# PATRONES DE DISEÑO

## FACADE

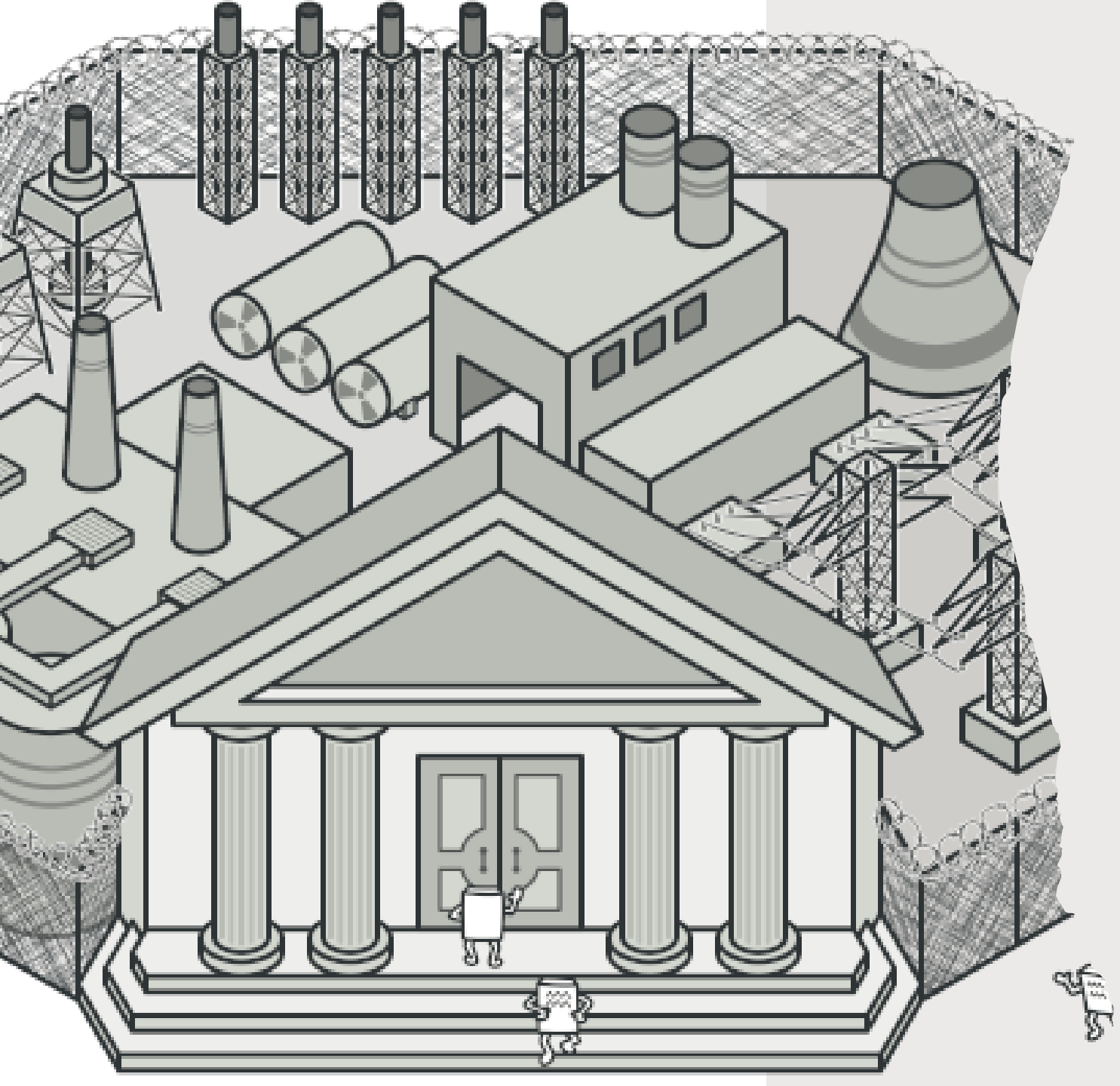
# PROBLEMA

- Imagina que debes lograr que tu código trabaje con un amplio grupo de objetos que pertenecen a una sofisticada biblioteca o *framework*. Normalmente, debes inicializar todos esos objetos, llevar un registro de las dependencias, ejecutar los métodos en el orden correcto y así sucesivamente.

