PROYECTO MICROONDAS

Alejandro Pascual Mellado

Microondas

En este proyecto, se va a desarrollar un microondas siguiendo el patrón estado.

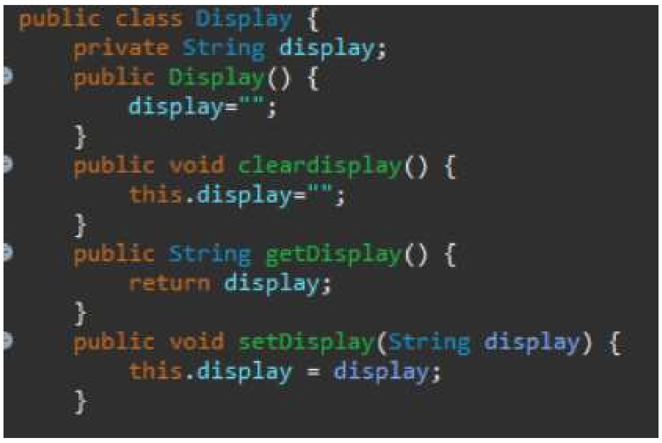
Este contara con distintos componentes que le darán funcionalidades como un plato giratorio, una lampara, un calentador o una bocina.

Además, tendrá un funcionamiento distinto en función del estado en el que se encuentre (Dependiendo de si la puerta esta cerrada o abierta, de si tiene un alimento en su interior o de si está funcionando)

Procedo a mostrar todo el código del proyecto:

Clase Display

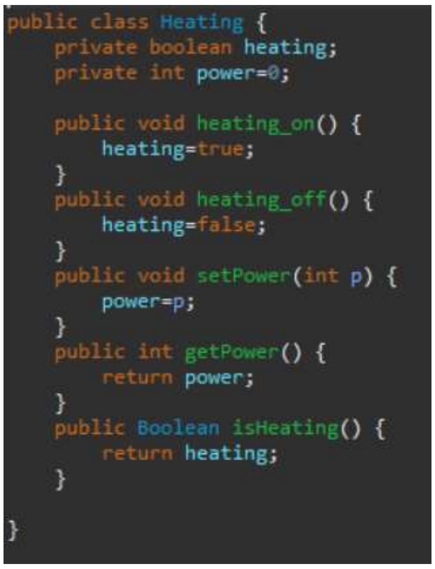
Pantalla que muestra al usuario distintos datos:



```java
public class Display {
    private String display;
    public Display() {
        display="";
    }
    public void cleardisplay() {
        this.display="";
    }
    public String getDisplay() {
        return display;
    }
    public void setDisplay(String display) {
        this.display = display;
    }
```

Clase Heating

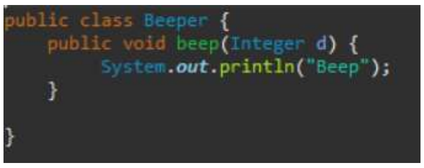
Motor de calor del microondas:



```java
public class Heating {
    private boolean heating;
    private int power=0;

    public void heating_on() {
        heating=true;
    }
    public void heating_off() {
        heating=false;
    }
    public void setPower(int p) {
        power=p;
    }
    public int getPower() {
        return power;
    }
    public Boolean isHeating() {
        return heating;
    }

}
```

Clase Beeper

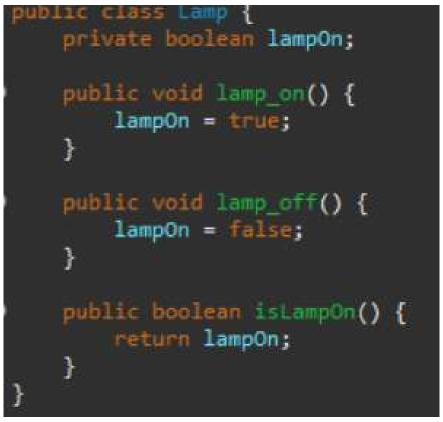
Bocina de alarma:



```
public class Beeper {
    public void beep(Integer d) {
        System.out.println("Beep");
    }

}
```

Clase Lamp

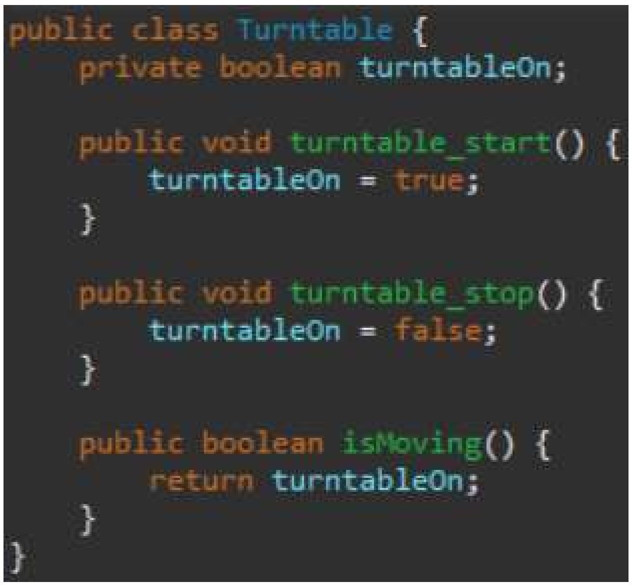
Bombilla que ilumina el interior:



```
public class Lamp {
    private boolean lampOn;

    public void lamp_on() {
        lampOn = true;
    }

    public void lamp_off() {
        lampOn = false;
    }

    public boolean isLampOn() {
        return lampOn;
    }
}
```

Clase Turntable

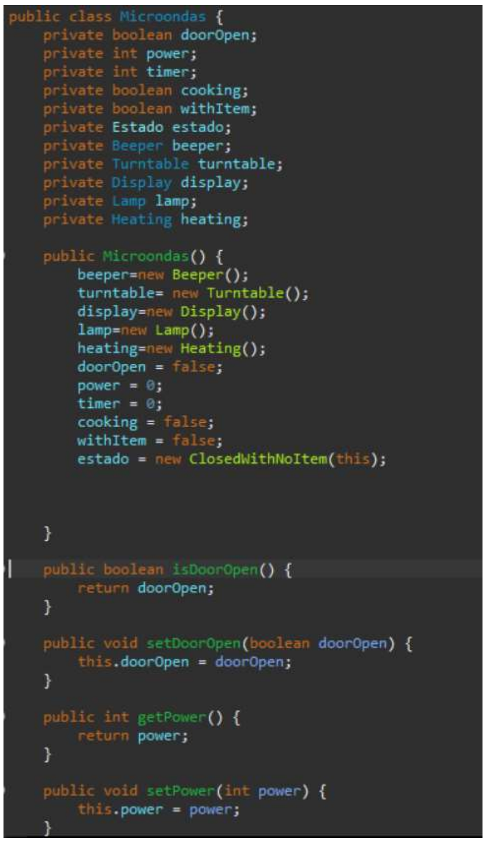
Plato giratorio:



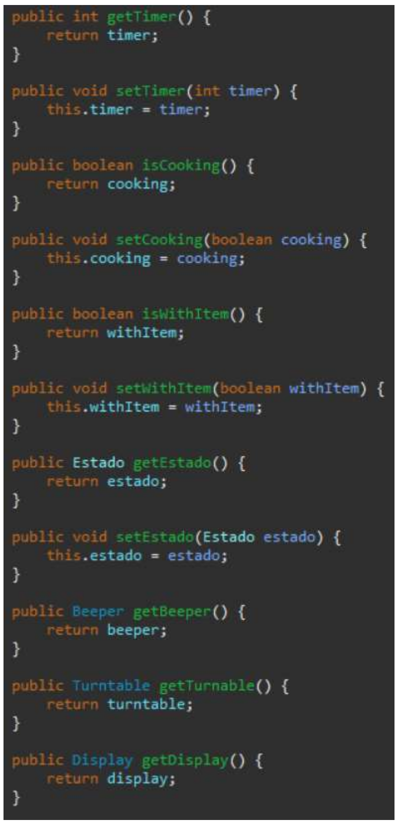
```
public class Turntable {
    private boolean turntableOn;

    public void turntable_start() {
        turntableOn = true;
    }

    public void turntable_stop() {
        turntableOn = false;
    }

    public boolean isMoving() {
        return turntableOn;
    }
}
```

