<!--Construyendo Objetos Complejos Paso a Paso-->

Patrón Creacional Builder {

<Por="Gerny Diaz"/>

}

¿Qué Exploraremos Hoy?

01 Patrones de Diseño Creacionales

02 El Patrón Builder: ¿Qué es y Para Qué Sirve?

03 ¿Qué Problema Resuelve?

04 ¿Cómo Opera el Builder?

05 Estructura y Participantes

06 Ejemplo Práctico

07 Ventajas y Desventajas

08 ¿Cuándo Debería Usarlo?

09

Patrones Creacionales {

> Los patrones creacionales son una forma de organizar la creación de objetos en un sistema. Estos patrones ofrecen formas más flexibles de hacerlo.

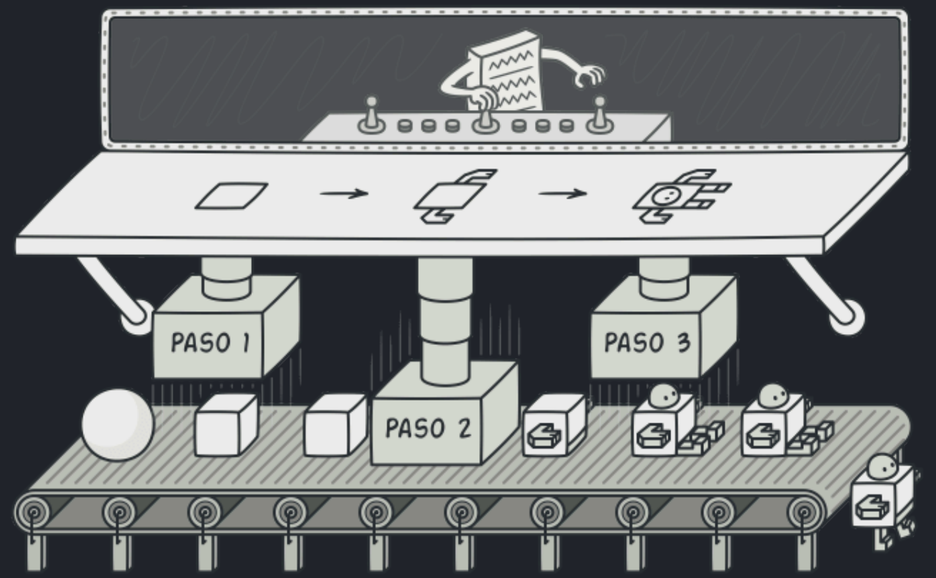
> El objetivo es que el sistema no dependa directamente de cómo se crean los objetos, qué clase usan, ni cómo se ensamblan. Es útil porque los sistemas aveces pueden combinar objetos.

