Patrón de diseño: Prototype

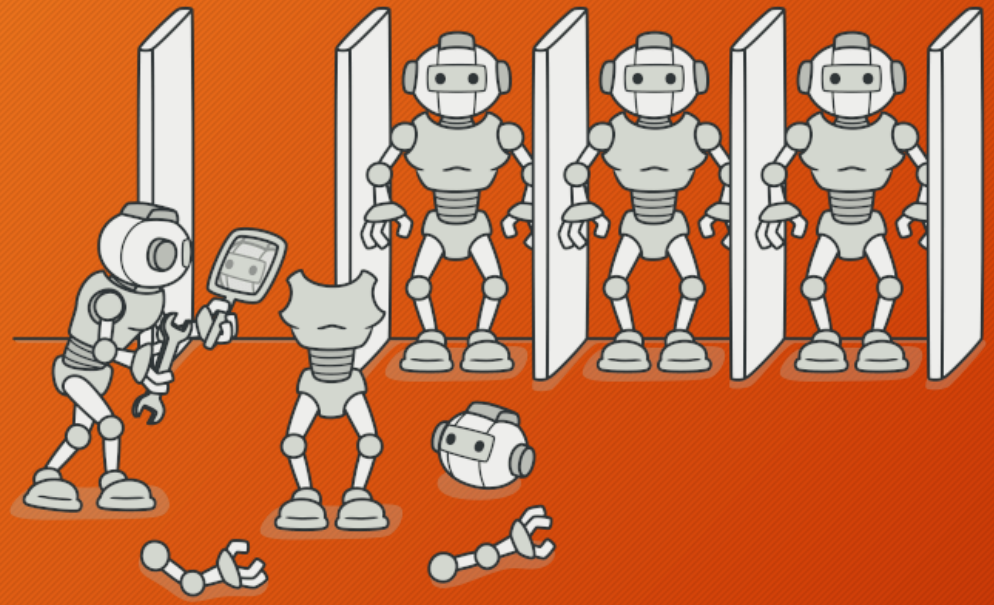
También llamado: Prototipo, Clon, Clone

¿Qué es *Prototype*?

- Es un patrón de diseño creacional, que tiene como objetivo crear a partir de un modelo. Especificar el tipo de objetos que se crearán mediante una instancia prototípica, y crear nuevos objetos copiando este prototipo.

- Tiene gran simplicidad ya que su concepto es de hacer una copia exacta de otro objeto, esto en lugar de crear uno nuevo, permite crear los objetos pre diseñados sin la necesidad de conocer los detalles de la creación.

Imágenes funny



¿Cuándo se utiliza *Prototype*?

- La programación de hoy en día tiene que ver directamente con costos. El ahorro es un gran problema cuando se trata de utilizar los recursos de la computadora, por lo que los programadores están haciendo su mejor esfuerzo para encontrar formas de mejorar el rendimiento. Cuando hablamos de la creación de objetos en los que podemos encontrar una manera óptima de instanciarlos, y es donde entra en juego la clonación. Si el costo de la creación de un nuevo objeto es grande y la creación es intensivo en recursos, clonamos el objeto.
  El patrón de diseño Prototype funciona para esto. Permite que un objeto pueda crear una copia de si mismo sin conocer su clase o ningún detalle sobre cómo crearlos.

Problema :s

- Digamos que tienes un objeto y quieres crear una copia exacta de él. ¿Cómo lo harías? En primer lugar, debes crear un nuevo objeto de la misma clase. Después debes recorrer todos los campos del objeto original y copiar sus valores en el nuevo objeto.

- No todos los objetos se pueden copiar de este modo, porque algunos de los campos del objeto pueden ser privados e invisibles desde fuera del propio objeto.


¿CREES QUE PODAMOS COPIARLO?


UN MES MÁS TARDE
¡PERFECTO!

Solución :D

- El patrón Prototype delega el proceso de clonación a los propios objetos que están siendo clonados. El patrón declara una interfaz común para todos los objetos que soportan la clonación. Esta interfaz nos permite clonar un objeto sin acoplar el código a la clase de ese objeto. Normalmente, dicha interfaz contiene un único método “Clonar”.

Hablando sobre los prototipos

-La implmentación del método “clonar” es muy parecida en todas las clases. El método crea un objeto a partir de la clase actual y lleva todos los valores de campo del viejo objeto, al nuevo. Se puede incluso copiar campos privados, porque la mayoría de los lenguajes de programación permite a los objetos acceder a campos privados de otros objetos que pertenecen a la misma clase.

-Un objeto que soporta la clonación se denomina *prototipo*. Cuando tus objetos tienen decenas de campos y miles de configuraciones posibles, la clonación puede servir como alternativa a la creación de subclases.