Clase Microondas

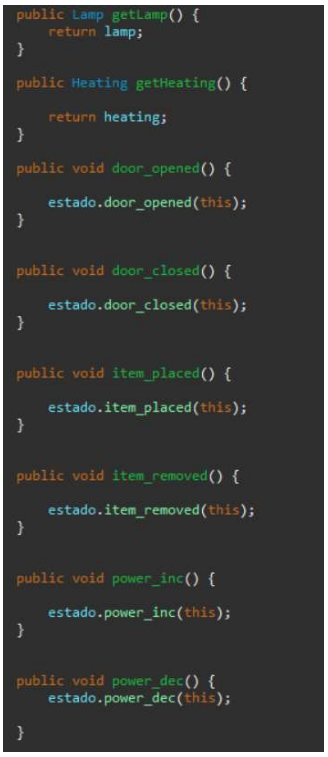
El producto principal, almacena todas las otras clases y tiene funcionalidades mas complejas que las anteriores:



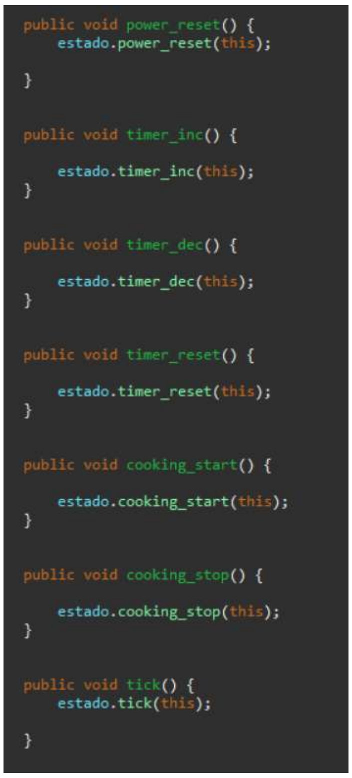
```java
public class Microondas {
    private boolean doorOpen;
    private int power;
    private int timer;
    private boolean cooking;
    private boolean withItem;
    private Estado estado;
    private Beeper beeper;
    private Turntable turntable;
    private Display display;
    private Lamp lamp;
    private Heating heating;

    public Microondas() {
        beeper=new Beeper();
        turntable= new Turntable();
        display=new Display();
        lamp=new Lamp();
        heating=new Heating();
        doorOpen = false;
        power = 0;
        timer = 0;
        cooking = false;
        withItem = false;
        estado = new ClosedWithNoItem(this);


    }

    public boolean isDoorOpen() {
        return doorOpen;
    }

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

    public int getPower() {
        return power;
    }

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



```java
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 boolean isWithItem() {
    return withItem;
}

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

public Estado getEstado() {
    return estado;
}

public void setEstado(Estado estado) {
    this.estado = estado;
}

public Beeper getBeeper() {
    return beeper;
}

public Turntable getTurnable() {
    return turntable;
}

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



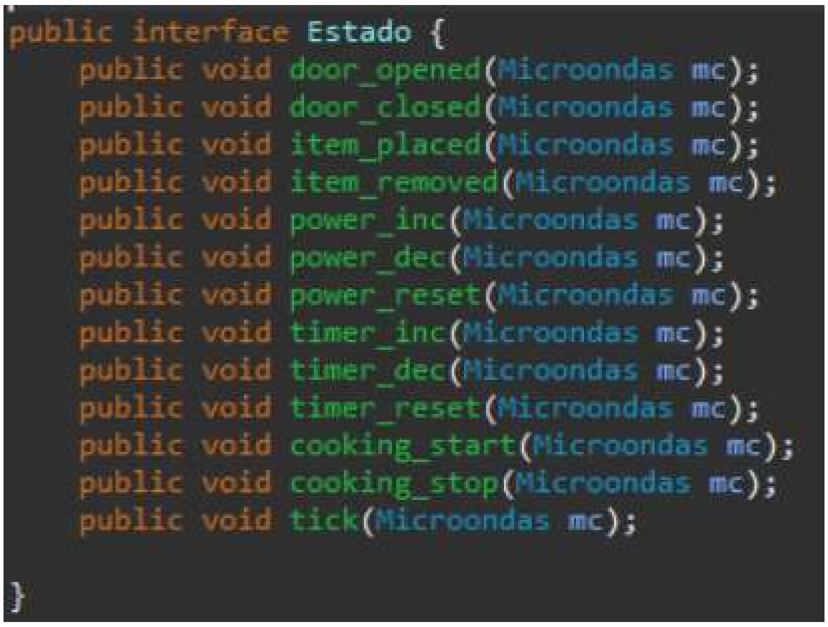
```java
public Lamp getLamp() {
    return lamp;
}

public Heating getHeating() {

    return heating;
}

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() {

    estado.power_inc(this);
}

public void power_dec() {
    estado.power_dec(this);

}
```



```java
public void power_reset() {
    estado.power_reset(this);

}

public void timer_inc() {

    estado.timer_inc(this);
}

public void timer_dec() {

    estado.timer_dec(this);
}

public void timer_reset() {

    estado.timer_reset(this);
}

public void cooking_start() {

    estado.cooking_start(this);
}

public void cooking_stop() {

    estado.cooking_stop(this);
}

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

}
```

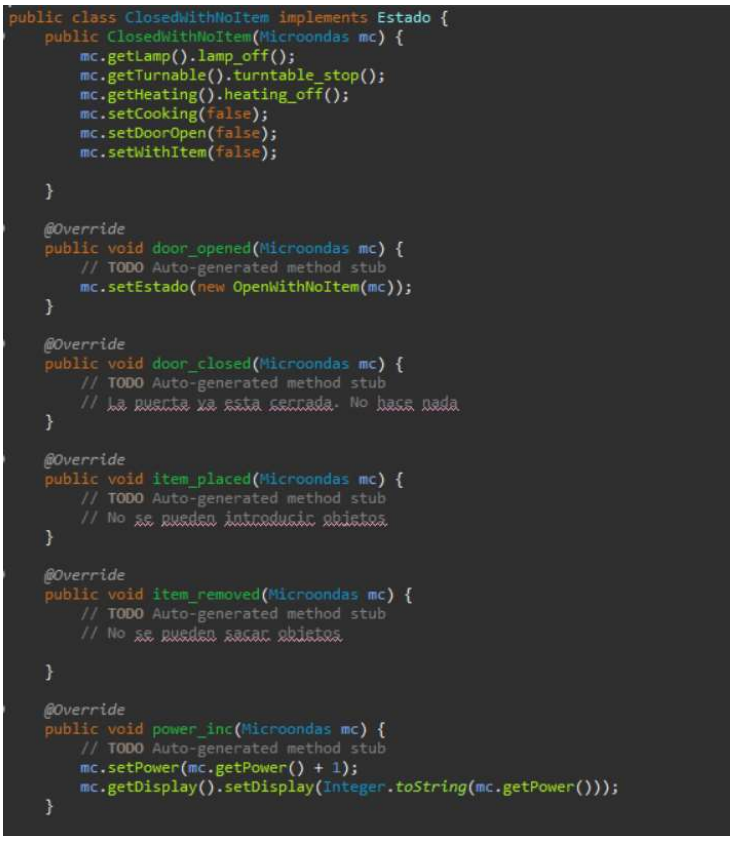
Interfaz estado

Interfaz de la que heredaran todos los posibles estados:

