# 处理机管理-电梯调度

1751022 李翠琪

1. **项目目的**

* 学习调度算法
* 通过实现电梯调度，体会操作系统调度过程
* 学习特定环境下多线程编程方法

1. **项目需求**

* 某一层楼20层，有五部互联的电梯。基于线程思想，编写一个电梯调度程序。（可考虑：楼层和电梯数可设置）。
* 每个电梯里面设置必要功能键：如数字键、关门键、开门键、上行键、下行键、报警键、当前电梯的楼层数、上升及下降状态等。
* 每层楼的每部电梯门口，应该有上行和下行按钮和当前电梯状态的数码显示器。
* 五部电梯门口的按钮是互联结的，即当一个电梯按钮按下去时，其他电梯的相应按钮也就同时点亮，表示也按下去了。
* 所有电梯初始状态都在第一层。每个电梯如果在它的上层或者下层没有相应请求情况下，则应该在原地保持不动。
* 调度算法自行设计。

1. **项目设计及算法**

**3.1项目开发概览**

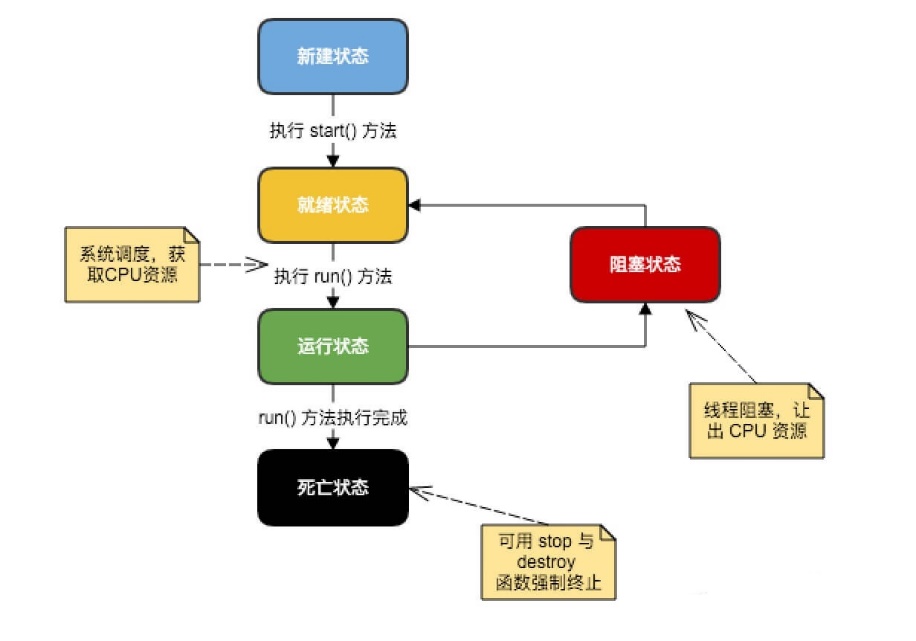
* + 采用JAVA开发语言,使用JAVA的Swing类库来进行图形界面设计与开发。
  + 在windows操作系统环境下，采用IDEA作为开发工具。
  + 通过实现 Runnable 接口来进行多线程编程。
  + 采用扫描算法（SCAN算法）来进行电梯调度。

**3.2扫描算法介绍**

* + 简介：扫描算法（SCAN） 是一种按照楼层顺序依次服务请求，它让电梯在最底层和最顶层之间连续往返运行，在运行过程中响应处在于电梯运行方向相同的各楼层上的请求。
  + 优点:进行寻找楼层的优化，效率比较高。较好地解决了电梯移动的问题，在这个算法中，每个电梯响应乘客请求使乘客获得服务的次序是由其发出请求的乘客的位置与当前电梯位置之间的距离来决定的。所有的与电梯运行方向相同的乘客的请求在一次电向上运行或向下运行的过程中完成，免去了电梯频繁的来回移动。
  + 缺点:扫描算法是一个非实时算法，他的平均响应时间较长。
  1. **项目多线程**

