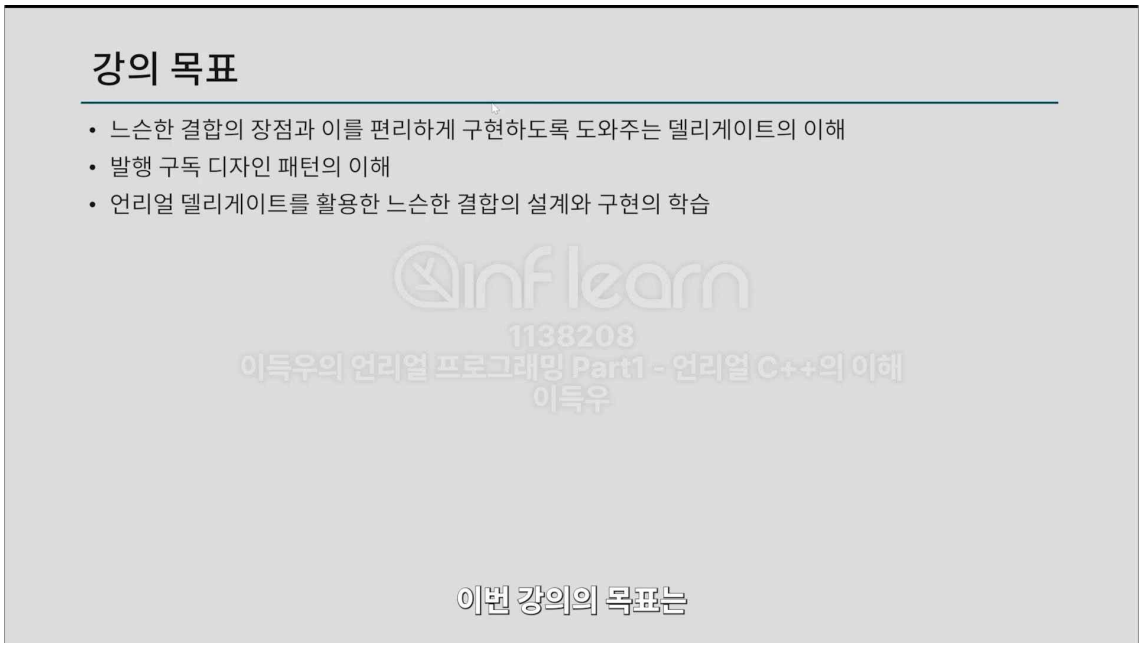
언리얼 프로그래밍 Part1-9

제목:언리얼 C++ 설계 III - 델리게이트

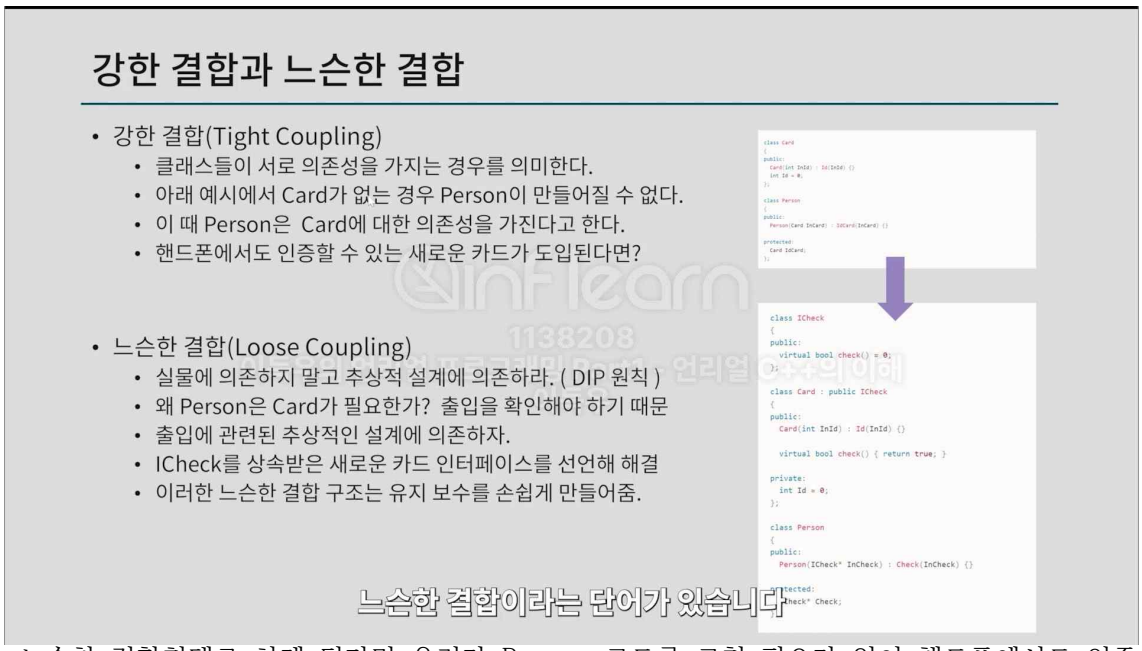
**강의 내용 : 언리얼 델리게이트를 사용해 클래스 간의 느슨한 결합을 구현하기

