进程管理 - 电梯调度_设计方案报告

学号	姓名	课号
1852137	张艺腾	42036901

进程管理 - 电梯调度 设计方案报告

```
项目需求
  功能描述
开发环境
项目结构
操作说明
系统分析
系统设计
  界面设计
    1.整体设计
     2.组件设计
  状态设计
  类设计
     1.MyButton类: 电梯按钮
     2.Buttons类: 电梯内按钮组
     3.Elevator类: 电梯类
     4.EventListener类: 实现事件监听接口
     5.Floor类: 楼层类
     6.MyBuilding类:大楼类(主类)
系统实现
  内命令处理
  外命令处理
  动画实现
项目功能截屏展示
作者
```

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。

功能描述

- 1. 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。
- 2. 每层楼应该有上行和下行按钮和当前电梯状态的数码显示器
- 3. 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**。

开发环境

开发环境: Windows 10开发软件: Eclipse

• 开发语言: JavaSE (jdk1.8.0_241)

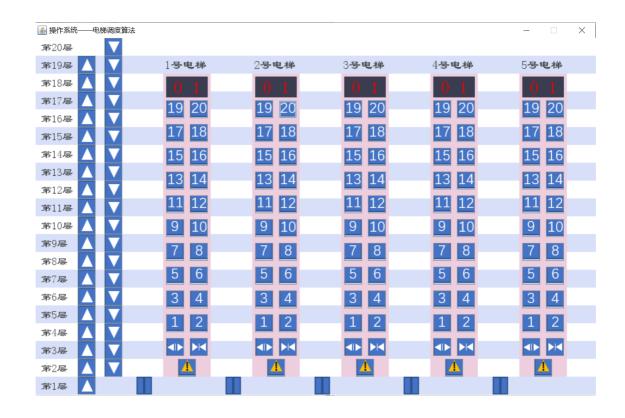
• **开发工具包**: Swing

项目结构

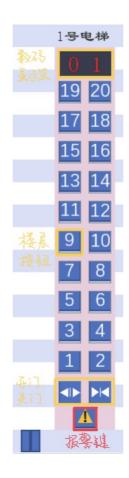
```
elevator.jar
 2
       README.md
 3
       电梯调度_设计方案报告.md
 4
 5
    ∟src
       ├-component
 6
 7
            Buttons.java
 8
            Elevator.java
 9
            EventListener.java
            Floor.java
10
11
            MyButton.java
12
       └─UI
13
            MyBuilding.java
14
       ∟image
                     (0<i<21)
15
            i.png
16
            ih.png (0<i<21)
17
            iA.png (0<i<21)
18
            door.png
19
            open.png
20
            openH.png
21
            openA.png
22
            close.png
23
            closeH.png
24
            closeA.png
25
            alarm.png
26
            alarmH.png
27
            up.png
28
            upH.png
29
            down.png
30
            downH.pngs
```

操作说明

- 在文件夹内双击运行 elevator.jar,进入电梯模拟系统如下图
 - 一开始电梯都停在1层,数码显示器显示01

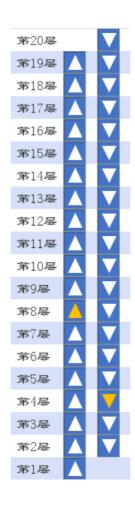


• 点击每部电梯的**功能键**(开/关键,报警器,楼层按钮),进行单部电梯内命令处理模拟



• 点击左侧楼层上下按钮,进行多部电梯外命令处理模拟。

注:由于空间有限无法给每一个电梯每一层都做上下按钮,故只在每层设置一组上下按钮,**表示该层有上行或下行请求**。其中20层不能继续上行、1层不能继续下行,故不设对应按钮。



