-Decidimos no usar el patrón State en los distintos cambios de estados que pueda tener un dispositivo(Apagado, Encendido, ModoAhorro) y usamos Enum, en el caso de un dispositivo estándar utilizamos una interfaz Adapter, pudiendo haber un AdapterEstandarInteligente y un AdapterEstandarDefecto, esta última en sus método de devolver el estado, devolvería null.

-Utilizamos los patrones de diseño Adapter y Command.

**Adapter**

Utilizamos el patrón adapter para transformar una clase(Estandar) en otra(Inteligente), tal que la clase Estandar que no podia utilizar la clase Inteligente, haga uso de ella a través del AdapterEstandarAInteligente.

**Command**

Utilizamos este patrón para solicitar una acción a la clase Actuador, sin que este conozca la lógica de su ejecutar/accionar, pudiendo conocer o no a su receptor, en este caso decidimos que el Actuador conozca a su Dispositivo que se le debe aplicar la acción.