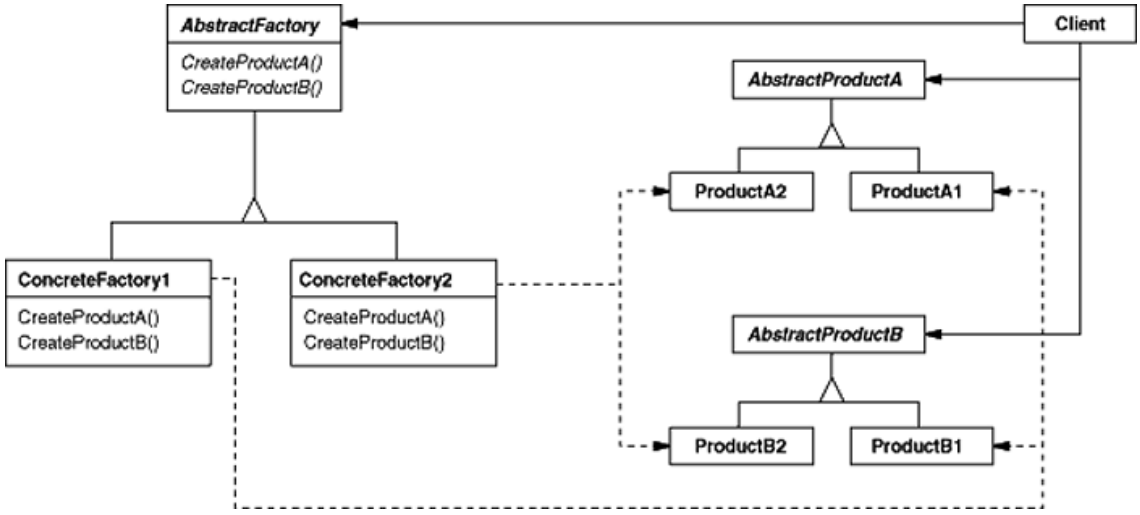
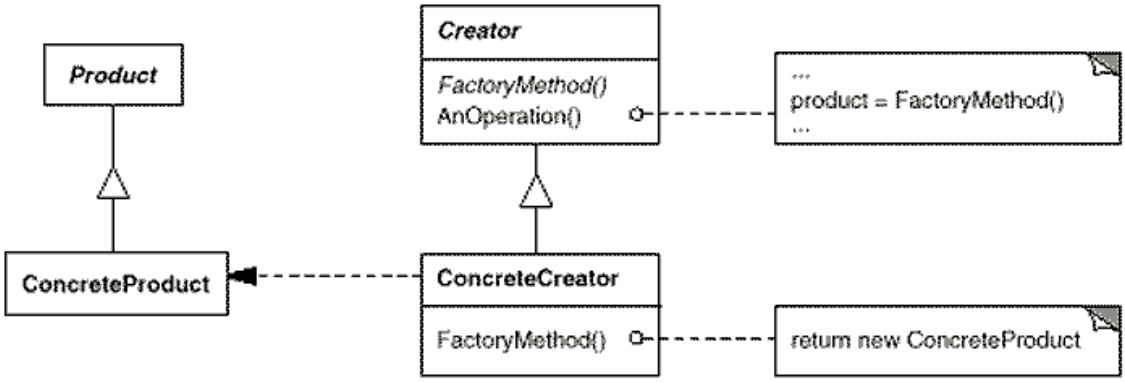
PATRONES CREACIONALES

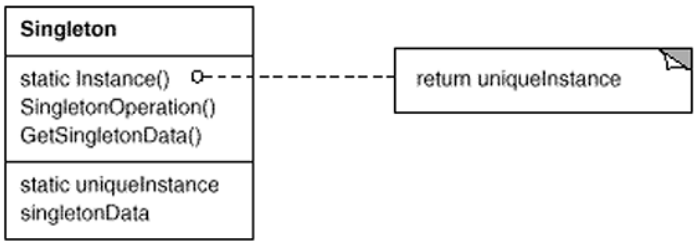
Abstract Factory: provee una interfaz para crear familias de objetos producto relacionados o que dependen entre sí, sin especificar sus clases concretas.



