Proyecto final IS-Avanzada: Horno-Microondas

Hugo Ávalos de Rorthais

Enlace: https://github.com/hugoavalos01/ISA_Microwave

1. Implementación en Java, usando el patrón de diseño Estado.

Interfaz Estado: Distintos métodos para manejar los diferentes componentes del microondas como la puerta, la luz, el temporizador, la potencia y meter o sacar la comida. Estos métodos harán que se cambie de un estado a otro.

```java
package microwave;

public interface Estado {

    public void door_opened(Microwave m);

    public void door_closed(Microwave m);

    public void item_placed(Microwave m);

    public void item_removed(Microwave m);

    public void cooking_start(Microwave m);

    public void cooking_stop(Microwave m);

    public void timer_reset(Microwave m);

    public void power_reset(Microwave m);

    public void tick(Microwave m);

    public void power_desc(Microwave m);

    public void timer_desc(Microwave m);
}

```

Clase Microwave: El constructor inicial crea un microondas cerrado y vacío, con tiempo y potencia 0. Tenemos los mismos métodos que la interfaz y además los getters y los setters.

```java
package microwave;
public class Microwave {

    private boolean doorOpen;
    private Integer power;
    private Integer timer;
    private boolean cooking;
    private boolean withItem;
    private Estado estado;
    private Heating heatingConnection = new Heating();
    private Lamp lampConnection = new Lamp();
    private Turnable turnableConnection = new Turnable();
    private Beeper beeperConnection = new Beeper();
    private Display displayConnection = new Display();

    //Constructor

    public Microwave() {

        doorOpen = false;
        cooking = false;
        withItem = false;
        power = 0;
        timer = 0;
        estado = new ClosedWithNoItem(this);

    }

    //Metodos

    public void door_opened() {
        estado.door_opened(this);
    }

    public void door_closed() {
        estado.door_closed(this);
    }

    public void item_placed() {
        estado.item_placed(this);
    }

    public void item_removed() {
        estado.item_removed(this);
    }

    public void power_inc() {
        power++;
        displayConnection.setDisplay(Integer.toString(power));
    }

    public void power_desc() {
        estado.power_desc(this);
    }

    public void power_reset() {
        estado.power_reset(this);
        displayConnection.setDisplay(Integer.toString(power));
    }

    public void timer_inc() {
        timer++;
        displayConnection.setDisplay(Integer.toString(timer));
    }

    public void timer_desc() {
        estado.timer_desc(this);
    }

    public void timer_reset() {
        estado.timer_reset(this);
        displayConnection.setDisplay(Integer.toString(timer));
    }
```

```java
public void cooking_start() {
    estado.cooking_start(this);
}

public void cooking_stop() {
    estado.cooking_stop(this);
}

public void tick() {
    estado.tick(this);
}

// Getters y Setters

public boolean isDoorOpen() {
    return doorOpen;
}

public void setDoorOpen(boolean doorOpen) {
    this.doorOpen = doorOpen;
}

public int getPower() {
    return power;
}

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

public int getTimer() {
    return timer;
}

public void setTimer(int t) {
    timer = t;
}

public boolean isCooking() {
    return cooking;
}

public void setCooking(boolean c) {
    cooking = c;
}

public boolean isWithItem() {
    return withItem;
}

public void setWithItem(boolean withItem) {
    this.withItem = withItem;
}

public Estado getEstado() {
    return estado;
}

public void setEstado(Estado e) {
    estado = e;
}

public Heating getHeatingConnection() {
    return heatingConnection;
}

public Lamp getLampConnection() {
    return lampConnection;
}

public Turnable getTurnableConnection() {
    return turnableConnection;
}

public Beeper getBeeperConnection() {
    return beeperConnection;
}

public Display getDisplayConnection() {
    return displayConnection;
}

}
```

Clase Beeper y beepListener: Para simular los sonidos del beeper se ha implementado el patrón Observador y se ha creado **beepListener**. Este recibe el número de beeps según el método de la clase Beeper y lo almacena en la variable "beeps". Mediante el método "times" nos aseguramos que han sonado los beeps esperados y reiniciamos la variable a 0.

```
package microwave;

public class Beeper {

    public void beep(int d) {
        beepListener.listen(d);
    }
}
```

```
package microwave;

public class beepListener {

    private static int beeps = 0;

    public static void listen (int b) {
        beeps = b;
    }

    public static boolean times(int t) {
        int b = beeps;
        beeps = 0;
        return b == t;
    }
}
```

Clase Lamp:

```
1  package microwave;
2
3  public class Lamp {
4
5      private boolean lampOn = false;
6
7      void lampOn() {
8          lampOn = true;
9      }
10
11     void lampOff() {
12         lampOn = false;
13     }
14
15     public boolean isLampOn() {
16         return lampOn;
17     }
18 }
```

Clase Heating:

```java
package microwave;

public class Heating {

    private boolean heating = false;
    private int power = 0;

    void heatingOn() {
        heating = true;
    }

    void heatingOff() {
        heating = false;
    }

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

    int getPower() {
        return power;
    }

    public boolean isHeating() {
        return heating;
    }
}
```

