**항목 36. 상속받은 비가상 함수를 파생 클래스에서 재정의 하는 것은 절대 금물**

D라는 이름의 클래스가 B라는 이름의 클래스로부터 public 상속에 의해 파생되었고, B클래스에는 mf라는 이름의 public 멤버 함수가 정의되어 있다고 가정합시다. Mf의 매개변수나 반환 타입은 중요한 사항이 아니기 때문에, 그냥 둘 다 void라고 한다.

|  |
| --- |
| class B  {  public:  void mf();  };  class D : public B { ... };  D x; // x는 D타입의 객체 입니다  B \*pB = &x; //x의 포인터  pB->mf(); //B타입 포인터를 통해 mf 호출합니다  D \*pD = &x; //x의 포인터  pD->mf(); //D타입 포인터를 통해 mf 호출합니다 |

양쪽의 경우에서 한결 같이 x 객체로부터 mf멤버 함수를 호출하고 있기 때문이다, 함수도 객체도 같으니, 동작도 같아야 한다. 이치적으로 그래야 하지만 다를 수 있다는 것이 문제 이다. 특히, mf가 비가상 함수이도 D클래스가 자체적으로 mf 함수를 또 정의하고 있으면 위와 같은 황당한 동작이 나오게 됩니다.

|  |
| --- |
| class D : public B  {  public:  void mf(); //이름 은폐  };  pB->mf(); //B::mf 호출합니다  pD->mf(); //D::mf 호출합니다 |

이렇게 동작하는 이유는 B::mf, D::mf 등의 비 가상 함수는 정적 바인딩(static binding)으로 묶이기 때문입니다. 반면에 가상 함수의 경우엔 동적 바인딩(dynamically binding)으로 묶입니다. Mf가 가상함수 였다면 mf가 pB에서 호출되던 pD에서 호출되던 D::mf가 호출됩니다. 분명히 D객체 인데도 불구하고 객체를 가리키는 포인터의 타입에 따라서 호출되는 함수가 다릅니다.

여기서 설명 하고자 하는 것은 상속 받은 비 가상 함수를 파생 클래스에서 재정의 하지 말자라는 것은 문법적으로 안 된다는 것이 아니라 가이드 라인에 가깝습니다. Public 상속의 의미는 is-a 관계 입니다.

**B객체에 해당되는 모든 것들이 D객체에 그대로 적용됩니다. 왜냐하면 모든 D객체는 B객체의 일종이기 때문입니다.**

**B에서 파생된 클래스는 mf 함수의 인터페이스와 구현을 모두 물려받게 됩니다. Mf는 B클래스에서 비가상 멤버 함수이기 때문입니다. D에서 mf를 재정의하면 설계에 모순이 생겨 납니다.**

**요약**

- 상속 받은 비가상 함수를 재정의하는 일은 절대로 하지 맙시다