

COMMAND

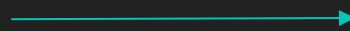
Patrón de Diseño

FINALIDAD

- Convertir una solicitud en un objeto independiente que contiene toda la información sobre la solicitud

DESVENTAJAS DE NO UTILIZARLO

Necesidad de crear múltiples subclases para manejar diferentes acciones



Estructura de código complicada y duplicación de código

Necesidad de invocar las mismas acciones desde varios lugares



Dificulta la mantenibilidad y la coherencia del código.

APLICABILIDAD

- Parametrizar objetos con operaciones.
- Implementar operaciones reversibles.
- Poner operaciones en cola, programar su ejecución, o ejecutarlas de forma remota.

VENTAJAS

- Desacoplar las clases que invocan operaciones de las que realizan esas operaciones.
- Introducir nuevos comandos en la aplicación sin descomponer el código existente.
- Implementar la ejecución diferida de operaciones.
- Ensamblar un grupo de comandos simples para crear uno complejo.

EJEMPLO

```
1 # Clase Receiver (Receptor) - Representa la televisión que realizará las acciones.
2 class Television:
3     def turn_on(self):
4         print("Televisión encendida")
5
6     def turn_off(self):
7         print("Televisión apagada")
8
9 # Interfaz Command (Comando) - Define un método "execute" que todas las clases de comando deben implementar.
10 class Command:
11     def execute(self):
12         pass
13
14 # Clase ConcreteCommand (Comando Concreto) - Implementa un comando específico.
15 class TurnOnCommand(Command):
16     def __init__(self, television):
17         self.television = television
18
19     def execute(self):
20         self.television.turn_on()
21
22 # Clase ConcreteCommand (Comando Concreto) - Implementa otro comando específico.
23 class TurnOffCommand(Command):
24     def __init__(self, television):
25         self.television = television
26
27     def execute(self):
28         self.television.turn_off()
```

```
30 # Clase Invoker (Invocador) - Encargada de ejecutar los comandos.
31 class RemoteControl:
32     def __init__(self):
33         self.command = None
34
35     def set_command(self, command):
36         self.command = command
37
38     def press_button(self):
39         self.command.execute()
40
41 # Uso del patrón Command
42 if __name__ == "__main__":
43     television = Television()
44
45     turn_on_command = TurnOnCommand(television)
46     turn_off_command = TurnOffCommand(television)
47
48     remote = RemoteControl()
49
50     remote.set_command(turn_on_command)
51     remote.press_button()
52
53     remote.set_command(turn_off_command)
54     remote.press_button()
```