Clase Display:

```java
package microwave;

public class Display {

    private String display;

    void clearDisplay() {
        display = null;
    }

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

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

Clase Turnable:

```java
package microwave;

public class Turnable {

    private boolean turnableOn = false;

    void turnable_start() {
        turnableOn = true;
    }

    void turnable_stop() {
        turnableOn = false;
    }

    public boolean isMoving() {
        return turnableOn;
    }
}

```

Microondas OpenWithNoItem:

```java
package microwave;

public class OpenWithNoItem implements Estado {

    public OpenWithNoItem(Microwave m) {
        // TODO Auto-generated constructor stub

        m.setDoorOpen(true);
        m.setWithItem(false);
        m.setCooking(false);

        m.getHeatingConnection().heatingOff();
        m.getTurnableConnection().turnable_stop();
        m.getLampConnection().lampOn();
    }

    @Override
    public void door_opened(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: La puerta ya esta abierta");
    }

    @Override
    public void door_closed(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new ClosedWithNoItem(m));

    }

    @Override
    public void item_placed(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new OpenWithItem(m));
    }

    @Override
    public void item_removed(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: No hay ning□n objeto para sacar");

    }

```

```
    @Override
    public void cooking_start(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: La puerta tiene que estar cerrada");

    }

    @Override
    public void cooking_stop(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: El microondas no esta cocinando");

    }

    @Override
    public void timer_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setTimer(0);
    }

    @Override
    public void power_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setPower(0);
    }

    @Override
    public void tick(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: El microondas no esta cocinando");

    }

    @Override
    public void power_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getPower() > 0) {
            m.setPower(m.getPower() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getPower()));
        }
    }
```

```
    @Override
    public void timer_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setTimer(0);
    }

    @Override
    public void power_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setPower(0);
    }

    @Override
    public void tick(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: El microondas no esta cocinando");

    }

    @Override
    public void power_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getPower() > 0) {
            m.setPower(m.getPower() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getPower()));
        }
    }

    @Override
    public void timer_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getTimer() > 0) {
            m.setTimer(m.getTimer() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getTimer()));
        }
    }

}
```

Microondas OpenWithItem:

```java
package microwave;

public class OpenWithItem implements Estado {

    public OpenWithItem (Microwave m) {

        m.setDoorOpen(true);
        m.setWithItem(true);
        m.setCooking(false);

        m.getHeatingConnection().heatingOff();
        m.getTurnableConnection().turnable_stop();
        m.getLampConnection().lampOn();

    }

    @Override
    public void door_opened(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: La puerta ya esta abierta");
    }

    @Override
    public void door_closed(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new ClosedWithItem(m));
    }

    @Override
    public void item_placed(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: La puerta ya esta abierta");

    }

    public void item_removed(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new OpenWithNoItem(m));
    }

    @Override
    public void cooking_start(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: La puerta tiene que estar cerrada");

    }

    @Override
    public void cooking_stop(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: El microondas no esta cocinando");

    }

    @Override
    public void timer_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setTimer(0);
    }

    @Override
    public void power_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setPower(0);
    }

    @Override
    public void tick(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: El microondas no esta cocinando");

    }
```

```
    @Override
    public void power_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getPower() > 0) {
            m.setPower(m.getPower() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getPower()));
        }
    }

    @Override
    public void timer_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getTimer() > 0) {
            m.setTimer(m.getTimer() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getTimer()));
        }
    }

}
```

Microondas ClosedWithNoItem:

```
1  package microwave;
2
3  public class ClosedWithNoItem implements Estado {
4
5      public ClosedWithNoItem(Microwave m) {
6          // TODO Auto-generated constructor stub
7
8          m.setDoorOpen(false);
9          m.setWithItem(false);
10         m.setCooking(false);
11
12         m.getHeatingConnection().heatingOff();
13         m.getTurnableConnection().turnable_stop();
14         m.getLampConnection().lampOff();
15     }
16
17     @Override
18     public void door_opened(Microwave m) {
19         // TODO Auto-generated method stub
20         m.setEstado(new OpenWithNoItem(m));
21     }
22
23     @Override
24     public void door_closed(Microwave m) {
25         // TODO Auto-generated method stub
26         throw new IllegalStateException("Error: La puerta ya esta cerrada");
27
28     }
29
30     @Override
31     public void item_placed(Microwave m) {
32         // TODO Auto-generated method stub
33         throw new IllegalStateException("Error: Hay que abrir la puerta primero");
34
```

```
37      @Override
38      public void item_removed(Microwave m) {
39          // TODO Auto-generated method stub
40          throw new IllegalStateException("Error: Hay que abrir la puerta primero");
41
42      }
43
44      @Override
45      public void cooking_start(Microwave m) {
46          // TODO Auto-generated method stub
47          throw new IllegalStateException("Error: No hay nada dentro");
48
49      }
50
51      @Override
52      public void cooking_stop(Microwave m) {
53          // TODO Auto-generated method stub
54          throw new IllegalStateException("Error: El microondas no esta cocinando");
55
56      }
57
58      @Override
59      public void timer_reset(Microwave m) {
60          // TODO Auto-generated method stub
61          m.setTimer(0);
62      }
63
64      @Override
65      public void power_reset(Microwave m) {
66          // TODO Auto-generated method stub
67          m.setPower(0);
68      }
69
70      @Override
71      public void tick(Microwave m) {
72          // TODO Auto-generated method stub
73          throw new IllegalStateException("Error: El microondas no esta cocinando");
```

