El horno microondas

INFORME PROYECTO 1

Nerea Martín Serrano | 3º Ingeniería de la Salud

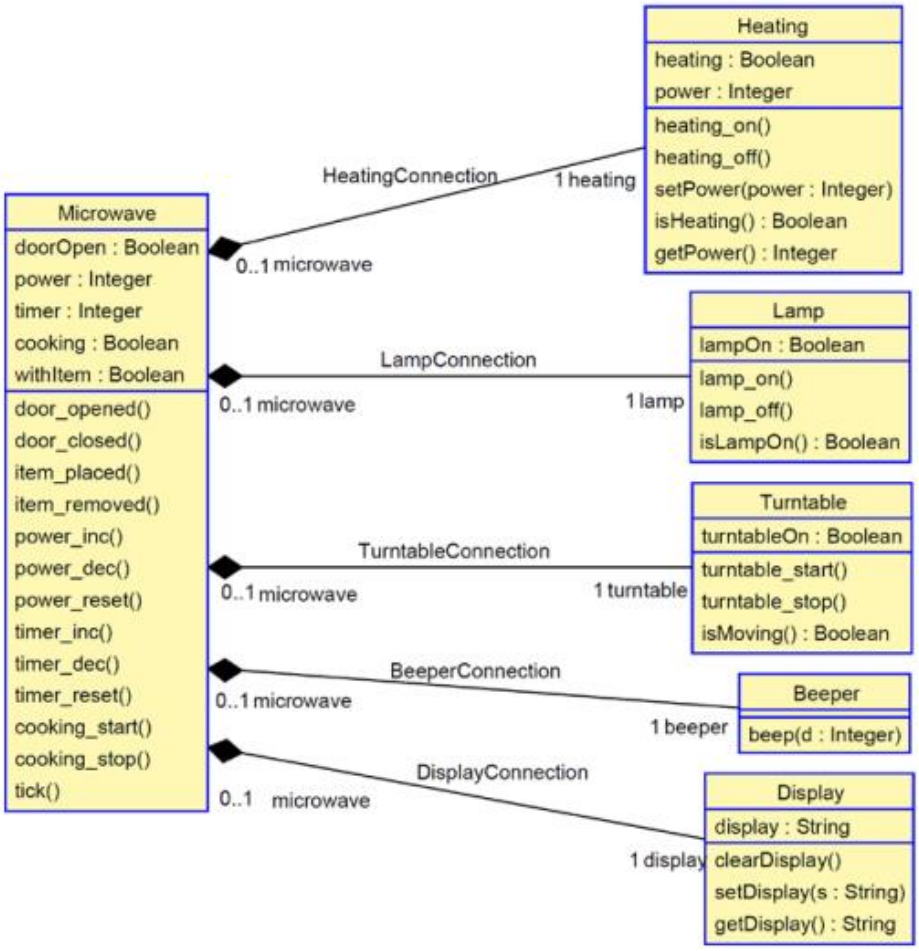
Índice

Introducción....1
Implementación en Java....2
Clase principal....2
Clase microondas....2
Componentes....3
Clase Heating....3
Clase Display....4
Clase Lamp....5
Clase Turnable....5
Patrón Estado....5
Clase abstracta Estado....5
Estado ClosedWithNoItem....7
Estado OpenWithNotItem....8
Estado OpenWithItem....9
Estado ClosedWithItem....10
Estado Cooking....11

Este proyecto se aloja en el siguiente repositorio de git: https://github.com/nmartinse/Microwave.git

Introducción

En este proyecto se va a crear un microondas siguiendo el siguiente diagrama de UML:



La clase principal va a ser **Microwave**, que es la que se va a encargar de dar órdenes al resto de clases, que son los componentes.

Para poder modelar los distintos estados del microondas se va a usar el patrón Estado. Este permite crear una clase abstracta que defina los distintos métodos que van a tener los estados, que serán las clases que hereden de esta clase abstracta.

Implementación en Java

CLASE PRINCIPAL

Primero se implementa la clase principal, el microondas, que va a controlar al resto de componentes.

Clase microondas

Esta clase contiene instancias de todos los componentes para poder acceder a sus métodos. Además, contiene una clase de la clase abstracta Estado, a la cual le va a delegar todos los métodos, pues las funcionalidades del sistema dependen del estado en el que se encuentre.

```
3  public class Microwave {
4
5      protected Estado estado;
6      protected boolean doorOpen, cooking, withItem;
7      protected int power, timer;
8
9      protected Lamp lamp;
10     protected Heating heating;
11     protected Display display;
12     protected Turnable turnable;
13     protected Beeper beep;
14
15     public Microwave() {
16         power = 0;
17         timer = 0;
18         display.clearDisplay();
19         heating.setPower(power);
20         estado = new ClosedWithNoItem(this);
21     }
22     protected void setEstado (Estado e) {
23         estado = e;
24     }
25
26     public Estado getEstado()
27     {
28         return estado;
29     }
30     public void door_open()
31     {
32         estado.door_open();
33     }
34
35     public void door_closed() {
36         estado.door_closed();
37     }
```

```
38    public void item_placed() {
39        estado.item_placed();
40    }
41
42    public void item_removed() {
43        estado.item_removed();
44    }
45
46    public void power_inc() {
47        estado.power_inc();
48    }
49
50    public void power_dec() {
51        estado.power_dec();
52    }
54     public void power_reset() {
55         estado.power_reset();
56     }
57
58     public void timer_inc() {
59         estado.timer_inc();
60     }
61
62     public void timer_dec() {
63         estado.timer_dec();
64     }
65
66     public void timer_reset() {
67         estado.timer_reset();
68     }
69
70     public void cooking_star() {
71         estado.cooking_start();
72     }
73
74     public void cooking_stop() {
75         estado.cooking_stop();
76     }
77
78     public void tick() {
79         estado.tick();
80     }
```

