AYUDANTÍA 9

PATRONES DE DISEÑO

Antonia Christensen

¿Qué veremos?

1

Mini repaso

Revisaremos los patrones vistos en clases, viendo para qué sirven.

2

Ejercicios Ies Pasadas

Revisaremos un par de ejercicios de ies pasadas que involucran escribir código de patrones.

3

Preguntas

Resolver cualquier duda que les haya quedado.

PATRÓN DE DISEÑO

“La solución de un problema en un contexto dado”. Cristopher Alexander

Lograr soluciones que **minimicen el acoplamiento** manteniendo la **cohesión**. Así el software puede ser **extendido y modificado con facilidad**.

3 categorías de patrones

CREACIONALES

1. Fábrica
2. Fábrica Abstracta
3. Builder
4. Singleton

COMPORTAMIENTO

1. Observer
2. Plantilla
3. Estrategia
4. Comando

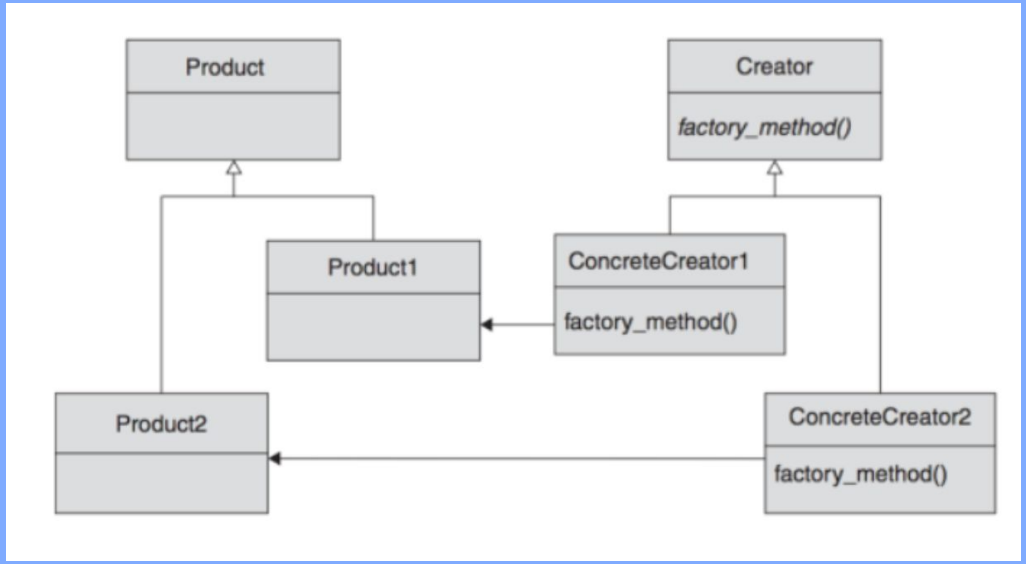
ESTRUCTURALES

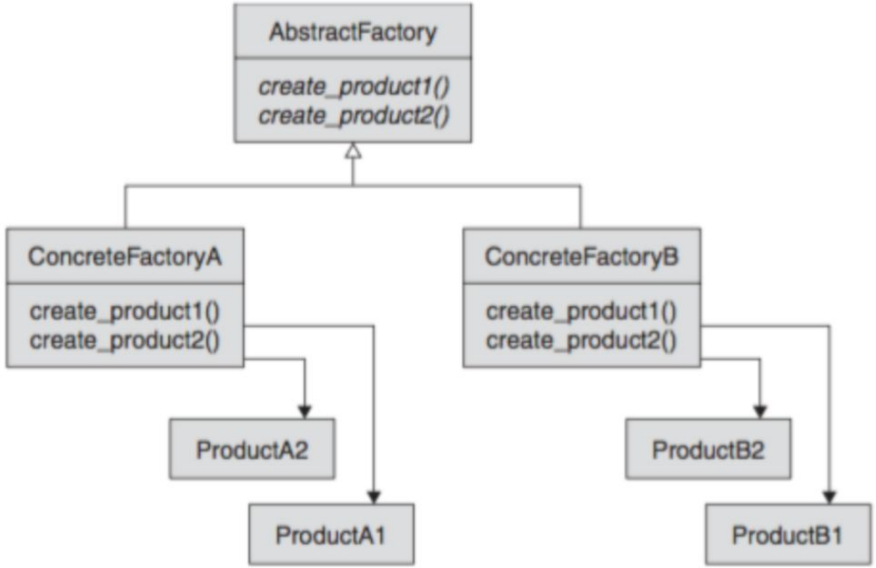
1. Adaptador
2. Fachada
3. Decorador
4. Composite

Fábrica

Permite **crear una serie de objetos**, indicando el **tipo** y la **cantidad**.

Es un template aplicado a la creación de objetos.



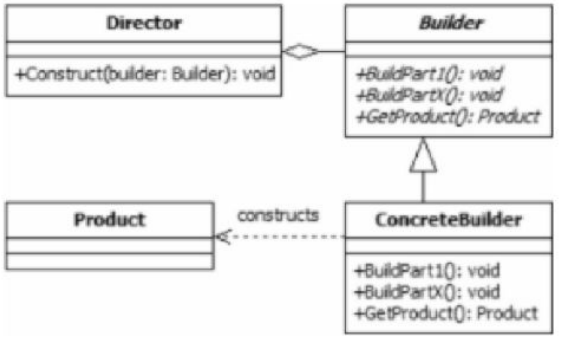


Fábrica Abstracta

Fábrica abstracta: define los métodos para fabricar cada uno de los productos .

Fábricas concretas: cada una es capaz de crear su propio set de productos, implementando los métodos de la fábrica abstracta.

Builder

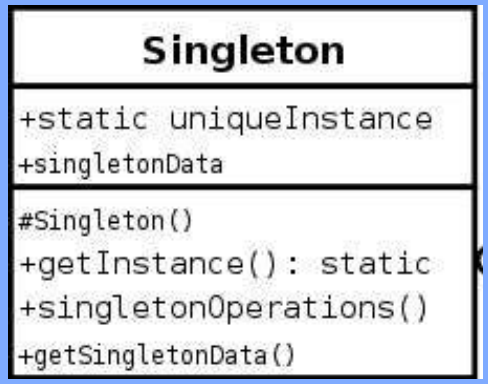


Busca crear **objetos complejos.**
Los **atributos pueden ser otros objetos** que hay que construir.
La clase builder provee métodos para crear el objeto y agregarle los elementos necesarios. **Distintas variaciones, distintos builders.**
A veces: **objeto director** --> dirige construcción del builder.

Singleton

Solo puede existir **una instancia** de una clase en particular.

Se busca garantizar **acceso global** a esa instancia.

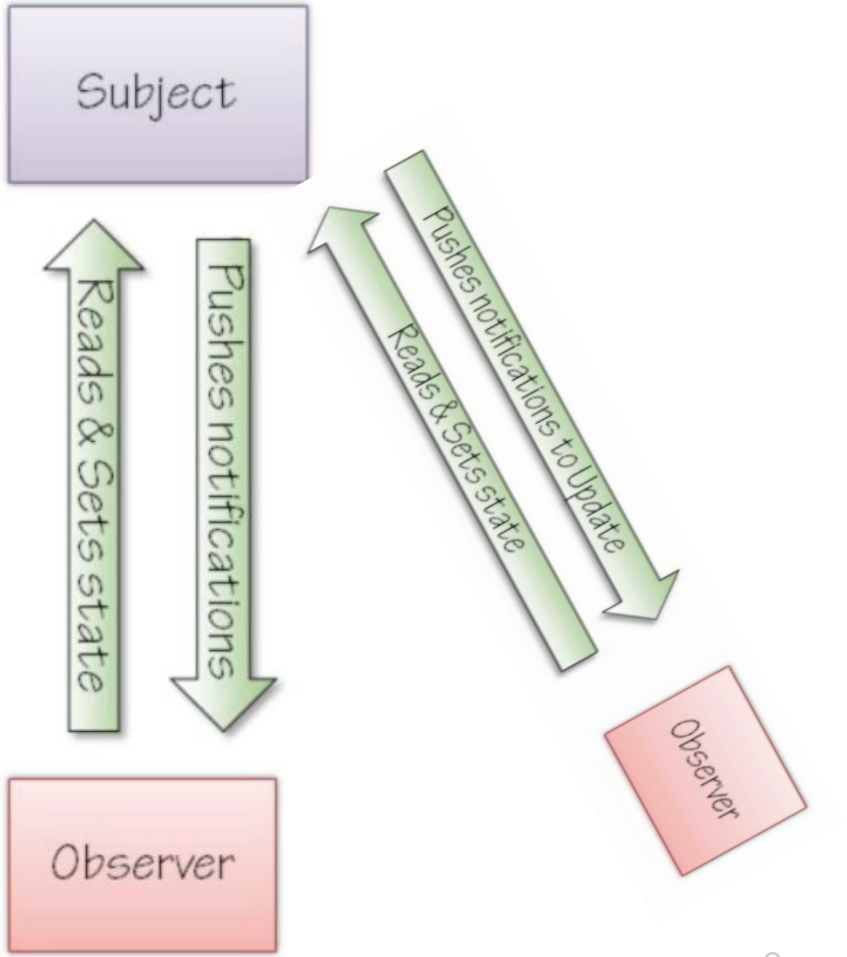



```
if (!uniqueInstance) {
  uniqueInstance = new Singleton();
}
return uniqueInstance;
```

Observer

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

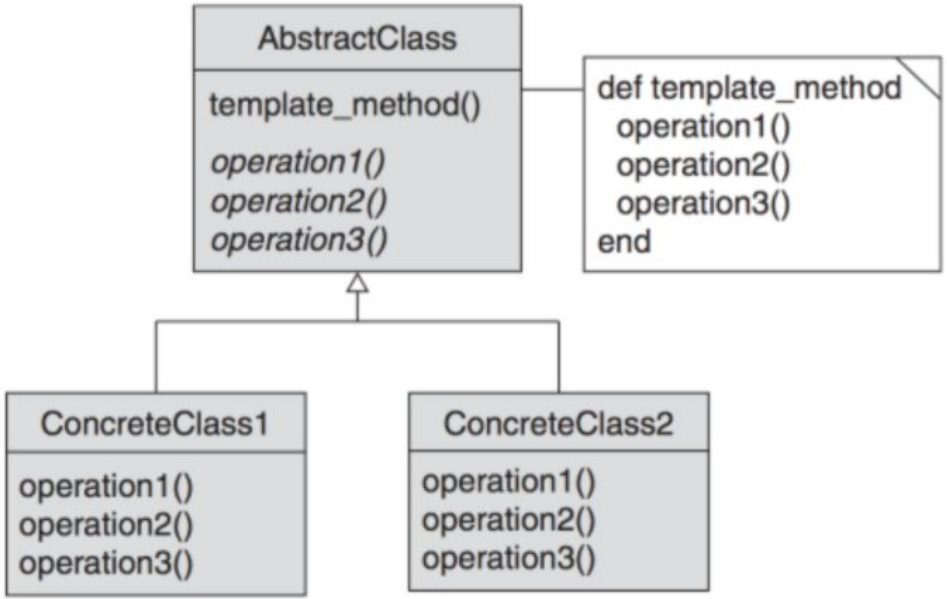
Relación **1:N.**



Plantilla

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

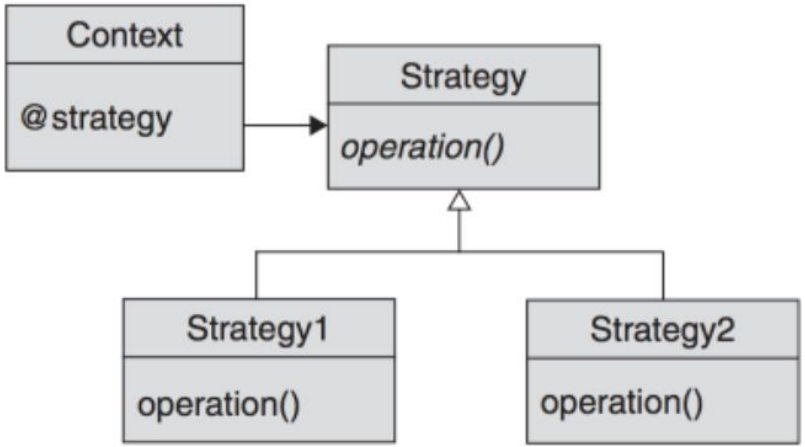
Objetivo: encapsular secuencia de operaciones cómo un **método con operaciones abstractas**.



Estrategia

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

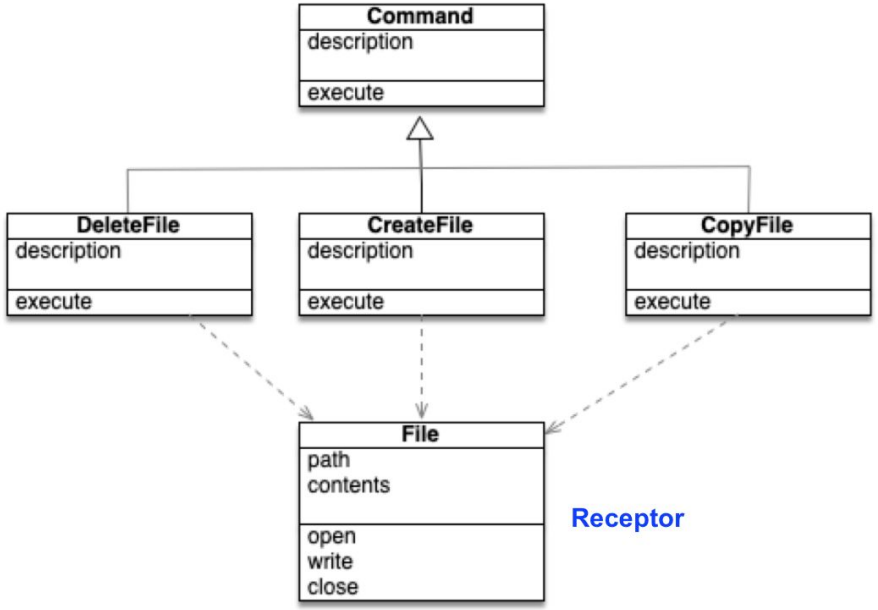
Permite definir una familia de algoritmos, poniéndolos en **clases separadas** y haciendo sus o**bjetos intercambiables.**



Comando

Representa acciones como objetos.

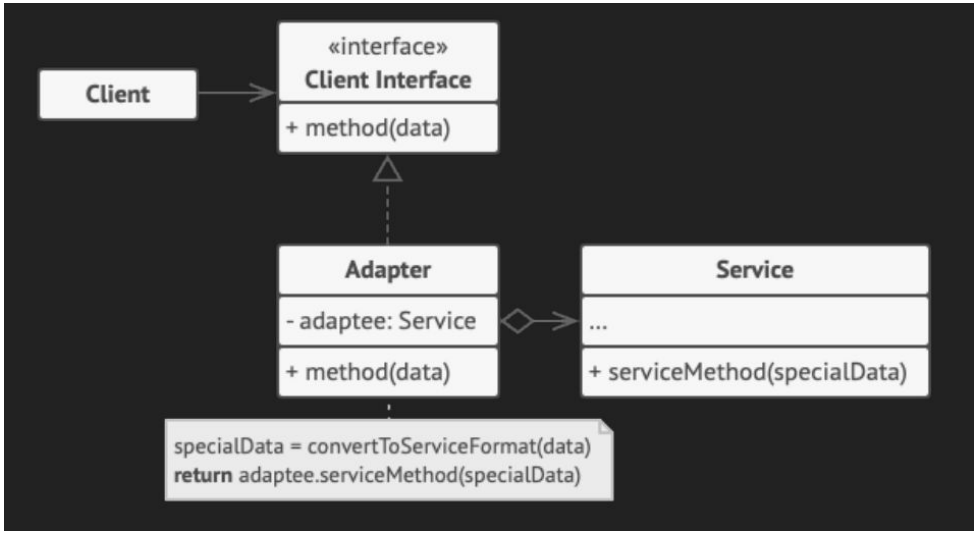
Permite ejecución no inmediata (colas).



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.

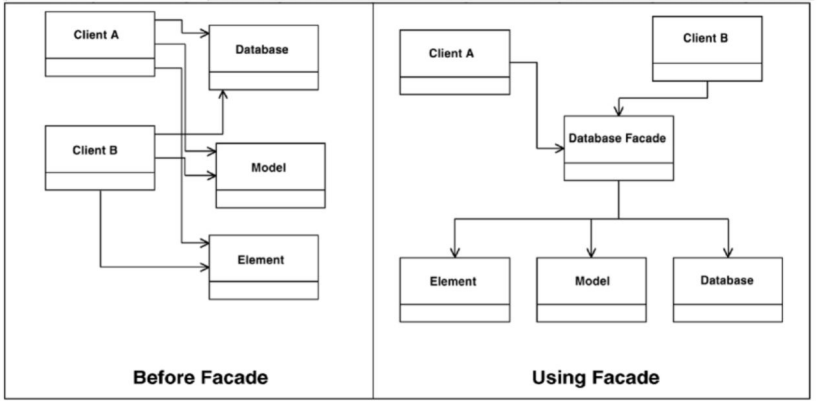


Fachada

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.

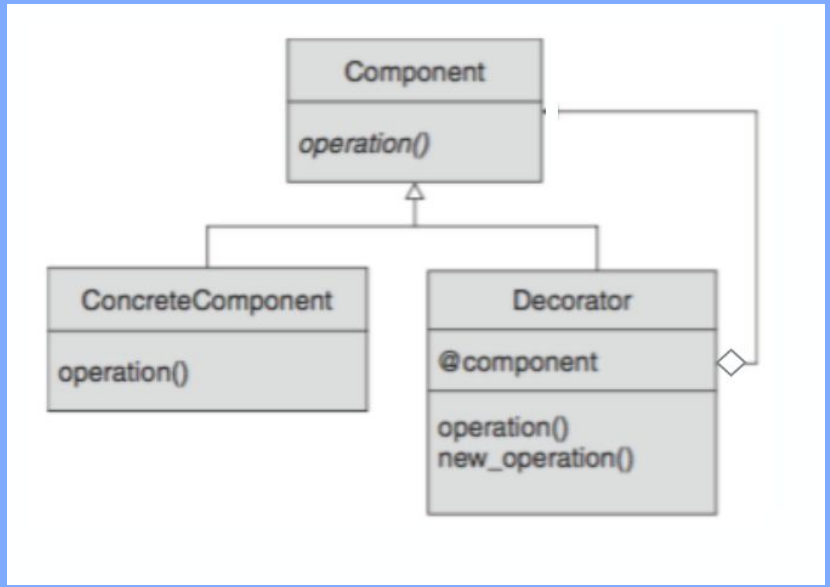


Decorador

Anexar responsabilidades en forma **dinámica** a un objeto.

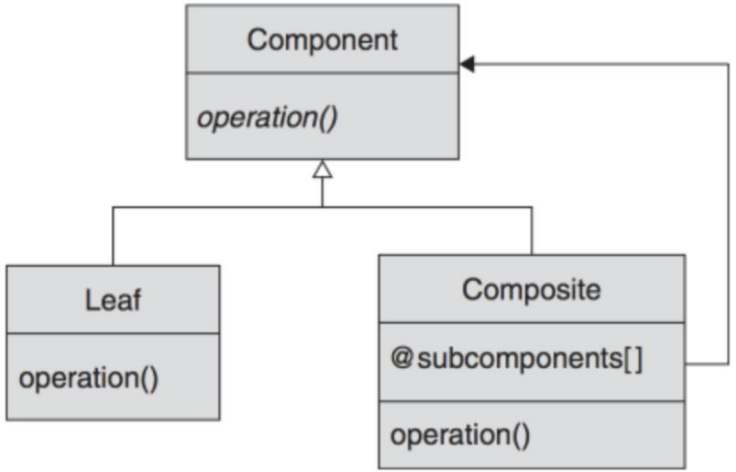
Permite **añadir comportamientos** a objetos. Poniendo estos en un objeto wrapper **en runtime.**

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



Composite

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



EJERCICIOS

I2 2019-2

I3 2019-1

I2 2020-1

¿Preguntas?

AYUDANTÍA 9

PATRONES DE DISEÑO

Antonia Christensen