**강의 목표



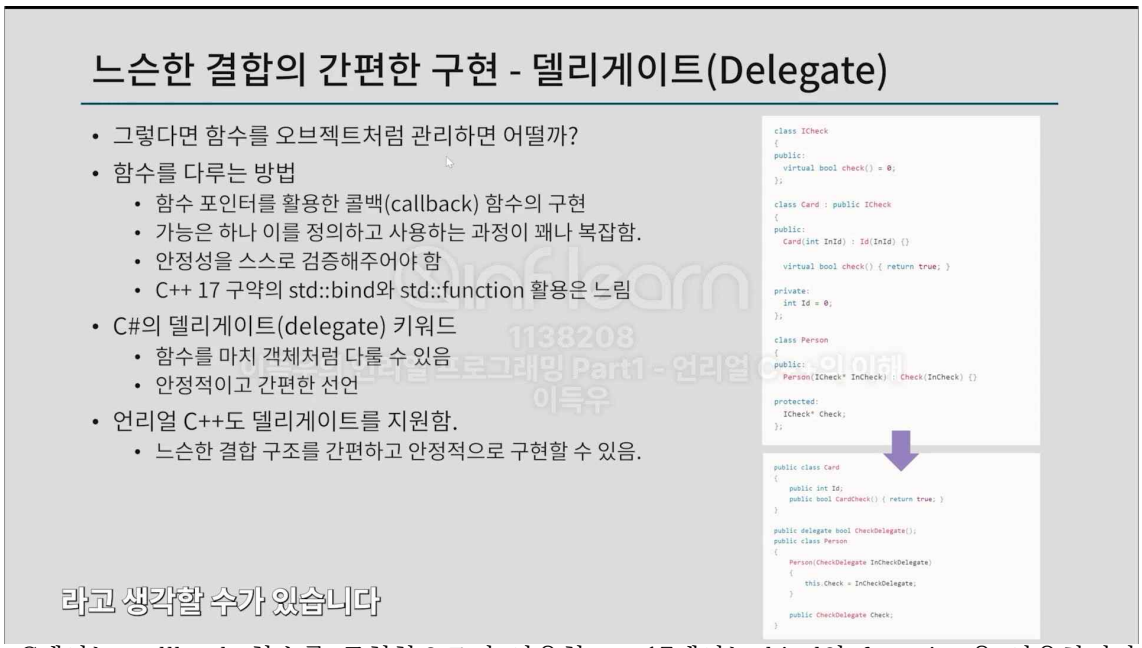
-모던객체 지향에서의 느슨한 결합을 알아볼거임.

**강한 결합과 느슨한 결합



-느슨한 결합형태로 하게 된다면 우리가 Person 코드를 고칠 필요가 없이 핸드폰에서도 인증 가능한 시스템 구현이 가능하다.

**느슨한 결합의 간편한 구현 - 델리게이트



-C에서는 callback 함수를 구현함으로써 사용함 c++17에서는 bind와 function을 사용하지만 속도가 느려 퍼포먼스가 중요한 게임에서는 사용을 잘 안하며 c#에서는 델리게이트 라는 키워드가 나왔는데 함수를 객체처럼 다룰 수 있음 또한언리얼 C++에서도 델리게이트를 사용함.
-밑의 코드를 보면 함수에 대한 것을 객체처럼 선언하고, 객체를 변수로 선언한 다음에 Card는 해당 델리게이트와 똑같은 유형의 함수만 지정해서 Person과 Card를 묶어주면 원하는 기능 구현이 가능하다.

