操作系统第一次课程项目——电梯调度

1552739 软件四班 曹君璐

一、项目简介

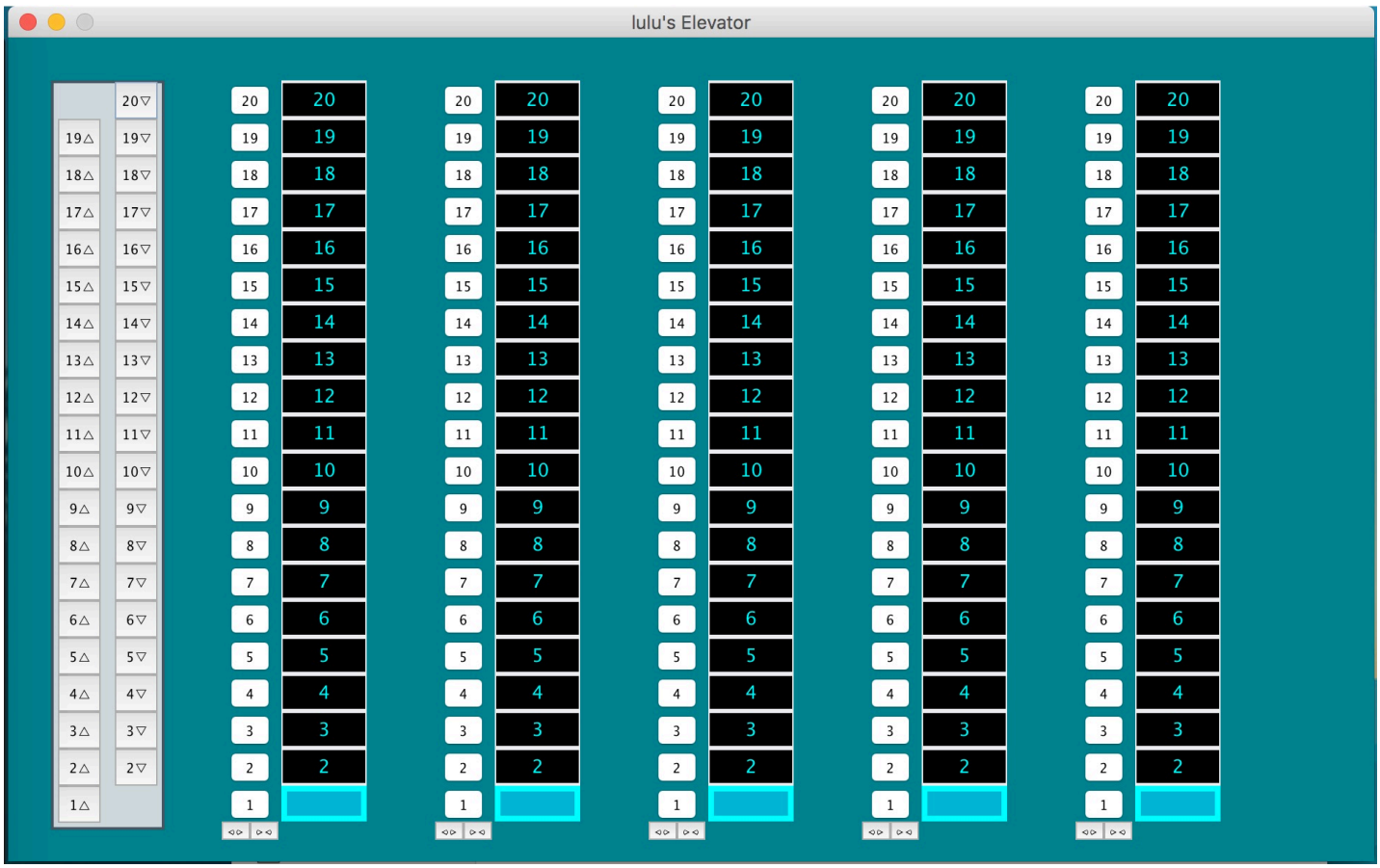
- 通过控制电梯调度，实现操作系统调度过程
- 学习特定环境下多线程编程方法
- 学习调度算法