系统分析

• 单部电梯内命令处理

- 内部事件及对应响应:
 - 用户点击楼层按钮
 - 若按键楼层与电梯当前的楼层数相同→该电梯开门,1秒后自动关门
 - 若按键楼层与电梯当前楼层数不同→将对应楼层的请求设为true,并进行调度
 - 若电梯正在上行且按键楼层在电梯当前楼层之上→电梯到达该层时停靠、开 门、1秒后自动关门
 - 若电梯正在下行且按键楼层在电梯当前楼层之下→电梯到达该层时停靠、开 门、1秒后自动关门
 - 若电梯正在上行而按键楼层在电梯当前楼层之下→电梯继续上行,完成此前 的任务后下行到达该层、开门、1秒后关门
 - 若电梯正在下行而按键楼层在电梯当前楼层之上→电梯继续下行,完成此前 的任务后上行到达该层、开门、1秒后关门
 - 用户点击开关按钮
 - 若电梯正在运行,不响应此请求
 - 若电梯处于静止状态,开门、1秒后关门
 - 用户点击报警按钮
 - 电梯立刻停止运行,数码显示器显示"ERR!"
 - 1秒后该电梯所有内部按键变红且失效

• 多部电梯外命令处理

○ 外部事件及对应响应:

- 用户按下某一楼层的上/下行按钮 (每层设置一个上/下行按钮)
 - a. 筛选处于正常状态的电梯, 计算每部电梯到达此层的路程:
 - 静止态的电梯:

当前楼层和请求楼层之间的距离即为路程(每层层高为30个像素点)

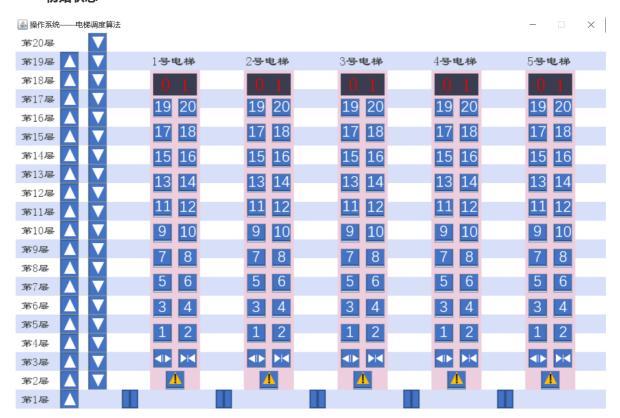
- 上行电梯:
 - 若请求楼层在电梯当前所在楼层之上,电梯当前楼层和请求楼层之间的距离 即为路程
 - 若请求楼层在电梯当前坐在楼层之下 路程=电梯上行到达任务中最远楼层的距离*2+当前楼层和请求楼层之间的距 离
- 下行电梯:
 - 若请求楼层在电梯当前所在楼层之下,电梯当前楼层和请求楼层之间的距离 即为路程
 - 若请求楼层在电梯当前坐在楼层之上 路程=电梯下行到达任务中最远楼层的距离*2+当前楼层和请求楼层之间的距 离
- b. 选出所需路程最短的电梯,将此请求加入该电梯的任务序列中等待响应

系统设计

界面设计

1.整体设计

• 初始状态



• 运行状态



2.组件设计

• 窗体模型: Java. Swing. JFrame

• **电梯模型:** Elevator 继承面板父类 Java. Swing. JPanel 实现线程接口 Java. lang. Runnable

- 电梯内部按钮 component. Buttons
 - 按钮后的粉底背板 Java. Swing. JLabel
 - 各类按钮 component. MyButton
 - 电梯状态的"数码显示器" Java. Swing. JLabel
- 电梯门 Java. Swing. JLabel
- 电梯文字 Java. Swing. JLabel
- 楼层模型: component. Floor
 - 楼层数字 Java. Swing. JLabel
 - 楼层背景 Java. Swing. JLabel
 - 楼层上下按钮 component. MyButton
- 大楼模型: UI. MyBuilding
 - 电梯*5
 - 楼层*20