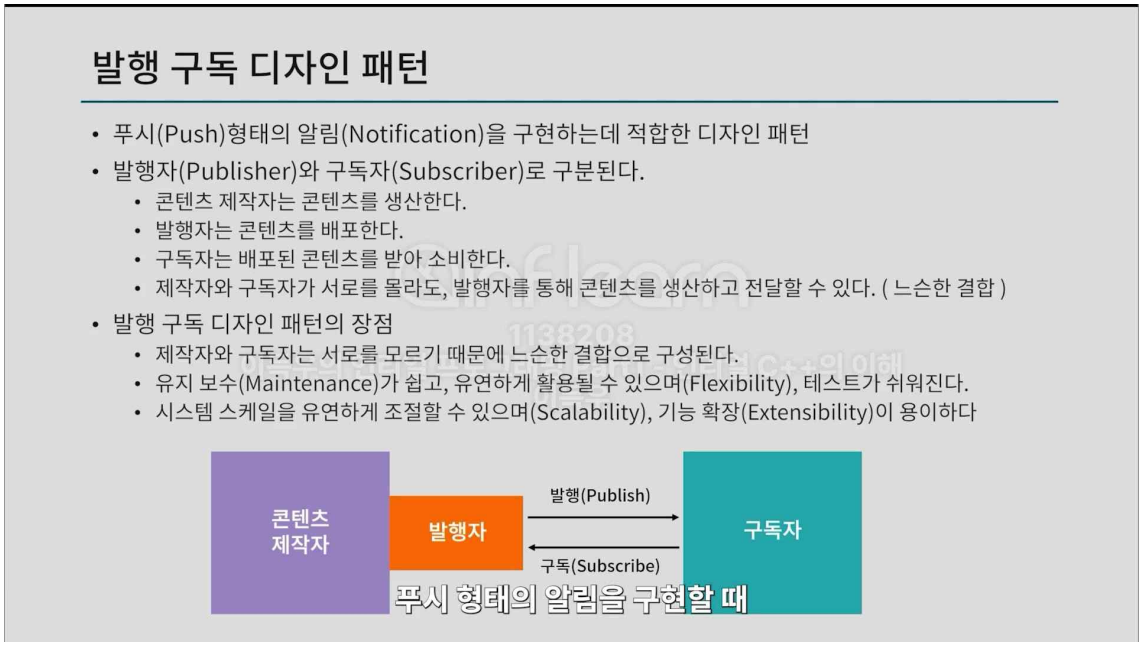
**언리얼 델리게이트
링크: https://dev.epicgames.com/documentation/ko-kr/unreal-engine/delegates-and-lamba-functions-in-unreal-engine?application_version=5.1
-델리게이트로 c++오브젝트 상의 멤버 함수 호출을 일반적이고 유형적으로 안전한 방식으로 할 수 있다. --> 우리가 어떤 객체를 사용할 때 객체 자체에 강한 결합을 하는 것이 아니라 어떤 객체가 가지고 있는 멤버 함수와 델리게이트를 연결하여 느슨한 결합을 만든다는 것으로 이해하자.
-델리게이트 오브젝트는 복사해도 완벽히 안전하다. 이것이 C의 함수포인터와 차별화된 기능이다.
-가급적 델리게이트는 항상 참조를 전달해야 한다.
-싱글-캐스트(형 변환)와 멀티-캐스트 모두 지원되며, 디스크에 안전하게 Serialize 시킬 수 있는 "다이내믹" 델리게이트등 다양한 델리게이트가 존재한다.
-'델리게이트 바인딩하기'는 우리가 선언한 델리게이트를 멤버함수, 일반함수, 정적함수 등 다양한 함수와 연결해주는 다양한 함수들이 있는데 그런 API를 살펴보는 항목임.(대부분 언리얼 오브젝트의 [BindUObject]를 사용하여 엮어주는 것을 많이 사용한다.)

