PROYECTO FINAL:
MICROONDAS

SERGIO MARTÍN VERA

Apartado A:

Implementar el sistema en Java, usando el patrón de diseño Estado.

Como viene especificado en la descripción del apartado, se ha procedido a la implementación en Java del objeto Microondas.

Para este proyecto se ha seguido el diagrama UML proporcionado. Sin embargo, debido a la aplicación de Gherkin, he tenido que incluir métodos extra en la clase Microwave que permiten devolver y parametrizar algunas acciones propias del Microondas y garantizar el cumplimiento de las tareas.

Implementación interfaz: IMicrowave

```
package system;

public interface IMicrowave {

    public void door_opened(Microwave mw);

    public void door_closed(Microwave mw);

    public void item_placed(Microwave mw);

    public void item_removed(Microwave mw);

    public void power_dec(Microwave mw);

    public void power_reset(Microwave mw);

    public void timer_dec(Microwave mw);

    public void timer_reset(Microwave mw);

    public void cooking_start(Microwave mw);

    public void cooking_stop(Microwave mw);

    public void tick(Microwave mw);

}
```

Estos métodos han sido elegidos porque son importantes en el cambio de las fases de funcionamiento de nuestro microondas. Los métodos *door_opened()* y *door_closed()* pueden cambiar el estado del microondas (OpenWithItem ←→ ClosedWithItem, OpenWithNoItem ←→ ClosedWithNoItem), los métodos *item_placed()* y *ítem_removed()* también pueden cambiar el estado (OpenWithItem ←→ OpenWithNoItem) y los métodos *cooking_start()* y *cooking_stop()* también (Cooking ←→ ClosedWithItem).

En los métodos *power_dec(), power_reset(), timer_dec()* y *timer_reset()* si los valores de potencia o tiempo pasan a 0, pueden llegar a cambiar el estado del microondas porque puede considerar que ha terminado un proceso (Cooking ←→ ClosedWithItem).

El método *tick()* solo tendrá valor en el estado Cooking para indicar el fin de un proceso.

No todos los métodos pueden utilizarse en cada estado, es por eso que, aquellos sin sentido en cierto estado (*door_closed()* en estado ClosedWithItem), se encargaran de determinar excepciones.

Implementación clase: Microwave

De forma inicial, el microondas permanece con la puerta cerrada, sin objetos y con potencia y tiempo a 0.

```java
package system;

public class Microwave {

    private boolean doorOpen;
    private int power;
    private int timer;
    private boolean cooking;
    private boolean withItem;
    private IMicrowave status;
    private Heating heatingElement = new Heating();
    private Lamp lampElement = new Lamp();
    private Turntable turnableElement = new Turntable();
    private Beeper beeperElement = new Beeper();
    private Display displayElement = new Display();

    public Microwave() {
        doorOpen = false;
        power = 0;
        timer = 0;
        cooking = false;
        withItem = false;
        status = new ClosedWithNoItem(this);
    }

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

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

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

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

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

    public void power_dec() {
        status.power_dec(this);
    }

    public void power_reset() {
        status.power_reset(this);
        displayElement.setDisplay(Integer.toString(power));
    }
```

```java
    public void timer_inc() {
        timer++;
        displayElement.setDisplay(Integer.toString(timer));
    }

    public void timer_dec() {
        status.timer_dec(this);
    }

    public void timer_reset() {
        status.timer_reset(this);
        displayElement.setDisplay(Integer.toString(timer));
    }

    public void cooking_start() {
        status.cooking_start(this);
    }

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

    public void tick() {
        status.tick(this);
    }
```

```
    // EXTRA METHODS

    public boolean isDoorOpen() {
        return doorOpen;
    }

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

    public boolean isWithItem() {
        return withItem;
    }

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

    public int getPower() {
        return power;
    }

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

    public int getTimer() {
        return timer;
    }

    public void setTimer(int timer) {
        this.timer = timer;
    }

    public boolean isCooking() {
        return cooking;
    }

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

    public IMicrowave getStatus() {
        return status;
    }

    public void setStatus(IMicrowave status) {
        this.status = status;
    }

    public Heating getHeatingElement() {
        return heatingElement;
    }
```

```
    public Lamp getLampElement() {
        return lampElement;
    }

    public Turntable getTurntableElement() {
        return turnableElement;
    }

    public Beeper getBeeperElement() {
        return beeperElement;
    }

    public Display getDisplayElement() {
        return displayElement;
    }

}
```

Implementación clase: Heating

```java
package system;

public class Heating {

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

    public void heating_on() {
        heating = true;
    }

    public void heating_off() {
        heating = false;
    }

    public void setPower(int power) {
        if (power >= 0) {
            this.power = power;
        }
    }

    public boolean isHeating() {
        return heating;
    }

    public int getPower() {
        return power;
    }

}
```

