Patrones GOF

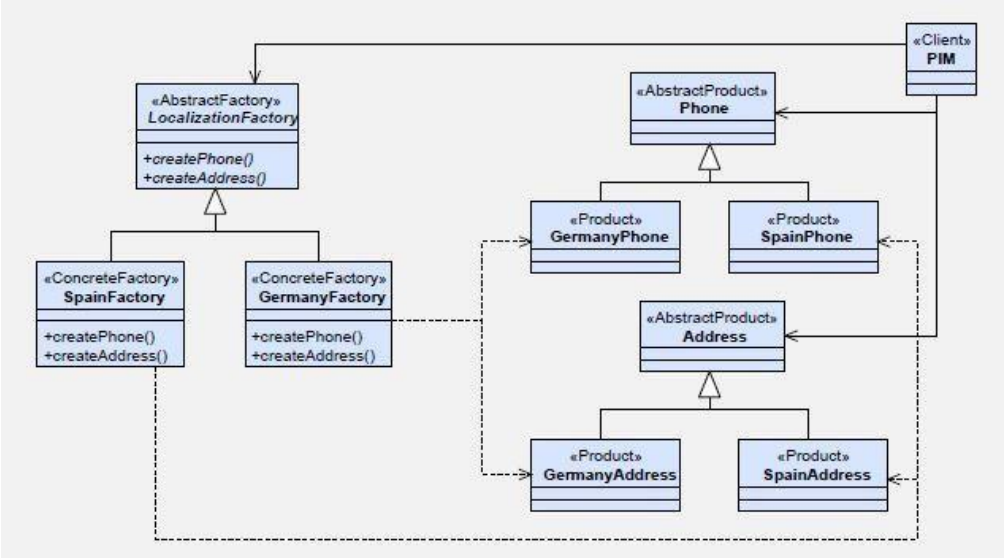
1. Creacionales

– Abstract Factory

Provee una interfaz para crear familias de objetos “producto” relacionados o que dependen entre sí, sin especificar sus clases concretas.

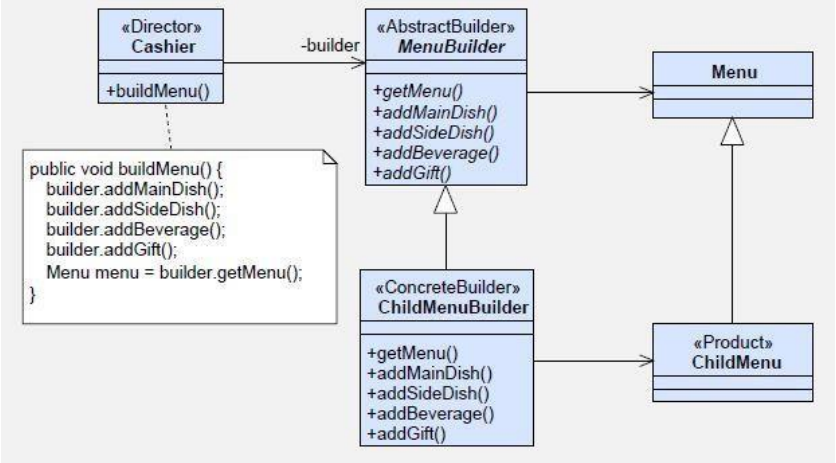
Uso:

- Cuando un sistema debe configurarse con una de las múltiples familias de productos
- Cuando un sistema debe ser independiente de cómo se crean, componen y representan sus productos
- Cuando los productos de la misma familia deben usarse en conjunto, los productos de familias diferentes no deben usarse juntos y esta restricción debe garantizarse
- Solo se revelan las interfaces del producto, las implementaciones permanecen ocultas para los clientes



– Builder

Define una instancia para crear un objeto, pero deja que las subclases decidan qué instanciar y permite un control preciso sobre el proceso de construcción.

