



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 15

Patrones de diseño

IIC2143 - Ingeniería de Software
Sección 1

Rodrigo Saffie

rasaffie@uc.cl

2 de mayo de 2018

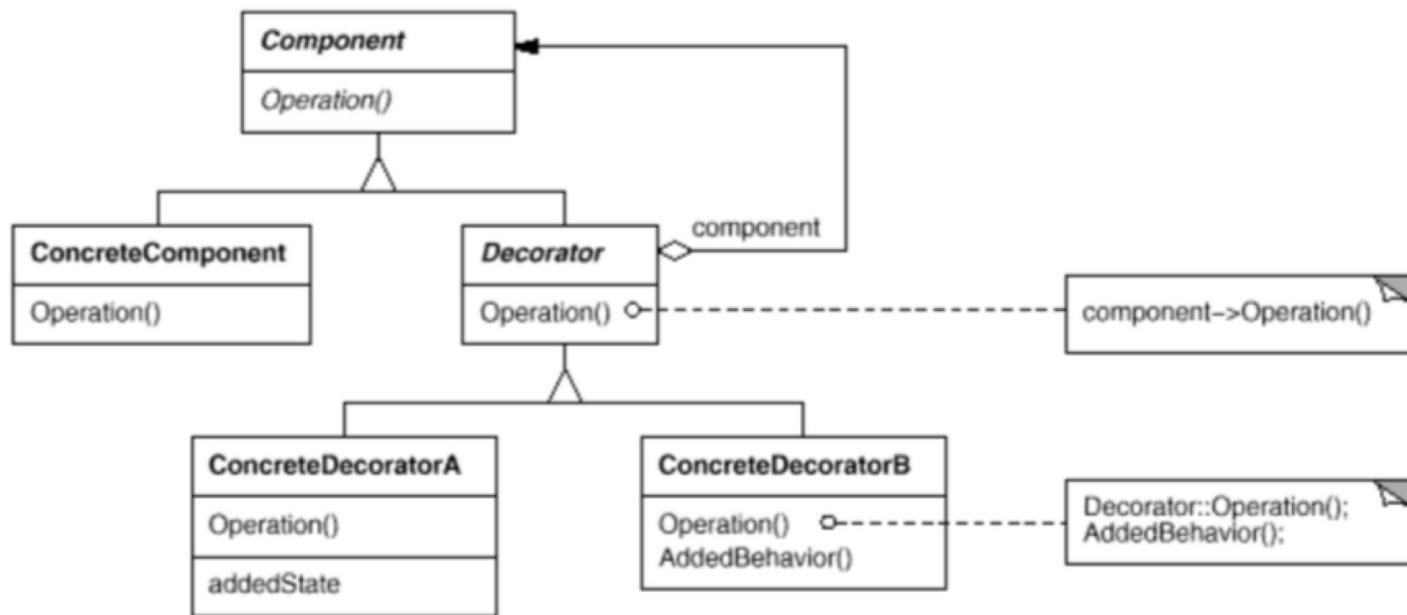
Patrones estructurales

Decorator

Agrega responsabilidad adicional a un objeto de manera dinámica. Decoradores permiten una alternativa flexible para extender funcionalidad de sub-clases.

Patrones estructurales

Decorator



[Ejemplo1](#) y [Ejemplo2](#)

