Prototype:

**Other name (if any)**

placeholder

**What it does**

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.

**Where to use**

1. Use the Prototype pattern when creating objects is more expensive or complex than copying existing ones.  
2. If object creation involves significant resources, such as database or network calls, and you have a similar object available, cloning can be more efficient.  
3. Use the Prototype pattern when your system needs to support a variety of objects with slight variations.  
4. Creating a clone can be faster than creating an object from scratch, especially when the initialization process is resource-intensive.

**Steps**

1. Prototype Interface: It will only have one mandatory function which is clone.  
2. Concrete Prototype Class: It will implement the clone function of the prototype interface. It may also have some additional functions if it needs. In the clone function, it will return the same class using new keyword. For example: return new ConcretePrototypeClass ();  
3. Client Code: It will create the object of the concrete prototype class using concrete prototype class. Then call the clone function to clone object like this:  
ConcretePrototypeClass cloneObject = (ConcretePrototypeClass) [object.clone](http://object.clone)();

**Special cases (if any)**

placeholder

**Advantages**

1. As there is a clone function with eger concrete prototype class, so the function can easily access and modify its private properties.

**Disadvantages**

1. If object creation is simple and does not involve significant resource consumption, and there are no variations of objects, using the Prototype pattern might be unnecessary complexity.

**Code**

Coding Concept

**Difference with similar pattern**

placeholder

**Diagram**

Geekforgeeks + Javatpoint