repOK为true。

1. classifyRQ()方法首先定义一些局部变量maxT、priT、priDN、priLN、j、order等，显然不改变repOK的取值；
2. 然后判断如果当前请求队列中的第一个请求是有效的，则接着判断该请求是否已经存在于电梯调度中的请求队列，若存在，则标记该请求为主请求，否则将该请求加入电梯调度中的请求队列，此时不会导致repOK为false；
3. 根据当前处理请求的请求时间和系统相对时间t的大小关系重新设置当前系统相对时间t，这一过程中t始终不小于0，因此repOK依然为真；
4. 根据电梯调度中的请求队列的楼层排序和运行方向标记基于当前主请求的捎带请求，在每次调度完成后删除请求队列中对应的请求且不断增加t的值，并输出相关捎带信息，此时不会导致repOK为false；
5. 因此，该方法根据请求时间的先后和楼层相对位置比较输出请求捎带信息，完成请求队列的重排序，其执行不会导致repOK为假，不违背表示不变式。

* 假设guideRunning()方法开始执行时，repOK为true。

1. guideRunning()方法根据电梯当前楼层和请求目标楼层设置方向，仅改变up和down的值，显然不会导致repOK为false；
2. 因此，该方法的执行不会导致repOK为假，不违背表示不变式。

* 假设moveConsole()方法开始执行时，repOK为true。

1. moveConsole()方法首先定义一些局部变量ult\_des\_n、flag、t、s等，显然不改变repOK的取值；
2. 然后设置当前请求队列中的第一个请求为主请求，接着判断当前请求队列中的其他请求是否可为可捎带请求，此时不会导致repOK为false；
3. 根据当前处理请求的请求时间和系统相对时间t的大小关系重新设置当前系统相对时间t，这一过程中t始终不小于0，因此repOK依然为真；
4. 根据电梯调度中的请求队列的楼层排序和运行方向标记基于当前主请求的捎带请求，在每次调度完成后删除请求队列中对应的请求且不断增加t的值，并重新设置电梯当前所处楼层，特别需要注意的是，此时loc\_n始终为1到10之间的某个整数，再输出相关停靠信息，此时不会导致repOK为false；
5. 因此，该方法根据捎带信息排序后的请求队列完成电梯调度，其执行不会导致repOK为假，不违背表示不变式。
6. 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。
7. 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。
8. 方法实现正确性论证

目标：在方法的前置条件和后置条件没有逻辑矛盾的前提下，根据前置条件和后置条件分析整理出完整的{<effect Y> with <input partition X>}，然后对照方法代码实现来论证给定每个<partition X>，方法执行结果都满足<effect Y>。

注意在继承层次下，子类方法如果是重载了父类方法，则需要论证：父类方法的前置条件蕴含子类方法的前置条件、子类方法的后置条件蕴含父类方法的后置条件。

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

二、FoolSchedule调度类实现正确性的论证

1. 抽象对象得到了有效实现论证

论证类的overview明确了抽象对象，且类的rep能够通过抽象函数映射到相应的抽象对象。

overview：描述单电梯傻瓜调度的基本实现，包括楼层、电梯、请求和队列等基本对象，以及响应请求完成调度等基本操作

rep：楼层F、电梯E、请求R、请求队列Q

不变式：none

1. 对象有效性论证

论证在任意时刻，一个类的任意对象都能保证repOK为真，即任意一个方法的执行都不会导致rep为假。该类的属性声明为public，无法进行有效论证。注意首先假设repOK的实现是正确的。在方法论证部分对repOK的正确性进行论证。

1. 针对构造方法，论证初始态对象的repOK为真。

FoolSchedule未提供任何构造方法，repOK的运行结果显然返回true。

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。

FoolSchedule提供了一个状态更新方法：F\_Schedule()，下面进行论证。

* 假设F\_Schedule()方法开始执行时，repOK为true。

1. F\_Schedule()方法首先定义一些局部变量i等，显然不改变repOK的取值；
2. 然后判断如果当前请求队列中的第一个请求是有效的，则响应该请求并输出电梯停靠信息，否则输出无效请求的提示信息，此时不会导致repOK为false；
3. 因此，该方法实现傻瓜调度，若请求无效则输出提示信息，否则按请求时间排序逐条响应完成调度，其执行不会导致repOK为假，不违背表示不变式。
4. 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。
5. 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。
6. 方法实现正确性论证

目标：在方法的前置条件和后置条件没有逻辑矛盾的前提下，根据前置条件和后置条件分析整理出完整的{<effect Y> with <input partition X>}，然后对照方法代码实现来论证给定每个<partition X>，方法执行结果都满足<effect Y>。

