객체 지향 프로그래밍 토론 정리(가상함수와 추상 클래스)

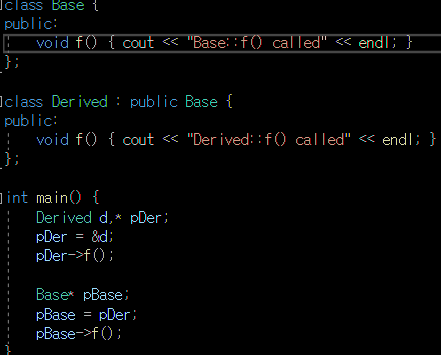
202004103 정원석, 202004904 이종하

1 함수 재정의 와 오버라이딩의 차이

오버라이딩의 경우에는 원래의 함수가 존재감을 잃어 함수가 호출되는 실행 시간에 동적 바인딩이 일어나지만 함수 재정의는 각각 컴파일 타임에 지정을 해주어 컴파일 시간에 결정된 함수가 단순히 호출되는 정적 바인딩이 일어난다. 오버라이딩을 하는 경우에는 가상 함수를 사용하면 된다.

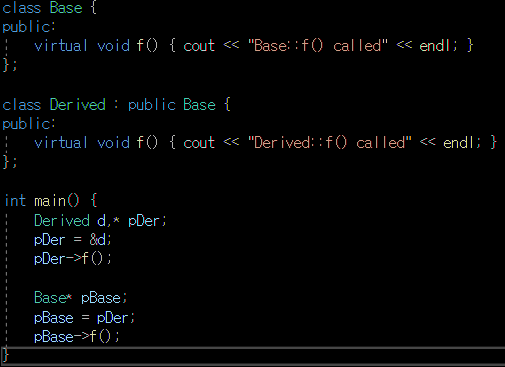
Java의 경우 함수 재정의가 오버라이딩이기 때문에 무조건 동적 바인딩이 일어난다.

함수 재정의의 경우(가상함수 미사용)



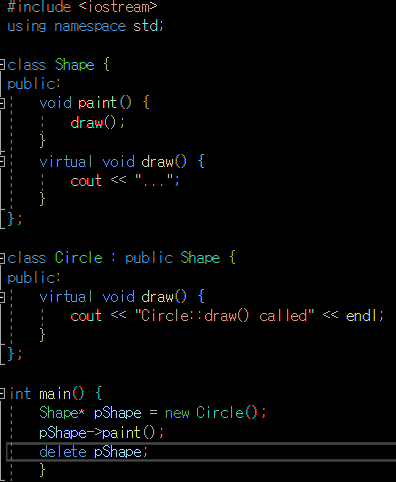


오버라이딩의 경우 (가상함수 사용)





오버라이딩된 함수를 호출하는 동적 바인딩

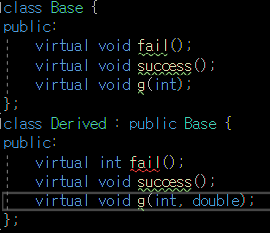




가상 함수를 사용해 오버라이딩하여 동적바인딩이 일어나 Shape속 draw가 아닌 Circle속 draw로 호출되었다.

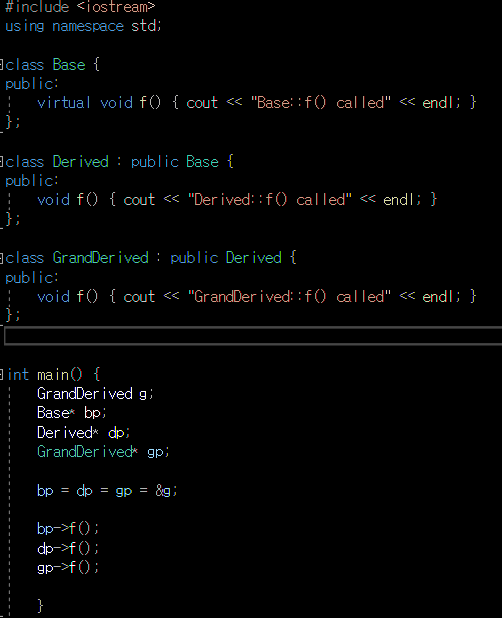
2. 오버라이딩의 성공 조건

가상 함수 이름, 매개 변수 타입과 개수, 리턴 타입이 모두 일치해야 한다.



fail()은 리턴 타입이 다름  
succese()는 성공  
g(int, double)은 오버라이딩이 아닌 오버 로딩이다.

3. 상속이 반복되는 경우 가상 함수 호출

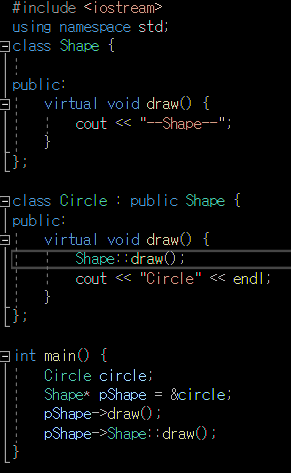


밑에 f() 둘 다 virtual이 생략되어 있지만 가상 함수를 사용한 오버라이딩이기 때문에 먼저 Base는 존재감을 잃었다. 근데 Derived도 참조되는 상황이 나와 Derived도 가상 함수를 사용한 오버라이딩을 한 것이기 때문에 존재감을 잃고 GrandDerived만 호출되는 상황이 나올 것이라고 생각된다.

