E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Proyecto
MICROONDAS

UNIVERSIDAD DE MÁLAGA

Carlos Beltrán López

https://github.com/carlosbelop/Microondas.git

ÍNDICE

INTRODUCCIÓN

El proyecto consiste en la implementación de un microondas en java. Estará compuesto de diferentes clases que interactuaran entre ellas dependiendo de las acciones realizadas. Para ello se usará el patrón estado estudiado en clase, mediante el cuál definiremos varias posibles situaciones para el microondas que son:

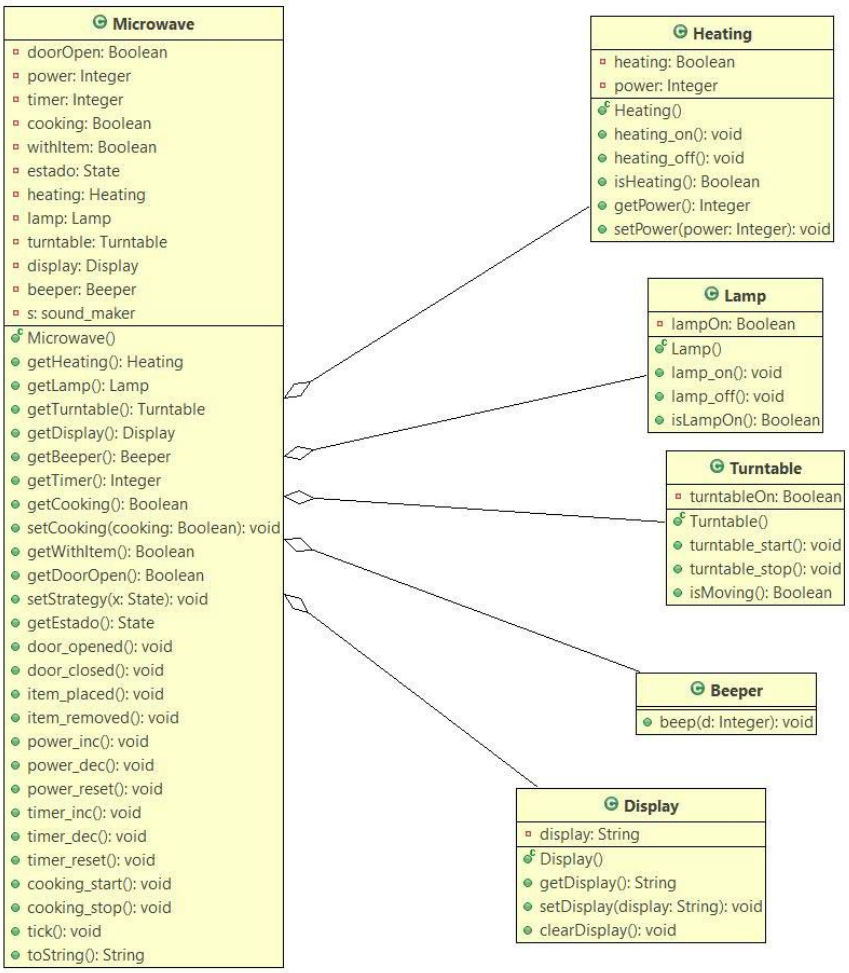
1. Cerrado y sin nada dentro
2. Abierto y sin nada dentro
3. Abierto y con comida dentro
4. Cerrado y con comida dentro
5. Cocinando

Para pasar de uno a otro, el microondas deberá de cumplir diferentes requisitos, por ejemplo, para meter comida, previamente la puerta debe estar abierta.

Además, el microondas está compuesto de otros componentes como la lámpara o el calentador que serán implementables y accesibles.

IMPLEMENTACIÓN EN JAVA

- MICROONDAS Y COMPONENTES



Se ha implementado la clase principal Microondas con todas las instancias de cada uno de los componentes, así como atributos de clase necesarios para determinar su estado y métodos para cambiarlos además de los getters y setters.

Además se incluye una instancia de la clase *sound_maker* que es la encargada de ser notificada por *Beeper* para hacer el sonido del pitido (en este caso representar en el display bip tantas veces como se indique).

Código

```
public class Microwave {
    private Boolean doorOpen;
    private Integer power;
    private Integer timer;
    private Boolean cooking;
    private Boolean withItem;
    private State estado;
    private Heating heating;
    private Lamp lamp;
    private Turntable turntable;
    private Display display;
    private Beeper beeper;
    private sound_maker s;

    public Microwave() {
        doorOpen = false;
        power = 0;
        timer = 0;
        cooking = false;
        withItem = false;
        heating = new Heating();
        lamp = new Lamp();
        turntable = new Turntable();
        display = new Display();
        beeper = new Beeper();
        estado = new ClosedWithNoItem(this);
        s=new sound_maker(this);
    }
    public Heating getHeating() {
        return heating;
    }
    public Lamp getLamp() {
        return lamp;
    }
    public Turntable getTurntable() {
        return turntable;
    }
```

```
public Display getDisplay() {
    return display;
}
public Beeper getBeeper() {
    return beeper;
}
public Integer getTimer() {
    return timer;
}
public Boolean getCooking() {
    return cooking;
}

public void setCooking(Boolean cooking) {
    this.cooking = cooking;
}

public Boolean getWithItem() {
    return withItem;
}

public Boolean getDoorOpen() {
    return doorOpen;
}

public void setStrategy(State x) {
    estado = x;
}

public State getEstado() {
    return estado;
}

public void door_opened() {
    doorOpen = true;
    estado.door_opened();
```

```
public void door_closed() {
    doorOpen = false;
    estado.door_closed();
}

public void item_placed() {
    if (doorOpen) {

        withItem = true;
        estado.item_placed();

    } else {
        display.setDisplay("You need to open the door first");
    }
}

public void item_removed() {
    if (doorOpen) {
        withItem = false;
        estado.item_removed();
    } else {
        display.setDisplay("You need to open the door first");
    }
}

public void power_inc() {

    estado.power_inc();
    if (power < 1000) {
        power += 100;
    }
    heating.setPower(power);
    display.setDisplay("Power: " + power.toString());

}
```

```
public void power_dec() {

    if (power - 100 <= 0) {
        estado.power_dec();
        power = 0;
    } else {
        power -= 100;
    }
    heating.setPower(power);
    display.setDisplay("Power: " + power.toString());

}

public void power_reset() {
    power = 0;
    heating.setPower(power);
    estado.power_reset();
    display.setDisplay("Power: " + power.toString());

}

public void timer_inc() {
    timer += 30;
    display.setDisplay("Time: " + timer.toString());

}

public void timer_dec() {

    if (timer - 30 <= 0) {
        estado.timer_dec();
        timer = 0;
    } else {
        timer -= 30;
    }
    display.setDisplay("Time: " + timer.toString());
```

```
public void timer_reset() {

    estado.timer_reset();
    timer = 0;
    display.setDisplay("Time: " + timer.toString());

}

public void cooking_start() {

    if (timer > 0 && power > 0 && !cooking)
        estado.cooking_start();

    if (timer <= 0)
        display.setDisplay("Choose cooking time");

    if (power <= 0)
        display.setDisplay("Choose cooking power");
}

public void cooking_stop() {

    estado.cooking_stop();
}

public void tick() {

    timer--;
    display.setDisplay("Time: " + timer.toString());

}
```

