Abstract Factory

Phan Hoàng Minh Quân - 20520713
Trần Gia Bảo – 21521863
Trần Đông Đông - 21521957

Nội dung

1. Tổng quan
   - Tên
   - Phân loại
   - Mục đích, ý nghĩa
2. Motivation
3. Khả năng ứng dụng
4. Đặc điểm
   - Cấu trúc mẫu
   - Ý nghĩa của từng thành viên
   - Sự cộng tác

Nội dung

1. Các hệ quả mang lại
   - Ưu điểm
   - Nhược điểm
2. Các chú ý liên quan đến cài đặt, demo, mã nguồn minh họa
3. Một số so sánh nếu có
4. Các ví dụ thực tế
5. Các mẫu liên quan

Tổng quan

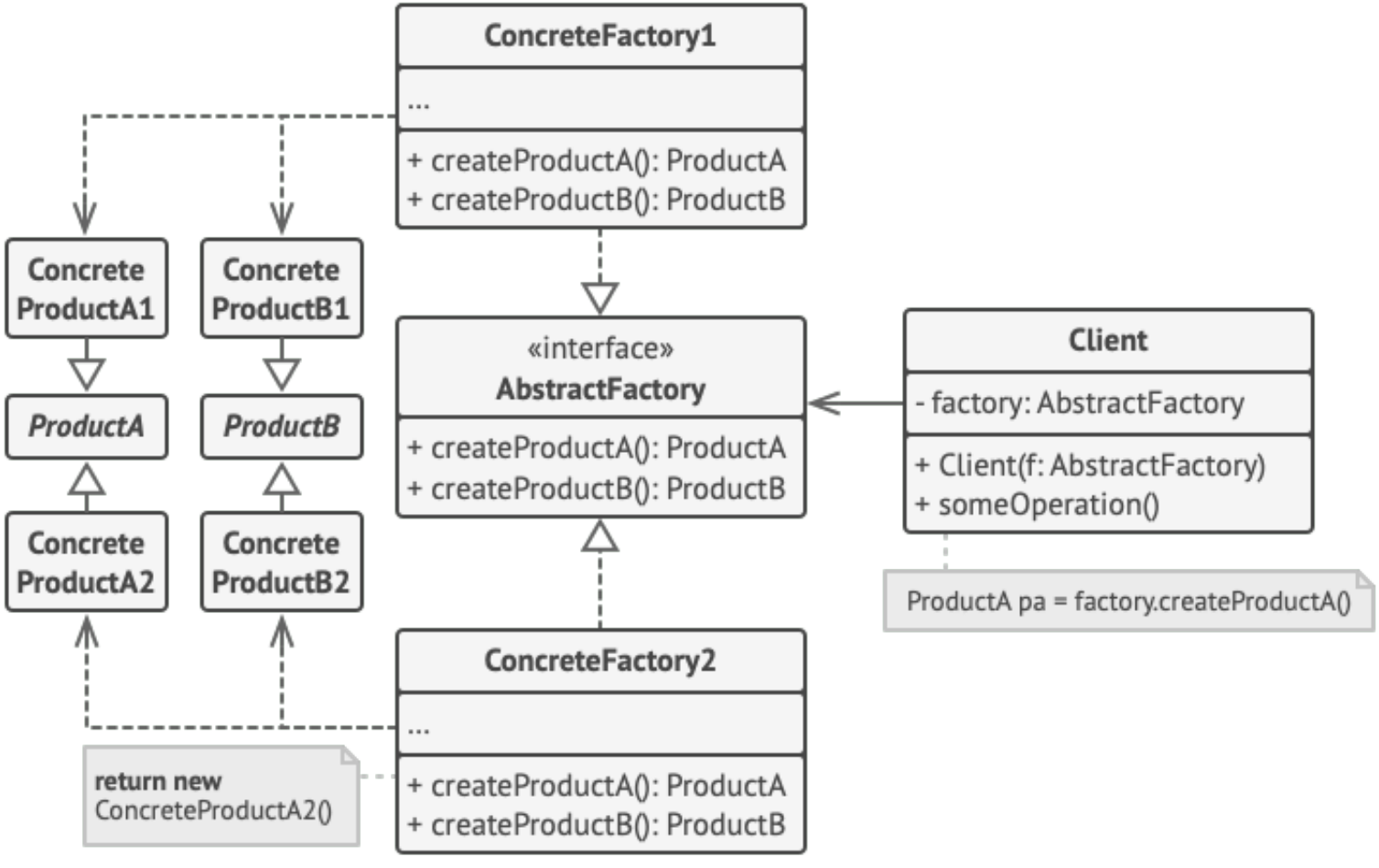
- Tên mẫu: Abstract Factory.
- Phân loại: Thuộc phân nhóm Creational Pattern Design.
- Mục đích: Cung cấp một interface cho việc khởi tạo các tập hợp của những object có đặc điểm giống nhau mà không cần quan tâm object đó là gì

Motivation

- Sử dụng Abstract Facotry khi cần làm việc với các product có tính chất gần giống nhau và liên quan đến nhau và không cần phụ thuộc vào các định nghĩa có sẵn trong class đó:
  - 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.