```
    @Override
    public void power_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getPower() > 0) {
            m.setPower(m.getPower() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getPower()));
        }
    }

    @Override
    public void timer_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getTimer() > 0) {
            m.setTimer(m.getTimer() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getTimer()));
        }
    }

}
```

Microondas ClosedWithItem:

```java
package microwave;

public class ClosedWithItem implements Estado {

    public ClosedWithItem(Microwave m) {
        // TODO Auto-generated constructor stub

        m.setDoorOpen(false);
        m.setWithItem(true);
        m.setCooking(false);

        m.getHeatingConnection().heatingOff();
        m.getTurnableConnection().turnable_stop();
        m.getLampConnection().lampOff();
    }

    @Override
    public void door_opened(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new OpenWithItem(m));
    }

    @Override
    public void door_closed(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: La puerta ya esta cerrada");

    }

    @Override
    public void item_placed(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: Hay que abrir la puerta primero");

    }
```

```java
    @Override
    public void item_removed(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: Hay que abrir la puerta primero");

    }

    @Override
    public void cooking_start(Microwave m) {
        // TODO Auto-generated method stub

        if (m.getTimer() == 0 || m.getPower() == 0) {

            throw new IllegalStateException("Error: Power y Timer deben ser mayor a 0");
        }

        else {

            m.setEstado(new Cooking(m));
        }

    }

    @Override
    public void cooking_stop(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: El microondas no esta cocinando");

    }

    @Override
    public void timer_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setTimer(0);
    }
```

```java
    @Override
    public void power_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setPower(0);
    }

    @Override
    public void tick(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: El microondas no esta cocinando");

    }

    @Override
    public void power_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getPower() > 0) {
            m.setPower(m.getPower() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getPower()));
        }
    }

    @Override
    public void timer_desc(Microwave m) {
        // TODO Auto-generated method stub
        if (m.getTimer() > 0) {
            m.setTimer(m.getTimer() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getTimer()));
        }
    }

}
```

Microondas Cooking:

```java
package microwave;

public class Cooking implements Estado {

    public Cooking(Microwave m) {
        // TODO Auto-generated constructor stub

        m.setDoorOpen(false);
        m.setWithItem(true);
        m.setCooking(true);

        m.getHeatingConnection().heatingOn();
        m.getTurnableConnection().turnable_start();
        m.getLampConnection().lampOn();
        m.getHeatingConnection().setPower(m.getPower());
    }

    @Override
    public void door_opened(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new OpenWithItem(m));
    }

    @Override
    public void door_closed(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: La puerta ya esta cerrada");

    }

    @Override
    public void item_placed(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: Hay que abrir la puerta primero");

    }

    @Override
    public void item_removed(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: Hay que abrir la puerta primero

    }

    @Override
    public void cooking_start(Microwave m) {
        // TODO Auto-generated method stub
        throw new IllegalStateException("Error: Ya est cocinando");

    }

    @Override
    public void cooking_stop(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new ClosedWithItem(m));
    }

    @Override
    public void timer_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new ClosedWithItem(m));
        m.setTimer(0);
    }

    @Override
    public void power_reset(Microwave m) {
        // TODO Auto-generated method stub
        m.setEstado(new ClosedWithItem(m));
        m.setPower(0);
    }
```

```
73      public void tick(Microwave m) {
74          // TODO Auto-generated method stub
75
76          if (m.getTimer() > 1) {
77              m.timer_desc();
78          } else {
79              m.timer_desc();
80              m.getBeeperConnection().beep(3);
81              m.getDisplayConnection().setDisplay("Listo");
82              cooking_stop(m);
83          }
84      }
85
86      @Override
87      public void power_desc(Microwave m) {
88          // TODO Auto-generated method stub
89
90          if (m.getPower() == 0) {
91              cooking_stop(m);
92
93          } else {
94
95              m.setPower(m.getPower() - 1);
96              m.getDisplayConnection().setDisplay(Integer.toString(m.getPower()));
97          }
98
99      }
```

```
    @Override
    public void timer_desc(Microwave m) {
        // TODO Auto-generated method stub

        if (m.getTimer() == 1) {
            m.getBeeperConnection().beep(3);
            m.getDisplayConnection().setDisplay("Listo");
            cooking_stop(m);

        } else {

            m.setTimer(m.getTimer() - 1);
            m.getDisplayConnection().setDisplay(Integer.toString(m.getTimer()));

        }
    }

}
```

2. Definir pruebas unitarias con Junit para cada uno de los componentes que conforman el sistema:

```java
package microwave;
import org.junit.jupiter.api.*;

public class Test_Microwave {

	private Microwave m;

	@BeforeEach
	void init() {
	    m = new Microwave();
	}


	/*
	 * Comprobamos los componentes del microondas
	 */

	@Test
	void testBeeper() {

	    Beeper b = new Beeper();

	    // Comprobamos que escucha los beeps
	    b.beep(3);
	    Assertions.assertTrue(beepListener.times(3));

