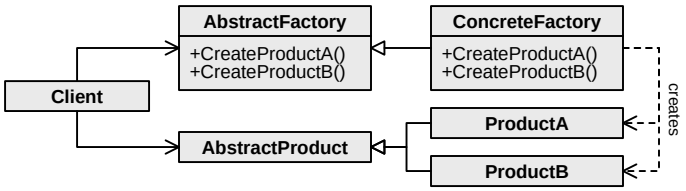
Patrones de Diseño

Patrones Creacionales

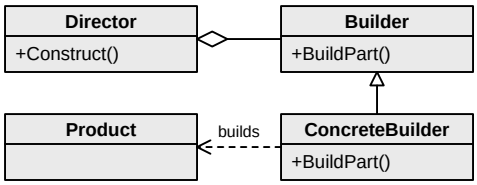
Abstract Factory

permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.



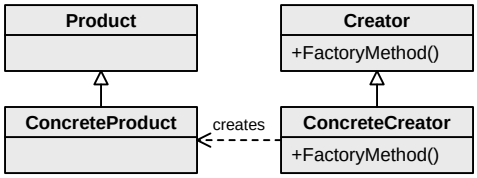
Builder

abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.



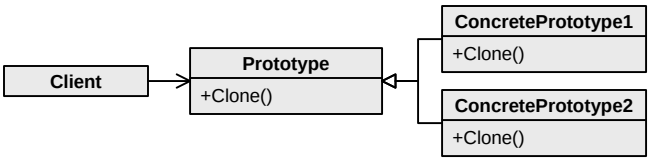
Factory Method

Define una interfaz para crear un objeto pero deja que las subclases decidan qué clase instanciar



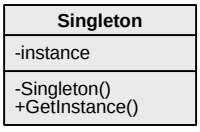
Prototype

crea nuevos objetos clonándolos de una instancia ya existente.



Singleton

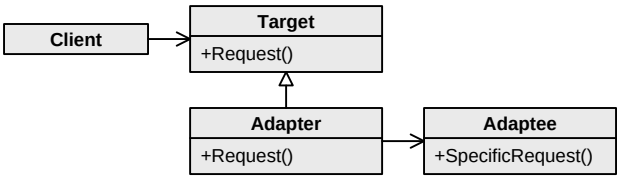
garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia



Patrones Estructurales

Adapter

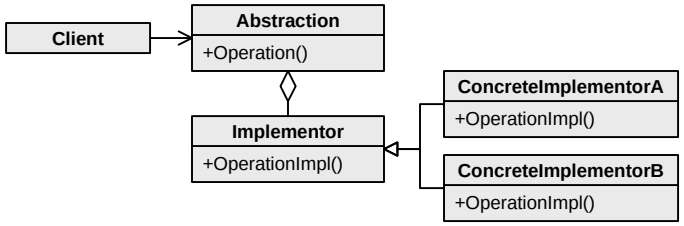
Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla



Patrones Estructurales (continuación)

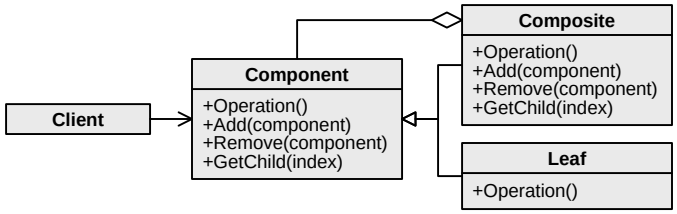
Bridge

Desacopla una abstracción de su implementación.



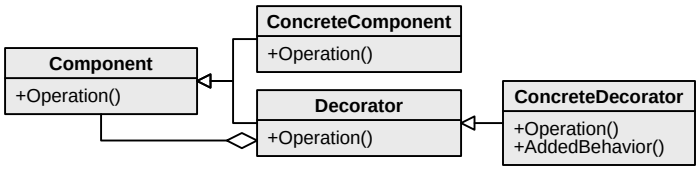
Composite

Permite tratar objetos compuestos como si de uno simple se tratase



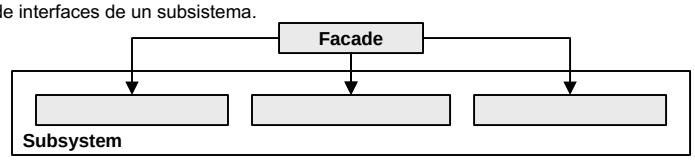
Decorator

Añade funcionalidad a una clase dinámicamente



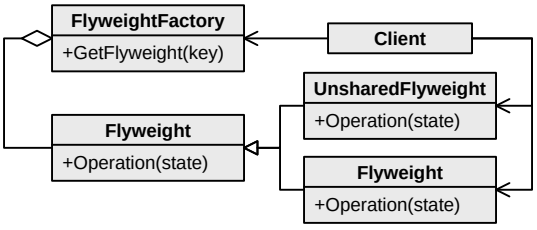
Facade

Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.



