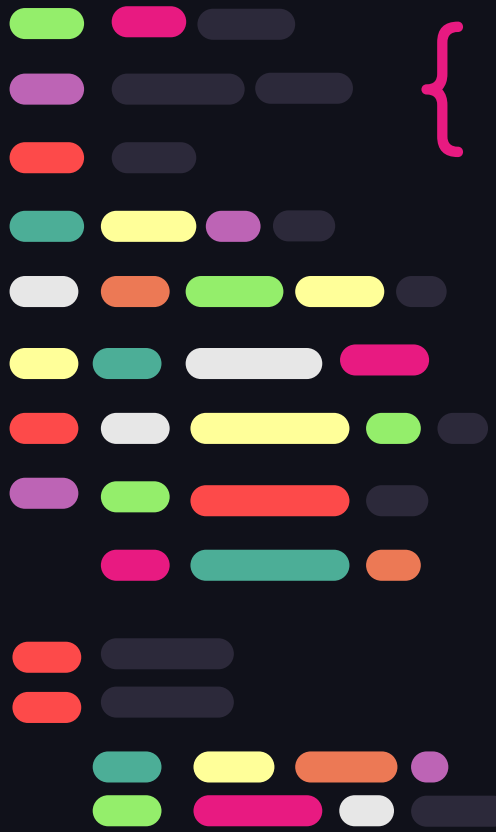
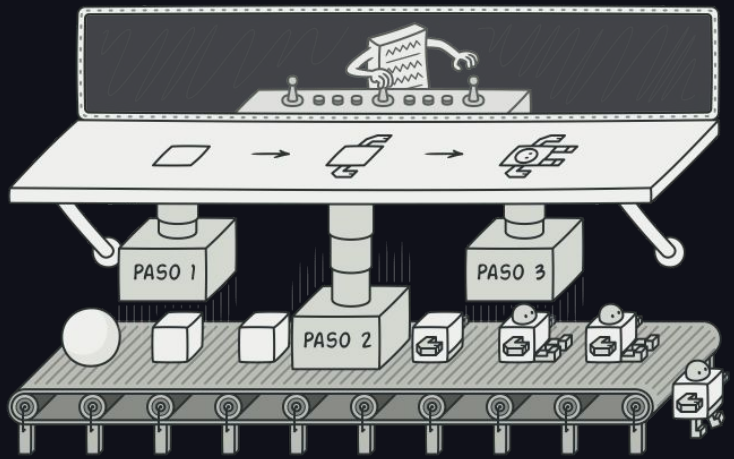


Patrón de diseño

Builder

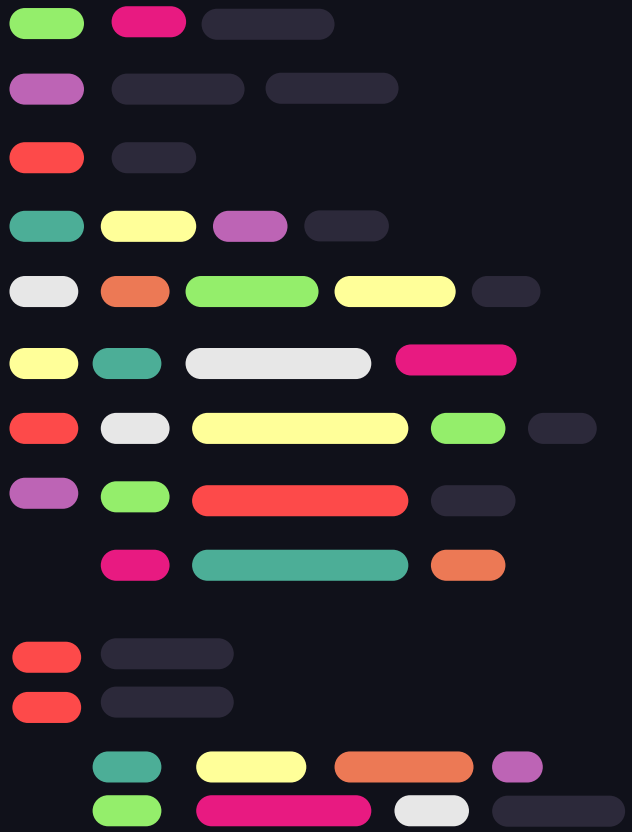


¿Que es?

< Builder es un patrón de diseño creacional que nos permite construir objetos complejos paso a paso, el patrón nos permite producir distintos tipos y representaciones de un objeto empleando el mismo código de construcción >

¿Para qué sirve?

El patrón de diseño “Builder” simplifica la creación de objetos complejos al separar la construcción de su representación, permitiendo flexibilidad reutilización y una mejor organización del código. Ideal para objetos de múltiples configuraciones.