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

}

¿Qué es el Patrón Builder? {

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

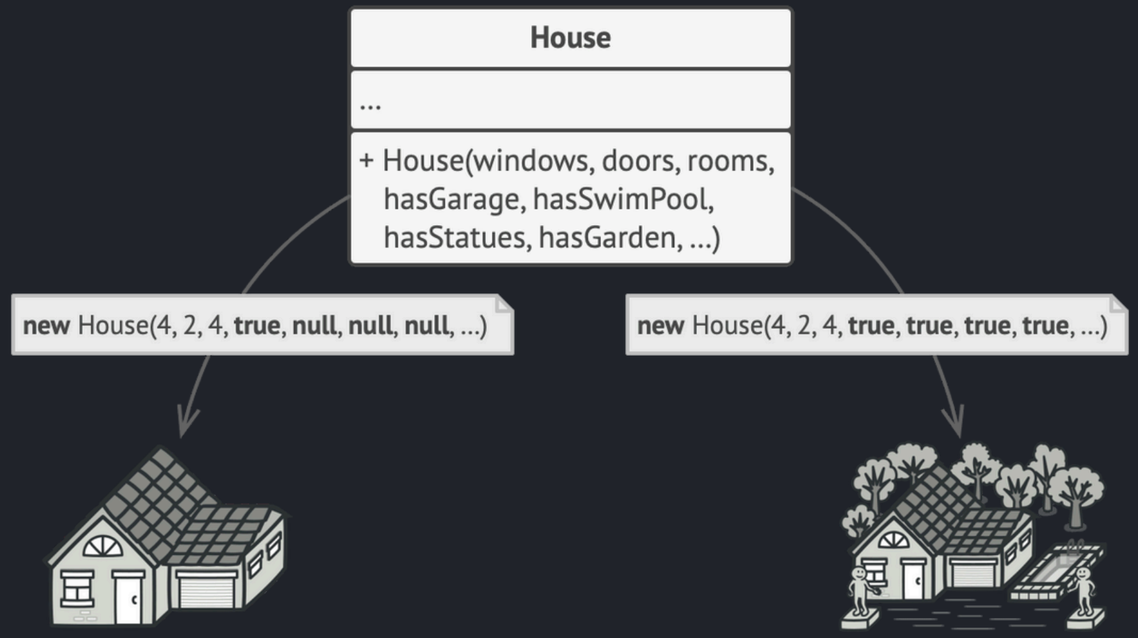


}

¿Qué Problema Resuelve? {

El patrón Builder nos permite resolver problemas al construir objetos complejos de manera controlada, clara y flexible, separando el proceso de construcción.

Esto permite que un mismo proceso de ensamblaje pueda generar distintas versiones del mismo tipo de objeto, sin necesidad de duplicar código o modificar la lógica principal cada vez que se requiere una nueva variante.



}

¿Cómo lo Resuelve? {

Constructores Monstruosos

Evita constructores con demasiados parámetros, haciéndolos difíciles de usar y propensos a errores.

Crear Objetos Complejos Paso a Paso

Permite construir un objeto complejo de forma incremental, controlando cada etapa del proceso

Separación de Responsabilidades

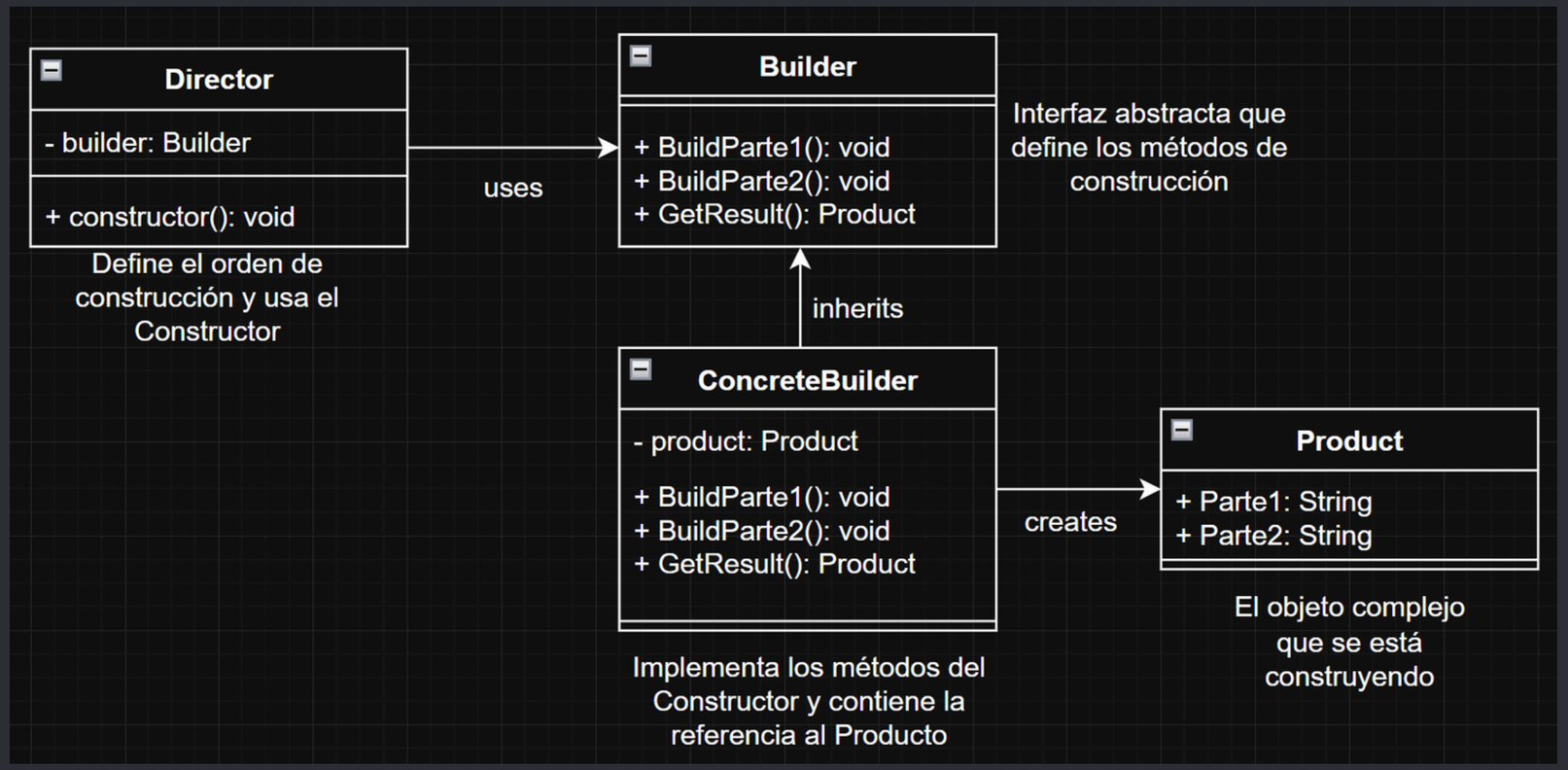
Facilita la creación de diferentes representaciones de un objeto utilizando el mismo proceso de construcción.

