언리얼C++ 설계 II - 컴포지션

(Unreal C++ Design II - Composition)

강의 내용

언리얼 C++만의 컴포지션 기법을 사용해
복잡한 언리얼 오브젝트를 효과적으로 생성하기

강의 목표

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

언리얼 오브젝트의 컴포지션

컴포지션(Composition)

- 객체 지향 설계에서 상속이 가진 Is-A 관계만 의존해서는 설계와 유지보수가 어려움.
- 컴포지션은 객체 지향 설계에서 Has-A 관계를 구현하는 설계 방법
- 컴포지션의 활용
  - 복합적인 기능을 거대한 클래스를 효과적으로 설계하는데 유용하게 사용할 수 있음.

```
class Card
{
public:
  Card(int InId) : Id(InId) {}
  int Id = 0;
};

class Person
{
public:
  Person(Card InCard) : IdCard(InCard) {}

protected:
  Card IdCard;
};
```

모던 객체 설계 기법과 컴포지션

- 좋은 객체지향 설계 패턴을 제작하기 위한 모던 객체 설계 기법 (SOLID)
- Single Responsibility Principle (단일 책임 원칙)
  - 하나의 객체는 하나의 의무만 가지도록 설계한다.
- Open-Closed Principle (개방 폐쇄 원칙)
  - 기존에 구현된 코드를 변경하지 않으면서 새로운 기능을 추가할 수 있도록 설계한다.
- Liskov Substitution Principle (리스코프 치환 원칙)
  - 자식 객체를 부모 객체로 변경해도 작동에 문제 없을 정도로 상속을 단순히 사용한다.
- Interface Segregation Design (인터페이스 분리 원칙)
  - 객체가 구현해야 할 기능이 많다면 이들을 여러 개의 단순한 인터페이스들로 분리해 설계한다.
- Dependency Injection Principle (의존성 역전 원칙)
  - 구현된 실물보다 구축해야 할 추상적 개념에 의존한다.

모던 객체 설계 기법의 설계 핵심은 상속을 단순화하고,

단순한 기능을 가진 다수의 객체를 조합해 복잡한 객체를 구성하는데 있음.

