

Re: Anders Hejlsberg comment on immutable objects

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-02/1098.html>

From: Michael Sparks (*michael.sparks_at_remove.this.sbcglobal.net*)

Date: 02/09/04

Date: Mon, 09 Feb 2004 06:25:58 GMT

"Magnus Lidbom" <magnus_lidbom@hotmail.com> wrote in message news:t75e20lkshm8hojmrd4oshsm8ccvlrkc2f@4ax.com...

> > *We all acknowledge that in the presence of 'mutable', the compiler can not*

> > *reasonably be expected to enforce the concept of 'const'.*

>

> *Actually, I don't. I expect it to enforce the concept of const, but*

> *not to guarantee that a programmer provides a correct implementation.*

> *In other words, I don't expect it to be able to see that the way that*

> *a programmer modifies a mutable variable constitutes a breach of*

> *contract, but I do expect it to catch the overwhelming majority of*

> *accidental modifications.*

I think the only remaining disagreement is over the definition of the word "contract" and how it applies to immutable. There appear to be two usages. One usage means something that is verifiable by the compiler and/or runtime, like the method signatures of an interface implementation – the contract can be specified in code. The other, more general usage, is about semantics. Most interfaces have a sort of semantic contract that the compiler can't enforce in any way.

Slightly condensed, my argument, again, is that the C++ meaning of immutable (const, mutable keywords) represents only a semantic contract – the kind that the compiler can't enforce 100%. You seem to agree with this by your statement above, and argue that it is acceptable. I also tend to think it is acceptable, since it is what I've been using in C++ for years and it does the job quite well.

A problem could arise, however, when unenlightened users of the language fail to see the subtle difference in the meanings of "contract". The absence of this sort of complexity is, IMO, what Microsoft has strived for with C#, but there are always tradeoffs.

> > 2. *Have 'const' and 'mutable'. Since the compiler and runtime can't*

microsoft.public.dotnet.languages.csharp: Re: Anders Hejlsberg comment on immutable objects

- > >enforce this, it is only marginally more trustworthy than comments and
- > >documentation.
- >
- > I strongly disagree with this estimation as to the trustworthiness of
- > const. See above.

Yes, I was probably being too clinical about it. :-) I actually think it is very useful for practical everyday problems.

>From a completely theoretical perspective, I'd still have to argue that if it's unenforceable even 1% of the time, then it is untrustworthy. I admit that assigning arbitrary values of trustworthiness (such as 'marginal') doesn't accomplish much in the theoretical perspective, as (again, in the theory) trustworthiness seems to be a completely binary quality.

Mike