**Lab 1: Tìm hiểu và cài đặt nhóm mẫu Creation**

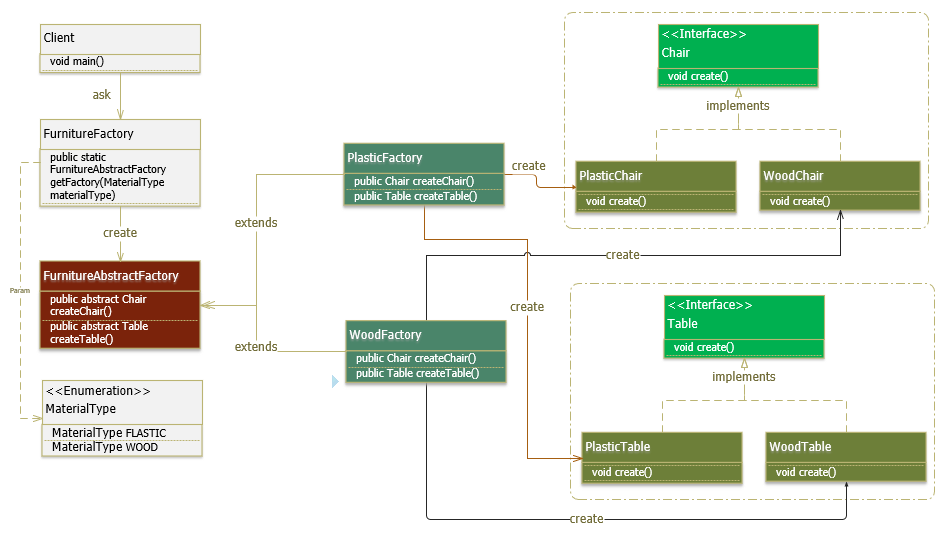
1. **Abstract Factory**

Là một trong những **Creational pattern**. Nó là phương pháp tạo ra một Super-factory dùng để tạo ra các Factory khác. Hay còn được gọi là Factory của các Factory. Abstract Factory Pattern là một Pattern cấp cao hơn so với Factory Method Pattern.

Một Abstract Factory Pattern bao gồm các thành phần cơ bản sau:

* 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.
* 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.

**Ví dụ**: Một công ty đồ nội thất chuyên sản xuất ghế (Chair): ghế nhựa (PlasticChair) và ghế gỗ (WoodChair). Với tình hình kinh doanh ngày càng thuận thợi nên công ty quyết định mở rộng thêm sản xuất bàn (Table). Với lợi thế là đã có kinh nghiệm từ sản xuất ghế nên công ty vẫn giữ chất liệu là nhựa (PlasticTable) và gỗ (WoodTable) cho sản xuất bàn. Tuy nhiên, quy trình sản xuất ghế/ bàn theo từng chất liệu (MaterialType) là khác nhau. Nên công ty tách ra là nhà máy (Factory): 1 cho sản xuất vật liệu bằng nhựa (PlasticFactory), 1 cho sản xuất vật liệu bằng gỗ (WoodFactory), nhưng cả 2 đều có thể sản xuất ghế và bàn (FunitureAbstractFactory). Khi khách hàng cần mua một món đồ nào, khách hàng (Client) chỉ cần đến cửa hàng để mua (FunitureFactory). Khi đó ứng với từng hàng hóa và vật liệu sẽ được chuyển về phân xưởng tương ứng để sản xuất (createXXX) ra bàn (Table) và ghế (Chair).



Hình 1: Minh họa hệ thống áp dụng Abstract Factory.

1. **Builder**.

**Builder pattern** là một trong những **Creational pattern**. Builder pattern là mẫu thiết kế đối tượng được tạo ra để xây dựng một đôi tượng phức tạp bằng cách sử dụng các đối tượng đơn giản và sử dụng tiếp cận từng bước, việc xây dựng các đối tượng đôc lập với các đối tượng khác.

Một builder gồm các thành phần cơ bản sau:

Product: đại diện cho đối tượng cần tạo, đối tượng này phức tạp, có nhiều thuộc tính.

Builder: là abstract class hoặc interface khai báo phương thức tạo đối tượng.

ConcreteBuilder: kế thừa Builder và cài đặt chi tiết cách tạo ra đối tượng. Nó sẽ xác định và nắm giữ các thể hiện mà nó tạo ra, đồng thời nó cũng cung cấp phương thức để trả các các thể hiện mà nó đã tạo ra trước đó.

Director/ Client: là nơi sẽ gọi tới Builder để tạo ra đối tượng.

1. **Factory** **Method**.

là một trong những Pattern thuộc nhóm **Creational Design Pattern**. Nhiệm vụ của Factory Pattern là quản lý và trả về các đối tượng theo yêu cầu, giúp cho việc khởi tạo đổi tượng một cách linh hoạt hơn.

