Singleton Pattern

* Định nghĩa: là một mẫu thiết kế khởi tạo có 2 mục tiêu:
  + Chỉ có một instance duy nhất của một class tạo ra trong suốt vòng đời của ứng dụng.
  + Cung cấp một điểm truy cập toàn cục.
* Tại sao cần phải dung Singleton?
  + Trong trường hợp thực tế, không phải lúc nào cũng có thể dễ dàng tạo ra các instance mới.
  + Việc tạo ra instance mới quá nhiều dẫn đến việc đụng độ tài nguyên, lỗi logic, sai lỗi trong truy xuất dữ liệu xử lý hoặc tệ hơn là bị treo đơ phần mềm sử dụng.
* Ý tưởng triển khai:
  + Nói ngắn gọn đó là việc ẩn đi tất cả các phương thức Constructor để không cho phép tạo mới của class.
  + Cung cấp một phương thức tĩnh để truy cập thông qua class đó, nếu class đó chưa có instance nào tồn tại thì mới khởi tạo, còn không thì đều xử lý trên instance đã tồn tại trong tài nguyên.
* Về ưu điểm:
  + Đảm bảo chỉ có một instance tồn tại của class đó.
  + Cho phép kiểm soát truy cập toàn cục đến instance.
* Về nhược điểm:
  + Do class vừa làm công việc chính vừa phải quản lý cả instance nên gây ra vi phạm SRP trong nguyên lý SOLID.
  + Gây một số khó khan hơn trong việc thực hiện testcase.
  + Cần có kỹ năng kiểm soát code khi sử dụng trong trạng thái toàn cục này.
* Code demo:

namespace Singleton

{

class Database

{

private Database() { }

public static Database Instance;

public static Database GetDatabase()

{

if(Instance == null)

{

Instance = new Database();

}

return Instance;

}

public void ConnectDb()

{

}

}

internal class Program

{

static void Main(string[] args)

{

Database db = Database.GetDatabase();

db.ConnectDb();

}

}

}