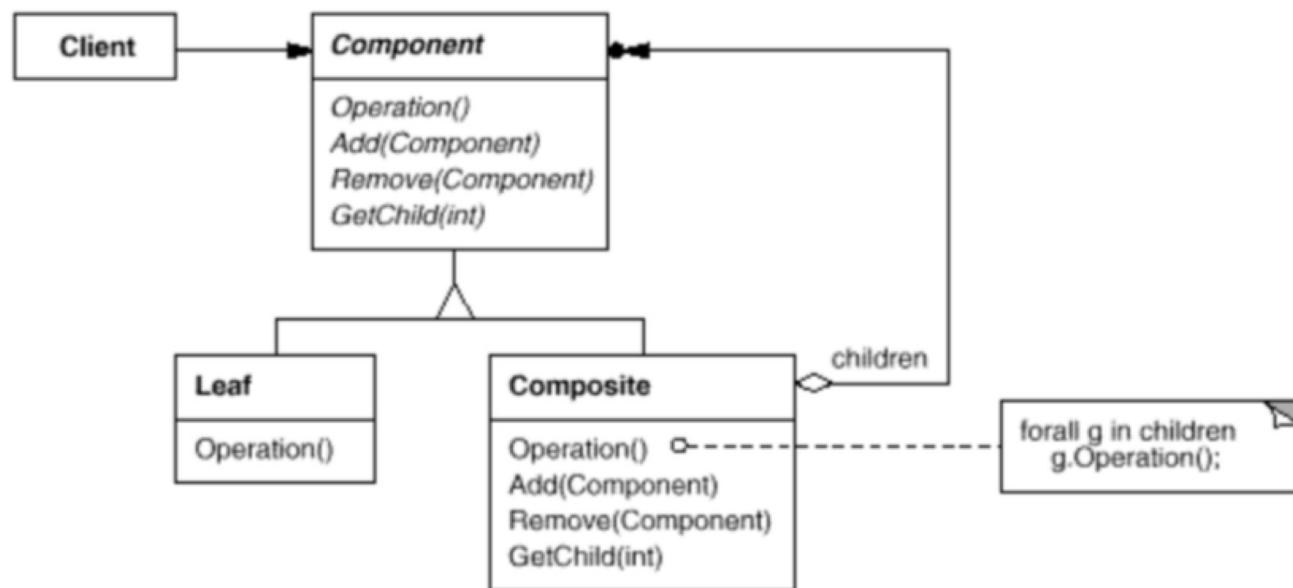
Patrones estructurales

Composite

Compone objetos en estructuras de árboles para representar jerarquías completas. *Composite* permite a los clientes tratar objetos y composiciones de objetos de igual manera.

Patrones estructurales

Composite



Ejemplo

Patrones creacionales

Singleton

Garantiza que una clase tenga solamente una instancia, y provee un acceso global a la instancia.

Patrones creacionales

Singleton



Ejemplo

Patrones creacionales

Factory Method

Define una interfaz para la creación de un objeto, pero delega a las subclases que decidan qué clase instanciar.

Patrones creacionales

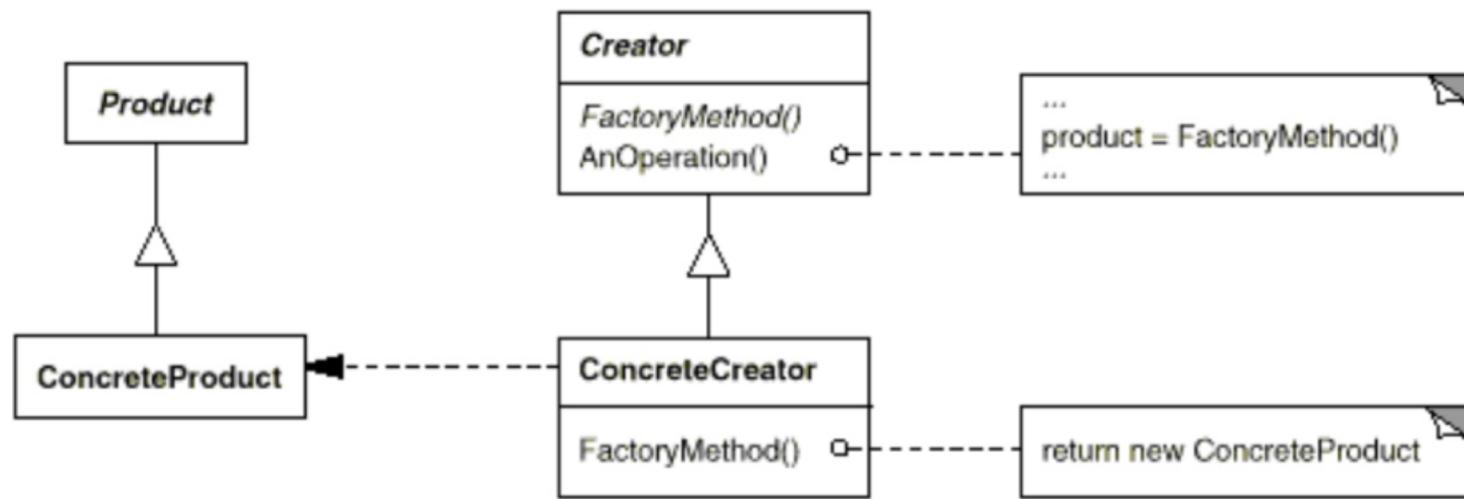
Factory Method

¿Cuándo se utiliza?

