**Design Pattern** là kỹ thuật lập trình sử dụng các mẫu thiết kế có sẵn, dùng để giải quyết một vấn đề.

## **Creational Pattern ( nhóm khởi tạo):**

Nhóm này sẽ giúp bạn rất nhiều trong việc khởi tạo đối tượng, mà bạn khó có thể nhận ra (nó sẽ không dùng từ khóa new như bình thường). Nhóm này gồm 9 mẫu design là:

* Abstract Factory.
* Builder.
* Factory Method.
* Multiton.
* Pool.
* Prototype.
* Simple Factory.
* Singleton.
* Static Factory

## **Structural (nhóm cấu trúc):**

Nhóm này sẽ giúp chúng ta thiết lập, định nghĩa quan hệ giữa các đối tượng. Nhóm này gồm có 11 mẫu design là:

* Adapter/ Wrapper.
* Bridge.
* Composite.
* Data Mapper.
* Decorator.
* Dependency Injection.
* Facade.
* Fluent Interface.
* Flyweight.
* Registry.
* Proxy

## **Behavioral patterns (nhóm ứng xử):**

Nhóm này sẽ tập trung thực hiện các hành vi của đối tượng. Gồm 12 mẫu design là:

* Chain Of Responsibilities.
* Command.
* Iterator.
* Mediator.
* Memento.
* Null Object.
* Observer.
* Specification.
* State.
* Strategy.
* Template Method.
* Visitor.

Ngoài ra thì trong thời gian gần đây đã xuất hiện thêm 4 mẫu design nữa đó là:

* Delegation.
* Service Locator.
* Repository.
* Entity-Attribute-Value (EAV).

**Singleton Pattern:** Là Pattern được sử dụng để tạo sự ràng buộc, chỉ cho phép với 1 class chỉ có duy nhất 1 đối tượng được khởi tạo. (Khóa Magic method \_\_construct() và \_\_Clone() và sử dụng phương thức tĩnh, thuộc tính tĩnh)

**Method Factory Pattern:** Hỗ trợ việc xử lý tạo đối tượng, mà chưa biết lớp nào sẽ thực thi công việc này (Sử dụng lớp trung gian, dựa vào đối số truyền vào để khởi tạo đối tượng)

**Dependency Injection:** quản lý sự phụ thuộc trong ứng dụng, giúp cho các lớp giảm đi sự phụ thuộc

**Strategy Pattern:** Đóng gói từng lớp riêng biệt dưới sự quản lý của Interface, mục đích tạo sự ràng buộc cho các lớp riêng biệt phải tuân thủ theo

Class\_exits($class) : Kiểm tra Class có tồn tại hay không?