二、开发环境

- 操作系统平台 `MAC OS`
- 语言 `JAVA`

三、电梯说明

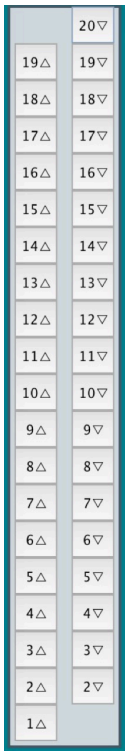
3.1 界面截图



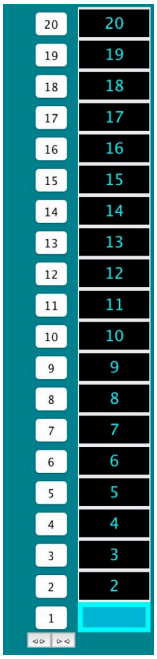
3.2 界面说明

界面主要由一系列 `JLabel` 和 `JButton` 组成。

- **左侧**电梯外部控制键
    - 左边的灰色栏有两列上下键，每行的上下键表示该层电梯外的上下控制键。



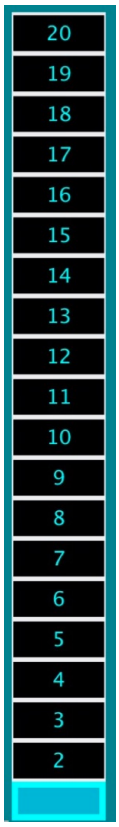
- **右侧**五台电梯内部控制键
    - 右边五列控件对应五台电梯。每台电梯由两栏组成。



- 左边白色栏是电梯内部的数字键，表示该台电梯内部的楼层按钮。

> 白色栏底部是电梯内部的开关键，表示该台电梯内部的开关按钮。

> 右边黑色栏代表电梯通道，每个框内部数字代表当前楼层。



- 黑色栏底部的蓝色矩形代表电梯，其初始位置在一层。

> 内部颜色为蓝色时代表电梯门关闭

> 内部颜色为透明时代表电梯门打开

四、程序设计

4.1 类的设计

`Elavator类`

函数名	功能概述
Elevator	构造函数，构造一台电梯
run	启动电梯
up	使电梯向上移动
down	使电梯向下移动
setDirection	（核心函数）判断电梯接下来的移动方向，返回给state
setArrival	响应电梯的请求按钮，将按钮所在楼层设置为电梯需要移动到达的楼层
getFloor	返回电梯当前楼层
open	打开电梯门
close	关闭电梯门
reopen	重新打开电梯门（延长关门时间）
search	当电梯处于***静止状态***时，判断电梯外部是否有按钮响应
add	把Elevator中的所有控件添加到panel中

`Outside类`

函数名	功能概述
Outside	构造函数，建立电梯外部布局

`Inside类`

函数名	功能概述
Inside	构造函数，建立电梯内部布局
buttonFloorListener	监听电梯内部按键，调用setArrival函数
closeFloorListener	监听关闭电梯门按钮，调用close函数
openFloorListener	监听打开电梯门按钮，调用open函数

`PressButton类`

函数名	功能概述
PressButton	构造函数，建立电梯外部的按钮的布局，绑定相关的监听器
pressButtonListener	监听电梯外部的按钮，对按钮的请求作出响应
isShortest	判断当前电梯是否为所有空闲电梯中，楼层距离最短的电梯，使其响应电梯外部对应的按钮
getIsPress	按照*“先到先得”*的原则，处理电梯外部未被响应的按钮

4.2 重要参数及变量

`Elavator类`

```
private boolean  arrival[];
//记录电梯内部的按钮所要到达的楼层
```

> arrive用于存放电梯内部的按钮发出的请求信息。
>
> > 0~18表示1~19楼的一个向上走的请求
> > 21~38表示20楼到2楼向下走的请求。

```
private int state;
//电梯的状态
```