Implementación clase: Lamp

```java
package system;

public class Lamp {

    private boolean lampOn = false;

    public void lamp_on() {
        lampOn = true;
    }

    public void lamp_off() {
        lampOn = false;
    }

    public boolean isLampOn() {
        return lampOn;
    }

}
```

Implementación clase: Turntable

```java
package system;

public class Turntable {

    private boolean turntableOn = false;

    public void turntable_start() {
        turntableOn = true;
    }

    public void turntable_stop() {
        turntableOn = false;
    }

    public boolean isMoving() {
        return turntableOn;
    }

}
```

Implementación clase: Beeper

```java
package system;

public class Beeper {

    public void beep(int b) {
        BeeperCounter.transfer(b);
    }

}
```

Se ha utilizado un patrón Observador para representar la funcionalidad del beeper. Para ello, se ha creado la clase **BeeperCounter**. Esta clase se encarga de recibir el párametro utilizado en la clase *beep(int b)* de **Beeper**, lo guarda con el método *transfer(int tb)* y comprueba que se han realizado los avisos esperados con el método *beeperSound(int t)*.

```java
package system;

public class BeeperCounter {

    private static int beeps = 0;

    public static void transfer(int tb) {
        beeps = tb;
    }

    // Reset the counter
    public static boolean beeperSound(int t) {
        int bp = beeps;
        beeps = 0;
        return (bp == t);
    }

}
```

Implementación clase: Display

```java
package system;

public class Display {

    private String display;

    public void clearDisplay() {
        display = null;
    }

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

    public String getDisplay() {
        return display;
    }

}
```

Implementación clase: Fase 1 (ClosedWithNoItem)

```java
package system;

public class ClosedWithItem implements IMicrowave {

    public ClosedWithItem(Microwave mw) {
        mw.getLampElement().lamp_off();
        mw.getHeatingElement().heating_off();
        mw.getTurntableElement().turntable_stop();
        mw.setCooking(false);
        mw.setDoorOpen(false);
        mw.setWithItem(true);
    }

    @Override
    public void door_opened(Microwave mw) {
        mw.setStatus(new OpenWithItem(mw));
    }

    @Override
    public void door_closed(Microwave mw) {
        throw new IllegalStateException("Error: Door already closed");
    }

    @Override
    public void item_placed(Microwave mw) {
        throw new IllegalStateException("Error: Door closed");
    }

    @Override
    public void item_removed(Microwave mw) {
        throw new IllegalStateException("Error: Door closed");
    }

    @Override
    public void power_dec(Microwave mw) {
        if (mw.getPower() > 0) {
            mw.setPower(mw.getPower() - 1);
            mw.getDisplayElement().setDisplay(Integer.toString(mw.getPower()));
        }
    }

    @Override
    public void power_reset(Microwave mw) {
        mw.setPower(0);
    }
```

```java
    @Override
    public void timer_dec(Microwave mw) {
        if (mw.getTimer() > 0) {
            mw.setTimer(mw.getTimer() - 1);
            mw.getDisplayElement().setDisplay(Integer.toString(mw.getTimer()));
        }
    }


    @Override
    public void timer_reset(Microwave mw) {
        mw.setTimer(0);
    }

    @Override
    public void cooking_start(Microwave mw) {
        if(mw.getTimer() > 0 && mw.getPower() > 0) {
            mw.setStatus(new Cooking(mw));
        } else if (mw.getTimer() > 0) {
            throw new IllegalStateException("Error: Power is 0");
        } else {
            throw new IllegalStateException("Error: Timer is 0");
        }
    }

    @Override
    public void cooking_stop(Microwave mw) {
        throw new IllegalStateException("Error: Microwave was not cooking");
    }

    @Override
    public void tick(Microwave mw) {
        throw new IllegalStateException("Error: Microwave not cooking");
    }

}
```

