Tìm hiểu về Flyweight Pattern

1. Flyweight Pattern là gì?

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.

Flyweigth Pattern được sử dụng khi chúng ta cần tạo một số lượng lớn các đối tượng của một lớp nào đó. Do mỗi đối tượng đều đòi hỏi chiếm giữ một khoảng không gian bộ nhớ, nên với một số lượng lớn các đối tượng được tạo ra có thể gây nên vấn đề nghiêm trọng đặc biệt đối với các thiết bị có dung lượng nhớ thấp. Flyweight Pattern có thể được áp dụng để giảm tải cho bộ nhớ thông qua cách chia sẻ đối tượng. Vì vậy performance của hệ thông được tối ưu

Flyweight object là immutable, nghĩa là không thể thay đổi khi nó đã được khởi tạo.

2. Hai trạng thái của Flyweight Object

Trạng thái của flyweight object là một phần quan trọng trong việc thiết kế Flyweight Pattern. Mục tiêu chính của Flyweight Pattern là giảm bộ nhớ bằng cách chia sẻ các đối tượng. Điều này có thể đạt được bằng cách tách các thuộc tính của đối tượng thành hai trạng thái: độc lập và phụ thuộc. Hay còn gọi là Intrinsic (trạng thái nội tại) và Extrinsic (trạng thái bên ngoài).

* Intrnsic State (trạng thái nội tại): Trạng thái này chứa dữ liệu không thể thay đổi (unchangeable) và không phụ thuộc (independent) vào ngữ cảnh (context) của đối tượng Flyweight. Những dữ liệu đó có thể được lưu trữ vĩnh viễn bên trong đối tượng Flyweight. Vì vậy mà Flyweight object có thể chia sẻ. Dữ liệu nội tại là phi trạng thái (stateless) và thường không thay đổi (unchanged). Tính năng này cho phép khả năng tái tạo các thuộc tính của đối tượng Flyweight giữa các đối tượng tương tự khác. Điều quan trọng cần lưu ý là các đối tượng Flyweight chỉ nên nhận trạng thái bên trong của chúng thông qua các tham số của hàm tạo và không cung cấp các phương thức setter hay các biến public.
* Extrinsic State (trạng thái bên ngoài): Trạng thái bên ngoài thể hiện tính chất phụ thuộc ngữ cảnh của đối tượng flyweight. Trạng thái này chứa các thuộc tính và dữ liệu được áp dụng hoặc được tính toán trong thời gian thực thi (runtime). Do đó, những dữ liệu đó không được lưu trữ trong bộ nhớ. Vì trạng thái bên ngoài là phụ thuộc ngữ cảnh và có thể thay đổi nên các đối tượng đó không thể được chia sẻ. Do đó, client chịu trách nhiệm truyền dữ liệu liên quan đến trạng thái bên ngoài cho đối tượng flyweight khi cần thiết, có thể thông qua các tham số.

3. Cài đặt Flywight Pattern như nào?

Các thành phần cơ bản trong Flyweight:

* Flyweight: là một interface/abstrac class, định nghĩa các thành phần của một đối tượng.
* ConcreteFlyweight: triển khai các phương thức đã định nghĩa sẵn 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 sửa thông tin, nhưng có thể tạo ra cá thể hiện không được chia sẻ (not shared). Trong những trường hợp này, thông tin cảu 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 tường Flyweight đã được tạo ra. Nó cung cấp một phương thức để truy cập đối tượng Flyweight được chia sẻ. 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 tự động đối tượng Flyweight đã tồn tại đã được yêu cầu từ Client hoặc tạo mới nếu không tồn tại.
* Client: sử dụng FlyweightFactory để khởi tạo đối tượng Flyweight.

4. Lợi ích của Flyweight Pattern

* Giảm số lượng đối tượng tạo ra bằng cách chia sẻ đối tượng. Vì vậy, tiết kiệm bộ nhớ và các thiết bị lưu trữ cần thiết
* Cải thiện khả năng cache dữ liệu vì thời gian đáp ứng nhanh
* Tăng performance

5. Sử dụng Flyweight Pattern khi nào

* Khi có một số lớn các đối tượng được ứng dụng tạo ra một cách lặp đi lặp lại
* Khi việc tạo ra đối tượng đòi hỏi nhiều bộ nhớ và thời gian
* Khi muốn tái sử dụng đối tượng đã tồn tại thay vì tốn thêm thời gian để tạo mới
* Khi muốn nhóm đối tượng chứa nhiều đối tượng tương tự và hai đối tượng trong nhóm không khác nhau nhiều