**Tên, phân loại:**

* Tên đầy đủ: Abstract Factory Pattern.
* Bí danh: Factory of Factories, Super Factory.
* Phân loại: Nhóm Khởi Tạo (Creational)

**Mục đích, ý định:**

* Cung cấp một interface có chức năng tạo ra các đối tượng có liên quan hoặc có phụ thuộc lẫn nhau mà không chỉ ra đó là những lớp cụ thể nào tại thời điểm khởi tạo chúng.
* Mẫu thiết kế Abstract Factory đóng gói một nhóm những lớp đóng vai trò "Người sản xuất" (Factory) trong ứng dụng, những lớp này sẽ được dùng để tạo ra các đối tượng. Các lớp sản xuất này có chung một giao diện được kế thừa từ một lớp cha thuần ảo gọi là "lớp sản xuất ảo".

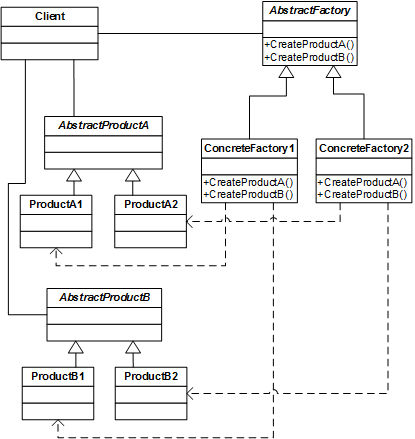
**Khi nào sử dụng:**

* Khi ta cần tạo ra các đối tượng mà đôi khi kiểu của chúng không xác định được trong thời điểm ta sử dụng chúng, khi đó nhờ vào kiểu cơ sở hay giao diện chung của các đối tượng đó và sử dụng mẫu này để các factory tự tạo ra các đối tượng phù hợp với yêu cầu.
* Hoạt động bên phía hệ thống bên phía client cần độc lập – (hay không quan tâm) cách thức các đối tượng được tạo ra.
* Khi hệ thống cần cấu hình để sử dụng các đối tượng thuộc cùng một cây phân cấp, hoặc sử dụng chúng một cách đồng thời.
* Đóng gói các thư viện cho phần mềm, che giấu cách thức thực thi bên trong.

**Khả năng ứng dụng:**

* Phía client sẽ không phụ thuộc vào việc những sản phẩm được tạo ra như thế nào.
* Ứng dụng sẽ được cấu hình với một hoặc nhiều họ sản phẩm.
* Các đối tượng cần phải được tạo ra như một tập hợp để có thể tương thích với nhau
* Chúng ta muốn cung cấp một tập các lớp và chúng ta muốn thể hiện các ràng buộc, các mối quan hệ giữa chúng mà không phải là các thực thi của chúng (interface). Trong trường hợp không định nghĩa được các lớp trừu tượng Product thì việc tạo ra các ConcreteProduct theo mẫu này là rất khó hoặc không thể.
* Mỗi khi có thêm một "sản phẩm" ta lại phải định nghĩa thêm một lớp "sản xuất"

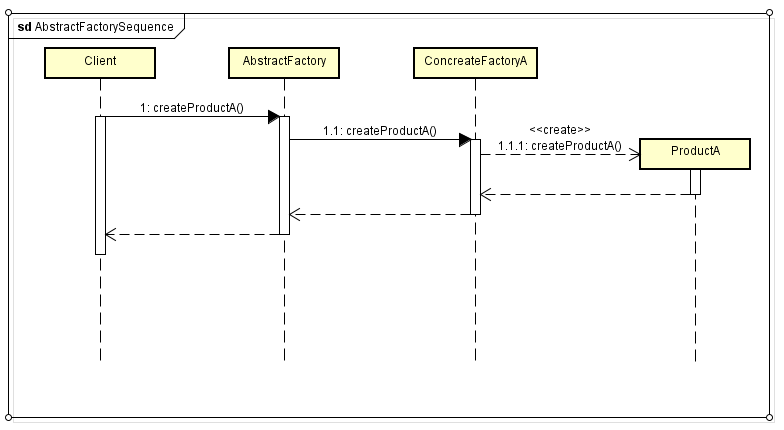
**Cấu trúc:**



**Các thành viên:**

* **AbstractFactory**: Khai báo dạng interface hoặc abstract class chứa các phương thức chung và đảm bảo tính nhất quán cho các đối tượng Factory kế thừa từ nó.
* **ConcreteFactory**: Xây dựng, cài đặt các phương thức tạo các đối tượng cụ thể
* **AbstractProduct**: Khai báo dạng interface hoặc abstract class để định nghĩa một giao diện chung cho đối tượng kế thừa từ nó, nhờ vào lớp này để xây dựng nên các đối tượng có mối quan hệ với nhau.
* **Product**: Cài đặt của các đối tượng cụ thể, cài đặt các phương thức được quy định tại AbstractProduct.
* **Client**: là đối tượng sử dụng AbstractFactory và các AbstractProduct để tạo ra các Product bằng cách gọi cách gọi các Factory con tương ứng mà không cần quan tâm đến kiểu dữ liệu của chúng.

**Sự cộng tác:**



**Các hệ quả mang lại:**

* Tách rời việc định nghĩa các đối tượng và việc tạo ra các đối tượng đó.
* Các thư viện được xây dựng bằng cách áp dụng mẫu này làm tăng tính đóng gói (encapsulation) che giấu các thực thi bên trong nó.
* Dễ dàng quản lý việc tạo ra các đối tượng có cùng mối quan hệ (cùng kế thừa từ inteface hay asbtract class).
* Mỗi khi có thêm một "sản phẩm" mới cần tạo ta lại phải định nghĩa thêm một lớp "sản xuất“.

**Chú ý liên quan đến việc cài đặt:**

* Lớp AbstractFactory  chỉ khai báo duy nhất 1 interface cho việc khởi tạo sản phẩm (products), còn việc tạo ra các thể hiện là nhiệm vụ của các ConcreteFactory.
* Một ứng dụng thường chỉ cần 1 thể hiện của lớp ConcreteFactory trên một tập sản phẩm. Ta có thể đảm bảo điều này bằng việc dùng mẫu Singleton.
* Đơn giản hóa và cải thiện hiệu suất mã lệnh bằng việc sử dụng mẫu Prototype hoặc Factory Method.

**Ví dụ về các hệ thống thực tế:**

* Trong .NET, các lớp kết nối CSDL dùng để kết nối đến các hệ quản trị cơ sở dữ liệu khác như như SQL Server, Oracle, MySQL..
* Các ứng dụng đa nền tảng: iOS, Android, Window phone, Web,...
* Hệ thống giao diện trên Window và MacOS

**Các mẫu liên quan:**

* Mẫu **Abstract Factory** thường được cài đặt cùng với mẫu [**Singleton**](http://hoclaptrinhviet.com/design-pattern-2-singleton-pattern.html), [**Factory Method**](http://hoclaptrinhviet.com/design-pattern-3-mau-factory-method) đôi khi còn đi chung với mẫu **Prototype**. Các lớp con cụ thể thường được cài đặt bằng mẫu singleton. Bởi vì mẫu singleton có thể tạo các đối tượng nhất quán ta có thể gọi nó ở bất cứ đâu trong chương trình

**Demo**