程序运行环境说明

请使在搭载 windows10 系统的非 MAC 系列电脑上 JDK 版本 jdk-8u161-windows-x64 环境中，使用 eclipse-java-oxygen-2-win32-x86_64 运行该程序。如果以上要求不满足，运行时有可能出现未知错误。注：仅支持控制台输入，不支持使用文件输入和命令行输入。

程序功能说明

模拟一个傻瓜电梯的运行，输入请求，输出电梯的合法请求相应的操作。

电梯调度策略

1) **程序运行开始或重置时设置电梯停靠在一层；**

2) 名词解释：

   a) 电梯运行状态：从电梯启动时刻（此时速度>0）,到电梯运动停止时（此时速度刚刚为 0）的运行状态。（启动时刻，运行停止时刻]，期间运行速度始终大于 0。

   b) 电梯开关门状态：电梯静止时，从门打开时刻（门开始动作），到门完全关闭时刻（门刚刚停止动作）时的状态。（门打开时刻，门关闭时刻]

   c) 电梯停留状态：电梯停在某层，且门长时间处于关闭状态。（门关闭时刻，门准备打开时刻或电梯准备启动时刻]

3) 电梯外**一行**只能发出**一个上行**或**下行请求。**电梯未到本楼层的时候，某个请求按钮变亮后，再按不会产生实际效果，但是发出上行请求后可以再发出下行请求，反之亦可，这

视为两个不同的请求，执行完一个后另一个仍需执行。在电梯到达某楼层处于开关门状态时，该楼层的多个同向请求只认为是一个请求。当电梯关门动作完成后（含门完全静止的那一刻），可以再产生新的上下行请求；

4) 电梯内的一个目标楼层按钮只能发出对应目标楼层的请求，一旦发出某个目标楼层请求后，在电梯到达该楼层并完成关门动作前（**包括关门完毕时刻**），目标楼层与该按钮楼层相同的多个电梯内请求被认为是一个请求。当电梯关门结束后，可以再发出任意目标楼层请求。

5) 所有请求按照请求发出的时间顺序被电梯系统管理和调度，按照时间上先来先服务的策略（First Arrived First Served，FAFS）进行调度。

6) 如果电梯同时收到了电梯内请求和楼层请求时，则按照输入时的请求排列顺序执行。

7) 电梯系统采用傻瓜式调度策略：控制系统不断扫描请求队列，按照 FAFS 策略取出待响应请求，只有当该请求被执行完毕之后，才会尝试调度下一个请求。**请注意此处不支持“顺路”的请求。**如从 2 层去 8 层期间，未到 6 层时有 6 层的上行请求，应先处理完 2 层到 8 层的请求，再处理 6 层的请求；或者 2 层到 8 层时，中间又发出了去 6 层的请求，同样先执行完到 8 层的动作，再处理去 6 层的请求。

输入格式及标准

1. 用户输入为按照请求产生时间排序的请求序列（注意：可以输入时间相同的两个请求，先输入的请求被优先执行），序列通过字符串表示；
2. 请求分为两类：一类是楼层请求，一类是电梯内请求。