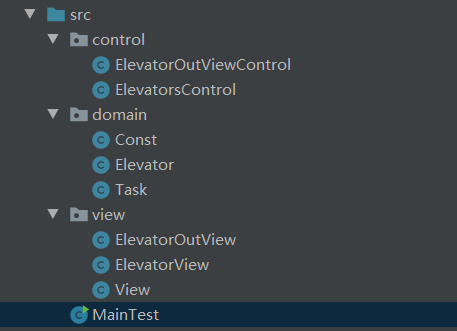
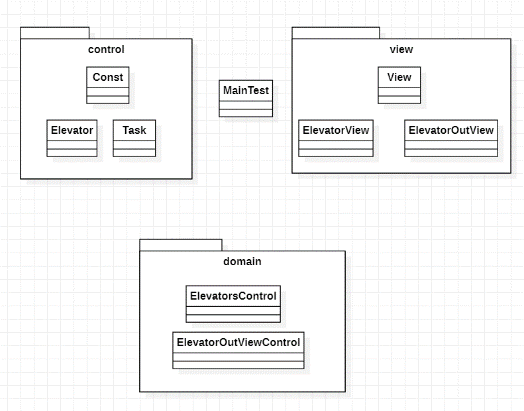
计划在项目中通过implement Runnable接口来实现多线程，之后重写里面的run（）方法。并且通过new Thread(该类).start()来开启一条多线程。

其中，电梯要采用多线程方式启动以此实现调度，并且电梯调度控制类也需要以多线程方式调度，以此实时监听任务并且调度电梯。除此之外，在电梯视图控制类也需要多线程方式启动，以此实时监听并且控制画面刷新。



**3.4项目架构**

* + 总体采用三层架构,分别是实体层，展示层,控制层。
  + 实体层中有三个类，Const类负责存储项目的各种常量，Elevator类负责创建电梯实体，Task类则负责存储实际需求任务。
  + 展示层则负责页面的展示，共有三个类，View负责展示整体画面。ElevatorView负责展示电梯内部的按钮，ElevatorOutView负责展示每层楼电梯外部按钮。
  + 控制层负责控制事件触发以及调度算法执行。ElevatorOutViewControl负责电梯按钮触发事件以及动态刷新页面等等。ElevatorsControl则根据SCAN算法负责对所有触发的任务进行电梯调度。
  + 最后，MainTest是函数的入口，负责以多线程的方式启动ElevatorsControl类，并且启动整个视图View。



1. **项目实现**

**4.1 domain包下类实现**

**4.1.1Const类实现**

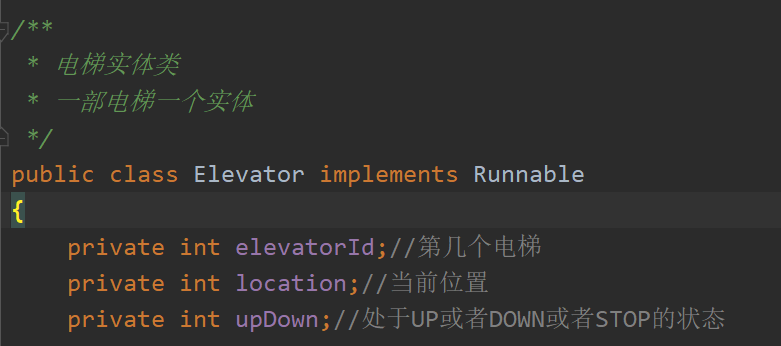
该类存储了项目需要的一些全局变量。例如电梯数量，楼层高度；各种视图的位置大小等，还有用UP，DOWN，STOP来代表电梯状态。



**4.1.2Elevator类实现**

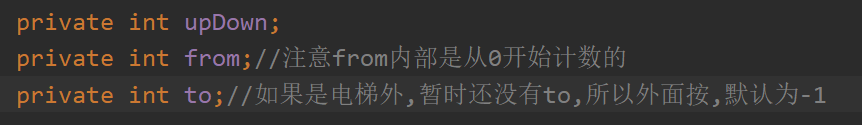
定义了电梯序号，当前该电梯所处位置，以及当前状态，例如UP,DOWN等。而且该类实现了Runnable接口，实现run（）方法，之后调度时就可以以多线程方式启动电梯。