Tipos de clonación

- **Clonación profunda**

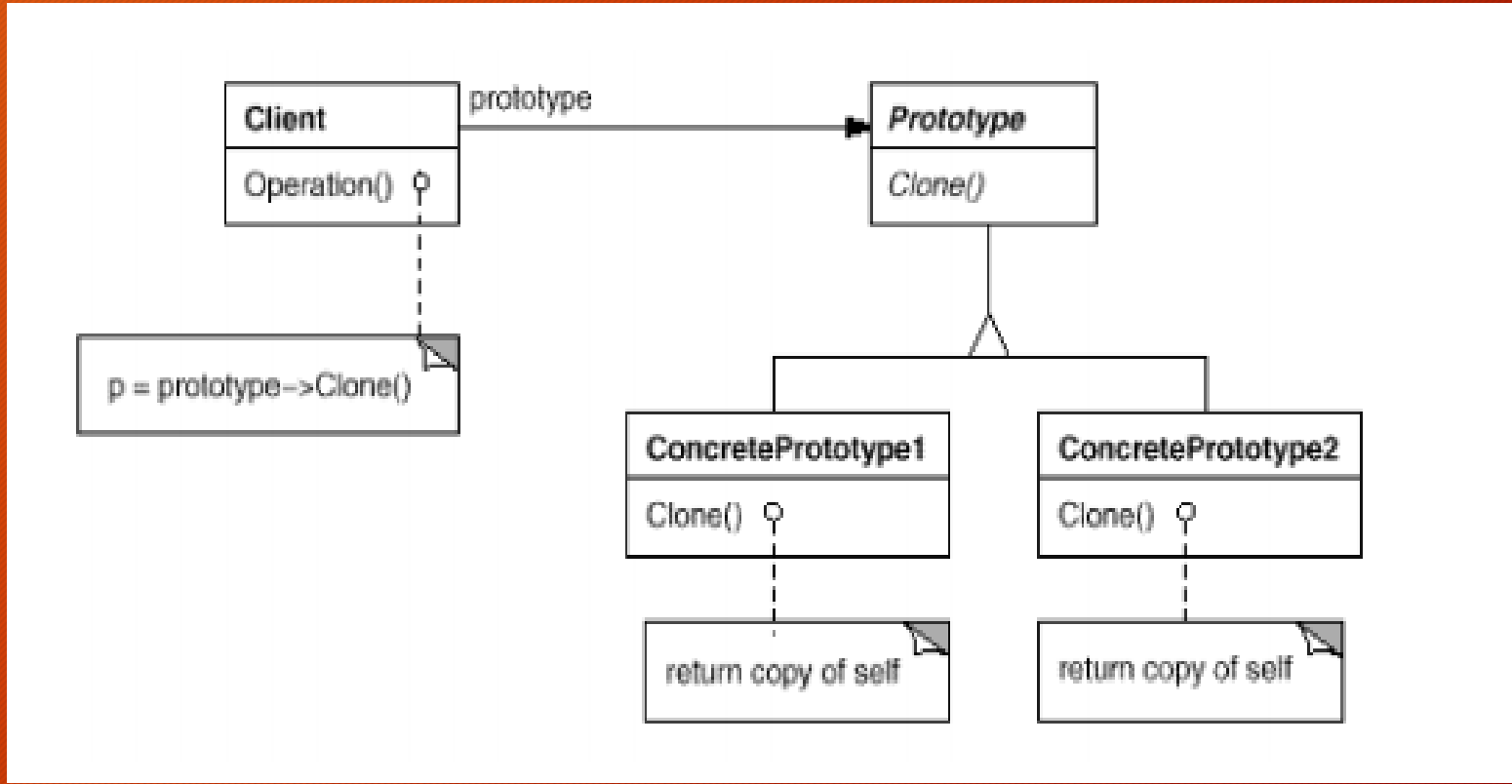
Esta clonación se realiza cuando el objeto que se quiere clonar tiene como atributos otros objetos (1 o más) que se deben clonar de igual manera, para retornar una clonación completa del objeto y no referencias a otros ya existentes.

- **Clonación superficial:**

Por otro lado, esta clonación es aquella que se realiza cuando el objeto que se quiere clonar no posee otros que se deban copiar(tales como Integer, Char, Bool). Se realiza a nivel de bits.

¿Cómo se implementa?

La idea de este patrón es que los objetos que necesiten ser clonados implementen una interfaz que tenga este método. O bien heredar de una clase abstracta que tenga el método abstracto clonar() para que lo desarrollen sus hijos.



Ejemplo de uso

- Los motores de procesamiento de tasas/ratios requieren a menudo la búsqueda de muchos valores de configuración diferentes, haciendo que la inicialización del motor sea un proceso relativamente costoso. Cuando se necesitan varias instancias del motor, por ejemplo, para importar datos de una manera multihilo, el costo de inicializar muchos motores es alto.
Utilizando el patrón de prototipo podemos asegurarnos de que sólo se tiene que inicializar una sola copia del motor, simplemente clonar el motor para crear un duplicado del objeto ya inicializado.El beneficio adicional de esto es que los clones pueden ser simplificados para incluir sólo datos relevantes para su situación.