Microwave

```java
public class Heating {
    private Boolean heating;
    private Integer power;

    public Heating() {
        heating = false;
        power = 0;
    }

    public void heating_on() {
        heating = true;
    }

    public void heating_off() {
        heating = false;
    }

    public Boolean isHeating() {
        return heating;
    }

    public Integer getPower() {
        return power;
    }

    public void setPower(Integer power) {
        this.power = power;
    }
```

Clase Heating

```java
public class Lamp {

    private Boolean lampOn;

    public Lamp() {
        lampOn = false;
    }

    public void lamp_on() {
        lampOn = true;
    }

    public void lamp_off() {
        lampOn = false;
    }

    public Boolean isLampOn() {

        return lampOn;
    }

}
```

Clase Lamp

```java
public class Turntable {

    private Boolean turntableOn;

    public Turntable() {
        turntableOn = false;
    }

    public void turntable_start() {
        turntableOn = true;
    }

    public void turntable_stop() {
        turntableOn = false;
    }

    public Boolean isMoving() {
        return turntableOn;
    }

}
```

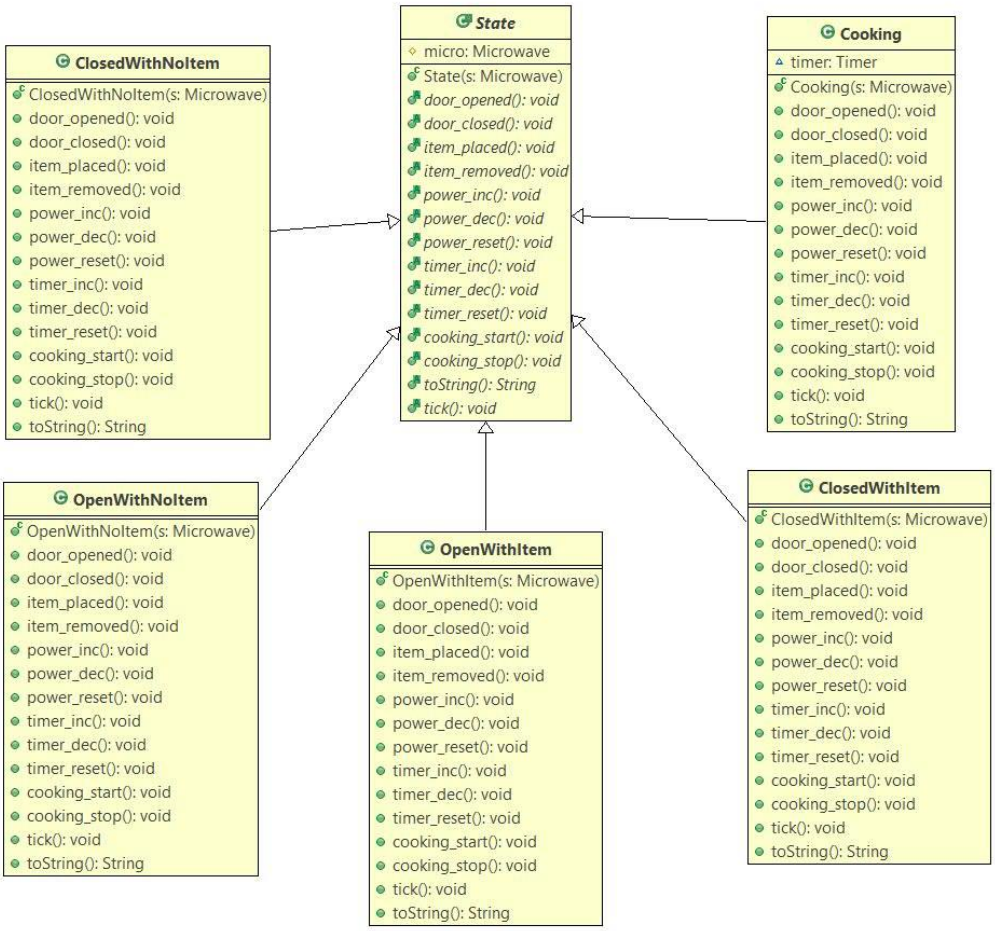
Clase Turntable

```java
public class Beeper {

    public void beep(Integer d) {
        sound_maker.activate(d);
        try {
            Thread.sleep(d*1150); //pongo 1.15 segundos por cada bip
            //para poder hacer el test y que compruebe el boolean de sound_maker
            //es decir, le pongo menos tiempo por numero que en sound_maker (que son 700+500=1200)
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

}
```

Clase Beeper

Se explicará más detenidamente la clase Beeper.

- ESTADOS



La clase abstracta *State* que guarda una instancia de *Microwave* al igual que esta última de la primera. *State* es el nodo del patrón estado, de ella heredan cada tipo de estado, los cinco definidos anteriormente. Dependiendo del estado guardado en la clase *Microwave* una acción se ejecutará en una clase hija de estado u otra, aunque para todas hay acciones en común que se realizan en la clase *Microwave*.

Código.

```
public abstract class State {
    protected Microwave micro;

    public State(Microwave s) {
        micro = s;
    }

    public abstract void door_opened();

    public abstract void door_closed();

    public abstract void item_placed();

    public abstract void item_removed();

    public abstract void power_inc();

    public abstract void power_dec();

    public abstract void power_reset();

    public abstract void timer_inc();

    public abstract void timer_dec();

    public abstract void timer_reset();

    public abstract void cooking_start();

    public abstract void cooking_stop();

    public abstract void tick();
}
```

Clase State

Defino la clase abstracta State que declara los métodos que se tendrán que implementar en las clases hijas, es decir, los diferentes estados.

```
public class ClosedWithNoItem extends State {

    public ClosedWithNoItem(Microwave s) {
        super(s);
        micro.getHeating().heating_off();
        micro.getLamp().lamp_off();
        micro.getTurntable().turntable_stop();
    }

    public void door_opened() {
        micro.setStrategy(new OpenWithNoItem(micro));
    }

    public void door_closed() {}
    public void item_placed() {}

    public void item_removed() {
        micro.getDisplay().setDisplay("There is no item");
    }

    public void power_inc() {}
    public void power_dec() {}
    public void power_reset() {}
    public void timer_inc() {}
    public void timer_dec() {}
    public void timer_reset() {}

    public void cooking_start() {
        micro.getDisplay().setDisplay("Introduce Item");
    }

    public void cooking_stop() {}
    public void tick() {}

}
```

Clase ClosedWithNoItem

- Si se intenta quitar el objeto, indica que no hay objeto que quitar.
- Si se le da a cocinar, te indica en el display que falta un objeto dentro.
- Si se abre la puerta, el microondas cambia de estado.

```
public class OpenWithNoItem extends State{

    public OpenWithNoItem(Microwave s) {
        super(s);
        micro.getHeating().heating_off();
        micro.getLamp().lamp_on();
        micro.getTurntable().turntable_stop();
    }

    public void door_opened() {}

    public void door_closed() {
        micro.setStrategy(new ClosedWithNoItem(micro));
    }

    public void item_placed() {
        micro.setStrategy(new OpenWithItem(micro));
    }

    public void item_removed() {
        micro.getDisplay().setDisplay("There is no item");
    }

    public void power_inc() {}
    public void power_dec() {}
    public void power_reset() {}
    public void timer_inc() {}
    public void timer_dec() {}
    public void timer_reset() {}
    public void cooking_start() {
        micro.getDisplay().setDisplay("Introduce Item");
    }

    public void cooking_stop() {}

    public void tick() {
    }
```

