# Design Pattern

## Khái niệm

- Design pattern là các giải pháp tổng thể đã được tối ưu hóa, được tái sử dụng cho các vấn đề phổ biến trong thiết kế phần mềm mà chúng ta thường gặp phải hàng ngày. Đây là tập các giải pháp đã được suy nghĩ, đã giải quyết trong tình huống cụ thể.

## Lợi ích khi sử dụng

- Giúp sản phẩm của chúng ta linh hoạt, dễ dàng thay đổi và bảo trì hơn.

- Design pattern cung cấp những giải pháp đã được tối ưu hóa, đã được kiểm chứng để giải quyết các vấn đề trong software engineering. Các giải pháp ở dạng tổng quát, giúp tăng tốc độ phát triển phần mềm bằng cách đưa ra các mô hình test, mô hình phát triển đã qua kiểm nghiệm.

## Kiến thức khi dùng déign pattern

- Design Pattern sử dụng nền tảng của lập trình hướng đối tượng nên áp dụng 4 đặc tính của OOP: Kế Thừa, Đa Hình, Trừu Tượng, Bao Đóng.

- Hiểu và áp dụng 2 khái niệm interface và abstract vì nó rất cần thiết.

- Tư duy hoàn toàn theo OOP, loại bỏ tư duy theo lối cấu trúc.

## Phân loại

- Creational: 5 mẫu

* Abstract Factory
* Builder
* Factory
* Prototype
* Singleton

- Structural: 7 mẫu

- Behavioral: 11 mẫu

# Factory Method

## Khái niệm

- Factory Method là một design pattern thuộc nhóm creational. Cung cấp giao diện cho người dùng tạo đối tượng của lớp cha nhưng ủy quyền cho các lớp con để quyết định class nào được trả về.

## Mục đích

- Đưa toàn bộ logic của việc tạo mới object vào trong factory, che giấu logic của việc khởi tạo.

- Giảm sự phụ thuộc nhằm tăng tính mở rộng.

## Các thành phần

Diagram

Description automatically generated

- Creator: Đây là nơi tạo ra factory method. Điểm quan trọng ở đây là kiểu trả về của factory method phải là kiểu Product.

- Product: là interface chung cho tất cả đối tượng có thể tạo ra bởi creator.

- Concrete Product: Là sản phẩm cụ thể sau khi được tạo thông qua factory method.

- Concrete Creator: Ghi đè phương thức gốc để trả về một sản phẩm cụ thể.

## Bài toán

- Viết phần mềm cho trung tâm giống vật nuôi phát giống các bác nông dân dựa vào nơi sống của họ. Biết phần mềm đang áp dụng cho 2 khu vực là Tây nguyên và đồng bằng Sông Cửu Long. Với tiêu chí:

* Tây Nguyên: cho phép chọn bất kỳ giống vật nuôi nào trong danh sách: bò, voi
* ĐBSCL: cho phép chọn tuần tự các giống vật nuôi trong danh sách: gà, vịt, lợn.

- Yêu cầu 1: Tiêu chí chọn lựa thay đổi cho đồng bằng sông Cửu Long, không ảnh hưởng đến logic xử lý tiêu chí lựa chọn của Tây Nguyên và ngược lại.

- Yêu cầu 2: Trong trường hợp khách hàng muốn mở rộng thị trường ra nhiều khu vực khác như: Đồng bằng sông Hồng, vùng núi Đông Bắc, vùng ngập mặn ven biển, … Phần mềm có thể mở rộng mà không ảnh hưởng đến logic của những khu vực trước đó.

## Cách giải quyết mà không dùng pattern

- Tạo 2 class vùng là: TayNguyen, DBSCL.

- Tạo 5 class vật nuôi: bo, voi, ga, vit, lon.

Diagram

Description automatically generated

- Vấn đề: Ta thấy bò, voi, gà, vịt, lợn đều là giống vật nuôi nên sẽ có nhiều thuộc tính và phương thức giống nhau. Tây Nguyên và ĐBSCL là các vùng nên cũng sẽ có nhiều thuộc tính và phương thức giống nhau.

## Cách giải quyết dùng factory method

- Tạo interface GiongVatNuoiFactory chứa phương thức chonGiong() để giúp khách hàng chọn giống phù hợp

- TayNguyen và DBSCL là hai lớp con của GiongVatNuoiFactory. Do đó, hai lớp này có thể ghi đè lại phương thức chonGiong() cho phù hợp

- Tạo interface Giong để tổng quát cho các giống vật nuôi

- Tạo 5 lớp con: ga, vit, lon, bo, voi kế thừa Giong

- Yêu cầu 1: Dễ dàng đáp ứng vì khi muốn thay đổi tiêu chí cho từng vùng thì chỉ cần thay đổi phương thức chonGiong() trong mỗi vùng đó.

- Yêu cầu 2: Khi muốn mở rộng ra các khu vực thì chỉ cần tạo thêm các class như DBSH, VNTB,… kế thừa GiongVatNuoiFactory. Nếu xuất hiện giống vật nuôi khác ví dụ như trâu, dê,… thì tạo thêm các lớp trau, de kế thừa lớp Giong.

Diagram

Description automatically generated

## Ưu điểm và nhược điểm

- Ưu điểm:

* Giúp hạn chế sự phụ thuộc giữa creator và concrete products.
* Dễ dàng mở rộng, thêm những đoạn code mới vào chương trình mà không cần phá vỡ các đối tượng ban đầu
* Giúp gom các đoạn code tạo ra product vào một nơi trong chương trình, nhờ đó giúp dễ theo dõi và thao tác

- Nhược điểm:

* Việc làm trình sẽ trở nên phức tạp khi có quá nhiều lớp con để triển khai pattern

# Tài liệu tham khảo

[1]: <https://codelearn.io/sharing/factory-method-pattern-trong-java>

[2]:<https://sourcemaking.com/design_patterns/factory_method/cpp/1#:~:text=A%20factory%20method%20is%20a,return%20existing%20instances%20multiple%20times>.