Implementación clase: Fase 2 (OpenWithNoItem)

```java
package system;

public class OpenWithNoItem implements IMicrowave {

    public OpenWithNoItem(Microwave mw) {
        mw.getLampElement().lamp_on();
        mw.getHeatingElement().heating_off();
        mw.getTurntableElement().turntable_stop();
        mw.setCooking(false);
        mw.setWithItem(false);
        mw.setDoorOpen(true);
    }

    @Override
    public void door_opened(Microwave mw) {
        throw new IllegalStateException("Error: Door already opened");
    }

    @Override
    public void door_closed(Microwave mw) {
        mw.setStatus(new ClosedWithNoItem(mw));
    }

    @Override
    public void item_placed(Microwave mw) {
        mw.setStatus(new OpenWithItem(mw));
    }

    @Override
    public void item_removed(Microwave mw) {
        throw new IllegalStateException("Error: No item there before");
    }

    @Override
    public void power_dec(Microwave mw) {
        if (mw.getPower() > 0) {
            mw.setPower(mw.getPower() - 1);
            mw.getDisplayElement().setDisplay(Integer.toString(mw.getPower()));
        }
    }

    @Override
    public void power_reset(Microwave m) {
        m.setPower(0);
    }
```

```java
    @Override
    public void timer_dec(Microwave mw) {
        if (mw.getTimer() > 0) {
            mw.setTimer(mw.getTimer() - 1);
            mw.getDisplayElement().setDisplay(Integer.toString(mw.getTimer()));
        }
    }

    @Override
    public void timer_reset(Microwave mw) {
        mw.setTimer(0);
    }

    @Override
    public void cooking_start(Microwave mw) {
        throw new IllegalStateException("Error: Can't star with door opened");
    }

    @Override
    public void cooking_stop(Microwave mw) {
        throw new IllegalStateException("Error: Microwave not cooking");
    }

    @Override
    public void tick(Microwave mw) {
        throw new IllegalStateException("Error: Microwave not cooking");
    }

}
```

Implementación clase: Fase 3 (OpenWithItem)

```java
package system;

public class OpenWithItem implements IMicrowave {

    public OpenWithItem(Microwave mw) {
        mw.getLampElement().lamp_on();
        mw.getHeatingElement().heating_off();
        mw.getTurntableElement().turntable_stop();
        mw.setCooking(false);
        mw.setWithItem(true);
        mw.setDoorOpen(true);
    }

    @Override
    public void door_opened(Microwave mw) {
        throw new IllegalStateException("Error: Door already opened");
    }

    @Override
    public void door_closed(Microwave mw) {
        mw.setStatus(new ClosedWithItem(mw));
    }

    @Override
    public void item_placed(Microwave mw) {
        throw new IllegalStateException("Error: Microwave is full");
    }

    @Override
    public void item_removed(Microwave mw) {
        mw.setStatus(new OpenWithNoItem(mw));
    }

    @Override
    public void power_dec(Microwave m) {
        if (m.getPower() > 0) {
            m.setPower(m.getPower() - 1);
            m.getDisplayElement().setDisplay(Integer.toString(m.getPower()));
        }
    }

    @Override
    public void power_reset(Microwave m) {
        m.setPower(0);
    }
```

```java
    @Override
    public void power_reset(Microwave m) {
        m.setPower(0);
    }

    @Override
    public void timer_dec(Microwave m) {
        if (m.getTimer() > 0) {
            m.setTimer(m.getTimer() - 1);
            m.getDisplayElement().setDisplay(Integer.toString(m.getTimer()));
        }
    }

    @Override
    public void timer_reset(Microwave mw) {
        mw.setTimer(0);
    }

    @Override
    public void cooking_start(Microwave mw) {
        throw new IllegalStateException("Error: Can't star with door opened");
    }

    @Override
    public void cooking_stop(Microwave mw) {
        throw new IllegalStateException("Error: Microwave not cooking");
    }

    @Override
    public void tick(Microwave m) {
        throw new IllegalStateException("Error: Microwave not cooking");
    }

}
```

Implementación clase: Fase 4 (ClosedWithItem)

```java
package system;

public class ClosedWithItem implements IMicrowave {

    public ClosedWithItem(Microwave mw) {
        mw.getLampElement().lamp_off();
        mw.getHeatingElement().heating_off();
        mw.getTurntableElement().turntable_stop();
        mw.setCooking(false);
        mw.setDoorOpen(false);
        mw.setWithItem(true);
    }

    @Override
    public void door_opened(Microwave mw) {
        mw.setStatus(new OpenWithItem(mw));
    }

    @Override
    public void door_closed(Microwave mw) {
        throw new IllegalStateException("Error: Door already closed");
    }

    @Override
    public void item_placed(Microwave mw) {
        throw new IllegalStateException("Error: Door closed");
    }

    @Override
    public void item_removed(Microwave mw) {
        throw new IllegalStateException("Error: Door closed");
    }

    @Override
    public void power_dec(Microwave mw) {
        if (mw.getPower() > 0) {
            mw.setPower(mw.getPower() - 1);
            mw.getDisplayElement().setDisplay(Integer.toString(mw.getPower()));
        }
    }

    @Override
    public void power_reset(Microwave mw) {
        mw.setPower(0);
    }
```

```java
    @Override
    public void timer_dec(Microwave mw) {
        if (mw.getTimer() > 0) {
            mw.setTimer(mw.getTimer() - 1);
            mw.getDisplayElement().setDisplay(Integer.toString(mw.getTimer()));
        }
    }


    @Override
    public void timer_reset(Microwave mw) {
        mw.setTimer(0);
    }

    @Override
    public void cooking_start(Microwave mw) {
        if(mw.getTimer() > 0 && mw.getPower() > 0) {
            mw.setStatus(new Cooking(mw));
        } else if (mw.getTimer() > 0) {
            throw new IllegalStateException("Error: Power is 0");
        } else {
            throw new IllegalStateException("Error: Timer is 0");
        }
    }

    @Override
    public void cooking_stop(Microwave mw) {
        throw new IllegalStateException("Error: Microwave was not cooking");
    }

    @Override
    public void tick(Microwave mw) {
        throw new IllegalStateException("Error: Microwave not cooking");
    }

}
```