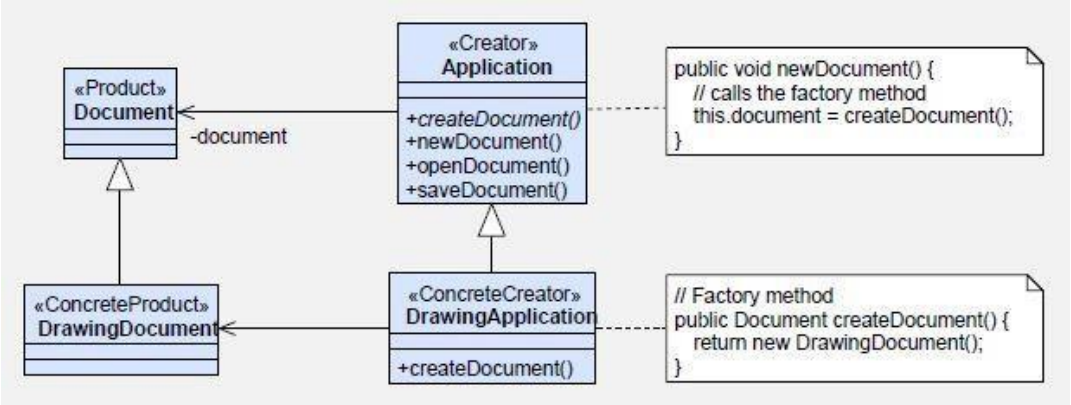


- ## Factory Method

Define una interfaz para crear objetos, pero deja a las subclases decidir qué clase instanciar y hace referencia al nuevo objeto creado a través de una interfaz común.

Uso:

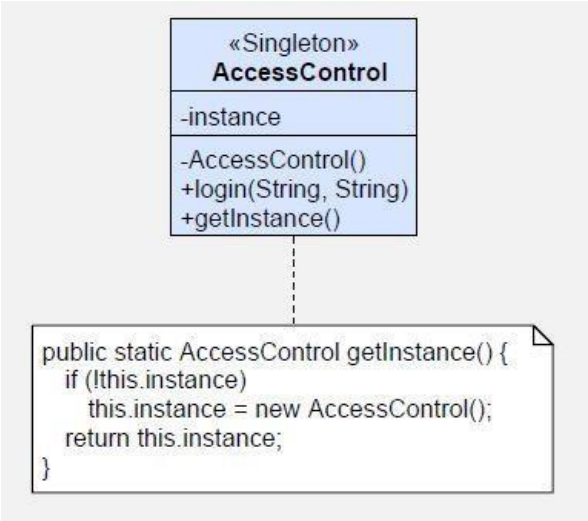
- Cuando un framework delega la creación de objetos derivados de una superclase común a la fábrica
- Cuando necesitamos flexibilidad para agregar nuevos tipos de objetos que la clase debe crear
- Cuando la clase de fábrica base no sabe de qué clases concretas se necesita crear instancias. Delega en sus subclases la creación de objetos concretos
- Cuando subclases de subclases de fábrica son necesarias para conocer las clases concretas que se deben instanciar



- ## Singleton

Asegura que solo se crea una instancia de la clase y provee un punto de acceso global para el objeto.