OpenWithNoItem

- Si se intenta quitar el objeto, se indica en el display que no hay objeto.
- Se cambia de estado si se cierra la puerta o se introduce un objeto.
- Si se intenta cocinar se imprime que introduzca comida.

```
public class OpenWithItem extends State {

    public OpenWithItem(Microwave s) {
        super(s);
        micro.getHeating().heating_off();
        micro.getLamp().lamp_on();
        micro.getTurntable().turntable_stop();
    }

    public void door_opened() {}
    public void door_closed() {
        micro.setStrategy(new ClosedWithItem(micro));

    }

    public void item_placed() {
        micro.getDisplay().setDisplay("There is something inside");
    }


    public void item_removed() {
        micro.setStrategy(new OpenWithNoItem(micro));
    }

    public void power_inc() {}
    public void power_dec() {}
    public void power_reset() {}
    public void timer_inc() {}
    public void timer_dec() {}
    public void timer_reset() {}
    public void cooking_start() {
        micro.getDisplay().setDisplay("Close the door");
    }

    public void cooking_stop() {}
    public void tick() {}

}
```

OpenWithItem

- Si se intenta meter comida el microondas indica que ya hay comida dentro.
- Si se le da a cocinar, dice que se cierre la puerta antes.
- Se cambia de estado si se quita el objeto o se cierra la puerta.

```
public class ClosedWithItem extends State {

    public ClosedWithItem(Microwave s) {
        super(s);
        micro.getHeating().heating_off();
        micro.getLamp().lamp_off();
        micro.getTurntable().turntable_stop();
    }

    public void door_opened() {
        micro.setStrategy(new OpenWithItem(micro));
    }

    public void door_closed() {}
    public void item_placed() {}
    public void item_removed() {}
    public void power_inc() {}
    public void power_dec() {}
    public void power_reset() {}
    public void timer_inc() {}
    public void timer_dec() {}
    public void timer_reset() {}

    public void cooking_start() {
        micro.setStrategy(new Cooking(micro));
    }

    public void cooking_stop() {}
    public void tick() {}

}
```

ClosedWithItem

- Se cambia de estado si se abre la puerta o se le da a cocinar.

```java
import java.util.Timer;

public class Cooking extends State {

    Timer timer = new Timer();

    public Cooking(Microwave s) {
        super(s);
        micro.getHeating().heating_on();
        micro.getLamp().lamp_off();
        micro.getTurntable().turntable_start();
        micro.item_placed();
        micro.door_closed();
        cooking_start();
    }

    public void door_opened() {
        timer.cancel();
        micro.setCooking(false);
        micro.setStrategy(new OpenWithItem(micro));
    }

    public void door_closed() {}
    public void item_placed() {}
    public void item_removed() {}
    public void power_inc() {}

    public void power_dec() {
        timer.cancel();
        micro.setCooking(false);
        micro.setStrategy(new ClosedWithItem(micro));
    }

    public void power_reset() {
        timer.cancel();
        micro.setCooking(false);
        micro.setStrategy(new ClosedWithItem(micro));
    }
```

- Si se abre la puerta se deja de cocinar, se para el timer y se cambia de estado.
- Igual si se reduce la potencia o el tiempo a cero.

```
public void timer_inc() {}

public void timer_dec() {
    timer.cancel();
    micro.setCooking(false);
    micro.setStrategy(new ClosedWithItem(micro));
}

public void timer_reset() {
    timer.cancel();
    micro.setCooking(false);
    micro.setStrategy(new ClosedWithItem(micro));
    micro.getDisplay().clearDisplay();
}

public void cooking_start() {
    micro.setCooking(true);
    micro.getDisplay().clearDisplay();
    timer.scheduleAtFixedRate(new TimerTask() {

        public void run() {

            System.out.println("Time left: " + micro.getTimer());
            tick();

            if (micro.getTimer() < 0) {
                timer.cancel();
                micro.setCooking(false);
                micro.getBeeper().beep(3);
                micro.getDisplay().setDisplay("Enjoy your meal");
            }
        }
    }, 1000, 1000);

}
```

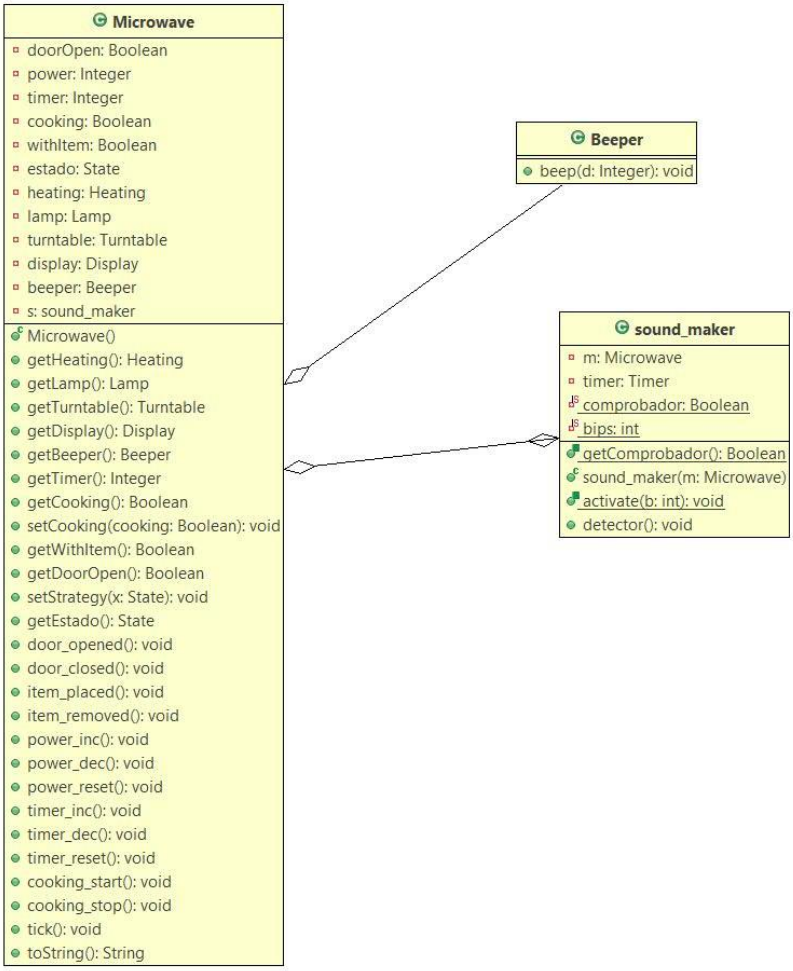
```
public void cooking_stop() {
    micro.setStrategy(new ClosedWithItem(micro));
    timer.cancel();
    micro.setCooking(false);

}

public void tick() {
    micro.getDisplay().setDisplay(micro.getTimer().toString());
    micro.tick();
}
```

Cooking

A la hora de ejecutar cooking_start(), se inicia un nuevo hilo de ejecución en el que se está cocinando y restando tiempo al contador pero sigue atento a cualquier otra orden que pueda solicitarse, como parar de cocinar, abrir la puerta, aumentar el tiempo…

El método tick(), que se llama dentro del nuevo hilo, reduce el valor del timer en 1 cada vez que se llama.