Implementación clase: Fase 5 (Cooking)

```java
package system;

public class Cooking implements IMicrowave{

    public Cooking(Microwave mw) {
        mw.getLampElement().lamp_on();
        mw.getHeatingElement().setPower(mw.getPower());
        mw.getHeatingElement().heating_on();
        mw.getTurntableElement().turntable_start();
        mw.setCooking(true);
        mw.setDoorOpen(false);
        mw.setWithItem(true);
    }
    @Override
    public void door_opened(Microwave mw) {
        mw.setStatus(new OpenWithItem(mw));
    }

    @Override
    public void door_closed(Microwave mw) {
        throw new IllegalStateException("Error: Door already closed");
    }

    @Override
    public void item_placed(Microwave mw) {
        throw new IllegalStateException("Error: Door closed");
    }

    @Override
    public void item_removed(Microwave mw) {
        throw new IllegalStateException("Error: Door closed");
    }

    @Override
    public void power_dec(Microwave mw) {
        if (mw.getPower() > 0) {
            mw.setPower(mw.getPower() - 1);
            mw.getDisplayElement().setDisplay(Integer.toString(mw.getPower()));
        } if(mw.getPower() == 0) {
            cooking_stop(mw);
        }
    }

    @Override
    public void power_reset(Microwave mw) {
        mw.setStatus(new ClosedWithItem(mw));
        mw.setPower(0);
    }
```

```java
    @Override
    public void timer_dec(Microwave mw) {
        if (mw.getTimer() > 0) {
            mw.setTimer(mw.getTimer() - 1);
            mw.getDisplayElement().setDisplay(Integer.toString(mw.getTimer()));
        } if (mw.getTimer() == 0) {
            mw.getBeeperElement().beep(3);
            mw.getDisplayElement().setDisplay("Item ready");
            cooking_stop(mw);
        }
    }

    @Override
    public void timer_reset(Microwave mw) {
        mw.setStatus(new ClosedWithItem(mw));
        mw.setTimer(0);
    }

    @Override
    public void cooking_start(Microwave mw) {
        throw new IllegalStateException("Error: Microwave already cooking");
    }

    @Override
    public void cooking_stop(Microwave mw) {
        mw.setStatus(new ClosedWithItem(mw));
    }

    @Override
    public void tick(Microwave mw) {
        if (mw.getTimer() > 1) {
            mw.timer_dec();
        } else {
            mw.timer_dec();
            mw.getBeeperElement().beep(3);
            mw.getDisplayElement().setDisplay("Item ready");
            cooking_stop(mw);
        }
```

Apartado B:

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

Implementación Tests con Junit: MicrowaveTest

```java
package system;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class MicrowaveTest {

    private Microwave mw = new Microwave();

    // Test for Heating element
    @Test
    public void heatingTest() {
        Heating h = new Heating();

        // Initial status
        assertEquals(0, h.getPower());
        assertFalse(h.isHeating());

        // Correct power input
        h.setPower(180);
        assertEquals(180, h.getPower());
        h.setPower(0);
        assertEquals(0, h.getPower());

        // Sequence of power on and power off
        h.heating_on();
        assertTrue(h.isHeating());
        h.heating_off();
        assertFalse(h.isHeating());

    }

    // Test for Lamp element
    @Test
    public void lampTest() {
        Lamp p = new Lamp();

        // Initial status
        assertFalse(p.isLampOn());

        // Sequence of lamp on and lamp off
        assertFalse(p.isLampOn());
        p.lamp_on();
        assertTrue(p.isLampOn());
        p.lamp_off();
        assertFalse(p.isLampOn());
    }
```

```java
// Test for Turntable element
@Test
public void turntableTest() {
    Turntable t = new Turntable();

    // Initial status
    assertFalse(t.isMoving());

    // Turn sequence
    assertFalse(t.isMoving());
    t.turntable_start();
    assertTrue(t.isMoving());
    t.turntable_stop();
    assertFalse(t.isMoving());
}

// Test for Beeper element
@Test
public void beeperTest() {
    Beeper p = new Beeper();

    // Correct beeper input
    p.beep(5);
    assertTrue(BeeperCounter.beeperSound(5));

    // After 1 iteration, beeper resets
    assertTrue(BeeperCounter.beeperSound(0));
}

// Test for Display element
@Test
public void displayTest() {
    Display d = new Display();

    // Initial status
    assertNull(d.getDisplay());

    // Set and clear display
    d.setDisplay("Message");
    assertEquals("Message", d.getDisplay());
    d.clearDisplay();
    assertNull(d.getDisplay());
}

// Methods to increase and decrease the power and timer
private void power_incPlus(int p) {
    for (int i = 0; i < p; i++) {
        mw.power_inc();
    }
}
```

```java
private void power_decPlus(int p) {
    for (int i = 0; i < p; i++) {
        mw.power_dec();
    }
}

private void timer_incPlus(int t) {
    for (int i = 0; i < t; i++) {
        mw.timer_inc();
    }
}

private void timer_decPlus(int t) {
    for (int i = 0; i < t; i++) {
        mw.timer_dec();
    }
}

private void timer_works(int t) {
    for (int i = 0; i < t; i++) {
        mw.tick();
    }
}
```

```java
    // Test for timer and power from Microwave
    @Test
    public void setupTest() {

        mw.timer_reset();
        mw.power_reset();
        mw.timer_dec();
        mw.power_dec();

        // Setup cycle for power
        assertEquals(0, mw.getPower());
        power_incPlus(200);
        assertEquals(200, mw.getPower());
        assertEquals("200", mw.getDisplayElement().getDisplay());
        power_decPlus(100);
        assertEquals(100, mw.getPower());
        assertEquals("100", mw.getDisplayElement().getDisplay());
        mw.power_reset();
        assertEquals(0, mw.getPower());
        assertEquals("0", mw.getDisplayElement().getDisplay());

        // Setup cycle for timer
        assertEquals(0, mw.getTimer());
        timer_incPlus(60);
        assertEquals(60, mw.getTimer());
        assertEquals("60", mw.getDisplayElement().getDisplay());
        timer_decPlus(30);
        assertEquals(30, mw.getTimer());
        assertEquals("30", mw.getDisplayElement().getDisplay());
        mw.timer_reset();
        assertEquals(0, mw.getTimer());
        assertEquals("0", mw.getDisplayElement().getDisplay());

    }
```