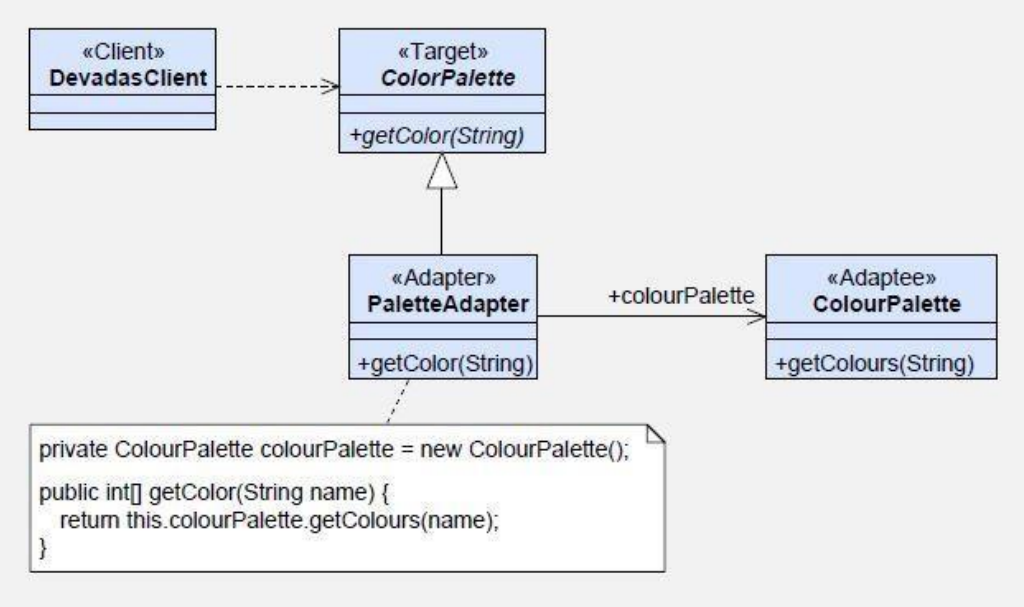
Uso: Debe usarse cuando debemos asegurarnos de que solo se crea una instancia de una clase y cuando la instancia debe estar disponible a través de todo el código. Se debe tener especial cuidado en entornos milti-threading cuando varios subprocesos deben acceder a los mismos recursos a través del mismo objeto singleton.



2. Estructurales

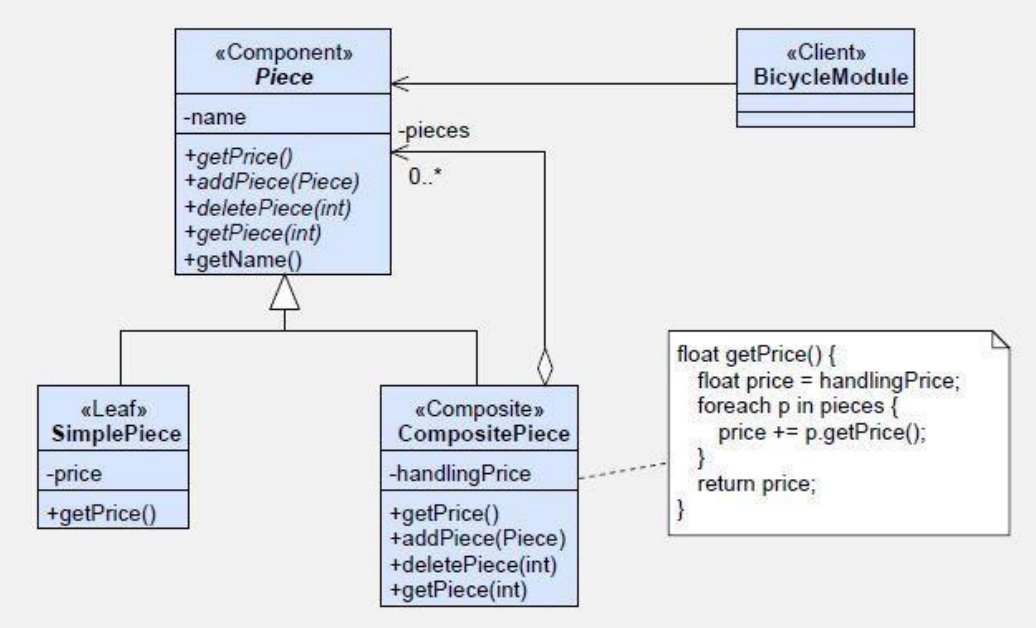
- ## Adapter

Convierte la interfaz de una clase en otra interfaz que los clientes esperan. Adapter permite que clases funcionen juntas, que de otra forma no podría ser debido a interfaces incompatibles.



