Prototype

Propósito

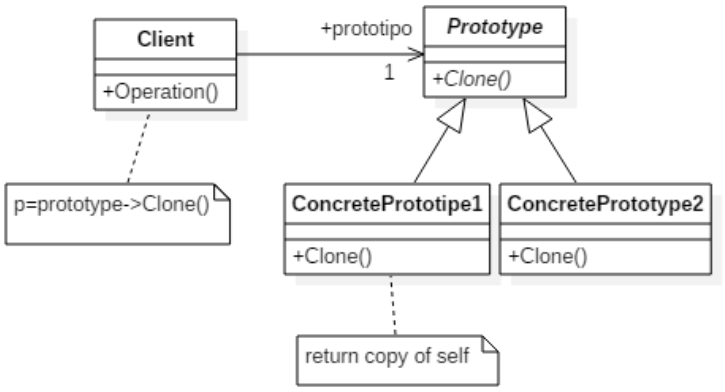
Especifica el tipo de objetos a crear utilizando una instancia prototípica, y los crea copiando este prototipo.

Aplicabilidad

El patrón Prototype se utiliza cuando un sistema debe ser independiente del modo en que sus productos se crean, componen y representan, y además:

- Cuando las clases a instanciar se especifican en tiempo de ejecución, por ejemplo, mediante carga dinámica, o
- Evitar construir una jerarquía de clases de fabricación paralela a la jerarquía de productos, o
- Cuando las instancias de una clase pueden tener sólo unas pocas combinaciones diferentes de estado. En ese caso puede ser más conveniente instalar una serie de prototipos y clonarlos en vez de tener que instanciar la clase manualmente cada vez que se requiere

Estructura



Participantes

- Prototype: declara una interface para clonarse a sí mismo
- ConcretePrototype: implementa la operación Clone para clonarse a sí mismo
- Cliente: crea un nuevo objeto solicitando al prototipo que se clone a sí mismo

Colaboraciones

- Un cliente solicita a un prototipo que se clone a sí mismo

Consecuencias

El patrón Prototype tiene las mismas consecuencias que el Abstract Factory y el Builder: oculta al cliente las clases de los productos concretos, reduciendo así el número de nombres que los

clientes conocen. Además, estos patrones permiten a los clientes trabajar con clases específicas de aplicación sin modificaciones.

Otras ventajas del patrón Prototype son:

1. Agregar y eliminar productos en tiempo de ejecución. Los prototipos nos permiten incorporar una nueva clase de producto en el sistema con solo registrar la instancia prototípica en el cliente. Es más flexible que otros patrones creacionales ya que un cliente puede instalar y eliminar prototipos en tiempo de ejecución.
2. Especificar nuevos objetos variando valores. Los sistemas altamente dinámicos te permiten definir nuevos comportamientos mediante la composición de objetos -al especificar valores en las variables de un objeto, por ejemplo- y no mediante la definición de nuevas clases. Puedes definir nuevos tipos de objetos mediante la instanciación de clases existentes y registrándolas como prototipos. Un cliente puede exhibir un nuevo comportamiento delegando responsabilidad en el prototipo.
   Este tipo de diseño permite a los usuarios definir nuevas clases sin programar. De hecho, clonar un prototipo es similar a instanciar una clase. El patrón Prototype puede reducir enormemente el número de clases de un sistema.
3. Especificar nuevos objetos puede variar la estructura. Muchas aplicaciones construyen objetos con partes y subpartes. El patrón Prototype permite definir las partes como prototipos y componerlas para formar un producto.
4. Reduce la subclasificación. El patrón Factory Method con frecuencia produce una jerarquía de clases Creator paralela a la jerarquía de productos. El patrón Prototype nos permite clonar un prototipo en vez de pedir al Factory method que haga un nuevo objeto.
5. Configurar dinámicamente una aplicación con clases. Algunos entornos de tiempo de ejecución te permiten cargar clases dinámicamente.

El principal inconveniente del patrón Prototype es que cada subclase de Prototype debe implementar la operación Clone, lo cual puede ser difícil porque las clases ya existan, no soporten la operación o por las referencias circulares.