

Can I have back my destructors, please?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vc/2008-02/msg00147.html>

- *From:* cctv.star@xxxxxxxxxx
 - *Date:* Wed, 20 Feb 2008 17:19:40 -0800 (PST)
-

In C++ you have RAII idiom that takes care of all your resources – memory, file handles, etc.

In C# you have GC that takes care of the memory, but it looks like you're completely on your own as far as other resources are concerned – just miss this Dispose() call and wait for all kind of interesting things to happen.

For every class that you consume the first thing you should note is whether it implements IDisposable or not, and handle it accordingly.

Now the problem: suppose you use some class, which does not implement IDisposable. All is well. And then, the maintainer of this class decides that he needs to use some other object, which does implement IDisposable, and to make this object a member. Although this should be just an implementation detail and completely private business, he has to make his own class implement IDisposable.

And now your code still compiles fine, but is no longer correct, although the only thing that really has changed is just implementation of a class you use.

In a sense, whether some class implements IDisposable or not is just an implementation artifact for the author, but is indeed an interface change for the consumer.

Actually, I'm still learning C#, so I hope something of what I've written above is incorrect. Otherwise, please advise how to deal with the problem I've mentioned in practice (is there any way, at least, to make the consumer code uncompileable after such change?).

.