- BEEPER



Sound maker está implementado como parte del microondas y está relacionado con beeper porque llama a su método *actívate* con el parámetro de las veces que se hace el beep.

Código

```java
import java.util.Timer;

public class sound_maker {

    private Microwave m;
    private Timer timer = new Timer();
    private static Boolean comprobador = false;
    public static Boolean getComprobador() {
        return comprobador;
    }

    private static int bips;

    public sound_maker(Microwave m) {
        this.m = m;
        detector();
    }

    public static void activate(int b) {
        comprobador = true;
        bips = b;
    }

    public void detector() {
        timer.scheduleAtFixedRate(new TimerTask() {

            public void run() {
                if (comprobador) {
                    for (int i = 0; i < bips; i++) {
                        m.getDisplay().setDisplay("beep");
                        try {
                            Thread.sleep(700);
                            m.getDisplay().setDisplay("");
                            Thread.sleep(500);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                    comprobador = false;
                }
            }
        }, 0, 10);
    }

}
```

Detector comienza a funcionar en el momento que se crea el microondas ejecutándose una vez cada centésima comprobando si *beep(integer: d)* ha sido invocado. Si ha sido invocado, envía al display de microondas los “beeps”, tantos como el entero se ha pasado. Con intervalos de tiempos entre cada uno de ellos.

```java
public class Beeper {

    public void beep(Integer d) {
        sound_maker.activate(d);
        try {
            Thread.sleep(d*1150); //pongo 1.15 segundos por cada bip
            //para poder hacer el test y que compruebe el boolean de sound_maker
            //es decir, le pongo menos tiempo por numero que en sound_maker (que son 700+500=1200)
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

}
```

PRUEBAS UNITARIAS CON JUNIT

```java
import static org.junit.Assert.assertTrue;

public class Test_Microwave {

    private Microwave m;

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

    /**
     *
     * Test que comprueba si inicializa bien
     *
     */

    @Test
    void testConstructor() {
        assertTrue(!m.getDoorOpen());
        Assertions.assertTrue(!m.getCooking());
        Assertions.assertTrue(!m.getWithItem());
        Assertions.assertTrue(!m.getHeating().isHeating());
        Assertions.assertEquals(0, m.getHeating().getPower());
        Assertions.assertEquals(0, m.getTimer());
    }

    /**
     *
     * Test que comprueba que el Heating se inicia bien
     *
     */

    @Test
    void Heating() {
        Assertions.assertTrue(!m.getHeating().isHeating());
        m.getHeating().heating_on();
        Assertions.assertTrue(m.getHeating().isHeating());
        Assertions.assertEquals(0, m.getHeating().getPower());
        m.getHeating().setPower(25);
        Assertions.assertEquals(25, m.getHeating().getPower());
        m.getHeating().heating_off();
        Assertions.assertTrue(!m.getHeating().isHeating());
    }
```

Test que comprueba el constructor, si sus atributos de clase tienen valores iniciales razonables.

<u>TESTS SOBRE LOS COMPONENTES:</u>

Test que comprueba que el componente Heating se haya creado bien, apagado y con potencia a 0.

```java
@Test
void Lamparon() {
    Assertions.assertTrue(!m.getLamp().isLampOn());
    m.getLamp().lamp_on();
    Assertions.assertTrue(m.getLamp().isLampOn());
    m.getLamp().lamp_off();
    Assertions.assertTrue(!m.getLamp().isLampOn());
}
```

Comprueba que se inicia apagada y que se cambia eficazmente.

Comprueba que se inicia bien y que cambia de estado.

```
@Test
void Turntable() {
    Assertions.assertTrue(!m.getTurntable().isMoving());
    m.getTurntable().turntable_start();
    Assertions.assertTrue(m.getTurntable().isMoving());
    m.getTurntable().turntable_stop();
    Assertions.assertTrue(!m.getTurntable().isMoving());
}
```

```
@Test
void Beeper() {
    m.getBeeper().beep(3);
    Assertions.assertTrue(sound_maker.getComprobador());

}
```

Comprueba que si cuando se llama se cambia el valor de la constante de sound_maker

Envía distintos valores al display para confirmar que se actualiza y se usan los tres métodos que ofrece.

```
@Test
void Display() {
    Assertions.assertEquals(m.getDisplay().getDisplay(), "");
    m.getDisplay().setDisplay("Cocinando");
    Assertions.assertEquals(m.getDisplay().getDisplay(), "Cocinando");
    m.getDisplay().clearDisplay();
    Assertions.assertEquals(m.getDisplay().getDisplay(), "");
    m.getDisplay().clearDisplay();
    Assertions.assertEquals(m.getDisplay().getDisplay(), "");
    m.getDisplay().setDisplay(null);
    Assertions.assertEquals(m.getDisplay().getDisplay(), null);
    m.getDisplay().setDisplay("Fin");
    Assertions.assertEquals(m.getDisplay().getDisplay(), "Fin");

}
```

COMPROBACIÓN DE LOS ATRIBUTOS DE CLASE

```java
@Test
void Door() {

    // Básico abre y cierra
    Assertions.assertTrue(!m.getDoorOpen());
    m.door_opened();
    Assertions.assertTrue(m.getDoorOpen());
    m.door_closed();
    Assertions.assertTrue(!m.getDoorOpen());

    // Forzamos fallos
    for (int i = 0; i < 9; i++) {
        m.door_closed();
    }
    Assertions.assertTrue(!m.getDoorOpen());
    m.item_placed();
    m.power_inc();
    m.timer_inc();
    Assertions.assertTrue(!m.getDoorOpen());
    m.door_opened();
    for (int i = 0; i < 9; i++) {
        m.door_opened();
    }
    Assertions.assertTrue(m.getDoorOpen());
    m.cooking_stop();
    m.cooking_start();
    Assertions.assertTrue(m.getDoorOpen());
}
```

doorOpen

```java
@Test
void powerade() {

    Assertions.assertEquals(0, m.getHeating().getPower());
    m.power_inc();
    Assertions.assertEquals(100, m.getHeating().getPower());
    for (int i = 0; i < 9; i++) {
        m.power_dec();
    }
    Assertions.assertEquals(0, m.getHeating().getPower());
    for (int i = 0; i < 15; i++) {
        m.power_inc();
    }
    Assertions.assertEquals(1000, m.getHeating().getPower());
    m.power_reset();
    Assertions.assertEquals(0, m.getHeating().getPower());
    m.power_reset();
    Assertions.assertEquals(0, m.getHeating().getPower());

    // ¿le afectan otros métodos de microondas?
    m.power_inc();
    m.door_opened();
    m.door_closed();
    m.item_placed();
    m.item_removed();
    m.timer_inc();
    m.timer_dec();
    m.timer_reset();
    m.cooking_start();
    m.cooking_stop();
    Assertions.assertEquals(100, m.getHeating().getPower());

}
```