	    // Comprobamos que se reinicia a 0
	    Assertions.assertTrue(beepListener.times(0));
	}
	@Test
	void testDisplay() {

	    Display d = new Display();

	    // Comprobamos que se inicia sin ningun mensaje
	    Assertions.assertEquals(null, d.getDisplay());


	    // Comprobamos que cambia el mensaje
	    d.setDisplay("Mensaje");
	    Assertions.assertEquals("Mensaje", d.getDisplay());

	    //Comprobamos que el mensaje se borra
	    d.clearDisplay();
	    Assertions.assertEquals(null, d.getDisplay());

	}

```

```java
@Test
void testTimer() {

    // Comprobamos que se inicia en 0
    Assertions.assertEquals(0, m.getTimer());

    // Comprobamos que incrementa el tiempo
    m.timer_inc();
    Assertions.assertEquals(1, m.getTimer());

    // Comprobamos que decrementa el tiempo
    m.timer_inc();
    m.timer_desc();
    Assertions.assertEquals(1, m.getTimer());

    //Comprobamos que el display muestra los cambios de tiempo
    Assertions.assertEquals("1", m.getDisplayConnection().getDisplay());

    // Comprobamos que se reinicia el tiempo
    m.timer_reset();
    Assertions.assertEquals(0, m.getTimer());

    // Comprobamos que no baja de 0
    m.timer_desc();
    Assertions.assertEquals(0, m.getTimer());

}

    @Test
    void testPower() {

        // Comprobamos que se inicia en 0
        Assertions.assertEquals(0, m.getPower());

        // Comprobamos que incrementa la potencia
        m.power_inc();
        Assertions.assertEquals(1, m.getPower());

        // Comprobamos que decrementa la potencia
        m.power_inc();
        m.power_desc();
        Assertions.assertEquals(1, m.getPower());

        //Comprobamos que el display muestra los cambios de potencia
        Assertions.assertEquals("1", m.getDisplayConnection().getDisplay());

        // Comprobamos que se reinicia la potencia
        m.power_reset();
        Assertions.assertEquals(0, m.getPower());

        // Comprobamos que no baja de 0
        m.power_desc();
        Assertions.assertEquals(0, m.getPower());

    }

    @Test
    void turnableTest() {

        Turnable t = new Turnable();

        // Comprobamos que se inicia parado
        Assertions.assertFalse(t.isMoving());

        // Comprobamos que gira y se para correctamente
        t.turnable_start();
        Assertions.assertTrue(t.isMoving());

        t.turnable_stop();
        Assertions.assertFalse(t.isMoving());

    }
```

```java
    @Test
    void heatingTest() {

        Heating h = new Heating();

        // Comprobamos que se inicia con potencia 0 y sin calentar
        Assertions.assertFalse(h.isHeating());
        Assertions.assertEquals(0, h.getPower());

        // Comprobamos que se enciende y se apaga
        h.heatingOn();
        Assertions.assertTrue(h.isHeating());

        h.heatingOff();
        Assertions.assertFalse(h.isHeating());

        // Comprobamos que podemos ajustar la potencia
        h.setPower(5);
        Assertions.assertEquals(5, h.getPower());

    }

    @Test
    void lampTest() {

        Lamp l = new Lamp();

        // Comprobamos que se inicia apagada
        Assertions.assertFalse(l.isLampOn());

        // Comprobamos que se enciende y apaga
        l.lampOn();
        Assertions.assertTrue(l.isLampOn());

        l.lampOff();
        Assertions.assertFalse(l.isLampOn());


    }
```

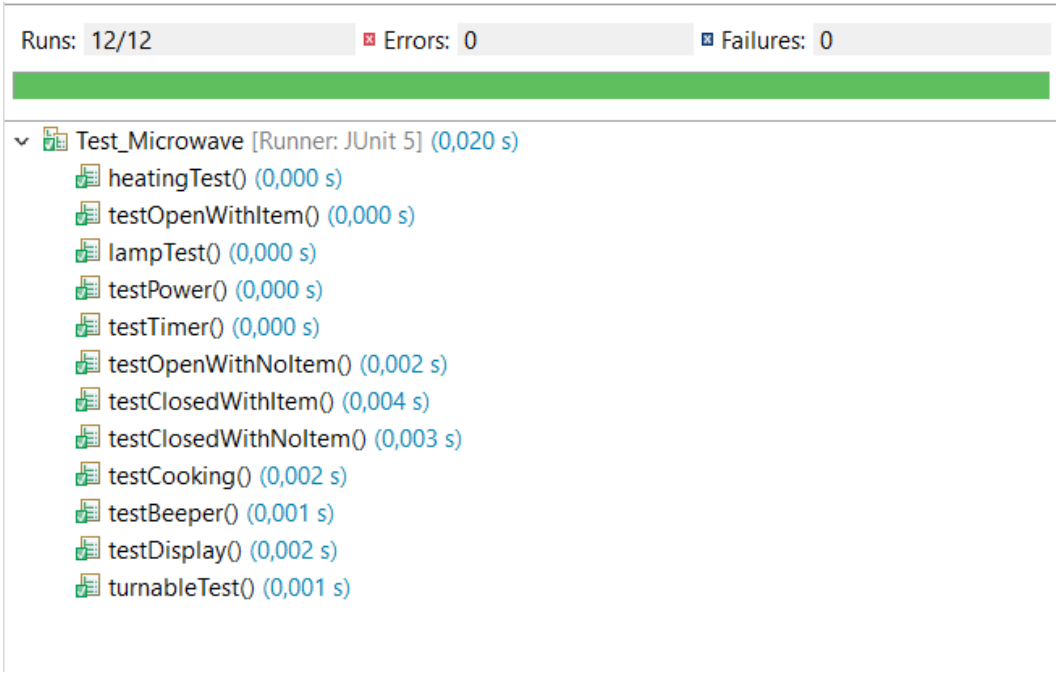
```java
    /*
     * Comprobamos los estados del microondas
     */