-페이로드 데이터는 묶는 객체에 대한 정보를 지정해서 하나의 구문으로 편하게 묶을 수 있다는 것을 의미한다. 이렇게 묶게되면 델리게이트를 실행해서 묶인 함수를 호출할 수가 있게 된다. 대표적인 API로 Execute()가 있으며 '일대다'인 경우에는 브로드캐스트라는 함수를 통해 일대 다 호출도 진행이 가능하다.

-사용 예제 8:15~

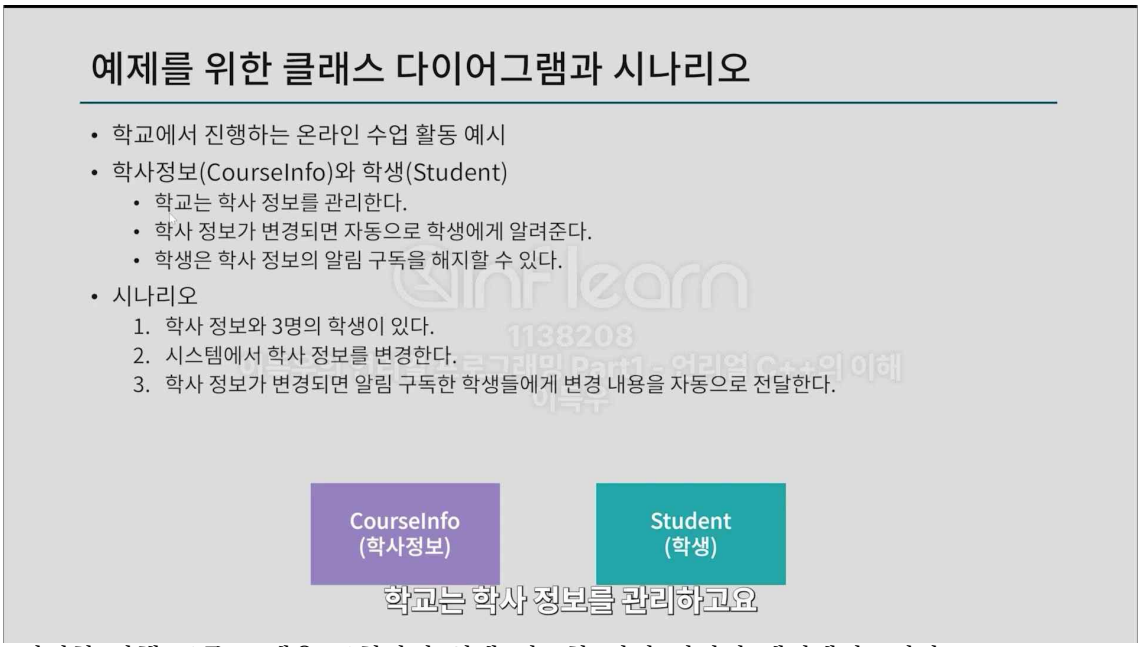
***발행 구독 디자인 패턴

**발행 구독 디자인 패턴

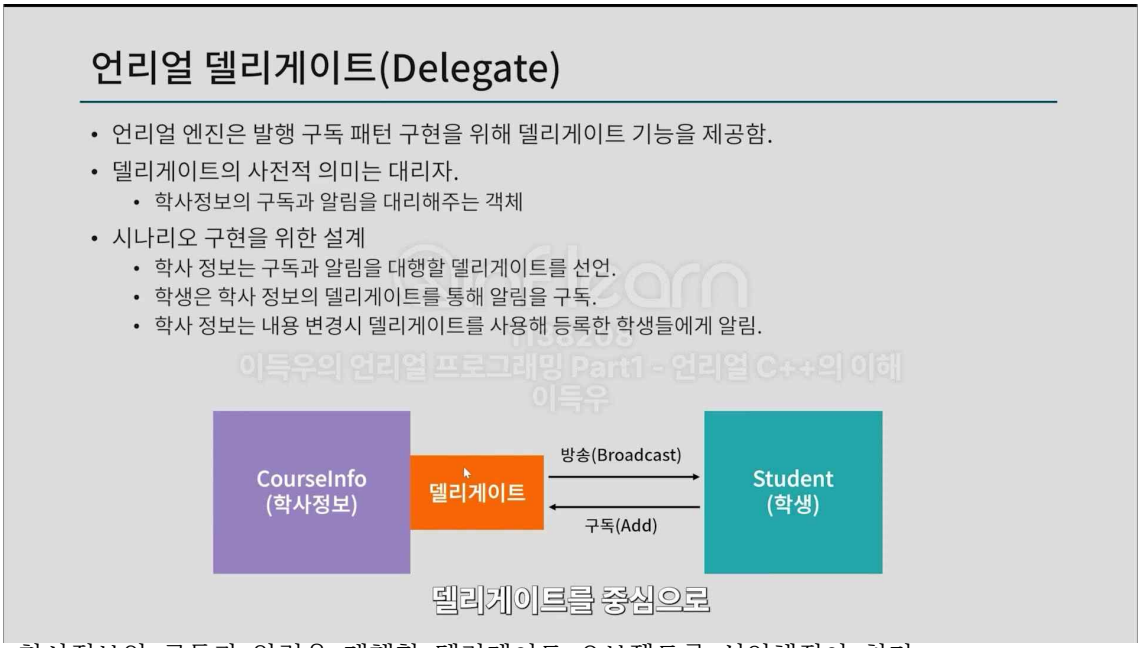


-콘텐츠 제작자가 콘텐츠를 생산할 때 자기를 대신해 줄 발행자를 새롭게 만들어 준다.