部分代码如下（下同）：



**4.1.3Task类实现**

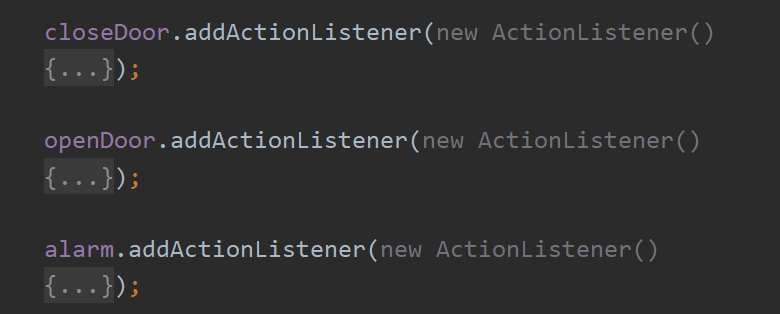
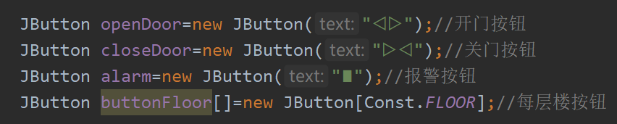
该类主要提供要操作的任务，共有三个变量：需要上行还是下行；当前位置，要去的位置。该类在电梯内外都能使用，当在电梯外时，to即要去的位置设置为-1即可。并且提供set，get方法，以及构造函数等。



**4.2 view包下类实现**

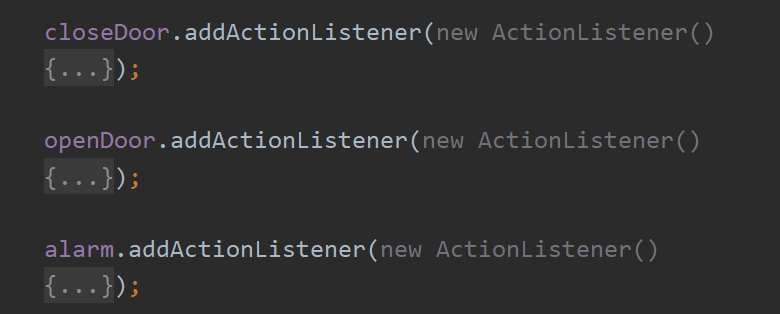
**4.2.1ElevatorView类实现**

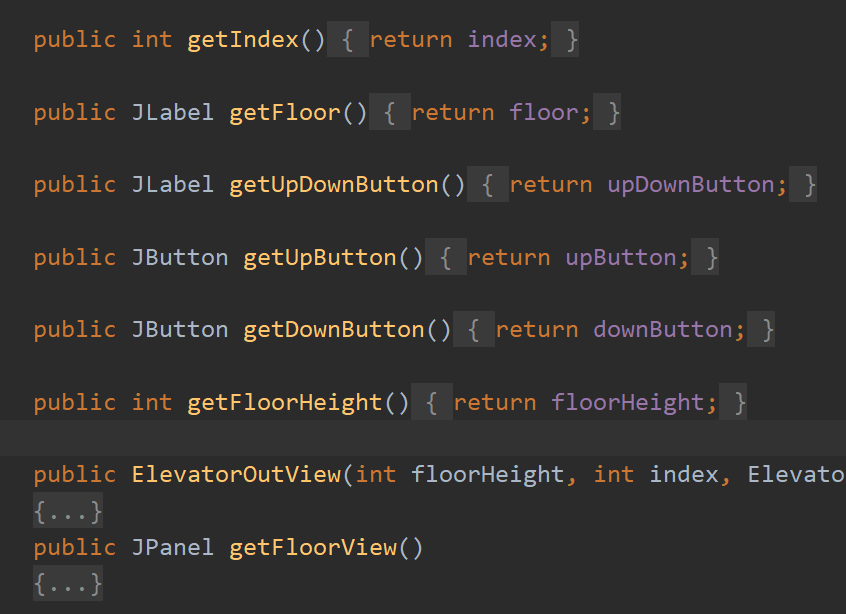
该类通过继承Jpanel类来负责展示电梯内部视图，例如开门关门按钮，要去的楼层按钮等，并且对按钮提供了事件监听。



**4.2.2ElevatorOutView类实现**

该类也继承了Jpanel，负责展示楼层间的电梯外部视图。主要有上升下降按钮，以及显示电梯状态以及位置的文本。还包含了一个ElevatorControl类对象，以为需要得到里面的电梯对象。