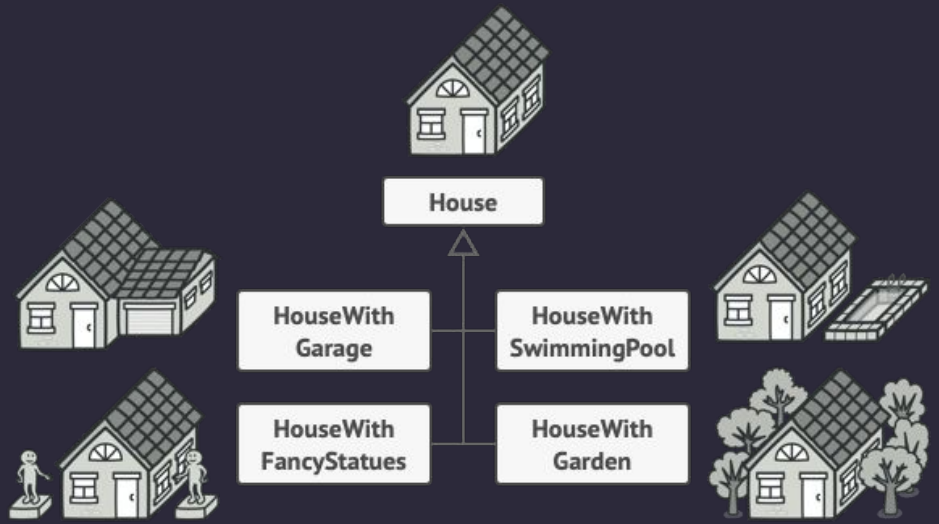


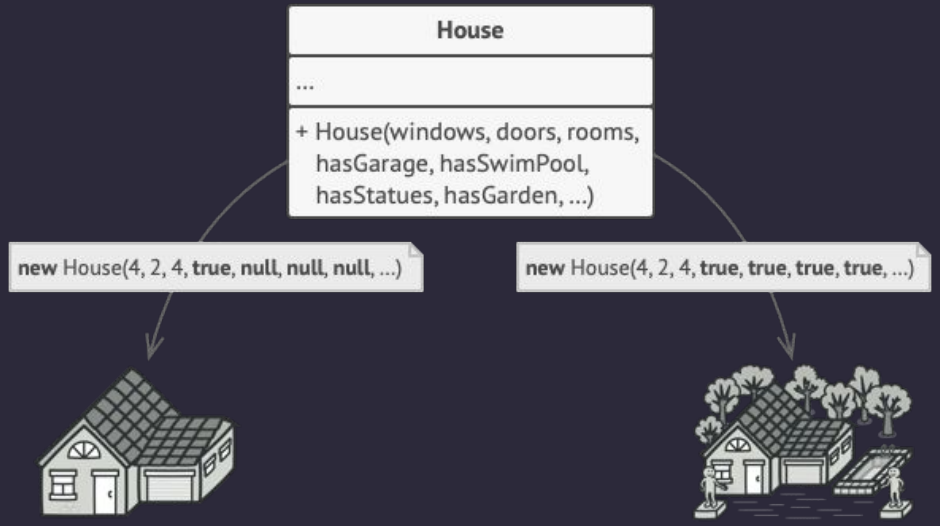
{ ..

Problema

} ..

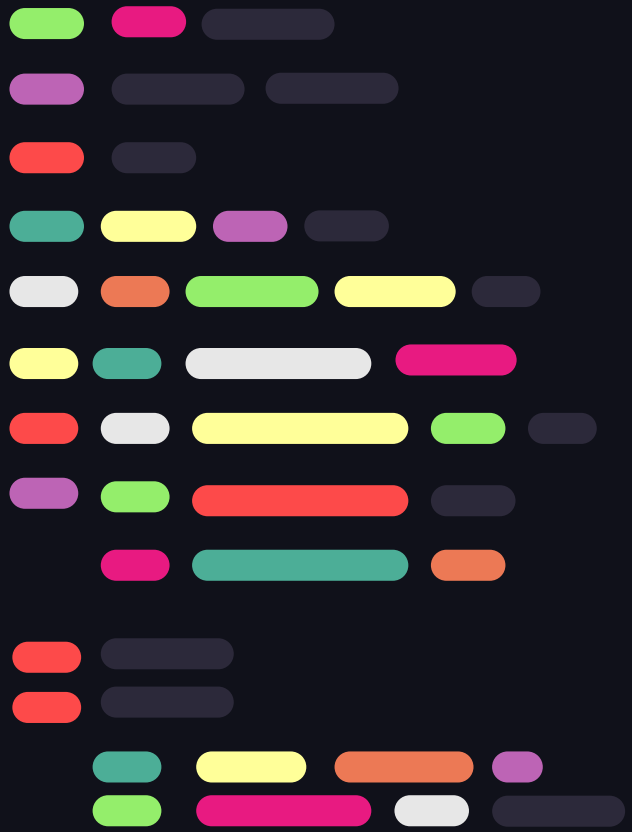
Imagina un objeto complejo con inicialización laboriosa en campos y objetos anidados. Ese objeto puede tener un constructor monstruoso o disperso en el código cliente.





