操作系统课程设计

张仕奇 **1152671**

项目名称：电梯调度程序

项目目的：通过编写电梯调度程序深入了解多进程的工作原理和实现方法

编译环境：Java

文件说明：

MyElevator.java:包含主函数

MyElevatorFrame.java:添加一个 panel，然后在上面添加 MyPanel

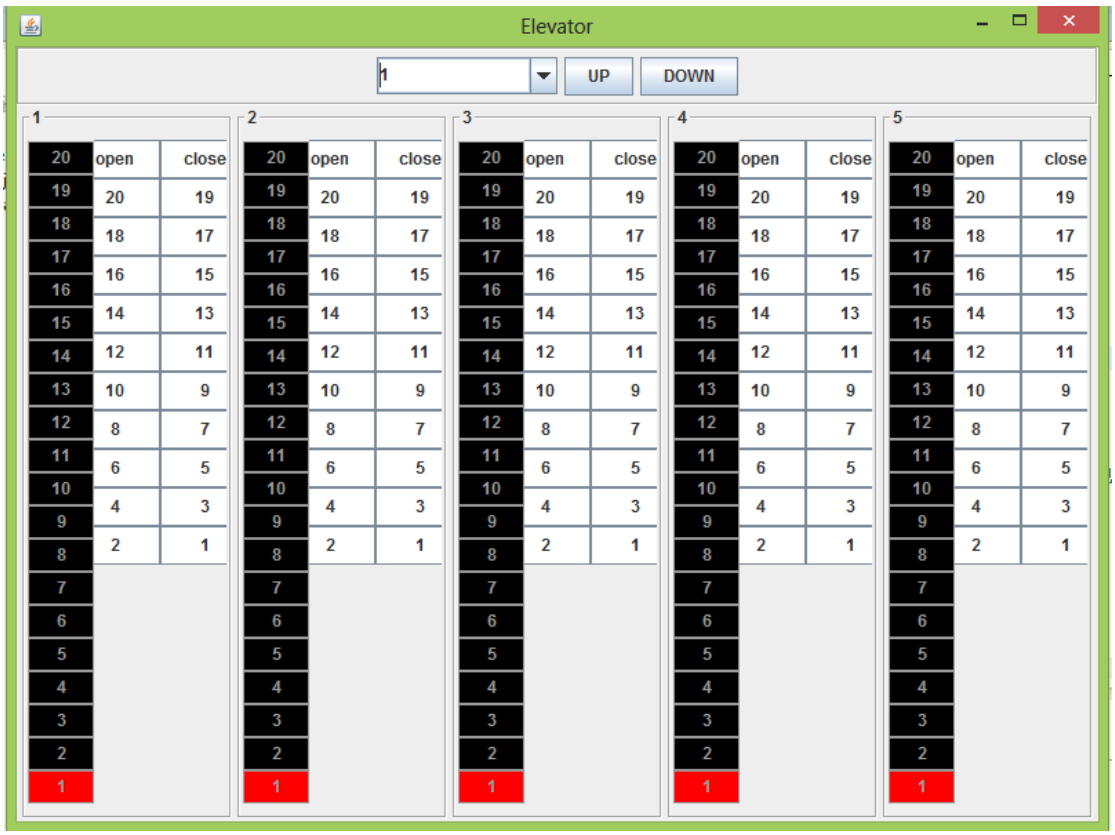
MyPanel.java：用于添加电梯外部按键，和电梯panel

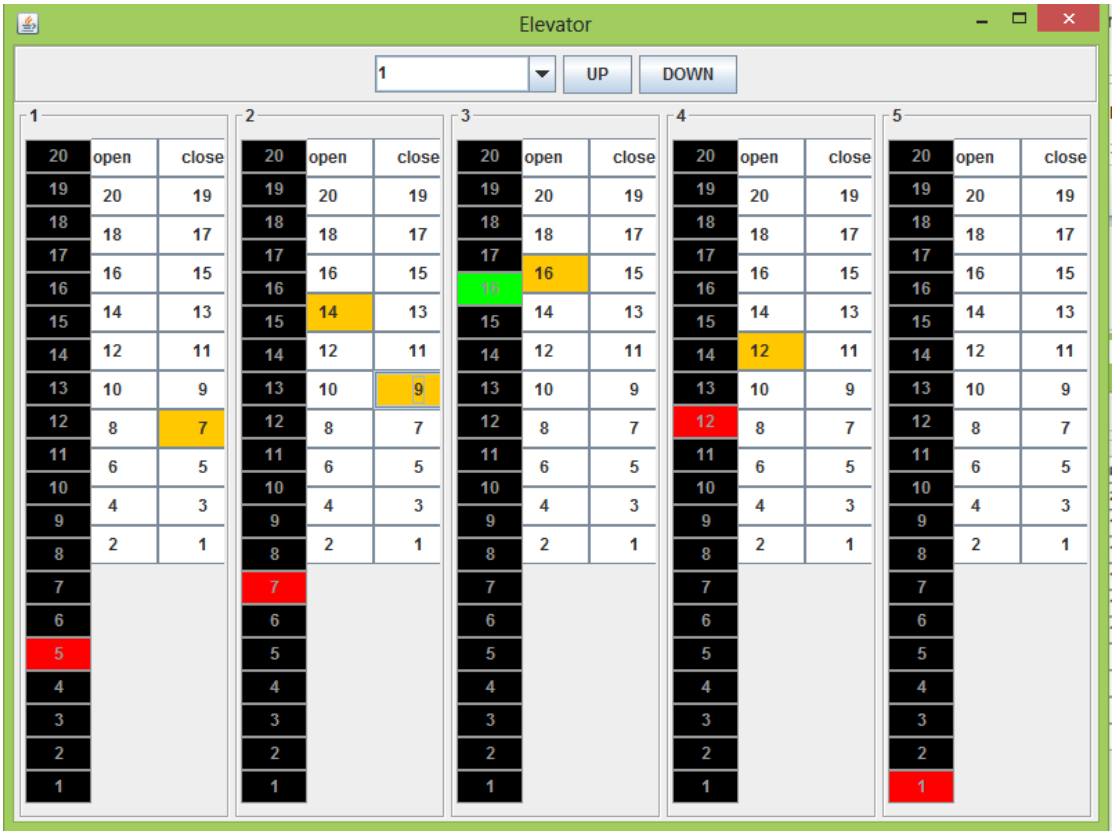
MyMainPanel.java：创建整体的电梯panel

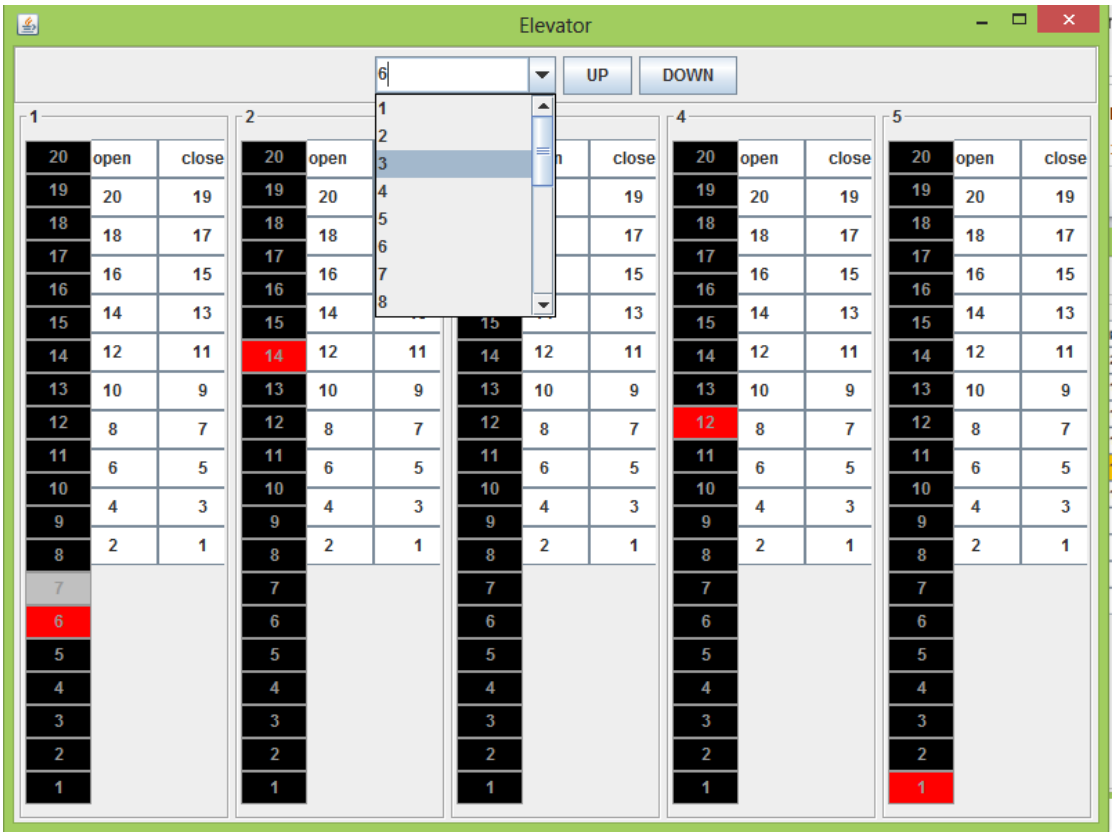
MyElevatorPanel.java：每部电梯的panel

ControlPanel.java：外部控制按钮的panel

程序说明：


Elevator
1
UP
DOWN
1
2
3
4
5
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
open
close
20 19
18 17
16 15
14 13
12 11
10 9
8 7
6 5
4 3
2 1


Elevator
1
UP
DOWN
1
2
3
4
5
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
open
close
20 19
18 17
16 15
14 13
12 11
10 9
8 7
6 5
4 3
2 1



界面如上图

电梯内部具有开关门键以及每层按钮键。电梯初始化在一楼，红色表示静止。电梯运动当中也为红色。当颜色变为绿色时表示电梯开门。按内部按钮的开，关可以控制电梯门的开关。

外部按钮我用下来列表配合 up, down 来展示。

界面设计：

在主 panel 上添加电梯的 panel 和外部上下键控制按键的 panel。在 MyMainPanel 用 setLayout(**new** GridLayout(1, 5));分成一行五列。每一部分放一个电梯。而电梯的每一层的图像通过循环添加 button 来展示。上下键的 panel 放在 Frame 的 north.