-발행자는 콘텐츠를 배포하는 역할을 담당하게 되는데 구독자는 발행자로부터 배포된 컨텐츠를 소비하게 된다.
-이러한 구조는 제작자가 콘텐츠 제작에 전념하게 해주고 구독자는 컨텐츠를 받아 소비하는데 전념하게 해준다.
-발행자라는 존재를 통해 제작자와 구독자는 느슨한 결합 관계를 유지할 수 있게 된다.
-자기가 하는 역할에만 전념할 수 있기 때문에 유지보수가 쉽고, 유연하며 테스트가 쉬워짐.
-시스템이 커지더라도 유연한 조절이 가능하며 기능확장에 용이하다.

**예제를 위한 클래스 다이어그램과 시나리오



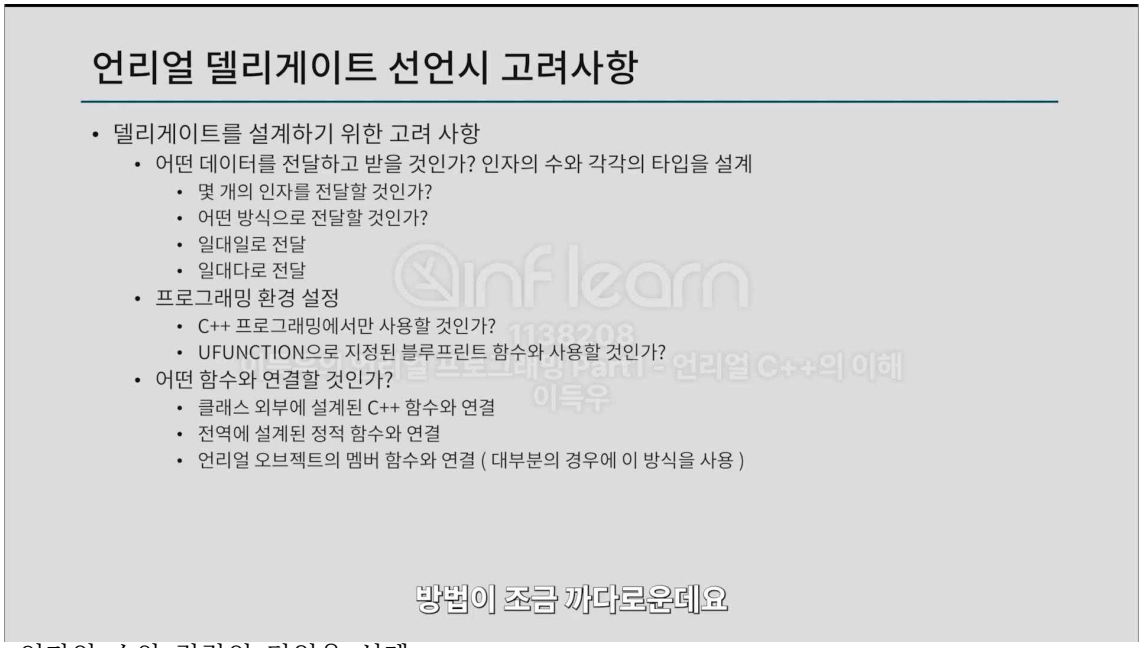
-이러한 발행 구동 모델을 구현하기 위해 필요한 것이 언리얼 델리게이트이다