- Una clase no puede anticipar las clases de los objetos que debe crear
- Una clase necesita permitir a sus subclases especificar la creación de objetos
- Una clase delega la responsabilidad de crear objetos a sus subclases, para así encapsular lógica

Patrones creacionales

Factory Method



Ejemplo

Patrones de comportamiento

Observer

Define una relación de dependencia entre uno y N objetos, para que cuando ese objeto cambie su estado, todos los N sean notificados y actualicen automáticamente.

Patrones de comportamiento

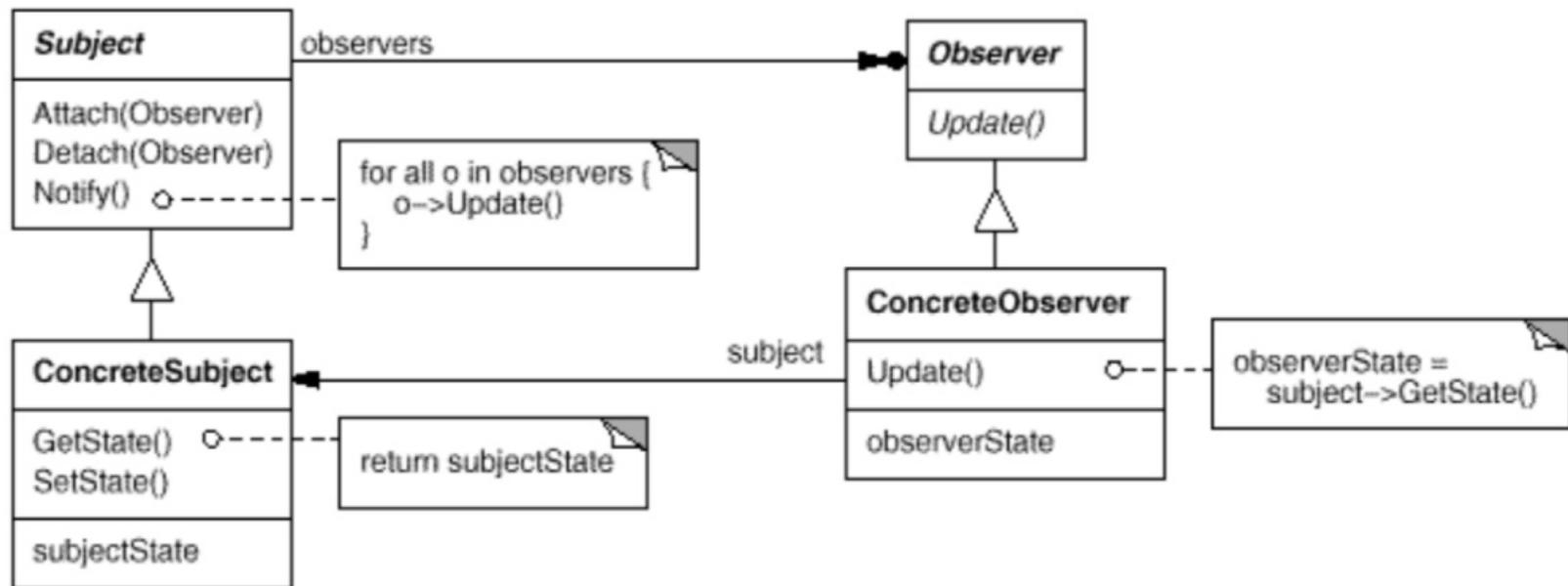
Observer

¿Cuándo se utiliza?

- Cuando una abstracción tiene dos aspectos, uno dependiente del otro. Encapsular estos aspectos en objetos separados permite variarlos y reutilizarlos independientemente.
- Cuando un cambio en un objeto genera cambios en otros, y no se sabe cuántos objetos deben cambiar.
- Cuando un objeto debería ser capaz de notificar a otros objetos sin realizar supuestos de quiénes son estos objetos. En otras palabras, se quiere reducir el acoplamiento entre objetos dependientes.

Patrones de comportamiento

Observer



Ejemplo

Patrones de comportamiento

Strategy

Define una familia de algoritmos, encapsula cada uno, y los hace intercambiables. Strategy permite al algoritmo cambiar independientemente de los clientes que lo utilizan.

