1. **Tên và phân loại:**

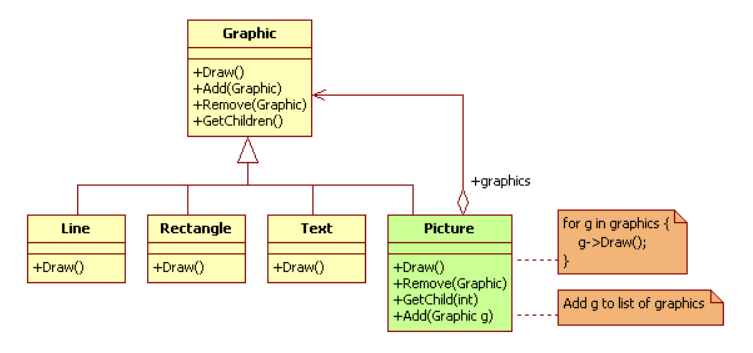
* **Tên:** Composite Pattern.
* **Phân loại:** Structural Pattern.

1. **Mục đích, ý nghĩa:**

* Kết hợp (compose) các đối tượng thành một cấu trúc hình cây để biểu diễn một cây phân cấp một phần – toàn bộ (part-whole hierarchies).
* Mẫu Composites cho phép client xử lý một đối tượng riêng hoặc toàn bộ đối tượng theo cùng một cách.

1. **Động lực sử dụng:**

* Giả sử chúng ta có một ứng dụng đồ họa, biểu diễn cho một đối tượng đồ họa là lớp Graphic.
* Khi đó, một Graphic có thể là:
  + Một Line**;**
  + Một Rectangle;
  + Một Text Box;
  + Hoặc là bất cứ sự kết hợp nào của các thành phần trên,…
* Sử dụng mẫu Composite, chúng ta có thể quản lý các các đối tượng graphic trên theo một cách như nhau.



1. **Khi nào sử dụng:**

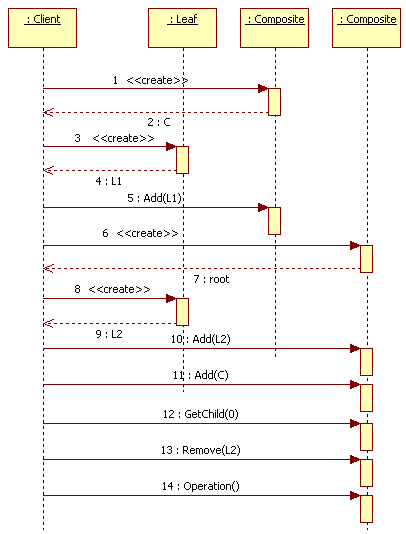
* Khi chúng ta muốn biểu diễn một cây phân cấp một phần-toàn bộ (part-whole hierarchies)của các đối tượng
* Khi chúng ta các client có khả năng bỏ qua các khác biệt giữa một sự kết hợp của các đối tượng và các đối tượng riêng biệt. Client sẽ đối xử chúng (các đối tượng trong kiến trúc kết hợp) theo cùng một cách như nhau.

1. **Cấu trúc**



1. **Các thành viên - Sự cộng tác**

* **Component**
  + Định nghĩa một giao diện cho các đối tượng trong composition.
  + Cài đặt các hành vi mặc định cho giao diện, các hành vi chung của các lớp một cách phù hợp.
  + Khai báo một giao diện cho việc truy cập và quản lý các thành phần con sẽ tham gia vào tập hợp.
  + Định nghĩa một giao diện cho việc truy cập các đối tượng cha của các thành phần theo một cấu trúc đệ quy và cài đặt nó nếu phù hợp.
* **Leaf:**
  + Đại diện cho một đối tượng lá là trong composition.
  + Lá là đối tượng cơ bản nhất, nó không chứa các đối tượng khác.
  + Định nghĩa các hành vi cho các đối tượng nguyên thuỷ trong composition.
* **Composite:**
  + Định nghĩa hành vi cho các thành phần có chứa các thành phần con.
  + Lưu trữ các thành phần con.
  + Hiện thực các hành vi liên quan đến các thành phần con được định nghĩa trong giao diện Component.
* **Client:**
  + Điều khiển các đối tượng trong composition thông qua giao diện Component.



1. **Các hệ quả mang lại:**

* Định nghĩa một cây phân cấp lớp bao gồm các đối tượng cơ bản và các đối tượng kết hợp.
  + Các đối tượng cơ bản có thể được gộp vào các đối tượng kết hợp khác phức tạp hơn, đến lượt các đối tượng kết hợp này lại được gộp lại, và cứ thế tiếp tục một cách đệ quy.
  + Bất cứ nơi nào code của client mong muốn một đối tượng cơ bản, nó cũng có thể chấp nhận một đối tượng kết hợp.
* Làm cho client đơn giản hơn:
  + Client có thể đối xử với các cấu trúc kết hợp và các đối tượng riêng biệt như nhau, client không quan tâm liệu họ có đang làm việc với đối tượng lá hay là một đối tượng kết hợp.
  + Điều này làm đơn giản mã nguồn phía client, bởi vì nó ngăn việc việc phải viết các hàm theo kiểu tag-và-câu-lệnh-case trong các lớp định nghĩa kết hợp.
* Dễ dàng thêm các kiểu thành phần mới
  + Các lớp con mới định nghĩa Composite hay Leaf sẽ tự động hoạt động với các cấu trúc và mã client có sẵn.
  + Client không cần phải thay đổi để có thể sử dụng các lớp Component mới.
* Làm cho thiết kết trở nên chung chung:
  + Quá dễ dàng thêm các thành phần mới làm cho chúng ta khó quản lí chặc chẽ được các component của một đối tượng kết hợp. Đôi khi chúng ta muốn đối tượng kết hợp chỉ chứa một kiểu component nhất định. Với composite, chúng ta không thể phụ thuộc vào hệ thống kiểu để chắc chắn thực thi các ràng buộc cho bạn. Thay vào đó, bạn cần phải sử dụng các kiểm tra vào thời điểm run-time.

1. **Chú ý liên quan đến cài đặt:**
2. **Hệ thống thực tế**

* Các hệ điều hành Linux sử dụng cấu trúc Composite để lưu trữ thông tin tập tin và thư mục.

1. **Mẫu liên quan**

* Thông thường liên kết “component”-“cha của component” được sử dụng trong mẫu Chain of Reponsibility.
* Mẫu Decorator cũng thường được sử dụng với Composite. Khi Decoratior và Composite cùng được sử dụng cùng nhau, chúng thường sẽ có một lớp cha chung. Do đó Decorator sẽ hỗ trợ giao diện Component với các thao tác như Add, Remove và GetChild.
* Flyweight để cho chúng ta chia sẻ các thành phần, nhưng chúng sẽ không còn tham chiếu đến cha của chúng.
* Iterator có thể được sử dụng để duyệt qua đối tượng composites.
* Visitor định vị các thao tác và các hành vi nào sẽ được phân phối trên lớp Composite lớp Leaf.