状态设计

- 电梯状态
 - o IsRun = false # 静止状态
 - o IsRun = true # 运行状态
 - o IsUp = true # 上行
 - o IsDown = true # 下行
- 电梯门状态

- Isopen = true # 开门状态
- o Isopen = false # 关门状态

• 楼层状态

- whichFloorIsWaitUp[i] = true # 第 i 层请求上行
- o whichFloorIsWaitDown[i] = true # 第 i 层请求下行

类设计

1.MyButton类: 电梯按钮

MyButton

+ num tot : static int

+ num:int

+ btn: JButton

MyButton(String path_nor, String path_high, int x, int y, int width, int height)

2.Buttons类: 电梯内按钮组

Buttons

+ buttons : MyButton[]

+ btnOpen : MyButton

+ btnClose : MyButton

+ btnAlarm : MyButton

+ floorNum : JLabel

+ Buttons()

+ setFloor(int floor)

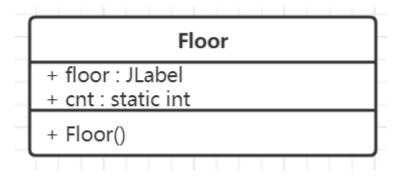
3.Elevator类: 电梯类

Elevator + buttons : Buttons + doorL: final Image + doorL : final Image + label : JLabel + IsUp : boolean + IsDown : boolean + IsOpen: boolean + IsRun : boolean + alarm: boolean + toWhichFloor: boolean[20] + cnt : static int - x : int - y : int + Elevator() + run() + moveToFloor(int floor) - openDoor() - closeDoor() - alarming() + dispatch() + getMyY(): int + getCurrentFloor(): int + getFarthest(): int

4.EventListener类: 实现事件监听接口

```
public class EventListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
       System.out.println("I'm listening!");
   }
}
```

5.Floor类: 楼层类



6.MyBuilding类: 大楼类 (主类)

MyBuilding

- + whichFloorIsWaitUp : boolean[20]
- + whichFloorIsWaitDown : boolean[20]
- + elevators : Elevator[5]
- + upButtons : static MyButton[20]
- + downButtons : static MyButton[20]
- + dispatchAlgorithm(int callFloor, int up): int
- + MyBuilding()
- + main(String args[]) : static void

系统实现

内命令处理

1. 报警器

- 。 用户点击报警按钮→对应电梯立即停止运行→数码显示器显示"ERR!"
- 。 该电梯所有内部按钮变红,并且无法再按下
- 禁用该电梯,外部调度将忽略此电梯

```
buttons.btnAlarm.btn.addActionListener(new EventListener() {
   public void actionPerformed(ActionEvent e) {
      alarm = true;
      buttons.setFloor(-1);
      buttons.btnAlarm.btn.setEnabled(false);
      alarming();
   }
}
```

```
//alarming
2
        private void alarming() {
            if (alarm) {
3
                for (int i = 0; i < 20; i++) {
 4
                    buttons.buttons[i].btn.setDisabledIcon(new
    ImageIcon(this.getClass().getResource("/image/" + (i + 1) + "A.png")));
6
7
                buttons.btnOpen.btn.setDisabledIcon(new
    ImageIcon(this.getClass().getResource("/image/openA.png")));
                buttons.btnClose.btn.setDisabledIcon(new
8
    ImageIcon(this.getClass().getResource("/image/closeA.png")));
9
10
                for (int i = 0; i < 20; i++) {
                    buttons.buttons[i].btn.setEnabled(false);
11
12
                buttons.btnOpen.btn.setEnabled(false);
13
                buttons.btnClose.btn.setEnabled(false);
14
            }
15
```