**4.2.3View类实现**

该类是视图的最外层，负责展示其他组件，如电梯内部，外部视图等。它通过继承Jframe类实现该功能。他可以调用ElevatorView类与ElevatorOutView类来批量创造电梯内外视图，达到最终效果。





**4.3 control包下类实现**

**4.3.1 ElevatorOutViewControl类实现**

该类主要有两个功能,一是对外部电梯按钮进行事件监听,并且向控制器添加任务, 二是对外部电梯标签进行刷新,以标识电梯目前的位置以及状态。

首先初始化时，会创建20\*5个ElevatorOutView对象，代表每层楼，每个电梯都有一个外部视图。其次，会分别给所有ElevatorOutView对象的开门关门键提供事件监听，当按下按钮时，会在控制器中添加任务，并且同楼层的电梯的相同方向按钮会同时变红，代表被按下。

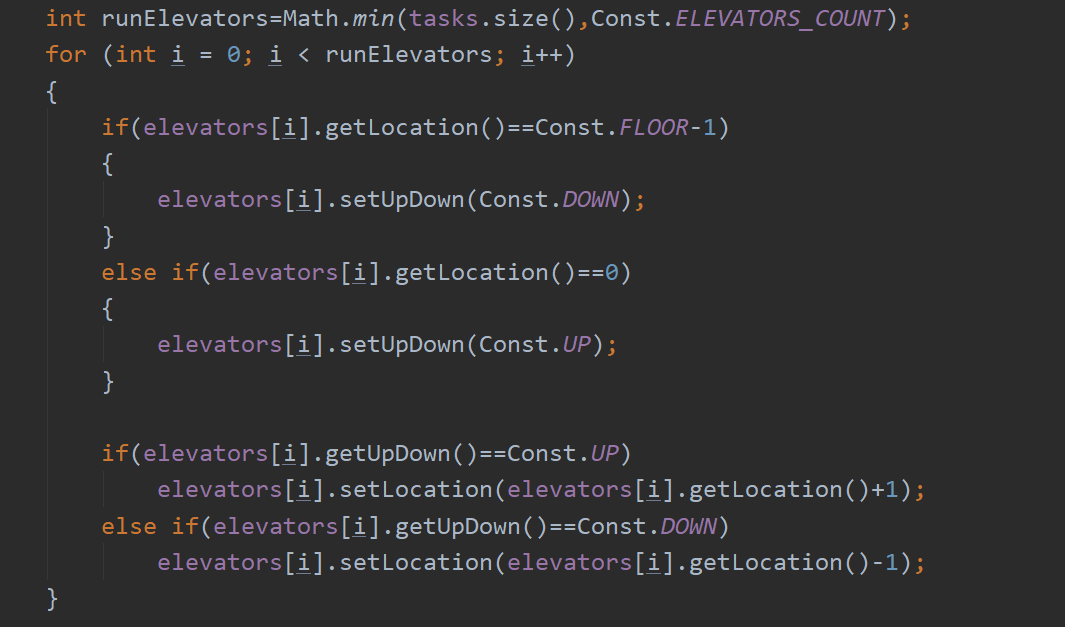
该类也实现了Runnable接口，在run方法中，每隔100ms刷新页面，将最新的电梯状态展示，并且将已完成任务从control类控制列表中删除。





**4.3.2 ElevatorsControl类实现**

负责电梯的实际调度，当没有任务时，电梯将静止不动。有任务时，电梯将持续的往返扫描运行，并且将Elevator对象中的location每秒向上或向下改变1楼（取决于当前电梯运行的状态）。且设置有1个任务则第一个电梯运行，有两个则前两个电梯运行..大于5个任务则所有电梯都不断扫描运行。因为会在ElevatorOutViewControl对象中实时响应任务，所有任务会实时更新，而电梯状态也会实时更新。例如当任务数量从5变为4时，则电梯5将变为静止状态。