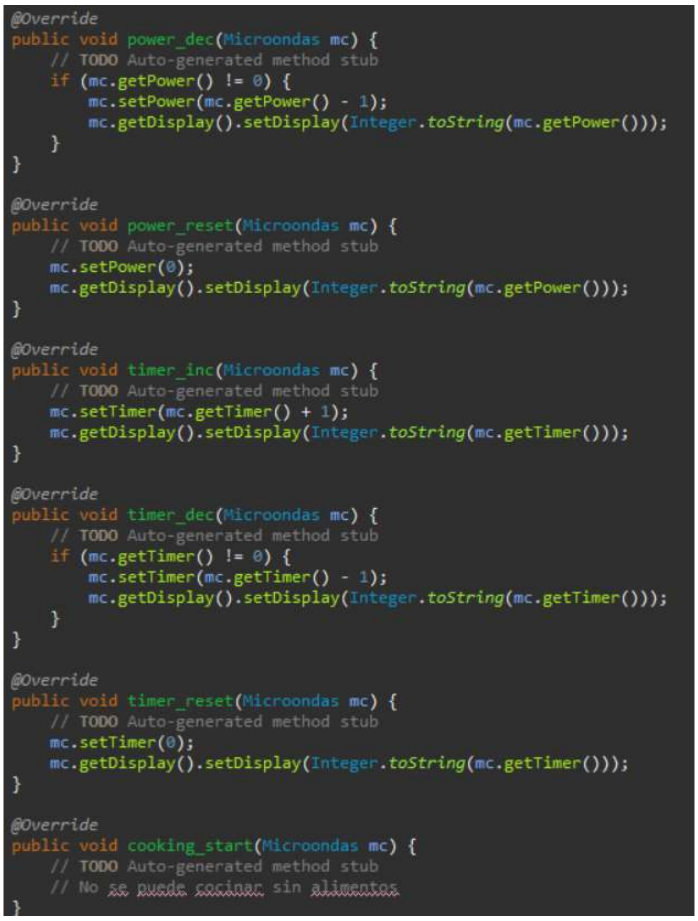


```java
public interface Estado {
    public void door_opened(Microondas mc);
    public void door_closed(Microondas mc);
    public void item_placed(Microondas mc);
    public void item_removed(Microondas mc);
    public void power_inc(Microondas mc);
    public void power_dec(Microondas mc);
    public void power_reset(Microondas mc);
    public void timer_inc(Microondas mc);
    public void timer_dec(Microondas mc);
    public void timer_reset(Microondas mc);
    public void cooking_start(Microondas mc);
    public void cooking_stop(Microondas mc);
    public void tick(Microondas mc);

}
```

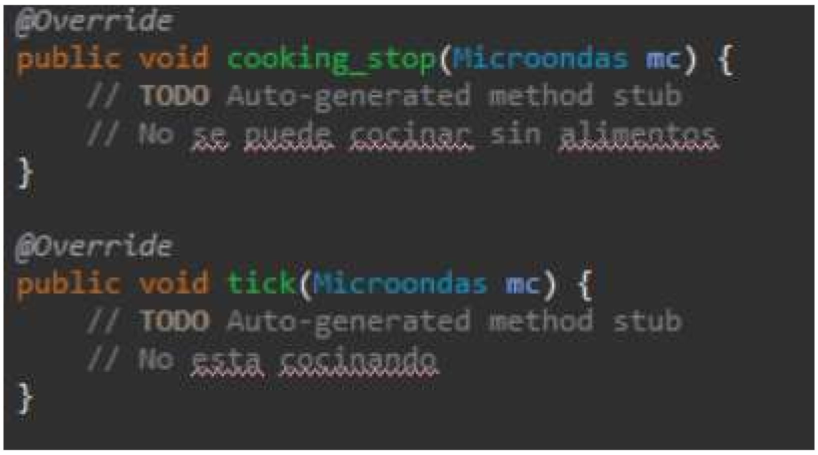
Estado Cerrado Sin item



```java
public class ClosedWithNoItem implements Estado {
    public ClosedWithNoItem(Microondas mc) {
        mc.getLamp().lamp_off();
        mc.getTurnable().turntable_stop();
        mc.getHeating().heating_off();
        mc.setCooking(false);
        mc.setDoorOpen(false);
        mc.setWithItem(false);

    }

    @Override
    public void door_opened(Microondas mc) {
        // TODO Auto-generated method stub
        mc.setEstado(new OpenWithNoItem(mc));
    }

    @Override
    public void door_closed(Microondas mc) {
        // TODO Auto-generated method stub
        // La puerta ya esta cerrada. No hace nada
    }

    @Override
    public void item_placed(Microondas mc) {
        // TODO Auto-generated method stub
        // No se pueden introducir objetos
    }

    @Override
    public void item_removed(Microondas mc) {
        // TODO Auto-generated method stub
        // No se pueden sacar objetos

    }

    @Override
    public void power_inc(Microondas mc) {
        // TODO Auto-generated method stub
        mc.setPower(mc.getPower() + 1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
    }
```

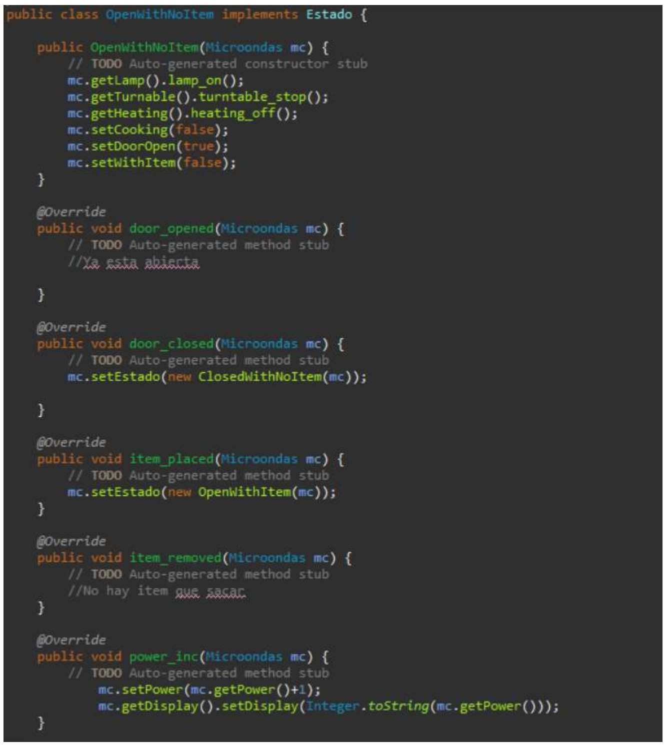


```java
@Override
public void power_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getPower() != 0) {
        mc.setPower(mc.getPower() - 1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
    }
}

@Override
public void power_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setPower(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
}

@Override
public void timer_inc(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setTimer(mc.getTimer() + 1);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}

@Override
public void timer_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getTimer() != 0) {
        mc.setTimer(mc.getTimer() - 1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
    }
}

@Override
public void timer_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setTimer(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}

@Override
public void cooking_start(Microondas mc) {
    // TODO Auto-generated method stub
    // No se puede cocinar sin alimentos
}
```

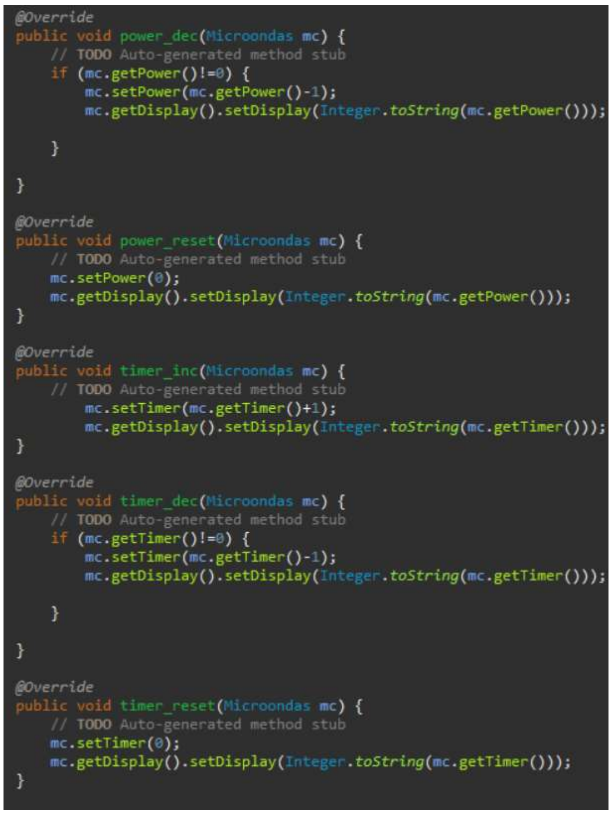


```java
@Override
public void cooking_stop(Microondas mc) {
    // TODO Auto-generated method stub
    // No se puede cocinar sin alimentos
}

@Override
public void tick(Microondas mc) {
    // TODO Auto-generated method stub
    // No esta cocinando
}
```