2. 楼层按键

- 将 towhichFloor[] 中对应楼层设为true
- o 改变该按钮的 enable 状态为 false, 按钮显示 false 状态时的图标
- 。 如果按钮楼层大于电梯当前楼层:
 - IsRun = true
 - Isup = true →上行过程中到该层时停下、开门、1秒后自动关门
 - IsDown = true → 下行至最远请求处后返回该楼层、开门、1秒后关门
 - IsRun = false
 - 电梯启动,运行至该层、开门、1秒后关门
- 如果按钮楼层小于电梯当前楼层
 - IsRun = true
 - IsDown = true →下行过程中到该层时停下、开门、1秒后自动关门
 - IsUp = true→上行至最远请求处后返回该楼层、开门、1秒后关门
 - IsRun = false
 - 电梯启动,运行至该层、开门、1秒后关门
- 。 如果按钮楼层刚好等于电梯当前楼层
 - IsRun = false
 - 开门、1秒后关门
 - 恢复按钮 enable 为 true ,显示正常状态图标,并且可按下
 - IsRun = true
 - 停下、开门、1秒后关门
 - 恢复按钮 enable 为 true ,显示正常状态图标,并且可按下

```
//add event listener to the button
1
2
    for (int i = 0; i < 20; i++) {
3
                MyButton btn = buttons.buttons[i];
                btn.btn.addActionListener(new EventListener() {
4
                    public void actionPerformed(ActionEvent e) {
 5
6
                        btn.btn.setEnabled(false);
                        toWhichFloor[btn.num] = true;
 7
8
                    }
9
                });
10
            }
```

```
// lonely dispatch this elevator
1
    public void dispatch() {
2
 3
        while ((!IsOpen) && (!alarm)) {
4
            buttons.btnOpen.btn.setEnabled(true);
5
            for (int i = 0; i < 20; i++) {
6
                if (toWhichFloor[i]) {
 7
                    moveToFloor(i + 1);
8
                    openDoor();
9
                    buttons.buttons[i].btn.setEnabled(true);
10
                    try {// wait for a second
11
                        Thread.sleep(2000);
12
                    } catch (InterruptedException e1) {
                         // TODO Auto-generated catch block
13
```

```
14
                         e1.printStackTrace();
15
                     }
16
                     closeDoor();
                 }
17
18
             }
19
            try {
20
                 Thread.sleep(50);
             } catch (InterruptedException e) {
21
22
                 e.printStackTrace();
23
24
        }
25
        try {
26
             Thread.sleep(2000);
27
        } catch (InterruptedException e) {
28
                 e.printStackTrace();
29
        }
30
        closeDoor();
31 }
```

3. **开/关门**

- 。 开门
 - 电梯 IsRun = true → 忽略该请求
 - 电梯 IsRun = false → 开门、1秒后自动关门

```
// add event listener to open button
buttons.btnOpen.btn.addActionListener(new EventListener() {
   public void actionPerformed(ActionEvent e) {
      buttons.btnOpen.btn.setEnabled(false);
      openDoor();
   }
};
```

```
1
    // open the door when it's allowed
 2
    private void openDoor() {
 3
        IsOpen = true;
        buttons.btnClose.btn.setEnabled(true);
 4
 5
        while (x > 15) {
 6
             if (IsRun) {
 7
                 break;
 8
             }
9
             x \rightarrow 1;
10
             this.repaint();
11
             try {
                 Thread.sleep(10);
12
             } catch (InterruptedException e) {
13
14
                 e.printStackTrace();
15
             }
16
        }
    }
17
```

```
// add event listener to close button
buttons.btnClose.btn.addActionListener(new EventListener() {
   public void actionPerformed(ActionEvent e) {
      buttons.btnClose.btn.setEnabled(false);
      closeDoor();
   }
}
```

```
//close the door
2
    private void closeDoor() {
 3
        while (x < 20) {
4
            x += 1;
 5
            this.repaint();
 6
            try {
 7
                Thread.sleep(10);
8
            } catch (InterruptedException e) {
9
                e.printStackTrace();
10
            }
11
        }
12
        IsOpen = false;
13
   }
```