power

```java
@Test
void tiempo() {

    Assertions.assertEquals(0, m.getTimer());
    m.timer_inc();
    Assertions.assertEquals(30, m.getTimer());
    for (int i = 0; i < 9; i++) {
        m.timer_dec();
    }
    Assertions.assertEquals(0, m.getTimer());
    for (int i = 0; i < 15; i++) {
        m.timer_inc();
    }
    Assertions.assertEquals(450, m.getTimer());
    m.timer_reset();
    Assertions.assertEquals(0, m.getTimer());
    m.timer_reset();
    Assertions.assertEquals(0, m.getTimer());

    // ¿le afectan otros métodos de microondas?
    m.timer_inc();
    m.door_opened();
    m.door_closed();
    m.item_placed();
    m.item_removed();
    m.power_inc();
    m.power_dec();
    m.power_reset();
    m.cooking_start();
    m.cooking_stop();
    Assertions.assertEquals(30, m.getTimer());

    // cocinando
    m.door_opened();
    m.item_placed();
    m.door_closed();
    m.power_inc();
    m.cooking_start();
```

timer

```
    try {
        Thread.sleep(1010); // espera un segundo para poder comprobar el tiempo bien
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    m.cooking_stop();
    Assertions.assertEquals(29, m.getTimer());
    m.cooking_start();
    try {
        Thread.sleep(1010); // espera un segundo para poder comprobar el tiempo bien
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    m.power_reset();
    Assertions.assertEquals(28, m.getTimer());

}
```

Segunda parte del test a timer.

```
@Test
void cooking() {

    Assertions.assertTrue(!m.getCooking());
    m.timer_inc();
    m.power_inc();
    m.door_opened();
    m.item_placed();
    m.door_closed();
    m.cooking_start();
    Assertions.assertTrue(m.getCooking());
    m.cooking_stop();
    Assertions.assertTrue(!m.getCooking());
    m.cooking_start();
    Assertions.assertTrue(m.getCooking());
    m.door_opened();
    Assertions.assertTrue(!m.getCooking());
}
```

cooking

```
@Test
void withItem() {

    Assertions.assertTrue(!m.getWithItem());
    m.door_opened();
    m.item_placed();
    Assertions.assertTrue(m.getWithItem());
    m.item_removed();
    Assertions.assertTrue(!m.getWithItem());

}
```

withItem

- Se usan for para invocar métodos repetidas veces con tal de explotar el programa y buscar errores.
- En el test de timer se hacen esperas para comprobar que el timer disminuye conforme va cocinando.

COMPROBACIÓN DE LOS ESTADOS DEL MICROONDAS

```
@Test
void ClosedWithNoItem() {
    //Comprobación de sus componentes
    Assertions.assertFalse(m.getCooking());
    Assertions.assertFalse(m.getHeating().isHeating());
    Assertions.assertFalse(m.getLamp().isLampOn());
    Assertions.assertFalse(m.getTurntable().isMoving());
    Assertions.assertFalse(m.getDoorOpen());
    Assertions.assertFalse(m.getWithItem());

    Assertions.assertTrue(m.getEstado() instanceof ClosedWithNoItem);
    m.door_opened();
    Assertions.assertTrue(!(m.getEstado() instanceof ClosedWithNoItem));
    m.door_closed();
    Assertions.assertTrue(m.getEstado() instanceof ClosedWithNoItem);
    m.item_placed();
    Assertions.assertEquals("You need to open the door first", m.getDisplay().getDisplay());
    m.power_inc();
    Assertions.assertEquals("Power: 100", m.getDisplay().getDisplay());
    m.timer_inc();
    Assertions.assertEquals("Time: 30", m.getDisplay().getDisplay());
    m.cooking_start();
    Assertions.assertEquals("Introduce Item", m.getDisplay().getDisplay());
    m.item_removed();
    Assertions.assertEquals("You need to open the door first", m.getDisplay().getDisplay());

}
```

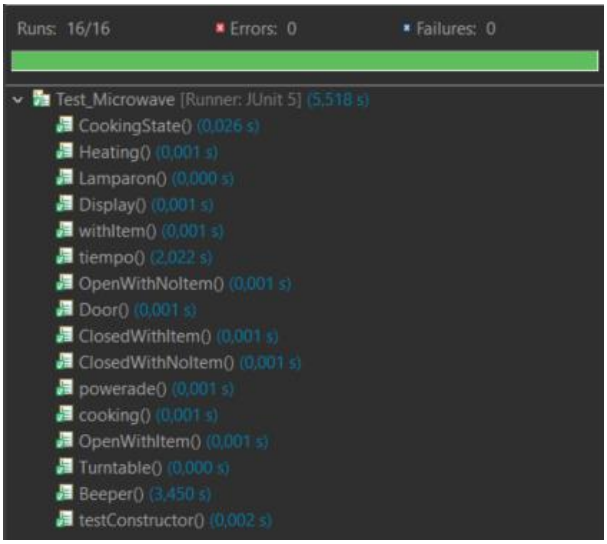
Primero comprueba los componentes en ese estado, después comprueba que es de tipo ClosedWithNoItem y por último a manejar los distintos métodos y a variar valores en busca de algún error.

```
@Test
void OpenWithNoItem() {
    m.door_opened();
    //Comprobación de sus componentes
    Assertions.assertFalse(m.getCooking());
    Assertions.assertFalse(m.getHeating().isHeating());
    Assertions.assertTrue(m.getLamp().isLampOn());
    Assertions.assertFalse(m.getTurntable().isMoving());
    Assertions.assertTrue(m.getDoorOpen());
    Assertions.assertFalse(m.getWithItem());

    m.door_closed();

    Assertions.assertTrue(!(m.getEstado() instanceof OpenWithNoItem));
    m.door_opened();
    Assertions.assertTrue(m.getEstado() instanceof OpenWithNoItem);
    m.item_placed();
    Assertions.assertTrue(!(m.getEstado() instanceof OpenWithNoItem));
    m.item_removed();
    Assertions.assertTrue(!m.getWithItem());
    m.power_inc();
    Assertions.assertEquals("Power: 100", m.getDisplay().getDisplay());
    m.timer_inc();
    Assertions.assertEquals("Time: 30", m.getDisplay().getDisplay());
    m.cooking_start();
    Assertions.assertEquals("Introduce Item", m.getDisplay().getDisplay());
    m.item_removed();
    Assertions.assertEquals("There is no item", m.getDisplay().getDisplay());

}
```

```java
@Test
void OpenWithItem() {
    m.door_opened();
    m.item_placed();
    //Comprobación de sus componentes
    Assertions.assertFalse(m.getCooking());
    Assertions.assertFalse(m.getHeating().isHeating());
    Assertions.assertTrue(m.getLamp().isLampOn());
    Assertions.assertFalse(m.getTurntable().isMoving());
    Assertions.assertTrue(m.getDoorOpen());
    Assertions.assertTrue(m.getWithItem());

    m.item_removed();
    m.door_closed();

    Assertions.assertTrue(!(m.getEstado() instanceof OpenWithItem));
    m.door_opened();
    m.item_placed();
    Assertions.assertTrue(m.getEstado() instanceof OpenWithItem);
    m.item_removed();
    Assertions.assertTrue(m.getEstado() instanceof OpenWithNoItem);
    m.item_placed();
    Assertions.assertTrue(m.getWithItem());
    m.power_inc();
    Assertions.assertEquals("Power: 100", m.getDisplay().getDisplay());
    m.timer_inc();
    Assertions.assertEquals("Time: 30", m.getDisplay().getDisplay());
    m.cooking_start();
    Assertions.assertEquals("Close the door", m.getDisplay().getDisplay());

}
```

```java
@Test
void ClosedWithItem() {
    m.door_opened();
    m.item_placed();
    m.door_closed();
    //Comprobación de sus componentes
    Assertions.assertFalse(m.getCooking());
    Assertions.assertFalse(m.getHeating().isHeating());
    Assertions.assertFalse(m.getLamp().isLampOn());
    Assertions.assertFalse(m.getTurntable().isMoving());
    Assertions.assertFalse(m.getDoorOpen());
    Assertions.assertTrue(m.getWithItem());

    m.door_opened();
    m.item_removed();
    m.door_closed();

    Assertions.assertFalse(m.getEstado() instanceof ClosedWithItem);
    m.door_opened();
    m.item_placed();
    m.door_closed();
    Assertions.assertTrue(m.getEstado() instanceof ClosedWithItem);
    m.item_removed();
    Assertions.assertEquals("You need to open the door first", m.getDisplay().getDisplay());
    m.item_placed();
    Assertions.assertEquals("You need to open the door first", m.getDisplay().getDisplay());
    Assertions.assertTrue(m.getWithItem());
    m.power_inc();
    Assertions.assertEquals("Power: 100", m.getDisplay().getDisplay());
    m.timer_inc();
    Assertions.assertEquals("Time: 30", m.getDisplay().getDisplay());
    m.cooking_start();
    Assertions.assertEquals(true, m.getCooking());

}
```

```java
@Test
void CookingState() {
    m.door_opened();
    m.item_placed();
    m.door_closed();
    m.timer_inc();
    m.power_inc();
    m.cooking_start();
    //Comprobación de sus componentes
    Assertions.assertTrue(m.getCooking());
    Assertions.assertTrue(m.getHeating().isHeating());
    Assertions.assertFalse(m.getLamp().isLampOn());
    Assertions.assertTrue(m.getTurntable().isMoving());
    Assertions.assertFalse(m.getDoorOpen());
    Assertions.assertTrue(m.getWithItem());

    m.door_opened();
    m.item_removed();
    m.door_closed();
    m.timer_reset();
    m.power_reset();

    Assertions.assertFalse(m.getEstado() instanceof Cooking);
    m.door_opened();
    m.item_placed();
    m.door_closed();
    m.timer_inc();
    m.power_inc();
    m.cooking_start();

    Assertions.assertTrue(m.getEstado() instanceof Cooking);
    m.item_removed();
    Assertions.assertEquals("You need to open the door first", m.getDisplay().getDisplay());
    m.item_placed();
    Assertions.assertEquals("You need to open the door first", m.getDisplay().getDisplay());
    Assertions.assertTrue(m.getWithItem());
    m.power_inc();
    Assertions.assertEquals("Power: 200", m.getDisplay().getDisplay());
    m.timer_inc();
    Assertions.assertEquals("Time: 60", m.getDisplay().getDisplay());
    m.cooking_start();
    Assertions.assertEquals(true, m.getCooking());
}
```



