□ Ejercicio: El Control Remoto Inteligente

Crea una clase `ControlRemoto` que simule un control remoto universal. Este control debe demostrar **encapsulamiento** (con getters/setters) y **adaptabilidad** (métodos flexibles).

Parte 1: Atributos PRIVADOS (encapsulamiento)

La clase debe tener estos atributos **privados** (usando `__`):

- `__dispositivo` (string): nombre del dispositivo controlado
- `__canal` (entero): canal actual (1-99, inicia en 1)
- `__volumen` (entero): volumen actual (0-100, inicia en 20)
- `__encendido` (booleano): True=encendido, False=apagado (inicia apagado)

Parte 2: Getters y Setters (acceso controlado)

Implementa estos métodos:

Getters (obtener valores):

- `get_dispositivo()`: devuelve el nombre del dispositivo
- `get_canal()`: devuelve el canal actual
- `get_volumen()`: devuelve el volumen actual
- `get_estado()`: devuelve "ENCENDIDO" o "APAGADO"

Setters (cambiar valores con validación):

- `set_canal(nuevo_canal)`: cambia el canal (solo si está encendido y entre 1-99)
- `set_volumen(nuevo_volumen)`: cambia el volumen (solo si está encendido y entre 0-100)
- `set_encendido(estado)`: enciende/apaga (True/False)

Parte 3: Método con comportamiento adaptable

Crea **un solo método** llamado `controlar` que haga **CUATRO cosas diferentes** según lo que reciba:

1. **controlar("encender")** o **controlar("apagar")**
→ Cambia el estado de encendido
2. **controlar("canal", 15)**
→ Cambia al canal indicado (usa el setter)
3. **controlar("volumen", 75)**
→ Cambia al volumen indicado (usa el setter)
4. **controlar("subir_volumen")** o **controlar("bajar_volumen")**
→ Sube/baja el volumen 10 puntos (usando getter y setter)
1. **controlar("siguiente_canal")** o **controlar("anterior_canal")**
→ Cambia al canal siguiente/anterior

Parte 4: Método especial str

- `__str__()`: Devuelve una descripción completa y formateada del control

Requisitos de validación:

1. Todos los setters deben validar:

- Que el dispositivo esté encendido (excepto `set_encendido`)
- Que los valores estén en rangos válidos
- Mostrar mensajes claros de éxito/error

2. El método `controlar` debe:

- Usar los getters/setters internamente
- Manejar comandos no reconocidos: `"Comando no válido: [comando]"`

3. Encapsulamiento:

- Los atributos deben ser realmente privados (`__atributo`)
- Solo se accede/modifica mediante los métodos públicos

INSTRUCCIONES PARA EL PROGRAMA PRINCIPAL

Paso 1: Crear un objeto ControlRemoto

•Crea una instancia de `ControlRemoto` llamada `tv`
•Pásale como parámetro el nombre `"Televisor Sala"`

Paso 2: Probar los métodos getters

•Usa `print()` para mostrar el dispositivo obtenido con `get_dispositivo()`
•Usa `print()` para mostrar el estado obtenido con `get_estado()`
•Deja una línea en blanco después con `print()`

Paso 3: Usar el método adaptable `controlar()`

•Llama a `controlar("encender")` para encender el televisor
•Llama a `controlar("canal", 25)` para cambiar al canal 25
•Llama a `controlar("volumen", 60)` para ajustar el volumen a 60
•Llama a `controlar("subir_volumen")` para aumentar el volumen 10 puntos
•Llama a `controlar("siguiente_canal")` para avanzar un canal
•Deja una línea en blanco después

Paso 4: Usar setters directamente

•Usa `set_canal(30)` para cambiar directamente al canal 30
•Usa `set_volumen(50)` para ajustar directamente el volumen a 50
•Deja una línea en blanco después

Paso 5: Mostrar información con `__str__`

•Imprime el objeto `tv` directamente usando `print(tv)`
•Esto automáticamente usará el método `__str__`
•Deja una línea en blanco después

Paso 6: Probar validaciones de error

•Llama a `controlar("apagar")` para apagar el televisor
•Intenta usar `set_canal(100)` (debe fallar porque está apagado)
•Intenta usar `set_canal(150)` (debe fallar por canal inválido