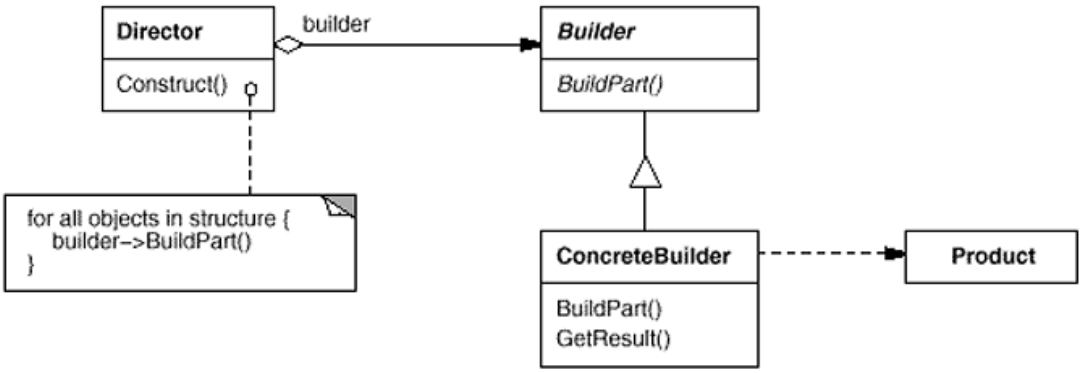
Factory Method: define una interfaz para crear un objeto delegando la decisión de qué clase crear en las subclases. Este enfoque también puede ser llamado constructor "virtual".



Singleton: asegura que una determinada clase sea instanciada una y sólo una vez, proporcionando un único punto de acceso global a ella.

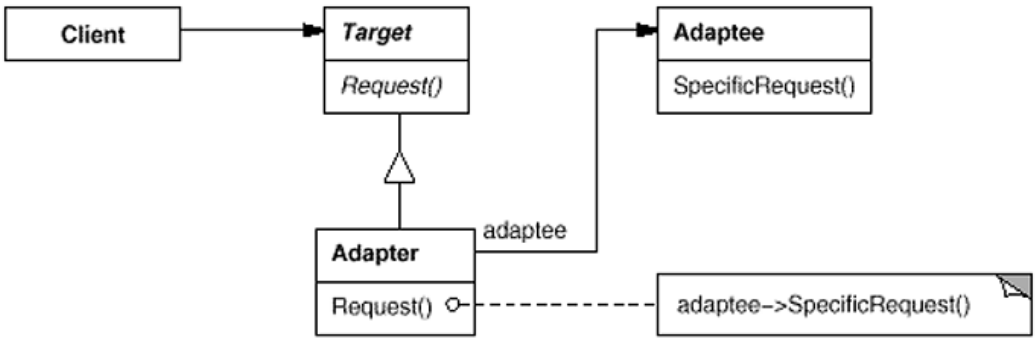


Builder: separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones. Simplifica la construcción de objetos con estructura interna compleja y permite la construcción de objetos paso a paso.