注意在继承层次下，子类方法如果是重载了父类方法，则需要论证：父类方法的前置条件蕴含子类方法的前置条件、子类方法的后置条件蕴含父类方法的后置条件。

三、elevator类实现正确性的论证

1. 抽象对象得到了有效实现论证

论证类的overview明确了抽象对象，且类的rep能够通过抽象函数映射到相应的抽象对象。

overview：描述一部电梯的基本实现，包含当前所处楼层位置、目标停靠位置和当前系统相对时间等基本对象，以及响应请求即搭载乘客等基本操作。

rep：电梯当前所处楼层loc\_n、运行目标楼层des\_n、当前系统相对时间t

不变式：1=<loc\_n<=10、1=<des\_n<=10、t>=0

1. 对象有效性论证

论证在任意时刻，一个类的任意对象都能保证repOK为真，即任意一个方法的执行都不会导致rep为假。该类的属性声明为public，无法进行有效论证。注意首先假设repOK的实现是正确的。在方法论证部分对repOK的正确性进行论证。

1. 针对构造方法，论证初始态对象的repOK为真。

elevator未提供了任何构造方法，repOK的运行结果显然返回true。

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。

elevator提供了一个状态更新方法：recordResponse(String str, floor F)，下面进行论证。

* 假设recordResponse(String str, floor F)方法开始执行时，repOK为true。

1. recordResponse(String str, floor F)方法首先判断输入的str是“FR”还是“ER”，若为前者，则基于F.loc\_n设定目标楼层des\_n，并根据F.t与当前的系统相对时间t的大小关系令t取较大者，显然des\_n始终为1到10 之间的一个整数且t始终不小于0，不改变repOK的取值；
2. 然后根据请求中的目标楼层和当前楼层相比较完成上行或下行的电梯调度，即更新电梯当前所处楼层loc\_n和系统相对时间t，这一过程中，显然loc\_n始终为1到10 之间的一个整数且t始终不小于0，不会导致repOK为false；
3. 根据当前处理请求的请求时间和系统相对时间t的大小关系重新设置当前系统相对时间t，这一过程中t始终不小于0，因此repOK依然为真；
4. 因此，该方法返回当前电梯运行方向，其中，1=UP、-1=DOWN、0=STATIC，并依据请求修改电梯状态信息，其执行不会导致repOK为假，不违背表示不变式。
5. 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。
6. 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。
7. 方法实现正确性论证

目标：在方法的前置条件和后置条件没有逻辑矛盾的前提下，根据前置条件和后置条件分析整理出完整的{<effect Y> with <input partition X>}，然后对照方法代码实现来论证给定每个<partition X>，方法执行结果都满足<effect Y>。

1. recordResponse(String str, floor F):

/\*

\* Requires：str==“ER” or "FR"

\* F!=null

\* Modifies：des\_n

\* t

\* loc\_n

\* Effects：返回当前电梯运行方向，依据请求修改电梯状态信息

\* 1=UP

\* -1=DOWN

\* 0=STATIC

\*/

根据上述过程规格，获得如下的划分：

<str!=”ER” && str!=”FR” || F=null> with <return 0>

<str=”FR”> with <des\_n=F.loc\_n>

<str=”FR” && F.t > t> with <t=F.t>

<(des\_n-loc\_n)>0> with <t+=1+0.5\*Math.abs(des\_n-loc\_n); loc\_n=des\_n; return 1>

<(des\_n-loc\_n)<0> with <t+=1+0.5\*Math.abs(des\_n-loc\_n); loc\_n=des\_n; return -1>

<(des\_n-loc\_n)=0> with <return 0>

* 方法首先检查确认输入str和F的状态，如果str不是“ER”或“FR”，或F为null，则直接返回0结束，表示电梯处于静止状态，满足<str!=”ER” && str!=”FR” || F=null> with <return 0>
* str为“FR”，通过方法检查。设置当前电梯系统相对时间为楼层求求发出时间。因此，满足<str=”FR”> with <des\_n = F.loc\_n>
* des\_n大于loc\_n，通过方法检查。设置t以(1+0.5\*Math.abs(des\_n-loc\_n))为基准叠加，令电梯当前所处楼层为目标楼层，并返回1，表示电梯完成当前请求调度且处于上行状态。因此，处理满足<(des\_n-loc\_n)>0> with <t+=1+0.5\*Math.abs(des\_n-loc\_n); loc\_n=des\_n; return 1>。
* des\_n小于loc\_n，通过方法检查。设置t以(1+0.5\*Math.abs(des\_n-loc\_n))为基准叠加，令电梯当前所处楼层为目标楼层，并返回-1，表示电梯完成当前请求调度且处于下行状态。因此，处理满足<(des\_n-loc\_n)<0> with <t+=1+0.5\*Math.abs(des\_n-loc\_n); loc\_n=des\_n; return -1>。
* des\_n等于loc\_n，通过方法检查。直接返回0结束，表示电梯处于静止状态。。因此，处理满足<<(des\_n-loc\_n)=0> with <return 0>。