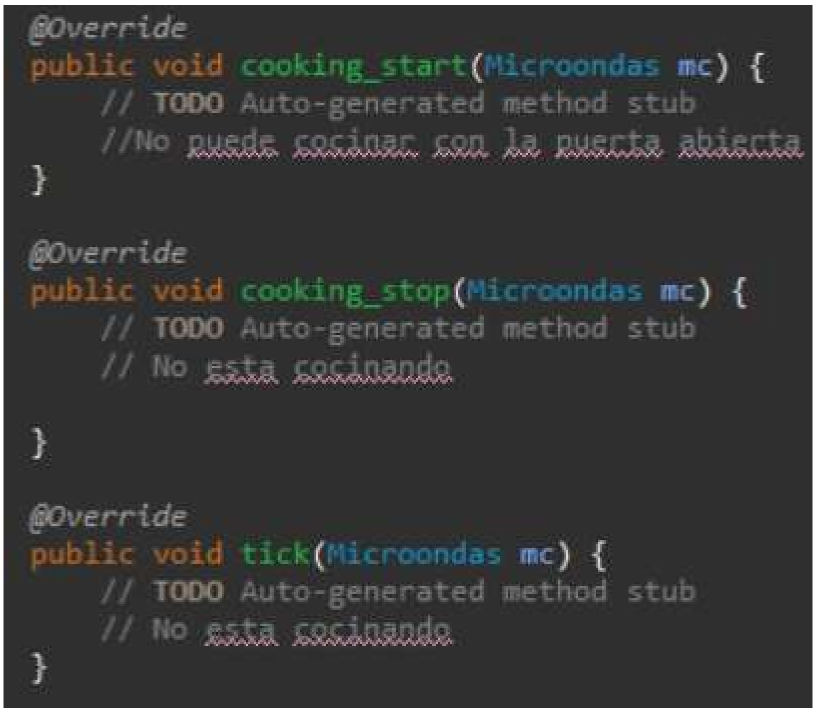
Estado Abierto Sin item



```java
public class OpenWithNoItem implements Estado {

    public OpenWithNoItem(Microondas mc) {
        // TODO Auto-generated constructor stub
        mc.getLamp().lamp_on();
        mc.getTurnable().turntable_stop();
        mc.getHeating().heating_off();
        mc.setCooking(false);
        mc.setDoorOpen(true);
        mc.setWithItem(false);
    }

    @Override
    public void door_opened(Microondas mc) {
        // TODO Auto-generated method stub
        //Ya esta abierta

    }

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

    }

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

    @Override
    public void item_removed(Microondas mc) {
        // TODO Auto-generated method stub
        //No hay item que sacar
    }

    @Override
    public void power_inc(Microondas mc) {
        // TODO Auto-generated method stub
            mc.setPower(mc.getPower()+1);
            mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
    }
```

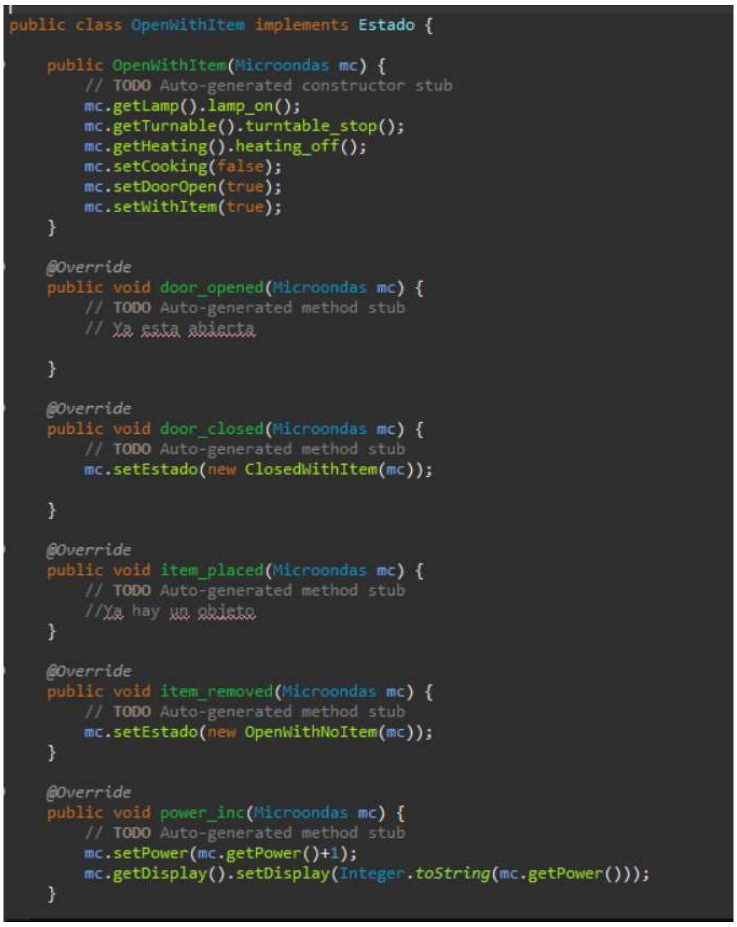


```java
@Override
public void power_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getPower()!=0) {
        mc.setPower(mc.getPower()-1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));

    }

}

@Override
public void power_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setPower(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
}

@Override
public void timer_inc(Microondas mc) {
    // TODO Auto-generated method stub
        mc.setTimer(mc.getTimer()+1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}

@Override
public void timer_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getTimer()!=0) {
        mc.setTimer(mc.getTimer()-1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));

    }

}

@Override
public void timer_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setTimer(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}
```

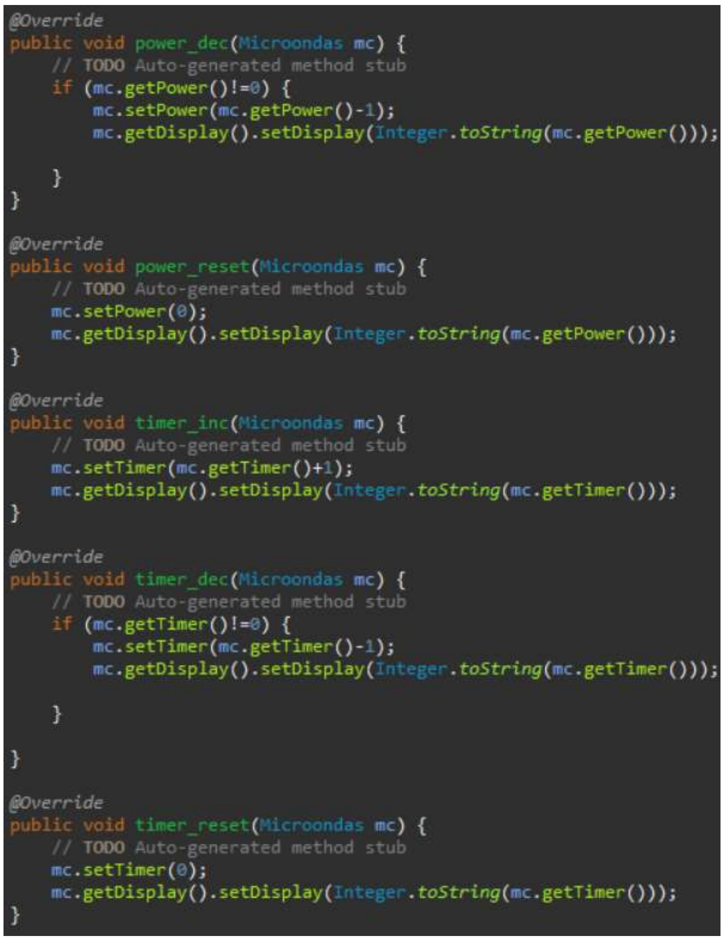
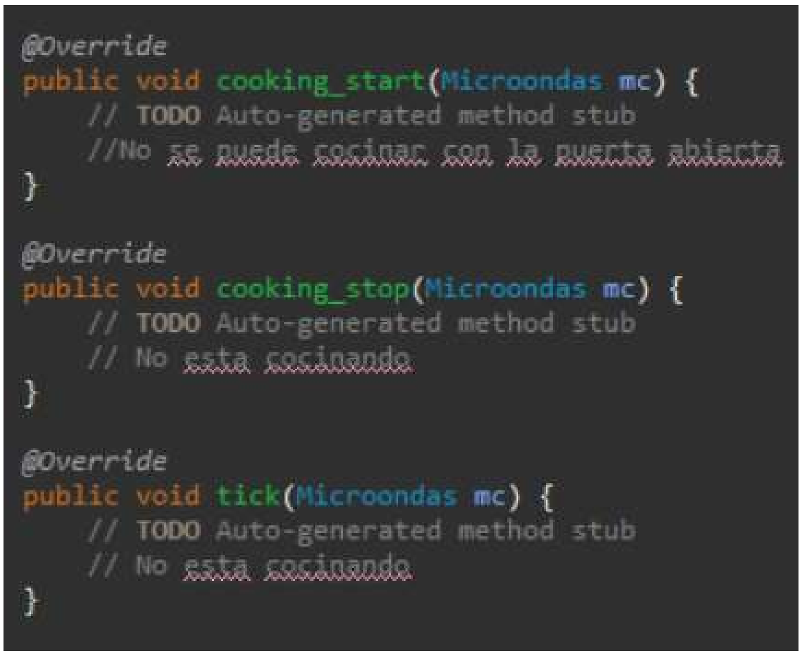


```
@Override
public void cooking_start(Microondas mc) {
    // TODO Auto-generated method stub
    //No puede cocinar con la puerta abierta
}

@Override
public void cooking_stop(Microondas mc) {
    // TODO Auto-generated method stub
    // No esta cocinando

}

@Override
public void tick(Microondas mc) {
    // TODO Auto-generated method stub
    // No esta cocinando
}
```

