

## Re: Un-ringing the bell: making parent methods unavailable to children

**Source:**

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vc/2004-12/0135.html>

---

**From:** Ronald Laeremans [MSFT] ([ronaldl\\_at\\_online.microsoft.com](mailto:ronaldl_at_online.microsoft.com))

**Date:** 12/04/04

Date: Fri, 3 Dec 2004 16:44:55 -0800

Your example did not have the method as virtual.

My assumption was that you were asking for a method that would make it impossible for an instance of a derived class to call it, not just less convenient. And the former cannot be done by any method I can think of. Of course making the method private in the derived class will prevent it from being called directly.

Thanks.

Ronald

"Peteroid" <[peter\\_oliphant@msn.com](mailto:peter_oliphant@msn.com)> wrote in message  
news:ubXNJeN2EHA.2608@TK2MSFTNGP10.phx.gbl...

> I see your point, but it does require going out of your way by virtue of  
> 'tricking' the compiler into treating it as an instance of its parent  
> (i.e.,  
> you have to know you are doing it purposefully, you can't 'accidentally'  
> use  
> Add() from an instance of the child class). I tend to avoid pointers and  
> typecasting in my code anyway, they can get you into this kind of trouble  
> (i.e., shoot yourself in the foot by forcing the compiler to treat a  
> pointer  
> to one type of data structure as a pointer to another type via overriding  
> typecasts)...  
>  
> So, based on your example, technically a child instance still can't use  
> Add() unless it is type case as its parent class. And even then, won't the  
> virtual function table still use the child's version of Add() via  
> polymorphism? If not, the trick you mentioned would also override  
> polymorphism in general. Note that in any case that Add() is polymorphed  
> into a NOP for safety, and I could put an 'assert(false)' inside it to make  
> sure it told me at run time if it ever tried to use it (as a last ditch  
> protection, having failed discovering problem at compile time).  
>

```
> Do you have a better way of making a parent public method unavailable to a
> child instance? Love to hear it... :)
>
> [==Peteroid==]
>
>
> "Ronald Laeremans [MSFT]" <ronaldl@online.microsoft.com> wrote in message
> news:%23jejl$x1EHA.3816@TK2MSFTNGP09.phx.gbl...
>> This doesn't work. See below.
>>
>> class Parent_Class
>> {
>> public:
>> void Add( ) { /* do something */;
>> };
>>
>> class Child_Class : public Parent_Class
>> {
>> private:
>> void Add( ) {}; // now child instance can't call Add( ) publicly
>> };
>>
>> int main()
>> {
>> Child_Class ck;
>> ((Parent_Class*)&ck)->Add();
>> }
>>
>> Ronald Laeremans
>> Visual C++ team
>> "Peteroid" <peter_oliphant@msn.com> wrote in message
>> news:%23RILLCY1EHA.1152@TK2MSFTNGP14.phx.gbl...
>>> Is it possible to make a public parent class method unavailable (i.e.,
>>> generate an error at compile time) to a particular child class?
>>>
>>> For example, say a parent class has a public method Add( ). I want to
>>> create
>>> a child class of this parent class that does not have an Add( ) method
>>> (while possibly another child class does).
>>>
>>> I think I figured it out while writing this, so tell me if this is the
>>> 'standard method'. Make the parent class public method virtual, and
>>> then
>>> polymorph it in the child class as private. That is:
>>>
>>> class Parent_Class
>>> {
>>> public:
>>> Add( ) { // do something // }
>>> };
>>>
```

microsoft.public.dotnet.languages.vc: Re: Un-ringing the bell: making parent methods unavailable to children

```
>>> class Child_Class : public Parent_Class
>>> {
>>> private:
>>> Add( ) {} // now child instance can't call Add( ) publically
>>> };
>>>
>>> [==Peteroid==]
>>>
>>>
>>>
>>
>>
>
>
```