

Re: Implicit overloads, non static

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2005-05/msg01275.html>

- *From:* Jon Skeet [C# MVP] <skeet@xxxxxxxx>
 - *Date:* Sat, 7 May 2005 06:54:11 +0100
-

Chad Z. Hower aka Kudzu <cpub@xxxxxxxx> wrote:

> Jon Skeet [C# MVP] <skeet@xxxxxxxx> wrote in
> news:MPG.1ce5f6ef1d5d5b1698c0ac@xxxxxxxxxxxxxxxxxxxxxxxx:
>> Well, you've disputed whether it's a matter of implementation rather
>> than principle. The fact that you'd *like* to do something which to me
>
> I didnt necessarily want to do it – the code I posted was just to show
> "What" it would do. Im using the copy constructor behaviour and am just
> fine with that.

Well, you said you'd *like* to do it in your first post – I was going on that.

>> I guess we can only talk about our own experiences, however different
>> they may be. The discrepancy does seem very odd though – I wonder what
>> the cause is.
>
> It's easy – 1 to 5, 0, to 4, goofing up of the condition mostly.

No, I mean I wonder what the cause of the discrepancy is – you seeing it lots, and me seeing it very rarely.

>> Sure, that's clearer – but I wasn't disputing that. You said you prefer
>> your developers to use while loops, and I was saying that I find them
>> harder to read than for loops. Maybe we're talking at cross-purposes
>> though.
>
> Aah – sorry I should have clarified that. I prefer they use while loops
> *when* its not a simple for i = 1 to 5 condition. Certainly not for that
> condition. :)

Right, okay.

>> Only if you'd really find life hard without them. I don't find life
>> hard in Java where I don't have them, therefore they're not that
>> important to me.
>
> I find a lot of things hard in Java. Ack what they call properties... They

Re: Implicit overloads, non static

> are methods! There's no property about it – its a bloody pattern I could
> do in VB... reminds me of OOP in SAS.

No-one's claiming it's anything more than a pattern in Java.

>> And for almost every use, I'd go along with that. Other than the types
>> defined in the language spec, I think it's pretty much always a bad
>> idea to have implicit conversion. Being able to provide *explicit*
>> conversion operators is not quite so bad, although it should still be
>> used with a lot of care.
>
> Here is where we will strongly disagree then for sure.

Making code self-explanatory is a big thing for me. Implicit conversions can definitely be confusing – I've seen *that* lots on the newsgroups. Again, it's something Java doesn't have and which I don't miss when I go back to Java.

>> No, I think we're disagreeing more fundamentally. Your examples have
>> shown that you think it's not always a bad idea to violate the
>> principle that if I do:
>>
>><variable> = <expression>;
>>
>> I should be able to look at <expression> and completely know the value
>> of the variable. That's not a matter of what's important or not, it's a
>> case of what violates entirely reasonable assumptions that pretty much
>> every developer makes subconsciously all the time.
>
> But you *do* know the value. Because its a new value type.
>
> BCD x = 4;
>
> decimal i = x;
>
> i equals 4.
>
> It *is* a new value type – and thus its value is 4. There are many other
> non value type uses for implicits too.

I wasn't talking about implicit conversions in general – I'm talking about the idea from your first post where you did:

```
MyStruct x;  
x.SomethingElse = 5;  
x = 4;
```

and you'd want the value of x *not* to be completely determined by the RHS (which is just 4). You'd want it to be determined by both the RHS and the previous value of x. That's counterintuitive.

Re: Implicit overloads, non static

That's what I was saying was a bad idea on principle, regardless of implementation.

>> Right. I'm not sure I'd call that a copy constructor myself, but that's
>> not terribly important. (I usually wouldn't use the phrase "copy
>> constructor" outside C++, and then only when the parameter to the
>> constructor was of the same type as the object being constructed.)
>
> Yes – true. Its not a copy constructor, its similar behavioru. I introduced
> the term when I was describing some behaviour, and its gone out of context
> since then.

Sure. So long as I know what you mean, it doesn't really matter.

--

Jon Skeet – <skeet@xxxxxxxxxx>

<http://www.pobox.com/~skeet>

If replying to the group, please do not mail me too

.

• **References:**

- ◆ **Implicit overloads, non static**
◇ From: Chad Z. Hower aka Kudzu
 - ◆ **Re: Implicit overloads, non static**
◇ From: Jon Skeet [C# MVP]
 - ◆ **Re: Implicit overloads, non static**
◇ From: Chad Z. Hower aka Kudzu
 - ◆ **Re: Implicit overloads, non static**
◇ From: Jon Skeet [C# MVP]
 - ◆ **Re: Implicit overloads, non static**
◇ From: Chad Z. Hower aka Kudzu
 - ◆ **Re: Implicit overloads, non static**
◇ From: Jon Skeet [C# MVP]
 - ◆ **Re: Implicit overloads, non static**
◇ From: Chad Z. Hower aka Kudzu
 - ◆ **Re: Implicit overloads, non static**
◇ From: Jon Skeet [C# MVP]
 - ◆ **Re: Implicit overloads, non static**
◇ From: Chad Z. Hower aka Kudzu
- Prev by Date: **Re: Implicit overloads, non static**
 - Next by Date: **Re: # Coding Convention**
 - Previous by thread: **Re: Implicit overloads, non static**
 - Next by thread: **Re: Implicit overloads, non static**
 - Index(es):
 - ◆ **Date**
 - ◆ **Thread**