Patrones de diseño

INGENIERIA DE SOFTWARE
IIC2143

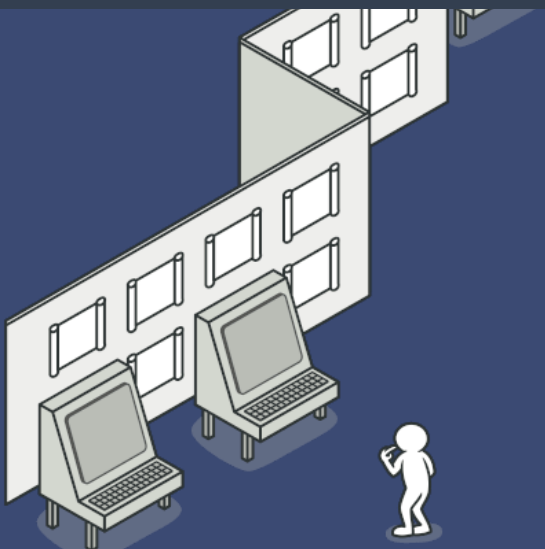
Patrones de diseño

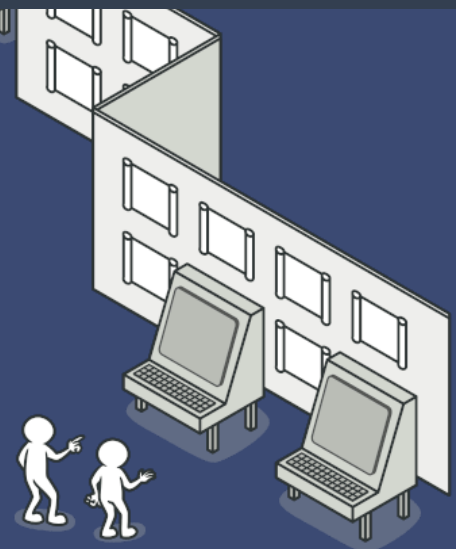
¿QUE SON?

Un patrón de diseño es una solución general y reutilizable para un problema que ocurre en el diseño de software.

VENTAJAS

Proporciona una plantilla o un modelo para resolver un problema particular de manera estructurada y eficiente.

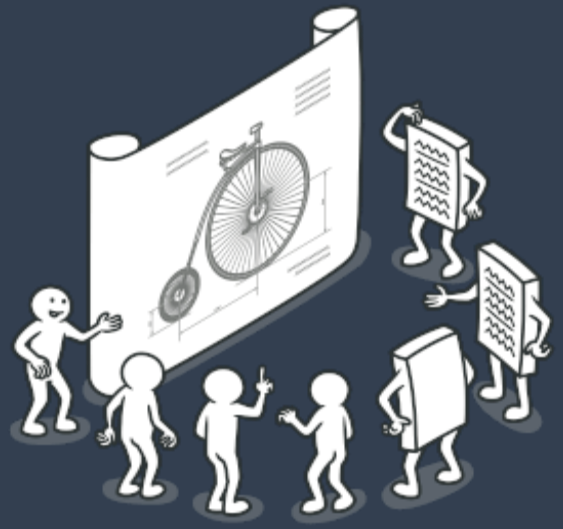




Clasificación

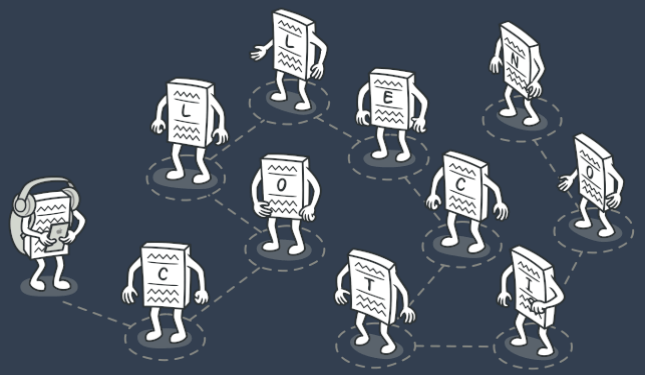
Patrones creacionales

Proporcionan mecanismos de creación de objetos que incrementan la flexibilidad y la reutilización de código existente.



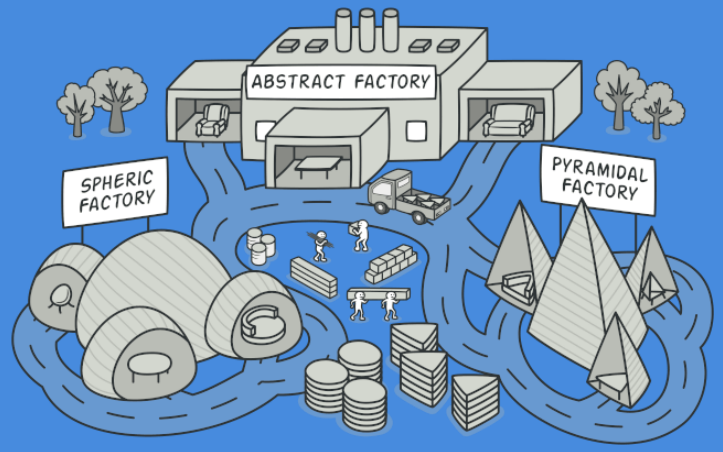
Patrones Estructurales

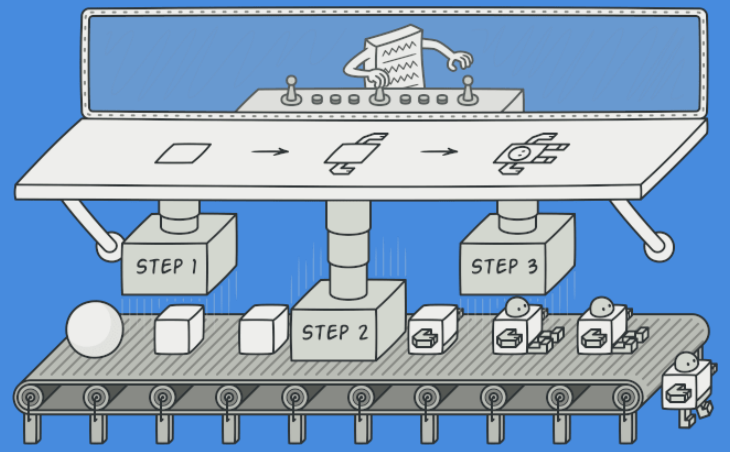
Explican cómo ensamblar objetos y clases en estructuras más grandes a la vez que se mantiene la flexibilidad y eficiencia de la estructura.



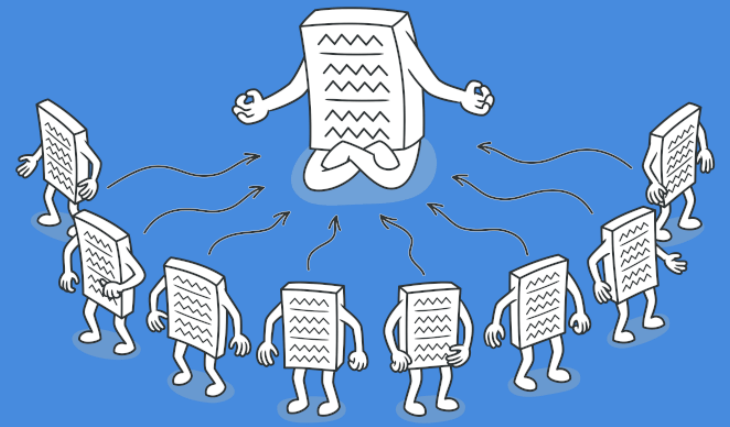
Patrones Comportamiento

Se encargan de una comunicación efectiva y la asignación de responsabilidades entre objetos





