Observer Pattern

결론 및 정리

정의

객체 사이에 일대다 의존성을 정의하는 패턴으로, 객체(Subject)의 상태 변화를 관찰하고, 이에 따라 다른 객체들 (Observer) 에게 자동으로 알림을 보내 업데이트를 수행할 수 있도록 함.

Observer Pattern 은 다음과 같은 구성요소로 이루어져 있음.

1. Subject Class : 상태 변화를 감지하는 주체. 관찰자들을 등록하고, 상태 변화가 발생하면 등록된 관찰자들에게 알림을 보낸다.
    - Subject Class 내부에 해당 상태 변화에 관한 변수가 들어있을 수도 있으며. 외부의 상태 변화 등을 체크하여, 보내줄 수도 있을 듯.
2. Observer Interface : 상태 변화를 감지하고 처리할 수 있는 메서드들을 정의하는 인터페이스. Subject Class의 변수의 타입으로 쓰인다.
3. Concrete Observer Class : 실질적으로 Subject Class 가 보낸 알림(notify) 을 받아, 해당 상태 변화에 대해 처리하는 Class. Subject Class에 실질적으로 등록되는 친구들이다.

> *평소에 Subject Class에 Observer 를 등록하고, 관리하고, 빼고.*
> *Subject Class 가 변화를 체크하고. 변화가 발생하면 등록된 Observer 들에게 알림을 보내고.*
> *Observer 는 알림이 들어오면, 그에 따라 활동이 이루어지는 방향으로 처리된다.*

구현

1. 알림이 왔을 때 이를 처리하는 Method 가 든, Observer Interface를 만든다.
2. List<Observer> 를 변수로서 가지고 있는 Observer 를 관리하고 알림을 보내줄 친구인 Subject Class 를 만든다.
3. 알림이 들어왔을 때 해줄 행동에 대한 각각의 내용이 들어있는 Concrete Observer Class를 만들어준다.
4. Concrete Observer Class를 Subject Class의 List<Observer> 변수에 추가하는 등으로써 관리해주고, 알림을 처리한다.

다른 구현

- Chapter 3. 유니티 프로그래밍 설명 - Delegate and Event : Delegate 와 Event 는 C# 에 들어있는 함수 포인터의 개념임. 따라서 위와 같이 Observer Interface 를 별도로 만들어줄 필요

없이 그냥 일반적인 함수를 등록함으로써 만들어줄 수 있으므로, 더 간편하다고 볼 수 있음. 단, 함수의 반환형 및 인수는 동일하게 맞출 필요가 있다.

- UnityEvent & UnityAction : UnityEvent 는 C#의 Event (직접적인 대입이 불가능하도록 막아둔 Delegate) 를 Unity 가 조금 변형한 Type이다. 일반적인 내용은 비슷하나, AddListener 란 함수를 사용하며, Inspector 상에 표시 되며, 인수형에 조금 더 유동적이라는 장점이 있다.

Context & Problems & Merits

다음과 같은 경우에서 활용할 수 있음.

1. 객체 간의 일대다 의존성: 하나의 객체의 상태 변화가 여러 개의 객체에 영향을 주어야 하는 경우.
2. 분산 이벤트 처리: 분산 환경에서 이벤트 발생에 따라 다른 객체들에게 알림을 전달해야 할 때.

객체 간의 느슨한 결합을 유지하며, 상태 변화에 따른 업데이트 동작을 유연하게 구현할 수 있다는 점이 장점임.

강의 들으며

> *어느 정도 중요한 느낌 드는 것만 체크하자.*

Subscribe
Unsubscribe

- Subscriber = Observer
- Event Sender = Subject
- Subscribe
- Notify
- UnSubscribe
- Subscriber (Observer) has "update" - this is interface
- Event Sender has Subscribe - Save Observer
- Notify - Send Notify to Observers we saved
- Unsubscribe - Delete Observer inside of event sender

=> 결국에 뭐다?
Subscribe 한 사람에게만, 변화나 전달하고 싶은 정보가 있을 때 전달하는 방법.

또한, 그 Subscribe 자들을 관리할 수 있는 방법 또한 제공하는 것.
Observer 는 중요한 정보가 있을 때 받아가고 싶고, Subject 에 그 정보가 들어있다 하자. 이 때, 가장 권장되는 방법은 Observer 가 존재하고. Subject 에 Observer 를 넣어 관리할 수 있도록 하고, Subject 는 정보의 변경 등이 일어났을 때, 직접 지금 구독하고 있는 친구들에게 정보를 전송하는 것이다.

일대다 연관성을 가진다.
Subject => Many Observers
만드는 방법 자체가 엄청 다양하다!
=> 내가 기존에 알고 있는 Unity Event 와 Delegates 를 이용하는 방법과, 여기에 나와 있는 방법을 비교하면 좋을 듯함.

> *Design Principle 5 : 상호작용하는 오브젝트 간 Loosely coupled 를 유지할 수 있도록 노력하기.*
>
> *왜?*
>
> - *변경이 쉬워지기 때문에.*
> - *뭔가를 했을 때, 깨지거나 할 가능성이 적어지기 때문에.*
> - *Interface를 가능한 만큼 넣어야, 해당 동작을 별도로 조작하는 게 쉬워짐!*

Weather Data 에 관한 Example 제공.
Swing Observer (Observer 대신 Listener 라는 표현을 사용. - ActionListener Interface를 가지는.), Person Observer,