Una posible solución sería crear una clase base Casa y crear un grupo de subclases que extiendan de la clase base.

Otra solución sería crear un enorme constructor dentro de la clase Casa con todos los parámetros posibles

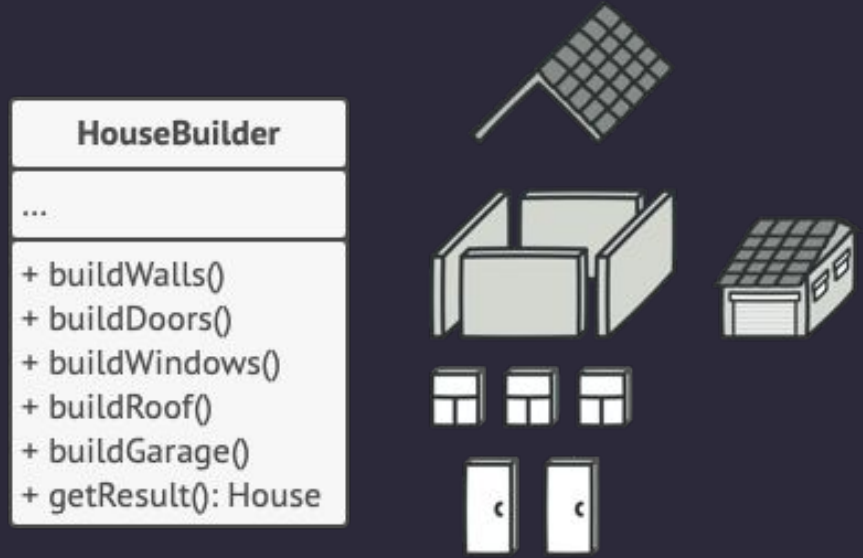


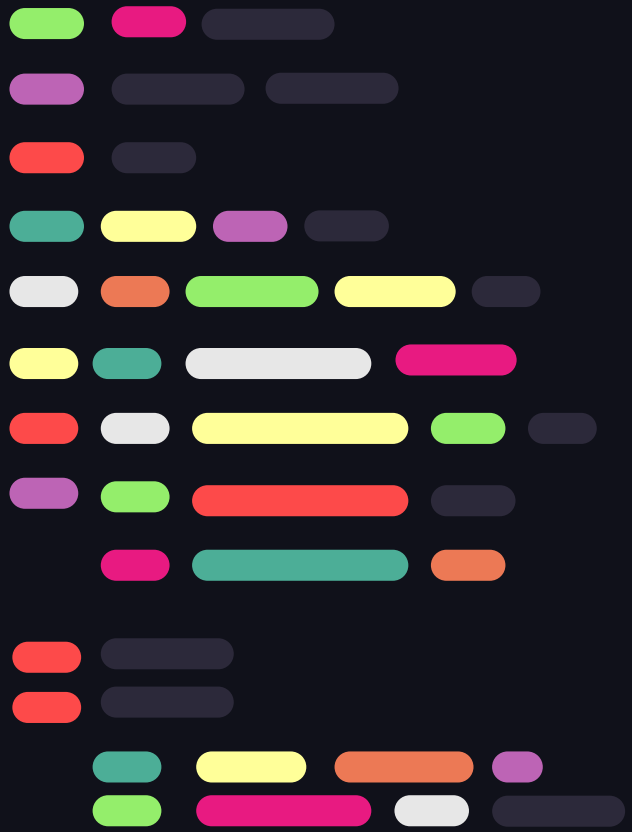
{ ..

Solucion

} ..