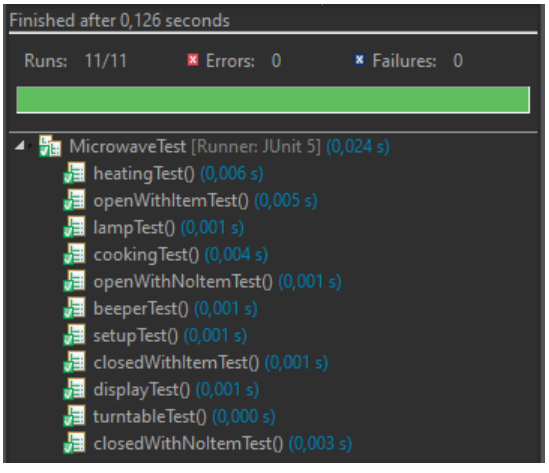
```java
// Phase 1: Test for a ClosedWithNoItem situation
@Test
public void closedWithNoItemTest() {

    // Exceptions check
    assertThrows(IllegalStateException.class, () -> mw.door_closed());
    assertThrows(IllegalStateException.class, () -> mw.cooking_start());
    assertThrows(IllegalStateException.class, () -> mw.cooking_stop());
    assertThrows(IllegalStateException.class, () -> mw.item_placed());
    assertThrows(IllegalStateException.class, () -> mw.item_removed());
    assertThrows(IllegalStateException.class, () -> mw.tick());

    // Inner status check
    assertFalse(mw.isCooking());
    assertFalse(mw.isWithItem());
    assertFalse(mw.isDoorOpen());
    assertFalse(mw.getHeatingElement().isHeating());
    assertFalse(mw.getLampElement().isLampOn());
    assertFalse(mw.getTurntableElement().isMoving());
    assertTrue(mw.getStatus() instanceof ClosedWithNoItem);

}

// Phase 2: Test for an OpenWithNoItem situation
@Test
public void openWithNoItemTest() {
    if (mw.getStatus() instanceof ClosedWithNoItem) {
        mw.door_opened();
    }

    // Exceptions check
    assertThrows(IllegalStateException.class, () -> mw.door_opened());
    assertThrows(IllegalStateException.class, () -> mw.cooking_start());
    assertThrows(IllegalStateException.class, () -> mw.cooking_stop());
    assertThrows(IllegalStateException.class, () -> mw.item_removed());
    assertThrows(IllegalStateException.class, () -> mw.tick());

    // Inner status check
    assertFalse(mw.isCooking());
    assertFalse(mw.isWithItem());
    assertTrue(mw.isDoorOpen());
    assertFalse(mw.getHeatingElement().isHeating());
    assertTrue(mw.getLampElement().isLampOn());
    assertFalse(mw.getTurntableElement().isMoving());
    assertTrue(mw.getStatus() instanceof OpenWithNoItem);

    // Timer and power check
    setupTest();

    // Test works when closing door
    mw.door_closed();
    closedWithNoItemTest();
}
```

```java
// Phase 3: Test for an OpenWithItem situation
@Test
public void openWithItemTest() {
    if (mw.getStatus() instanceof ClosedWithNoItem) {
        mw.door_opened();
        mw.item_placed();
    }

    // Exceptions check
    assertThrows(IllegalStateException.class, () -> mw.door_opened());
    assertThrows(IllegalStateException.class, () -> mw.cooking_start());
    assertThrows(IllegalStateException.class, () -> mw.cooking_stop());
    assertThrows(IllegalStateException.class, () -> mw.item_placed());
    assertThrows(IllegalStateException.class, () -> mw.tick());

    // Inner status check
    assertFalse(mw.isCooking());
    assertTrue(mw.isWithItem());
    assertTrue(mw.isDoorOpen());
    assertFalse(mw.getHeatingElement().isHeating());
    assertTrue(mw.getLampElement().isLampOn());
    assertFalse(mw.getTurntableElement().isMoving());
    assertTrue(mw.getStatus() instanceof OpenWithItem);

    // Timer and power check
    setupTest();

    // Test works when removing item
    mw.item_removed();
    assertTrue(mw.getStatus() instanceof OpenWithNoItem);
    openWithNoItemTest();
}
```

```java
// Phase 4: Test for a ClosedWithItem situation
@Test
public void closedWithItemTest() {
    if (mw.getStatus() instanceof ClosedWithNoItem) {
        mw.door_opened();
        mw.item_placed();
        mw.door_closed();
    }

    // Exceptions check
    assertThrows(IllegalStateException.class, () -> mw.door_closed());
    assertThrows(IllegalStateException.class, () -> mw.cooking_stop());
    assertThrows(IllegalStateException.class, () -> mw.item_placed());
    assertThrows(IllegalStateException.class, () -> mw.item_removed());
    assertThrows(IllegalStateException.class, () -> mw.tick());

    // Inner status check
    assertFalse(mw.isCooking());
    assertTrue(mw.isWithItem());
    assertFalse(mw.isDoorOpen());
    assertFalse(mw.getHeatingElement().isHeating());
    assertFalse(mw.getLampElement().isLampOn());
    assertFalse(mw.getTurntableElement().isMoving());
    assertTrue(mw.getStatus() instanceof ClosedWithItem);

    // Timer and power check
    setupTest();

    // Test works when opening door
    mw.door_opened();
    assertTrue(mw.getStatus() instanceof OpenWithItem);
    openWithItemTest();
}
```

```java
    // Phase 5: Test for a Cooking situation
    @Test
    public void cookingTest() {
        if (mw.getStatus() instanceof ClosedWithNoItem) {
            mw.door_opened();
            mw.item_placed();
            mw.door_closed();
        }

        // Can't start cooking with wrong inputs
        mw.timer_reset();
        assertThrows(IllegalStateException.class, () -> mw.cooking_start());
        timer_incPlus(15);
        assertThrows(IllegalStateException.class, () -> mw.cooking_start());
        mw.timer_reset();
        power_incPlus(50);
        assertThrows(IllegalStateException.class, () -> mw.cooking_start());
        mw.power_reset();

        // Start cooking
        timer_incPlus(20);
        power_incPlus(100);
        mw.cooking_start();

        // Exceptions check
        assertThrows(IllegalStateException.class, () -> mw.door_closed());
        assertThrows(IllegalStateException.class, () -> mw.cooking_start());
        assertThrows(IllegalStateException.class, () -> mw.item_placed());
        assertThrows(IllegalStateException.class, () -> mw.item_removed());

        // Inner status check
        assertTrue(mw.isCooking());
        assertTrue(mw.isWithItem());
        assertFalse(mw.isDoorOpen());
        assertTrue(mw.getHeatingElement().isHeating());
        assertTrue(mw.getLampElement().isLampOn());
        assertTrue(mw.getTurntableElement().isMoving());
        assertTrue(mw.getStatus() instanceof Cooking);

        // Microwave stops when opening door
        mw.door_opened();
        assertTrue(mw.getStatus() instanceof OpenWithItem);
        assertFalse(BeeperCounter.beeperSound(3));
        openWithItemTest();

        // Restart process
        mw.door_opened();
        mw.item_placed();
        mw.door_closed();
        timer_incPlus(10);
        power_incPlus(100);
        mw.cooking_start();

        // Microwave stop when timer ends
        assertEquals(10, mw.getTimer());
        timer_works(10);
        assertEquals(0, mw.getTimer());
        assertTrue(mw.getStatus() instanceof ClosedWithItem);
        assertEquals("Item ready", mw.getDisplayElement().getDisplay());
        assertTrue(BeeperCounter.beeperSound(3));

        // Test works when ending cooking
        closedWithItemTest();
    }

}
```

