**Prototype**

É um padrão de projeto criacional que permite copiar objetos existentes sem fazer seu código ficar dependente de suas classes.

Ele delega o processo de clonagem para o próprio objeto que está sendo clonado, declarando uma interface comum para todos os projetos que suportam clonagem. Essa interface permite que você clone um objeto sem acoplar seu código à classe desse objeto, e geralmente, tal interface contém apenas um único método clonar.

A implementação desse método é muito parecida em todas as classes. O método cria um objeto da classe atual e carrega todos os valores de campo do antigo objeto para o novo objeto. Pode-se até mesmo copiar campos privados porque a maioria das linguagens de programação permite objetos acessar campos privados de outros objetos que pertençam a mesma classe.

Um objeto que suporta clonagem é chamado de um protótipo. Quando seus objetos têm dúzias de campos e centenas de possíveis configurações, cloná-los pode servir como uma alternativa à subclasses.

**Funcionamento**

É criado um conjunto de objetos, configurados de diversas formas. Quando for necessitar de um objeto parecido com o que foi configurado, precisa-se apenas clonar um protótipo ao invés de construir um novo objeto a partir do nada.

**Implementação**

1. Cria-se uma interface protótipo e declare o método clonar nela, ou apenas adicione o método para todas as classes de uma hierarquia de classes existente, se você tiver uma.
2. Uma classe protótipo deve definir o construtor alternativo que aceita um objeto daquela classe como um argumento. O construtor deve copiar os valores de todos os campos definidos na classe do objeto passado para a nova instância recém criada. Se você está mudando uma subclasse, você deve chamar o construtor da classe pai para permitir que a superclasse lide com a clonagem de seus campos privados.

Se a sua linguagem de programação não suporta sobrecarregamento de métodos, você pode definir um método especial para copiar os dados do objeto. O construtor é um local mais conveniente para se fazer isso porque ele entrega o objeto resultante logo depois que você chamar o operador new.

1. O método de clonagem geralmente consiste em apenas uma linha: executando um operador new com a versão protótipo do construtor. Observe que toda classe deve explicitamente sobrescrever o método de clonagem e usar sua própria classe junto com o operador new. Do contrário, o método de clonagem pode produzir um objeto da classe superior.
2. Opcionalmente, crie um registro protótipo centralizado para armazenar um catálogo de protótipos usados com frequência.

Você pode implementar o registro como uma nova classe factory ou colocá-lo na classe protótipo base com um método estático para recuperar o protótipo. Esse método deve procurar por um protótipo baseado em critérios de busca que o código cliente passou para o método. O critério pode ser tanto uma string ou um complexo conjunto de parâmetros de busca. Após o protótipo apropriado ser encontrado, o registro deve cloná-lo e retornar a cópia para o cliente.

Por fim, substitua as chamadas diretas para os construtores das subclasses com chamadas para o método factory do registro do protótipo.

**Vantagem**

* Você pode clonar objetos sem acoplá-los a suas classes concretas.
* Você pode se livrar de códigos de inicialização repetidos em troca de clonar protótipos pré-construídos.
* Você pode produzir objetos complexos mais convenientemente.
* Você tem uma alternativa para herança quando lidar com configurações pré determinadas para objetos complexos.

**Desvantagem**

* Clonar objetos complexos que têm referências circulares pode ser bem complicado.

**Referências**

PADRÕES de projetos,. **Refactoring.guro**, 2022. Disponível em: https://refactoring.guru/pt-br/design-patterns/prototype . Acesso em: 03 de nov. de 2022.