3. 楼层请求格式为：(FR, m, UP/DOWN, T)，其中 FR 为楼层请求标识，m 为发出请求的楼层号，UP 为向上请求，DOWN 为向下请求，T 为发出时刻。（注释：相当于请求者在楼道里的某楼层按“上行”或“下行”键）
4. 电梯内请求格式为：(ER, n, T)，其中 ER 为电梯内请求标识，n 为请求前往的目标楼层号，T 为发出时刻。（注释：相当于人在电梯里按一个目标楼层号）
5. 使用除以上两种以及 RUN 命令的任何多余或错误的输入，均会判定为无效输入。
6. 所有的逗号应采用 ASCII 字符集中的逗号“,”，而不是中文字符逗号“，”。请求之间必须通过换行进行分隔，**两条请求之间不允许有空行（有空行的话处理详见补充事项）。**一条请求的内部元素之间可以有空格，程序能够自动过滤。
7. T 为请求产生的相对时刻，第一个请求的 T 值必须设置为 0，否则视为无效输入！等到第一个合法的 T 值为 0 的输入，才正式作为第一条指令接受指令。
8. 标准输入的请求是按照时间排序的，如果遇到一个乱序的请求，即请求产生时间小于前面一个请求产生时间，判定为无效输入。
9. 设电梯运行一个楼层距离的时间消耗为 0.5s；达到楼层后一次开关门动作时间消耗为 1.0s。
10. 合法的请求产生时刻为非负整数（要求最大为 4 字节的非负整数，关于符号和前导 0 请看补充事项中的叙述），n，m 为 1~10 之间（包含）的正整数。
11. 不正确的标识符，不正确的方向，不正确的数字范围，多余的其他非允许字符，均认定为不合法输入，即无效输入。
12. 特别地，对于 FR 标识符，1 楼的 DOWN 和 10 楼的 UP 也认为是无效输入。
13. 在一行内只能输入一条请求，一行内输入多条请求，整条被视为无效输入。将输入全部请求后，必须再键入 RUN 表示输入结束，回车后程序开始执行调度。 多于 100 的情

况请看补充事项中的叙述。

14. **本次作业不要求楼层请求和电梯请求的顺序符合真实情况，不考虑有没有人。**

输入示例：

(FR,3,DOWN,0)

(FR,1,UP,1)

(ER,1,2)

(ER,6,4)

RUN

输出格式标准

1. 程序的输出为按照时间排序的电梯运行状态描述，包括以下内容：电梯停靠的楼层、停靠前的运动方向及停靠时刻（**即电梯刚到达目标楼层由运动转为静止状态，尚未执行开关门的时刻**）：

   格式为：(n,UP/DOWN,t)

2. 一行只有一个输出结果

3. 其中 n 为楼层号，UP/DOWN 为电梯运行方向（只显示 UP 或 DOWN 之一）；t 为相对于第一个请求发生的时间（即相对于时间初始值 0）。t 的规格如下：t 是保留小数点后一位有效位的浮点数，如果 t=3，则输出为 3.0；如果 t=0，则输出 0.0。