Đặc điểm

- Cấu trúc mẫu



Đặc điểm

- Ý nghĩa của từng thành viên:
  - Abstract Products: Khai báo các interfaces cho các loại riêng biệt nhưng các product có liên quan sẽ tạo thành một dòng product.
  - Concrete Products: Là các implement khác nhau của abstract product, được nhóm lại bởi các biến thể. Mỗi abstract product (Shoe, Dress) phải được implement trong tất cả các biến thể (Summer, Winter).
  - Abstract Factory: Interface khai báo một tập các phương thức cho việc tạo mỗi abstract product

Đặc điểm

- Ý nghĩa của từng thành viên:
  - Concrete Factories: Implement các phương thức tạo product của abstract factory. Mỗi concrete factory có trách nhiệm đến một biến thể của product và chỉ tạo các product của biến thể này.
  - Client: Mặc dù concrete factories tạo ra concrete products, kiểu trả về của các phương thức tạo product phải trả về abstract products tương ứng. Với cách này client code sử dụng một factory không được kết hợp với một biến thể của product mà nó nhận về một factory. Client có thể làm việc với bất kì concrete factory nào (biến thể product), miễn là nó giao tiếp với objects của chúng thông qua abstract interfaces.

Hệ quả mang lại

- Ưu điểm:
  - Bạn có thể đảm bảo rằng products mà bạn nhận từ factory tương thích với những cái khác.
  - Bạn tránh được việc gắn chặt giữa một concrete products và client code.
  - **Single Responsibility Principle**. Bạn có thể tách code tạo product tới một nơi, giúp code dễ dàng để hỗ trợ
  - **Open/Closed Principle**. Dễ dàng để thêm biến thể sản phẩm mới mà không làm ảnh ảnh hưởng đến code cũ

Hệ quả mang lại

- Nhược điểm:
  - Code có thể trở nên phức tạp hơn mức cần thiết nếu như có nhiều interfaces và classes được thêm vào.

Một số ví dụ thực tế

- Giả sử chúng ta có một nhà máy sản xuất thời trang với nhiều phân xưởng. Cái thì phụ trách sản xuất giày, cái sản xuất váy, cái thì sản xuất mũ,..Yêu cầu đặt ra là chúng ta cần tạo ra nhà máy sao cho nó đáp ứng được hầu hết các thay đổi thị hiếu người dùng và thị trường, như mùa hè thì sản xuất mẫu A, mùa đông lại sản xuất mẫu B cho từng dòng sản phẩm giày, váy, quần áo...Với điều kiện là không phải đầu tư thêm máy móc và nhân công hay sắp xếp lại bộ máy nhân sự vì rất tốn kém và mất nhiều thời gian. Điều này cũng tượng tự như việc thiết kế hệ thống phần mềm, ở bài viết này tôi giả sử phần mềm này là một nhà máy sản xuất thời trang như vậy.
- Với yêu cầu đặt ra như vậy chúng ta cần một cách để tạo ra các dòng sản phẩm một cách riêng biệt. Ví dụ mùa hè thì có giày mùa hè, váy mùa hè và mua đông lại có giày và váy của mùa đông.

Hiện thực/cài đặt

Abstract Factory Implements

Các mẫu liên quan

- Factory Method: Abstract Factory class thường dựa trên một số phương thức Factory, nhưng cũng có thể sử dụng Prototype để kết hợp các phương thức trong Abstract class đó.
- Singleton: Abstract Factory, Factory và Prototype đều được triển khai dưới dạng Singleton.
- Prototype: Abstract Factory thường dựa trên một tập hợp Factory Method, nhưng cũng có thể sử dụng Prototype để soạn các phương thức trên các lớp này.
- Builder: Builder tập trung vào việc xây dựng các đối tượng phức tạp theo từng bước. Abstract Factory chuyên tạo các family đối tượng liên quan. Abstract Factory trả lại product ngay lập tức, trong khi Builder cho phép bạn chạy một số bước xây dựng bổ sung trước khi tìm nạp product.