**Singleton Design Partern là gì?**

* Là 1 Creational Design Partern.
* Đảm bảo rằng 1 class chỉ được khởi tạo 1 lần duy nhất (chỉ có 1 instance duy nhất trong toàn bộ ứng dụng), đảm bảo giá trị các thuộc tính (Property) của class không bị thay đổi sau khi class được khởi tạo.
* Cung cấp 1 Global Entrypoint để các class khác truy cập.

**Ưu điểm:**

* Đảm bảo class chỉ có duy nhất 1 instance.
* Có thể truy cập ở bất cứ đâu trong ứng dụng. Đến đây bạn có thể thấy khá giống với biến toàn cục (global variable) nhưng Singleton Class khác với biến toàn cục ở chỗ không bị ghi đè giá trị.
* Chỉ được khởi tạo khi có yêu cầu.

Nhược điểm:

* Vi phạm quy tắc Single Responsibility Principle (1 đối tượng chỉ phục vụ 1 mục đích duy nhất) trong SOLID. Vì thế có thể gọi ISngleton là 1 Aniti Partern.
* Phải rất cẩn thận khi sử dụng đa luồng (Multi-Thread) vì có thể nhiều Thread cùng truy cập vào 1 object trong 1 thời điểm.
* Khó khan trong Unit Test

**Khi nào sử dụng Singleton?**

* Quản lý cấu hình (Configuration Management).
* Quản lý kết nối cơ sở dữ liệu (Database Connection).
* Logging (Ghi nhật ký).
* Trình điều khiển (Driver).
* Cache.
* Thread Pool (Bộ xử lý luồng).
* File System Manager (Quản lý hệ thống tệp.

**Cấu trúc của Singleton Partern**

* Private Constructor: Đảm bảo các class sử dụng Singleton Class không thể khởi tạo instance mới.
* GetInstance Method: Kiểm tra xem class đã được khởi tạo chưa? Nếu chưa thì tạo mới instance ngược lại trả về instance hiện tại.

**Demo**

* Trong loạt bài này mình sử dụng ngôn ngữ C# để Demo các Design Partern.
* Đầu tiên mình tạo ra class Singleton A screenshot of a computer

  Description automatically generated
* Sau đó trong Program.cs, mình tạo ra 2 biến s1, s2 để tạo các instance của class Singleton

A screenshot of a computer

Description automatically generated- OK! Mọi thứ hoạt động tốt khi chỉ trả lại 1 instance duy nhất của class Singleton.