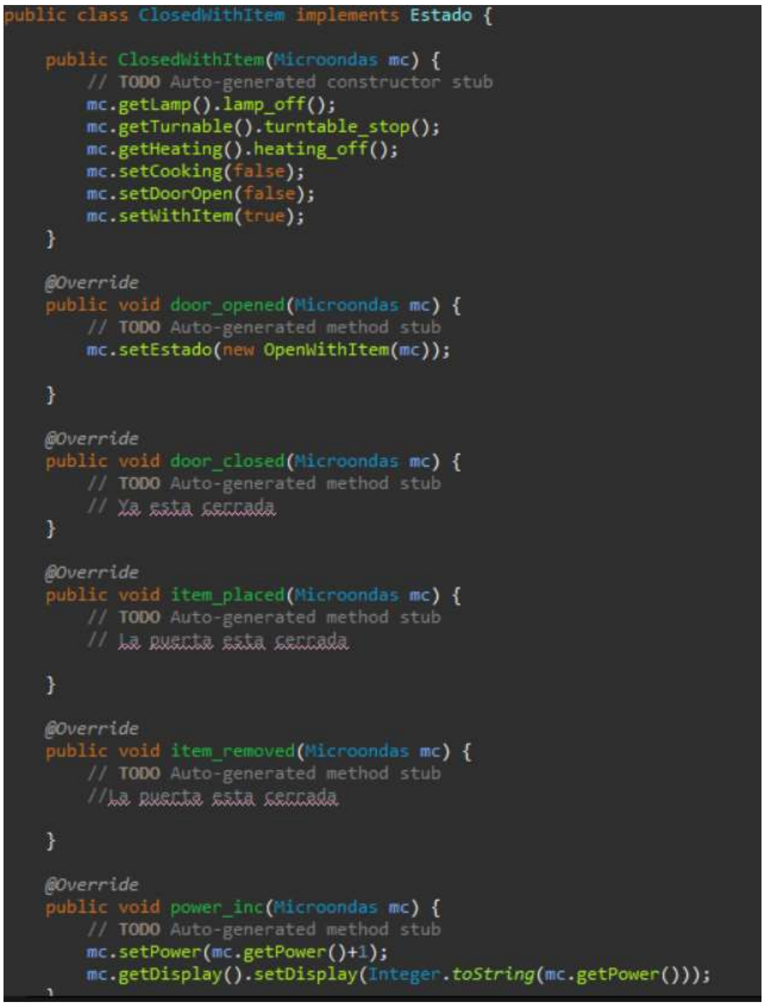
Estado Abierto con Item



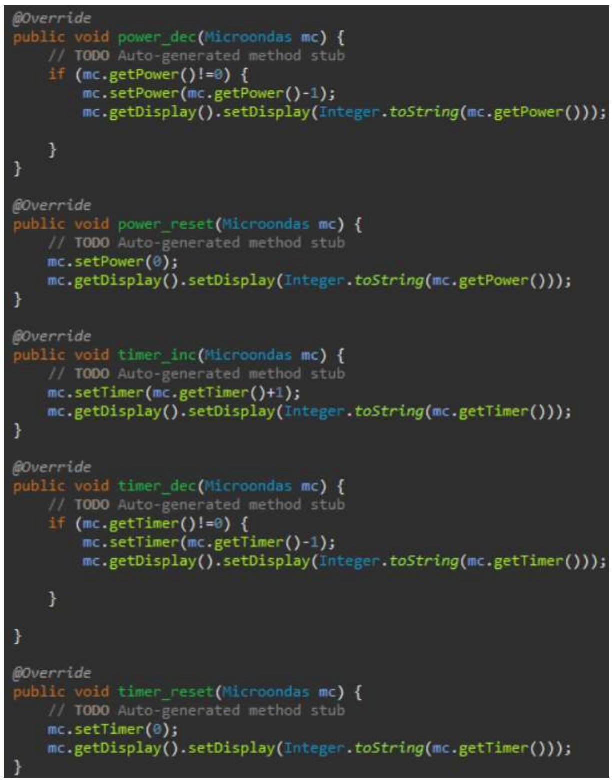
```
public class OpenWithItem implements Estado {

    public OpenWithItem(Microondas mc) {
        // TODO Auto-generated constructor stub
        mc.getLamp().lamp_on();
        mc.getTurnable().turntable_stop();
        mc.getHeating().heating_off();
        mc.setCooking(false);
        mc.setDoorOpen(true);
        mc.setWithItem(true);
    }

    @Override
    public void door_opened(Microondas mc) {
        // TODO Auto-generated method stub
        // Ya esta abierta

    }

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

    }

    @Override
    public void item_placed(Microondas mc) {
        // TODO Auto-generated method stub
        //Ya hay un objeto
    }

    @Override
    public void item_removed(Microondas mc) {
        // TODO Auto-generated method stub
        mc.setEstado(new OpenWithNoItem(mc));
    }

    @Override
    public void power_inc(Microondas mc) {
        // TODO Auto-generated method stub
        mc.setPower(mc.getPower()+1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
    }
```


```java
@Override
public void power_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getPower()!=0) {
        mc.setPower(mc.getPower()-1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));

    }
}

@Override
public void power_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setPower(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
}

@Override
public void timer_inc(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setTimer(mc.getTimer()+1);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}

@Override
public void timer_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getTimer()!=0) {
        mc.setTimer(mc.getTimer()-1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));

    }

}

@Override
public void timer_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setTimer(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}
```

```java
@Override
public void cooking_start(Microondas mc) {
    // TODO Auto-generated method stub
    //No se puede cocinar con la puerta abierta
}

@Override
public void cooking_stop(Microondas mc) {
    // TODO Auto-generated method stub
    // No esta cocinando
}

@Override
public void tick(Microondas mc) {
    // TODO Auto-generated method stub
    // No esta cocinando
}
```

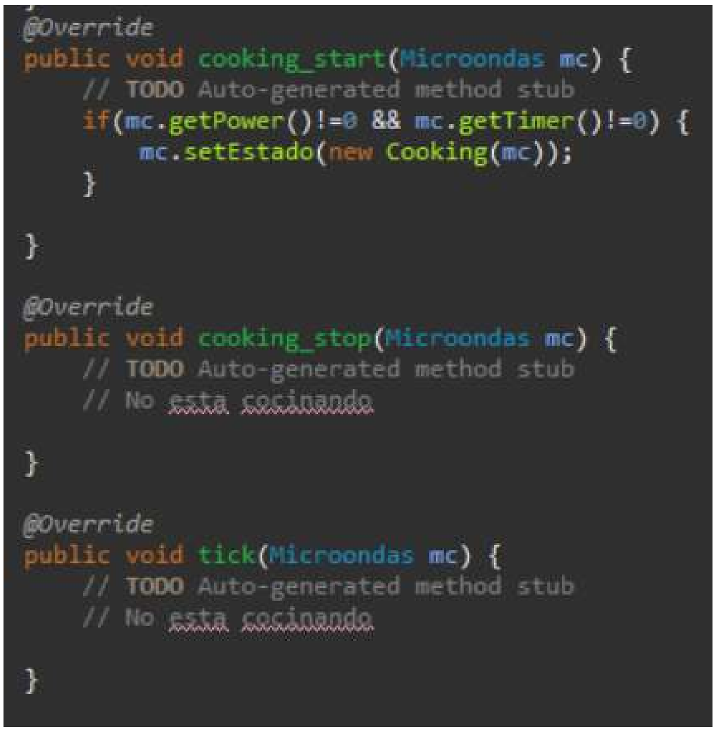
Estado Cerrado con Item



```java
public class ClosedWithItem implements Estado {

    public ClosedWithItem(Microondas mc) {
        // TODO Auto-generated constructor stub
        mc.getLamp().lamp_off();
        mc.getTurnable().turntable_stop();
        mc.getHeating().heating_off();
        mc.setCooking(false);
        mc.setDoorOpen(false);
        mc.setWithItem(true);
    }

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

    }

    @Override
    public void door_closed(Microondas mc) {
        // TODO Auto-generated method stub
        // Ya esta cerrada
    }

    @Override
    public void item_placed(Microondas mc) {
        // TODO Auto-generated method stub
        // La puerta esta cerrada

    }

    @Override
    public void item_removed(Microondas mc) {
        // TODO Auto-generated method stub
        //La puerta esta cerrada

    }

    @Override
    public void power_inc(Microondas mc) {
        // TODO Auto-generated method stub
        mc.setPower(mc.getPower()+1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
```

