1. Khái niệm cơ bản

* Là kĩ thuật thiết kế hướng đối tượng
* Xác định các class, các đối tượng: vai trò, sự cộng tác và trách nhiệm của từng phần

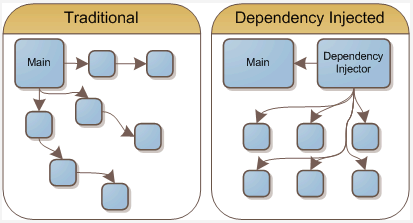
1. FACTORY METHOD

* Là một trong những đối tượng thuộc nhóm CREATIONAL DESIGN PATTERN.
* Nhiệm vụ chính là quản lý và trả về các đối tượng theo yêu cầu, giúp cho việc khởi tạo các đối tượng một cách linh hoạt
* Factory Method được sử dụng khi có 1 class cha và nhiều class con, yêu cầu trả về đối tượng thuộc một trong các class con đó
* Các thành phần của một FACTORY METHOD
  + Super Class: có thể là Inteface, abtract class hoặc 1 class thông thường
  + Sub Class: các class con sẽ thực thi các method dựa vào yêu cầu nghiệp vụ của nó
  + Factory Class: một class chịu trách nhiệm tạo ra các đối tượng theo subclass dựa theo tham số đầu vào ( Lưu ý: class này là singleton hoặc sẽ cung cấp các static method cho việc truy xuất và khởi tạo đối tượng)
* Sử dụng FACTORY PATTERN khi nào:
  + Cần trả về 1 đối tượng của 1 class con theo đầu vào cua chúng. Đưa trách nhiệm khởi tạo đổi tượng từ phía người dùng sang FACTORY METHOD
  + Sau này có thể phải mở rộng thêm nhiều loại lớp con phụ thuộc vào yêu cầu đầu vào
* Lợi ích chinh của FACTORY PATTERN:
  + Giảm sự phụ thuộc giữa các module ( loose Couple). Cung cấp 1 hướng tiếp cần Interface thay vì implement, giúp chương trình độc lập với các class cụ thể mà ta cần tạo ra các đối tượng
  + Dễ dàng mở rộng code: khi cần mở rộng, ta chỉ cần khởi tạo thêm các subclass và implement thêm vào FACTORY
  + Khởi tạo các đối tượng mà che dấu đi xử lý logic của việc khởi tạo ấy
  + Dễ dàng quản lý LifeCycle của các object được tạo ra bởi FACTORY
  + Thông nhất về name convention

1. SINGLETON

* Khái niệm:
  + Là 1 trong 5 design pattern của nhóm Creational Design Pattern
  + Đảm bảo mỗi class chỉ có 1 thể hiện ( instance) và cung cấp một phương thức có thể truy xuất thể hiện đó trong mọi nơi của chương trình
* Sử dụng Singleton khi chúng ta muốn:
  + Đảm bảo mỗi lớp chỉ có 1 instance
  + Quản lý việc truy cập tốt hơn vì chỉ có 1 instance
  + Hoặc có thể giới hạn số instance của 1 lớp với số lượng nhât định
* Nguyên tắc cơ bản để implement Singleton Pattern:
  + Private Constructor để hạn chế việc khởi tạo đối tượng từ bên ngoài
  + Đặt private static final variable để đảm bảo biến chỉ được khởi tạo trong class
  + Có một static public method để trả về instacne được khởi tạo trên
* Các phương pháp để implement Singleton Pattern
  + Khởi tạo ngay khi được gọi đến, tuy nhiên nhược điếm của phương pháp này là đã được khởi tạo tuy nhiên có thê sẽ ko được dùng đến
  + Static block initialization: cũng giống cách trên chỉ khác block static cung cấp thêm lựa chon cho việc handle exception hay sử lý các khác

1. DEPENDENCY INJECTION PATTERN

* Dependency là 1 loại quan hệ giữa 2 class trong đó 1 class hoạt động độc lập và 1 class phụ thuộc vào class kia. Sự phụ thuộc này gây khó khăn trong quá trình mở rộng và nâng cấp hệ thống. DI sinh ra để ngăn chặn sự phụ thuộc này
* 
  1. Khái niệm DIP, IOC, DI
* Đều là 3 nguyên lý hương đến mục đích tạo ra ứng dụng ít kết dính ( loosely coupling)
* Dependency Inversion Principle (nguyên lý đảo ngược sự phụ thuộc). Nội dung nguyên lý:
  + Các module cao cấp không nên phụ thuộc và module cấp thấp, Cả 2 nên phụ thuộc vào Abtraction
  + Abtraction ( ở đây là inteface hoặc abtract class) không nên phụ thuộc vào chi tiết mà ngược lại. Các class giao tiếp thông qua inteface, không phải thông qua implement
* IOC (Inverter Of Control) : là một design pattern được tạo ra để code tuân thủ nguyên lý DIP. Có nhiều cách đê thực hiện pattern này, DI là một trong các cách đó
* DI ( Dependency Injection)