**항목 19. 클래스 설계는 타입 설계와 똑같이 취급하자.**

좋은 클래스를 설계하기란 꽤 까다로운 도전입니다.  
좋은 타입은 일단 **문법(syntax)이 자연스럽고, 의미구조(semantics)가 직관적이며, 효율적인 구현이 한 가지 이상 가능해야 하는데**, C++에서는 충분한 고민 없이 클래스 정의를 했다가는 이 3가지중 어느 것도 달성하기 힘들 수 있습니다.

**1. 새로 정의한 타입의 객체 생성 및 소멸은 어떻게 이루어져야 하는가?**

이 부분이 어떻게 되느냐에 따라 클래스 생성자 및 소멸자의 설계가 바뀝니다. 그뿐 아니라 메모리 할당 함수(operator new, operator new[], operator delete, operator delete[] (8))를 직접 작성할 경우에는 이들 함수의 설계에도 영향을 미칩니다.

**2. 객체 초기화는 객체 대입과 어떻게 달라야 하는가?**

생성자와 대입 연산자의 동작 및 둘 사이의 차이점을 결정짓는 요소입니다.  
초기화와 대입을 헷갈리지 않는 것이 가장 중요한데, 각각에 해당되는 함수 호출이 아예 다르기 때문입니다(4)

**3. 새로운 타입으로 만든 객체가 값에 의해 전달되는 경우에 어떤 의미를 줄 것인가?**

어떤 타입에 대해 ‘값에 의한 전달’을 구현하는 쪽은 바로 복사 생성자 입니다.

**4. 새로운 타입이 가질 수 있는 적법한 값에 대한 제약은 무엇으로 잡을 것인가?**

전부는 아니지만, 클래스의 데이터 멤버의 몇 가지 조합 값만은 반드시 유효해야 합니다. 이런 조합을 가리켜 클래스의 불변속성(invariant)이라고 하며, 클래스 차원에서 지켜주어야 하는 부분입니다.

이 불변속성에 따라 클래스 멤버 함수 안에서 해 주어야 할 에러 점검 루틴이 좌우되는데, 특히 생성자, 대입연산자, 각종

“쓰기(setter)” 함수는 불변속성에 많이 좌우됩니다. 그뿐 아니라 불변속성은 여러분의 함수가 발생시키는 예외에도 영향을 미치며, 혹시나 여러분이 예외 지정(exception specification)을 쓴다면 그 부분에도 영향을 줍니다.

**5. 기존의 클래스 상속 계통망(inheritance graph)에 맞출 것인가?**

이미 갖고 있는 클래스로부터 상속을 시킨다고 하면, 당연히 여러분의 설계는 이들 클래스에 의해 제약을 받게 됩니다.  
특히 멤버 함수가 가상인가 비가상인가의 여부가 가장 큰 요인입니다  
여러분이 만든 클래스를 다른 클래스들이 상속할 수 있게 만들자고 결정했다면, 이에 따라 멤버 함수의 가상 함수 여부가 결정됩니다. 특히 소멸자가 그렇습니다.

**6. 어떤 종류의 타입 변환을 허용할 것인가?**

여러분의 타입과 다른 타입 사이에 변환수단이 있어야 할까요?  
T1 타입의 객체를 T2 타입의 객체로 **암시적으로(implicit)** 변환되도록 만들고 싶다면, T1 클래스에 타입 변환 함수를 하나 넣어두든가(이를 테면 operator T2) 아니면 인자 한 개로 호출될 수 있는 비명시호출(non-exlicit) 생성자를 T2 클래스에 넣어두어야 할 것입니다.  
명시적(explicit) 타입 변환만 허용하고 싶을 경우에는, 해당 변환을 맡는 별도 이름의 함수를 만들되 타입 변환 연산자 혹은 (인자 하나로 호출될 수 있는) 비명시호출 생성자는 만들지 말아야 할 것이고요(명시적/암시적 변환은 15)

**7. 어떤 연산자와 함수를 두어야 의미가 있을까?**

여러분의 클래스 안에 선언할 함수가 바로 여기서 결정됩니다. 어떤 것들은 멤버 함수로 적당할 것이고, 또 몇몇은 그렇지 않을 것입니다.

**8. 표준 함수들 중 어떤 것을 허용하지 말 것인가?**

private로 선언해야 하는 함수가 여기에 해당

**9. 새로운 타입의 멤버에 대한 접근권한을 어느 쪽에 줄 것인가?**

어떤 클래스 멤버를 public, protected, private 영역에 둘 것인가를 결정하는데 도움을 주게 될 질문이다.   
또한 프렌드로 만들어야 할 클래스 및 함수를 정하는 것은 물론이고, 한 클래스를 다른 클래스에 중첩시켜도 되는가에 대한 결정을 내리는 데도 이 질문이 여러분을 거들어 줄 것입니다.

**10. ‘선언되지 않은 인터페이스’로 무엇을 둘 것인가?**

여러분이 만들 타입이 제공할 보장이 어떤 종류일까에 대한 질문으로서,   
**보장할 수 있는 부분은 수행 성능 및 예외 안전성(29) 그리고 자원 사용(잠금 및 동적 메모리 등)** 입니다.  
이들에 대해 여러분이 보장하겠다고 결정한 결과는 클래스 구현에 있어서 제약으로 작용하게 됩니다.

**11. 새로 만드는 타입이 얼마나 일반적인가?**

실상은 타입 하나를 정의하는 것이 아닐지도 모릅니다.  
여러분이 정의하는 것이 **동일 계열의 타입군(family of types)** 전체일지도 모른다고요

진짜 그렇다면 여러분이 원하는 것은 ‘새로운’ 클래스가 아닙니다. ‘새로운’ 클래스 **템플릿**을 정의해야 할 것입니다.

**12. 정말로 꼭 필요한 타입인가?**

기존의 클래스에 대해 기능 몇 개가 아쉬워서 파생 클래스를 새로 뽑고 있다면, 차라리 간단하게 비 멤버 함수라든지 템플릿을 몇 개 더 정의하는 편이 낫습니다.

**요약.**

- 클래스 설계는 타입 설계 입니다. 새로운 타입을 정의하기 전에, 이번 항목에 나온 모든 고려사항을 빠짐없이 점검해 보십시오.