Estructurales

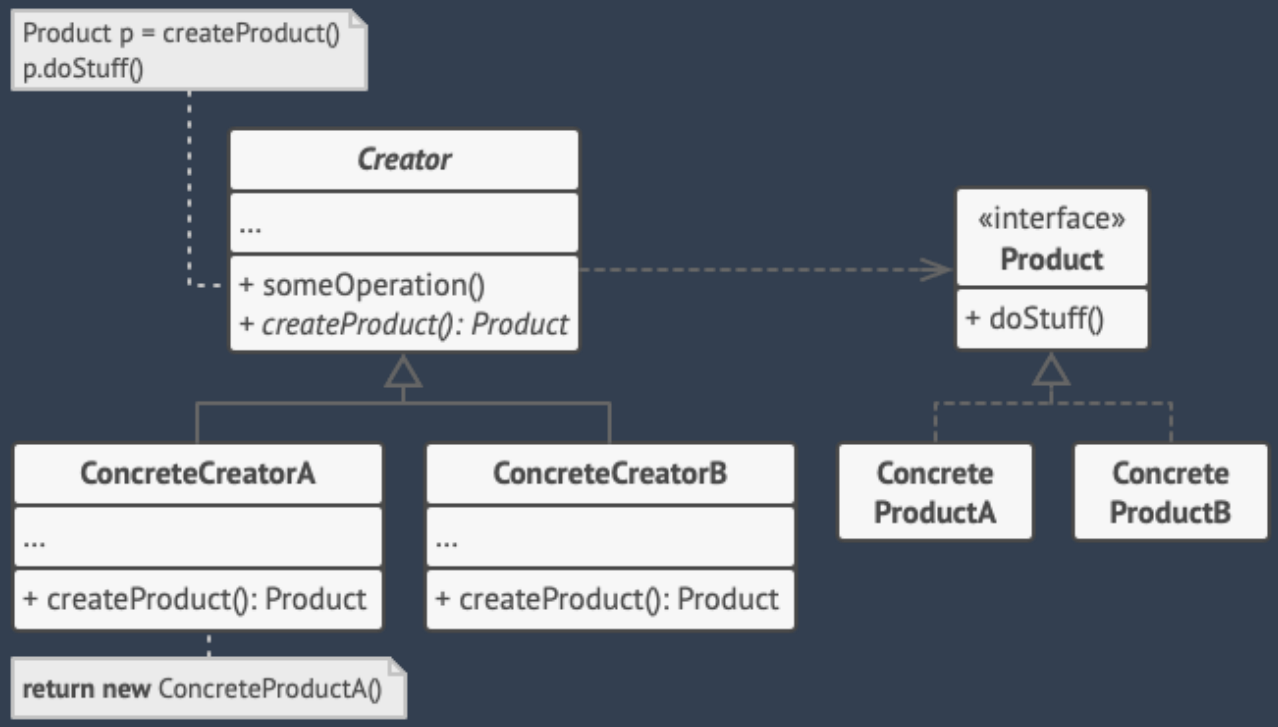


FACTORY METHOD

Proporciona una
interfaz para crear objetos
en una superclase

Permite a
las subclases alterar el tipo
de objetos que se crearán.

Objetivo: crear
objetos a través de
una clase generalizada.

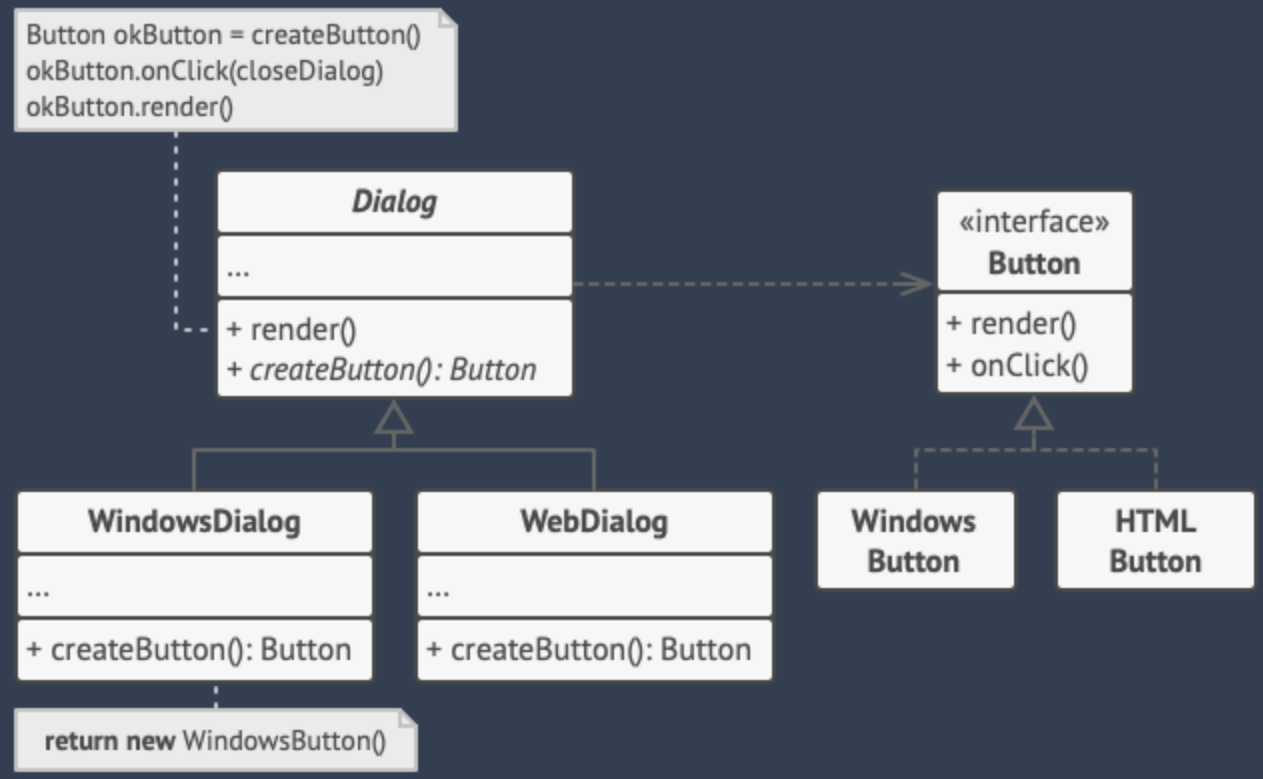


FACTORY METHOD

Proporciona una interfaz para crear objetos en una superclase

Permite a las subclases alterar el tipo de objetos que se crearán.

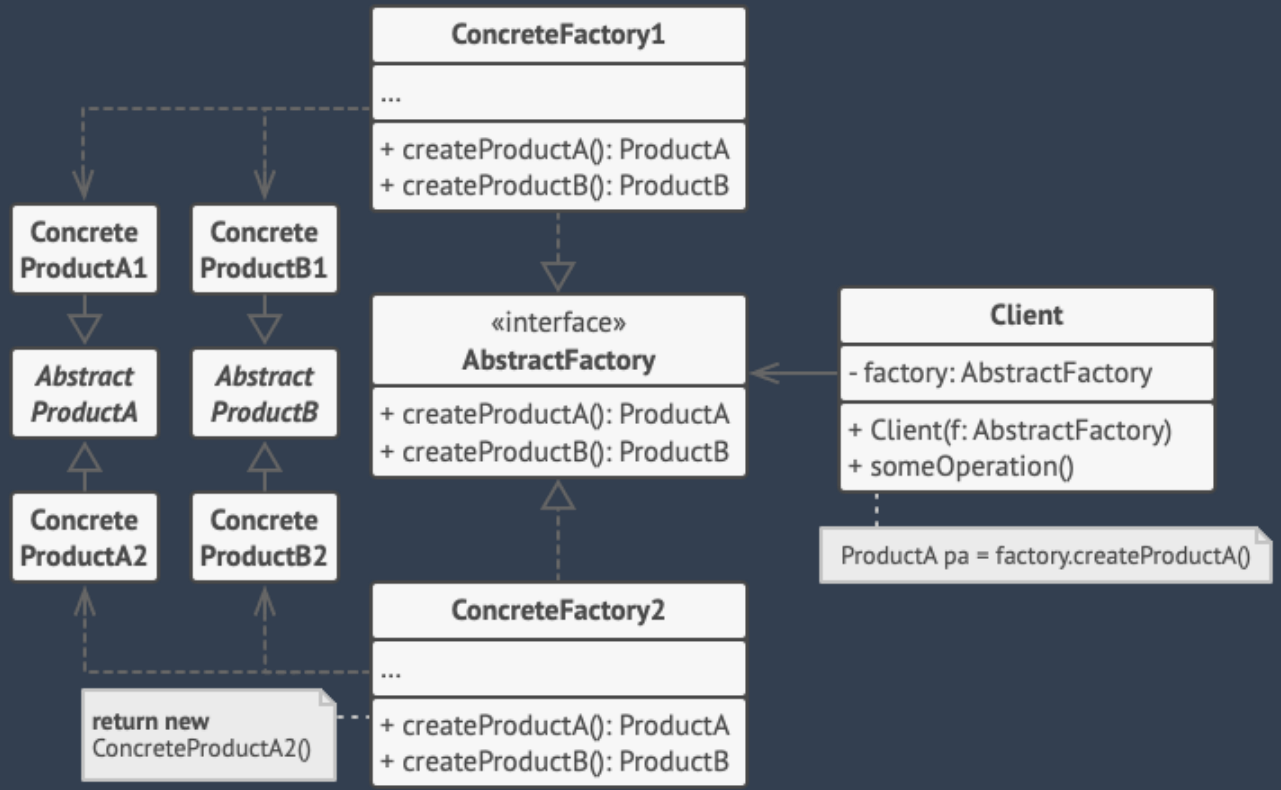
Objetivo: crear objetos a través de una clase generalizada.



ABSTRACT FACTORY

Nos permite producir familias de objetos relacionados sin especificar sus clases concretas.