    @Test
    void testOpenWithNoItem() {

        m.setEstado(new OpenWithNoItem(m));
        Assertions.assertTrue(m.getEstado() instanceof OpenWithNoItem);

        // Comprobamos que se inicia correctamente
        Assertions.assertFalse(m.isWithItem());
        Assertions.assertTrue(m.isDoorOpen());
        Assertions.assertFalse(m.isCooking());

        Assertions.assertFalse(m.getHeatingConnection().isHeating());
        Assertions.assertTrue(m.getLampConnection().isLampOn());
        Assertions.assertFalse(m.getTurnableConnection().isMoving());

        // Comprobamos que se puede modificar el tiempo y la potencia
        testPower();
        testTimer();

        // Comprobamos las excepciones
        Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_stop());
        Assertions.assertThrows(IllegalStateException.class, () -> m.item_removed());
        Assertions.assertThrows(IllegalStateException.class, () -> m.door_opened());
        Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_start());
        Assertions.assertThrows(IllegalStateException.class, () -> m.tick());

        // Comprobamos que podemos cerrar la puerta y cambia de estado
        m.door_closed();
        Assertions.assertTrue(m.getEstado() instanceof ClosedWithNoItem);

        // Comprobamos que si metemos comida, cambia de estado
        m.setEstado(new OpenWithNoItem(m));
        m.item_placed();
        Assertions.assertTrue(m.getEstado() instanceof OpenWithItem);
    }
```

```java
@Test
void testClosedWithNoItem() {

    m.setEstado(new ClosedWithNoItem(m));
    Assertions.assertTrue(m.getEstado() instanceof ClosedWithNoItem);

    // Comprobamos que se inicia correctamente
    Assertions.assertFalse(m.isWithItem());
    Assertions.assertFalse(m.isDoorOpen());
    Assertions.assertFalse(m.isCooking());

    Assertions.assertFalse(m.getHeatingConnection().isHeating());
    Assertions.assertFalse(m.getLampConnection().isLampOn());
    Assertions.assertFalse(m.getTurnableConnection().isMoving());

    // Comprobamos que se puede modificar el tiempo y la potencia
    testPower();
    testTimer();

    // Comprobamos las excepciones
    Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_stop());
    Assertions.assertThrows(IllegalStateException.class, () -> m.item_removed());
    Assertions.assertThrows(IllegalStateException.class, () -> m.item_placed());
    Assertions.assertThrows(IllegalStateException.class, () -> m.door_closed());
    Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_start());
    Assertions.assertThrows(IllegalStateException.class, () -> m.tick());

    // Comprobamos que podemos abrir la puerta y cambia de estado
    m.door_opened();
    Assertions.assertTrue(m.getEstado() instanceof OpenWithNoItem);

}

@Test
void testOpenWithItem() {

    m.setEstado(new OpenWithItem(m));
    Assertions.assertTrue(m.getEstado() instanceof OpenWithItem);

    // Comprobamos que se inicia correctamente
    Assertions.assertTrue(m.isWithItem());
    Assertions.assertTrue(m.isDoorOpen());
    Assertions.assertFalse(m.isCooking());

    Assertions.assertFalse(m.getHeatingConnection().isHeating());
    Assertions.assertTrue(m.getLampConnection().isLampOn());
    Assertions.assertFalse(m.getTurnableConnection().isMoving());

    // Comprobamos que se puede modificar el tiempo y la potencia
    testPower();
    testTimer();
```

```
        // Comprobamos las excepciones
        Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_stop());
        Assertions.assertThrows(IllegalStateException.class, () -> m.item_placed());
        Assertions.assertThrows(IllegalStateException.class, () -> m.door_opened());
        Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_start());
        Assertions.assertThrows(IllegalStateException.class, () -> m.tick());

        // Comprobamos que podemos cerrar la puerta y cambia de estado
        m.door_closed();
        Assertions.assertTrue(m.getEstado() instanceof ClosedWithItem);

        // Comprobamos que si sacamos comida, cambia de estado
        m.setEstado(new OpenWithItem(m));
        m.item_removed();
        Assertions.assertTrue(m.getEstado() instanceof OpenWithNoItem);
    }

    @Test
    void testClosedWithItem() {

        m.setEstado(new ClosedWithItem(m));
        Assertions.assertTrue(m.getEstado() instanceof ClosedWithItem);