外命令处理

1. 楼层上下行按钮

- 。 用户按下楼层中的上下按钮
- o whichFloorIsWaitUp[] 和 whichFloorIsWaitDown[] 对应楼层设为true表示该层有上行/下行请求

```
1 // add event listener for up button
2
   button.btn.addActionListener(new EventListener() {
3
      public void actionPerformed(ActionEvent e) {
           button.btn.setEnabled(false);
4
5
           whichFloorIsWaitUp[(button.num + 1)] = true;
6
           System.out.println((button.num + 1) + "层请求上行");
           elevators[dispatchAlgorithm(button.num +
   1)].toWhichFloor[button.num] = true;
8
       }
9
   });
```

```
1
    // add event listener for down button
2
    button.btn.addActionListener(new EventListener() {
3
        public void actionPerformed(ActionEvent e) {
            button.btn.setEnabled(false);
 4
 5
            System.out.println(button.num);
            whichFloorIsWaitDown[button.num - 17] = true;
6
 7
            System.out.println((button.num - 17) + "层请求下行");
8
            elevators[dispatchAlgorithm(button.num -
    17)].toWhichFloor[button.num - 18] = true;
9
        }
10
   });
```

2. 电梯调度

遍历五部电梯,找出在当前任务数情况下,到达此层**路程最短**的电梯,将此请求加入该电梯的请求 序列中

```
// dispatch the elevators
    public int dispatchAlgorithm(int callFloor, int up) {
 3
        int elevatorNum = 0;
 4
        int wayLenth[] = \{ 0, 0, 0, 0, 0 \};
 5
        for (int j = 0; j < 5; j++) {
            int currentY = elevators[j].getMyY();
 6
            if (elevators[j].alarm) {
                wayLenth[j] += 2000;
 8
 9
            } else {
                if (elevators[j].IsDown) {
10
11
                    wayLenth[j] += elevators[j].getFarthest() * 2;
12
                     // Firstly add the distance it takes to
                     // run to the farthest destination
13
14
                     // and then back to the current level.
15
                     wayLenth[j] += currentY - (600 - 30 * callFloor);
16
                     // Plus the journey from the current layer to this
    layer.
17
                } else if (elevators[j].IsUp) {
                     wayLenth[j] += elevators[j].getFarthest() * 2;
18
19
                     // Firstly add the distance it takes to
20
                     // run to the farthest destination
21
                     // and then back to the current level.
22
                     wayLenth[j] += (600 - 30 * callFloor) - currentY;
23
                     // Plus the journey from the current layer to this
    layer.
24
                } else {
25
                     if (600 - 30 * callFloor >= currenty) {
26
                         wayLenth[j] += (600 - 30 * callFloor) - currentY;
27
                     } else {
                         wayLenth[j] += currentY - (600 - 30 * callFloor);
28
29
                     }
30
                if ((up == 1) && (this.whichFloorIsWaitDown[callFloor])
31
32
                     && (elevators[j].toWhichFloor[callFloor - 1])) {
33
                         wayLenth[j] += 2000;
                }else if ((up == 0) && (this.whichFloorIsWaitUp[callFloor])
34
35
                           && (elevators[j].toWhichFloor[callFloor - 1])) {
                     wayLenth[j] += 2000;
36
37
38
                 if (wayLenth[j] < wayLenth[elevatorNum])</pre>
39
                     elevatorNum = j;
40
            }
41
        }
42
        return elevatorNum;
43 }
```

动画实现

1. 开门动画

```
1 private void openDoor() {
```

```
2
        IsOpen = true;
 3
        buttons.btnClose.btn.setEnabled(true);
 4
        while (x > 15) {
 5
        if (IsRun) {
 6
            break;
 7
        }
 8
        x -= 1;
 9
        this.repaint();
10
        try {
11
            Thread.sleep(10);
12
        } catch (InterruptedException e) {
13
            e.printStackTrace();
14
            }
15
        }
16 }
```

