Singleton Pattern

결론 및 정의

정의

클래스의 인스턴스를 오직 하나만 생성하도록 보장하는 패턴. 다음과 같은 특징을 가짐.

- 여러 개의 인스턴스가 생성되는 것을 방지하여 리소스를 효율적으로 관리하고 일관성을 유지하는 데 도움.
- 어디서든 접근 가능.

구현

일반적인 경우

- Constructor 를 private 으로 하여, 외부에서 해당 Class를 생성할 수 없도록 함.
- 자기자신을 가지는 static variable 을 만듬.
  - eager instantiation : field 자리에 해당 내용을 입력함으로써, 해당 Class가 load 되었을 때 해당 instantiation 동작 또한 자동으로 될 수 있도록 해줌.
  - 호출했을 때 생성이 되도록 하는 lazy instantiation 을 사용해줄 수도 있음!
- static public MySingleton getInstance () 라는 method 를 만들어, 외부 어디서나 해당 Singleton 에 접근할 수 있도록 해줌.

Unity

보통 Property 를 기반으로 한 lazy instantiation 을 통해 구현됨.
static 을 통하여 외부에서 접근 하도록 하는 방식 자체는 동일. 단, Scene 이 반복해서 실행될 때 중첩될 수 있다는 경우를 고려하여, Scene의 생성 시 존재를 체크함.

Context & Problem (당면 상황)

- 하나의 인스턴스만 존재해도 되며, 이를 돌려쓰면 될 때 (리소스 절약)
- 어디서든 접근 가능한 Class의 Instance 를 만들어 관리해주고 싶을 때

Goal

결국 Context & Problem 에서 제시된 문제를 해결할 수 있다는 것이 Goal 이 아닐까 함.

강의 들으며

해당 Class 에 대한 하나의 Object 만 존재하며, 어디서든 해당 Object 에 접근할 수 있도록 하는 것.

Singleton 자체의 정의가 이것임.
해당 Class 가 하나의 Instance 만 가지도록 하며, 접근할 수 있는 Global Point 를 제공하는 것.

이를 어떻게 구현할 수 있을까?

- Constructor 를 private 으로 하여, 외부에서 해당 Class를 생성할 수 없도록 함.
- 자기자신을 가지는 static variable 을 만듬.
  - eager instantiation : field 자리에 해당 내용을 입력함으로써, 해당 Class가 load 되었을 때 해당 instantiation 동작 또한 자동으로 될 수 있도록 해줌.
  - 호출했을 때 생성이 되도록 하는 lazy instantiation 을 사용해줄 수도 있음!
- static public MySingleton getInstance () 라는 method 를 만들어, 외부 어디서나 해당 Singleton 에 접근할 수 있도록 해줌.

Singleton Pattern 은 이러한 구조를 의미하기도 하나, Singleton은 이렇게 구성된 Class 를 의미하기도 하는 듯!
Strategy가 Strategy Pattern 에서 각 동작의 추상을 의미하는 것이기도 했던 것처럼!

또한 Singleton 은 구조에 대한 Pattern 일 뿐이고. 이를 사용하면 다른 곳에서 접근하거나, 하나만 유지하는 것이 쉬워질 뿐임.
따라서, 그 내에는 Singleton으로 만들어줄 필요가 있었던 동작이 나오게 될 것.

- Game에서 Singleton 을 어떻게 사용하는지에 대한 예시.
- Main Manager를 만들고, 다른 Manager를 만들어 연결하여 둠으로써 사용.