* 싱글톤

일반적으로 다양한 곳에서 하나의 변수를 처리하기 위해서 사용한다.

Ex) 점수 처리가 대표적.

자신의 단일 인스턴스만 생성될 수 있게 해주는 클래스이며 일반적으로 해당 인스턴스에 대한 간단한 액세스를 제공한다.

(매니저 클래스, 컨트롤러 클래스를 싱글톤을 이용하면 좋을 것 같음)

즉, 애플리케이션이 시작되리 때 어떤 클래스가 최초로 한번만 메모리를 할당하고(static) 그 메모리에 인스턴스를 만들어서 사용하는 디자인 패턴.

생성자가 여러 차례 호출되더라도 실제로 생성되는 객체는 하나고 최초 생성 이후에 호출된 생성자는 최초에 생성한 객체를 반환한다.

굳이 쓸 필요가 있을까? 할 수도 있음

싱글톤으로 만들어진 클래스의 인스턴스는 전역 인스턴스이기 때문에 다른 클래스의 인스턴스들이 데이터를 공유하기가 ‘쉽다’

그리고 쓰레드풀, 캐시, 대화상자, 사용자 설정, 레지스트리 설정, 로그 기록 객체 등 DBCP(DataBase Connection Pool)처럼 공통된 객체를 여러 개 생성해서 사용해야하는 상황에서 많이 사용한다.

물론 싱글톤 인스턴스가 너무 많은 일을 하거나 많은 데이터를 공유시키면 다른 클래스의 인스턴스들 간 결합도가 높아지게 되면서(-> 객체 지향 설계에서 점점 멀어진다는 소리) 수정이 어려워지고 테스트하기가 어려워진다.

싱글톤은 유니티에서 2가지버전으로 구분할 수 있다.

싱글톤은 MonoBehaviour 상속을 받지 않는다.

 public class GameData

 {



     private static GameData instance=null;

     public static GameData Instance

     {

         get

         {

             if (instance==null)

             {

                 instance = new GameData();

             }

             return instance;

         }

     }



     private GameData()

     {

     }



     public int \_test; // 선언할 변수들

 }

보통 이런 방식으로 이용한다고 함

패턴 여러 개니까 더 찾아서 추가