- Como resultado, la lógica de negocio de tus clases se vería estrechamente acoplada a los detalles de implementación de las clases de terceros, haciéndola difícil de comprender y mantener.

# PROPÓSITO

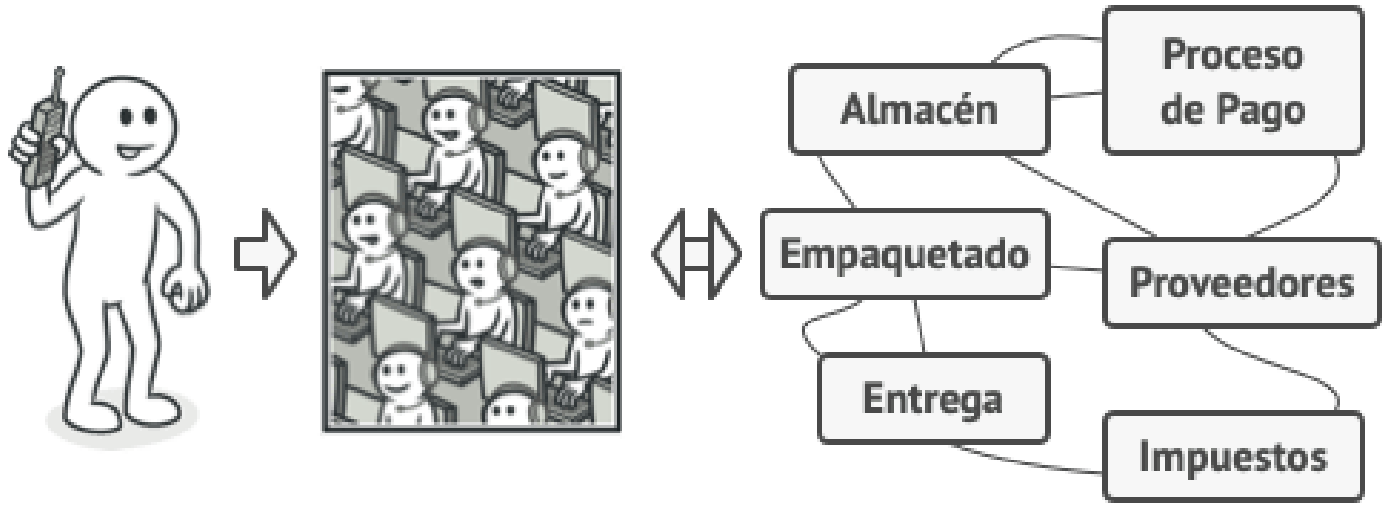
- **Facade** es un patrón de diseño estructural que proporciona una interfaz simplificada a una biblioteca, un framework o cualquier otro grupo complejo de clases.