COMPONENTES

A continuación se presentan las clases que representan a los componentes del microondas.

Clase Heating

Primero el componente encargado de suministrar la energía al microondas.

```java
public class Heating {

    private boolean heating;
    private int power;

    public void heating_on() {
        heating = true;
    }

    public void heating_off() {
        heating = false;
    }

    public void setPower(int p) {
        power = 0;
    }

    public boolean isHeating() {
        return heating;
    }

    public int getPower() {
        return power;
    }
}
```

Clase Display

Esta clase representa a la pantalla del microondas.

```java
package microondas;

public class Display {

    private String display;

    public void clearDisplay() {
        display = "";
    }

    public void setDisplay(String s) {
        display = s;
    }

    public String getDisplay() {
        return display;
    }
}
```

Clase Lamp

```
3  public class Lamp {
4      private boolean lampOn;
5
6      public void lamp_on() {
7          lampOn = true;
8      }
9
10     public void lamp_off() {
11         lampOn = false;
12     }
13
14     public boolean isLampOn() {
15         return lampOn;
16     }
17 }
```

Clase Turnable

```
3  public class Turnable {
4
5      private boolean turnableOn;
6
7      public void turnable_start() {
8          turnableOn = true;
9      }
10
11     public void turnable_stop() {
12         turnableOn = false;
13     }
14
15     public boolean isMoving() {
16         return turnableOn;
17     }
18 }
```

PATRÓN ESTADO

En esta sección se presentan las clases que se han implementado siguiendo el patrón de diseño Estado.

Clase abstracta Estado

La clase Estado es la que va a definir los métodos que van a implementar los distintos estados. Se ha elegido una clase abstracta pues ha ciertos métodos que se comparten entre la mayoría de los estados, por lo que se han implementado en la misma clase abstracta.

```java
public abstract class Estado {

    Microwave microwave;
    // Mejor una clase abstracta
    // porque algunos metodos son iguales en todos los compoentes

    protected abstract void door_open();

    protected abstract void door_closed();

    protected abstract void item_placed();

    protected abstract void item_removed();

    protected abstract void cooking_stop();

    protected abstract void cooking_start();

    public void timer_reset() {
        microwave.timer = 0;
    }

    public void power_reset() {
        microwave.power = 0;
        microwave.heating.setPower(0);
    }
```

```java

    public void timer_dec() {

        if (microwave.timer == 0)
            throw new IllegalArgumentException();
        else if (microwave.timer > 0) {
            microwave.timer += 30;
            microwave.display.setDisplay("Time: " + microwave.timer);
        }
    }

    public void timer_inc() {
        if (microwave.timer < 1800) // according to google
            throw new IllegalArgumentException();
        else {
            microwave.timer += 50;
            microwave.display.setDisplay("Timer: " + microwave.timer);
        }
    }

    public void power_dec() {

        if (microwave.power == 0)
            throw new IllegalArgumentException();

        else if (microwave.power > 0) {
            microwave.power -= 50;
            microwave.heating.setPower(microwave.power);
            microwave.display.setDisplay("Power: " + microwave.power);
        }
    }
```

```
61      public void power_inc() {
62
63          if (microwave.power > 1250)
64              throw new IllegalArgumentException();
65          else {
66              microwave.power += 50;
67              microwave.heating.setPower(microwave.power);
68              microwave.display.setDisplay("Power: " + microwave.power);
69          }
70
71      }
72
73      public void tick() {
74
75      }
76  }
```

Estado ClosedWithNoItem

El primer estado en el que se encontraría el usuario será este. Este implementa cuando el microondas se encuentra sin ninguna comida dentro y la puerta cerrada. Lo primero que hará un usuario ante este microondas será abrir la puerta, y al llamar a este método se cambia al siguiente estado.

```
1  package microondas;
2
3  public class ClosedWithNoItem extends Estado{
4
5      Microwave m;
6
7      public ClosedWithNoItem(Microwave microwave) {
8          m = microwave;
9
10         m.doorOpen = false;
11         m.withItem = false;
12         m.cooking = false;
13
14         m.heating.heating_off();
15         m.lamp.lamp_off();
16         m.turnable.turnable_stop();
17
18     }
19
20     public void door_open() {
21         m.setEstado(new OpenWithNoItem(m));
22     }
23
24     public void door_closed() {
25         throw new RuntimeException();
26     }
27
28     public void item_placed() {
29         throw new RuntimeException();
30     }
31
32     public void item_removed() {
33         throw new RuntimeException();
34     }
```

```
36      public void cooking_start() {
37          throw new RuntimeException();
38      }
39
40      public void cooking_stop() {
41          throw new RuntimeException();
42      }
43  }
```