-학사정보의 구독과 알림을 대행할 델리게이트 오브젝트를 선언해줘야 한다.
-학사정보의 구독와 알림을 대행할 델리게이트 오브젝트를 선언한다.
-언리얼 델리게이트를 선언한 후에는 델리게이트를 중심으로 학사정보와 학생정보를 엮는다.
-학사 정보가 변경될 때마다 구독한 학생들에게 자동으로 알림이 전달된다.

***언리얼 델리게이트의 선언

**언리얼 델리게이트 선언시 고려사항



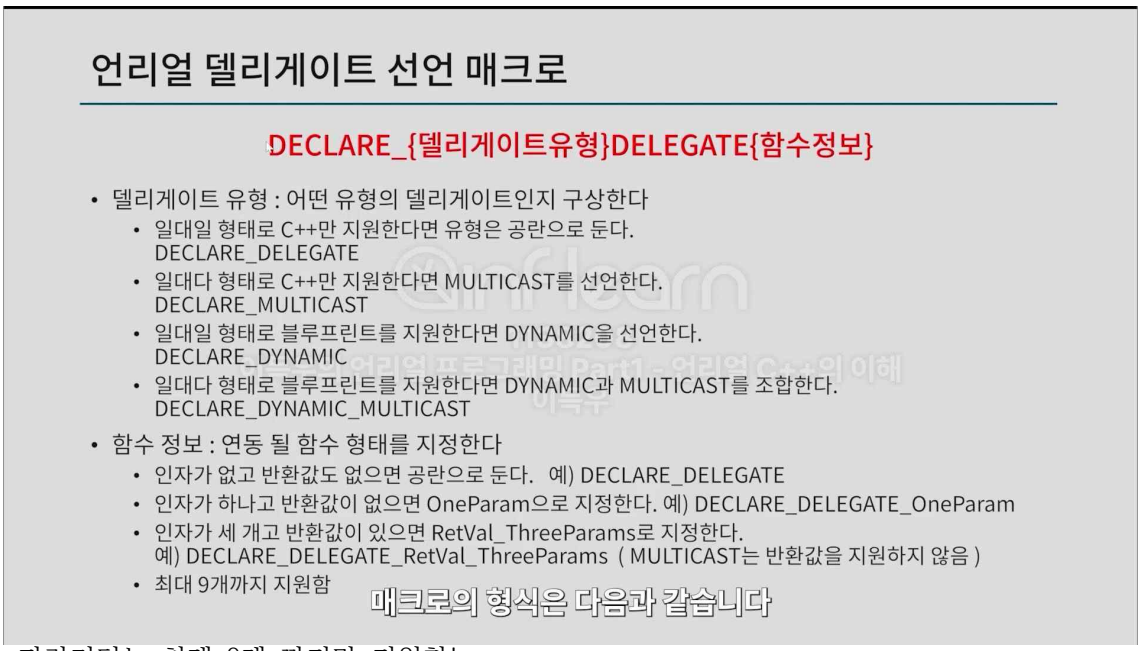
-인자의 수와 각각의 타입을 설계

-1:1로 전달할지 일대다로 전달할지 설정

-프로그래밍 환경 설정해야 하는데 c++에만 사용할 것인지 UFUNCTION으로 지정된 블루프린트와 사용할것인지 설정해야 한다.

-어떤 함수와 연결할것인지 설정해야 하는데 대부분 언리얼 오브젝트의 멤버함수와 연결한다.

**언리얼 델리게이트 선언 매크로



-파라미터는 최대 9개 까지만 지원함!