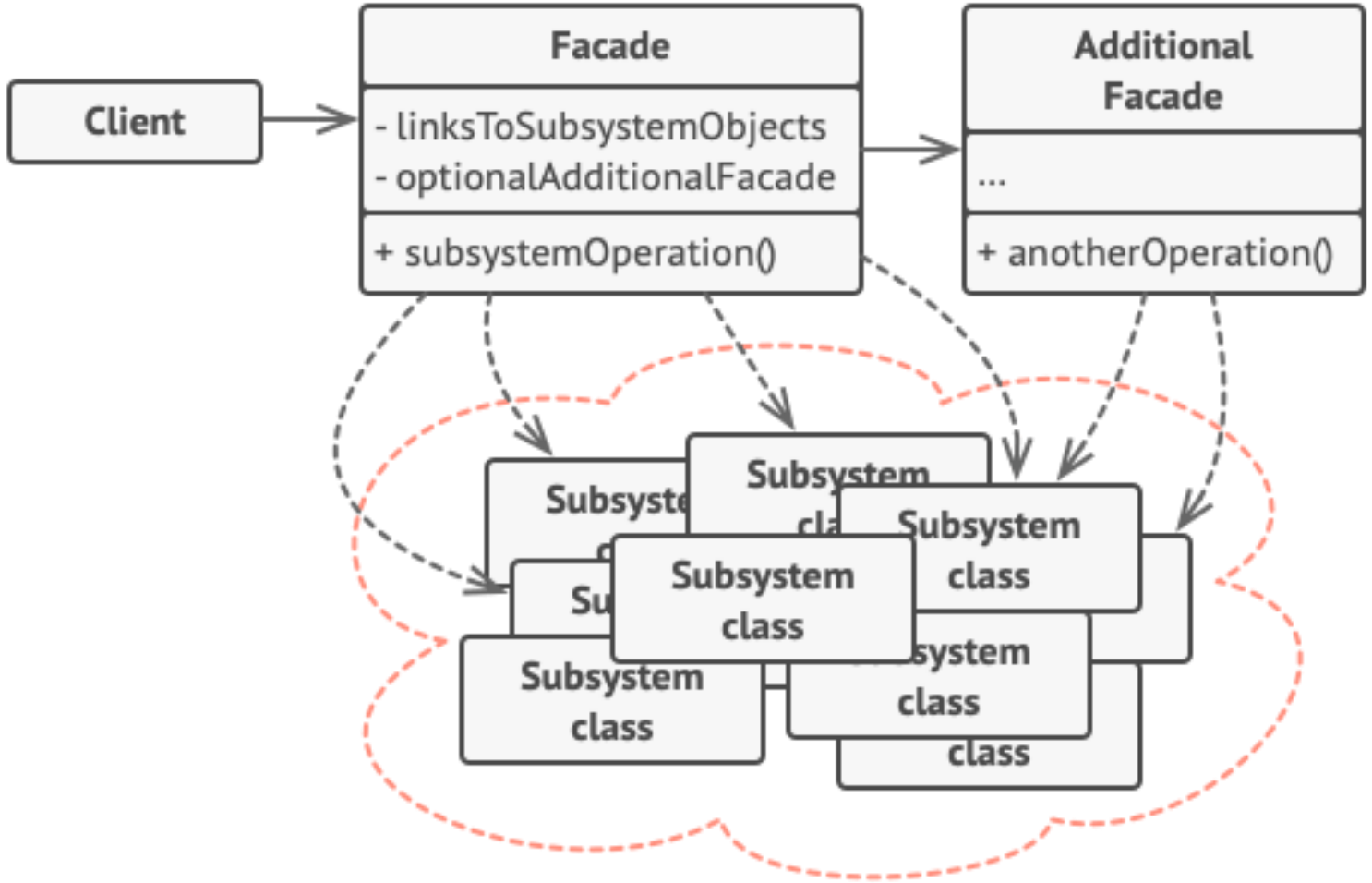
# SOLUCIÓN

- Una fachada es una clase que proporciona una interfaz simple a un subsistema complejo que contiene muchas partes móviles.