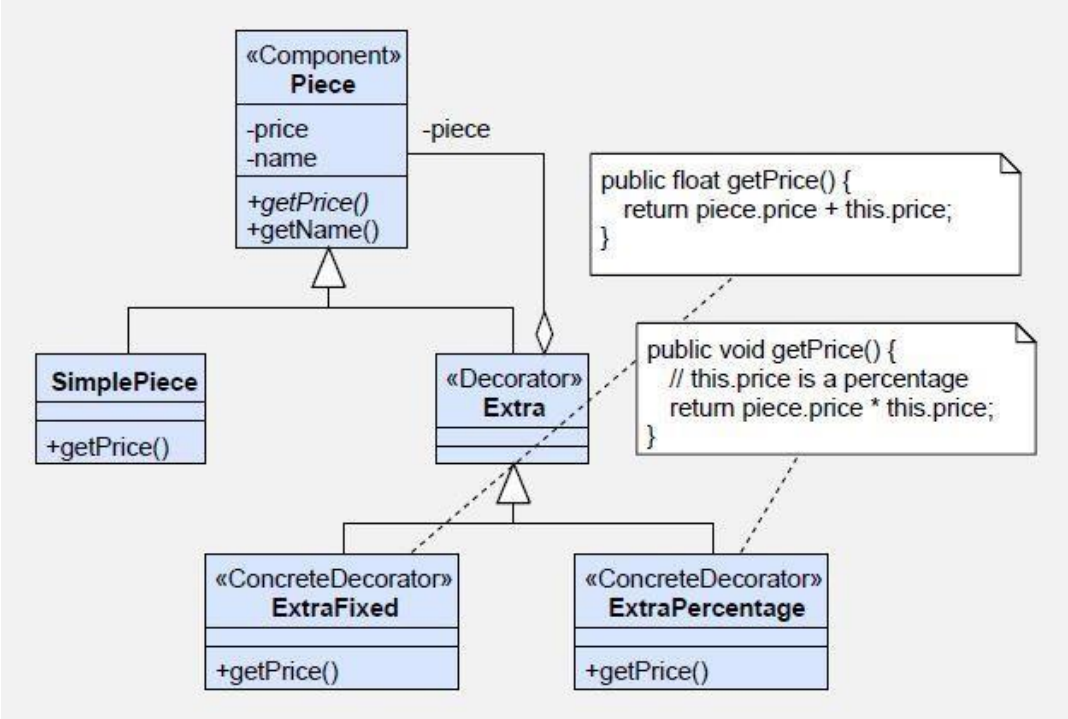
- ## Composite

Compone objetos en estructuras de árboles para representar jerarquías parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los complejos. Si todas las piezas simples tienen el mismo comportamiento debería evitarse tener una clase para cada pieza simple, sino que es mejor agruparlas en la misma clase.