1. **项目界面展示**

**5.1总体界面展示**

****

左边为五部电梯的内部视图，灰色标签表明具体是哪部电梯的内部视图，其中按钮则分别是要去的楼层，报警键，开门关门键。

右边则是楼层视图，以行来分析，则一行中，最左边是楼层高度，其次是电梯在该楼层的上行下行按钮以及电梯的状态和电梯目前的位置（因为初始在1楼而且为静止状态，所以显示为∎1）。因为有五个电梯，所以有5个电梯的按钮及状态显示。共有20行，代表20楼，每行为每楼可进行的按钮。

以一列来看，则每一列代表了每个电梯在各个楼层的上下行操作按钮，以及电梯状态，因为为同一个电梯，所以每一列的电梯状态都相同，标志了该电梯目前的状态以及该电梯所处的楼层。因为有五个电梯，所以有五列。

**5.2电梯运行界面展示**



如图为按下了2个上楼,2个下楼按钮的电梯图,可以看出,同楼层按钮是互联结的,并且此时一号电梯处于上升状态,且目前在7楼,其他电梯同理。注意，此时电梯5处于静止状态，并且在一楼。

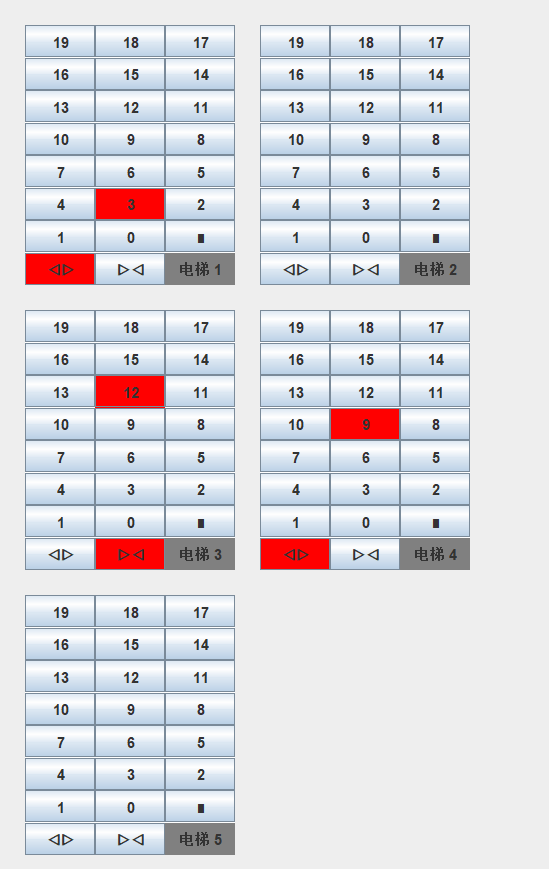


接上图，在电梯上行的时候，处理了楼层8，楼层11的上行请求，所以上行请求消失。



如图，因为电梯1下行，处理完了两个下行请求。且之后没有任务时，就处于静止状态。

* 1. **电梯内部按钮界面**

****

如图，电梯内部按钮都可以按下,如选择哪层楼，开门关门等。

1. **项目实现情况与可行性分析**

**6.1实现情况**

* 实现多线程电梯及实现Scan电梯调度算法。
* 五部电梯门口的按钮是互联结的，实现同时按下。
* 楼层和电梯数可设置。
* 所有电梯初始状态都在第一层。
* 每个电梯如果在它的上层或者下层没有相应请求情况下，则保持不动。
* 每个电梯里面设置必要功能键。
* 每层楼的每部电梯门口，有上行和下行按钮和当前电梯状态的数码显示器。
* 只能在楼层等待电梯时发出任务请求。进入电梯内部后不能操纵电梯内部按钮（如开门，关门，上升下降到某层楼等）。

**6.2可行性分析**

* 该项目基本实现了电梯的调度功能，基本能完成需求
* 没有实现进入电梯内部后的任务请求调度，只能是进入电梯后等待电梯不断上下扫描到达目的地。
* 调度算法不够合理，电梯只会不断上下扫描，响应可能会比较久。例如当所有电梯都处于8楼，而且是上行时。此时如果在7楼新提出了一个上行请求时，电梯会先上升到顶楼，再下到1楼，再上升到7楼才会响应这个请求。但幸运的是，这种情况出现的概率比较小，因为有5辆电梯。他们大概率处于不同的状态，或者是静止状态，所以能很快做出响应。
* 没有考虑到电梯人数超载问题