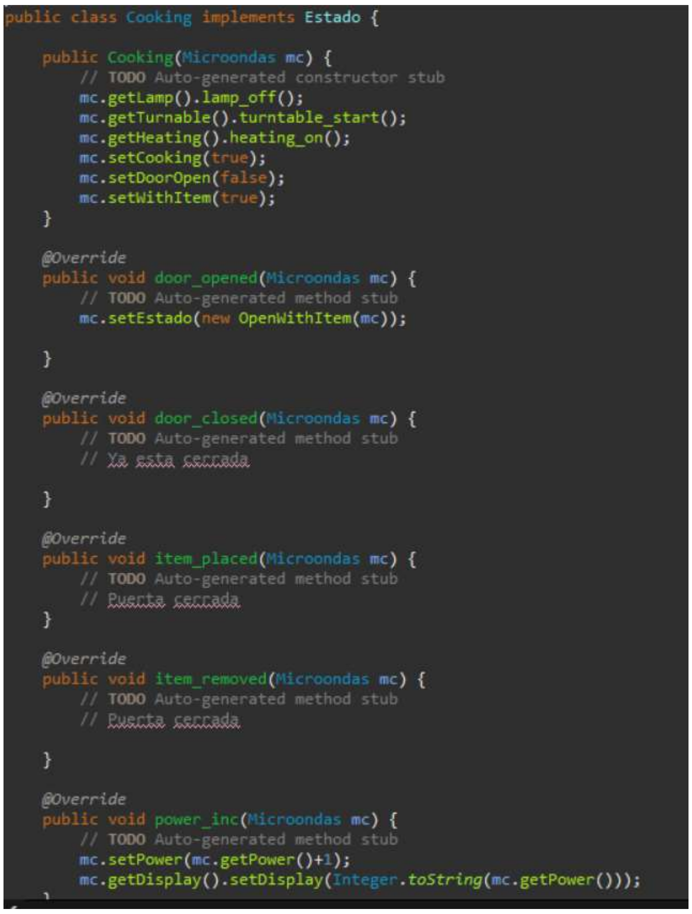


```java
@Override
public void power_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getPower()!=0) {
        mc.setPower(mc.getPower()-1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));

    }
}

@Override
public void power_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setPower(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
}

@Override
public void timer_inc(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setTimer(mc.getTimer()+1);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}

@Override
public void timer_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getTimer()!=0) {
        mc.setTimer(mc.getTimer()-1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));

    }

}

@Override
public void timer_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setTimer(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}
```

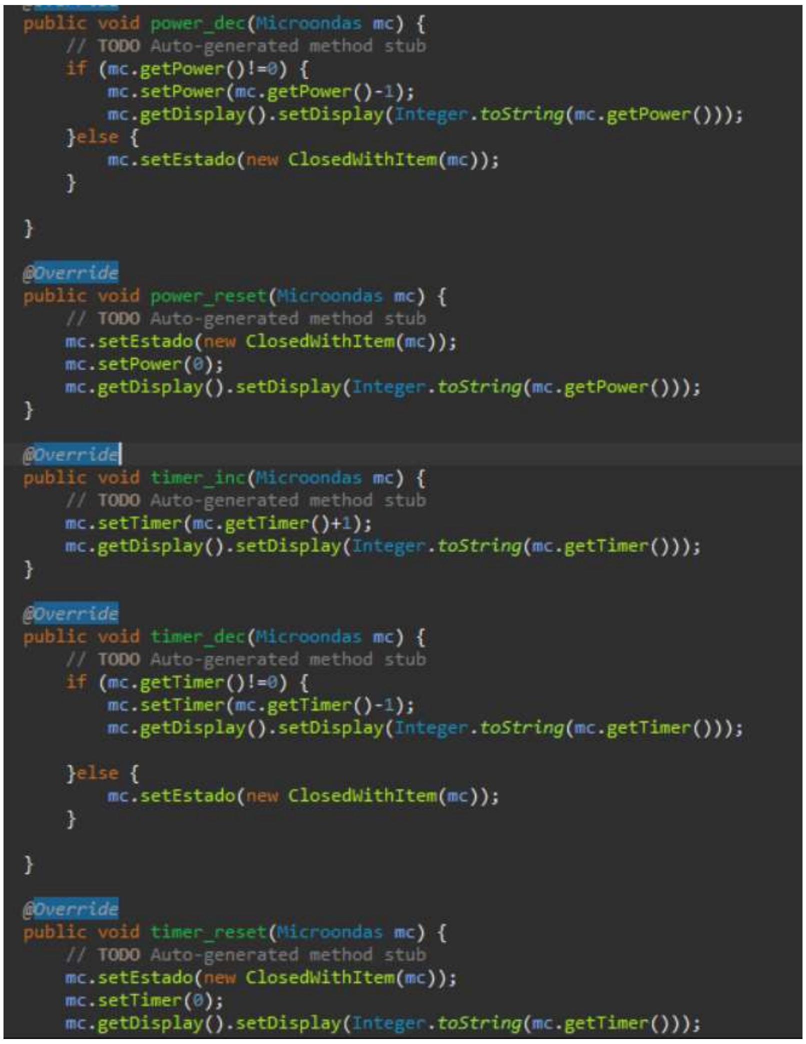


```java
    @Override
    public void cooking_start(Microondas mc) {
        // TODO Auto-generated method stub
        if(mc.getPower()!=0 && mc.getTimer()!=0) {
            mc.setEstado(new Cooking(mc));
        }

    }

    @Override
    public void cooking_stop(Microondas mc) {
        // TODO Auto-generated method stub
        // No esta cocinando

    }

    @Override
    public void tick(Microondas mc) {
        // TODO Auto-generated method stub
        // No esta cocinando

    }
```

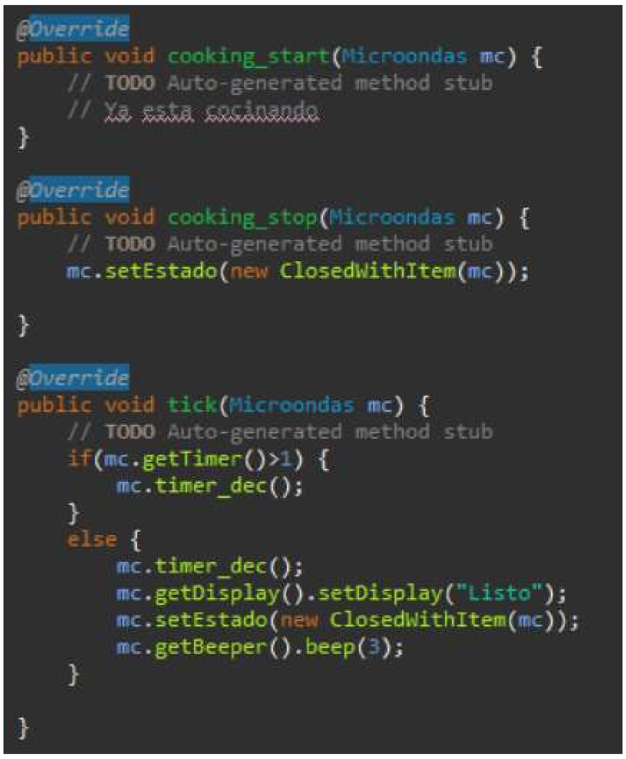
Estado Cocinando



```java
public class Cooking implements Estado {

    public Cooking(Microondas mc) {
        // TODO Auto-generated constructor stub
        mc.getLamp().lamp_off();
        mc.getTurnable().turntable_start();
        mc.getHeating().heating_on();
        mc.setCooking(true);
        mc.setDoorOpen(false);
        mc.setWithItem(true);
    }

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

    }

    @Override
    public void door_closed(Microondas mc) {
        // TODO Auto-generated method stub
        // Ya esta cerrada

    }

    @Override
    public void item_placed(Microondas mc) {
        // TODO Auto-generated method stub
        // Puerta cerrada
    }

    @Override
    public void item_removed(Microondas mc) {
        // TODO Auto-generated method stub
        // Puerta cerrada

    }

    @Override
    public void power_inc(Microondas mc) {
        // TODO Auto-generated method stub
        mc.setPower(mc.getPower()+1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
```