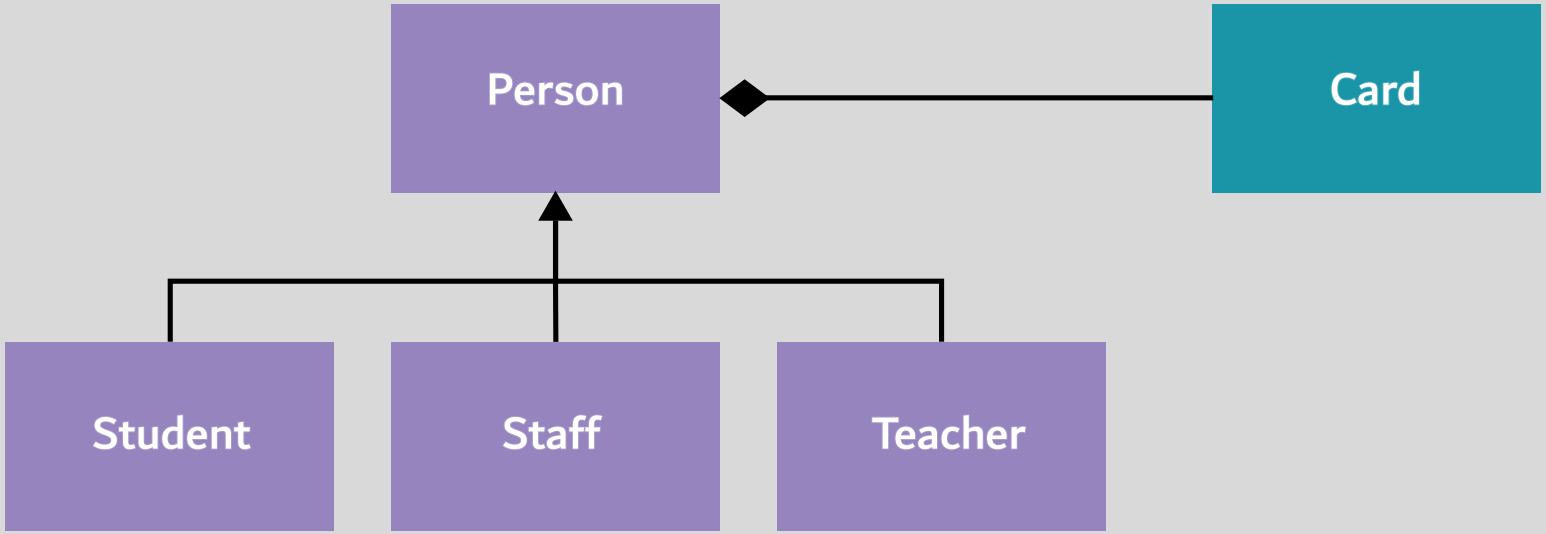
컴포지션 설계 예시

- 학교 구성원 시스템의 설계 예시
  - 학교 구성원을 위해 출입증을 만들기로 한다.
  - 출입증은 Person에서 구현해 상속시킬 것인가? 아니면 컴포지션으로 분리할 것인가?
- Person에서 직접 구현해 상속시키는 경우의 문제
  - 새로운 형태의 구성원이 등장한다면(예를 들어 출입증이 없는 외부 연수생) Person을 수정할 것인가?
  - 상위 클래스 Person을 수정하면, 하위 클래스들의 동작은 문제 없음을 보장할 수 있는가?
- 따라서 설계적으로 출입증은 컴포지션으로 분리하는 것이 바람직함.
- 그렇다면 컴포지션으로만 포함시키면 모든 것이 해결될 수 있는가?

효과적인 설계를 위해 프로그래밍 언어가 제공하는 고급 기법을 활용해야 함.

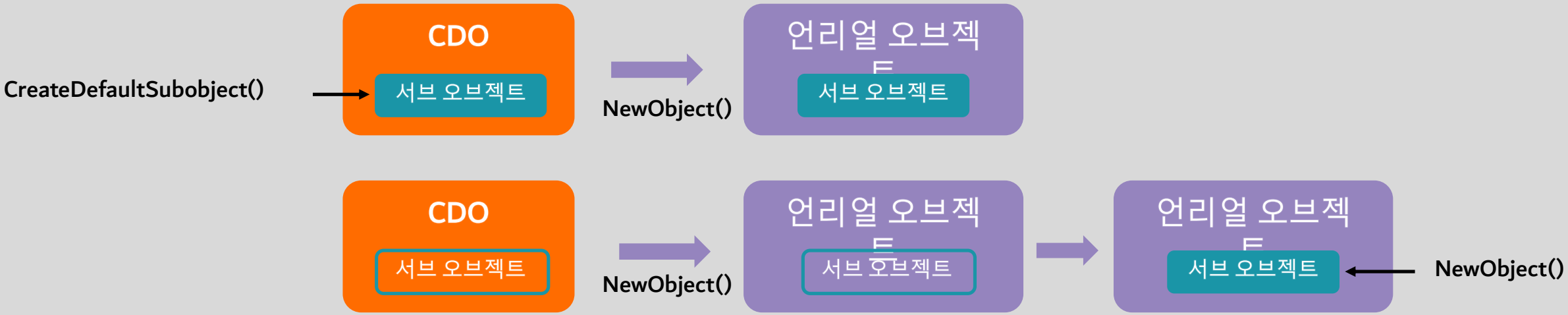
예제를 위한 클래스 다이어그램

- 학교 구성원임을 증명하는 출입증 카드의 부여
  - 학생, 교사, 직원 모두가 상시 지니고 있음
  - 향후 확장성을 고려해 컴포지션으로 구현함.



언리얼 엔진에서의 컴포지션 구현 방법

- 하나의 언리얼 오브젝트에는 항상 클래스 기본 오브젝트 CDO가 있다.
- 언리얼 오브젝트간의 컴포지션은 어떻게 구현할 것인가?
- 언리얼 오브젝트에 다른 언리얼 오브젝트를 조합할 때 다음의 선택지가 존재한다.
  - 방법 1 : CDO에 미리 언리얼 오브젝트를 생성해 조합한다. (필수적 포함)
  - 방법 2 : CDO에 빈 포인터만 넣고 런타임에서 언리얼 오브젝트를 생성해 조합한다. (선택적 포함)
- 언리얼 오브젝트를 생성할 때 컴포지션 정보를 구축할 수 있다.
  - 내가 소유한 언리얼 오브젝트를 Subobject라고 한다.
  - 나를 소유한 언리얼 오브젝트를 Outer라고 한다.



정리

컴포지션을 활용한 언리얼 오브젝트 설계

1. 언리얼 C++은 컴포지션을 구현하는 독특한 패턴이 있다.
2. 클래스 기본 객체를 생성하는 생성자 코드를 사용해 복잡한 언리얼 오브젝트를 생성할 수 있음.
3. 언리얼 C++ 컴포지션의 Has-A 관계에 사용되는 용어
   - 내가 소유한 하위 오브젝트 Subobject
   - 나를 소유한 상위 오브젝트 : Outer
4. 언리얼 C++이 제공하는 확장 열거형을 사용해 다양한 메타 정보를 넣고 활용할 수 있다.

언리얼 **C++**의 컴포지션 기법은

게임의 복잡한 객체를 설계하고 생성할 때 유용하게 사용된다.