调度算法：

主要程序见 MyMainPanel.java , MyElevatorPanel.java.

1. 当外部按钮按下时，会调用 moveTo 函数，moveTo(destFloor,direction),destFlor 为目标楼层，direction 为运动方向，up 为 1，down 为-1. 而该函数会调用用 getSuitableElevator() 方法实现进程的调度，根据五部电梯的状态，寻找离当前楼层，最近的且和请求同方向的电梯给予响应；若方向均不相同，则为最近的电梯。
2. 程序运行，五部电梯五个线程进入 wait 状态，通过对各个按钮进行监听，一旦有请求，作出相应修改（如修改暂停列表等），然后用 getSuitableElevator()方法寻找到相应电梯，

唤醒相应线程，对电梯状态进行改变，每到一个楼层检查对应的暂停列表，若为 true 则停下开门，关门继续运行，到达目标楼层后，再一次检查全部暂停列表，若有 true 继续运行，没有则进入 wait 状态。

3. 电梯运动的代码主要由 public void moveElevatorFloor(boolean move)来实现。

4. getSuitableElevator()代码解析

通过参数来判断运动方向，这里以向上运动为例。通过 for 循环检测 5 部电梯，如果电梯为向下运动则忽略对该电梯的请求。若电梯向上运行，但所在层数高于请求层数，则忽略对该电梯的请求。对于剩下的电梯，计算到电梯现在所在的 floor 与目标楼层之间的最短距离，距离最短的那部电梯为最适合电梯，然后调用 moveTo 函数将电梯移动到目标电梯相应的层数。

getSuitableElevator()代码：（附有注释）

```
private int getSuitableElevator(int destFloor,int direction)
    {
        int []distance = new int[]{21,21,21,21,21};//将距离初始为21，以防出错
        int minDistance = 21;
        int suitableElevator = -1;
        if(direction==1)//请求方向向上
        {
            for(int i=0;i<panel.length;i++)
            {
                if(panel[i].runningStatus==-1)//若电梯正在向下运行，忽略请求
                    distance[i] = 21;
                else if(panel[i].runningStatus == 1)
                {
                    if(panel[i].currentFloor>=destFloor)//若电梯向上运行，但所在层数高于请求层数，忽略请求
                    {
                        distance[i] = 21;
                    }
                    else
                        distance[i] = Math.abs(panel[i].currentFloor-destFloor);
                }
                else
                    distance[i] = Math.abs(panel[i].currentFloor-destFloor);
            }
        }
```

```
        else if (direction==-1)//请求方向向下
        {
            for(int i=0;i<panel.length;i++)
            {
                if(panel[i].runningStatus==1) //若电梯正在向上运行，忽略请求
                    distance[i] = 21;
                else if(panel[i].runningStatus == -1)
                {
                    if(panel[i].currentFloor<=destFloor)//若电梯向下运行，
但所在层数低于请求层数，忽略请求
                    {
                        distance[i] =panel[i].currentFloor - destFloor;
                    }
                    else
                        distance[i] =
Math.abs(panel[i].currentFloor-destFloor);
                }
                else
                    distance[i] =
Math.abs(panel[i].currentFloor-destFloor);
            }
        }
        //若所有电梯都在向同一个方向运行
        if((panel[0].runningStatus==1&&panel[1].runningStatus==1

    &&panel[2].runningStatus==1&&panel[3].runningStatus==1&&panel[4].
runningStatus==1)||
            (panel[0].runningStatus==-1&&panel[1].runningStatus==-1

    &&panel[2].runningStatus==-1&&panel[3].runningStatus==-1&&panel[4
].runningStatus==-1))
        {
            for(int i=0;i<panel.length;i++)
            {
                distance[i] = Math.abs(panel[i].currentFloor-destFloor);
            }
        }
        //计算最小距离
        for(int i=0;i<5;i++)
        {
            if(minDistance>distance[i])
            {
                minDistance = distance[i];
                suitableElevator = i;
```

```
            }
        }
        if(minDistance == 21)
            return -1;
        else
            return suitableElevator;
    }
```

项目总结：

通过这个程序，体会并理解了多线程概念，学习 l 特定环境下多线程编程方法， 体会了电梯调度思想

有的时候觉得写代码之前需要先详细地设计一下，但是没有写的话又不知道问题在哪里。但是如果不设计就写吧，又很容易出问题。所以这就成了一种矛盾。

在我和同学的通论过程中，各种调度方案层出不穷。每个人似乎都有每个人的调度方案，又对其他人的方案不太认同。在我看来，要找到一个实践中完美的方案，可能不是这两三个星期能解决的问题。所以我感觉，只要调度得正确，不存在接不到人、送不到位、开不了门之类的情况，应该都还是可以接受的调度算法。

最后一点，自我感觉我的界面不太友好，希望以后继续努力。