항목 27: 캐스팅은 절약, 또 절약! 잊지 말자

**캐스팅은 여러 가지 문제를 발생 시킬 수 있습니다**. 어떤 문제들을 일으킬 수 있는지 살펴보고 해결 방법에 대해 알려드리겠습니다.

**문제 1.**

**(\*this)를 기본 클래스 객체로 캐스팅하면 사본이 생성 된다.**

예제)

*class Window{ //기본 클래스*

*public:*

*virtual void onResize( ){ . . . }// 기본 클래스의 onResize 구현 결과*

*. . .*

*};*

*class SpecialWindow: public Window { // 파생 클래스*

*public:*

*virtual void onResize( ){ // 파생 클래스의 onResize 구현 결과*

***static\_case<Window>(\*this).onResize( );***

***// \*this를 Window로 캐스팅하고 그것의 onResize를 호출***

*}*

파생 클래스의 onResize가 하고 싶은 일은 우선 기본 클래스버전의 onResize를 호출 하고 SpecialWindow에서만 필요한 작업을 수행 하는 것입니다. 그러나 캐스팅 부분에서 문제가 발생합니다. **캐스팅이 일어나면서 \*this의 기본 클래스 부분에 대한 사본이 임시적으로 만들어 지는데 이 임시 사본이 기본클래스의 onResize함수를 호출 한 것입니다**. 결국 현재 자신은 onResize함수를 호출하지 않습니다.

해결방법을 알아봅니다.

*class SpecialWindow: public Window {*

*public:*

*virtual void onResize( ) {*

***Window::onResize( ); // \*this에서 Window::onResize를 호출합니다.***

*. . .*

*}*

*}*

일단 캐스팅을 빼버립니다. 간단하게 해결 됩니다

**문제 2.**

*dynamic\_cast는 느립니다. 다음 잘 못 된 코드를 보여드리겠습니다.*

*class Window { . . . }; 기본 클래스*

*class SpecialWindow: public Window {*

*public:*

*void blink( );*

*. . .*

*};*

*shared\_ptr<Window> VPW;*

*VPW winPtrs;*

*. . .*

*for(VPW::iterator iter = winPtrs.begin( ) iter != winPtrs.end( ); ++iter){*

*못 된 코드*

***if(SpecialWindow\* psw = dynamic\_cast<SpecialWindow\*>(iter->get( )))***

*{ psw -> blink( ); }*

*}*

dynamic\_cast를 이용해 기본 클래스의 blink( )를 호출 하고 있습니다. **dynamic\_cast를 사용하면 수행 속도가 느려지고 위와 같이 for문 안에서 여러 번 사용할 경우 더더욱 느려질 것입니다.** 이렇게 사용하지 말고 다음과 같이 할 수 있습니다.

**올바른 방법**

*typedef vector<shared\_ptr<SpecialWindow>> VPSW;*

***VPSW winPtrs; (dynamic\_cast없이 파생 클래스 객체 컨테이너를 담아둠)***

*for(VPST::iterator iter = winPtrs.begin( ); iter != winPtrs.end( ); ++iter)*

*{ // 더 괜찮은 코드*

***(\*iter)->blink( );***

*}*

**또 다른 올바른 방법**

*class Window {*

*public:*

***virtual void blink( ){ } // 기본 구현은 ‘아무 동작 안 하기’ 입니다.****};*

*class SpecialWindow: public Window {*

*public:*

***virtual void blink( ){ . . . } // 이 클래스의 blink함수는 특정 동작을 합니다.***

*};*

*typedef vector<shared\_ptr<Window>> VPW;*

***VPW winPtrs; // 이 컨테이너는 Window에서 파생된 모든 타입의 객체들을 담아요.***

*for(VPW::iterator iter = winPtrs.begin( ); iter != winPtrs.end( ); ++iter){*

***(\*iter)->blink( );***

*}*

**SpecialWindow에만 있던 blink함수를 기본 클래스의 가상함수로 올려 둠으로써 기본 클래스의 파생클래스들 전부 조작할 수 있습니다.** 여러분이 원하는 조작을 가상 함수 집합으로 정리해서 기본 클래스에 넣어두면 됩니다.

* **절대 해서는 안 되는 구조**

*폭포식 dynamic\_case*

*class Window{ . . . };*

*typedef vector<shared\_ptr<Window>> VPW;*

*VPW winPtrs;*

*for(VPW::iterator iter = winPtrs.begin( ); iter != winPtrs.end( ); ++iter )*

*{*

***if(SpecialWindow1\* psw1 = dynamic\_cast<SpecialWindow1\*>(iter->get( ))){…}***

***else if(SpecialWindow2\* psw2 = dynamic\_cast<SpecialWindow2\*>(iter->get( ))){…}***

***else if(SpecialWindow3\* psw3 = dynamic\_cast<SpecialWindow3\*>(iter->get( ))){…}***

***. . .***

*}*

**속도도 둔하고 만약 파생 클래스 하나가 추가되면 위의 폭포식 코드에 계속해서 조건 분기문을 넣어야 합니다.**

정말 잘 작성된 C++ 코드는 **캐스팅을 거의 쓰지 않습니다.** 만약 어쩔 수 없이 캐스팅이 필요하다면 **함수 안에 숨기도록** 하십시오. **또한 구형 스타일의 캐스트 보다는 C++ 스타일의 캐스트를 선호하세요.** 발견하기도 쉽고, 컴파일러가 에러도 잡아주며 설계자가 어떤 역할을 의도했는지 더 자세히 알 수 있습니다.