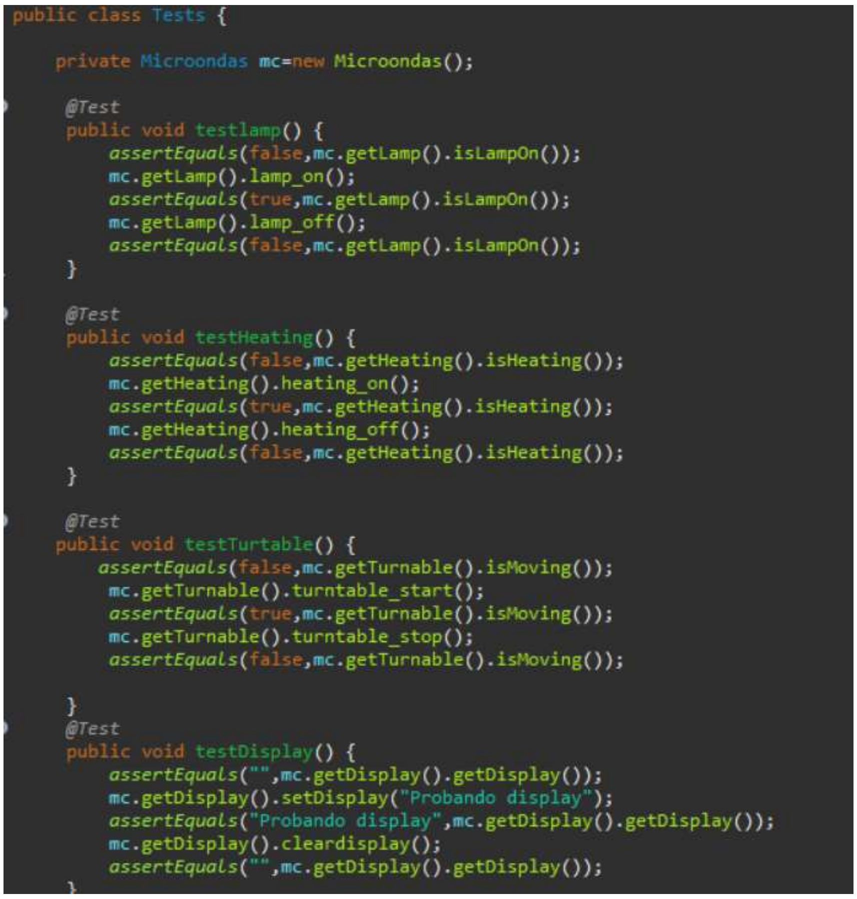


```
public void power_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getPower()!=0) {
        mc.setPower(mc.getPower()-1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
    }else {
        mc.setEstado(new ClosedWithItem(mc));
    }

}

@Override
public void power_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setEstado(new ClosedWithItem(mc));
    mc.setPower(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getPower()));
}

@Override
public void timer_inc(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setTimer(mc.getTimer()+1);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
}

@Override
public void timer_dec(Microondas mc) {
    // TODO Auto-generated method stub
    if (mc.getTimer()!=0) {
        mc.setTimer(mc.getTimer()-1);
        mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));

    }else {
        mc.setEstado(new ClosedWithItem(mc));
    }

}

@Override
public void timer_reset(Microondas mc) {
    // TODO Auto-generated method stub
    mc.setEstado(new ClosedWithItem(mc));
    mc.setTimer(0);
    mc.getDisplay().setDisplay(Integer.toString(mc.getTimer()));
```



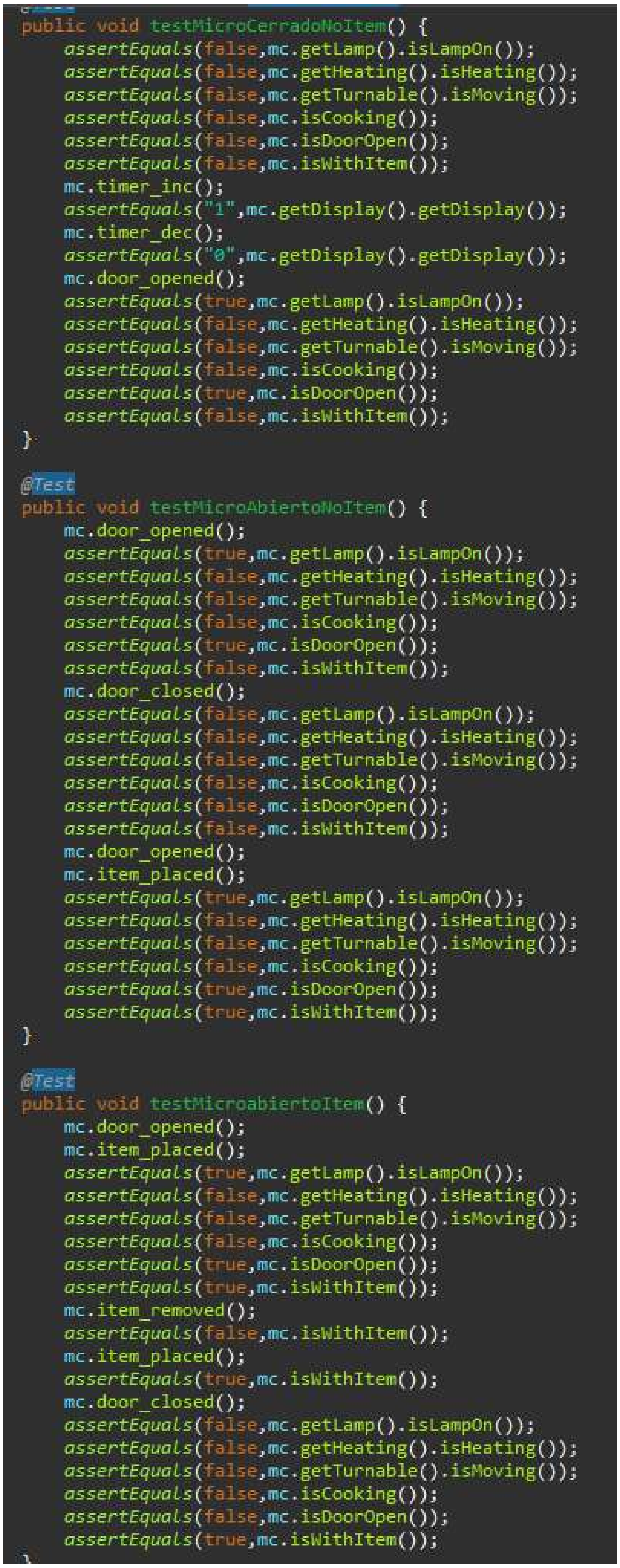
```
@Override
public void cooking_start(Microondas mc) {
    // TODO Auto-generated method stub
    // Ya esta cocinando
}

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

}

@Override
public void tick(Microondas mc) {
    // TODO Auto-generated method stub
    if(mc.getTimer()>1) {
        mc.timer_dec();
    }
    else {
        mc.timer_dec();
        mc.getDisplay().setDisplay("Listo");
        mc.setEstado(new ClosedWithItem(mc));
        mc.getBeeper().beep(3);
    }

}
```