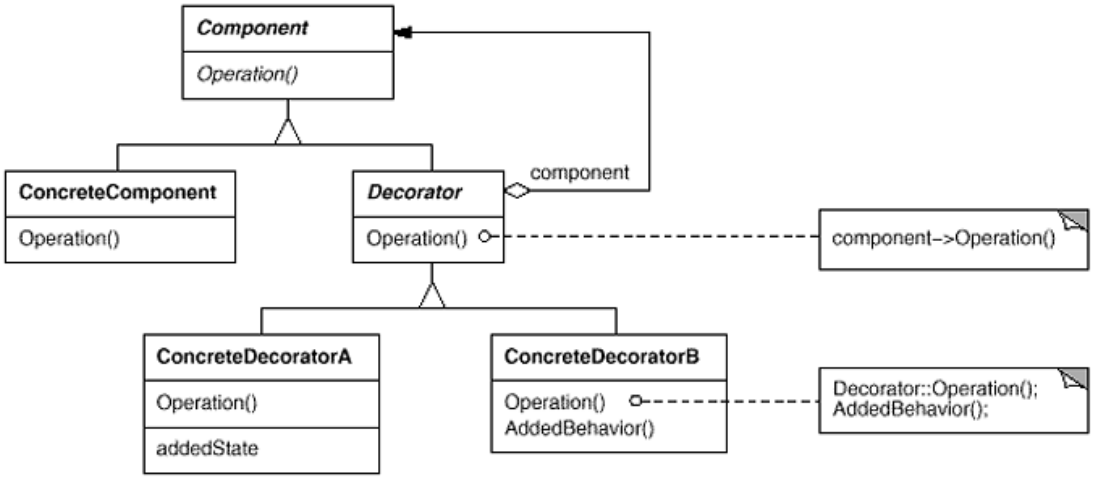


PATRONES ESTRUCTURALES

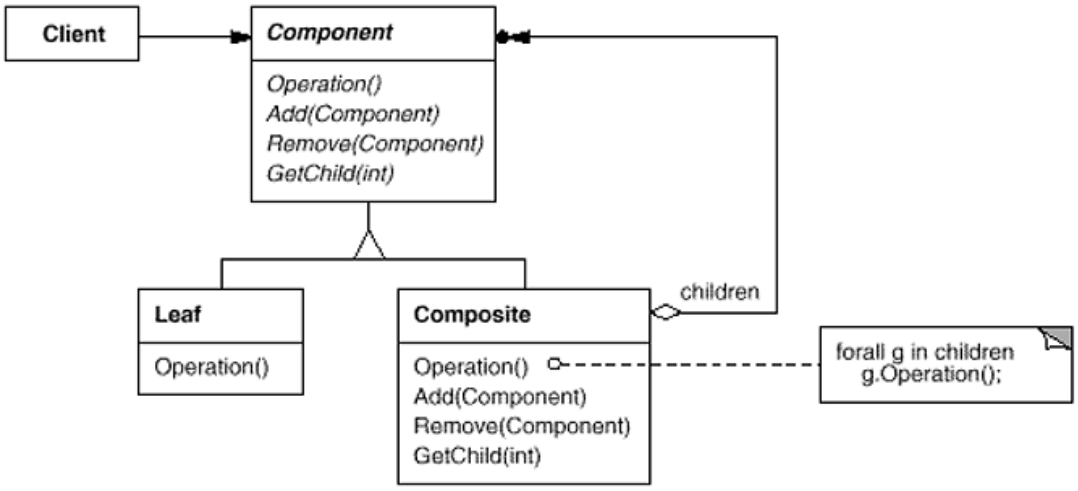
Adapter: oficia de intermediario entre dos clases cuyas interfaces son incompatibles de manera tal que puedan ser utilizadas en conjunto.



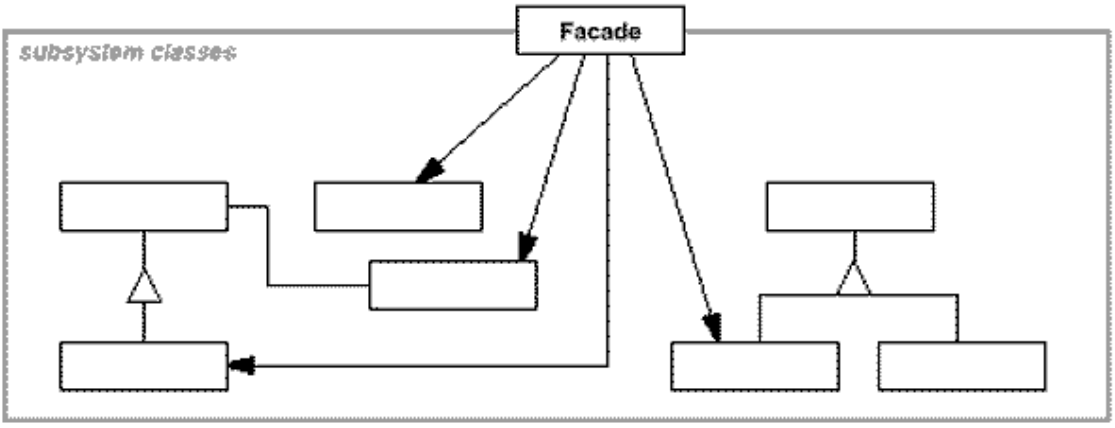
Decorator: agrega o limita responsabilidades adicionales a un objeto de forma dinámica, proporcionando una alternativa flexible a la herencia para extender funcionalidad.



