**FACTORY METHOD**

1. **Nó là gì?:** 
   * Một nhà máy, “sản xuất” các đối tượng theo yêu cầu của chúng ta
   * Factory Method giúp tạo đối tượng mà không để lộ logic tạo đối tượng ở phía người dùng và tham chiếu đến đối tượng mới được tạo ra bằng cách sử dụng 1 interface chung
2. **Thành phần cơ bản:**
   * Super class: interface/ abstract class/ class
   * Sub classes: implement các phương thức của super class
   * Factory Class: khởi tạo các đối tượng sub class dựa theo tham số đầu vào. Thường sử dụng if-else hoặc switch-case để xác định class con đầu ra
3. **Khi nào nên sử dụng:**
   * Khi có 1 class cha (super class: interface/abstract class/class) với nhiều class con, và dựa trên đầu vào, ta cần trả về 1 class con => đưa trách nhiệm khởi tạo một lớp từ client sang Factory
   * Không biết sau này cần đến những lớp con nào nữa => khi cần mở rộng, chỉ cần tạo sub-class và implement thêm vào factory method
4. **Lợi ích:**
   * Giảm sự phụ thuộc giữa các module (loose coupling)
   * Mở rộng code dễ dàng
   * Che giấu logic của các objects được khởi tạo
   * Dễ quản lý life cycle
5. **Code demo**

Ex: BankFactory (FactoryPatternDemo2\_Bank)

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 MBBank và BIDV. 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.

* Muốn thêm mới 1 bank chỉ cần tạo thêm 1 class implement interface Bank, thêm vào logic khởi tạo Bank trong BankFactory => không ảnh hưởng code từ client.