Se han ejecutado todos correctamente y han servido para arreglar bastantes errores.

TEST CON GHERKIN Y CUCUMBER

Cinco features se han creado, uno por cada estado del microondas.

- ClosedWithNoItem
- OpenWithNoItem
- OpenWithItem
- ClosedWithItem
- Cooking

1. Closed With No Item

```
Feature: A closed microwave without anything in it
  We want to test some posibilities of an empty closed microwave

  Scenario: We want to use a microwave
    Given a closed microwave with no item
    Then the door is closed
    And it is not heating
    And the lamp of the microwave is off
    And the turntable is still
    And it is not cooking right now
    And it is empty

    Scenario: We want to use a microwave
    Given a closed microwave with no item
    Then it is closed without anything in it

    Scenario: We want to use a microwave
    Given a closed microwave with no item
    When I open the microwave
    Then it is opened without anything in it

    Scenario: We want to use a microwave
    Given a closed microwave with no item
    When I introduce an item
    Then it warns me with 'You need to open the door first'

    Scenario: We want to use a microwave
    Given a closed microwave with no item
    When I press the time increment button
    Then it sets timer to 30 seconds

    Scenario: We want to use a microwave
    Given a closed microwave with no item
    When I press the power increment button
    Then it sets power to 100
```

```
Scenario Outline: We want to use a microwave
Given a closed microwave with no item
When I press power_inc button <a> times
Then the display should show "<b>"

 Examples:
  | a   | b          |
  | 1 | Power: 100 |
  | 2 | Power: 200 |
  | 5 | Power: 500 |
  | 10 | Power: 1000 |

Scenario Outline: We want to use a microwave
Given a closed microwave with no item
When I press time_inc button <a> times
Then the display should show "<b>"

 Examples:
  | a   | b          |
  | 1 | Time: 30 |
  | 2 | Time: 60 |
  | 3 | Time: 90 |
  | 4 | Time: 120 |

Scenario: We want to use a microwave
Given a closed microwave with no item
When I press the cook button
Then it warns me with 'Choose cooking power'

Scenario: We want to use a microwave
Given a closed microwave with no item
When I press the power increment button
And I press the time increment button
And I press the cook button
Then it warns me with 'Introduce Item'

Scenario: We want to use a microwave
Given a closed microwave with no item
When I press the power increment button
And I press the time increment button
And I press the power reset button
And I press the timer reset button
Then it sets power to 0
And it sets timer to 0 seconds
```

```
Scenario: We want to use a microwave
Given a closed microwave with no item
When I press the time increment button
And I press the time decrement button
And I press the time increment button
Then it sets timer to 30 seconds

Scenario: We want to use a microwave
Given a closed microwave with no item
When I press the power increment button
And I press the power decrement button
Then it sets power to 0

Scenario: We want to use a microwave
Given a closed microwave with no item
When I press the power decrement button
Then it sets power to 0

Scenario: We want to use a microwave
Given a closed microwave with no item
When I press the time decrement button
Then it sets timer to 0 seconds
```

2. Open With No Item

```
Feature: An opened microwave without anything in it
  We want to test some posibilities of an empty opened microwave

  Scenario: We want to use a microwave
    Given an opened microwave with no item
    Then the door is opened
    And it is not heating
    And the lamp of the microwave is on
    And the turntable is still
    And it is not cooking right now
    And it is empty

    Scenario: We want to use a microwave
    Given an opened microwave with no item
    Then it is opened without anything in it

    Scenario: We want to use a microwave
    Given an opened microwave with no item
    When I close the microwave
    Then it is closed without anything in it

    Scenario: We want to use a microwave
    Given an opened microwave with no item
    When I introduce an item
    Then it is opened with an item in it

    Scenario: We want to use a microwave
    Given an opened microwave with no item
    When I press the time increment button
    Then it sets timer to 30 seconds

    Scenario: We want to use a microwave
    Given an opened microwave with no item
    When I press the power increment button
    Then it sets power to 100
```

```
Scenario Outline: We want to use a microwave
Given an opened microwave with no item
When I press power_inc button <a> times
Then the display should show "<b>"

 Examples:
  | a   | b          |
  | 1 | Power: 100 |
  | 2 | Power: 200 |
  | 5 | Power: 500 |
  | 10 | Power: 1000 |

Scenario Outline: We want to use a microwave
Given an opened microwave with no item
When I press time_inc button <a> times
Then the display should show "<b>"

 Examples:
  | a   | b          |
  | 1 | Time: 30 |
  | 2 | Time: 60 |
  | 3 | Time: 90 |
  | 4 | Time: 120 |

Scenario: We want to use a microwave
Given an opened microwave with no item
When I press the time increment button
And I press the power increment button
And I press the cook button
Then it warns me with 'Introduce Item'

Scenario: We want to use a microwave
Given an opened microwave with no item
When I press the power increment button
And I press the time increment button
And I press the power reset button
And I press the timer reset button
Then it sets power to 0
And it sets timer to 0 seconds
```

```
Scenario: We want to use a microwave
Given an opened microwave with no item
When I press the time increment button
And I press the time decrement button
And I press the time increment button
Then it sets timer to 30 seconds

Scenario: We want to use a microwave
Given an opened microwave with no item
When I press the power increment button
And I press the power decrement button
Then it sets power to 0

Scenario: We want to use a microwave
Given an opened microwave with no item
When I press the power decrement button
Then it sets power to 0

Scenario: We want to use a microwave
Given an opened microwave with no item
When I press the time decrement button
Then it sets timer to 0 seconds
```