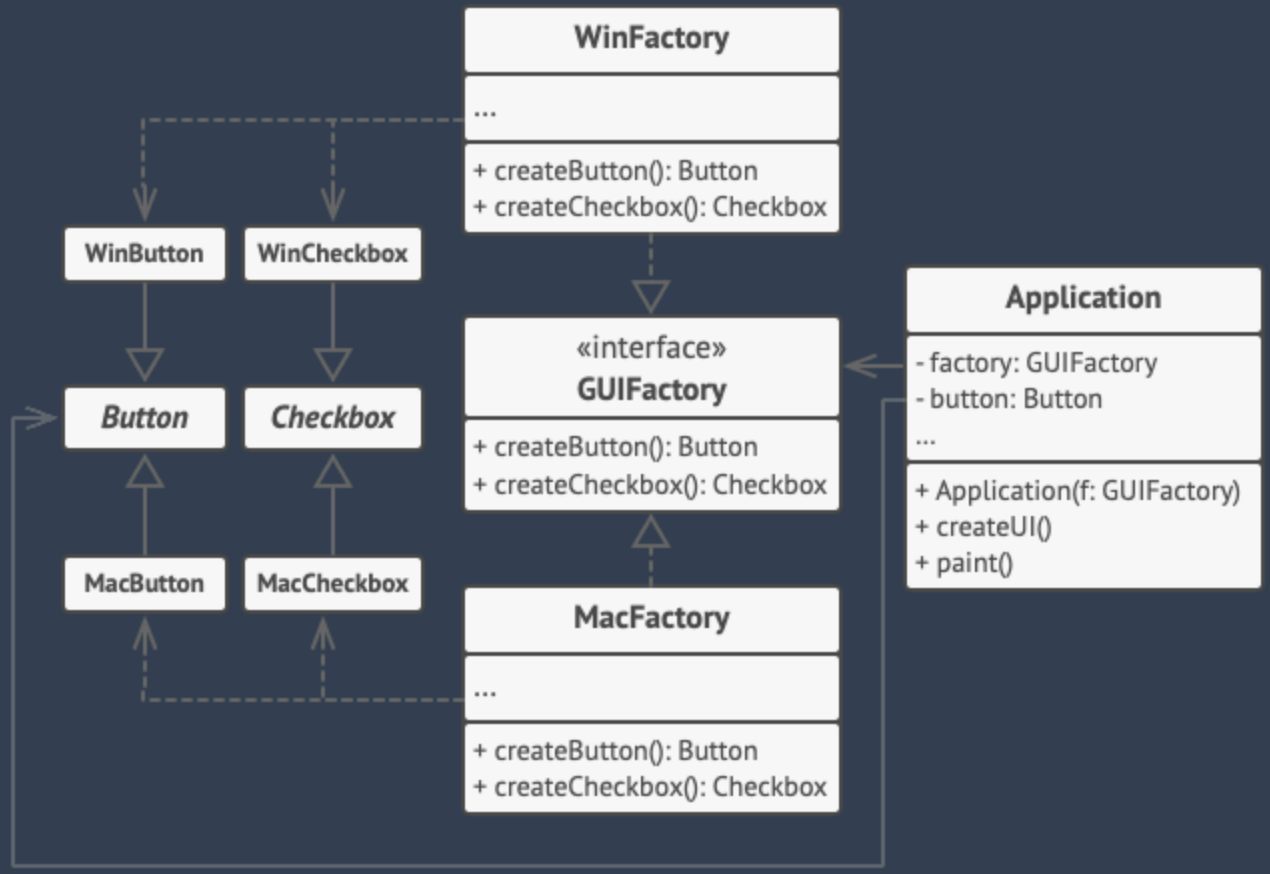
Objetivo: una interfaz con una lista de métodos de creación para todos los productos que son parte de la familia de productos.



ABSTRACT FACTORY

Nos permite producir familias de objetos relacionados sin especificar sus clases concretas.

Objetivo: una interfaz con una lista de métodos de creación para todos los productos que son parte de la familia de productos.

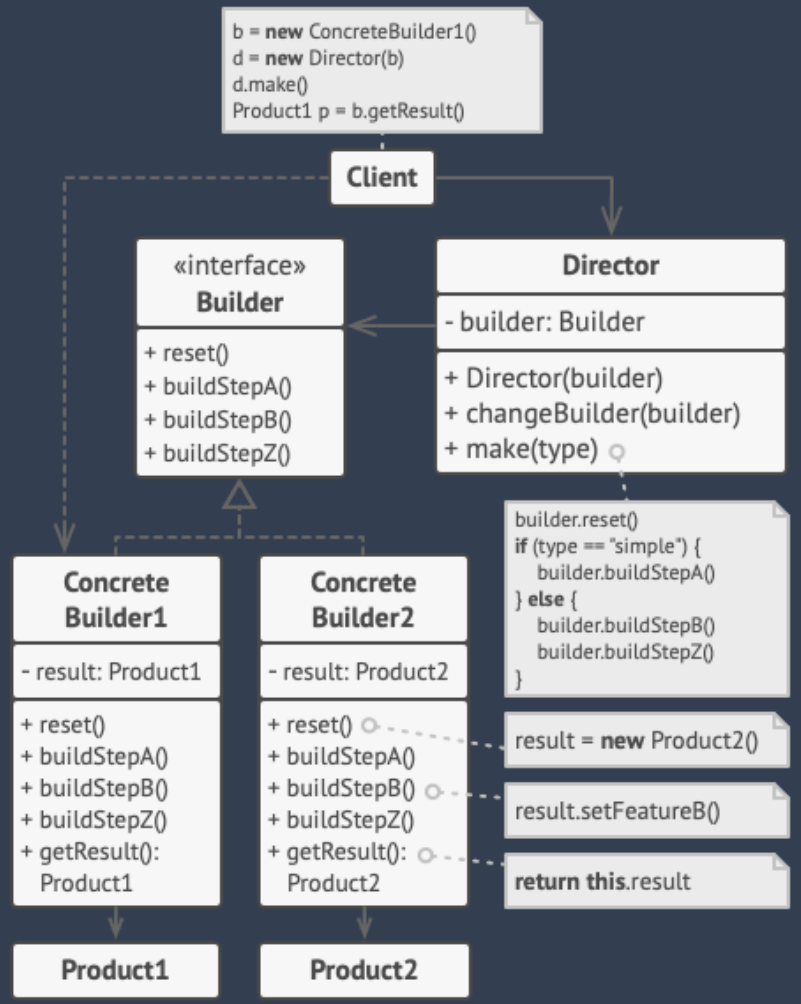


BUILDER

Permite construir objetos complejos paso a paso.

El patrón nos permite producir distintos tipos y representaciones de un objeto empleando el mismo código de construcción.

Objetivo: organiza la construcción de objetos en una serie de pasos.

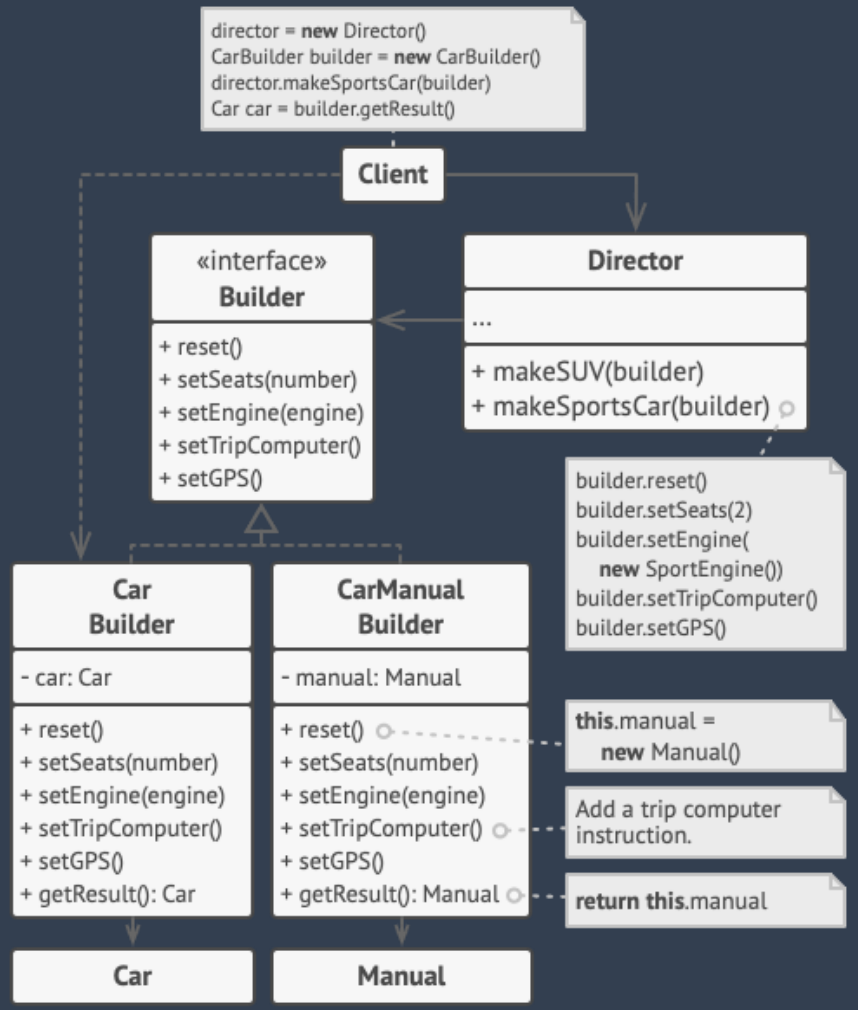


BUILDER

Permite construir objetos complejos paso a paso.

El patrón nos permite producir distintos tipos y representaciones de un objeto empleando el mismo código de construcción.

Objetivo: organiza la construcción de objetos en una serie de pasos.

