**Nhóm 3**

Nguyễn Huỳnh Minh Tiến – 18110377

Nguyễn Thị Minh Hoàng – 18110285

Thòng Chủ Thành – 18110365

Nguyễn Quyết Sinh – 18110354

**PROTOTYPE PATTERN**

**1. Giới thiệu mẫu thiết kế**

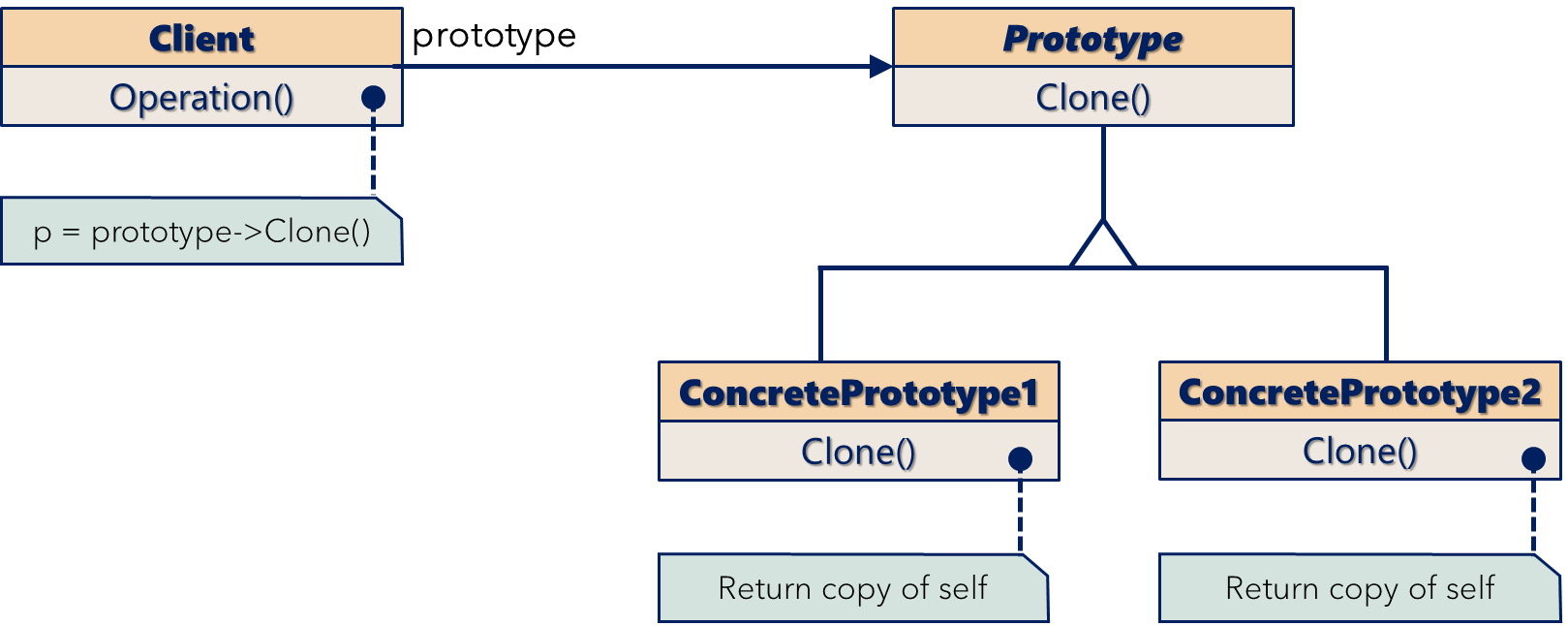
* Prototype Pattern là một trong những pattern phổ biến trong lập trình hướng đối tượng, là một pattern thuộc nhóm Creational Patterns.
* Nó có nhiệm vụ khởi tạo một đối tượng bằng cách clone một đối tượng đã tồn tại thay vì khởi tạo với từ khoá new. Đối tượng mới là một bản sao có thể giống 100% với đối tượng gốc, chúng ta có thể thay đổi dữ liệu của nó mà không ảnh hưởng đến đối tượng gốc.

**2. Mục đích sử dụng của mẫu thiết kế.**

Prototype Pattern được dùng khi việc tạo một object tốn nhiều chi phí và thời gian trong khi bạn đã có một object tương tự tồn tại.

Prototype ẩn đi những lớp con rời rạc từ phía client, do vậy làm giảm đi số lớp con mà client cần biết. Hơn thế nữa pattern này làm cho client hoạt động với những lớp con cụ thể mà không phải thay đổi gì.

**3. Vẽ và giải thích cấu trúc của mẫu thiết kế (cấu trúc chung, các lớp/đối tượng tham gia, giao tiếp giữa các đối tượng, mục đích của các phương thức, thuộc tính).**



* Prototype pattern gồm 3 phần:
  + Client class: tạo mới Object bằng cách gọi Prototype thực hiện clone chính nó.
  + Prototype: khai báo một Interface hoặc Abtract class cho việc clone chính nó.
  + ConcretePrototype: triển khai những Operation cho việc clone chính nó.
* Quá trình Clone():
  + Bắt đầu với việc khởi tạo class sẽ lấy làm mẫu.
  + Client class yêu cầu một object mới của loại đó và gửi yêu cầu tới class Prototype.
  + Một ConcretePrototype (phụ thuộc vào loại object mà Client cần) đảm nhận sao chép object client thông qua phương thức Clone(), khi đó một object mới thuộc loại Client sẽ được tạo ra.

**4. Những trường hợp sử dụng của mẫu.**

* Khi muốn tạo đối tượng run-time.
* Khi muốn truyền đối tượng vào một hàm nào đó để xử lý, thay vì truyền đối tượng gốc có thể ảnh hưởng dữ liệu thì ta có thể truyền đối tượng được sao chép (được clone).
* Khi chi phí của việc tạo mới đối tượng (bằng cách sử dụng toán tử new) là quá lớn.
* Cải thiện performance: giảm chi phí để tạo ra một đối tượng mới theo chuẩn, điều này sẽ làm tăng hiệu suất so với việc sử dụng từ khóa new để tạo đối tượng mới.
* Giảm độ phức tạp cho việc khởi tạo đối tượng: do mỗi lớp chỉ implement cách clone của chính nó.
* Giảm việc phân lớp, tránh việc tạo nhiều lớp con cho việc khởi tạo đối tượng như của Abstract Factory Pattern.
* Khởi tạo object mới bằng cách thay đổi một vài thuộc tính của object (các object có ít điểm khác biệt nhau): Một hệ thống linh động sẽ để cho chúng ta tự định nghĩa một hành động nào đó thông qua sự kết hợp với một object (nghĩa là một phương thức của một class) hơn là định nghĩa một class mới.
* Khởi tạo object mới bằng cách thay đổi cấu trúc: Rất nhiều ứng dụng xây dựng hệ thống từ nhiều phần và các phần con. Các phần con lại khởi tạo từ nhiều phần con khác (chia nhỏ bài toán). Prototype pattern cũng hỗ trợ điều này. Nghĩa là các phần đó có thể được khởi tạo từ việc copy một nguyên mẫu từ một “cấu trúc” khác. Miễn là các phần kết hợp đều thể hiện clone() và được sử dụng với cấu trúc khác nhau làm nguyên mẫu.

**5. Ví dụ minh họa của mẫu (có mã nguồn kèm theo).**

**Text

Description automatically generated**

**Text

Description automatically generated**

6. Nêu ý nghĩa của mẫu thiết kế (Gợi ý: So sánh giữa hai trường hợp có sử dụng và không sử dụng mẫu).

|  |  |
| --- | --- |
| **SỬ DỤNG MẪU PROTOTYPE** | **KHÔNG SỬ DỤNG MẪU PROTOTYPE** |
| Rút ngắn quá trình nhân bản các đối tượng có giá trị thuộc tính gần như nhau  🡪 Tiết kiệm thời gian cho các lập trình viên | Đổ dữ liệu lần lượt lại từ đầu gây ra sự rườm rà trong lúc ta tạo mới nếu đã dữ liệu mẫu |
| Nếu sử dụng hiệu quả mẫu thiết kế sẽ tạo ra được các đối tượng có giá trị giống nhau nhưng không có tình trạng tham chiếu | Gán đối tượng mới bằng nguyên đối tượng sẵn có có thể gây ra tình trạng tham chiếu giữa các đối tượng |
| Giám sát được các mẫu dữ liệu chung để sử dụng trong ứng dụng | Khó khăn trong việc xử lý các mẫu dữ liệu chung |

7. Mối quan hệ với các mẫu thiết kế khác (nếu có)

- Nhiều thiết kế bắt đầu bằng cách sử dụng Factory Method và phát triển theo hướng Abstract Factory, Prototype hoặc Builder

- Abstract Factory dựa trên một tập hợp các Factory Methods, nhưng chúng ta cũng có thể sử dụng Prototype để compose các phương thức trên các lớp này.

- Các thiết kế sử dụng nhiều Composite và Decorator thường có thể được hưởng lợi từ việc sử dụng Prototype. Áp dụng mẫu cho phép bạn sao chép các cấu trúc phức tạp thay vì xây dựng lại chúng từ đầu.

- Abstract Factories, Builders và Prototypes đều có thể được triển khai dưới dạng các Singleton.

- Prototype không dựa trên tính kế thừa, vì vậy nó không có nhược điểm. Mặt khác, Prototype yêu cầu khởi tạo đối tượng nhân bản phức tạp. Factory Method dựa trên kế thừa nhưng không yêu cầu bước khởi tạo.

Tham khảo

<https://www.stdio.vn/software-architecture/design-pattern-prototype-pattern-Fhru1>

<https://topdev.vn/blog/prototype-pattern-mot-trong-nhung-pattern-pho-bien-nhat/>

<https://viblo.asia/p/design-patterns-prototype-gGJ59XqxlX2>