OO-第十四次作业-正确性论证

# 控制类（Controler\_incidentally）

## 1.1 抽象对象有效实现

该类实现了捎带的调度方式，并可以将电梯状态进行输出,实现了电梯状态对象。

## 对象有效性

* + 1. Controler\_incidentally()是构造函数，新建一个控制对象，将status初始化为“STILL”使其不为null，将prn初始化为0。
    2. Schedule执行可捎带的调度过程，期间电梯状态不会成为null，因此不会影响repOK。
    3. FindMin、Threemin、FindEleBuUP、FindEleBuDOWN、FindnextTime、toString这些方法都不会修改对象。因此不会影响repOK。

所以，所有方法执行前和执行后的repOK都为true。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。

## 方法实现正确性

**void** schedule (Input De,Floor F,Elevator E)



<do nothing> with <De == null || F == null || E == null>

<RunTime += 0.5> with <any>

<set status as “STILL”> with <DST == nowfloor>

<set status as “UP”> with <Floor[first] > nowfloor>

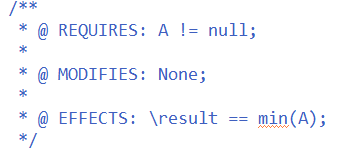
<set status as “DOWN”> with <Floor[first] < nowfloor>

<set status as “STILLUP”> with <DST == nowfloor && UPsDOWNb == true>

<set status as “STILLDOWN”> with <DST == nowfloor && DOWNsUPb == true>

* 方法首先检查确认传入的De、F和E是否为null，如果为null，直接返回
* 对于任意的情况，每进行一次循环，运行时间会加0.5秒。
* 对于当前楼层等于目标楼层，将电梯状态设置为STTILL
* 对于接下来的第一个命令楼层大于当前楼层，将状态设置为UP
* 对于接下来的第一个命令楼层小于当前楼层，将状态设置为DOWN
* 对于当前楼层等于目标楼层且UPsDOWNb标志为true时，将状态设置为STILLUP
* 对于当前楼层等于目标楼层且DOWNsUPb标志为true时，将状态设置为STILLDOWN

**int** FindMin(**int**[] A,**int** begin)

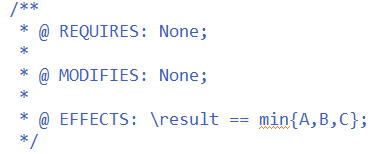


<\result == -1> with <A == null>

<\result == min(A)> with <A != null>

* 方法先检查传入的数组A是否为null，如果为null，直接返回-1
* 如果不为null，则返回数组A中从A[begin]开始最小的元素

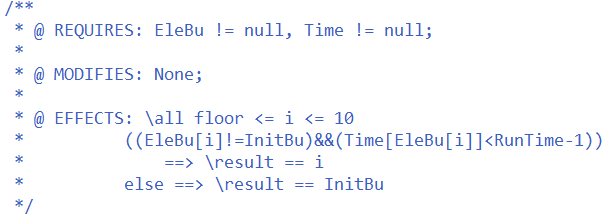
**int** Threemin(**int** A,**int** B,**int** C)



<\result == min{A,B,C}> with <any>

* 方法对于任何情况都会返回ABC中最小的数

**int** FindEleBuUP(**int**[] EleBu,**double**[] Time ,**int** floor)



<\result == -1> with <EleBu == null || Time == null>

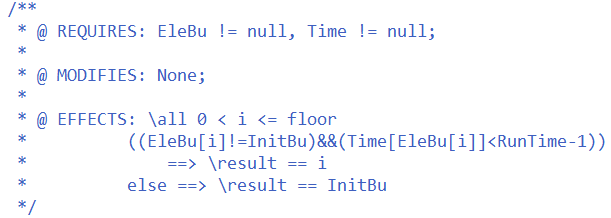
<\result == i> with

<(EleBu[i]!=InitBu)&&(Time[EleBu[i]]<RunTime-1)>

<\result == InitBu> with <other condition>

* 方法先检查传入的EleBu和Time数组是否为null，若为null直接返回-1
* 对于任意的floor<=i<=10,且电梯第i层的按钮被按下、按下的时间小于运行时间减去开门时间，则返回i
* 若无上述情况则返回InitBu

**int** FindEleBuDOWN(**int**[] EleBu,**double**[] Time ,**int** floor)



<\result == -1> with <EleBu == null || Time == null>

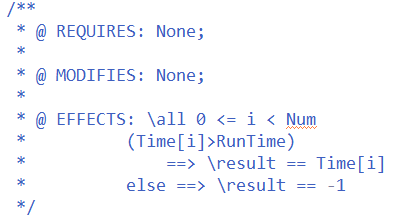
<\result == i> with

<(EleBu[i]!=InitBu)&&(Time[EleBu[i]]<RunTime-1)>

<\result == InitBu> with <other condition>

* 方法先检查传入的EleBu和Time数组是否为null，若为null直接返回-1
* 对于任意的0<i<=floor,且电梯第i层的按钮被按下、按下的时间小于运行时间减去开门时间，则返回i
* 若无上述情况则返回InitBu

