**4. Prototype**

**Tên, phân loại:**

* Tên chính thức: Prototype Pattern
* Phân loại: Creational.

**Mục đích, ý định:**

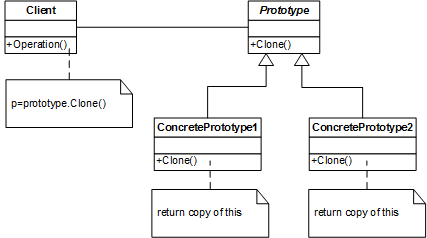
Xác định kiểu của đối tượng sẽ được tạo ra bằng một đối tượng nguyên mẫu – prototype, và tạo ra các đối tượng mới bằng cách sao chép nguyên mẫu này.

**Khi nào sử dụng:**

* Khi chương trình của chúng ta cần sử dụng một lớp tổng quát và chúng ta không cần biết lớp cụ thể nào sẽ được sử dụng cho đến khi chương trình được thực thi.
* Khi một việc tạo một đối tượng mới tốn quá nhiều chi phí (bộ nhớ, thời gian). Ta tạo ra một đối tượng mới bằng cách sao chép từ prototype tương ứng.
* Khi hệ thống cần độc lập với các sản phẩm mà nó tạo ra, kết hợp và thể hiện chúng.
* Khi ta muốn tránh việc tạo ra cây phân cấp các Factory.
* Khi việc sao chép một đối tượng dễ dàng thực hiện hơn là tạo ra một đối tượng mới.

**Khả năng ứng dụng:**

**Cấu trúc:**

****

**Các thành viên:**

* **Prototype**: Là một giao diện với một hàm Clone() để các lớp hiện thực nó định nghĩa việc sao chép các đối tượng thuộc lớp đó.
* **ConcretePrototype**: Các lớp hiện thực giao diện **Prototype** để thực hiện việc sao chép các đối tượng được tạo ra từ chúng.
* **Client** – nơi sử dụng giao diện Prototype.

**Sự cộng tác:**

* Quá trình sao chép bắt đầu với việc khởi tạo một lớp.
* Client yêu cầu một đối tượng mới của kiểu nào đó và gửi yêu cầu đến lớp Prototype.
* Một ConcretePrototype, tùy thuộc vào kiểu đối tượng mà Client cần, sẽ đảm nhận việc sao chép phương thức clone() mà nó hiện thực khi thực thi giao diện Prototype. Khi đó, một đối tượng mới thuộc kiểu mà Client cần sẽ được tạo ra.

**Các hệ quả mang lại:**

* Cũng như các mẫu Khởi tạo khác, mẫu Prototype nâng cao khả năng đóng gói của chương trình, client không hề biết chi tiết cách thức đối tượng được tạo ra một cách chi tiết.

**Chú ý liên quan đến việc cài đặt:**

* Trong C#, việc sao chép một đối tượng có thể diễn ra theo 2 cách: shallow copy và deep copy. Cần chú ý đến vấn đề này để có không gặp trục trặc trong quá trình sao chép một đối tượng.

**Ví dụ về các hệ thống thực tế:**

* Trong C#, giao diện IClonable cho phép các lớp nào hiện thực nó khả năng tự sao chép bản thân theo kiểu deep copy hoặc theo kiểu shallow copy.
* Hàm MemberwiseClone() của lớp Object trong C# cũng có chức năng tương tự.

**Các mẫu liên quan:**

**Mã nguồn minh họa:**

* **Bài toán:**
* **Sơ đồ lớp:**
* **Code mẫu:**