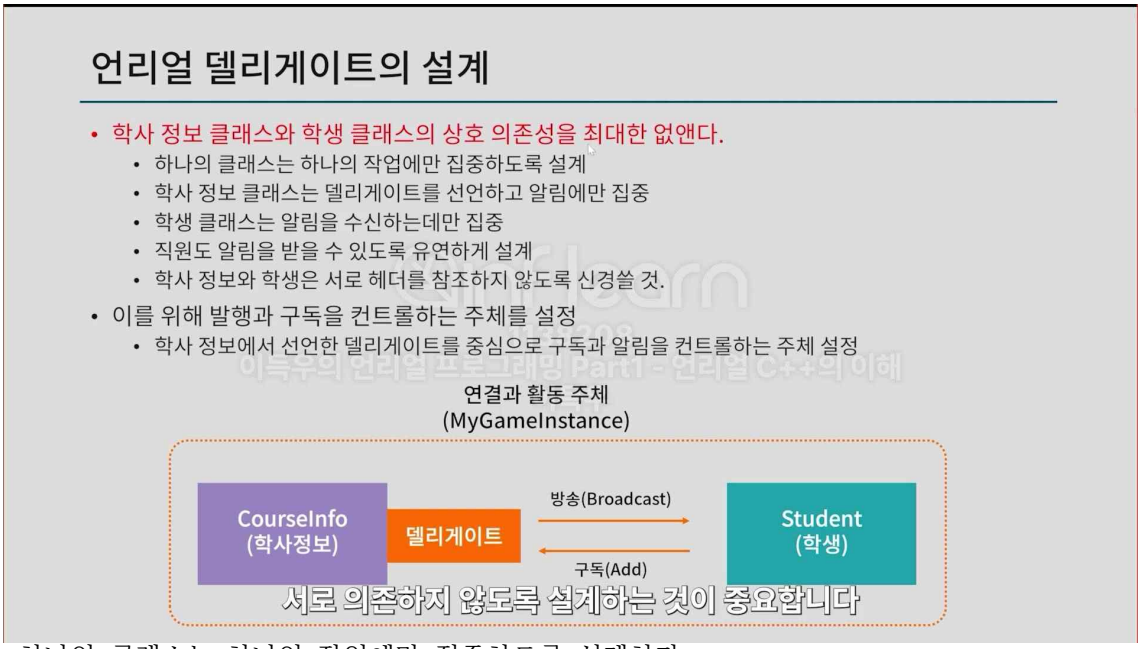
**언리어 델리게이트 매크로 선정 예시

-인자:알림 주체, 내용 ==>총2개

-변경된 학사정보는 다수인원에게 발송==>멀티캐스트

-오직C++==>다이나믹 사용 안함

**언리얼 델리게이트의 설계



-하나의 클래스는 하나의 작업에만 집중하도록 설계하자
-학사정보를 선언한 헤더와 학생을 선언한 헤더가 서로 include 하지 않도록 신경써야 하고 이렇게 할려면 학사정보에서 선언한 델리게이트를 중심으로 구독과 알림을 컨트롤하는 주체를 우리 학교에 대입시킨 MyGameInstance로 지정해서 엮어보자

**예제 실습
-앞서서 매크로를 지정해야 하는데 두 개의 인자를 가진 DECLARE_MULTICAST_DELEGATE_TwoParams()을 지정한다.

-인자는 다음과 같이 넣어준다. (FCourseInfoOnChangedSignature,const FString&, const FString&) [첫 번째:델리게이트 이름 (보통 F를 넣고 이벤트가 발생했다는 On을 넣고 변rudehlTek는 changed를 넣고, 이런 델리게이트는 보통 Signature라는 접두사를 많이 붙인다.) 그리고 다음에는 const FString으로 두 개의 인자를 받도록 선언한다.]

```
-FCourseInfoOnChangedSignature OnChanged; //마치 객체처럼 델리게이트 변수를 멤버변수로 써준다.
//우리가 이제 델리게이트의 어떤 함수가 와서 바인딩 한다면 우리가 송출해줘야 하는데 우리는 "내용이 변경되면 바로 송출해준다."
void ChangeCourseInfo(const FString& InSchoolName, const FString& InNewContents);
//윗줄이 "내용이 변경되면 바로 송출해준다." 부분임.
```

-학생까지 수정을 마쳤다면 우리가 눈여겨 봐야 할곳은 “우리가 어디에서 CourseInfo에 대한 헤더를 포함하지 않았다는 것이다.“ courseInfo에도 Student나 Teacher와 같은 헤더를 포함하지 않았다. 즉 완전히 개별적으로 구현한 것이다.

-중간에 이를 중재해주는 객체가 있으면 좋은데 이를 MyGameInstance가 진행할 것이다.