¿Entonces rifa o no rifa?

- PROS:

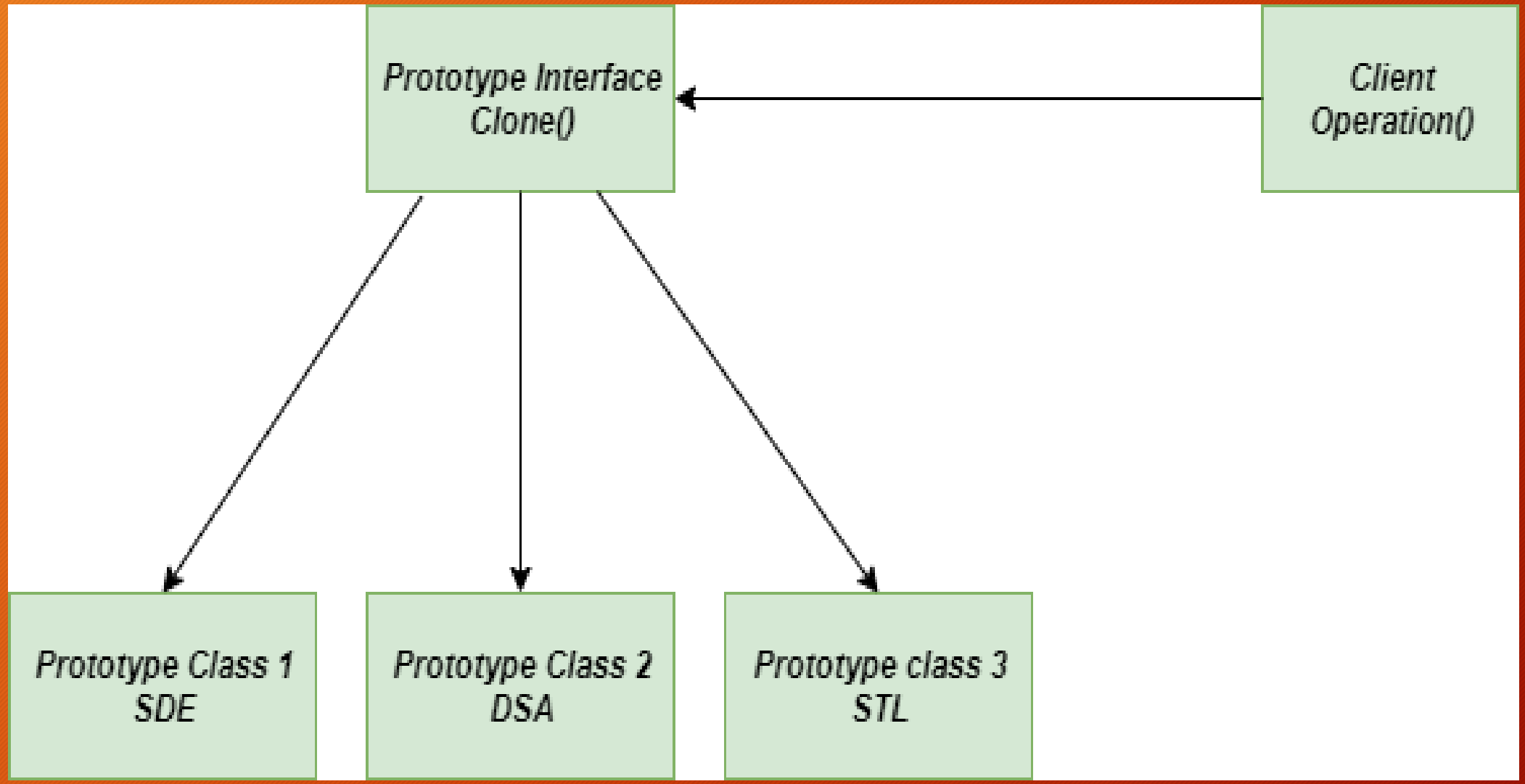
- Clonar un objeto es mucho más rápido que crearlo.

- Un programa puede añadir y borrar dinámicamente objetos prototipo en tiempo de ejecución.

- Menor número de subclases.

- Proporciona valores variables a nuevos objetos: todos los sistemas altamente dinámicos le permiten definir un nuevo comportamiento a través de la composición de objetos especificando valores para las variables de un objeto y no definiendo nuevas clases.

- CONTRAS:
- En objetos muy complejos, implementar la interfaz Prototype puede ser muy complicada.
- La jerarquía de prototipos debe ofrecer la posibilidad de clonar un elemento y esta operación puede no ser sencilla de implementar. Por otro lado, si la clonación se produce frecuentemente, el coste puede ser importante.
- Desperdicio de recursos en un nivel inferior: podría demostrarse como un exceso de recursos para un proyecto que utiliza muy pocos objetos.


Prototype Interface
Clone()
Client
Operation()
Prototype Class 1
SDE
Prototype Class 2
DSA
Prototype class 3
STL