Ejercicio: Sistema de Gestión de Dispositivos Inteligentes

Contexto:

Se te ha asignado la tarea de diseñar un sistema para gestionar diferentes **dispositivos inteligentes** que están conectados a una red domótica. Los dispositivos incluyen luces inteligentes, termostatos y cámaras de seguridad. Todos estos dispositivos tienen características comunes: pueden encenderse, apagarse y deben reportar su estado.

Al principio, podría parecer razonable crear una jerarquía de clases utilizando herencia, donde todos los dispositivos heredan de una clase base `DispositivoInteligente`. Sin embargo, en este ejercicio **deberás aplicar la composición** en lugar de herencia para resolver el problema. Esto permitirá una mayor flexibilidad al agregar nuevas funcionalidades sin necesidad de modificar las clases existentes o crear jerarquías complicadas.

Requerimientos funcionales:

1. **DispositivoInteligente**: Cada dispositivo debe poder encenderse, apagarse y reportar su estado (encendido/apagado).
2. **Luces Inteligentes**: Las luces deben permitir ajustar el brillo (un valor de 0 a 100%).
3. **Termostatos**: Los termostatos deben permitir ajustar la temperatura (un valor numérico en grados).
4. **Cámaras de Seguridad**: Las cámaras deben poder grabar video y activar un sensor de movimiento.

Requerimientos no funcionales:

- El diseño debe promover la reutilización de código y evitar la duplicación.
- Se debe permitir agregar nuevos tipos de dispositivos fácilmente en el futuro (por ejemplo, un sistema de alarma).

Instrucciones:

1. **Composición sobre herencia**:
    - En lugar de usar una jerarquía de clases como `DispositivoInteligente -> LuzInteligente -> Termostato -> Camara`, utiliza clases que representen **comportamientos** específicos, como `Encendible`, `AjusteBrillo`, `AjusteTemperatura`, etc. Usa estos comportamientos como componentes que se pueden combinar para crear los distintos dispositivos.
2. **Creación de Clases**:
    - Define una clase `DispositivoInteligente` que pueda agregar diferentes componentes según el tipo de dispositivo.

    - Implementa comportamientos como clases separadas (por ejemplo, `Encendible`, `AjusteBrillo`, `AjusteTemperatura`).
    - Cada dispositivo (luz, termostato, cámara) deberá ser una instancia de `DispositivoInteligente` que combina varios de estos comportamientos mediante composición.
3. **Ampliación**:
    - Se debe poder agregar fácilmente nuevos dispositivos en el futuro, como alarmas, sin necesidad de modificar las clases existentes.
4. **Prueba**:
    - Implementa una instancia de cada tipo de dispositivo y demuestra cómo funciona el sistema. Por ejemplo, crea una luz que pueda encenderse, apagarse y ajustar el brillo, un termostato que ajuste la temperatura, y una cámara que encienda y active el sensor de movimiento.

Entregable:

- Diagrama de clases que muestre la relación entre las clases del sistema.
- Implementación del sistema usando el enfoque de **composición sobre herencia**.
- Explicación del porqué mejora la solución con este enfoque