        // Comprobamos que se inicia correctamente
        Assertions.assertTrue(m.isWithItem());
        Assertions.assertFalse(m.isDoorOpen());
        Assertions.assertFalse(m.isCooking());

        Assertions.assertFalse(m.getHeatingConnection().isHeating());
        Assertions.assertFalse(m.getLampConnection().isLampOn());
        Assertions.assertFalse(m.getTurnableConnection().isMoving());

        // Comprobamos que se puede modificar el tiempo y la potencia
        testPower();
        testTimer();

        // Comprobamos las excepciones
        Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_stop());
        Assertions.assertThrows(IllegalStateException.class, () -> m.item_removed());
        Assertions.assertThrows(IllegalStateException.class, () -> m.item_placed());
        Assertions.assertThrows(IllegalStateException.class, () -> m.door_closed());
        Assertions.assertThrows(IllegalStateException.class, () -> m.tick());

        // Comprobamos que podemos abrir la puerta y cambia de estado
        m.door_opened();
        Assertions.assertTrue(m.getEstado() instanceof OpenWithItem);
```

```
302             // Comprobamos que lo podemos poner a cocinar si tiene la potencia
303             // y tiempo definido, sino salta una excepcion
304             m.setEstado(new ClosedWithItem(m));
305             Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_start());
306 
307             m.setPower(1);
308             m.setTimer(1);
309             m.cooking_start();
310             Assertions.assertTrue(m.getEstado() instanceof Cooking);
311         }
312 
313         @Test
314         void testCooking() {
315 
316             m.setEstado(new ClosedWithItem(m));
317             m.setPower(10);
318             m.setTimer(10);
319             m.cooking_start();
320             Assertions.assertTrue(m.getEstado() instanceof Cooking);
321 
322             // Comprobamos que se inicia correctamente
323             Assertions.assertTrue(m.isWithItem());
324             Assertions.assertFalse(m.isDoorOpen());
325             Assertions.assertTrue(m.isCooking());
326 
327             Assertions.assertTrue(m.getHeatingConnection().isHeating());
328             Assertions.assertTrue(m.getLampConnection().isLampOn());
329             Assertions.assertTrue(m.getTurnableConnection().isMoving());
330 
331             // Comprobamos que si el tiempo se reinicia para y cambia de estado
332             m.timer_reset();
333             Assertions.assertFalse(m.isCooking());
334             Assertions.assertTrue(m.getEstado() instanceof ClosedWithItem);
335 
336             // Comprobamos que si la potencia se reinicia 0 para y cambia de estado
337             m.setTimer(10);
338             m.cooking_start();
339             m.power_reset();
340             Assertions.assertFalse(m.isCooking());
341             Assertions.assertTrue(m.getEstado() instanceof ClosedWithItem);
342 
343             // Comprobamos que podemos cambiar la potencia y el tiempo mientras cocina
344             m.setTimer(1);
345             m.setPower(10);
346             m.cooking_start();
347 
348             m.power_desc();
349             Assertions.assertEquals(9, m.getPower());
350 
351             m.power_inc();
352             Assertions.assertEquals(10, m.getPower());
353 
354             m.timer_inc();
355             Assertions.assertEquals(2, m.getTimer());
356 
357             m.timer_desc();
358             Assertions.assertEquals(1, m.getTimer());
359 
360             // Comprobamos que si el tiempo llega a 0 se para y cambia de estado
361             m.timer_desc();
362             Assertions.assertFalse(m.isCooking());
363             Assertions.assertTrue(beepListener.times(3));
364 
365             // Comprobamos que al abrir la puerta se para y cambia de estado
366             m.setEstado(new Cooking(m));
367             m.door_opened();
368             Assertions.assertFalse(m.isCooking());
369             Assertions.assertTrue(m.getEstado() instanceof OpenWithItem);
370 
371         }
372 }
373 
```

Resultados:



3. Definir un conjunto de escenarios de prueba para el sistema completo con Gherkin, e implementarlas en Cucumber.

Resultados:

```
[INFO] Tests run: 53, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.528 s - in hellocucumber.RunCucumberTest
[INFO] Running microwave.Test_Microwave
[INFO] Tests run: 12, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.101 s - in microwave.Test_Microwave
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 65, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.362 s
[INFO] Finished at: 2022-06-09T21:34:08+02:00
[INFO] ------------------------------------------------------------------------
```

StepDefinitions:

```java
package hellocucumber;

import io.cucumber.java.en.Given;

public class StepDefinitions {

    private Microwave m;

    /*
     * Given
     */

    @Given("closed empty microwave")
    public void closedEmpty() {
        m = new Microwave();
    }

    @Given("closed full microwave")
    public void closedFull() {
        openedFull();
        m.door_closed();
    }

    @Given("opened empty microwave")
    public void openedEmpty() {
        closedEmpty();
        m.door_opened();
    }

    @Given("opened full microwave")
    public void openedFull() {
        openedEmpty();
        m.item_placed();
    }

    @Given("cooking microwave with {int} power and {int} time")
    public void cooking(Integer power, Integer time) {
        closedFull();
        setTimer(time);
        setPower(power);
        m.cooking_start();
    }