Múltiples Versiones de un Mismo Objeto

Aísla la lógica de construcción de la representación del objeto, mejorando la modularidad y facilitando cambios

}

Estructura del Patrón (Diagrama UML) {



}

Los Participantes y sus Roles {

Declara la interfaz para crear las partes del Product.
No sabe qué producto concreto se está haciendo.
Suele incluir un método para obtener el resultado.

Builder

Implementa la interfaz Builder.
Construye y ensambla las partes del producto.
Mantiene la representación del producto que está creando.
Proporciona el método para recuperar el producto final.

ConcreteBuilder

Construye el objeto usando la interfaz Builder.
Conoce la secuencia de pasos ("la receta"),
pero no los detalles de cada paso.

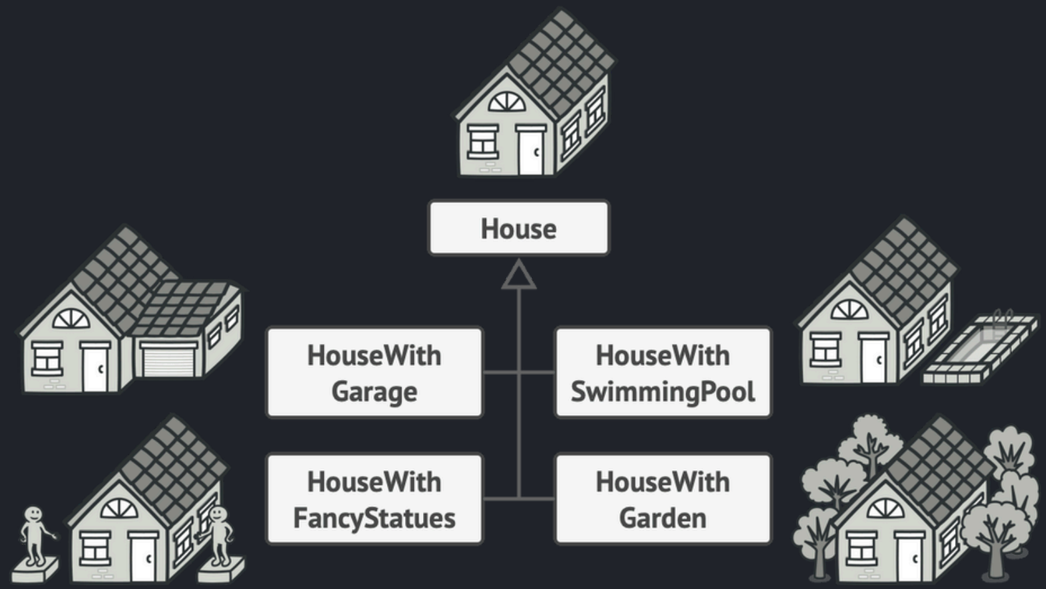
Director

El objeto complejo que se está construyendo.
Sus clases pueden variar mucho.