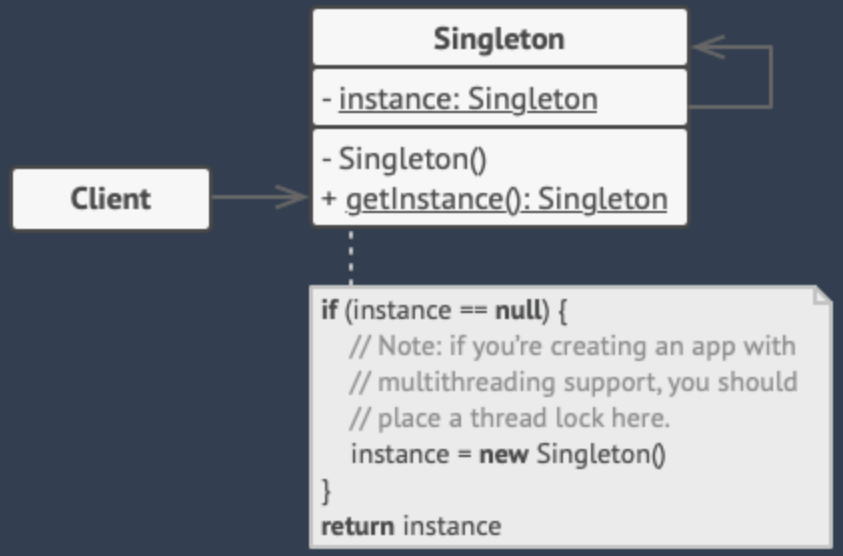


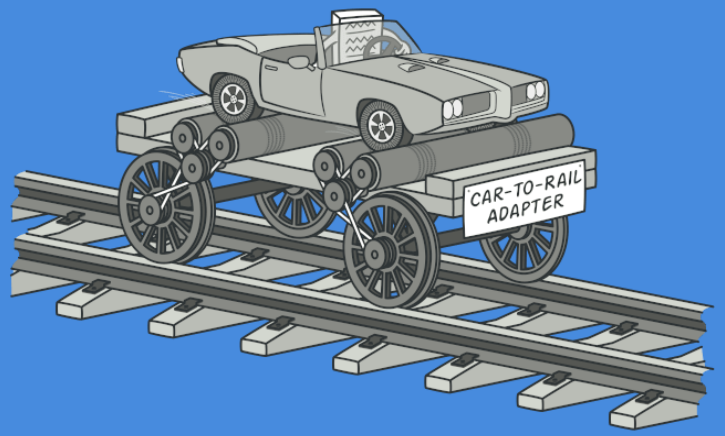
SINGLETON

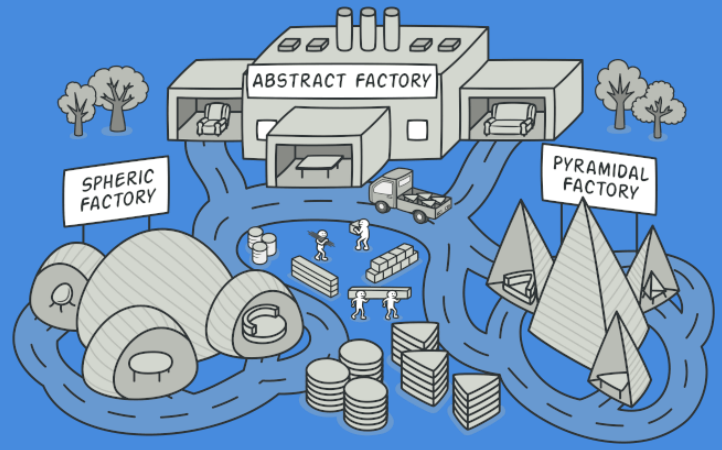
Permite asegurarnos de que una clase tenga una única instancia

Proporciona un punto de acceso global a dicha instancia.

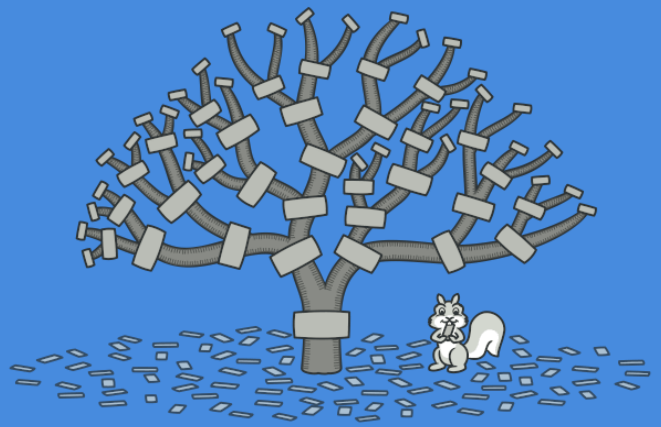
Objetivo: restringe a que una clase solo pueda ser instanciada una vez.

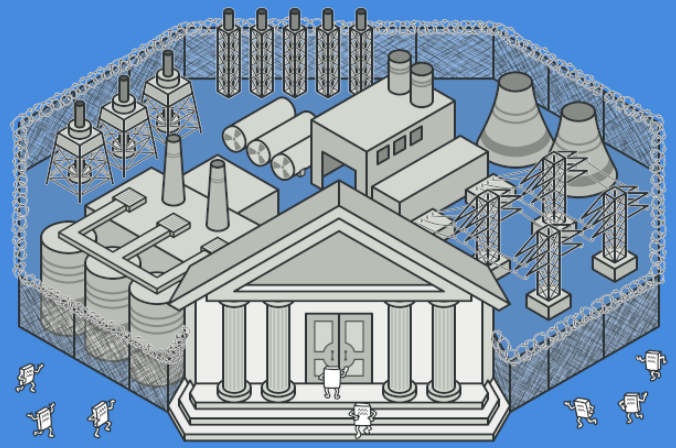






Creacionales

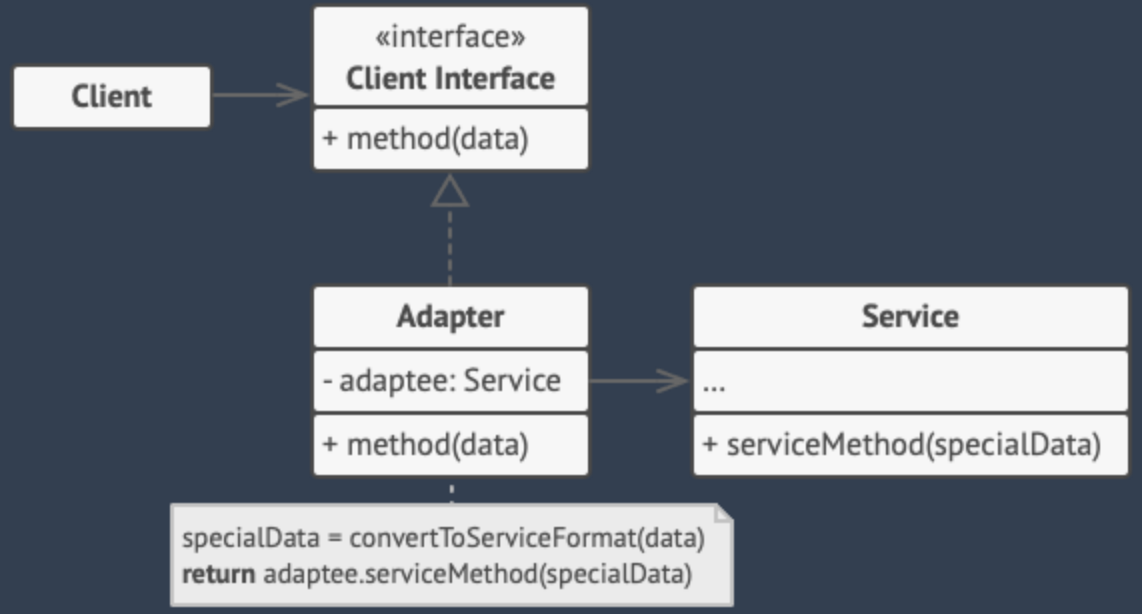




ADAPTADOR

Hacer calzar
un objeto existente pero
que no controlamos
a una interfaz que no
es apropiada para él.