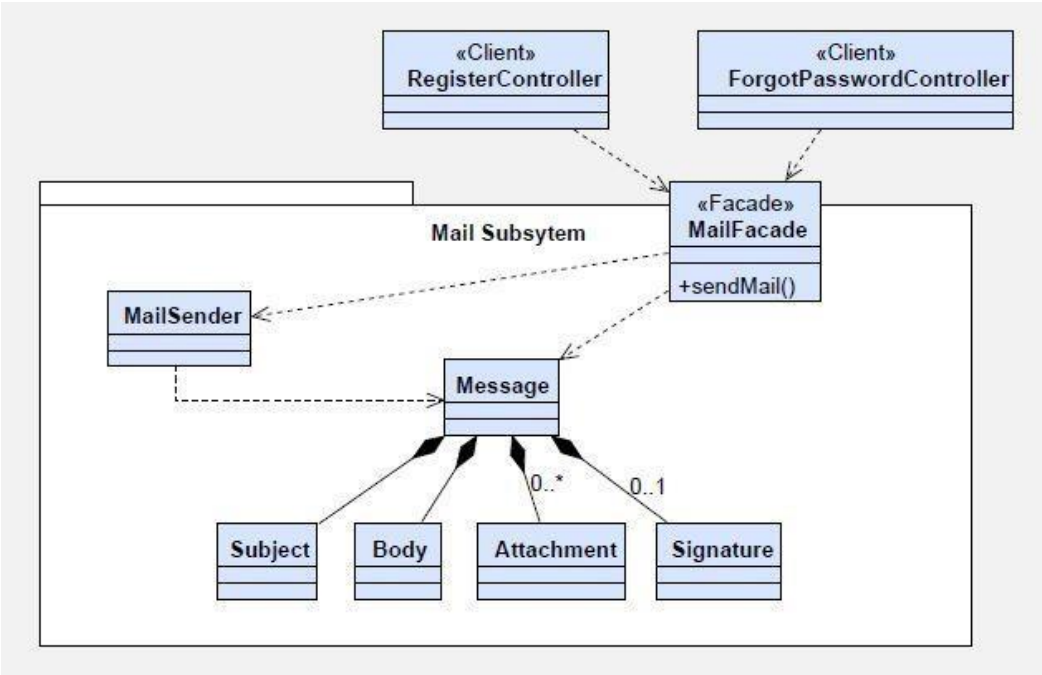
- Decorator

  Agregar responsabilidades/funcionalidades adicionales dinámicamente a un objeto.



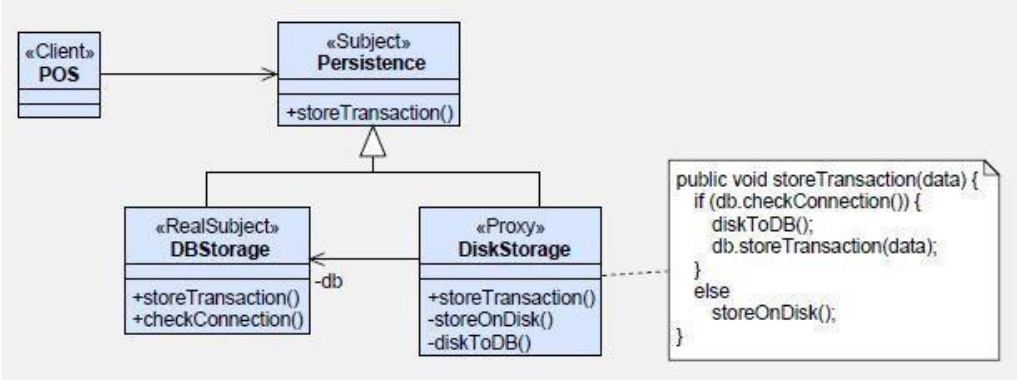
- Facade

  Estructura un entorno de programación y reduce su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre estos.