> 电梯的状态有以下五种
>
> | 数字 | 状态 |
> | --- | --- |
> | 0 | 静止状态 （空闲） |
> | 1 | **向上**移动中，响应**上楼**的请求 |
> | 2 | **向上**移动，响应**下楼**的请求 |
> | -1 | **向下**移动，响应**下楼**的请求 |
> | -2 | **向下**移动，响应**上楼**的请求 |

```
private int willArrive;
//电梯此刻运行方向将要到达的楼层
```

```
private int floor;
//电梯目前所在楼层
```

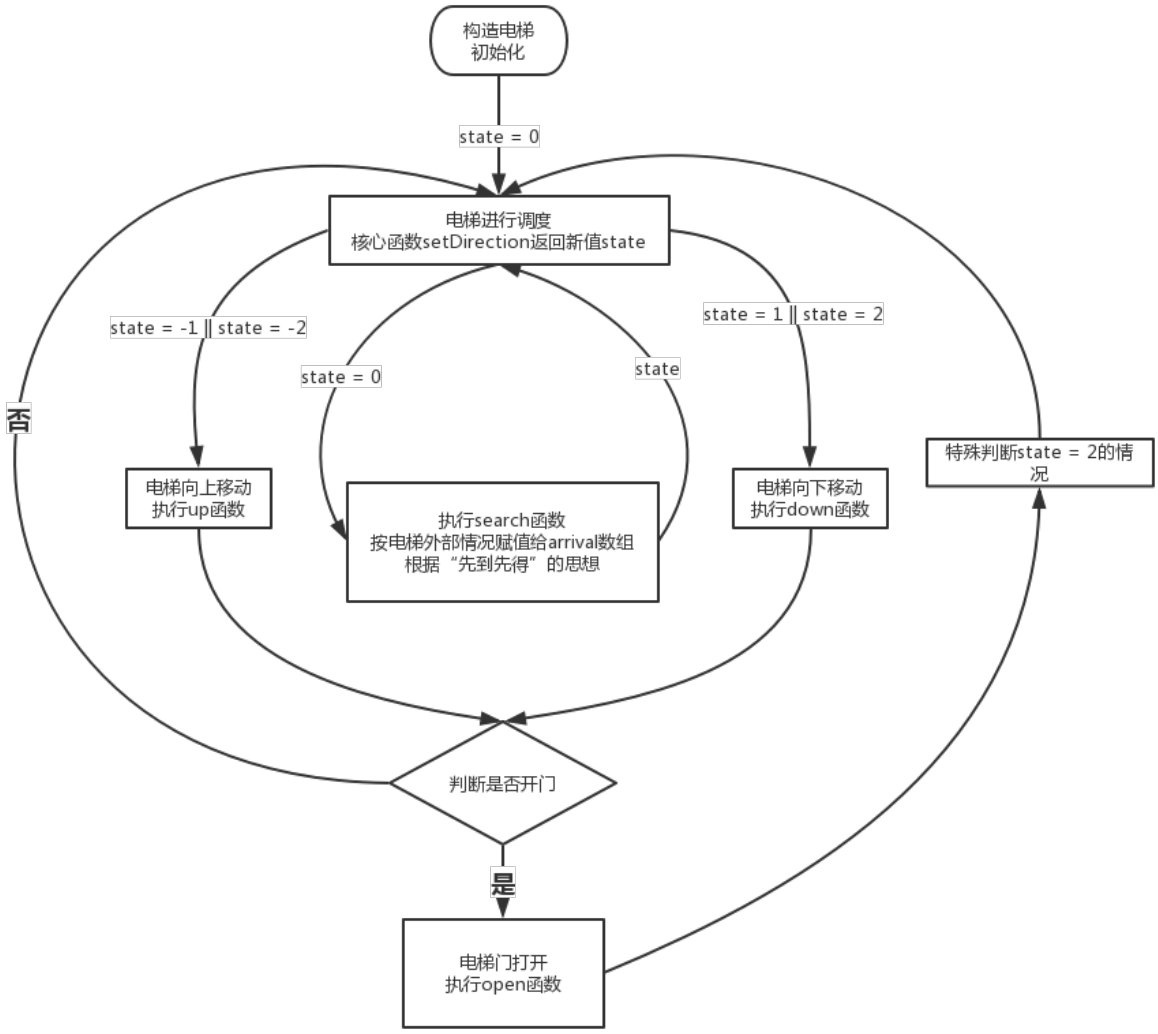
`PressButton类`

```
private boolean require[];
//电梯外部的按钮是否被按下
```

```
public static LinkedList<Integer> taskList;
//记录电梯外部按钮的任务队列
```