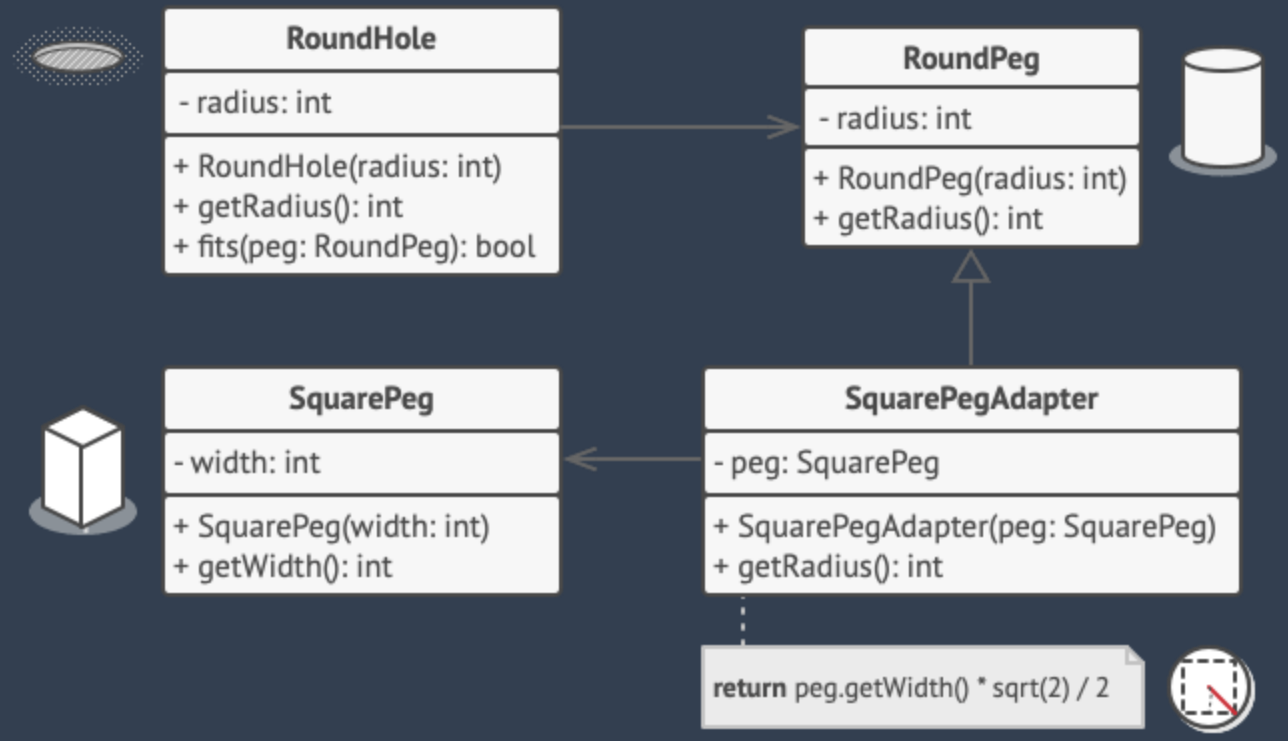
Solución: objeto
que actúa cómo adaptador
o envoltorio.



ADAPTADOR

Hacer calzar
un objeto existente pero
que no controlamos
a una interfaz que no
es apropiada para él.

Solución: objeto
que actúa cómo adaptador
o envoltorio.

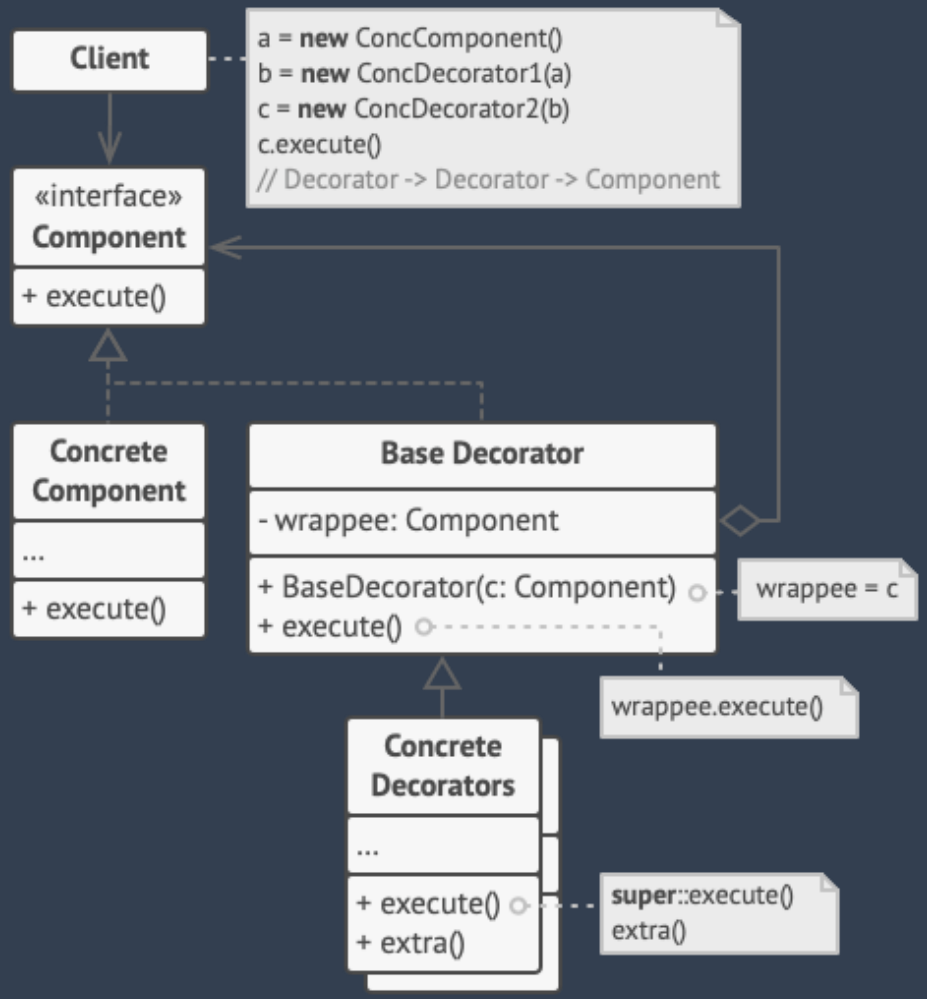


DECORATOR

Anexar responsabilidades en forma dinámica a un objeto.

Permite añadir comportamientos a objetos. Poniendo estos en un objeto wrapper en runtime.

Wrapper: objeto que puede relacionarse con otro y modificar su comportamiento.

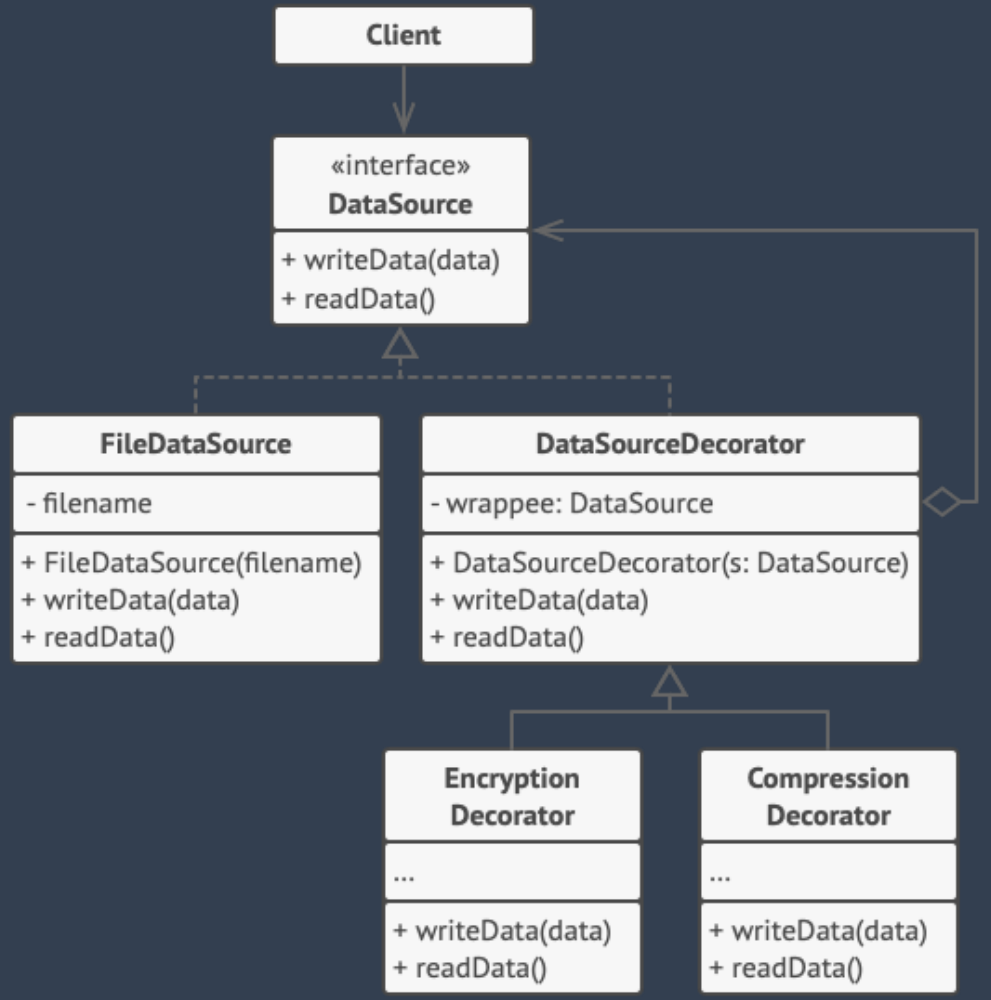


DECORATOR

Anexar responsabilidades
en forma dinámica a un objeto.

Permite
añadir comportamientos a
objetos. Poniendo estos en un
objeto wrapper en runtime.

Wrapper: objeto que
puede relacionarse con otro y
modificar su comportamiento.

