

# 1단계\_08. C++설계 #2 컴포지션

📅 수강일	@2023/03/31
➦ 이름	 전영재
🔍 멘토	Min-Kang Song, 현웅 최
⚙️ 작성 상태	In progress
📁 단계	1단계
☑️ 강의 시청 여부	<input checked="" type="checkbox"/>

## Contents



강의 제목

[1] 엔리얼 오브젝트의 컴포지션

[2] 컴포지션 실습

엔리얼 엔진에서의 오브젝트 조합 방법

[3] Enum의 사용

C++ 오브젝트 포인터 프로퍼티

0n. C++설계#2 컴포지션 강의 과제

## 강의 제목



### 강의 목표

- 엔리얼 C++의 컴포지션 기법을 사용해 오브젝트의 포함 관계를 설계
- 엔리얼 C++이 제공하는 확장 열거형 타입의 선언과 활용 방법의 학습

## [1] 엔리얼 오브젝트의 컴포지션

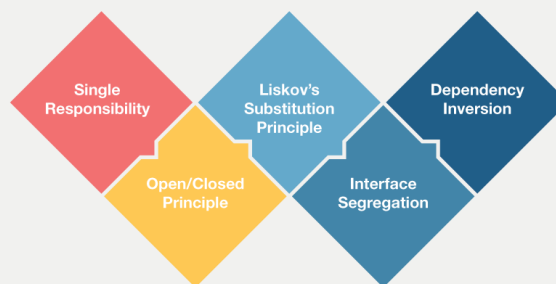
객체 지향 설계 중 **합성**으로, 성질이 다른 두 객체의 **소유 관계**를 의미한다.

**하나의 클래스가 다른 클래스를 소유**하며 이를 통해 복합적인 기능을 가진 거대한 클래스를 효과적으로 설계 가능하다.



다음 **모던 객체 설계 기법(SOLID)**을 유의하며 컴포지션을 설계해야 한다.


# S.O.L.I.D.



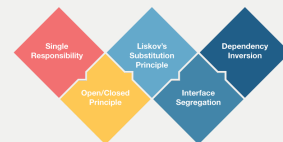
SOLID원칙은 몇 번을 강조해도 부족하다!

SOLID 원칙, 어렵지 않다!

객체지향 프로그래밍 설계 원칙에 대해 알아보기

 [https://velog.io/@haero\\_kim/SOLID-%EC%9B%90%EC%B9%99-%EC%96%B4%EB%A0%B5%EC%A7%80-%EC%95%8A%EB%8B%A4](https://velog.io/@haero_kim/SOLID-%EC%9B%90%EC%B9%99-%EC%96%B4%EB%A0%B5%EC%A7%80-%EC%95%8A%EB%8B%A4)

# S.O.L.I.D.



## [2] 컴포지션 실습

학교 상황으로 컴포지션을 만들어보자!

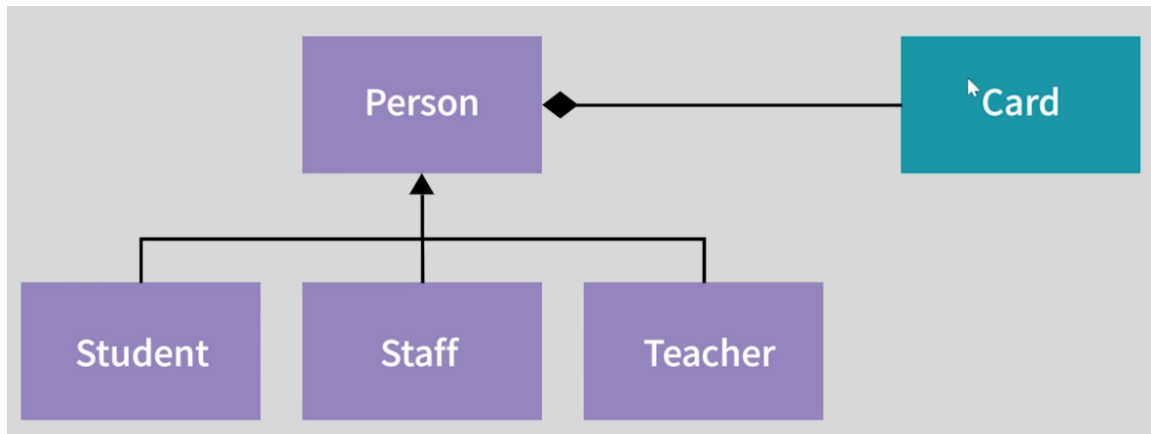
학교 구성원이 사용할 **‘출입증’**을 만든다고 할 때, 이를 어떻게 구현할 것인가?

### 1. Person에서 직접 구현

⇒ 새로운 형태의 구성원(ex. 출입증이 없는 외부 연수생)이 생기면 Person을 수정해야 한다.