Resultados del test realizado:



Apartado C:

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

Se ha implementado una feature por cada estado del microondas (5).

Implementación feature: ClosedWithNoItemMW

```
Feature: Microwave closed with no item

  Scenario: Open the door of a closed microwave
    Given A closed microwave with no item
    When Open the door
    Then Door opens
    And Heating doesn't heats
    And Lamp turns on
    And Turntable doesn't turns

  Scenario Outline: Set power in a microwave
    Given A closed microwave with no item
    When Set the power with <a> W
    Then Display shows "<b>"

    Examples:
      | a   | b   |
      | -10 |   0 |
      |   0 |   0 |
      |  10 |  10 |

  Scenario: Reset power in a microwave
    Given A closed microwave with no item
    When Reset the power
    Then Power goes to zero

  Scenario Outline: Set timer in a microwave
    Given A closed microwave with no item
    When Set the timer with <a> seconds
    Then Display shows "<b>"

    Examples:
      | a   | b   |
      | -60 |   0 |
      |   0 |   0 |
      |  60 |  60 |


  Scenario: Reset timer in a microwave
    Given A closed microwave with no item
    When Reset the timer
    Then Timer goes to zero

  Scenario: Trying to cook in a closed microwave with no item
    Given A closed microwave with no item
    When Set the power with 800 W
    And Set the timer with 60 seconds
    And Set microwave to start cooking
    Then Microwave doesn't start cooking
```

Resultados:

```
10 Scenarios (10 passed)
35 Steps (35 passed)
0m0,453s
```

Implementación feature: OpenWithNoItemMW

```
Feature: Microwave opened with no item

  Scenario: Close an opened microwave
    Given An opened microwave with no item
    When Close the door
    Then Door closes
    And Heating doesn't heats
    And Lamp turns off
    And Turntable doesn't turns

  Scenario: Placing item in microwave
    Given An opened microwave with no item
    When Place an item
    Then Microwave has an item

  Scenario Outline: Set power in a microwave
    Given An opened microwave with no item
    When Set the power with <a> W
    Then Display shows "<b>"

    Examples:
      | a   | b   |
      | -10 |   0 |
      |   0 |   0 |
      |  10 |  10 |

  Scenario: Reset power in a microwave
    Given An opened microwave with no item
    When Reset the power
    Then Power goes to zero

  Scenario Outline: Set timer in a microwave
    Given An opened microwave with no item
    When Set the timer with <a> seconds
    Then Display shows "<b>"

    Examples:
      | a   | b   |
      | -60 |   0 |
      |   0 |   0 |
      |  60 |  60 |


  Scenario: Reset timer in a microwave
    Given An opened microwave with no item
    When Reset the timer
    Then Timer goes to zero

  Scenario: Trying to cook in an opened microwave with no item
    Given An opened microwave with no item
    When Set the power with 800 W
    And Set the timer with 60 seconds
    And Set microwave to start cooking
    Then Microwave doesn't start cooking
```

Resultados:

```
11 Scenarios (11 passed)
38 Steps (38 passed)
0m0,456s
```

Implementación feature: OpenWithItemMW

```
Feature: Microwave opened with item

  Scenario: Close an opened microwave
    Given An opened microwave with item
    When Close the door
    Then Door closes
    And Heating doesn't heats
    And Lamp turns off
    And Turntable doesn't turns

  Scenario: Removing item from microwave
    Given An opened microwave with item
    When Remove an item
    Then Microwave has no item

  Scenario Outline: Set power in a microwave
    Given An opened microwave with item
    When Set the power with <a> W
    Then Display shows "<b>"

    Examples:
      | a   | b   |
      | -10 |   0 |
      |   0 |   0 |
      |  10 |  10 |

  Scenario: Reset power in a microwave
    Given An opened microwave with item
    When Reset the power
    Then Power goes to zero

  Scenario Outline: Set timer in a microwave
    Given An opened microwave with item
    When Set the timer with <a> seconds
    Then Display shows "<b>"

    Examples:
      | a   | b   |
      | -60 |   0 |
      |   0 |   0 |
      |  60 |  60 |


  Scenario: Reset timer in a microwave
    Given An opened microwave with item
    When Reset the timer
    Then Timer goes to zero

  Scenario: Trying to cook in an opened microwave with item
    Given An opened microwave with item
    When Set the power with 800 W
    And Set the timer with 60 seconds
    And Set microwave to start cooking
    Then Microwave doesn't start cooking
```

