1. **Link: (Wiki)**

[**https://vi.wikipedia.org/wiki/Abstract\_factory**](https://vi.wikipedia.org/wiki/Abstract_factory)

Abstract Factory (Abstract Factory pattern): là [thiết kế mẫu](https://vi.wikipedia.org/wiki/M%E1%BA%ABu_thi%E1%BA%BFt_k%E1%BA%BF_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) hướng đối tượng trong việc thiết kế [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m) cho [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh)

* Cung cấp một [giao diện lớp](https://vi.wikipedia.org/w/index.php?title=Giao_di%E1%BB%87n_l%E1%BB%9Bp&action=edit&redlink=1) có chức năng tạo ra một [tập hợp](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_h%E1%BB%A3p) các đối tượng liên quan hoặc phụ thuộc lẫn nhau
* Không chỉ ra đó là những [lớp](https://vi.wikipedia.org/wiki/L%E1%BB%9Bp) cụ thể nào tại thời điểm [thiết kế](https://vi.wikipedia.org/wiki/Thi%E1%BA%BFt_k%E1%BA%BF).

Mẫu thiết kế Abstract Factory đóng gói một nhóm những lớp đóng vai trò "sản xuất" (Factory) trong ứng dụng → tạo lập các đối tượng. Các lớp sản xuất này có chung một giao diện lập trình được kế thừa từ một lớp cha thuần ảo gọi là "lớp sản xuất ảo".

Là một giao diện hoặc lớp trừu tượng (interface, abstract).

Phân loại đối tượng dựa trên tham số và thủ tục createObject() còn phụ thuộc thêm vào các yếu tố phụ khác như môi trường hệ điều hành.

1. **Link:**

[**https://vi.fitwp.com/abstract-factory-pattern/**](https://vi.fitwp.com/abstract-factory-pattern/)

1. **Định nghĩa**

Tương tự như Factory Method Pattern, Abstract Factory Pattern cũng là một design pattern thuộc nhóm khởi tạo (creational patterns) tuy nhiên nó cao cấp hơn Factory Method Pattern ở chỗ nó cho phép tạo ra một super factory dùng để tạo ra các factory khác.

Abstract Factory như là một nhà máy lớn chứa nhiều nhà máy nhỏ, trong các nhà máy đó có những xưởng sản xuất, các xưởng đó tạo ra những sản phẩm khác nhau.

1. **Cấu trúc**

Abstract Factory Pattern bao gồm năm thành phần cơ bản là: Abstract Factory, Concrete Factory, Abstract Product, Concrete Product và Client.

* **AbstractFactory:** Khai báo dạng interface hoặc abstract class chứa các phương thức để tạo ra các đối tượng abstract.
* **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 đối tượng abstract.
* **ConcreteProduct:** 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.

1. **Ưu điểm**

* Đảm bảo rằng các product mà bạn nhận được từ một factory đều tương thích với nhau.
* Hạn chế sự phụ thuộc giữa creator và concrete products..
* Gom các đoạn code tạo ra product vào một nơi trong chương trình, nhờ đó giúp dễ theo dõi và thao tác.
* Có thể thoải mái thêm nhiều loại product mới vào chương trình mà không làm thay đổi các đoạn code nền tảng đã có trước đó.

1. **Khuyến điểm:** Code có thể trở nên nhiều hơn và phức tạp hơn do đòi hỏi phải sử dụng nhiều class mới có thể cài đặt được pattern này.