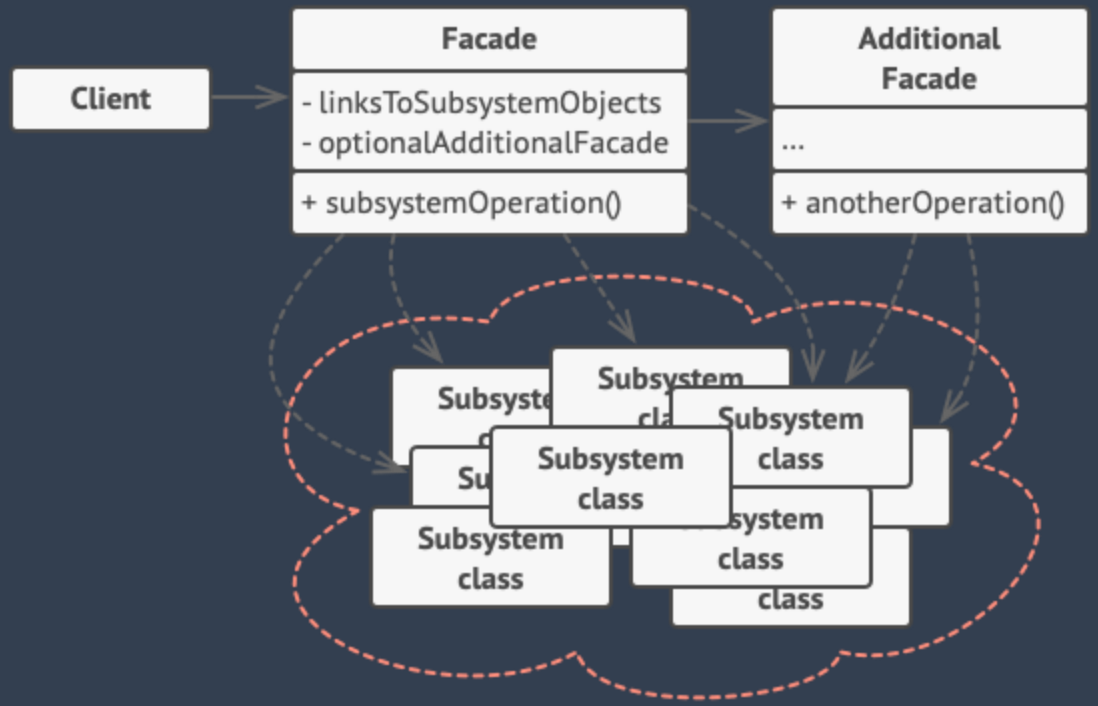


FACADE

Simplificar el uso de un sistema complejo a través de una fachada.

No se busca conocer todos los detalles de ese sistema.

Solución: nueva interfaz simplificada que no expone todas las funcionalidades de la original.

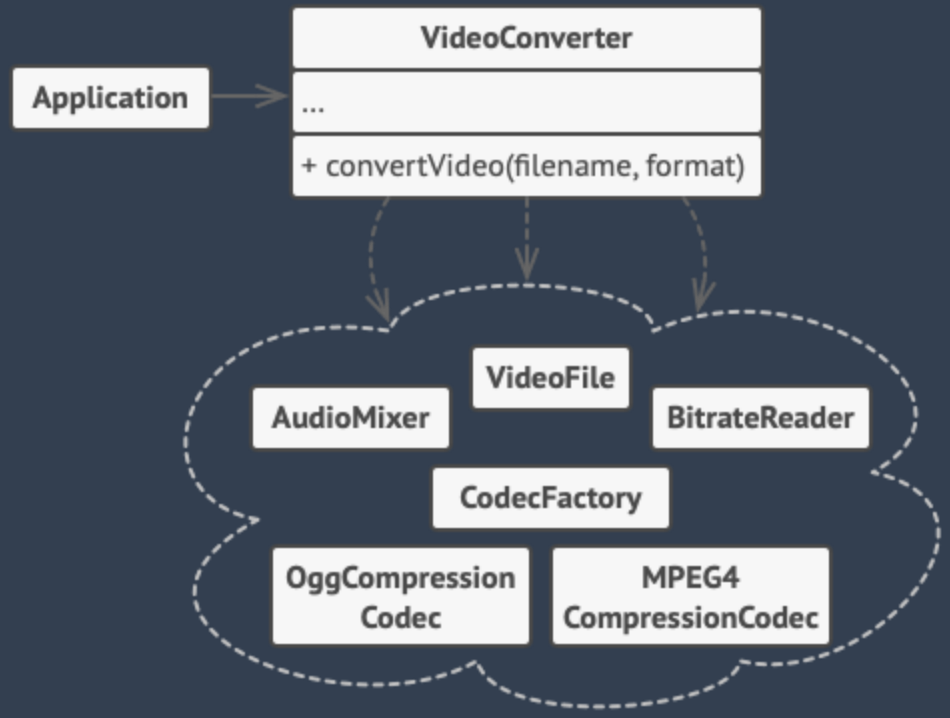


FACADE

Simplificar el uso de un sistema complejo a través de una fachada.

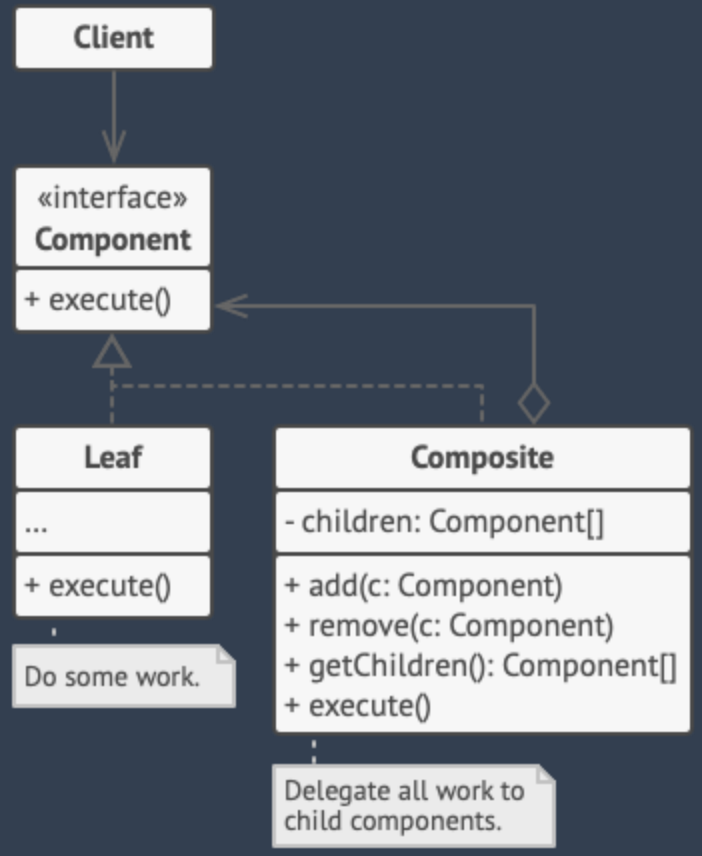
No se busca conocer todos los detalles de ese sistema.

Solución: nueva interfaz simplificada que no expone todas las funcionalidades de la original.



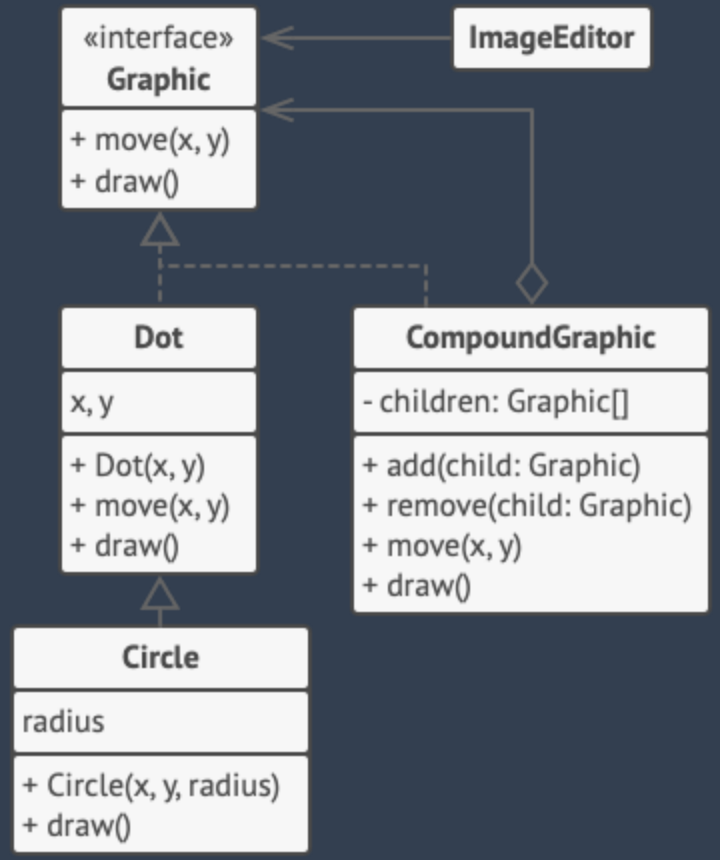
COMPOSITE

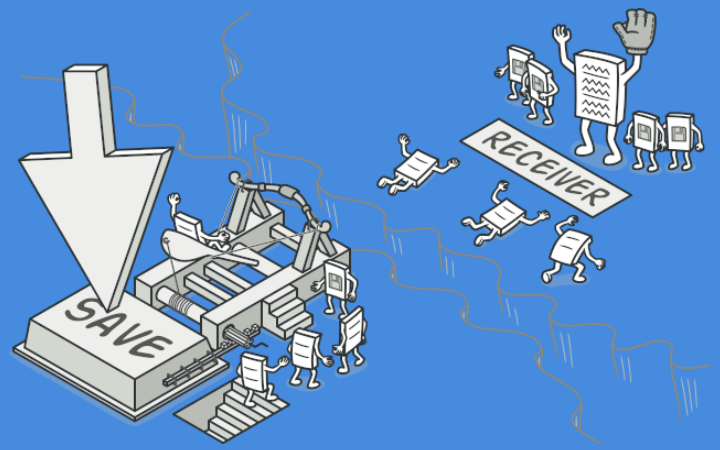
Manejo de objetos que tienen estructuras jerárquicas de forma que una subestructura se maneje igual que la estructura completa.