2. 关门动画

```
private void closeDoor() {
 2
        while (x < 20) {
 3
            x += 1;
            this.repaint();
 4
 5
            try {
 6
                Thread.sleep(10);
 7
            } catch (InterruptedException e) {
 8
                e.printStackTrace();
 9
            }
10
        }
        IsOpen = false;
11
12 }
```

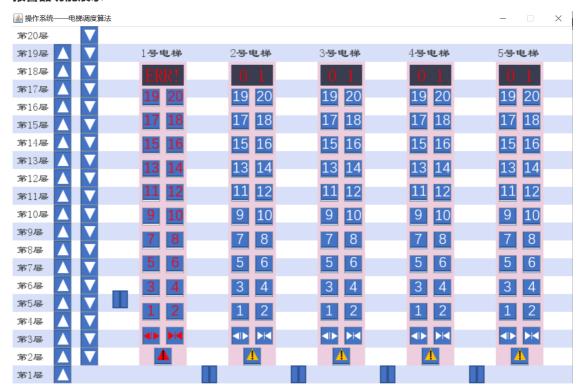
3. 电梯运行动画

```
1
    while (y > 600 - floor * 30) {
2
       IsUp = true;
3
       IsDown = false;
4
       IsRun = true;
5
       if (alarm) {
6
           return;
7
       }
       y = 1;
8
9
       this.repaint();
       //此处还要检测是否上行过程中经过某些需要停靠的楼层
10
11
       //但这个部分主要说动画就不将代码放在这里了
12
       try {
13
           Thread.sleep(20);
           } catch (InterruptedException e) {
14
15
               e.printStackTrace();
16
           }
17
18
    while (y < 600 - floor * 30) {
19
       IsDown = true;
```

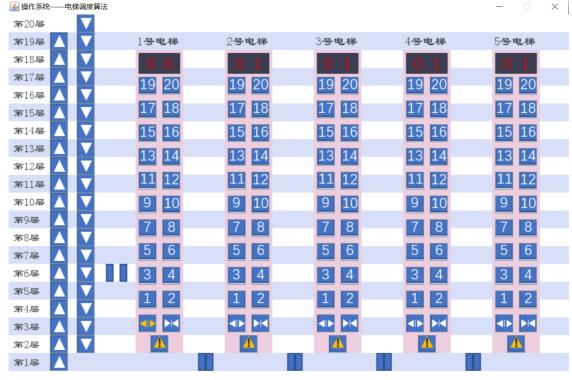
```
20
        IsUp = false;
        IsRun = true;
21
22
        if (alarm) {
23
        return;
24
25
        y += 1;
26
        this.repaint();
27
        try {
28
        Thread.sleep(20);
29
        } catch (InterruptedException e) {
30
             e.printStackTrace();
31
        }
32
    }
33
34
```

项目功能截屏展示

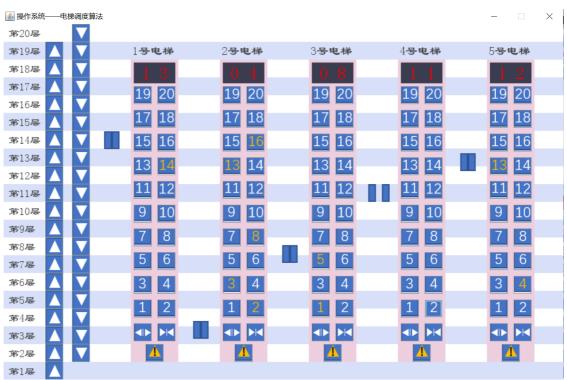
• 报警器功能展示



• 开/关门功能展示



• 内部指令处理与多线程



• 外部指令处理与电梯调度



作者

学号: 1852137

姓名: 张艺腾

指导老师: 张慧娟

课号: 42036901

联系方式: emial:kerr99801@gmail.com