- Proxy

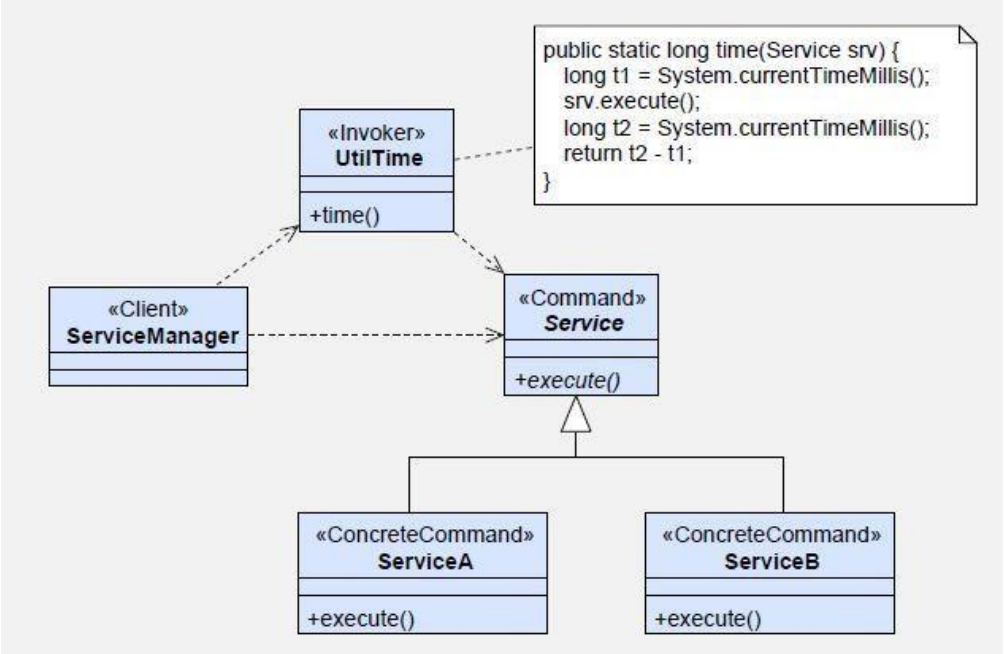
  Proporcionar un intermediario (placeholder) para que un objeto controle las referencias a él.



3. De Comportamiento

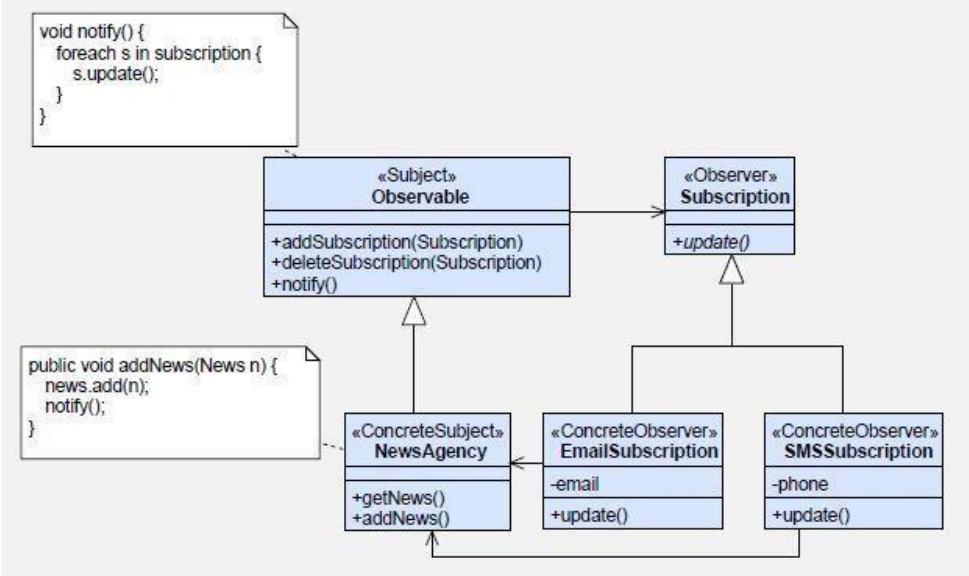
- Command

  Encapsula una solicitud en un objeto, permite la parametrización de clientes con diferentes solicitudes y permite guardar las solicitudes en una cola. Delega la ejecución de los servicios, de manera que ahora será una clase (Invoker) la encargada de ejecutarlos. Para esto es necesario convertir los servicios (operaciones) en objetos, manteniendo su inicialización donde se hacía originalmente, y añadiéndoles un método que permita ejecutarlos.