3. Open With Item

```
Feature: An opened microwave with an item inside it
  We want to test some posibilities of a non empty opened microwave

  Scenario: We want to use a microwave
    Given an opened microwave with an item inside it
    Then the door is opened
    And it is not heating
    And the lamp of the microwave is on
    And the turntable is still
    And it is not cooking right now
    And it is not empty

    Scenario: We want to use a microwave
    Given an opened microwave with an item inside it
    Then it is opened with an item in it

    Scenario: We want to use a microwave
    Given an opened microwave with an item inside it
    When I close the microwave
    Then it is closed with an item in it

    Scenario: We want to use a microwave
    Given an opened microwave with an item inside it
    When I introduce an item
    Then it warns me with 'There is something inside'

    Scenario: We want to use a microwave
    Given an opened microwave with an item inside it
    When I press the time increment button
    Then it sets timer to 30 seconds

    Scenario: We want to use a microwave
    Given an opened microwave with an item inside it
    When I press the power increment button
    Then it sets power to 100
```

```
Scenario Outline: We want to use a microwave
Given an opened microwave with an item inside it
When I press power_inc button <a> times
Then the display should show "<b>"

 Examples:
  | a   | b          |
  | 1 | Power: 100 |
  | 2 | Power: 200 |
  | 5 | Power: 500 |
  | 10 | Power: 1000 |

Scenario Outline: We want to use a microwave
Given an opened microwave with an item inside it
When I press time_inc button <a> times
Then the display should show "<b>"

 Examples:
  | a   | b          |
  | 1 | Time: 30 |
  | 2 | Time: 60 |
  | 3 | Time: 90 |
  | 4 | Time: 120 |

Scenario: We want to use a microwave
Given an opened microwave with an item inside it
When I press the time increment button
And I press the power increment button
And I press the cook button
Then it warns me with 'Close the door'

Scenario: We want to use a microwave
Given an opened microwave with an item inside it
When I press the power increment button
And I press the time increment button
And I press the power reset button
And I press the timer reset button
Then it sets power to 0
And it sets timer to 0 seconds
```

```
Scenario: We want to use a microwave
Given an opened microwave with an item inside it
When I press the time increment button
And I press the time decrement button
And I press the time increment button
Then it sets timer to 30 seconds

Scenario: We want to use a microwave
Given an opened microwave with an item inside it
When I press the power increment button
And I press the power decrement button
Then it sets power to 0

Scenario: We want to use a microwave
Given an opened microwave with an item inside it
When I press the power decrement button
Then it sets power to 0

Scenario: We want to use a microwave
Given an opened microwave with an item inside it
When I press the time decrement button
Then it sets timer to 0 seconds
```

4. Closed With Item

```
Feature: a closed microwave with an item inside it
  We want to test some posibilities of a closed microwave with food inside it

  Scenario: We want to use a microwave
    Given a closed microwave with an item inside it
    Then the door is closed
    And it is not heating
    And the lamp of the microwave is off
    And the turntable is still
    And it is not cooking right now
    And it is not empty

    Scenario: We want to use a microwave
    Given a closed microwave with an item inside it
    Then it is closed with an item in it

    Scenario: We want to use a microwave
    Given a closed microwave with an item inside it
    When I open the microwave
    Then it is opened with an item in it

    Scenario: We want to use a microwave
    Given a closed microwave with an item inside it
    When I introduce an item
    Then it warns me with 'You need to open the door first'

    Scenario: We want to use a microwave
    Given a closed microwave with an item inside it
    When I press the time increment button
    Then it sets timer to 30 seconds

    Scenario: We want to use a microwave
    Given a closed microwave with an item inside it
    When I press the power increment button
    Then it sets power to 100
```

```
Scenario Outline: We want to use a microwave
Given a closed microwave with an item inside it
When I press power_inc button <a> times
Then the display should show "<b>"

 Examples:
   | a   | b          |
   | 1 | Power: 100 |
   | 2 | Power: 200 |
   | 5 | Power: 500 |
   | 10 | Power: 1000 |

Scenario Outline: We want to use a microwave
Given a closed microwave with an item inside it
When I press time_inc button <a> times
Then the display should show "<b>"

 Examples:
   | a   | b          |
   | 1 | Time: 30 |
   | 2 | Time: 60 |
   | 3 | Time: 90 |
   | 4 | Time: 120 |

Scenario: We want to use a microwave
Given a closed microwave with an item inside it
When I press the time increment button
And I press the power increment button
And I press the cook button
Then it cooks

Scenario: We want to use a microwave
Given a closed microwave with an item inside it
When I press the power increment button
And I press the time increment button
And I press the power reset button
And I press the timer reset button
Then it sets power to 0
And it sets timer to 0 seconds
```

```
Scenario: We want to use a microwave
Given a closed microwave with an item inside it
When I press the time increment button
And I press the time decrement button
And I press the time increment button
Then it sets timer to 30 seconds

Scenario: We want to use a microwave
Given a closed microwave with an item inside it
When I press the power increment button
And I press the power decrement button
Then it sets power to 0

Scenario: We want to use a microwave
Given a closed microwave with an item inside it
When I press the power decrement button
Then it sets power to 0

Scenario: We want to use a microwave
Given a closed microwave with an item inside it
When I press the time decrement button
Then it sets timer to 0 seconds
```