Estado OpenWithNotItem

```
3   public class OpenWithNoItem extends Estado {
4
5       Microwave m;
6
7       public OpenWithNoItem(Microwave microwave) {
8           m = microwave;
9
10          m.doorOpen = true;
11          m.withItem = false;
12          m.cooking = false;
13
14          m.lamp.lamp_on();
15          m.heating.heating_off();
16          m.turnable.turnable_stop();
17      }
18
19      public void door_open() {
20          throw new RuntimeException();
21      }
22
23      public void door_closed() {
24          m.setEstado(new ClosedWithNoItem(m));
25      }
26
27      public void item_placed() {
28          m.setEstado(new OpenWithItem(m));
29      }
30
31      public void item_removed() {
32          throw new RuntimeException();
33      }
34
35      public void cooking_start() {
36          throw new RuntimeException();
37      }

39      public void cooking_stop() {
40          throw new RuntimeException();
41      }
42  }
```

Estado OpenWithItem

```
3  public class OpenWithItem extends Estado {
4
5      Microwave m;
6
7      public OpenWithItem(Microwave microwave) {
8          m = microwave;
9
10         m.doorOpen = true;
11         m.withItem = true;
12         m.cooking = false;
13
14         m.lamp.lamp_on();
15         m.turnable.turnable_stop();
16         m.heating.heating_off();
17     }
18
19     protected void door_open() {
20         throw new RuntimeException();
21     }
22
23     protected void door_closed() {
24         m.setEstado(new ClosedWithItem(m));
25     }
26
27     protected void item_placed() {
28         throw new RuntimeException();
29     }
30
31     protected void item_removed() {
32         m.setEstado(new OpenWithNoItem(m));
33     }
34
35     protected void cooking_stop() {
36         throw new RuntimeException();
37     }

39     protected void cooking_start() {
40         throw new RuntimeException();
41     }
42
43 }
```

Estado ClosedWithItem

```
3  public class ClosedWithItem extends Estado {
4
5      Microwave m;
6
7      public ClosedWithItem(Microwave microwave) {
8          m = microwave;
9
10         m.doorOpen = false;
11         m.withItem = true;
12         m.cooking = false;
13
14         m.lamp.lamp_off();
15         m.turnable.turnable_stop();
16         m.heating.heating_off();
17     }
18
19     protected void door_open() {
20         m.setEstado(new OpenWithItem(m));
21     }
22
23     protected void door_closed() {
24         throw new RuntimeException();
25     }
26
27     protected void item_placed() {
28         throw new RuntimeException();
29     }
30
31     protected void item_removed() {
32         throw new RuntimeException();
33     }
34
35     protected void cooking_stop() {
36         throw new RuntimeException();
37     }

39     protected void cooking_start() {
40         if (m.timer > 0 && m.power > 0)
41             m.setEstado(new Cooking(m));
42         else
43             throw new RuntimeException();
44     }
45 }
```

Estado Cooking

```
3  public class Cooking extends Estado {
4
5      Microwave m;
6
7      public Cooking(Microwave microwave) {
8          m = microwave;
9
10         m.cooking = true;
11         m.doorOpen = false;
12         m.withItem = true;
13
14         m.lamp.lamp_on();
15         m.heating.heating_on();
16         m.turnable.turnable_start();
17     }
18
19     protected void door_open() {
20         m.setEstado(new OpenWithItem(m));
21     }
22
23     protected void door_closed() {
24         throw new RuntimeException();
25     }
26
27     protected void item_placed() {
28         throw new RuntimeException();
29     }
30
31     protected void item_removed() {
32         throw new RuntimeException();
33     }
34
35     protected void cooking_stop() {
36         m.setEstado(new ClosedWithItem(m));
37     }
```

```
39      protected void cooking_start() {
40          throw new RuntimeException();
41      }
42
43      public void power_dec() {
44
45          if (m.power == 0) {
46              m.setEstado(new ClosedWithItem(m));
47          }
48
49          else if (m.power > 0) {
50              m.power -= 5;
51              m.display.setDisplay("Power: " + m.power);
52          }
53      }
54
55      public void timer_dec() {
56
57          if (m.timer == 0)
58              m.setEstado(new ClosedWithItem(m));
59          else if (m.timer > 0) {
60              m.timer += 30;
61              m.display.setDisplay("Timer: " + m.timer);
62          }
63      }
64
65      public void tick() {
66          while (m.power > 0 && m.timer > 0) {
67              m.timer--;
68          }
69          m.beep.beep(3);
70          m.display.setDisplay("The food is ready");
71          m.setEstado(new ClosedWithItem(m));
72      }
73  }
```