Este patrón nos sugiere sacar todo el código de construcción del objeto de su propia clase y colocarlos dentro de objetos independientes llamados **constructores**

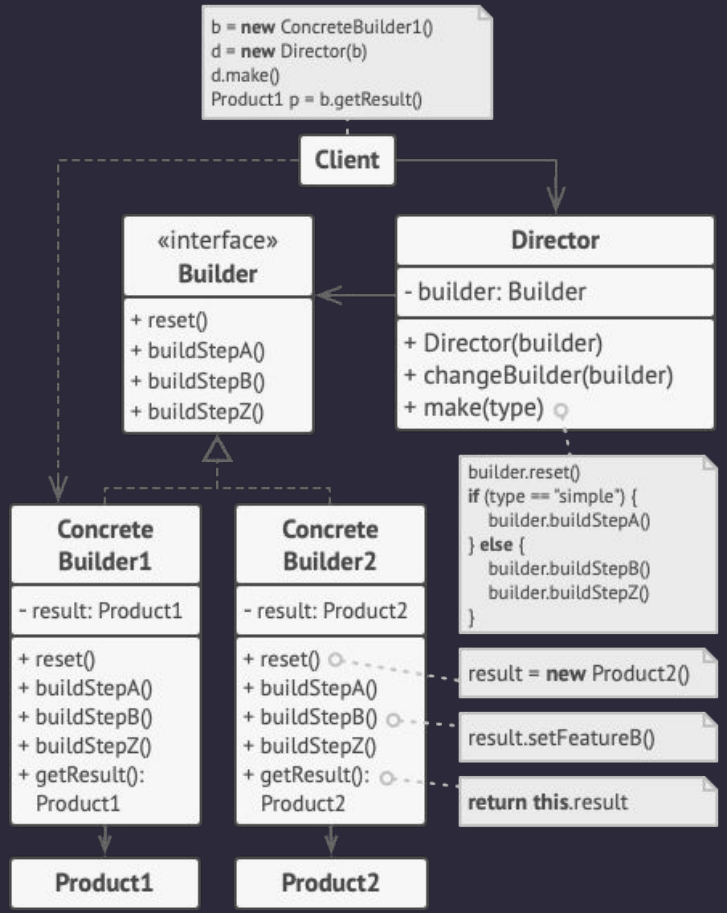




{ ..

Estructura

} ..



{

El patrón se compone de cuatro elementos principales:

- Interfaz Constructor
- Constructor Concreto
- Producto
- Director

}

¿Como funciona?

Pasos para implementar el patron de diseño:

1. El cliente crea una instancia del Director.
2. El cliente crea una instancia del Constructor Concreto y la pasa al Director.
3. El Director utiliza los métodos del Constructor Concreto para construir el objeto paso a paso.
4. El cliente obtiene el Producto resultante del Director.

Ventajas y desventajas

Ventajas

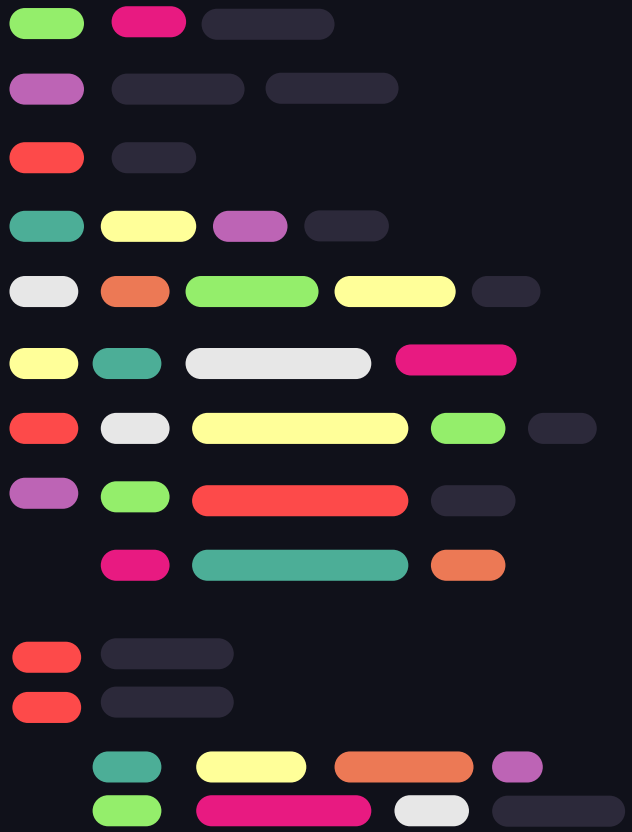
- Proporciona una separación clara entre la construcción del objeto y su representación.
- Permite crear objetos complejos de manera flexible y modular.
- El Director simplifica el proceso de construcción y oculta los detalles de implementación del Cliente.

Desventajas

- Puede introducir una complejidad adicional en el código, especialmente cuando se tienen objetos con configuraciones simples y limitadas. En estos casos, la implementación del patrón puede parecer excesiva y complicada.

Casos de uso

- Sistemas de procesamiento de documentos
- Aplicaciones de comercio electrónico donde los clientes pueden personalizar su producto
- En un sistema de pedidos en un restaurante, puede ser utilizado para crear los diferentes menús
- En editores gráficos
- En un sistema de generación de informes, los usuarios pueden crear informes personalizados con diferentes datos, gráficos y formatos.



{ ..

Gracias!

} ..