    /*
     * When
     */

    @When("open the door")
    public void openDoor() {
        m.door_opened();
    }
```

```java
    @When("close the door")
    public void closeDoor() {
        m.door_closed();
    }

    @When("place food")
    public void placeFood() {
        m.item_placed();
    }

    @When("remove food")
    public void removeFood() {
        m.item_removed();
    }

    @When("set the timer to {int}")
    public void setTimer(Integer t) {
        m.timer_reset();
        for (int i = 0; i < t; i++) {
            m.timer_inc();
        }
    }

    @When("set the power to {int}")
    public void setPower(Integer p) {
        m.power_reset();
        for (int i = 0; i < p; i++) {
            m.power_inc();
        }
    }

    @When("increase timer")
    public void incTimer() {
        m.timer_inc();
    }

    @When("increase power")
    public void incPower() {
        m.power_inc();
    }

    @When("decrease timer")
    public void decTimer() {
        m.timer_desc();
    }

    @When("decrease power")
    public void decPower() {
        m.power_desc();
    }
```

```java
    @When("reset timer")
    public void resetTimer() {
        m.timer_reset();
    }

    @When("reset power")
    public void resetPower() {
        m.power_reset();
    }

    @When("start cooking")
    public void startCooking() {
        try {
            m.cooking_start();
        } catch (IllegalStateException ex) {
            notStartCooking();
        }
    }

    @When("timer goes down {int} seconds")
    public void timePasses(Integer t) {
        for (int i = 0; i < t; i++) {
            m.tick();
        }
    }


    /*
     * Then
     */

    @Then("microwave doesnt start cooking")
    public void notStartCooking() {
        Assertions.assertThrows(IllegalStateException.class, () -> m.cooking_start());

    }

    @Then("microwave starts cooking")
    public void StartCooking() {
        Assertions.assertTrue(m.isCooking());
    }

    @Then("microwave isnt cooking")
    public void notCooking() {
        Assertions.assertFalse(m.isCooking());
    }

    @Then("door opens")
    public void doorOpens() {
        Assertions.assertTrue(m.isDoorOpen());
    }
```

```
163     @Then("door closes")
164     public void doorCloses() {
165         Assertions.assertFalse(m.isDoorOpen());
166     }
167
168     @Then("light turns on")
169     public void lightOn() {
170         Assertions.assertTrue(m.getLampConnection().isLampOn());
171     }
172
173     @Then("light turns off")
174     public void lightOff() {
175         Assertions.assertFalse(m.getLampConnection().isLampOn());
176     }
177
178     @Then("microwave is full")
179     public void isFull() {
180         Assertions.assertTrue(m.isWithItem());
181     }
182
183     @Then("microwave is empty")
184     public void isEmpty() {
185         Assertions.assertFalse(m.isWithItem());
186     }
187
188     @Then("turnable turns")
189     public void turnableT() {
190         Assertions.assertTrue(m.getTurnableConnection().isMoving());
191     }
192
193     @Then("turnable not turns")
194     public void turnableNT() {
195         Assertions.assertFalse(m.getTurnableConnection().isMoving());
196     }
197
198     @Then("power is zero")
199     public void powerZero() {
200         Assertions.assertEquals("0", m.getDisplayConnection().getDisplay());
201     }
202
203     @Then("timer is zero")
204     public void timerZero() {
205         Assertions.assertEquals("0", m.getDisplayConnection().getDisplay());
206     }
207
208     @Then("heating on")
209     public void heatingOn() {
210         Assertions.assertTrue(m.getHeatingConnection().isHeating());
211     }
212
213     @Then("heating off")
214     public void heatingOff() {
215         Assertions.assertFalse(m.getHeatingConnection().isHeating());
216     }
217
218     @Then("display shows {string}")
219     public void displayShows(String s) {
220         Assertions.assertEquals(s, m.getDisplayConnection().getDisplay());
221     }
222
223     @Then("beeper sounds {int} times")
224     public void soundTimes(Integer t) {
225         Assertions.assertTrue(beepListener.times(t));
226     }
227 }
228
```

Features:

ClosedWithItem:

ClosedWithItem.feature

```
Feature: Closed full microwave


  Scenario: Open a closed microwave
    Given closed full microwave
    When open the door
    Then door opens
    And light turns on
    And heating off
    And turnable not turns

  Scenario: Reset power
    Given closed full microwave
    When reset power
    Then power is zero

  Scenario: Reset timer
    Given closed full microwave
    When reset timer
    Then timer is zero

Scenario: Try to cook
    Given closed full microwave
    When set the power to 5
    And set the timer to 5
    And start cooking
    Then microwave starts cooking

Scenario Outline: Set the timer
    Given closed full microwave
    When set the timer to <a>
    Then display shows "<b>"

    Examples:
      | a   | b   |
      |  -5 |   0 |
      |   0 |   0 |
      |   5 |   5 |

  Scenario Outline: Set the power
    Given closed full microwave
    When set the power to <a>
    Then display shows "<b>"

    Examples:
      | a   | b   |
      |  -5 |   0 |
      |   0 |   0 |
      |   5 |   5 |

```

ClosedWithNoItem:

```
Feature: Closed empty microwave


  Scenario: Open a closed microwave
    Given closed empty microwave
    When open the door
    Then door opens
    And light turns on
    And heating off
    And turnable not turns

  Scenario: Reset power
    Given closed empty microwave
    When reset power
    Then power is zero

  Scenario: Reset timer
    Given closed empty microwave
    When reset timer
    Then timer is zero

Scenario: Try to cook
    Given closed empty microwave
    When set the power to 5
    And set the timer to 5
    And start cooking
    Then microwave doesnt start cooking

Scenario Outline: Set the timer
    Given closed empty microwave
    When set the timer to <a>
    Then display shows "<b>"

    Examples:
      | a   | b   |
      |  -5 |   0 |
      |   0 |   0 |
      |   5 |   5 |

  Scenario Outline: Set the power
    Given closed empty microwave
    When set the power to <a>
    Then display shows "<b>"

    Examples:
      | a   | b   |
      |  -5 |   0 |
      |   0 |   0 |
      |   5 |   5 |

```

OpenWithNoItem:

```
Feature: Opened empty microwave


  Scenario: Place food in the microwave
    Given opened empty microwave
    When place food
    Then microwave is full

   Scenario: Close an opened microwave
    Given opened empty microwave
    When close the door
    Then door closes
    And light turns off
    And heating off
    And turnable not turns

   Scenario: Reset power
    Given opened empty microwave
    When reset power
    Then power is zero

   Scenario: Reset timer
    Given opened empty microwave
    When reset timer
    Then timer is zero

  Scenario: Try to cook
    Given opened empty microwave
    When set the power to 5
    And set the timer to 5
    And start cooking
    Then microwave doesnt start cooking

  Scenario Outline: Set timer
    Given opened empty microwave
    When set the timer to <a>
    Then display shows "<b>"

    Examples:
      | a   | b   |
      |  -5 |   0 |
      |   0 |   0 |
      |   5 |   5 |

```

```
   Scenario Outline: Set power
    Given opened empty microwave
    When set the power to <a>
    Then display shows "<b>"

    Examples:
      | a   | b   |
      |  -5 |   0 |
      |   0 |   0 |
      |   5 |   5 |
```

OpenWithItem:

```
Feature: Opened full microwave


  Scenario: Remove food from the microwave
    Given opened full microwave
    When remove food
    Then microwave is empty

   Scenario: Close an opened microwave
    Given opened full microwave
    When close the door
    Then door closes
    And light turns off
    And heating off
    And turnable not turns

   Scenario: Reset power
    Given opened full microwave
    When reset power
    Then power is zero

   Scenario: Reset timer
    Given opened full microwave
    When reset timer
    Then timer is zero

  Scenario: Try to cook
    Given opened full microwave
    When set the power to 5
    And set the timer to 5
    And start cooking
    Then microwave doesnt start cooking

  Scenario Outline: Set timer
    Given opened full microwave
    When set the timer to <a>
    Then display shows "<b>"

    Examples:
      | a   | b   |
      |  -5 |   0 |
      |   0 |   0 |
      |   5 |   5 |


      Scenario Outline: Set power
       Given opened full microwave
       When set the power to <a>
       Then display shows "<b>"

       Examples:
         | a   | b   |
         |  -5 |   0 |
         |   0 |   0 |
         |   5 |   5 |
```

Cooking:

```
Feature: microwave cooking

  Scenario: start cooking
    Given closed full microwave
    When set the power to 5
    And set the timer to 5
    And start cooking
    Then microwave starts cooking
    And light turns on
    And heating on
    And turnable turns

  Scenario: increase power
    Given cooking microwave with 5 power and 5 time
    When increase power
    Then display shows "6"

  Scenario: decrease power
    Given cooking microwave with 5 power and 5 time
    When decrease power
    Then display shows "4"

   Scenario: increase timer
    Given cooking microwave with 5 power and 5 time
    When increase timer
    Then display shows "6"

  Scenario: decrease timer
    Given cooking microwave with 5 power and 5 time
    When decrease timer
    Then display shows "4"

  Scenario: stop cooking after resetting the power
    Given cooking microwave with 5 power and 5 time
    When reset power
    Then microwave isnt cooking
    And heating off
    And light turns off
    And turnable not turns
    And display shows "0"


  Scenario: stop cooking after resetting the timer
    Given cooking microwave with 5 power and 5 time
    When reset timer
    Then microwave isnt cooking
    And heating off
    And light turns off
    And turnable not turns
    And display shows "0"
```

```
Scenario: stop cooking if the door is opened
  Given cooking microwave with 5 power and 5 time
  When open the door
  Then door opens
  And microwave isnt cooking
  And heating off
  And light turns on
  And turnable not turns

Scenario Outline: cooking stop if timer finishes
  Given closed full microwave
  When set the power to <a>
  And set the timer to <b>
  And start cooking
  And timer goes down <c> seconds
  Then microwave isnt cooking
  And light turns off
  And turnable not turns
  And heating off
  And beeper sounds 3 times
  And display shows "Listo"

  Examples:
    | a   | b   | c   |
    |  10 |   5 |   5 |
    |  10 | 100 | 100 |
    |  10 |  15 |  15 |
```