1단계_07. C++설계 #1 인터페이스

수강일	@2023/03/30
이름	전영재
멘토	Min-Kang Song,현웅 최
작성 상태	In progress
단계	1단계
강의 시청 여부	☑

Contents

C++설계 #1 인터페이스

💡 **강의 목표**

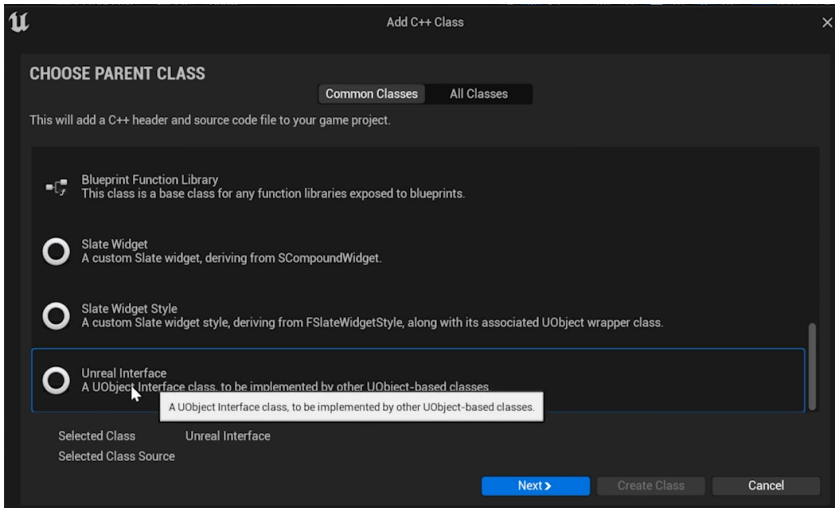
- 언리얼 C++ 인터페이스 클래스를 사용해 안정적으로 클래스를 설계하는 기법의 학습

---

[1] 인터페이스

객체가 **반드시 구현해야 할 행동을 지정**하는데 활용되며 이를 통해 **다형성의 구현**과 **의존성의 분리**가 가능하다.

기존 C++에서는 인터페이스를 지원하지 않지만, 언리얼엔진에서는 **언리얼 C++인터페이스**를 제공한다.



언리얼 인터페이스의 특징

- 언리얼 인터페이스를 생성하면 '**U 타입클래스**'와 '**I 인터페이스클래스**'가 생성된다.
  이때, 타입 정보는 **런타임에서 인터페이스 구현 여부를 파악하는 용도**로 사용되는 것이기에, 실질적으로 여기서 작업을 할 필요는 없다. 모든 작업은 인터페이스 클래스에서 진행된다

```cpp
#include "LessonInterface.generated.h"

// This class does not need to be modified.
UINTERFACE(MinimalAPI)
class ULessonInterface : public UInterface
{
    GENERATED_BODY()
};

/**
 *
 */
class UNREALINTERFACE_API ILessonInterface
{
    GENERATED_BODY()

public:
    virtual void DoLesson() = 0;
};
```

- Java, C#은 추상타입으로만 선언할 수 있지만, 언리얼엔진은 실질적으로 클래스를 작성하는 것이기에 인터페이스 클래스에서도 구현이 가능하다.
  - 모던 객체지향과는 다르지만, 언리얼 C++에서는 이걸 활용하는 부분이 종종 있다. 재량껏 사용하자.
- C++ 클래스는 단일 상속 구조이기에 인터페이스는 `Super::` 키워드를 사용할 수 없다.

- 이 경우 `ILessonInterface ::DoLesson();` 이런 식으로 직접 불러와야 한다.

> 💡 언리얼은 형변환을 실패할 경우 nullptr을 반환하기에 안전한 로직 작성이 가능하다.
> 그렇기에 인터페이스로 형변환을 시도하면 성공여부에 따라, 해당 클래스의 인터페이스 구현 여부를 알 수 있다.
>
> ```
> for(const auto Person : persons)
> {
>   IInterface* LessonInterface = Cast<IInterface>(person);
>   if(LessonInterface) { UE_LOG(LogTemp, Warning, TEXT("수업에 참여할 수 있다!"))};
>   else { UE_LOG(LogTemp, Warning, TEXT("캐스팅실패 = 수업참여 못함"))};
> }
> ```

📌 **Summary**

- 인터페이스를 통해 **반드시 구현**해야 하는 기능을 지정할 수 있다.

0n. C++설게 #1 인터페이스 강의 과제

> ❓ **Q1. C++언어가 제공하는 다중 상속 기능에서 문제가 될 수 있는 부분은 어떤 것인지 살펴보고, Java와 C#과 같은 후발언어들은 왜 다중 상속을 사용하지 않고 인터페이스를 사용한 제한적 상속을 구현했는지 정리해보세요.**

C++의 다중 상속은 여러 개의 클래스 멤버를 상속받을 수 있는 강력한 방법이다.
그러나 상속받은 여러 개의 클래스 중 같은 이름의 멤버가 존재할 수도 있으며 하나의 클래스를 간접적으로 중복하여 상속받아 문제를 야기할 수도 있다.

그렇기에 Java와 C#은 다중상속을 사용하지 않고도 해당 멤버를 빠뜨리지 않기 위해 강제성을 부여하는 인터페이스를 사용한다.

❓ Q2. Person 클래스의 GetName 함수를 다음과 같이 선언했다. 이들의 차이점을 정리해보시오.

```
FString& GetName() { return Name; }
FString GetName() { return Name; }
const FString& GetName() { return Name; }
FString GetName() const { return Name; }
const FString& GetName() const { return Name; }
```

📖 Reference

코딩교육 티씨피스쿨

4차산업혁명, 코딩교육, 소프트웨어교육, 코딩기초, SW코딩, 기초코딩부터 자바 파이썬 등

TCPSchool

http://www.tcpschool.com/cpp/cpp_inheritance_multiple