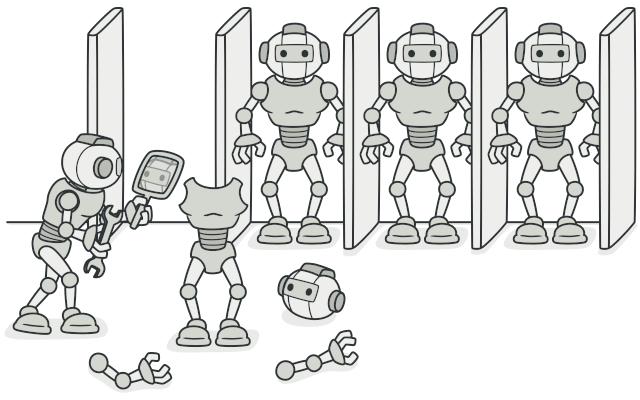
**Prototype**

**(Also known as:Clone)**

**1.Introduction:**

**Prototype** is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.



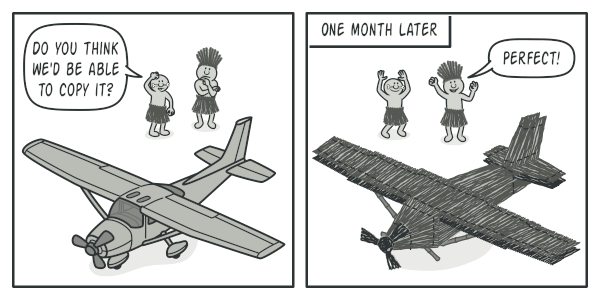
2.Define:

Prototype pattern refers to creating duplicate object while keeping performance in mind. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

3.Purpose of use

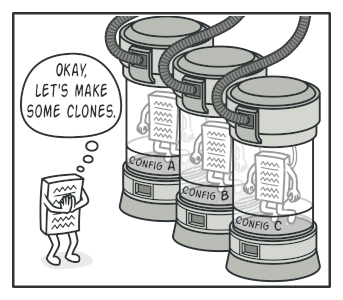
a)Problem

Say you have an object, and you want to create an exact copy of it. How would you do it?

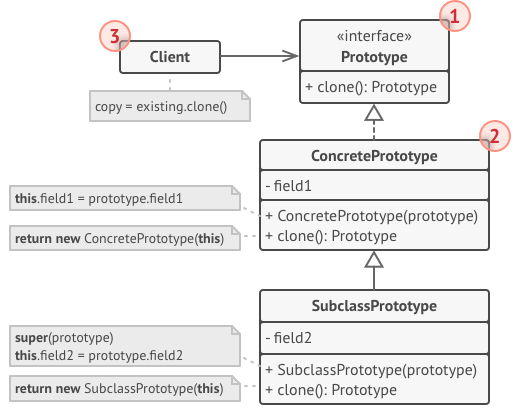


b)Solution:

You create a set of objects, configured in various ways. When you need an object like the one you’ve configured, you just clone a prototype instead of constructing a new object from scratch.



4)Architecture:



Components:

1. The **Prototype** interface declares the cloning methods.
2. The **Concrete Prototype** class implements the cloning method.
3. The **Client** can produce a copy of any object that follows the prototype interface.

According to the following structure, we can claim that:

* Prototype can avoid to creating too much subclasses like Abstract Factory Pattern
* It reduces the cost of creating new “standard” object
* Using prototype gains more performance than using new() operator to create new object.

In prototype, there are two way to clone:

* Shallow Copy - In shallow copy, we only clone the parent object and not its containing objects.
* Deep Copy - In a deep copy, we clone the parent object as well as its containing objects.

## 5) Applicability

* When the classes are instantiated at runtime.
* When the cost of creating an object is expensive or complicated.
* When you want to keep the number of classes in an application minimum.
* When the client application needs to be unaware of object creation and representation

6) Advantages and disadvantages:

Advantages:

* It reduces the need of sub-classing.
* It hides complexities of creating objects.
* The clients can get new objects without knowing which type of object it will be.
* It lets you add or remove objects at runtime.

Disadvantages:

* For projects, which require less number of objects, implementing prototype patterns can be an overkill.
* It hides the actual structure of the object/program from the client.
* must implement, the clone() method, and in some cases, this implementation can be difficult.
* loning complex objects that have circular references might be very tricky.

Many designs start by using Factory Method (less complicated and more customizable via subclasses) and evolve toward Abstract Factory, Prototype, or Builder (more flexible, but more complicated).

7)Relations with Other Patterns

* Abstract Factory classes are often based on a set of Factory Methods, but you can also use Prototype to compose the methods on these classes.
* Prototype can help when you need to save copies of Commands into history.
* Designs that make heavy use of Composite and Decorator can often benefit from using Prototype. Applying the pattern lets you clone complex structures instead of re-constructing them from scratch.
* Prototype isn’t based on inheritance, so it doesn’t have its drawbacks. On the other hand, Prototype requires a complicated initialization of the cloned object. Factory Method is based on inheritance but doesn’t require an initialization step.
* Sometimes Prototype can be a simpler alternative to Memento. This works if the object, the state of which you want to store in the history, is fairly straightforward and doesn’t have links to external resources, or the links are easy to re-establish.
* Abstract Factories, Builders and Prototypes can all be implemented as Singletons.