Patrón de Diseño

FACADE

Problema

- Se tiene un conjunto amplio de objetos que pertenecen a librerías o clases complejas.
- Normalmente se tendría que inicializar los objetos, llevar registro de las dependencias y ejecutar los métodos en el orden correcto.
- Como resultado la aplicación adquiere alto acoplamiento.

Solucion

- Se crea una clase Facade que proporciona una interfaz simple contra uno y varios subsistemas complejos.

- Esta clase puede limitar la funcionalidad del subsistema.

- Resulta útil a la hora de integrar la aplicación con un framework o librería compleja

Contexto

- Facade se utiliza cuando se necesita un interfaz limitada, simple y sencilla para un subsistema complejo.
- También cuando se necesita estructurar al subsistema en capas.

Objetivo

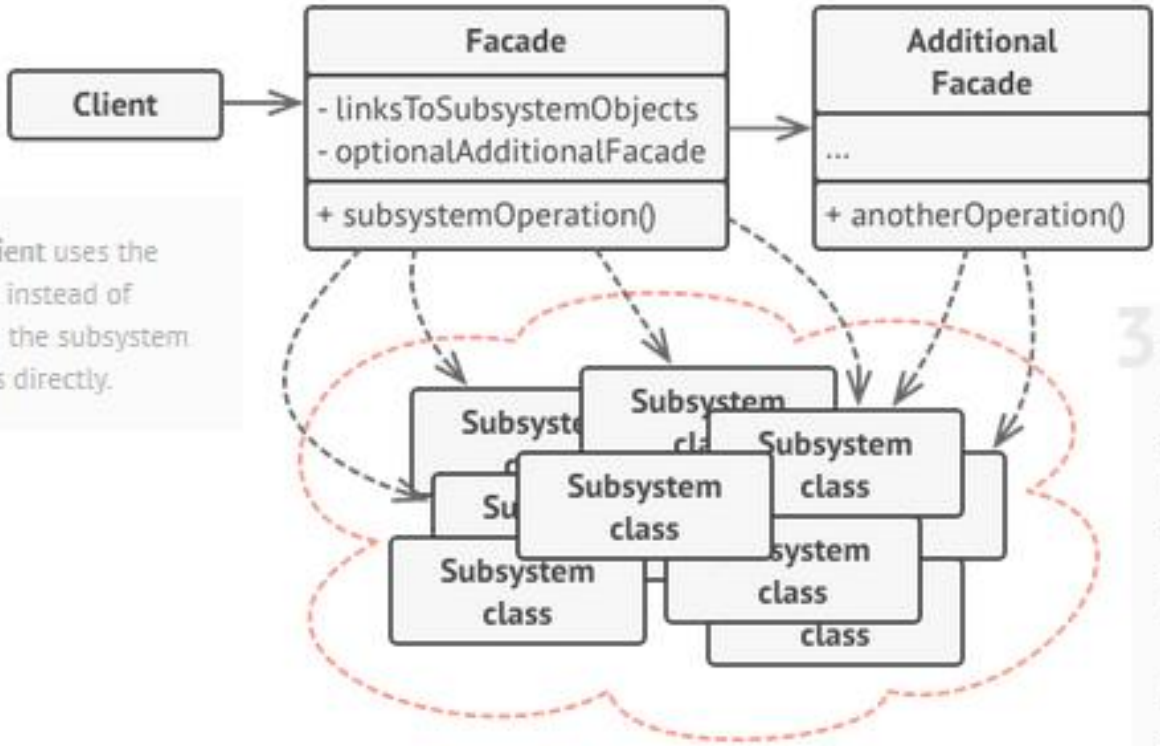
El patrón Facade es un patrón de diseño estructural que crea una interfaz simplificada para una librería, framework o un conjunto complejo de clases

Implementación

- Comprobar si es posible proveer una interfaz más simple que alguno de los subsistemas.
- Implementar la clase Facade, esta clase debería enviar mensajes a los objetos apropiados del subsistema.
- El código cliente deberá acceder al subsistema únicamente a través de la clase Facade.
- Si la clase Facade crece demasiado, se deberá extraer parte del comportamiento a un a nueva clase Facade.

1 The **Facade** provides convenient access to a particular part of the subsystem's functionality. It knows where to direct the client's request and how to operate all the moving parts.

2 An **Additional Facade** class can be created to prevent polluting a single facade with unrelated features that might make it yet another complex structure. Additional facades can be used by both clients and other facades.



4 The **Client** uses the facade instead of calling the subsystem objects directly.

3 The **Complex Subsystem** consists of dozens of various objects. To make them all do something meaningful, you have to dive deep into the subsystem's implementation details, such as initializing objects in the correct order and supplying them with data in the proper format.

Subsystem classes aren't aware of the facade's existence. They operate within the system and work with each other directly.

Patrones Relacionados

Singleton: el patrón Singleton puede ser aplicado a Facade, ya que solamente es necesario una instancia de la clase Facade.

Adapter: El patrón Adapter se aplica a solamente un objeto, mientras que Facade se aplica a un subsistema entero.

Proxy: Ambos patrones funcionan como buffer para el subsistema, la diferencia radica en que Proxy utiliza la misma interfaz que el objeto