5. Cooking

```
Feature: a microwave microwaving
  We want to test some posibilities of a microwave while cooking

  Scenario: We want to use a microwave
    Given a microwave microwaving
    Then the door is closed
    And it is heating
    And the lamp of the microwave is off
    And the turntable is moving
    And it cooks
    And it is not empty

    Scenario: We want to use a microwave
    Given a microwave microwaving
    Then it is cooking

    Scenario: We want to use a microwave
    Given a microwave microwaving
    When I open the microwave
    Then it is opened with an item in it

    Scenario: We want to use a microwave
    Given a microwave microwaving
    When I introduce an item
    Then it warns me with 'You need to open the door first'

    Scenario: We want to use a microwave
    Given a microwave microwaving
    When I press the time increment button
    Then it sets timer to 60 seconds

    Scenario: We want to use a microwave
    Given a microwave microwaving
    When I press the power increment button
    Then it sets power to 200
```

```
Scenario Outline: We want to use a microwave
Given a microwave microwaving
When I press power_inc button <a> times
Then the display should show "<b>"

 Examples:
  | a   | b          |
  | 1 | Power: 200 |
  | 2 | Power: 300 |
  | 5 | Power: 600 |
  | 10 | Power: 1000 |

Scenario Outline: We want to use a microwave
Given a microwave microwaving
When I press time_inc button <a> times
Then the display should show "<b>"

 Examples:
  | a   | b          |
  | 1 | Time: 60 |
  | 2 | Time: 90 |
  | 3 | Time: 120 |
  | 4 | Time: 150 |

Scenario: We want to use a microwave
Given a microwave microwaving
When I press the time increment button
And I press the power increment button
And I press the cook button
Then it cooks

Scenario: We want to use a microwave
Given a microwave microwaving
When I press the power increment button
And I press the time increment button
And I press the power reset button
And I press the timer reset button
Then it sets power to 0
And it sets timer to 0 seconds
And it is not cooking
```

```
Scenario: We want to use a microwave
Given a microwave microwaving
When I press the time increment button
And I press the time decrement button
And I press the time increment button
Then it sets timer to 60 seconds

Scenario: We want to use a microwave
Given a microwave microwaving
When I press the power increment button
And I press the power decrement button
Then it sets power to 100

Scenario: We want to use a microwave
Given a microwave microwaving
When I press the power decrement button
Then it sets power to 0
And it is not cooking

Scenario: We want to use a microwave
Given a microwave microwaving
When I press the time decrement button
Then it sets timer to 0 seconds
And it is not cooking
```

STEP DEFINITIONS

```java
package MicroCucumber;

import io.cucumber.java.en.Given;

public class StepDefinitions {
    private Microwave m;

    @Given ("a closed microwave with no item")
    public void a_closed_microwave_with_no_item() {
        m=new Microwave();
    }

    @Given ("an opened microwave with no item")
    public void opened_microwave_with_no_item() {
        m=new Microwave();
        m.door_opened();
    }

    @Given ("an opened microwave with an item inside it")
    public void opened_microwave_with_an_item() {
        m=new Microwave();
        m.door_opened();
        m.item_placed();
    }

    @Given ("a closed microwave with an item inside it")
    public void closed_microwave_with_an_item() {
        m=new Microwave();
        m.door_opened();
        m.item_placed();
        m.door_closed();
    }

    @Given ("a microwave microwaving")
    public void microwaving() {
        m=new Microwave();
        m.door_opened();
        m.item_placed();
        m.door_closed();
        m.timer_inc();
        m.power_inc();
        m.cooking_start();
    }
```

```java
@When ("I open the microwave")
public void open_microwave() {
    m.door_opened();
}

@When ("I close the microwave")
public void close_microwave() {
    m.door_closed();
}

@When ("I introduce an item")
public void insert_item() {
    m.item_placed();
}

@When ("I press the time decrement button")
public void time_dec() {
    m.timer_dec();
}

@When ("I press the time increment button")
public void time_inc() {
    m.timer_inc();
}

@When ("I press the power decrement button")
public void power_dec() {
    m.power_dec();
}

@When ("I press the power increment button")
public void power_inc() {
    m.power_inc();
}

@When ("I press power_inc button {int} times")
public void set_power(int p) {
    for (int i=0;i<p;i++) {
    m.power_inc();
    }
}
```

```java
@When ("I press time_inc button {int} times")
public void set_timer(int t) {
    for (int i=0;i<t;i++) {
        m.timer_inc();
    }
}

@When ("I press the cook button")
public void start_cooking() {
    m.cooking_start();
}

@When ("I press the power reset button")
public void power_reset() {
    m.power_reset();
}

@When ("I press the timer reset button")
public void timer_reset() {
    m.timer_reset();
}

@Then ("it is not heating")
public void not_heating() {
    Assertions.assertFalse(m.getHeating().isHeating());
}

@Then ("it is heating")
public void heating() {
    Assertions.assertTrue(m.getHeating().isHeating());
}

@Then ("the lamp of the microwave is off")
public void lamp_is_off() {
    Assertions.assertFalse(m.getLamp().isLampOn());
}

@Then ("the lamp of the microwave is on")
public void lamp_is_on() {
    Assertions.assertTrue(m.getLamp().isLampOn());
}
```

```java
@Then ("the turntable is still")
public void still_turntable() {
    Assertions.assertFalse(m.getTurntable().isMoving());
}

@Then ("the turntable is moving")
public void moving_turntable() {
    Assertions.assertTrue(m.getTurntable().isMoving());
}

@Then ("it is not cooking right now")
public void not_cooking() {
    Assertions.assertFalse(m.getCooking());
}

@Then ("it is empty")
public void is_empty() {
    Assertions.assertFalse(m.getWithItem());
}

@Then ("it is not empty")
public void not_empty() {
    Assertions.assertTrue(m.getWithItem());
}

@Then ("the door is closed")
public void the_door_is_closed() {
    Assertions.assertFalse(m.getDoorOpen());
}

@Then ("the door is opened")
public void the_door_is_opened() {
    Assertions.assertTrue(m.getDoorOpen());
}

@Then ("it is closed without anything in it")
public void state_ClosedWithNoItem() {
    Assertions.assertTrue(m.getEstado() instanceof ClosedWithNoItem);
}

@Then ("it is opened without anything in it")
public void state_OpenWithNoItem() {
    Assertions.assertTrue(m.getEstado() instanceof OpenWithNoItem);
}
```

```java
public void state_OpenWithItem() {
    Assertions.assertTrue(m.getEstado() instanceof OpenWithItem);
}

@Then ("it is closed with an item in it")
public void state_ClosedWithItem() {
    Assertions.assertTrue(m.getEstado() instanceof ClosedWithItem);
}

@Then ("it is cooking")
public void cooking() {
    Assertions.assertTrue(m.getEstado() instanceof Cooking);
}

@Then ("it warns me with {string}")
public void open_door_first(String s) {
    Assertions.assertEquals(s,m.getDisplay().getDisplay());
}

@Then ("it sets timer to {int} seconds")
public void tiempo(int t) {
    Assertions.assertEquals(t,m.getTimer());
}

@Then ("it sets power to {int}")
public void power(int p) {
    Assertions.assertEquals(p,m.getHeating().getPower());
}

@Then ("the display should show {string}")
public void display_power(String s) {
    Assertions.assertEquals(s,m.getDisplay().getDisplay());
}

@Then ("it cooks")
public void cocking() {
    Assertions.assertTrue(m.getCooking());
}

@Then ("it is not cooking")
public void notcocking() {
    Assertions.assertFalse(m.getCooking());
}
```

```
[INFO] Tests run: 101, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.798 s - in MicroCucumber.RunCucumberTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 101, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.868 s
[INFO] Finished at: 2022-06-09T17:07:28+02:00
[INFO] ------------------------------------------------------------------------
```

Todos los tests se han ejecutado correctamente, son básicamente los mismos que los implementados en JUNIT en el apartado anterior.

INTERFAZ GRÁFICA

Se ha implementado una interfaz gráfica que junta las tres interfaces indicadas en el ejercicio, se ha realizado más que nada para usarla como una ayuda en la implementación del microondas, dando posibilidad a encontrar gran cantidad de fallos.