결과



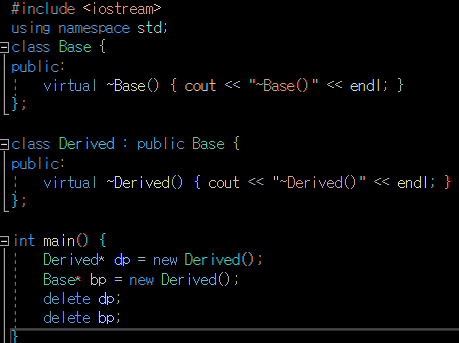
4. 범위 지정 연산자를 사용한 가상 함수 호출



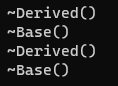


가상 함수를 사용하여 오버라이딩 하더라도 범위를 지정하고 사용하면 호출할 수 있다.

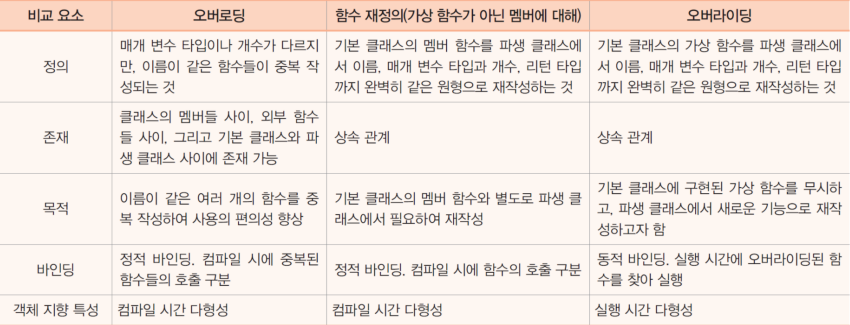
5. 소멸자를 가상 함수로 선언



가상 함수를 사용하여 동적 바인딩이 발생해 Derived부터 소멸하여 소멸자를 먼저 발생하지만 Base도 같이 소멸하기 때문에 그 다음 Base도 소멸자를 실행할 것이다.

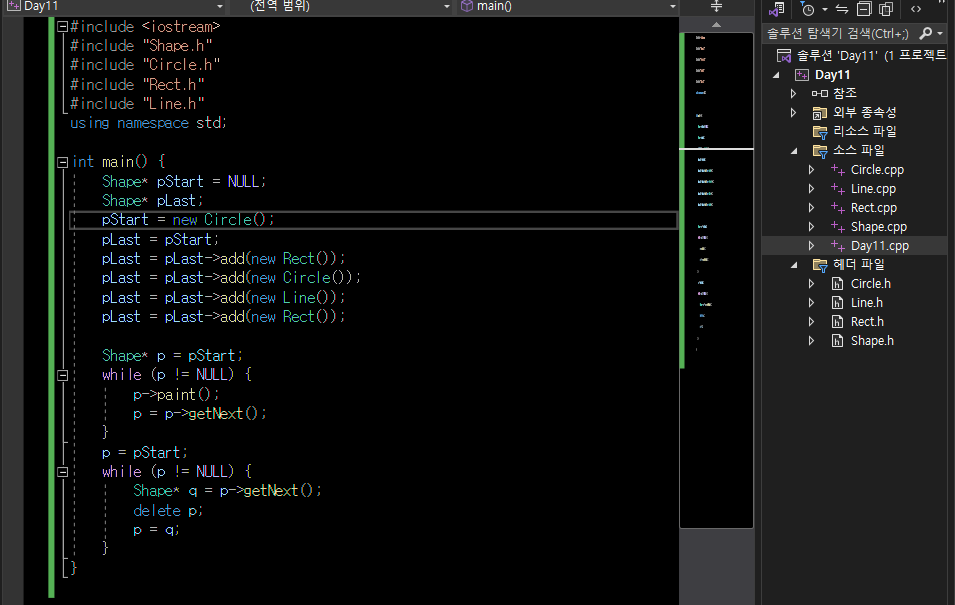


오버로딩, 함수 재정의(가상 함수가 아닌), 오버라이딩

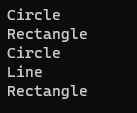


항상 사용하려는 목적과 방법을 이해하고 사용해야 자신이 원하는 방식으로 구현을 할 수 있다.

6. 파생 클래스의 가상 함수 실행



#include "Shape.h" #include "Circle.h" #include "Rect.h" #include "Line.h"를 통해 클래스를 호출해 사용한다.  
pStart에서 첫 객체를 만들고 pLast를 pStart에 연결한다. 시작을 설정하였으니 add(\* p)를 함으로써 각각의 객체를 연결하여 링크드리스트를 만들었다.  
paint()로 실행하면 오버라이딩 되어 각각의 객체에 맞는 draw가 실행되고 링크드리스트로 연결되어 있어 getNext()를 이용하여 다음을 지정하여 넘어간다.



예제 9 – 7 구현)