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。

四、request类实现正确性的论证

1. 抽象对象得到了有效实现论证

论证类的overview明确了抽象对象，且类的rep能够通过抽象函数映射到相应的抽象对象。

overview：描述请求的基本实现，包括目标楼层、请求发出时间、方向等基本对象，以及判断请求是否有效等基本操作

rep：请求各项tag、目标楼层des\_n、请求时间t、请求字符串s、上行up、下行down

不变式：1=<des\_n<=10、t>=0、3=<tag.length<=4

1. 对象有效性论证

论证在任意时刻，一个类的任意对象都能保证repOK为真，即任意一个方法的执行都不会导致rep为假。该类的属性声明为public，无法进行有效论证。注意首先假设repOK的实现是正确的。在方法论证部分对repOK的正确性进行论证。

1. 针对构造方法，论证初始态对象的repOK为真。

request未提供了任何构造方法，repOK的运行结果显然返回true。

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。

request提供两个状态更新方法：judgeRequest(String str)、realizeRequest(floor F,elevator E)，下面逐个进行论证。

* 假设judgeRequest(String str)方法开始执行时，repOK为true。

1. judgeRequest(String str)方法首先定义一些局部变量i等，显然不改变repOK的取值；
2. 然后判断读入的字符串str是否为空，若为空，则直接返回false，否则由正则表达式得到tag，此时不会导致repOK为false；
3. 根据tag的长度和各元素组成判断读入的字符串格式是否正确，若有任何一项不符合规格要求，则直接返回false，否则所有项均满足要求并最终返回true，在此过程中，des\_n可能会随着读入的请求发生变化，但始终保持为1到10 之间的一个整数，同样地，t在不小于零的基础上非递减变化，repOK依然为真；
4. 因此，该方法判断请求是否有效，若有效返回true，并设置请求基本状态参数，否则返回false，其执行不会导致repOK为假，不违背表示不变式。

* 假设realizeRequest(floor F,elevator E)方法开始执行时，repOK为true。

1. realizeRequest(String str)方法首先判断当前读入请求为“ER”还是“FR”，若为前者，则设置F.loc\_n为des\_n，设置F.t为t，并最终返回true，若为后者，则仅当E.t不大于t时设置E.des\_n为des\_n，设置E.t为t，并最终返回true，否则直接返回false，显然不改变repOK的取值；
2. 因此，该方法判断当前请求是否可行，若可行则返回true，并根据请求类型设置好楼层和电梯的相关状态参数，否则返回false，其执行不会导致repOK为假，不违背表示不变式。
3. 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。
4. 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。
5. 方法实现正确性论证

目标：在方法的前置条件和后置条件没有逻辑矛盾的前提下，根据前置条件和后置条件分析整理出完整的{<effect Y> with <input partition X>}，然后对照方法代码实现来论证给定每个<partition X>，方法执行结果都满足<effect Y>。

1. judgeRequest(String str):

/\*

\* Requires：str!=null

\* Modifies：tag

\* s

\* des\_n

\* t

\* Effects：判断请求是否有效

\* 若有效返回true，并设置请求基本状态参数

\* 否则返回false

\*/

根据上述过程规格，获得如下的划分：

<str=null> with <return false>

<tag.length<3 || tag.length>4> with <return false>

<tag[0]=”ER” && 1=<tag[1]<=10 && tag[2] is digit && tag.length=3> with <return true>

<tag[0]=”FR” && 1=<tag[1]<=10 && (tag[2]=”UP” || tag[2]=”DOWN”) &&tag[3] is digit && tag.length=4> with <return true>

* 方法首先检查确认输入str的状态，如果str为null，则直接返回false结束，表示输入无效，满足<str=null> with <return false>
* tag=str.split("\\,")，tag的长度小于3或大于4，通过方法检查。直接返回false结束，表示输入无效。因此，满足<tag.length<3 || tag.length>4> with <return false>
* tag[0]为“ER”，tag[1]为1到10之间的某个整数，tag[2]为整数，tag的长度为3，通过方法检查。最终返回true结束，表示电梯内请求有效。因此，满足<tag[0]=”ER” && 1=<tag[1]<=10 && tag[2] is digit && tag.length=3> with <return true>
* tag[0]为“FR”，tag[1]为1到10之间的某个整数，tag[2]为“UP”或“DOWN”，tag[3]为整数，tag的长度为4，通过方法检查。最终返回true结束，表示楼层间请求有效。因此，满足<tag[0]=”ER” && 1=<tag[1]<=10 && tag[2] is digit && tag.length=3> with <return true>。