Product

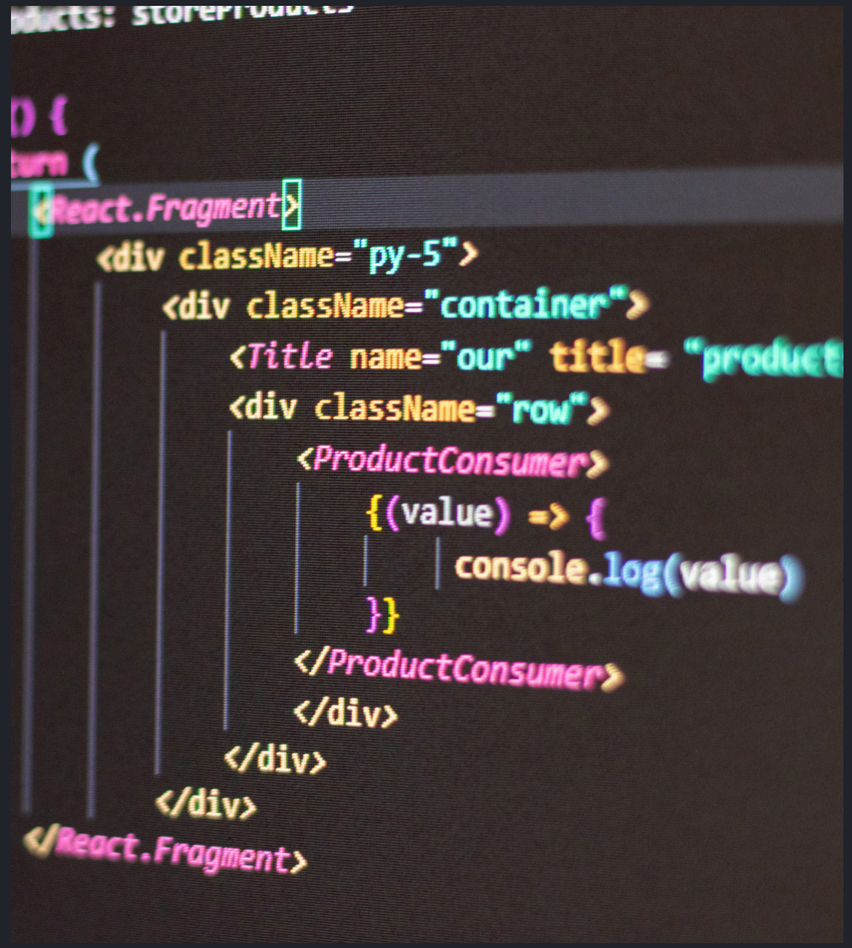
}

Builder en la Práctica {



}

Código de Ejemplo {



}

Ventajas {

Permite variar la representación interna del producto: El cliente no se acopla a los detalles de cómo se crea el producto.

Mejora la legibilidad y reduce errores: Evita constructores con muchos parámetros (constructores telescópicos). Hace el código de creación del cliente más claro.

Aísla el código de construcción y representación: Mejora la modularidad. Cambiar la representación interna no afecta al Director.

Control más fino sobre el proceso de construcción: Se construye paso a paso, permitiendo más flexibilidad que crear todo de una vez.

}

Desventajas {

Puede ser verboso para objetos simples: Si el objeto no es muy complejo, el patrón puede ser una sobrecarga innecesaria.

La interfaz del Builder debe ser lo suficientemente general para cubrir todas las posibles construcciones.

Aumento del número de clases: Se necesita un ConcreteBuilder por cada tipo de producto o representación diferente, lo que puede llevar a una jerarquía de clases paralela.

}

Referencias {

https://refactoring.guru/es/design-patterns/builder

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

}

<!--Construyendo Objetos Complejos Paso a Paso-->

Gracias {

<Por="Gerny Diaz"/>

}