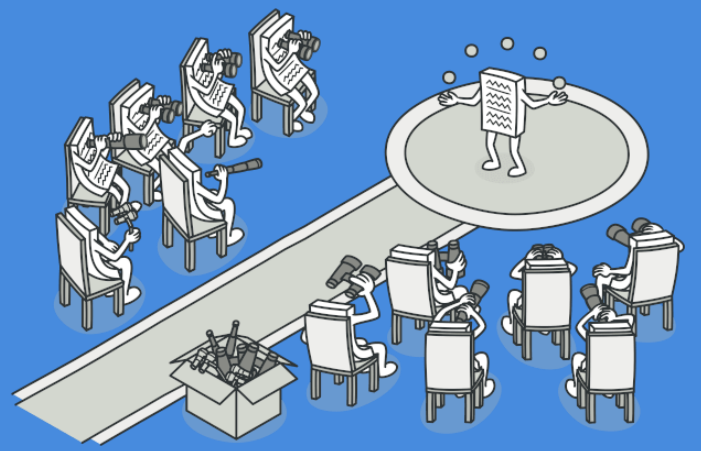


COMPOSITE

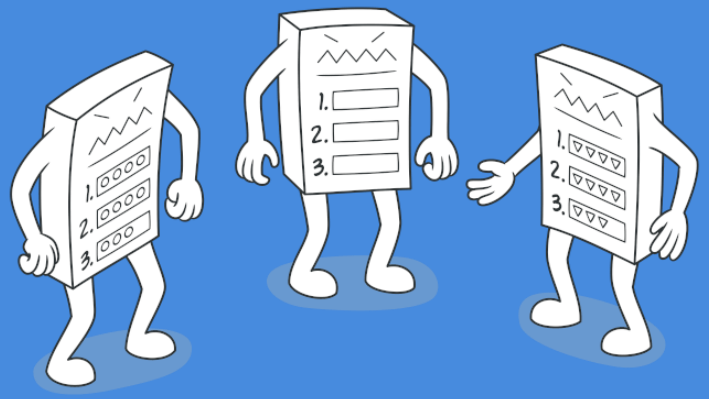
Manejo de objetos que tienen estructuras jerárquicas de forma que una subestructura se maneje igual que la estructura completa.

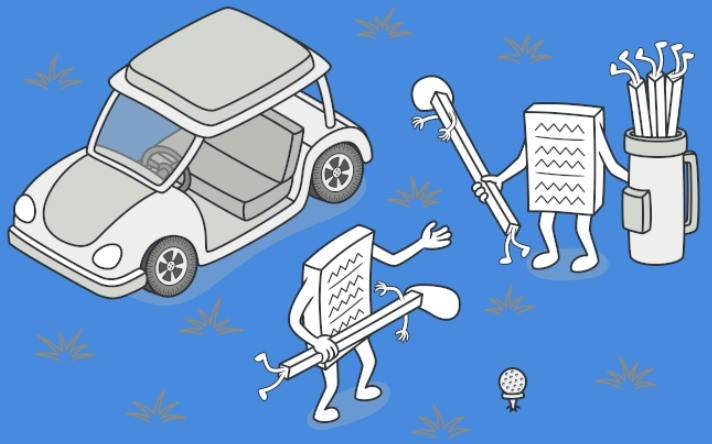






Comportamiento

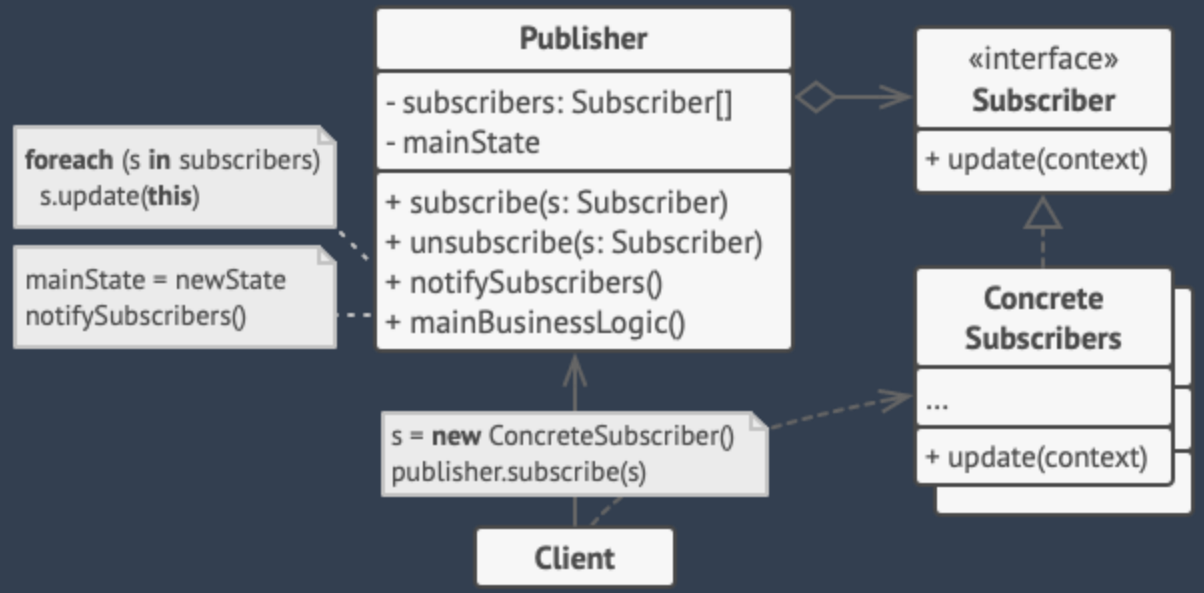




OBSERVER

Mecanismo de suscripción para notificar a objetos observadores acerca de eventos que le ocurren a un sujeto siendo observado.

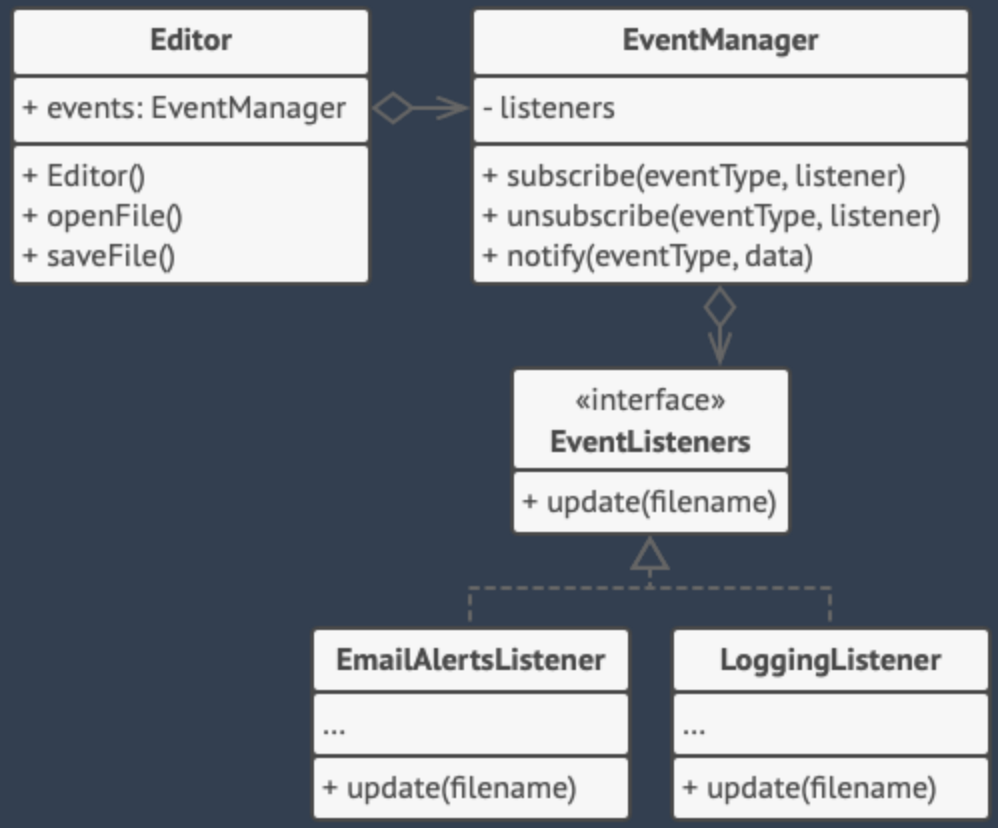
Relación 1:N.



OBSERVER

Mecanismo de suscripción para notificar a objetos observadores acerca de eventos que le ocurren a un sujeto siendo observado.