> 用链表实现，在电梯方向与请求方向一致的情况下删除任务

4.3 电梯状态流程图



五、实现思路

5.1 一部电梯的内部按钮

> 优先响应电梯内部按钮的请求，只有在顺路时考虑电梯外部按钮的请求。

- `run` 函数中调用 `setDirection` 函数

  `setDirection` 函数会根据上一步的state按照不同的顺序遍历arrival数组，返回值给新的state。

- 电梯内部按钮请求与arrival数组
    - 若按下按钮的层数 > floor(电梯所在楼层), `setDirection` 表示上楼请求
    - 若按下按钮的层数 < floor(电梯所在楼层), `setDirection` 发出下楼请求
    - 若按下按钮的层数 = floor(电梯所在楼层)
        - 当电梯处于开门状态，延长开门时间
        - 当电梯正在向上走，发出向下的请求
        - 当电梯正在向下走，发出向上请求
- 开门与关门
    - `open` 函数运行时，先判断 `isOpen` ，若为真则响应；否则不响应。
    - 在电梯到达楼层之后，电梯门自动打开，持续2s。此时用户可以按下开门键，电梯会执行 `open` ，门再次打开2s。
    - `close` 函数运行时， 先判 `isOpen` 。若门开着，电梯可以立刻关门，向下一层移动。

5.2 一部电梯的外部按钮

> 只有在电梯静止状态下响应外部按钮，除非是电梯“顺路”的情况。

state = 0 时，在 `setDirection` 函数中，首先遍历arrival数组。若全为假，表示电梯内部没有需要到达的楼层，则在search函数中，按照*"先到先得"*的思想，赋值给arrival数组。将isPress数组对应值改为假。

5.3 响应多部电梯

> 使用search函数，协调5部电梯

- 5部电梯是5个线程，线程之间相互工作，互不干涉。五台电梯内部原理与上述一致，创建的时候创建5个Elevator类。
- 电梯外部外部按钮有所不同
    - 使用队列取出最先按下的press楼层之后，将press分配给arrival数组的情况有略微差别。
    - 采用最短调度法: 对于外部响应的电梯，在所有的静止电梯中，找到楼层距离最短的电梯调度过去。

思考总结

6.1 性能分析

- 本调度算法考虑了电梯调度的实时性，在整个系统的动态变化中，每次调度都是选择当前时刻最适合的电梯，这样，就解决了在分配任务后和任务完成之间某时刻又出现更加适合电梯的问题，从而提高了系统的整体性能。
- 电梯的调度采取***“先到先得”***的思想。但这可能会带来一些缺陷，比如某些楼层的请求可能会比别的楼层多，提高它们的优先级会更加的合理。

6.2 心得体会

- 对于操作系统有了更深的理解，用线程模拟进程，每一部电梯是一个进程，而所有的请求是资源，我们做这个电梯的目的是模拟进程调度的问题，也就是模拟所有的请求资源如何分配给电梯，电梯应该先完成哪些资源的情况。
- 通过本程序，我练习了Java中多线程的使用，并加深理解了进程间的调度算法。
- 本程序还有许多不足之处，如电梯外部的按钮在第一次被点击之后形状会由原来的圆角矩形变成方角矩形。