**Design Pattern**

**Creational Pattern**

1. Factory Pattern

Là pattern thuộc nhóm creational pattern, chức năng của nó dùng để tạo ra đối tượng mà không cần quan tâm tới logic lúc khởi tạo

Ưu điểm : dễ cài đặt

Nhược điểm :

Gây khó khăn khi có nhiều implentation của class đối tượng.

Khó để thay đổi cài đặt khi khởi tại đối tượng

1. Abstract Factory Pattern

Là Factory Pattern nhưng mở rộng hơn, thay vì khởi tạo đối tượng thì khởi tạo Factory

Ưu điểm : khắc phục nhược điểm trên của Factory Pattern khi có tính phân loại

Nhược điểm : khó để thay đổi cài đặt khi khởi tại đối tượng

1. Singleton Pattern

Tạo ra một đối tượng duy nhất của class trong quá trình chạy

Ưu điểm : đơn giản

Nhược điểm : khó kiểm soát để đảm bảo rằng chỉ có một đối tượng được tạo

1. Builder Pattern

Đưa ra cách tạo ra đối tượng cách chi tiết và dễ tùy chỉnh khi đối tượng có nhiều thuộc tính, khắc phục nhược điểm của Factory Pattern.

Ưu điểm : dễ custom

Nhược điểm : tăng độ phức tạp khi cài đặt

1. Prototype Pattern

Tạo ra các đối tượng mà nguồn để tạo ra đối tượng là giới hạn, VD (lấy object từ database)

Ưu điểm : tối ưu về tốc độ khi không spam vào nguồn dữ liệu khi bị hạn chế về phần cứng

Nhược điểm : khó kiểm soát khi dữ liệu gốc bị thay đổi, dẫn đến không đồng bộ

**Structual Pattern**

1. Adapter Pattern

Cho phép những object không liên quan đến nhau có thể hoạt động và tương thích với nhau

Ưu điểm : các object không cần có mối liên nào giữa chúng

Nhược điểm : cài đặt lằng nhằng nếu logic phức tạp

1. Bridge Pattern

Tạo ra cầu nối giữa các object mà có chung tính trừu tượng

Ưu điểm : object dễ dàng khởi tạo độc lập với nhau

Nhược điểm : tìm ra được tính trừ tượng tổng quát nhất của các object liên quan

1. FilterPattern

Dựa trên ý tưởng loose coupling để tạo ra các tác vụ độc lập nhằm để phân loại dữ liệu