1. realizeRequest(floor F,elevator E):

/\*

\* Requires：F!=null

\* E!=null

\* Modifies：F

\* E

\* Effects：判断当前请求是否可行

\* 若可行则返回true，并根据请求类型设置好楼层和电梯的相关状态参数

\* 否则返回false

\*/

根据上述过程规格，获得如下的划分：

<F=null || E=null> with <return false>

<tag[0]=”FR”> with <F.loc\_n=des\_n; F.t=t; return true>

<tag[0]=”ER” && Double.parseDouble(tag[2])>=E.t> with <E.des\_n=des\_n; E.t=t; return true>

<tag[0]=”FR” && 1=<tag[1]<=10 && (tag[2]=”UP” || tag[2]=”DOWN”) &&tag[3] is digit && tag.length=4> with <return true>

* 方法首先检查确认输入F和E的状态，如果F为null或E为null，则直接返回false结束，表示当前请求无效，满足<F=null || E=null> with <return false>。
* tag[0]为”FR”，通过方法检查。设置楼层当前位置为电梯目标楼层，设置楼层当前系统时间为请求产生时间，并最终返回true，表示电梯成功设置相关状态参数。因此，满足<tag[0]=”ER” && Double.parseDouble(tag[2])>=E.t> with <E.des\_n=des\_n; E.t=t; return true>。

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

五、queue类实现正确性的论证

1. 抽象对象得到了有效实现论证

论证类的overview明确了抽象对象，且类的rep能够通过抽象函数映射到相应的抽象对象。

overview：描述一个队列的基本实现，包括判断是否满足入队条件、入队等基本操作。

rep：请求字符串队列RQ

不变式：none

1. 对象有效性论证

论证在任意时刻，一个类的任意对象都能保证repOK为真，即任意一个方法的执行都不会导致rep为假。该类的属性声明为public，无法进行有效论证。注意首先假设repOK的实现是正确的。在方法论证部分对repOK的正确性进行论证。

1. 针对构造方法，论证初始态对象的repOK为真。

queue未提供任何构造方法，repOK的运行结果显然返回true。

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。

queue提供了两个状体更新方法：addQueue()、judgeQueue()，下面逐个进行论证。

* 假设addQueue()方法开始执行时，repOK为true。

1. addQueue()方法首先定义一些局部变量s等，设s为当前读入的字符串请求并忽略空格，显然不改变repOK的取值；
2. 然后判断如果当前字符串请求为“END”，则直接返回-1，表示输入结束，此时不会导致repOK为false；
3. 当且仅当s的长度合适时，判断s的起始字符和终止字符是否为一对小括号，若是，则去掉小括号并将新的s加入字符串请求队列RQ，并返回1，表示入队成功，此时repOK依然为真；
4. 除上述的任何其他情况都认为是读入的字符串请求无效，并返回0，此时不会导致repOK为false；
5. 因此，该方法若输入结束则返回-1，反之，入队成功返回1，否则返回0，其执行不会导致repOK为假，不违背表示不变式。

* 假设judgeQueue()方法开始执行时，repOK为true。

1. judgeQueue()方法首先定义一些局部变量IR、cn、index、sign等，显然不改变repOK的取值；
2. 然后判断读入的字符串请求是否有效，若有效，则接着读入，否则标记该无效请求的序号，此时不会导致repOK为false；
3. 若未读入任何请求，则输出相关提示信息，否则输出所有标记的无效请求的提示信息，repOK依然为真；
4. 因此，该方法读入字符串请求并判断是否有效，若RQ为空则提示输入中不包含有效请求，针对入队失败的请求输出提示信息，其执行不会导致repOK为假，不违背表示不变式。
5. 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。
6. 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。
7. 方法实现正确性论证

目标：在方法的前置条件和后置条件没有逻辑矛盾的前提下，根据前置条件和后置条件分析整理出完整的{<effect Y> with <input partition X>}，然后对照方法代码实现来论证给定每个<partition X>，方法执行结果都满足<effect Y>。

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

六、floor类实现正确性的论证

1. 抽象对象得到了有效实现论证

论证类的overview明确了抽象对象，且类的rep能够通过抽象函数映射到相应的抽象对象。在有继承层次的情况下，注意区分子类抽象函数与父类抽象函数的关系，并能够有效重用父类的论证结果。

overview：描述一个楼层的基本实现，包括楼层号和时间等基本对象

rep：楼层位置loc\_n、当前系统相对时间t

不变式：1=<loc\_n<=10、t>=0

1. 对象有效性论证

论证在任意时刻，一个类的任意对象都能保证repOK为真，即任意一个方法的执行都不会导致rep为假。该类的属性声明为public，无法进行有效论证。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。

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