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.

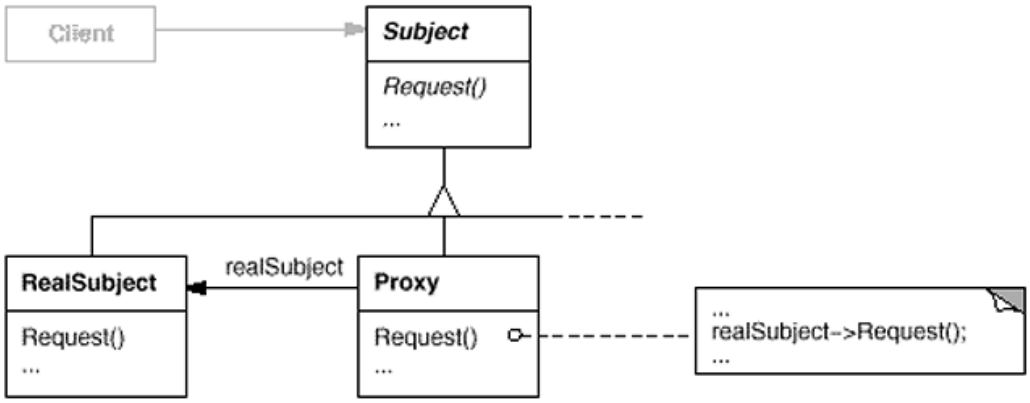


Facade: proporciona una interfaz simplificada para un conjunto de interfaces de subsistemas. Define una interfaz de alto nivel que hace que un subsistema sea más fácil de usar.



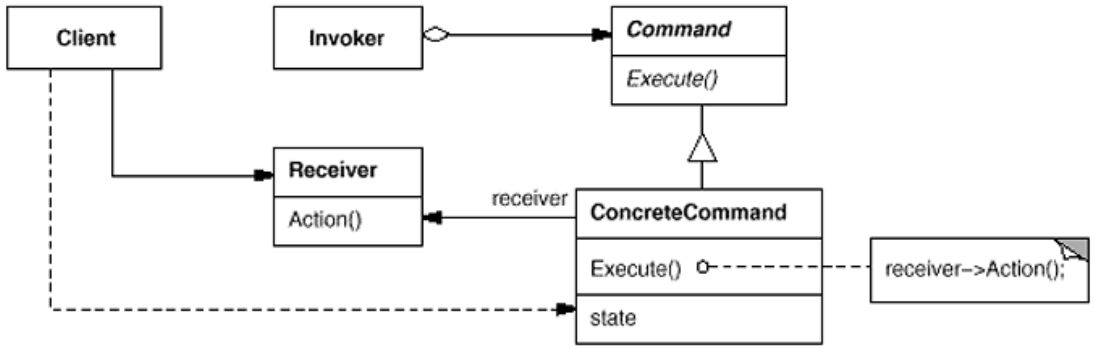
Proxy: provee un sustituto o representante de un objeto para controlar el acceso a éste. Este patrón posee las siguientes variantes:

- Proxy remoto: se encarga de representar un objeto remoto como si estuviese localmente.
- Proxy virtual: se encarga de crear objetos de gran tamaño bajo demanda.
- Proxy de protección: se encarga de controlar el acceso al objeto representado.