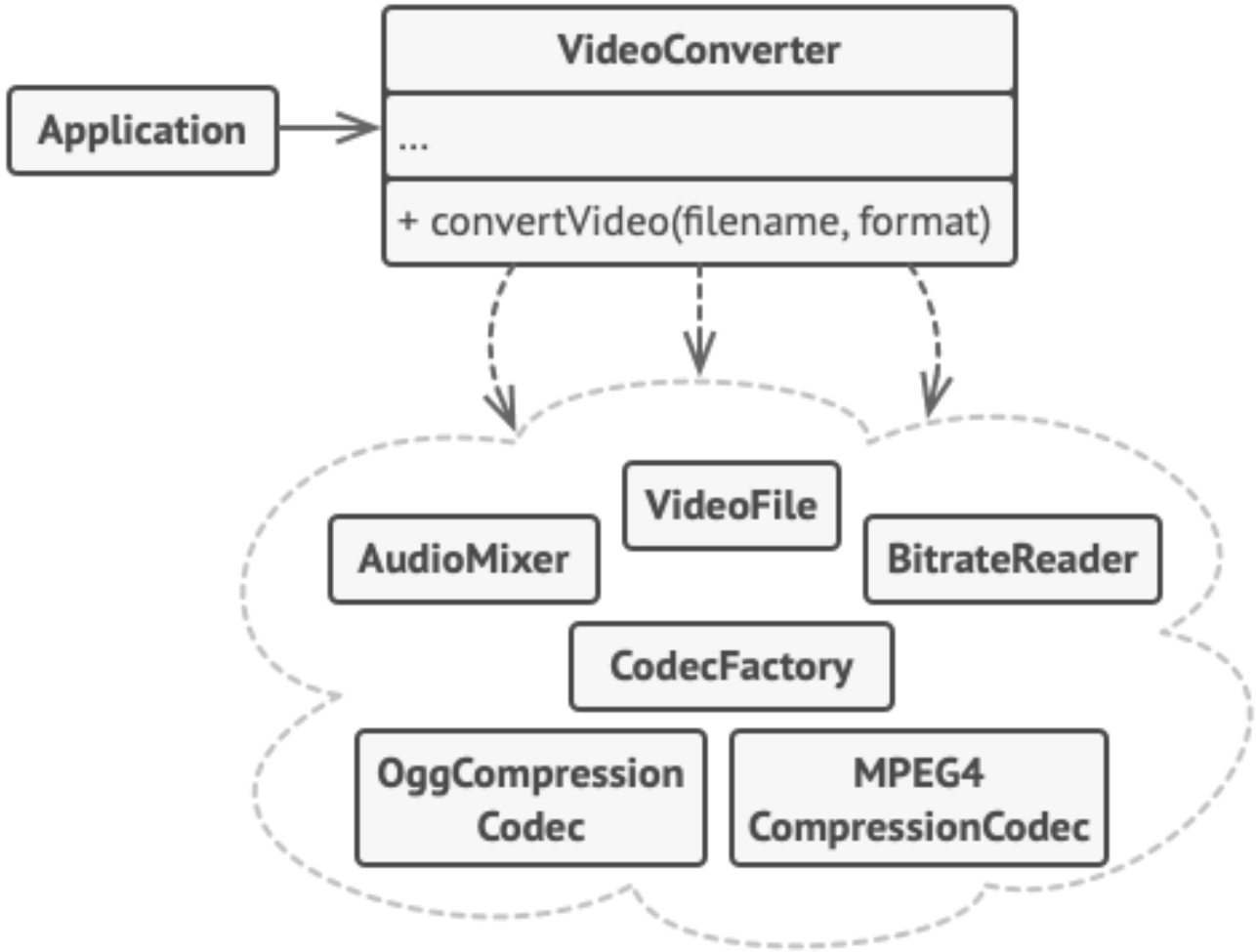
# ANALOGÍA EN EL MUNDO REAL



# ESTRUCTURA


Client
Facade
- linksToSubsystemObjects
- optionalAdditionalFacade
+ subsystemOperation()
Additional Facade
...
+ anotherOperation()
Subsystem class
Subsystem class
Subsystem class
Subsystem class
Subsystem class
Subsystem class

# PSEUDOCÓDIGO



# CUANDO USARLO?

- Cuando necesites una interfaz limitada pero directa a un subsistema complejo.
- Cuando quieras estructurar un subsistema en capas.

# COMO IMPLEMENTARLO?

- Comprueba si es posible proporcionar una interfaz más simple que la que está proporcionando un subsistema existente.
- Declara e implementa esta interfaz en una nueva clase fachada.
- Para aprovechar el patrón al máximo, haz que todo el código cliente se comunique con el subsistema únicamente a través de la fachada.
- Si la fachada se vuelve demasiado grande, piensa en extraer parte de su comportamiento y colocarlo dentro de una nueva clase fachada refinada.

# VENTAJAS

- Aislamiento: Puedes aislar tu código de la complejidad de un subsistema.
- Pruebas: Usar fachadas hace el proceso de "testing" mas fácil si tiene los métodos comunes para realizarlas
- Bajo acoplamiento: La posibilidad de tener un bajo acoplamiento entre los clientes y los subsistemas

# DESVENTAJAS

- Cambios en los métodos: Como sabemos, en el método de Fachada, los métodos posteriores se adjuntan a la capa de Fachada y cualquier cambio en el método posterior puede generar cambios en la capa de Fachada que no son favorables.
- Una fachada puede convertirse en un objeto todopoderoso acoplado a todas las clases de una aplicación.