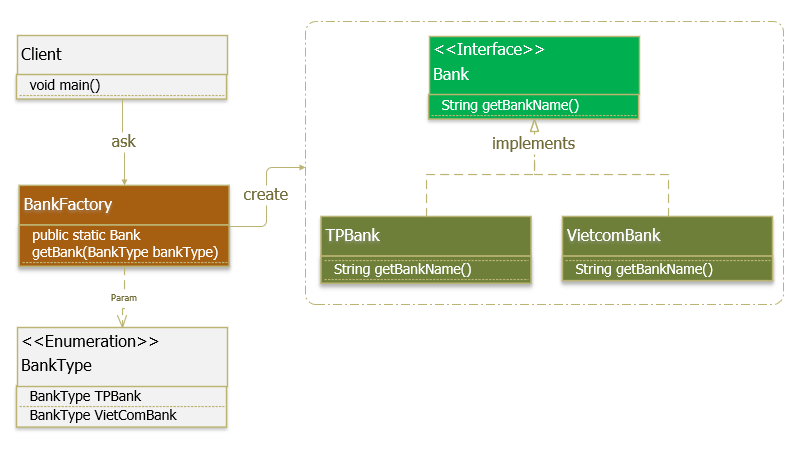
Một Factory Pattern bao gồm các thành phần cơ bản sau:

Super Class: môt supper class trong Factory Pattern có thể là một interface, abstract class hay một class thông thường.

Sub Classes: các sub class sẽ implement các phương thức của supper class theo nghiệp vụ riêng của nó.

Factory Class: một class chịu tránh nhiệm khởi tạo các đối tượng sub class dựa theo tham số đầu vào. Lưu ý: lớp này là [Singleton](https://gpcoder.com/4190-huong-dan-java-design-pattern-singleton/)hoặc cung cấp một public static method cho việc truy xuất và khởi tạo đối tượng. Factory class sử dụng if-else hoặc switch-case để xác định class con đầu ra.

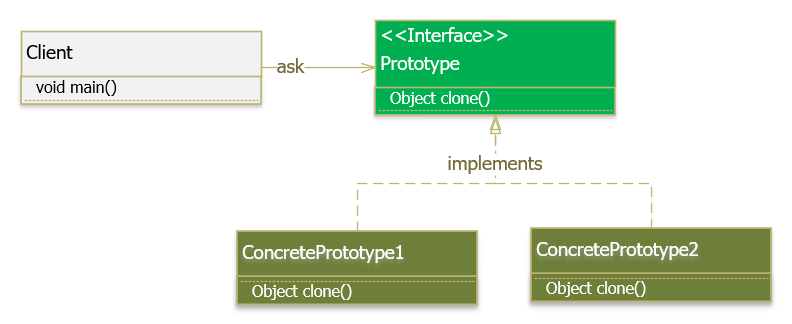
Ví dụ: Tất cả hệ thống ngân hàng có cung cấp API để truy cập đến hệ thống của họ. Team được giao nhiệm vụ thiết kế một API để client có thể sử dụng dịch vụ của một ngân hàng bất kỳ. Hiện tại, phía client chỉ cần sử dụng dịch vụ của 2 ngân hàng là VietcomBank và TPBank. Tuy nhiên để dễ mở rộng sau này, và phía client mong muốn không cần phải thay đổi code của họ khi cần sử dụng thêm dịch vụ của ngân hàng khác. Với yêu cầu như vậy, chúng ta có thể sử dụng một Pattern phù hợp là Factory Method Pattern.



Hình 2: Minh họa hệ thống áp dụng Factory Method.

1. **Prototype**.

Là một trong những **Creational pattern**. Nó có nhiệm vụ khởi tạo một đối tượng bằng cách **clone** một đối tượng đã tồn tại thay vì khởi tạo với từ khoá **new**. Đối tượng mới là một bản sao có thể giống 100% với đối tượng gốc, chúng ta có thể thay đổi dữ liệu của nó mà không ảnh hưởng đến đối tượng gốc.



Một Prototype Pattern gồm các thành phần cơ bản sau:

Prototype: khai báo một class, interface hoặc abtract class cho việc clone chính nó.

ConcretePrototype class: các lớp này thực thi interface (hoặc kế thừa từ lớp abstract) được cung cấp bởi Prototype để copy (nhân bản) chính bản thân nó. Các lớp này chính là thể hiện cụ thể phương thức clone(). Lớp này có thể không cần thiết nếu: Prototype là một class và nó đã implement việc clone chính nó.

Client class: tạo mới object bằng cách gọi Prototype thực hiện clone chính nó.

Ví dụ: Một công ty có cấu hình máy tính đều giống nhau cho tất cả nhân viên, bao gồm: Hệ điều hành (os), Phần mềm văn phòng (office), Phần mềm diệt virus (antivirus), Trình duyệt (Browser), và một số phần mềm khác (others) tùy theo nhu cầu của mỗi nhân viên sẽ được cài đặt thêm. Việc cài đặt tất cả phần mềm trên rất tốn thời gian, nên anh IT đã nghĩ ra một cách là sẽ tạo ra một bản chuẩn cho một máy tính và có thể clone() lại cấu hình đó cho một nhân viên khác mà không cần phải cài đặt lại từ đầu.

Cách làm này được minh họa như sau:

Đầu tiên, chúng ta sẽ tạo một lớp Computer và implement từ interface Cloneable.

Tiếp theo, chúng ta sẽ tạo một computer chuẩn (instance).

Sau đó, mỗi khi có nhân viên mới, chỉ việc clone() từ computer chuẩn đã tạo. Tùy theo, nhu cầu của mỗi nhân viên có thể thay đổi lại cho phù hợp (*setOthers()*).

1. **Singleton**

**Singleton** đảm bảo chỉ duy nhất **một thể hiện (instance)** được tạo ra và nó sẽ cung cấp cho bạn một method để có thể truy xuất được thể hiện duy nhất đó mọi lúc mọi nơi trong chương trình.