Patrones de comportamiento

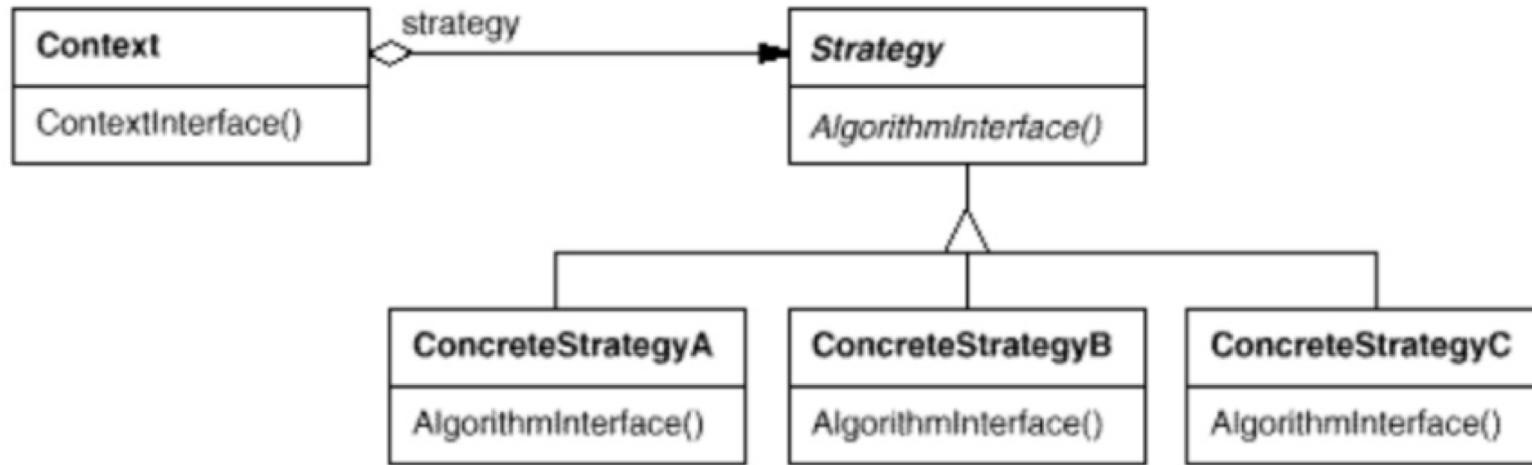
Strategy

¿Cuándo se utiliza?

- Varias clases relacionadas se diferencian solamente por su comportamiento. *Strategy* permite cambiar la configuración de una clase a través de la definición de su comportamiento.
- Se necesita diferenciar las variantes de un algoritmo.
- Para ocultar estructuras complejas utilizadas solamente por un algoritmo.
- Una clase define muchos comportamientos, que aparecen como condicionales en sus operaciones. En vez de reutilizar condicionales, se pueden encapsular alternativas bajo una clase *Strategy*.

Patrones de comportamiento

Strategy



Ejemplo

Patrones de comportamiento

Template Method

Define la estructura de un algoritmo, delegando algunos pasos a sub-clases. *Template Method* permite que las sub-clases redefinan algunos pasos del algoritmo, sin cambiar la estructura de este.