-MyGameInstance는 우선 학교를 우리가 대상으로 선언 하였으며 학교는 학사 시스템을 소유하여야 하기 때문에 학사 시스템을 소유하도록 CourseInfo를 추가해야 한다.

-학사 정보는 언리얼 오브젝트이고 포인터로 관리하기 때문에 전방 선언 사용이 가능하다. 또한 이렇게 선언하는 경우 언리얼 오브젝트의 포인터를 멤버변수로 지정할땐 TObjectPtr을 사용해야 한다.

```
UPROPERTY()
TObjectPtr<class UCrouseInfo> CourseInfo;
```

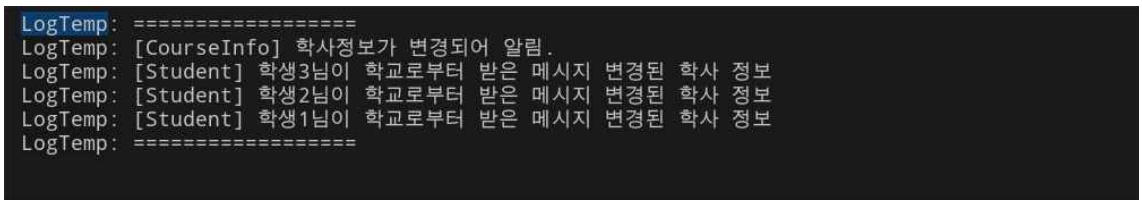
-MygameInstance.cpp에서 CourseInfo를 CDO안에서도 생성이 가능하지만 이번 강의에서는 외부에서 필요할때만 생성하도록 해볼 것이다.(사실은 학사정보는 당연히 학교에 필요한 것이라서 CDO에 생성하는 것이 더욱 적합하다.)

```
CourseInfo = NewObject<UCrouseInfo>()
```

-NewObject로 생성할 때 첫 번째 인자를 넣어줄 수 있는데 Outer를 지정해줄 수가 있다. 즉 우리가 생성한 객체는 클래스 멤버변수에 들어가서 앞으로 관리를 받고, 특별한 일이 있지 않는 한 메모리가 계속 유지가 되는데 이때 MyGameInstance는 courseInfo를 포함해야 한다. 이때 outer에 현재 MygameInstance를 선언하면 CourseInfo는 MyGameInstance의 서브 오브젝트가 되고 CourseInfo는

-CourseInfo에 있는 Onchaged라는 함수에다가 우리가 만든 Student객체들은 각각 연결해줘야 한다. 또한 AddUObject라는 별도의 함수를 사용하면 어떠한 클래스 인스턴스를 지정하고 멤버 함수(멤버변수 아님)를 직접 묶을수 있다.

-결과 화면



(1) 이렇게 학사정보가 변경되어 알림을 발송한다는 메시지가 뜨고 이후에 브로드캐스트가 진행이 됨.
(2) 그러면 학생 3명이 학교로부터 메시지를 받게 된다.
(3) 이렇게 학사정보를 학생에게 전달하는 기능을 구현했는데 학사정보와 학생은 어떠한 의존 관계를 가지지 않는다. 코드도 굉장히 심플하게 끝남. 이것이 언리얼 델리게이트가 가지는 장점이다.

**정리

언리얼 C++ 델리게이트

1. 느슨한 결합(Loose Coupling)이 가지는 장점
   - 향후 시스템 변경 사항에 대해 손쉽게 대처할 수 있음.
2. 느슨한 결합(Loose Coupling)으로 구현된 발행 구독 모델의 장점
   - 클래스는 자신이 해야 할 작업에만 집중할 수 있음.
   - 외부에서 발생한 변경 사항에 대해 영향받지 않음.
   - 자신의 기능을 확장하더라도 다른 모듈에 영향을 주지 않음.
3. 언리얼 C++의 델리게이트의 선언 방법과 활용
   - 몇 개의 인자를 가지는가?
   - 어떤 방식으로 동작하는가? (MULTICAST 사용 유무 결정)
   - 언리얼 에디터와 함께 연동할 것인가? (DYNAMIC 사용 유무 결정)
   - 이를 조합해 적합한 매크로 선택

데이터 기반의 디자인 패턴을 설계할 때 유용하게 사용

느슨한 결합이 가지는