Resultados:

```
11 Scenarios (11 passed)
38 Steps (38 passed)
0m0,451s
```

Implementación feature: ClosedWithItemMW

```
Feature: Microwave closed with item

  Scenario: Open the door of a closed microwave
    Given A closed microwave with item
    When Open the door
    Then Door opens
    And Heating doesn't heats
    And Lamp turns on
    And Turntable doesn't turns

  Scenario Outline: Set power in a microwave
    Given A closed microwave with item
    When Set the power with <a> W
    Then Display shows "<b>"

    Examples:
      | a   | b   |
      | -10 |   0 |
      |   0 |   0 |
      |  10 |  10 |

  Scenario: Reset power in a microwave
    Given A closed microwave with item
    When Reset the power
    Then Power goes to zero

  Scenario Outline: Set timer in a microwave
    Given A closed microwave with item
    When Set the timer with <a> seconds
    Then Display shows "<b>"

    Examples:
      | a   | b   |
      | -60 |   0 |
      |   0 |   0 |
      |  60 |  60 |


  Scenario: Reset timer in a microwave
    Given A closed microwave with item
    When Reset the timer
    Then Timer goes to zero

  Scenario: Trying to cook in a closed microwave with item
    Given A closed microwave with item
    When Set the power with 800 W
    And Set the timer with 60 seconds
    And Set microwave to start cooking
    Then Microwave start cooking
```

Resultados:

```
10 Scenarios (10 passed)
35 Steps (35 passed)
0m0,454s
```

Implementación feature: CookingMW

```
Feature: Microwave cooking

  Scenario: Start cooking in a closed microwave with item
    Given A closed microwave with item
    When Set the power with 800 W
    And Set the timer with 60 seconds
    And Set microwave to start cooking
    Then Microwave start cooking
    And Heating heats
    And Lamp turns on
    And Turntable turns

  Scenario: Increase power while cooking
    Given A cooking microwave with 200 power and 60 timer
    When Increase the power
    Then Display shows "201"

  Scenario: Decrease power while cooking
    Given A cooking microwave with 200 power and 60 timer
    When Decrease the power
    Then Display shows "199"

  Scenario: End cooking resetting the power
    Given A cooking microwave with 200 power and 60 timer
    When Reset the power
    Then Microwave not cooking
    And Heating doesn't heats
    And Lamp turns off
    And Turntable doesn't turns
    And Display shows "0"

  Scenario: Increase time while cooking
    Given A cooking microwave with 200 power and 60 timer
    When Increase the timer
    Then Display shows "61"
    And Timer has 61 seconds

  Scenario: Decrease time while cooking
    Given A cooking microwave with 200 power and 60 timer
    When Decrease the timer
    Then Display shows "59"
    And Timer has 59 seconds

  Scenario: End cooking resetting the timer
    Given A cooking microwave with 200 power and 60 timer
    When Reset the timer
    Then Microwave not cooking
    And Heating doesn't heats
    And Lamp turns off
    And Turntable doesn't turns
    And Display shows "0"
```

```
  Scenario: Stop cooking opening the door
    Given A cooking microwave with 200 power and 60 timer
    When Open the door
    Then Door opens
    And Microwave not cooking
    And Heating doesn't heats
    And Lamp turns on
    And Turntable doesn't turns

  Scenario Outline: Cooking time finishes correctly
    Given A closed microwave with item
    When Set the power with <a> W
    And Set the timer with <b> seconds
    And Set microwave to start cooking
    Then Microwave start cooking
    And Timer has <c> seconds
    Then Microwave not cooking
    And Heating doesn't heats
    And Lamp turns off
    And Turntable doesn't turns
    And Beeper sounds 3 times
    And Display shows "Item ready"

    Examples:
      | a   | b   | c   |
      | 100 |  30 |  30 |
      | 200 |  60 |  60 |
      | 500 | 120 | 120 |
```

Resultados:

```
11 Scenarios (11 passed)
79 Steps (79 passed)
0m0,471s
```

Enlace a repositorio en GitHub

https://github.com/smv762e/Microwave_Project.git