## 2. 출입증을 **컴포지션**으로 분리한다.

⇒ 하지만, 그냥 구현하면 안되고 모던객체지향 고급기법을 사용해야한다. 왜?



## 언리얼 엔진에서의 오브젝트 조합 방법

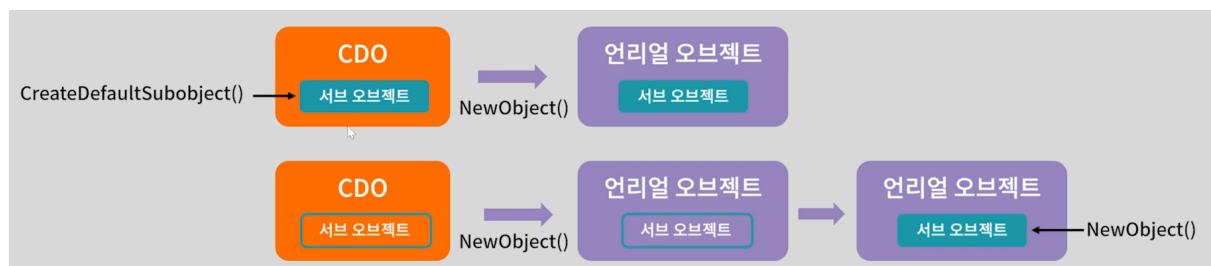
두 가지 방법이 존재한다.

### 1. CDO에 미리 언리얼 오브젝트를 생성하고 조합.

⇒ `CreateDefaultSubobject()` 함수를 사용하는 것으로, 내가 항상 하던 그것이 **CDO에서 미리** 만드는 과정이었다!!

### 2. CDO에선 빈 포인터만 생성하고 런타임에서 언리얼 오브젝트를 생성해 조합

⇒ **런타임** 도중 `NewObject()` 를 통해 오브젝트를 생성하여 포인터에 할당





내가 소유한 언리얼 오브젝트를 **Subobject**라고 한다.  
 반대로 나를 소유하는 언리얼 오브젝트를 **Outer**라고 한다.  
 ⇒ 학생이 봤을때, 카드는 학생의 Subobject  
 ⇒ 카드가 봤을때, 학생은 카드의 Outer

## [3] Enum의 사용

멤버변수 구분의 가독성을 위해서 열거형을 사용하는 것이 좋다.

C++에서는 **Enum**이라는 열거형 클래스를 선언해줄 수 있는데, 언리얼엔진에서는 여기서 더 나아가 **UENUM** 매크로를 통해 에디터와 연동하고 **Meta지정자**를 사용할 수 있게 된다.

ex) Student = 1 UMETA(DisplayName = "For Student"), ...

## C++ 오브젝트 포인터 프로퍼티

UE4까지는 C++에서 코드를 작성할때, 헤더에서 멤버변수를 선언하는 경우 **전방선언**을 통해 원시 오브젝트 포인터를 사용하는 것이 정석이었다.

하지만 **UE5**로 넘어오면서 템플릿 기반 64비트 포인터 시스템인 **TObjectPtr**를 도입했으며, 이들은 변수 또는 함수로 전달될 때 원시포인터로 변환되므로 기존의 사용법과는 차이가 없다.

그렇기에, cpp에서의 구현은 똑같이 포인터로 하되, 헤더에서의 선언은 **TObjectPtr**를 사용하는 것을 권장한다.



공식 문서에 따르면, 성능에 영향을 미치지 않지만 더 나은 개발 경험을 할 수 있다 한다.

언리얼 엔진 5 마이그레이션 가이드

언리얼 엔진 4 프로젝트로 언리얼 엔진 5로 이주하는 방법 및 요구 사항.

④ <http://docs.unrealengine.com/5.0/ko/unreal-engine-5-migration-guide/>





## Summary

- 언리얼 엔진 C++의 컴포지션에 대한 이해와 사용법
- 언리얼 Enum의 차이와 Meta 지정자 사용 예시
  - C++ 오브젝트 포인터 프로퍼티의 사용 ⇒ `TObjectPtr`

## 0n. C++설계#2 컴포지션 강의 과제



**Q1. 언리얼 오브젝트의 컴포지션을 활용한 예제를 고안해보고 직접 구현해보세요.**

- 두 개의 언리얼 오브젝트 A와 B(가칭)를 선언합니다.
- B는 A의 서브 오브젝트입니다.
- A의 CDO에서 B를 생성합니다.
- NewObject로 A를 생성하면 서브오브젝트인 B도 생성되어 있는지 확인해봅니다.
- B 오브젝트의 포인터를 가져온 후 B의 GetOuter() 함수를 호출해 Outer가 A를 가리키는지 확인해봅니다.