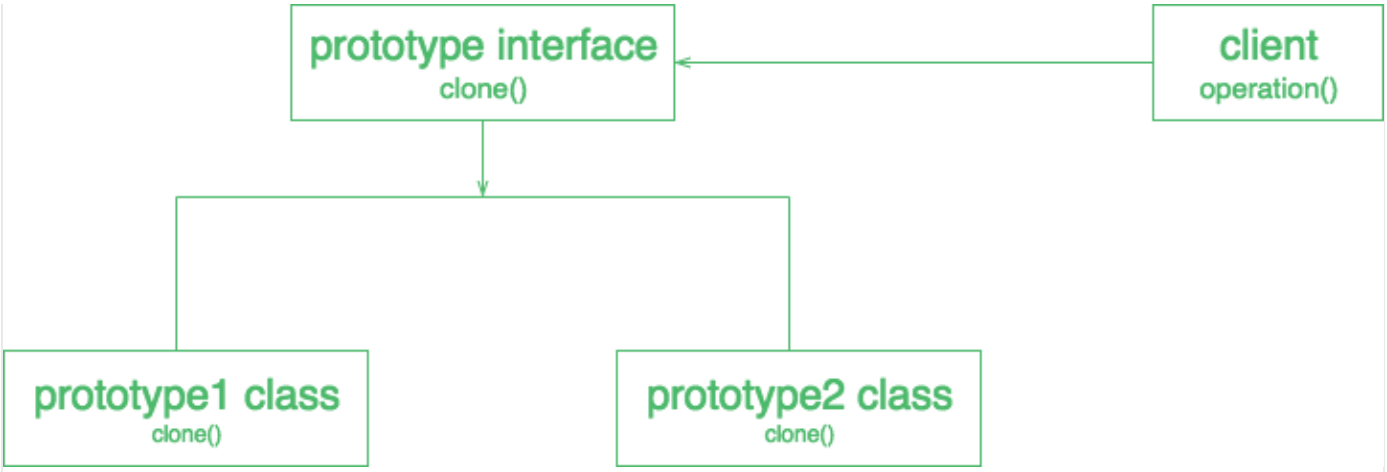

Advantages of Prototype Design Pattern

- **Adding and removing products at run-time –** Prototypes let you incorporate a new concrete product class into a system simply by registering a prototypical instance with the client. That's a bit more flexible than other creational patterns, because a client can install and remove prototypes at run-time.
- **Specifying new objects by varying values –** Highly dynamic systems let you define new behavior through object composition by specifying values for an object's variables and not by defining new classes.
- **Specifying new objects by varying structure –** Many applications build objects from parts and subparts. For convenience, such applications often let you instantiate complex, user-defined structures to use a specific subcircuit again and again.
- **Reduced subclassing –** Factory Method often produces a hierarchy of Creator classes that parallels the product class hierarchy. The Prototype pattern lets you clone a prototype instead of asking a factory method to make a new object. Hence you don't need a Creator class hierarchy at all.

Disadvantages of Prototype Design Pattern

- Overkill for a project that uses very few objects and/or does not have an underlying emphasis on the extension of prototype chains.
- It also hides concrete product classes from the client
- Each subclass of Prototype must implement the clone() operation which may be difficult, when the classes under consideration already exist. Also implementing clone() can be difficult when their internals include objects that don't support copying or have circular references.

This article is contributed by **Saket Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your arti to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page a elp other Geeks.