6. Nêu ý nghĩa của mẫu thiết kế

-Khi không sử dụng mẫu PROTOTYPE

* Mỗi lần giả sử chúng ta muốn tạo ra một đối tượng có giá trị gần giống với các đối tượng trước đây sẵn có thì ta phải đổ dữ liệu lần lượt lại từ đầu gây ra sự rườm rà trong lúc ta tạo mới
* Hoặc nếu ta gán đối tượng mới bằng nguyên đối tượng sẵn có có thể gây ra tình trạng tham chiếu nghĩa là đối tượng một trong hai đối tượng nếu có thay đối thì sẽ dẫn đến đối tượng kia thay đổi theo.

-Khi sử dụng mẫu PROTOTYPE

* Rút ngắn được quá trình nhân bản các đối tượng có giá trị thuộc tính gần như nhau, tiết kiệm thời gian cho các lập trình viên
* Giám sát được các mẫu dữ liệu chung để sử dụng trong ứng dụng
* 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

Và diễn nhiên với khi sử dụng và không sử dụng mẫu thì chúng đề có ý nghĩa riêng thực hiện các mục đích khác nhau, không thực sự thay thế cho nhau.

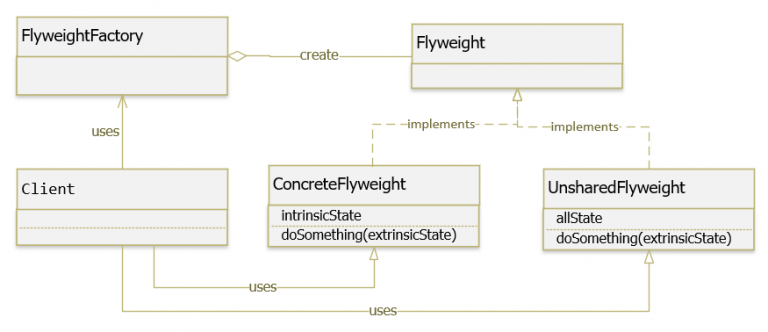
7. Mối quan hệ với các mẫu thiết kế khác (nếu có)

Flyweight Pattern là một trong những Pattern thuộc nhóm cấu trúc (Structural Pattern). Nó cho phép tái sử dụng đối tượng tương tự đã tồn tại bằng cách lưu trữ chúng hoặc tạo đối tượng mới khi không tìm thấy đối tượng phù hợp.

Trong một số tình huống trong phát triển phần mềm, chúng ta có thể cải thiện hiệu suất của ứng dụng với việc sử dụng Cache.

Hãy tưởng tượng rất nhiều đối tượng được tạo ra và lãng phí bộ nhớ. Mô hình Flyweight được tạo ra để tránh vấn đề này và tối ưu hóa hiệu suất.

Flyweight Pattern được sử dụng khi chúng ta cần tạo một số lượng lớn các đối tượng của 1 lớp nào đó



* **Flyweight** : là một interface/ abstract class, định nghĩa các các thành phần của một đối tượng.
* **ConcreteFlyweight** : triển khai các phương thức đã được định nghĩa trong Flyweight. Việc triển khai này phải thực hiện các khả năng của trạng thái nội tại. Đó là dữ liệu phải không thể thay đổi (unchangeable) và có thể chia sẻ (shareable). Các đối tượng là phi trạng thái (stateless) trong triển khai này. Vì vậy, đối tượng ConcreteFlyweight giống nhau có thể được sử dụng trong các ngữ cảnh khác nhau.
* **UnsharedFlyweight** : mặc dù mẫu thiết kế Flyweight cho phép chia sẻ thông tin, nhưng có thể tạo ra các thể hiện không được chia sẻ (not shared). Trong những trường hợp này, thông tin của các đối tượng có thể là stateful.
* **FlyweightFactory** (Cache): lớp này có thể là một Factory Pattern được sử dụng để giữ tham chiếu đến đối tượng Flyweight đã được tạo ra. Nó cung cấp một phương thức để truy cập đối tượng Flyweight được chia sẽ. FlyweightFactory bao gồm một Pool (có thể là HashMap, không cho phép bên ngoài truy cập vào) để lưu trữ đối tượng Flyweight trong bộ nhớ. Nó sẽ trả về đối tượng Flyweight đã tồn tại khi được yêu cầu từ Client hoặc tạo mới nếu không tồn tại.
* **Client** : sử dụng FlyweightFactory để khởi tạo đối tượng Flyweight