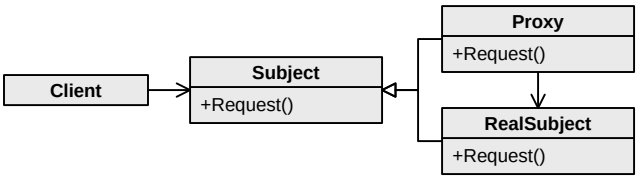
Flyweight

Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información



Proxy

Proporciona un intermediario de un objeto para controlar su acceso

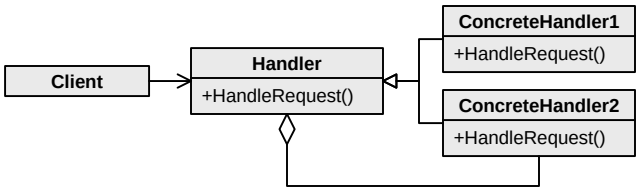


Patrones de Diseño

Patrones de Comportamiento

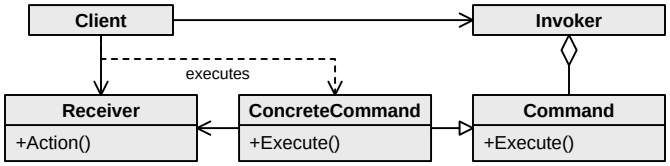
Chain of Responsibility

Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.



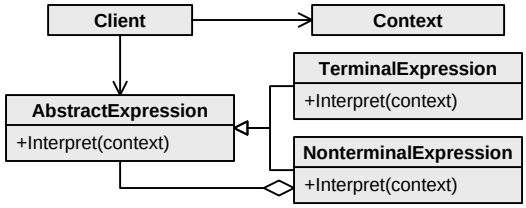
Command

Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.



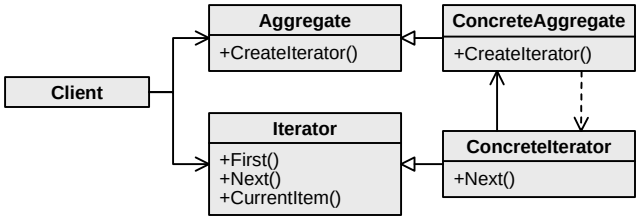
Interpreter

Dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.



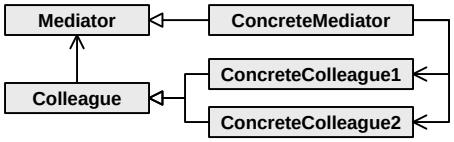
Iterator

Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos



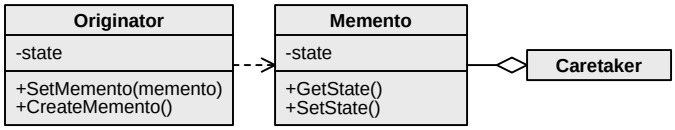
Mediator

Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto



Memento

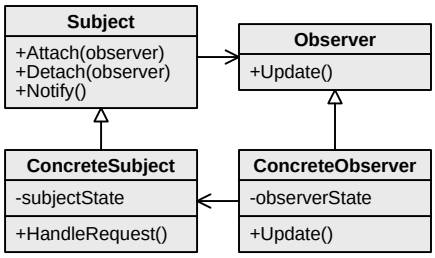
Sin violar la encapsulación, captura el estado de un objeton en un momento dado de forma que ese recuerdo permita modificar el estado y volver a un estado anterior



Patrones de Comportamiento (continuación)

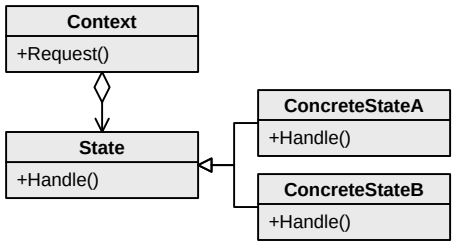
Observer

Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.



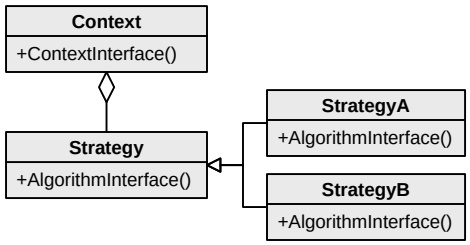
State

Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno



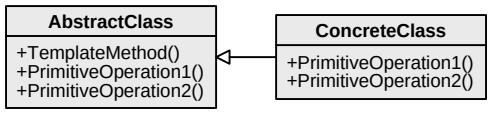
Strategy

Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución



TemplateMethod

Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.



Visitor

Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera.