

Patrones de diseño I

UNRN

Universidad Nacional
de Río Negro

XVI
I
2025



**Algo que me
faltó por el
camino...**

Documentación

Clases

```
/**  
 * Representa a una Persona según como este en su documento  
 * de identificación nacional.  
 */  
public class Persona {
```

**Empezar con un verbo en tercera persona del singular
(x ej. "Representa...", "Proporciona...", "Implementa...").**

Genéricas

```
/**  
 * Representa un conjunto ordenado de elementos de  
 * tipo E similar a un arreglo.  
 * </p>  
 * Una descripción mas detallada.  
 * @param E es el tipo genérico con el que trabajará  
 */  
public class Secuencia<E>
```

Atributos

```
public class Persona {  
    /// Este es el nombre como se encuentra en la identificación  
    private String nombre;  
    /**  
     * Es el número como está en la identificación nacional  
     * Con el código de país al principio; 'AR99.999.999'  
     * No puede ser nulo.  
     */  
    private String identificacion;
```

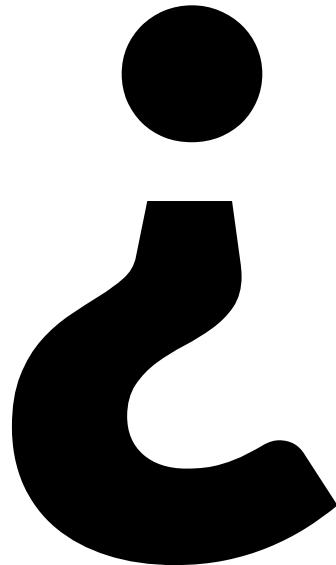
**Una descripción
breve de una línea
es más que
suficiente**

Referencias cruzadas

```
/**  
 * Agrega valor en la posición indicada aumentando el  
 * tamaño de la secuencia.  
 * @throws PosicionInvalidaException  
 * @see #verificar(int)  
 */  
void insertar(int posicion, T valor)
```

Como método de la misma clase

Patrones de diseño



¿Qué son?



Vocabulario

1

Condensan un diseño para una situación específica en una palabra, *ayudan a la documentación*

Experiencia

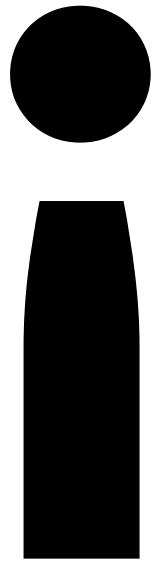
2

Son soluciones probadas a situaciones comunes en el diseño de programas.

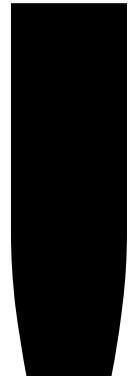
No son reglas estrictas

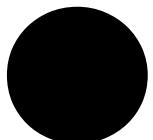
3

Como son conceptos y experiencia,
podemos tomar adaptarlos a nuestras
necesidades



**No son exclusivos
de Orientación a
Objetos**

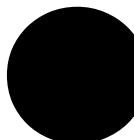




Cuando utilizarlos

**Es importante identificar
cuál es el problema a
*patronizar***

Cuando un problema se parece a un patrón



Cuando NO utilizarlos

Donde no hay problemas

Ojo con aplicar el criterio
“*papita a todo*”
Pablito dixit



**Donde el
problema es
simple
para no enroscarnos solos**

Un poco de historia

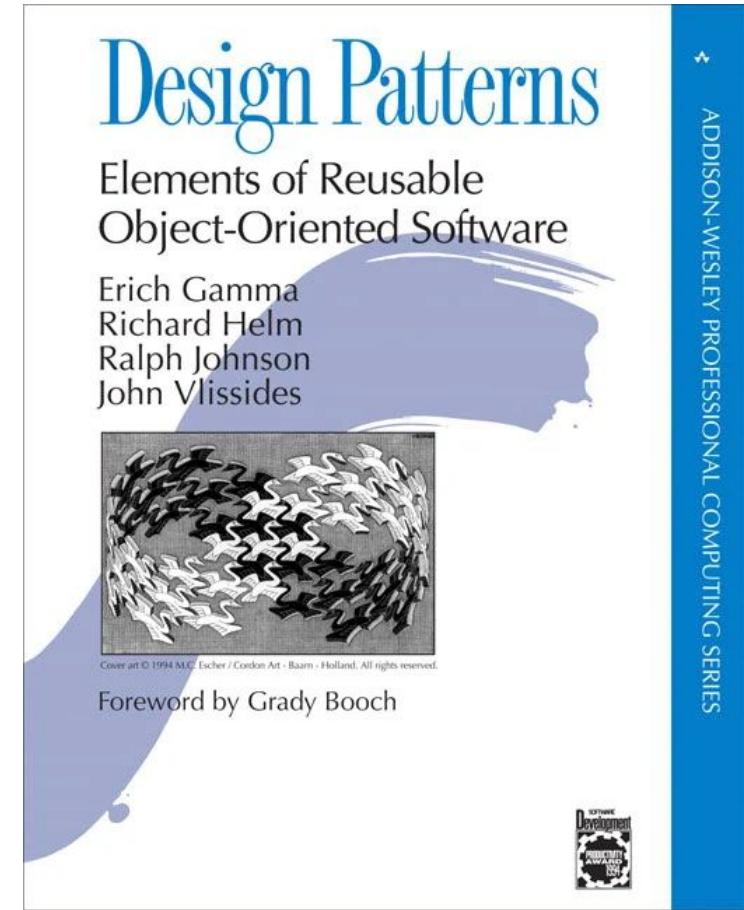


La banda de los cuatro (GoF)

UNRN

Universidad Nacional
de Río Negro

1995



Ahora disponible en el repositorio ;-)

UNRN

Universidad Nacional
de Río Negro

Desarrolla los conceptos de

Una lectura muy recomendada
(pero no utiliza Java)

- La herencia como factor que rompe el encapsulamiento.
- Delegación antes que herencia.
- Agregación y Composición como formas de conexión diferentes.



¿Preguntas?



Tipos de patrones

Creación

Estructural

Comportamiento
O

Creación

C

Se centran en la forma de crear objetos, desacoplando el proceso de creación de la lógica del cliente.

Estructural

S

Se enfocan en cómo componer clases y objetos para formar estructuras más grandes y flexibles.

Comportamiento

B

se enfocan en la interacción y distribución de responsabilidades entre objetos.

Creacionales

- Factory Method
- Abstract Factory
- Builder
- Prototype
- Singleton

Estructurales

- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

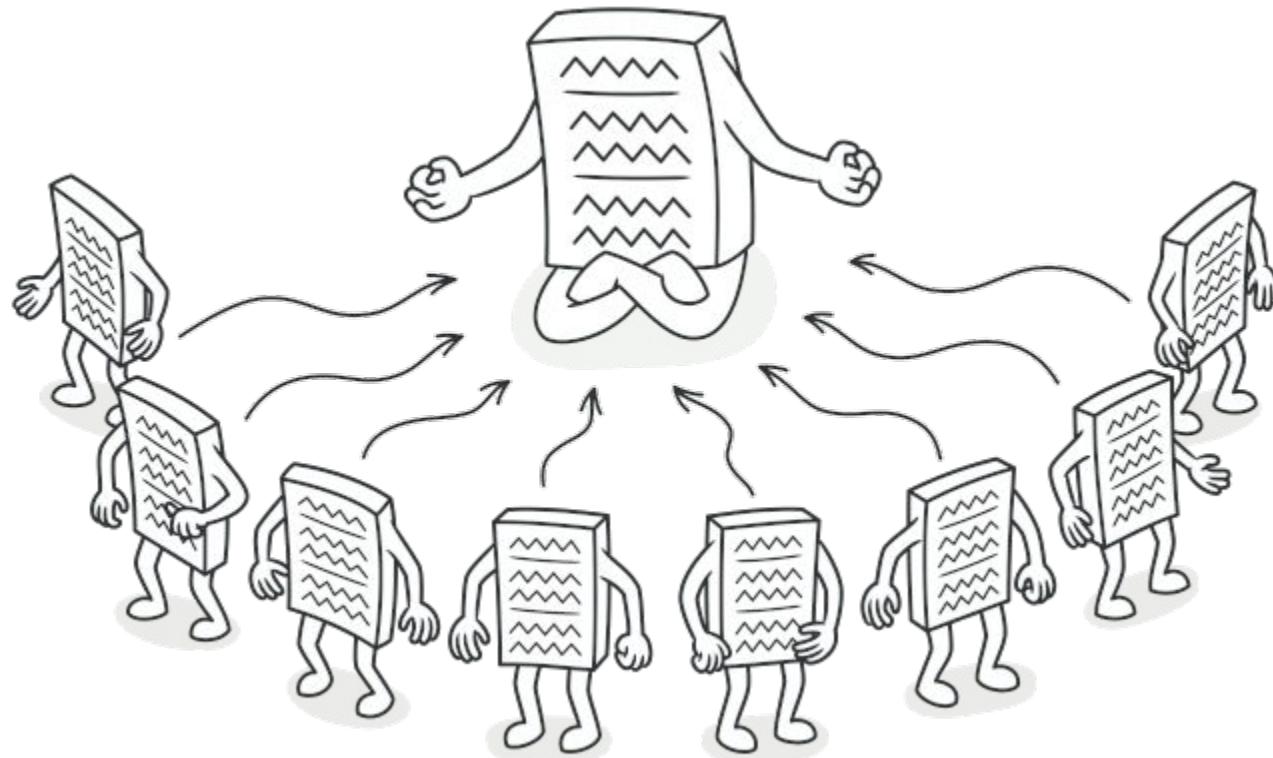
Comportamiento

- Chain of Responsibility
- Command
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor



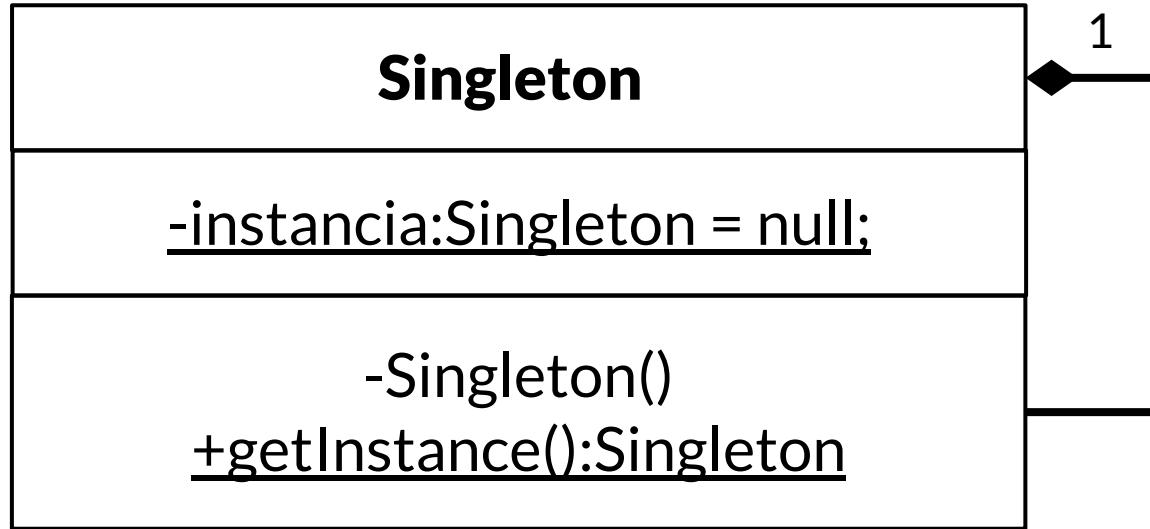
Creacionales

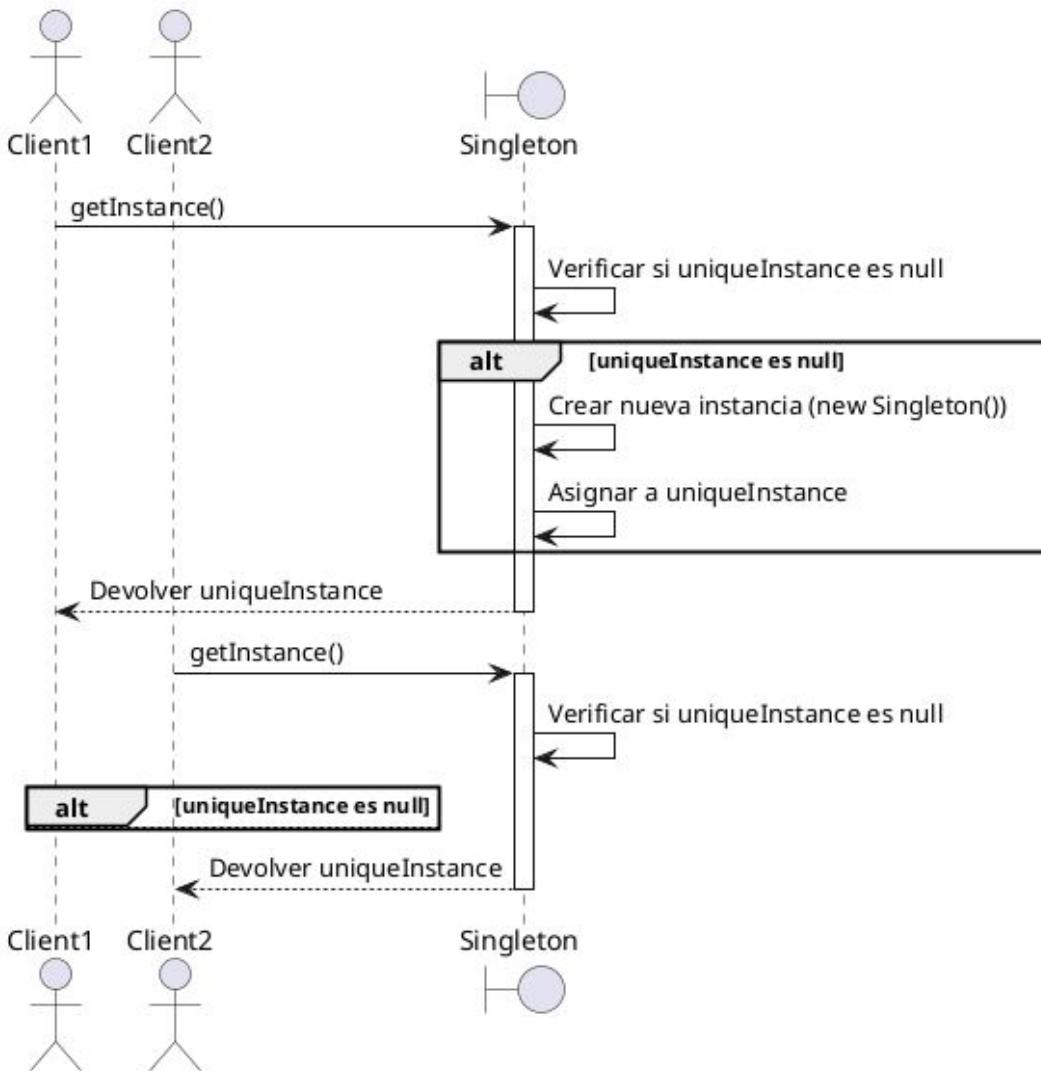
Singleton

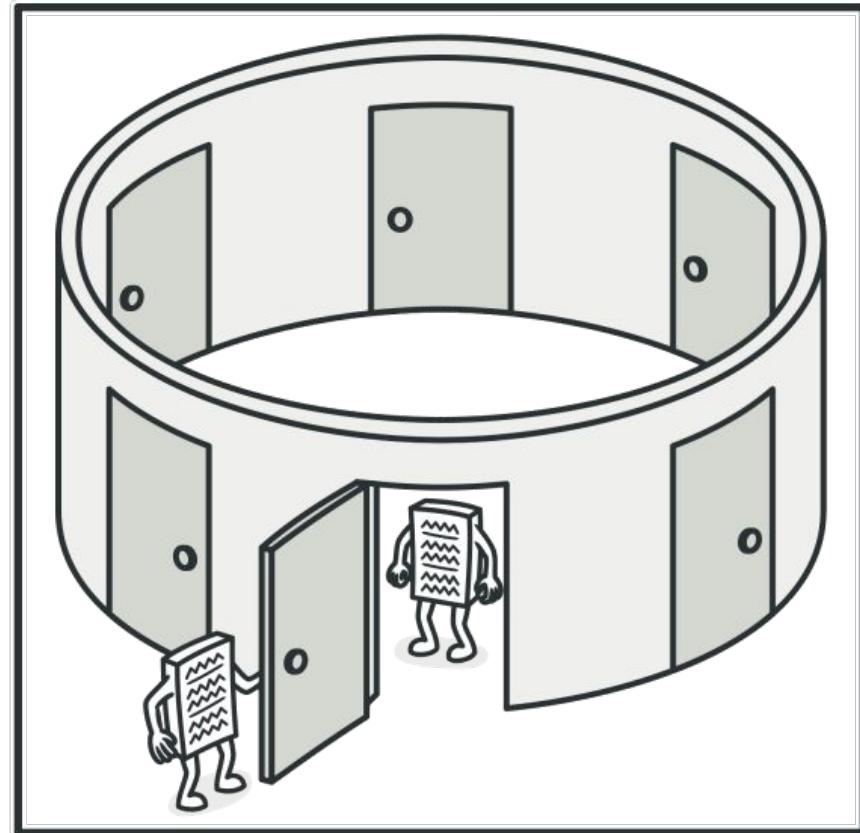


Singleton

Garantizar que una clase sólo tenga una* instancia y da un punto de acceso global a ella.



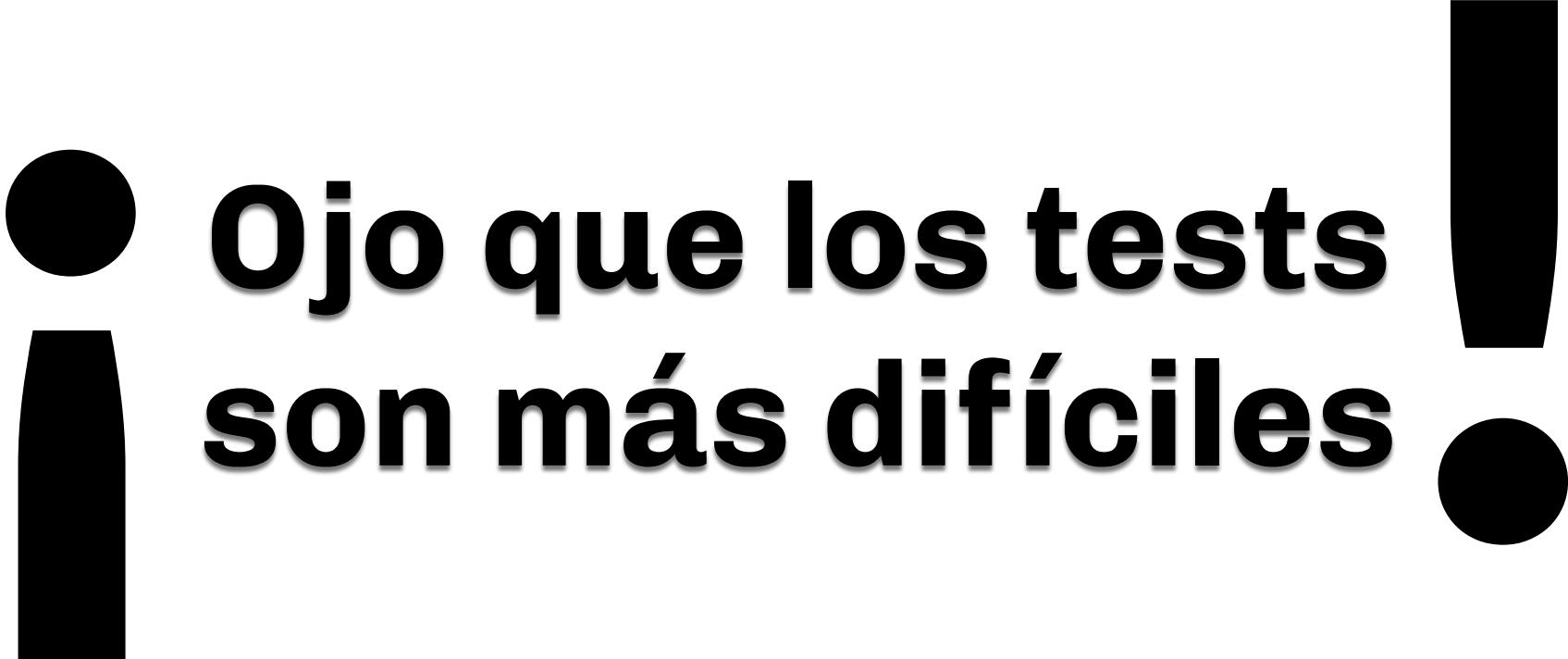




Es trabajar siempre con el mismo :-D

UNRN

Universidad Nacional
de Río Negro



**Ojo que los tests
son más difíciles**

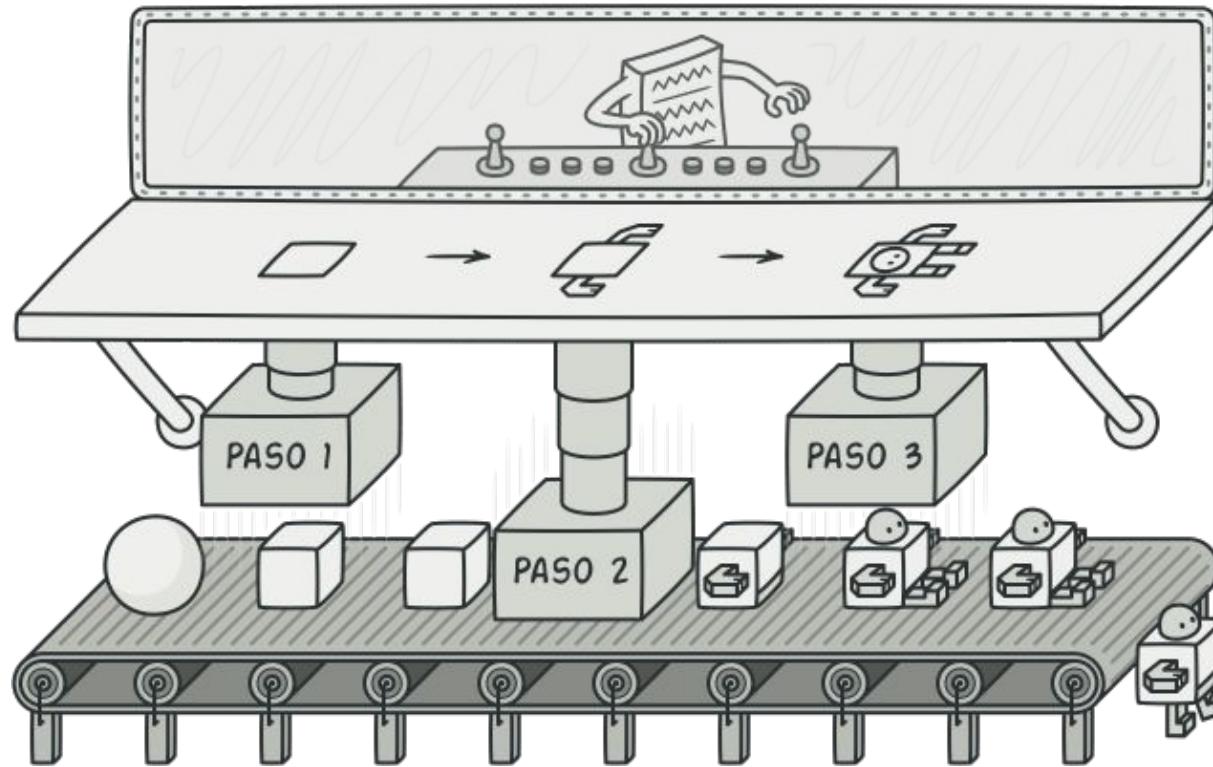
**Termina siendo
funcionando como
una variable
global**

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?

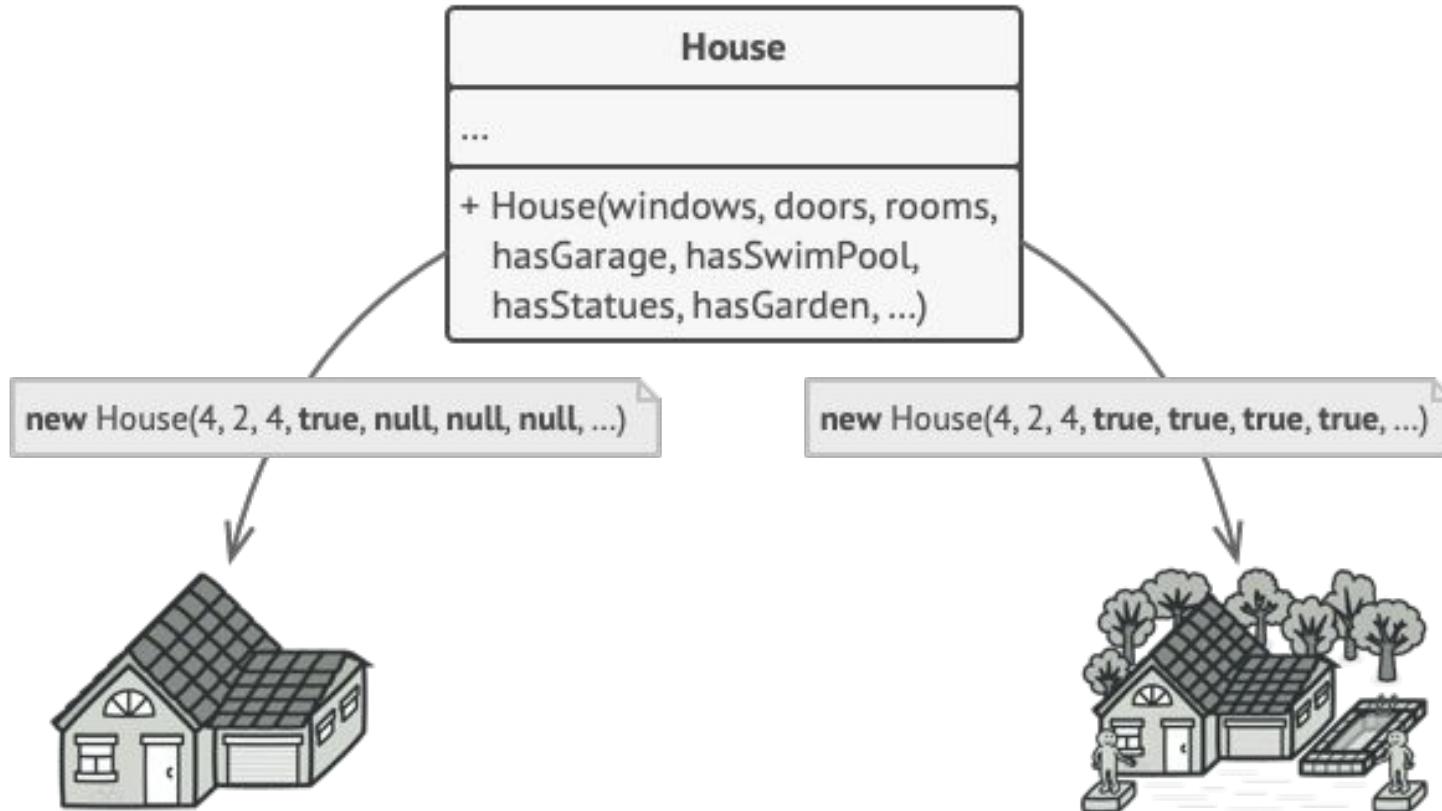


Builder



Builder

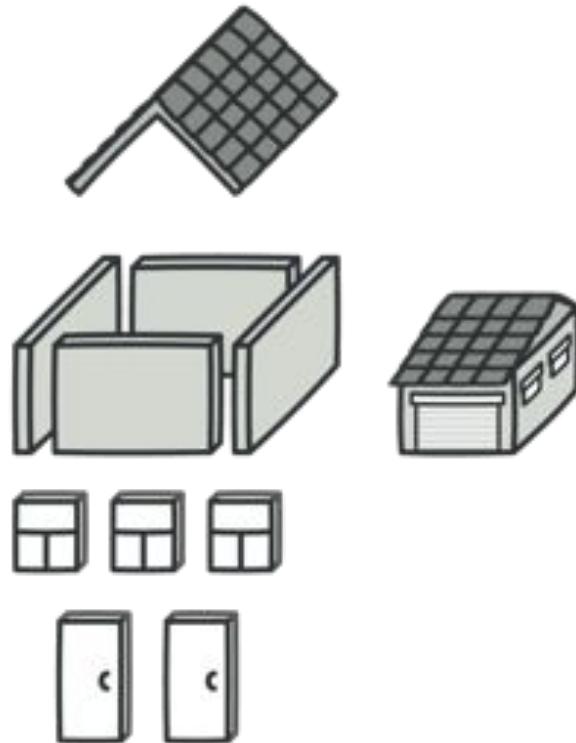
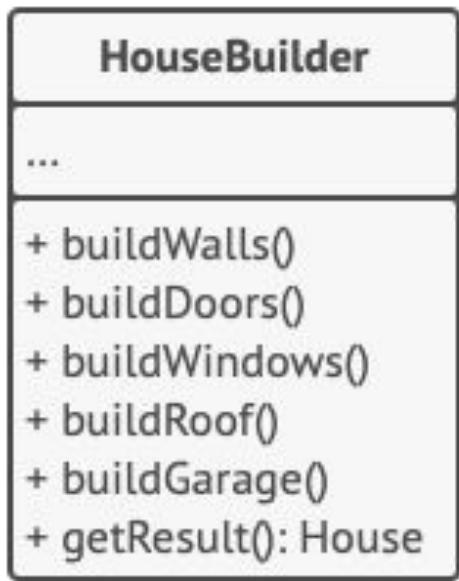
Se puede crear un objeto complejo a partir de una variedad de partes que contribuyen individualmente a su creación.



¿Qué hacemos con el constructor?

UNRN

Universidad Nacional
de Río Negro

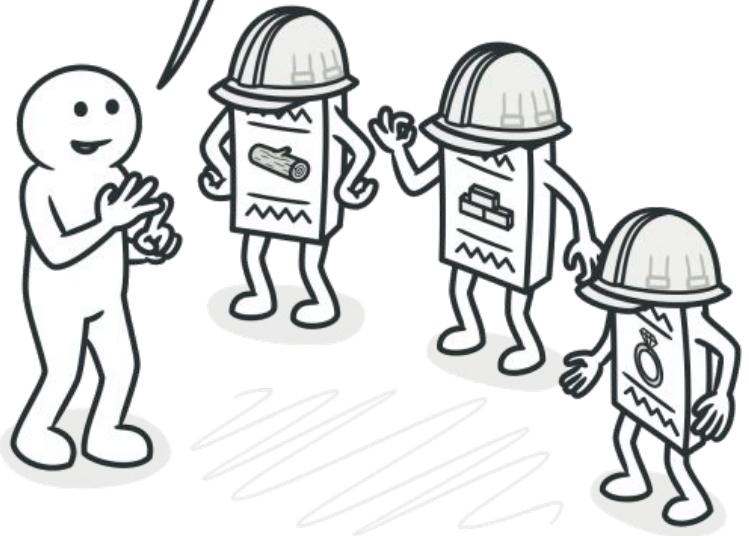


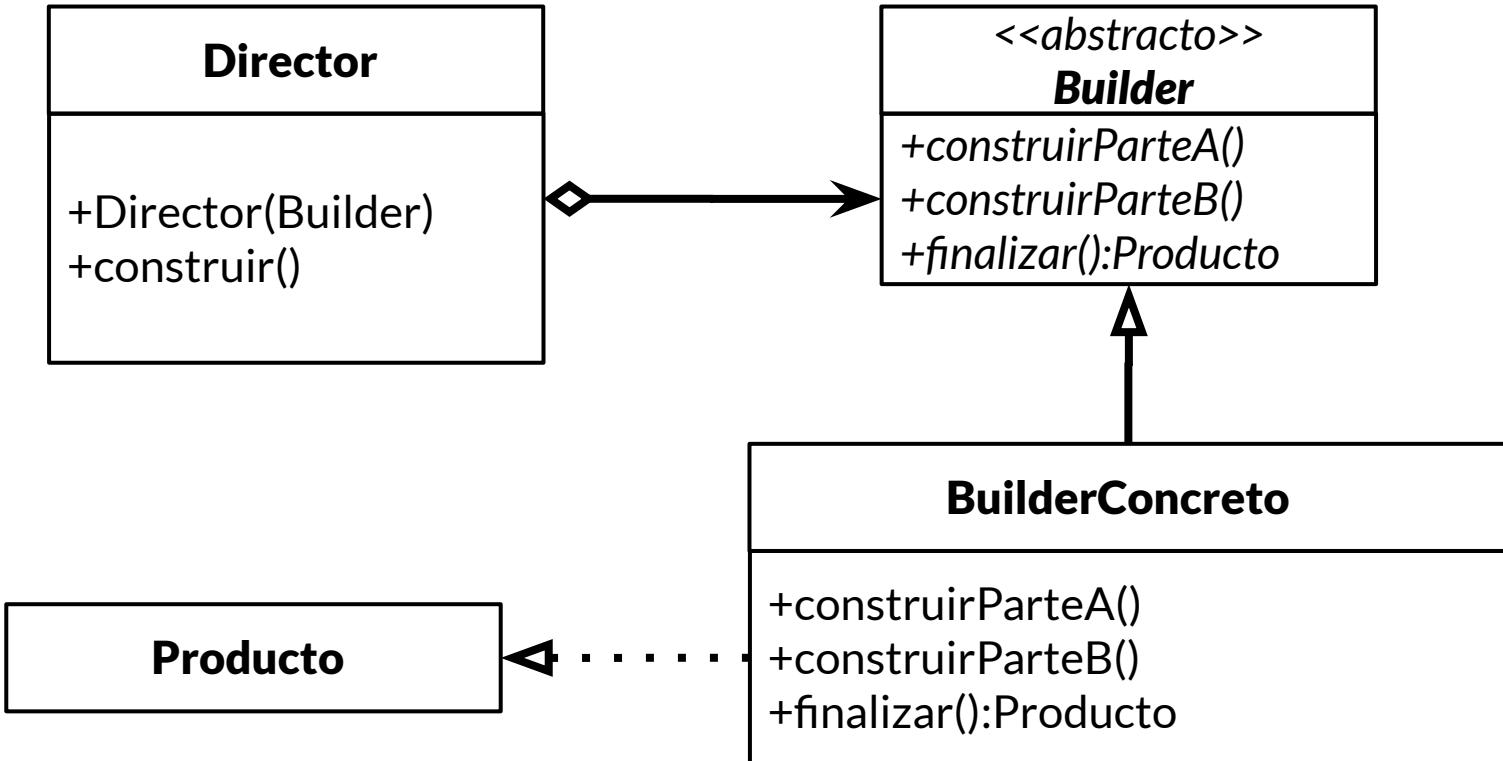
Un Builder, permite una creación compleja y detallada

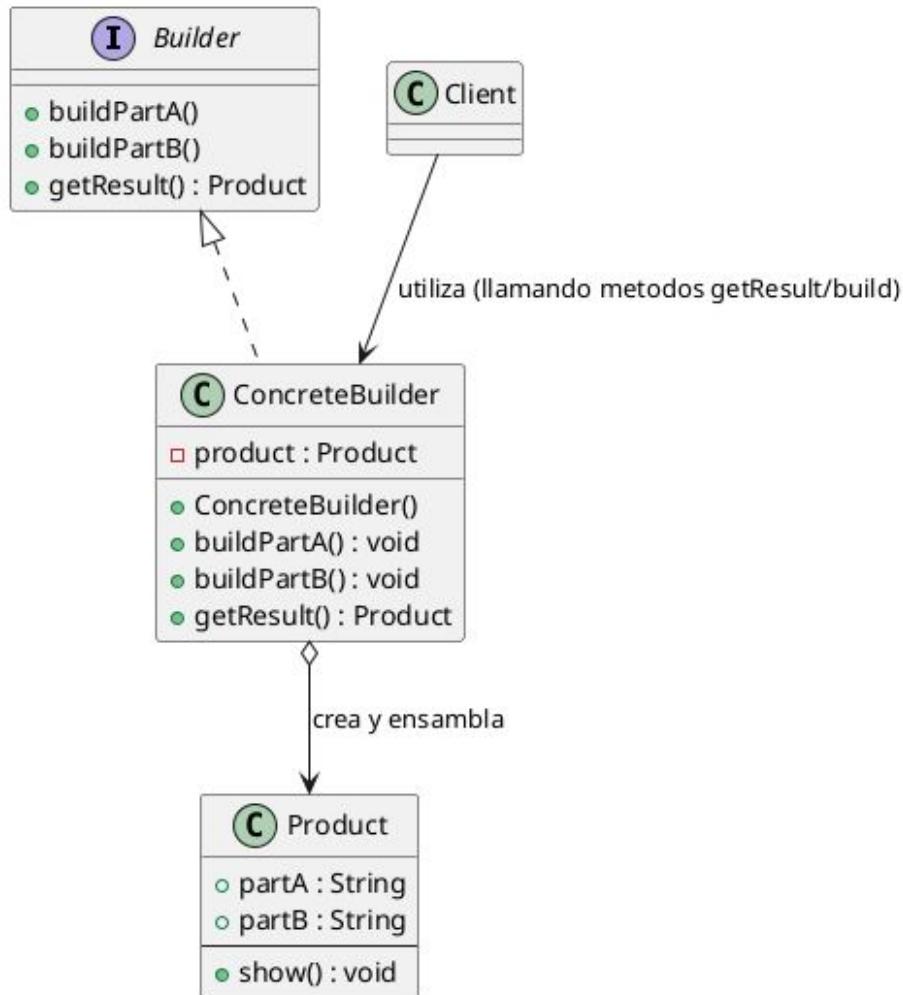
UNRN

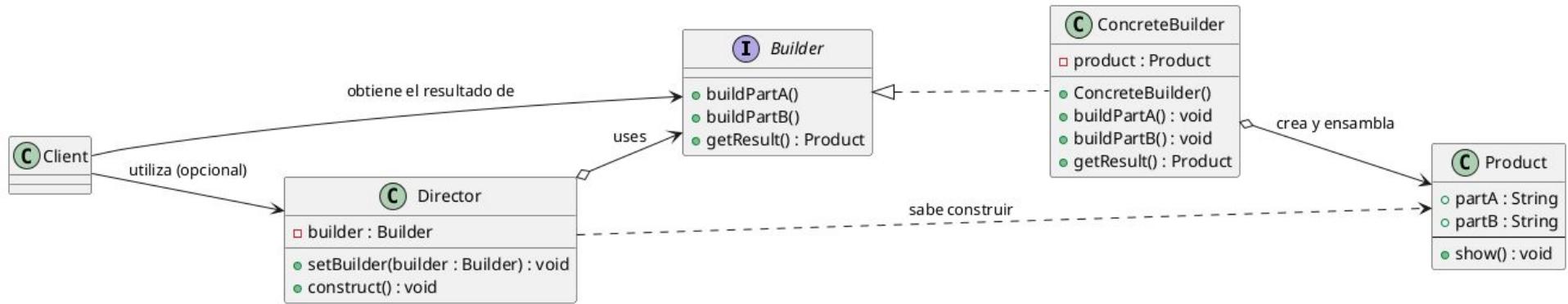
Universidad Nacional
de Río Negro

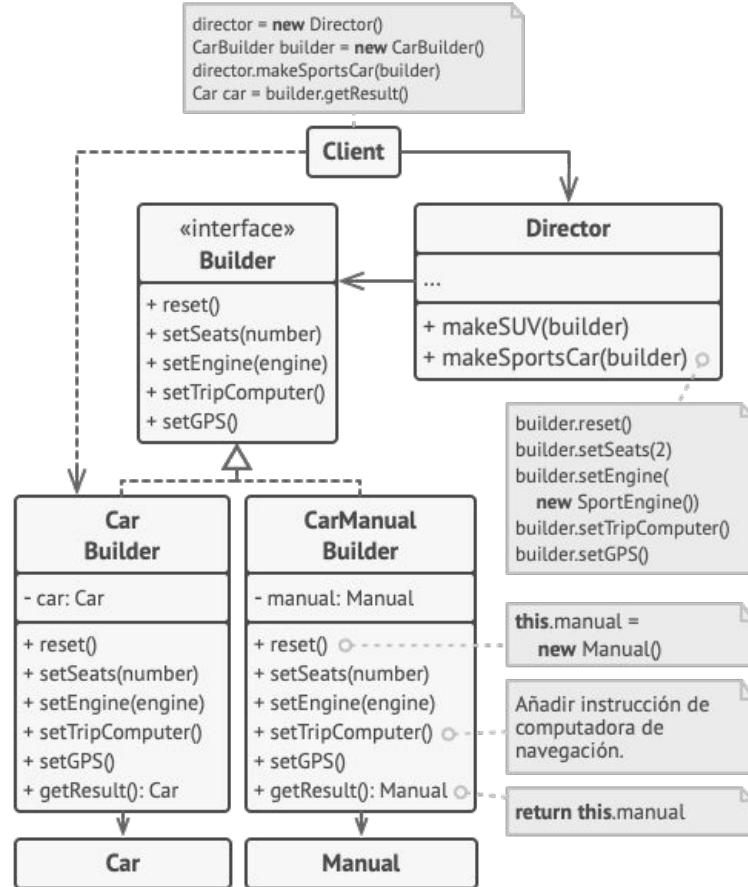
CHICOS, NECESITO UNA CASA CON
4 PAREDES, 1 PUERTA, 2 VENTANAS, ...



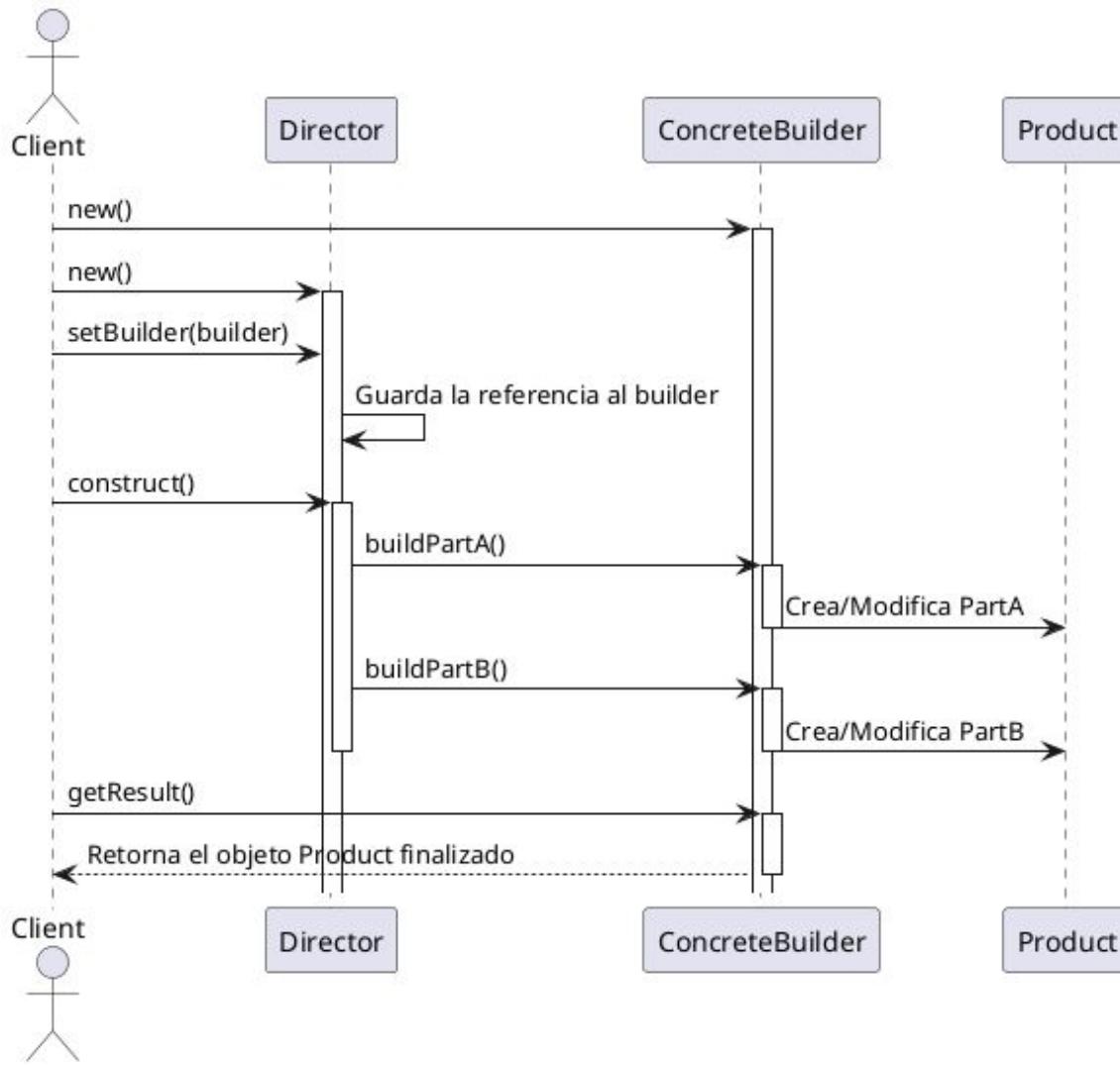








Construcción de dos aspectos de autos



Se separan las responsabilidades

**Los
componentes
pueden cambiar**

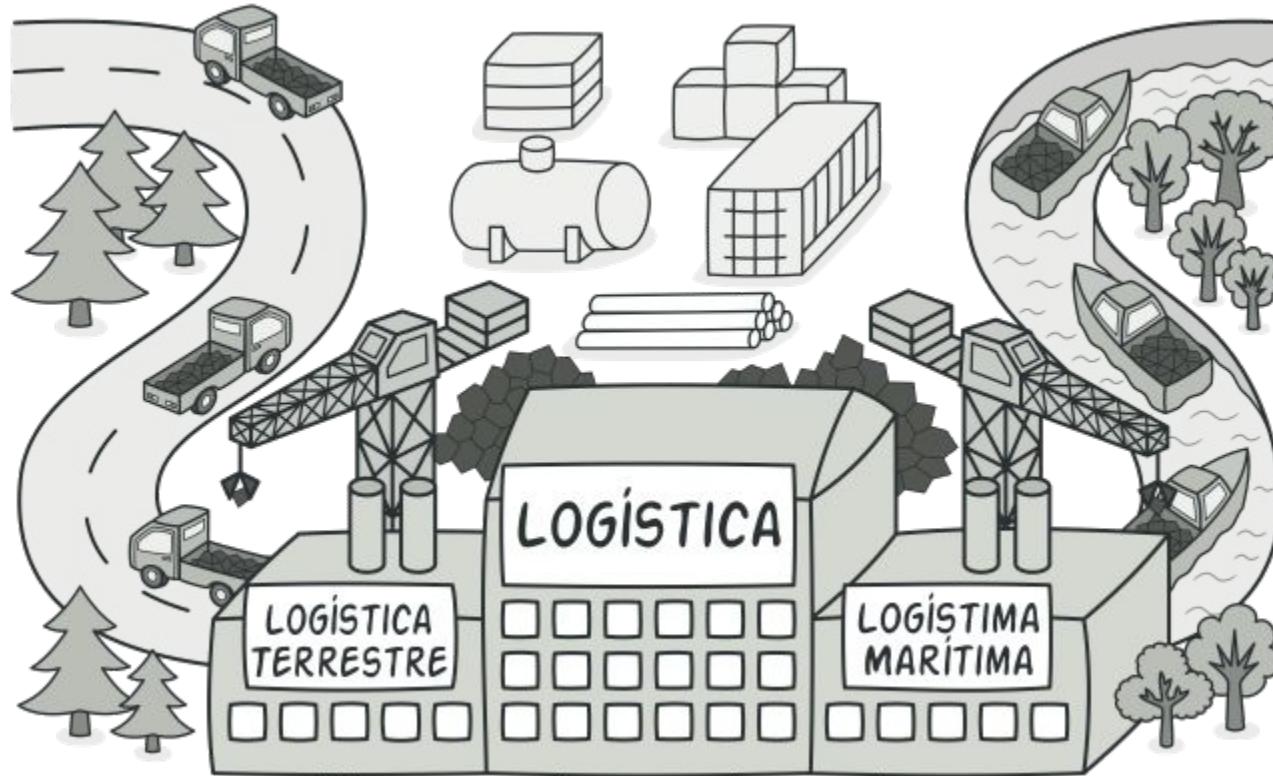
Y la
construcción
también

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?

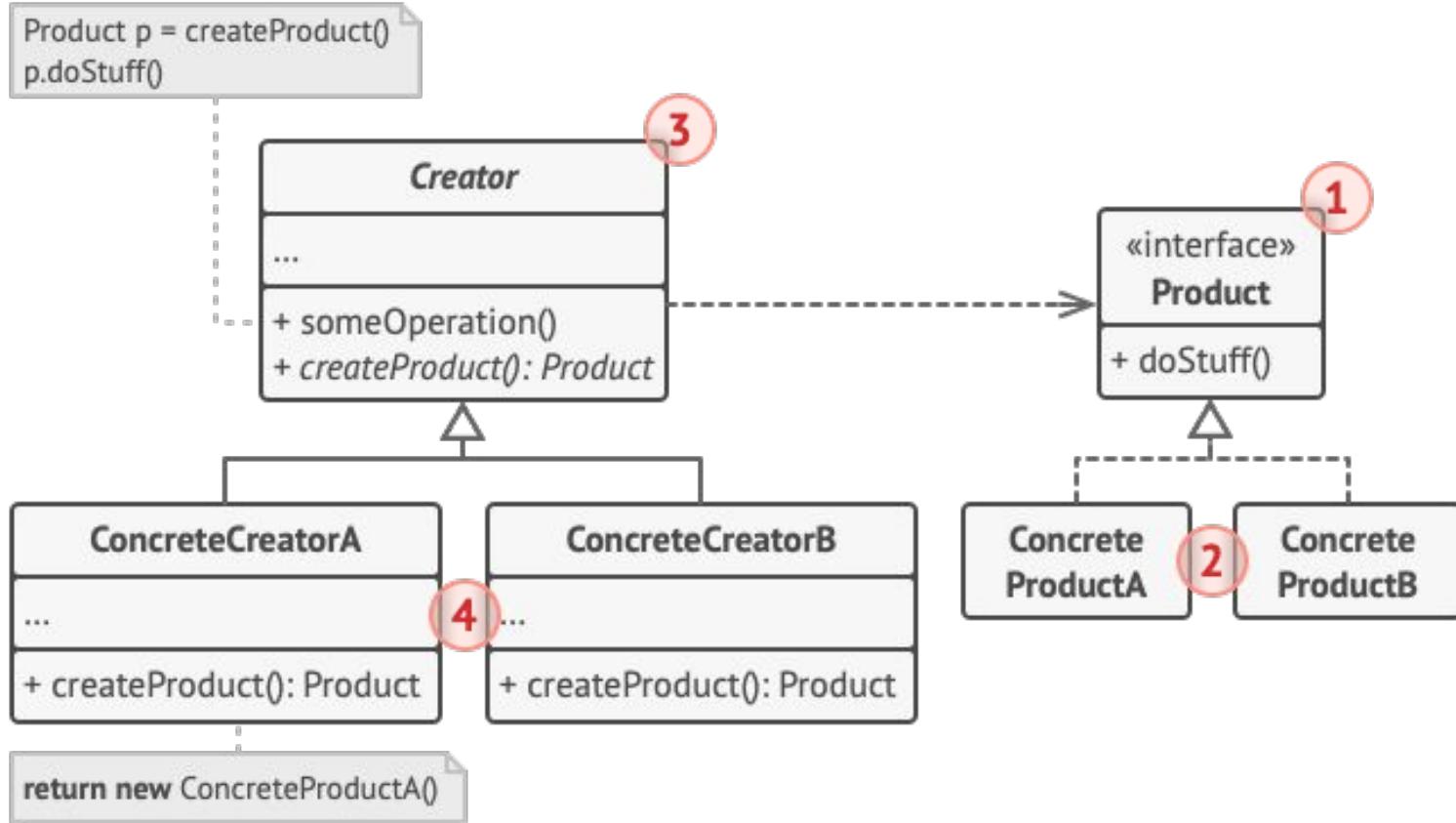


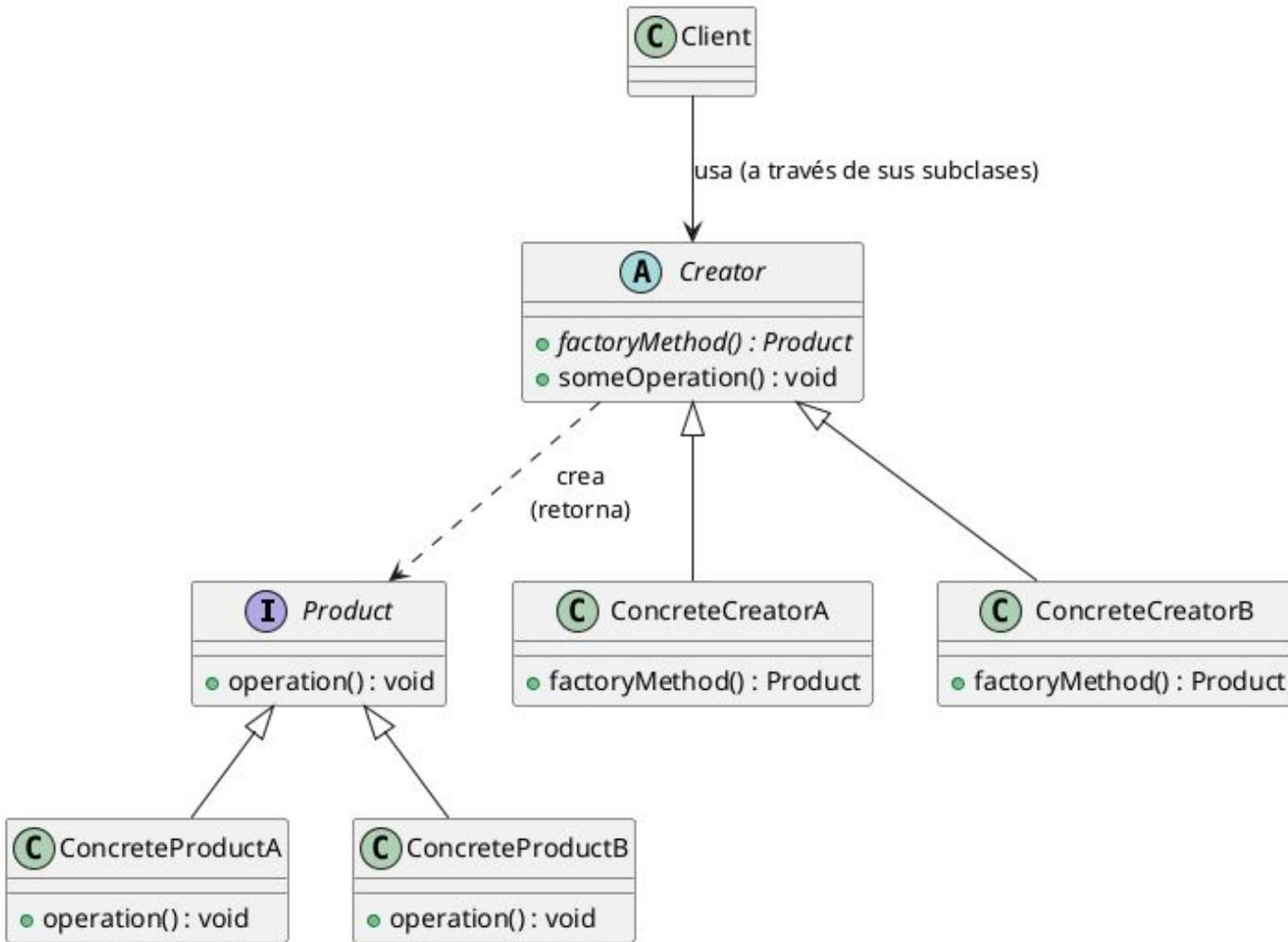
Factory

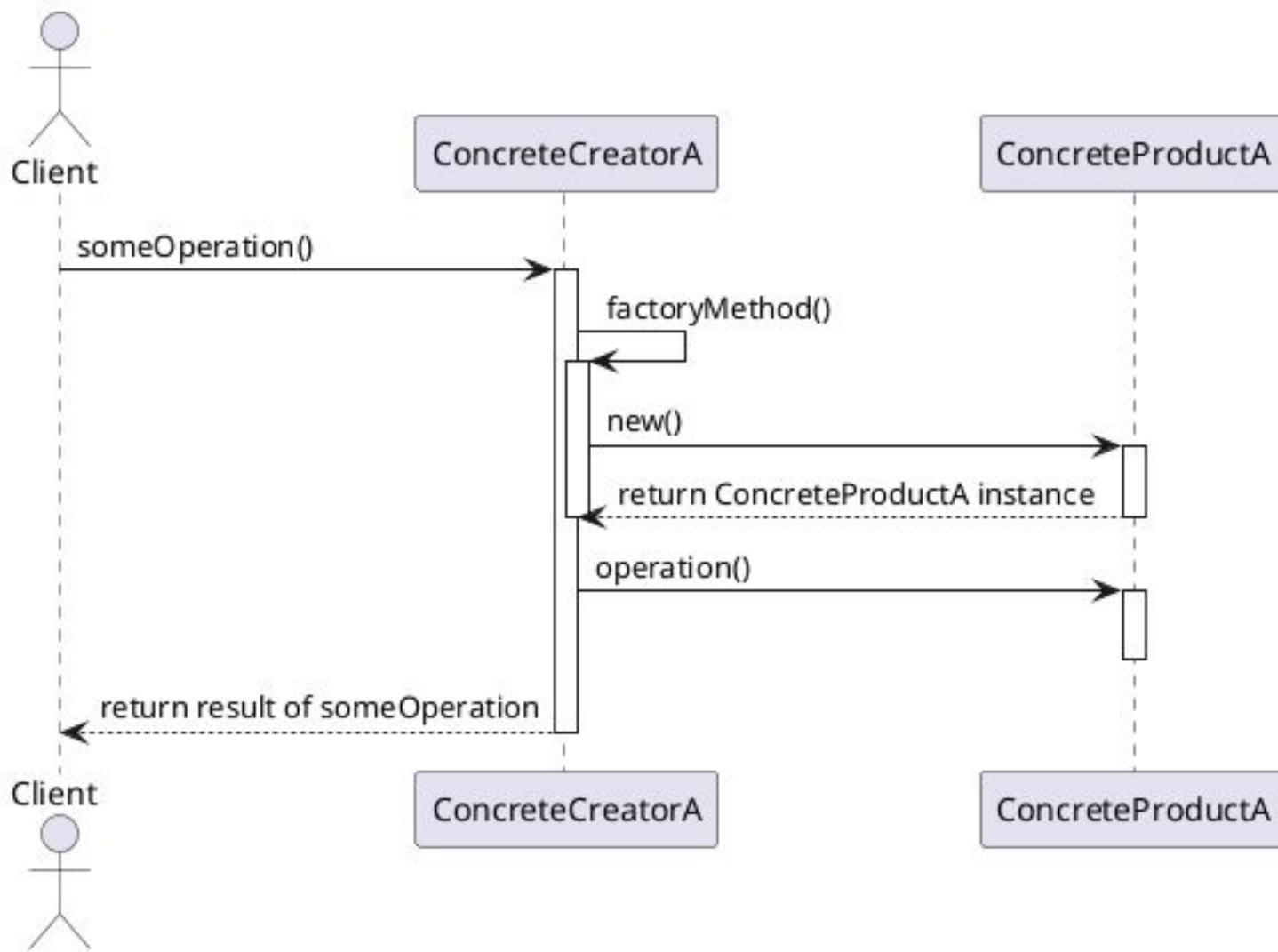


Factory

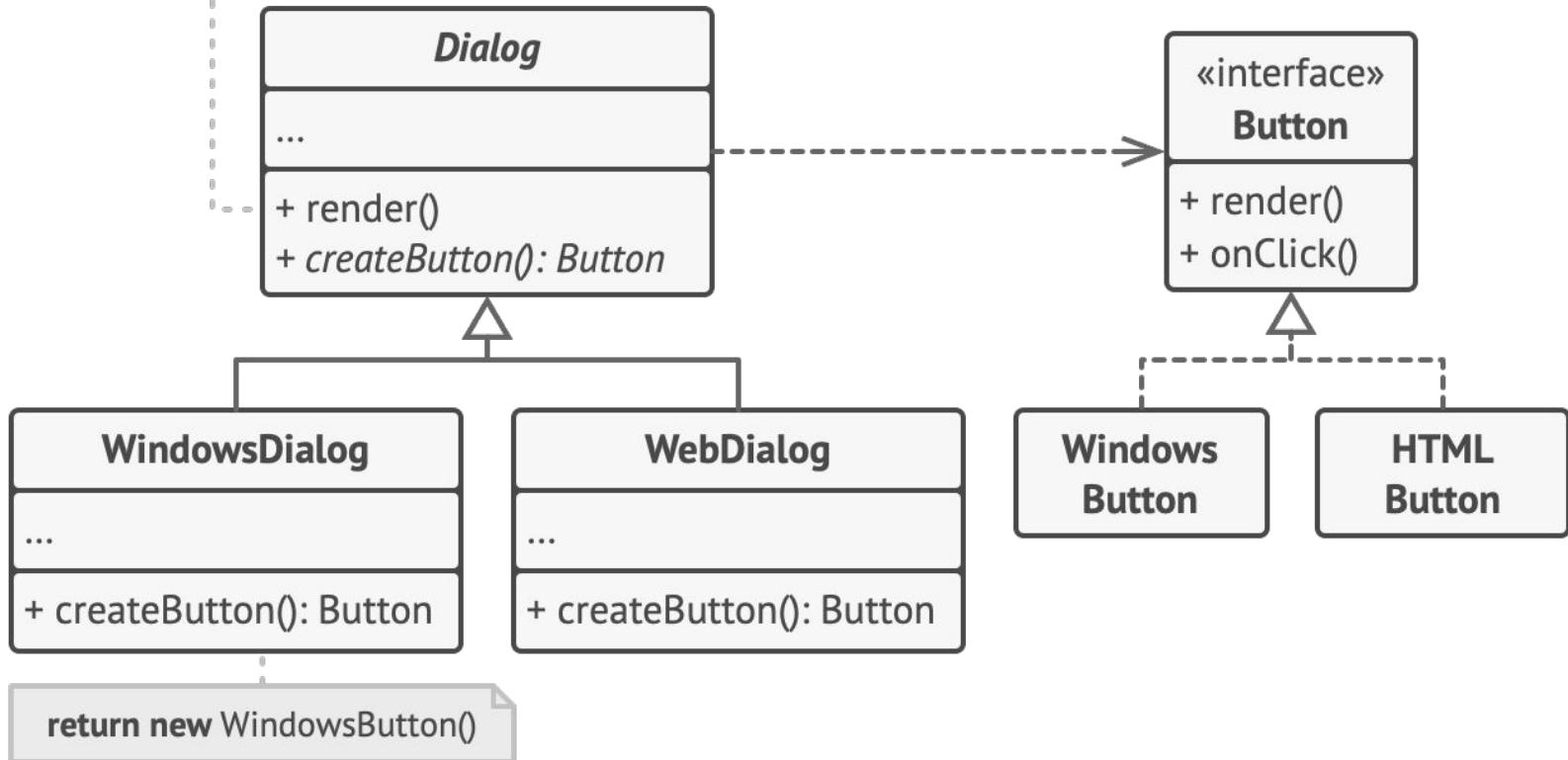
es un patrón de diseño creacional que proporciona una interfaz para crear objetos en una superclase, mientras permite a las subclases alterar el tipo de objetos que se crearán







```
Button okButton = createButton()  
okButton.onClick(closeDialog)  
okButton.render()
```



**Con un cambio de
fábrica, pasamos
a HTML**

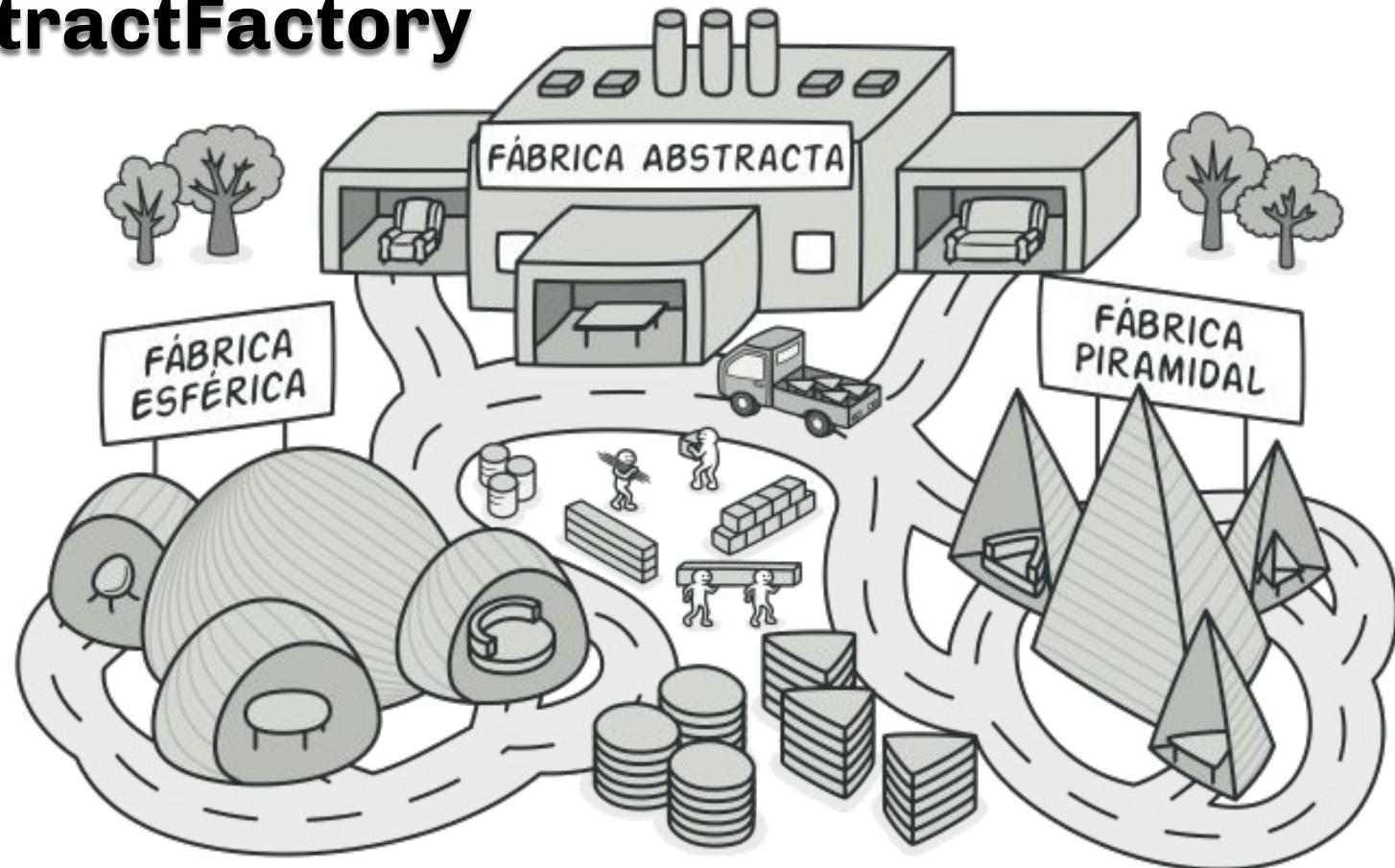
A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



Consecuencias

AbstractFactory



VERÁ, ENCARGUÉ UNAS SILLAS
LA SEMANA PASADA, PERO CREO
QUE NECESITO TAMBIÉN UN SOFÁ...



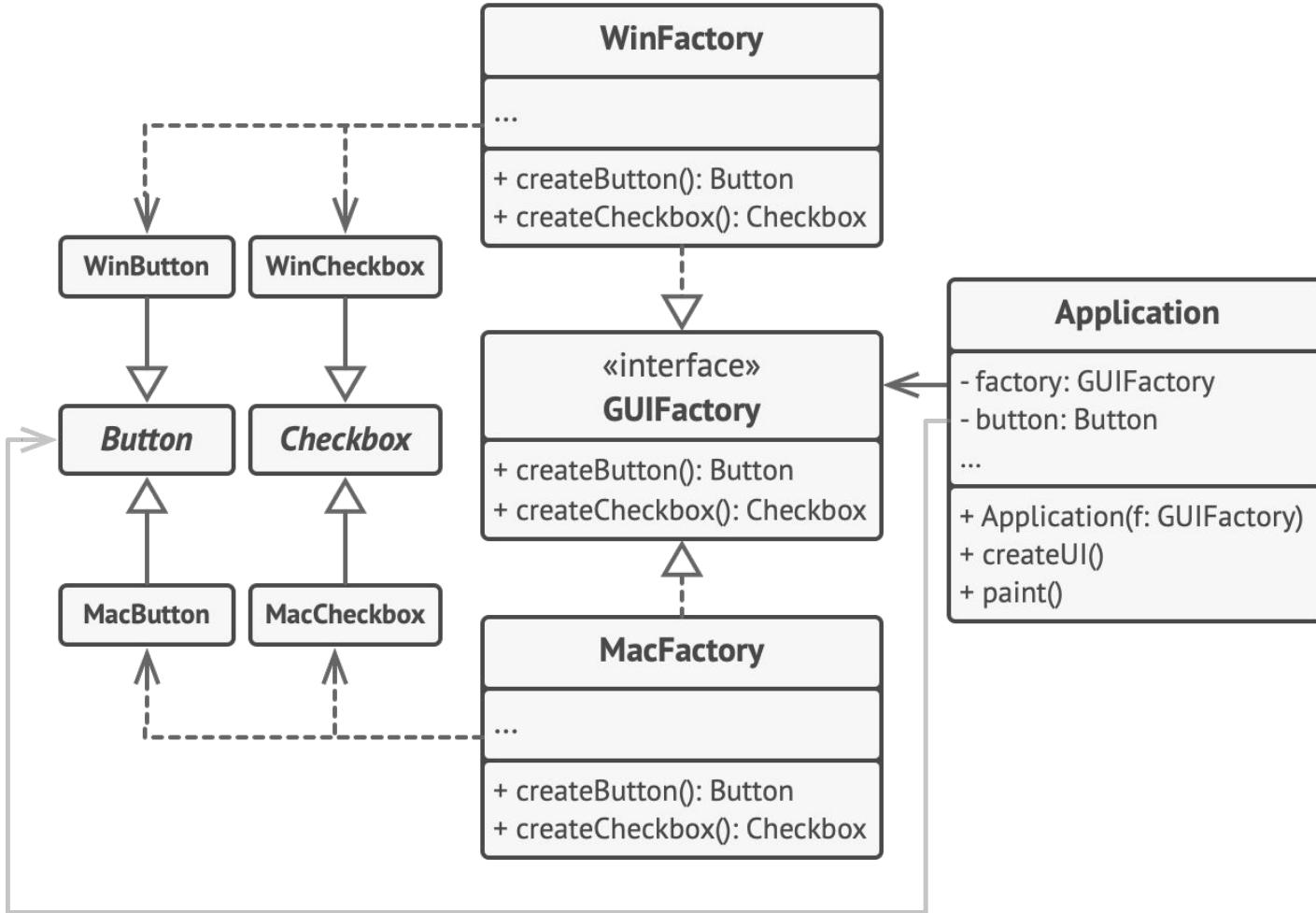
HUM... HAY
ALGO QUE
NO ENCAJA.

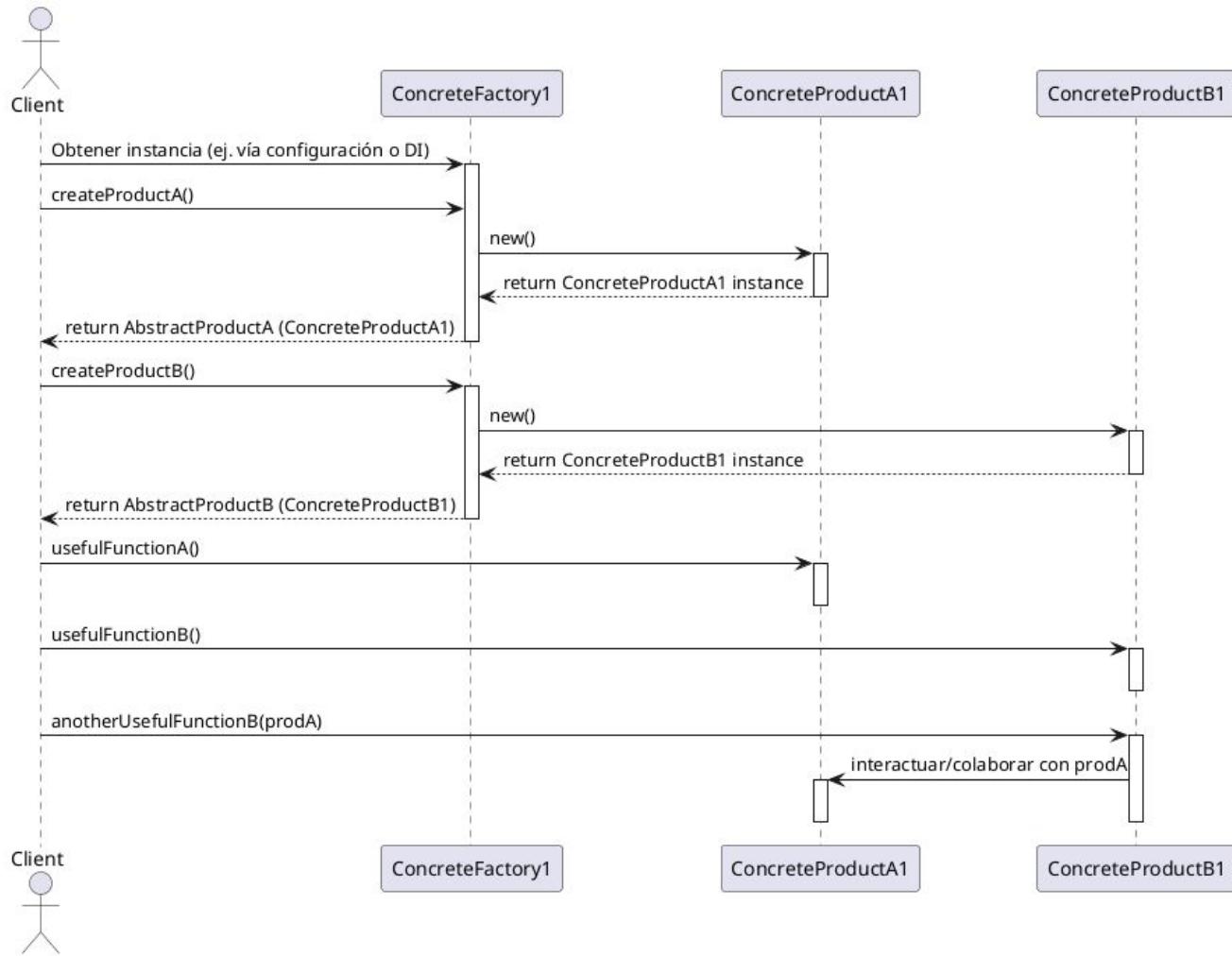


Abstract Factory

es un patrón de diseño creacional que nos permite producir familias de objetos relacionados sin especificar sus clases concretas.

	Silla	Sofá	Mesilla
Art Decó			
Victoriana			
Moderna			





**Lo que sale de la
fabrica es
compatible entre
sí**

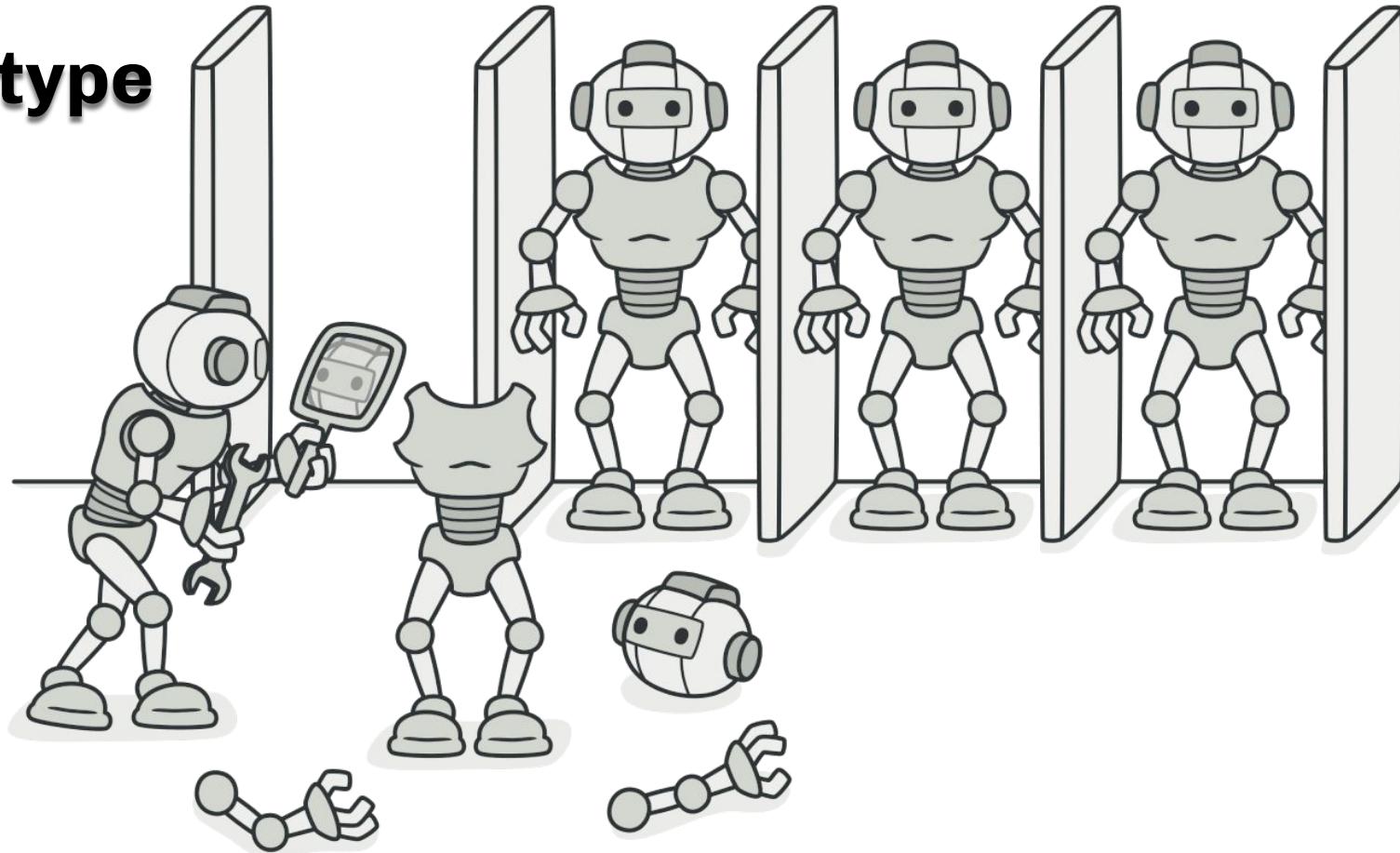
*Claramente la
complejidad
estructural sube*

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



Prototype



¿CREEES QUE
PODAMOS
COPIARLO?



UN MES MÁS TARDE

¡PERFECTO!



Prototype

Crea nuevos objetos mediante la copia de un objeto existente

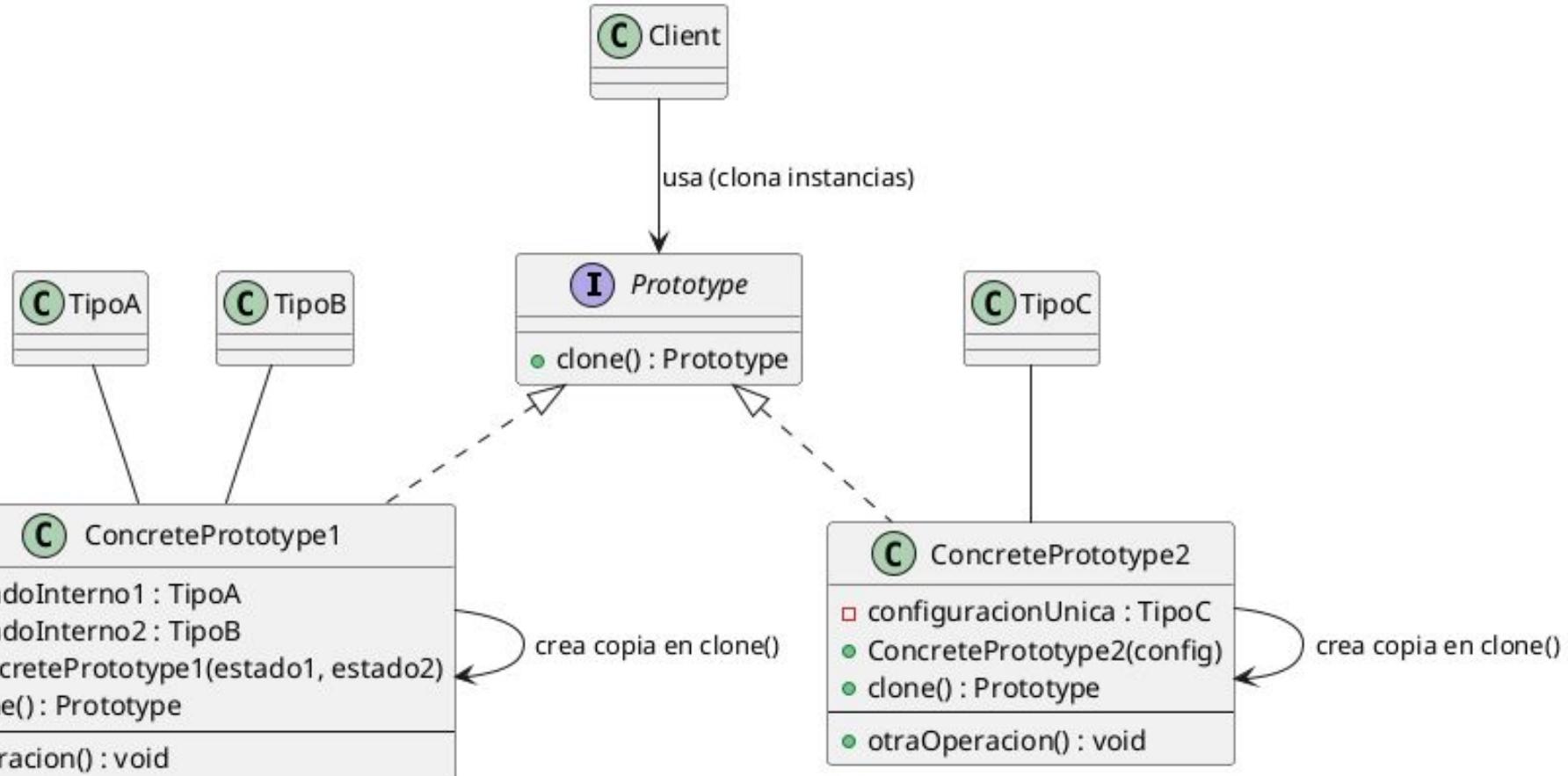


BUENO,
HAGAMOS
UNOS CLONES.

CONFIG A

CONFIG B

CONFIG C





Client

**Prototype Original
:ConcretePrototype1**

Obtener instancia prototipo (ej. desde un registro)

clone()

Crear nueva instancia (Copia 1)

Copiar estado del original

**Copia 1
:ConcretePrototype1**

Retornar Copia 1

Retornar Copia 1

clone()

Crear nueva instancia (Copia 2)

Copiar estado del Copia 1

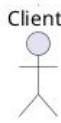
**Copia 2
:ConcretePrototype1**

Retornar Copia 2

Retornar Copia 2

operacion()

operacion()

**Prototype Original
:ConcretePrototype1****Copia 1
:ConcretePrototype1****Copia 2
:ConcretePrototype1**

**La
implementación
de clonar es
crítica**

Es similar a
singleton

**Es una alternativa
más simple a los
Factory**

= se lo puede
combinar con un
Factory
que combina copias con
construcción

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



unrn.edu.ar

UNRN

Universidad Nacional
de Río Negro



| unrnionegro