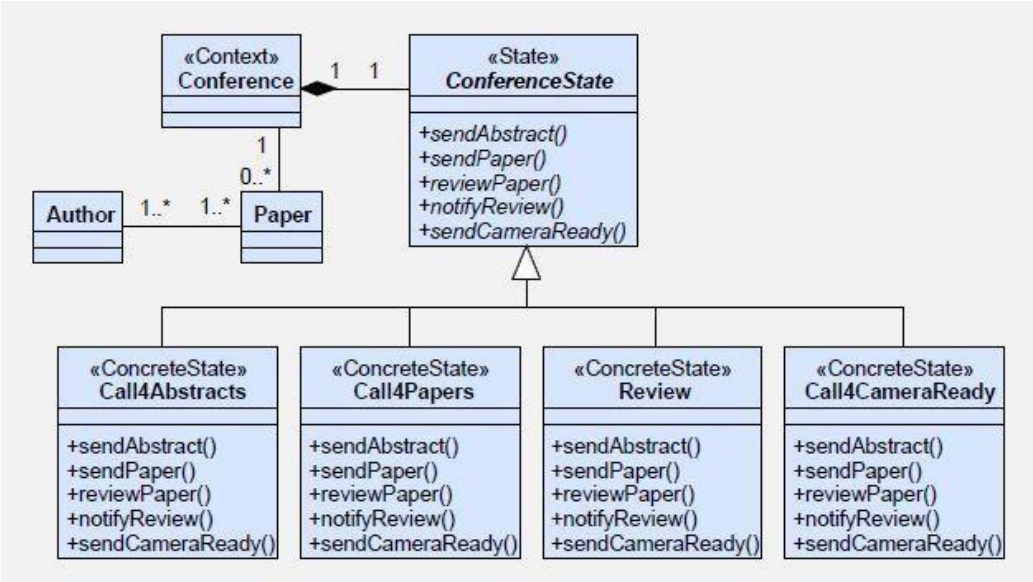
- ## Observer

Define una dependencia de uno a muchos entre los objetos para que cuando un objeto cambie de estado, todos sus dependientes sean notificados y actualizados automáticamente.



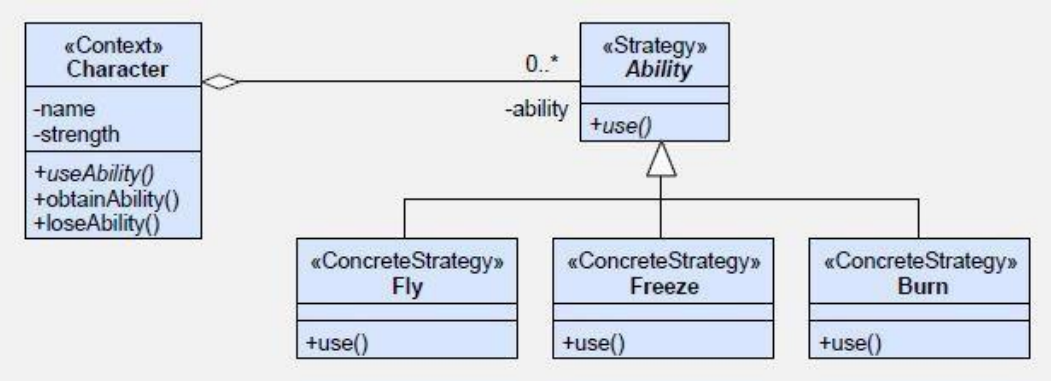
- ## State

Se utiliza cuando el comportamiento de un objeto cambia dependiendo del estado del mismo.



- ## Strategy

Define una jerarquía de clases que representan algoritmos, encapsula cada una y hace que sean intercambiables. Estos algoritmos pueden ser intercambiados por la aplicación en tiempo de ejecución. Strategy permite que el algoritmo varíe independientemente de los clientes que lo usan.



- ## Template Method

Define el esqueleto de un algoritmo en una operación, difiriendo algunos pasos a subclases. Template Method permite a las subclases redefinir ciertos pasos de un algoritmo sin permitirles cambiar la estructura del algoritmo.