Pruebas implementas con Junit

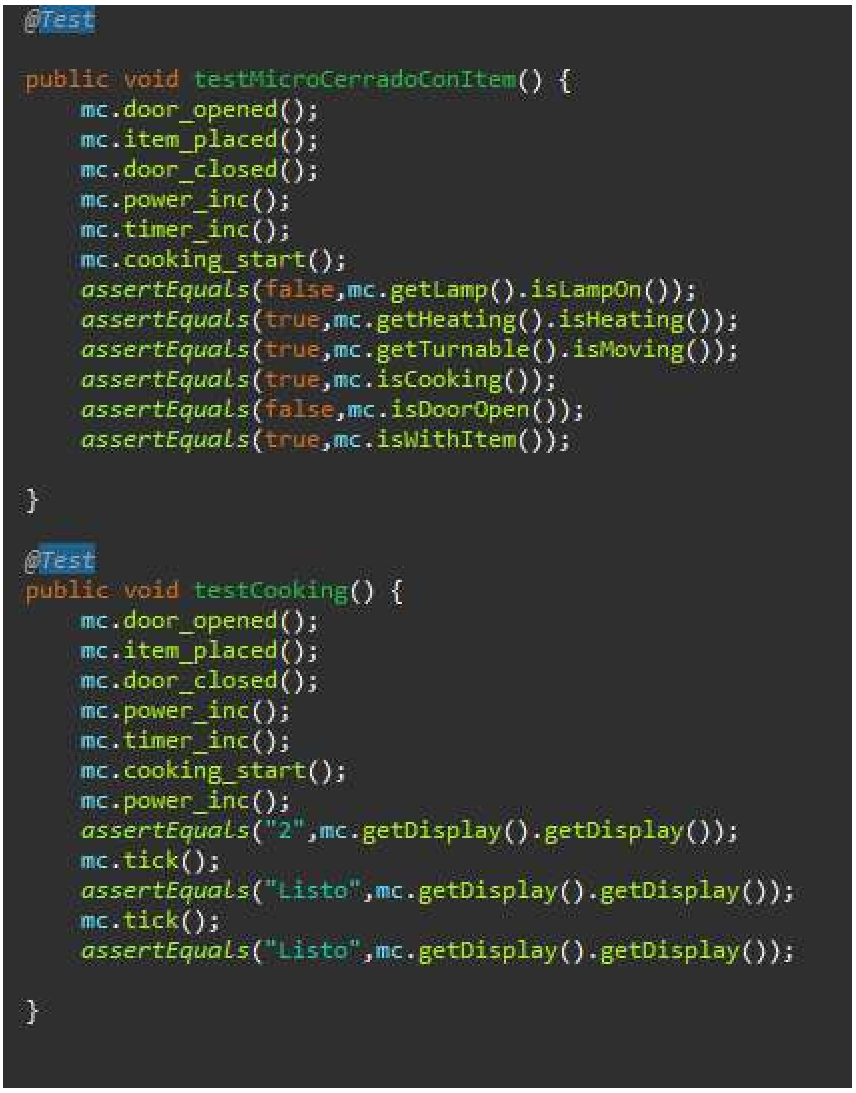


```java
public class Tests {

    private Microondas mc=new Microondas();

     @Test
     public void testlamp() {
         assertEquals(false,mc.getLamp().isLampOn());
         mc.getLamp().lamp_on();
         assertEquals(true,mc.getLamp().isLampOn());
         mc.getLamp().lamp_off();
         assertEquals(false,mc.getLamp().isLampOn());
     }

     @Test
     public void testHeating() {
         assertEquals(false,mc.getHeating().isHeating());
         mc.getHeating().heating_on();
         assertEquals(true,mc.getHeating().isHeating());
         mc.getHeating().heating_off();
         assertEquals(false,mc.getHeating().isHeating());
     }

     @Test
    public void testTurtable() {
        assertEquals(false,mc.getTurnable().isMoving());
         mc.getTurnable().turntable_start();
         assertEquals(true,mc.getTurnable().isMoving());
         mc.getTurnable().turntable_stop();
         assertEquals(false,mc.getTurnable().isMoving());

     }
     @Test
     public void testDisplay() {
         assertEquals("",mc.getDisplay().getDisplay());
         mc.getDisplay().setDisplay("Probando display");
         assertEquals("Probando display",mc.getDisplay().getDisplay());
         mc.getDisplay().cleardisplay();
         assertEquals("",mc.getDisplay().getDisplay());
```

Pruebas de los distintos componentes del microondas



```
public void testMicroCerradoNoItem() {
    assertEquals(false,mc.getLamp().isLampOn());
    assertEquals(false,mc.getHeating().isHeating());
    assertEquals(false,mc.getTurnable().isMoving());
    assertEquals(false,mc.isCooking());
    assertEquals(false,mc.isDoorOpen());
    assertEquals(false,mc.isWithItem());
    mc.timer_inc();
    assertEquals("1",mc.getDisplay().getDisplay());
    mc.timer_dec();
    assertEquals("0",mc.getDisplay().getDisplay());
    mc.door_opened();
    assertEquals(true,mc.getLamp().isLampOn());
    assertEquals(false,mc.getHeating().isHeating());
    assertEquals(false,mc.getTurnable().isMoving());
    assertEquals(false,mc.isCooking());
    assertEquals(true,mc.isDoorOpen());
    assertEquals(false,mc.isWithItem());
}

@Test
public void testMicroAbiertoNoItem() {
    mc.door_opened();
    assertEquals(true,mc.getLamp().isLampOn());
    assertEquals(false,mc.getHeating().isHeating());
    assertEquals(false,mc.getTurnable().isMoving());
    assertEquals(false,mc.isCooking());
    assertEquals(true,mc.isDoorOpen());
    assertEquals(false,mc.isWithItem());
    mc.door_closed();
    assertEquals(false,mc.getLamp().isLampOn());
    assertEquals(false,mc.getHeating().isHeating());
    assertEquals(false,mc.getTurnable().isMoving());
    assertEquals(false,mc.isCooking());
    assertEquals(false,mc.isDoorOpen());
    assertEquals(false,mc.isWithItem());
    mc.door_opened();
    mc.item_placed();
    assertEquals(true,mc.getLamp().isLampOn());
    assertEquals(false,mc.getHeating().isHeating());
    assertEquals(false,mc.getTurnable().isMoving());
    assertEquals(false,mc.isCooking());
    assertEquals(true,mc.isDoorOpen());
    assertEquals(true,mc.isWithItem());
}

@Test
public void testMicroabiertoItem() {
    mc.door_opened();
    mc.item_placed();
    assertEquals(true,mc.getLamp().isLampOn());
    assertEquals(false,mc.getHeating().isHeating());
    assertEquals(false,mc.getTurnable().isMoving());
    assertEquals(false,mc.isCooking());
    assertEquals(true,mc.isDoorOpen());
    assertEquals(true,mc.isWithItem());
    mc.item_removed();
    assertEquals(false,mc.isWithItem());
    mc.item_placed();
    assertEquals(true,mc.isWithItem());
    mc.door_closed();
    assertEquals(false,mc.getLamp().isLampOn());
    assertEquals(false,mc.getHeating().isHeating());
    assertEquals(false,mc.getTurnable().isMoving());
    assertEquals(false,mc.isCooking());
    assertEquals(false,mc.isDoorOpen());
    assertEquals(true,mc.isWithItem());
}
```



```
@Test

public void testMicroCerradoConItem() {
    mc.door_opened();
    mc.item_placed();
    mc.door_closed();
    mc.power_inc();
    mc.timer_inc();
    mc.cooking_start();
    assertEquals(false,mc.getLamp().isLampOn());
    assertEquals(true,mc.getHeating().isHeating());
    assertEquals(true,mc.getTurnable().isMoving());
    assertEquals(true,mc.isCooking());
    assertEquals(false,mc.isDoorOpen());
    assertEquals(true,mc.isWithItem());

}

@Test
public void testCooking() {
    mc.door_opened();
    mc.item_placed();
    mc.door_closed();
    mc.power_inc();
    mc.timer_inc();
    mc.cooking_start();
    mc.power_inc();
    assertEquals("2",mc.getDisplay().getDisplay());
    mc.tick();
    assertEquals("Listo",mc.getDisplay().getDisplay());
    mc.tick();
    assertEquals("Listo",mc.getDisplay().getDisplay());

}
```

Pruebas de los distintos estados y su comportamiento esperado.

Link del repositorio en Github

https://github.com/alexpascualm/Microondas