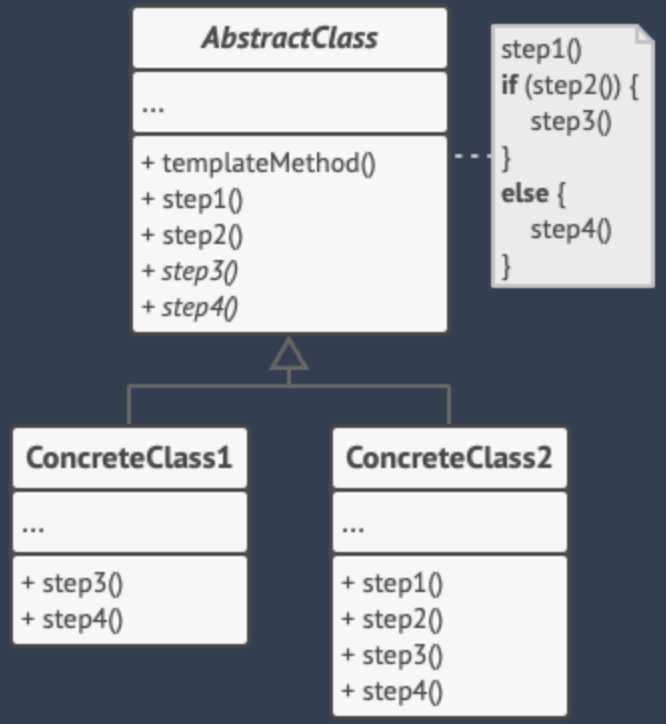
Relación 1:N.



TEMPLATE METHOD

Secuencia de pasos que siempre se repite, pero el detalle de cómo se ejecuta cada paso puede cambiar.

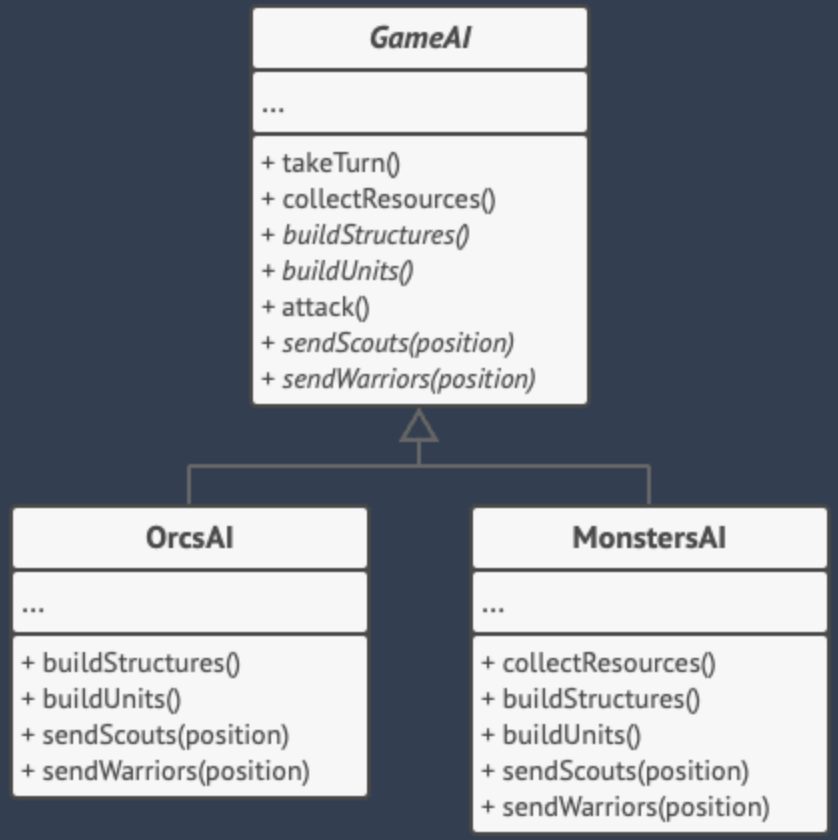
Objetivo: encapsular secuencia de operaciones cómo un método con operaciones abstractas.



TEMPLATE METHOD

Secuencia de pasos que siempre se repite, pero el detalle de cómo se ejecuta cada paso puede cambiar.

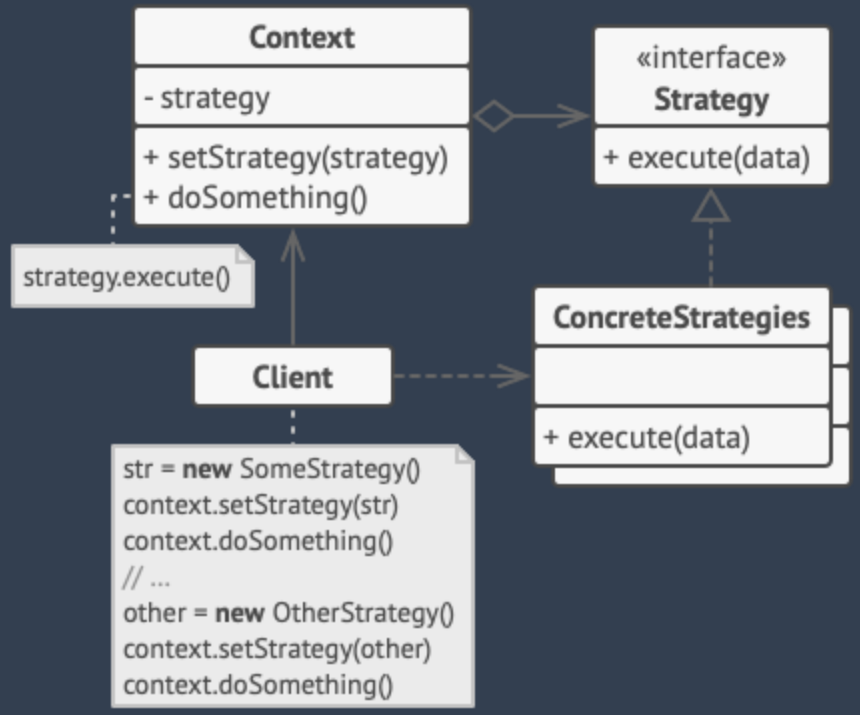
Objetivo: encapsular secuencia de operaciones cómo un método con operaciones abstractas.



STRATEGY

Encapsula un algoritmo en un objeto que puede ser invocado por la clase que lo quiere utilizar.

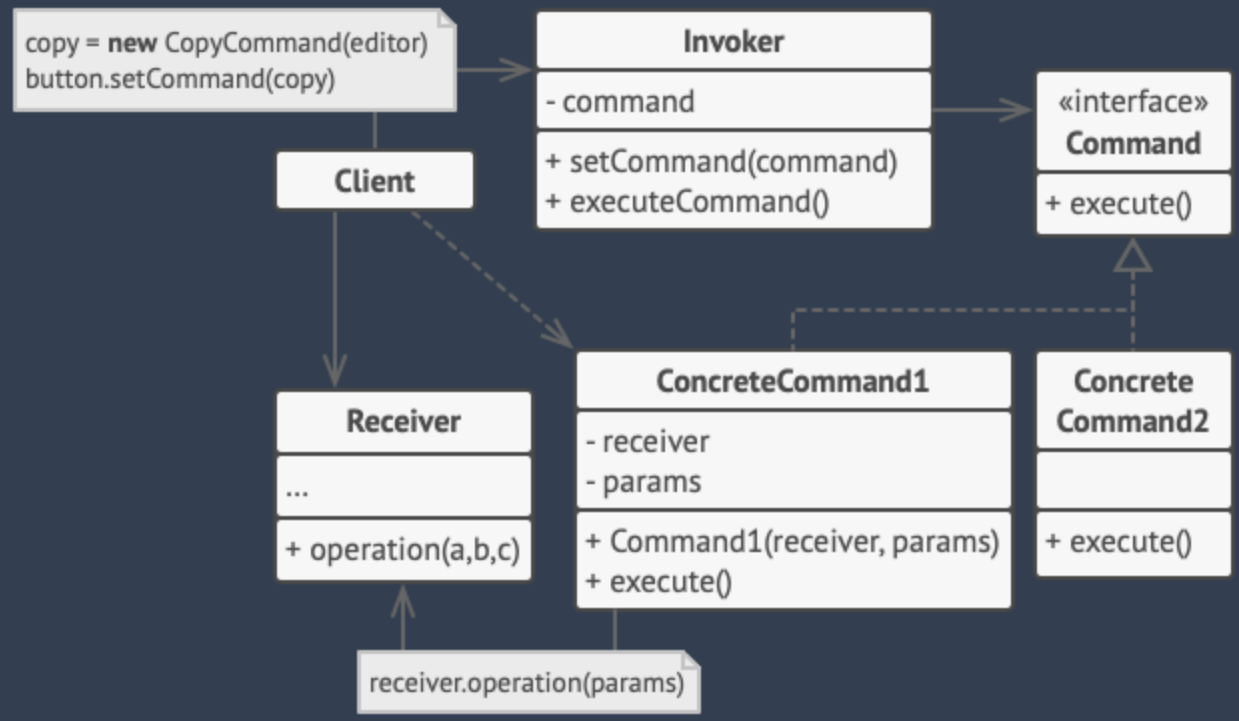
Permite definir una familia de algoritmos, poniéndolos en clases separadas y haciendo sus objetos intercambiables.



COMMAND

Representa acciones como objetos.

Permite ejecución no inmediata (colas).



COMMAND

Representa acciones como objetos.

Permite ejecución no inmediata (colas).