**double** FindnextTime(**double**[] Time,**int** Num)

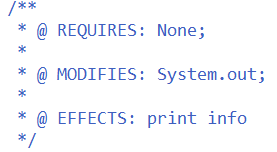


<\result == Time[i]> with <Time[i] > RunTime>

<\result == -1> with <other condition>

* 对于任意的0<=i<=Num,如果第i个指令的时间大于运行时间，则返回第i个指令的时间
* 若无满足的指令时间，则返回-1

**void** toString (**int**[] Floor,**int**[] Method,**double**[] Time,**int** nextIn)

****

<print information of arrival in standard form> with <any>

* 对于任意的情况都会根据传入的数据输出一次电梯的状态

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

# 电梯类(Elevator)

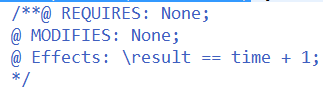
## 2.1 对象有效性

* + 1. Elevator只有opendoor和running方法，所以repOk，不涉及类中数据更改，所以恒为true。

所以，所有方法执行前和执行后的repOK都为true。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。

## 方法实现正确性

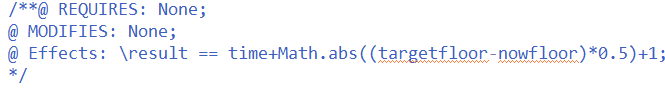
**double** opendoor(**double** time)



<time ++> with <any>

* 对于任意的情况都会返回时间+1

**double** running(**double** time,**int** nowfloor,**int** targetfloor)



<\reslut == time+Math.abs((targetfloor-nowfloor)\*0.5)+1> with <any>

* 对于任意的情况都会返回到目标楼层花费的时间加上开门时间

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。

# 楼层类（Floor）

## 3.1 抽象对象有效实现

包含了floor这个数据，实现了上下楼的操作。

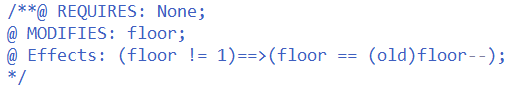
## 3.2 对象有效性

* + 1. Floor()是构造函数，将楼层设为1符合repOK在1楼到10楼之间。
    2. down和up会在上下楼前判断是否到达1楼和10楼所以不会影响repOK.

所有方法执行前和执行后的repOK都为true。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。

## 3.3方法实现正确性

**void** down()

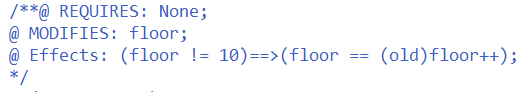


<do nothing> with <floor == 1>

<floor--> with <floor != 1>

* 若电梯到达最底层则不会有任何影响
* 若电梯未到最低层则下一层

**void** up()



<do nothing> with <floor == 10>

<floor++> with <floor != 10>

* 若电梯到达最高层则不会有任何影响
* 若电梯未到最高层则上一层

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。

# 请求类（Request）

## 4.1 抽象对象有效实现

实现了请求的匹配和处理。

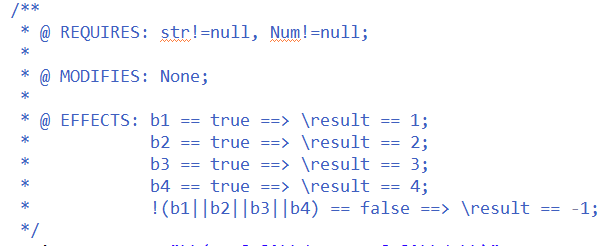
## 4.2 对象有效性

* + 1. 构造函数不会将rep设为null，所以不会影响repOK。
    2. MatchD只会根据传入的字符串判断并返回，所以不影响repOK。

所有方法执行前和执行后的repOK都为true。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。

## 4.3方法实现正确性

**int** MatchD(String str,**int** Num)



<return 1> with <str.matches("\\(FR,[+]?\\d+,DOWN,[+]?\\d+\\)")>

<return 2> with <str.matches("\\(FR,[+]?\\d+,UP,[+]?\\d+\\)")>

<return 3> with <str.matches("\\(ER,[+]?\\d+,[+]?\\d+\\)")>

<return 4> with <str.matches("RUN")>

<return -1> with <above condition all false>

* 若与"\\(FR,[+]?\\d+,DOWN,[+]?\\d+\\)"匹配上，则返回1
* 若与"\\(FR,[+]?\\d+,UP,[+]?\\d+\\)"匹配上，则返回2
* 若与"\\(ER,[+]?\\d+,[+]?\\d+\\)"匹配上，则返回3
* 若与"RUN"匹配上，则返回4
* 若都匹配不上，则返回-1

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

# 读入类（Input）

## 5.1 抽象对象有效实现

实现了对输入的处理和储存。

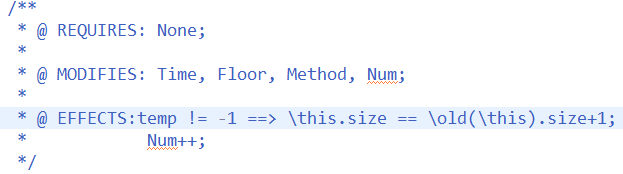
## 5.2 对象有效性

* + 1. 构造函数不会将rep设为null，所以不会影响repOK。
    2. Input只会根据控制台输入修改数组的项，不会使得数组为null由于设定了100的最大输入，也不会越界。所以不影响repOK。

所有方法执行前和执行后的repOK都为true。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。

## 5.3方法实现正确性

**void** Inputhandle()



<Time.add(scanner.nextline.time)> with <temp != -1>

<Floor.add(scanner.nextline.floor)> with <temp != -1>

<Method.add(scanner.nextline.method)> with <temp != -1>

<Num++> with <temp != -1>

* 如果匹配上，则将输入的信息加入到时间、楼层、方法数组中，并将指令数目加一

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