4. 有同层请求时（即电梯停在某层，此时有目标为该层的请求），则输出为：（n,STILL,t），**此处 t 应考虑电梯执行一次开关门动作的时间。**

5. 本程序要求一次性输入所有请求，然后执行程序进行电梯调度并输出结果。

6. 要求**程序能够忽略相同的请求**，包括产生时刻**相同的相同请求和产生时刻不同但是实质**

上相同的请求。此类输入为同质输入，同质输入在输出是会反馈如下：

#Invalid Request

7. 在输出时如不满足要求，为无效输入，将有说明性信息（具体如下）格式如下

ERROR

#解释

8. 对于无效输入，包括但不局限于非法字符括号不匹配，时间超过标准，输入负楼层和时间，时间超过要求，楼层不存在，输入格式不对等大部分无效输入，反馈的解释：

ERROR

Wrong Input

9. 对于格式基本正确，但是首次输入时间不为 0 的无效输入，反馈的解释：

ERROR

#The first request must start from relative time 0

10. 对于格式基本正确，电梯外的第一层要求向下的无效输入，反馈的解释：

ERROR

#There is no DOWN button ont the Floor 1

11. 对于格式基本正确，电梯外的十层要求向上的无效输入，反馈的解释：

ERROR

#There is no UP button ont the Floor 10

12. 对于格式基本正确，时间乱序的无效输入，反馈的解释：

ERROR

#Time Never Goes Back

13. 同时出现以上错误的时候理论上按住从上到下进行检测报错，但以上也仅是参考，供测

试者查看方便，不应该以以上逻辑层次作为绝对标准。

14. 当出现特殊情况的时候，会爆出

ERROR

#Wrong

输入输出补充事项

1. 所有无效输入在其回车后直接反馈 ERROR，输入完毕后运转时，同质请求将在输出的对应位置输出

#Invalid Request

2. 输入的请求时间和楼层必须为非零十进制整数，可以带或不带“+”，但是不能为负数。请求时间和楼层允许使用前导 0，但是应注意，符号以外的数字长度不得超过 50 位。
3. RUN 行中的空格也会被忽略，但是如果 RUN 行出问题会被划归为命令出问题（比如输入 run 或 Run，会爆 ERROR\n#Wrong Input ）但不会停止输入，会继续等待输入正确的 RUN。
4. RUN 前面的输入均被判定无效与无输入直接 RUN 两种情况可以看作一种情况。映射到现实生活，可以理解为该电梯没人会用和无人使用，故 RUN 之后无任何输出（但是前一种情况中 RUN 之前的 ERROR 报错会正常报错）
5. 虽然禁止了输入空行（直接回车或只有空格的行），如果输入还是有空行，会被忽略，即对空行不做任何处理。（如果你全是空行然后 RUN，理论上应该什么也不输出，但是期望输入会不能填）。另外虽然空行不做处理，但是会算一条输入。即 6 中的 100 条限制，

包含空行，空行也算作一条。

6. 包括 RUN 的输入总行数<=100 时，可以完全准确的运行。 当不包括的 RUN 的输入达到 100 时，电梯不会马上启动，输入第 101 行并回车后，电梯启动，从第 101 行输入起的任何输入系统不会做出响应。具体来说，如果你在 console 一条条输入，第 101 条是可以输入的，并且系统会直接将此条当作 RUN（即系统将强行将 101 行输入置为 RUN）。102 行将无法输入。对于不是一条条输入而是粘贴到 console 的输入。系统只对前 100 行响应，第 101 行视为 RUN。

类说明文档

Scheduler 类

属性：

private Elevator elevator; 电梯本体主要用于执行请求

private Queue queue; 请求序列，对其中的请求进行操作

private double clock; 时钟记录时间

private Request lastRq; 记录上一条请求

方法：

public Scheduler() 构造函数

public void run() 开始运行电梯的入口

public void command() 调用 schedule 方法获取请求，使用电梯的 move 方法使电梯运行指令

public Request schedule() 根据电梯状态，清洗队列终的无效请求，获取下一条有效请求

public void parse(String str) 调用 Queue 类的 parseRq 方法进行输入的字符串的分析，根据错误类型进行报错

public static void main(String args[]) 程序入口

Request 类

属性：

private Requester guest; 记录请求者

private int askfloor; 记录 FR 请求的请求楼层

private int target; 记录 ER 目标楼层

private Direction direction; 记录 FR 的按下方向

private long time; 记录请求的时间

方法：

public Requester getGt() 返回请求者

public int getTg() 返回请求的请求楼层

public int getAf() 返回 ER 目标楼层

public Direction getDr() 返回 FR 的按下方向

public long getTime() 返回请求的时间

public Request(Requester a, int b, Direction c, long d) FR 请求的构造函数

public Request(Requester a, int b, long d) ER 请求的构造函数

Queue 类

属性：

private static final int MAX = 200; 设定常数边界

private Request[] rqList; 储存队列的全部请求

private Boolean[] validity; 储存每个请求的合法性（即是否同质）

private int front; 记录请求队列队首位置

private int rear; 记录请求队列的队尾位置

方法：

public Queue() 构造函数

public Request frontRq() 返回队首的请求

public boolean frontVal() 返回队首请求的合法性

public void moveFront(int n) 将队首 n 个位置，即忽略现在队首的 n 个请求

public boolean end() 返回是否整个请求队列是否已经处理完毕

public void wash(Request lastRq, double clock) 根据时间和刚运行完的上一条请求，清洗同质请求将其合法性置为非法

public int parseRq(String str) 将输入的字符串解析为请求储存，返回错误类型

Elevator 类

属性：

private int floorNow; 记录当前楼层

private double clock; 时钟

方法：

public Elevator() 构造函数

public double move(Request rq) 根据请求上下移动，返回新的 clock