Patrones de comportamiento

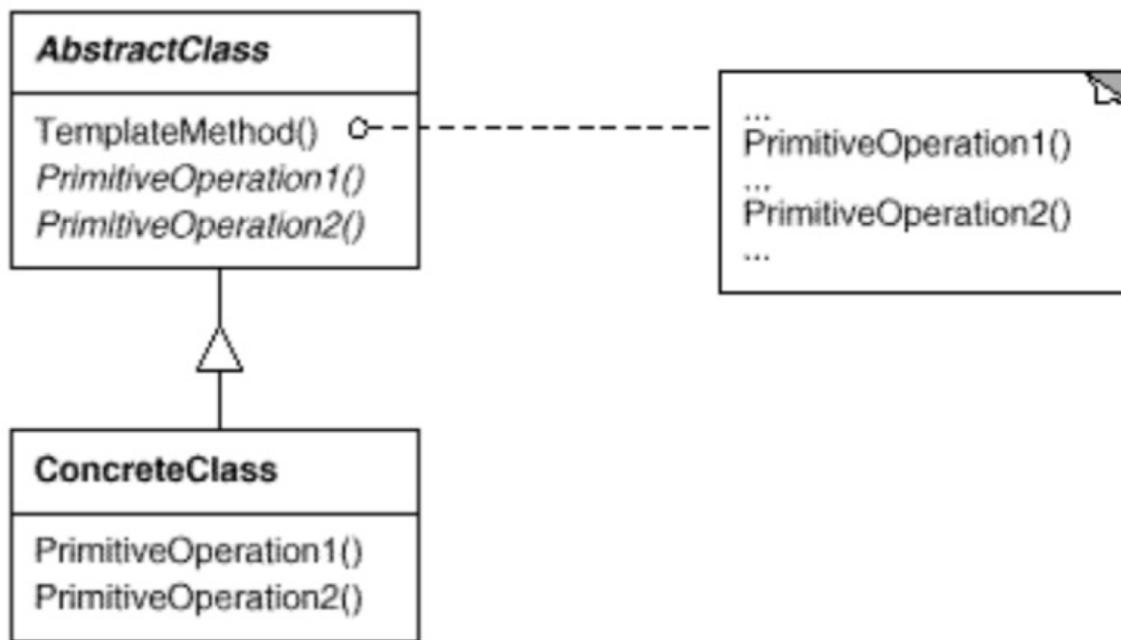
Template Method

¿Cuándo se utiliza?

- Para implementar una sola vez la parte invariante de un algoritmo, y permitir que algunos pasos cambien.
- Para agrupar comportamiento común, y así evitar código duplicado.

Patrones de comportamiento

Template Method



[Ejemplo](#)

Patrones de comportamiento

Command

Encapsula una solicitud como un objeto, para así permitir la parametrización de clientes con solicitudes distintas, encolar o registrar solicitudes, y soportar operaciones que se pueden deshacer.

Patrones de comportamiento

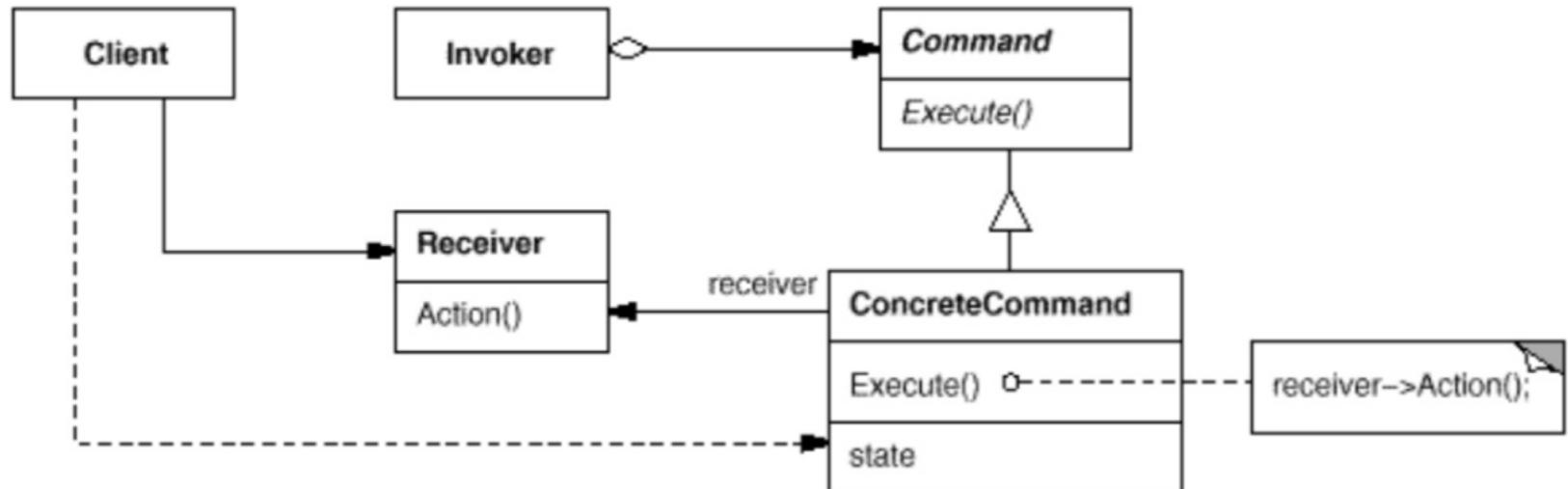
Command

¿Cuándo se utiliza?

- Se necesita parametrizar objetos con una acción a realizar. *Command* es un reemplazo para *callbacks* en OOP.
- Especificar, encolar y ejecutar solicitudes en momentos diferentes.
- Soportar que la solicitud se deshaga.
- Estructurar un sistema alrededor de operaciones complejas, formadas a partir de operaciones primitivas.

Patrones de comportamiento

Command



Ejemplo



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 15

Patrones de diseño

IIC2143 - Ingeniería de Software
Sección 1

Rodrigo Saffie

rasaffie@uc.cl

2 de mayo de 2018