```java
package semaforo.control.simple;

import java.time.LocalTime;


public class OneWaySemaphoreControl implements SemaphoreControl {

    private List<TrafficLightControl> trafficLights = new ArrayList<>();

    private int greenMillis = 10_000;
    private int yellowMillis = 2_000;
    private int redMillis = 5_000;

    private LocalTime alertStart = LocalTime.of(0, 0);
    private LocalTime alertEnd = LocalTime.of(5, 30);

    private OnOff state = OnOff.OFF;

    public OneWaySemaphoreControl(List<TrafficLightControl> trafficLights) {

        this.trafficLights = trafficLights;
    }

    public OneWaySemaphoreControl(TrafficLightControl...trafficLights) {

        this(Arrays.asList(trafficLights));
    }

    private boolean isAlertPeriod() {

        boolean START_MIDNIGHT_END = alertStart.isAfter(alertEnd);

        LocalTime now = LocalTime.now();

        if(START_MIDNIGHT_END)
            return (now.isAfter(alertStart) || now.isBefore(alertEnd));

        return (now.isAfter(alertStart) && now.isBefore(alertEnd));
    }

    private void doAlert() throws InterruptedException {

        while(isAlertPeriod()) {

            trafficLights.forEach(e->e.turnAlert());
            Thread.sleep(1_000);
        }
    }

    private void doYellowRedGreen() throws InterruptedException {

        trafficLights.forEach(e->e.turnYellow());
        Thread.sleep(yellowMillis);

        trafficLights.forEach(e->e.turnRed());
        Thread.sleep(redMillis);

        trafficLights.forEach(e->e.turnGreen());
        Thread.sleep(greenMillis);
    }

    private void run() {
```

```
68
69          Runnable runnable = ()->{
70
71              while (state == OnOff.ON) {
72
73                  try {
74
75                      doAlert();
76                      doYellowRedGreen();
77                  }
78                  catch(InterruptedException exception) {
79
80                      trafficLights.forEach(e->e.turnAlert());
81                  }
82              }
83          };
84          Thread thread = new Thread(runnable);
85          thread.start();
86      }
87
88      @Override
89      public void turnOn() {
90          if (state == OnOff.ON) return;
91          state = OnOff.ON;
92          run();
93      }
94
95      @Override
96      public void turnOff() {
97          state = OnOff.OFF;
98
99          trafficLights.forEach(e->e.turnAlert());
100     }
101
102     @Override
103     public boolean isOn() {
104         return state == OnOff.ON;
105     }
106
107     @Override
108     public boolean isOff() {
109         return state == OnOff.OFF;
110     }
111
112     @Override
113     public void setGreenSeconds(int seconds) {
114         this.greenMillis = seconds * 1000;
115     }
116
117     @Override
118     public void setYellowSeconds(int seconds) {
119         this.yellowMillis = seconds * 1000;
120     }
121
122     @Override
123     public void setRedSeconds(int seconds) {
124         this.redMillis = seconds * 1000;
125     }
126
127     @Override
128     public void setAlertPeriod(LocalTime start, LocalTime end) {
129         this.alertStart = start;
```

```
130         this.alertEnd = end;
131     }
132
133 }
134
```