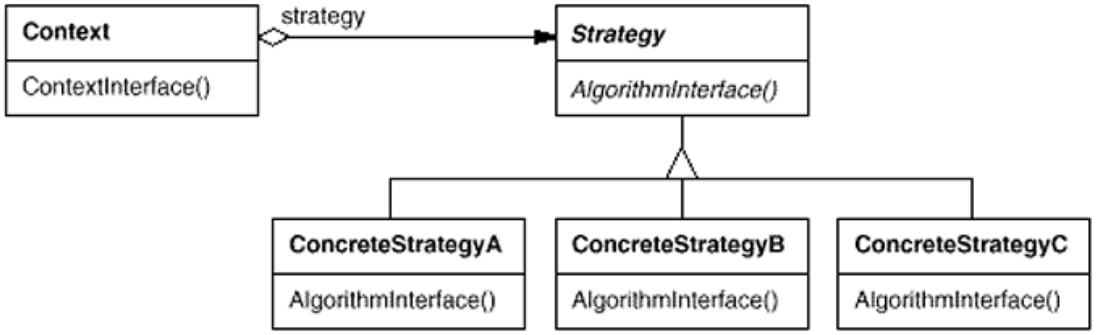


PATRONES DE COMPORTAMIENTO

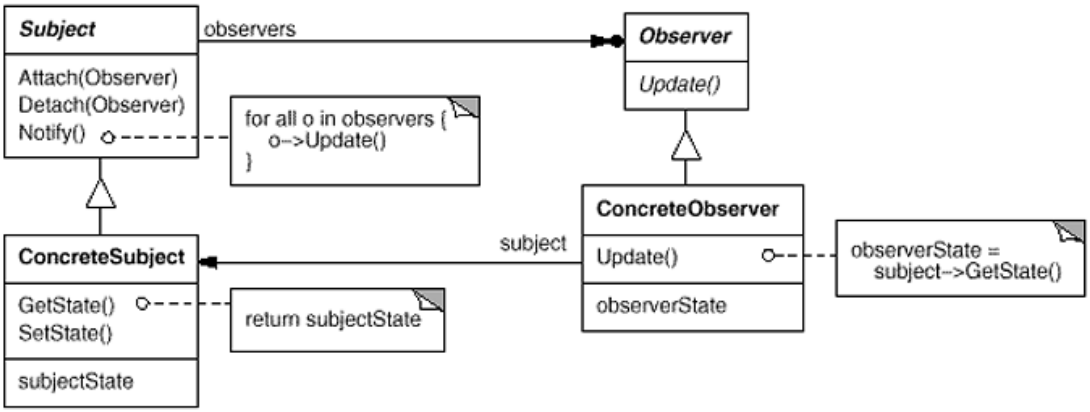
Command: representa una solicitud con un objeto, de manera tal de poder parametrizar a los clientes con distintas solicitudes, encolarlas o llevar un registro de las mismas, y poder deshacer las operaciones. Estas solicitudes, al ser representadas como un objeto también pueden pasarse como parámetro o devolverse como resultados.



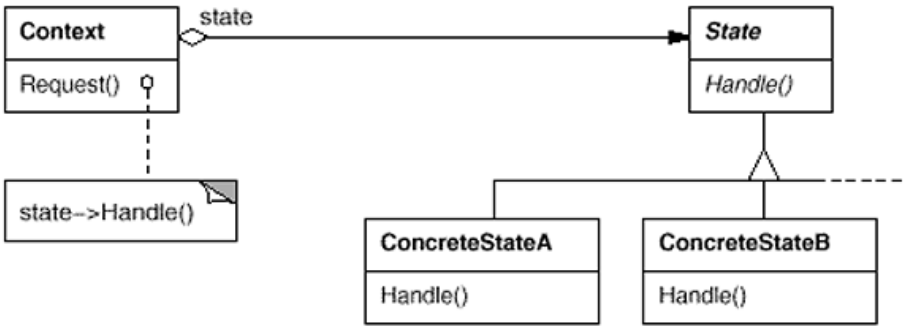
Strategy: define una jerarquía de clases que representan algoritmos, los cuales son intercambiables. Estos algoritmos pueden ser intercambiados por la aplicación en tiempo de ejecución.



Observer: brinda un mecanismo que permite a un componente transmitir de forma flexible mensajes a aquellos objetos que hayan expresado interés en él. Estos mensajes se disparan cuando el objeto ha sido actualizado, y la idea es que quienes hayan expresado interés reaccionen ante este evento.



State: permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. El objeto parecerá que cambió de clase.



Template Method: define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.