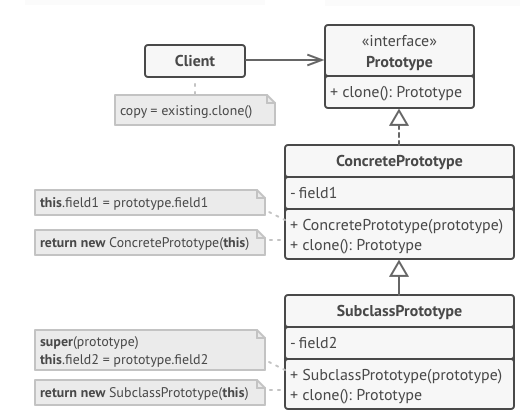
**Prototype**



**¿Qué es?**

El patrón Prototype es un patrón de diseño creacional que permite copiar objetos existentes sin acoplar el código a sus clases específicas. En lugar de crear una nueva instancia de un objeto desde cero, se crea una copia exacta de un objeto ya existente mediante un método de clonación, generalmente llamado clone() o similar.

**¿Para qué funciona?**

* Evitar el uso excesivo de new en la creación de objetos.
* Crear copias de objetos complejos o costosos de construir.
* Duplicar objetos manteniendo su estado actual.
* Permitir la clonación de objetos sin conocer su tipo exacto en tiempo de compilación.

**Ventajas**

* **Eficiencia:** Reducir el costo de creación de objetos complejos o pesados.
* **Flexibilidad:** No depende del tipo concreto del objeto, solo de que implemente la interfaz de clonación.
* **Independencia:** Permite duplicar objetos sin necesidad de depender de sus constructores o lógica de inicialización.
* **Fácil extensión:** Nuevas clases pueden ser fácilmente clonadas si implementan la interfaz adecuada.

**Desventajas**

* **Clonación profunda:** Puede ser difícil implementar clonación profunda si los objetos tienen referencias a otros objetos complejos.
* **Complicación del diseño:** Añadir la funcionalidad de clonación puede introducir complejidad innecesaria si no es estrictamente necesario.
* **Errores con clone():** En lenguajes como Java, el uso de Object.clone() puede ser problemático debido a su comportamiento superficial por defecto.
* **Mantenimiento:** Si se modifica la estructura del objeto, es necesario actualizar la lógica de clonación.

**Cuándo usarlo**

* Cuando la creación de objetos es costosa (por ejemplo, con mucha configuración, datos cargados de base de datos, etc.).
* Cuando se necesita crear múltiples objetos similares a partir de un prototipo.
* Cuando se desea copiar objetos en tiempo de ejecución sin conocer su clase exacta.
* En sistemas que trabajan con editores, formularios o configuraciones reutilizables.

**Cuándo no usarlo**

* Cuando los objetos son simples y fáciles de crear con un constructor.
* Cuando no se necesita duplicar el estado del objeto.
* Cuando el proceso de clonación es más complejo que crear un nuevo objeto.
* Si el lenguaje no ofrece soporte adecuado para la clonación de objetos (o requiere implementación manual compleja).

EXPLICACION CODIGO   
creación de objetos duplicados (copias exactas) a partir de una instancia existente, sin necesidad de volver a construirlos desde cero. Esto se logra mediante una interfaz que define un método de clonación, y una clase concreta que implementa esa interfaz y realiza el clon usando el método clone().

Se crea una clase que representa un objeto (en este caso, un registro de usuario con un valor). Ese objeto se instancia y se configura inicialmente. Luego, en lugar de crear otro objeto manualmente, se genera una **copia clonada** utilizando el método definido por la interfaz. De esta forma, se tiene un nuevo objeto con las mismas características que el original.

El propósito del patrón en este contexto es mostrar cómo se pueden crear múltiples objetos similares de manera eficiente, reutilizando la configuración de uno ya existente. Es útil especialmente cuando la creación o configuración de un objeto es compleja, o cuando se requiere flexibilidad para duplicar sin acoplar el código a clases concretas.