

Re: Local variables – quick question

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-02/6004.html>

From: Jon Skeet [C# MVP] (skeet_at_pobox.com)

Date: 02/27/04

Date: Fri, 27 Feb 2004 08:55:58 -0000

Stefan Turalski (stic) <stefan.turalski@kruk-inkaso.com.pl> wrote:

> > *In what context? You can't use a variable which isn't in scope within*
> > *code.*
>
> *Yes, that is why i can't write:*
> *for(int i=0;i<10;i++){}*
> *Console.WriteLine("{0}",i);*
>
> *And this I understood well, but what if I want to check if there is*
> *possibility to declare class argument int i – only signal for me is compiler*
> *errors ?*

You could declare an instance variable i, yes.

> *I really try but I don't get it (yet ;) why compiler don't allow me to use*
> *'child' scope variable on this level ?*
> *for(int i=0;i<10;i++){}*
> *int i=0;*
>
> *This is as you said variable which isn't in scope – 'child' scope is this*
> *for-loop scope..*

If you have a local variable hiding an instance variable, you can always get to the instance variable using this.i. You wouldn't be able to get to the "outer scope" for another local variable. In addition, I suspect the language designers considered it would make code less readable in general.

> > *Debugger could do it in some why ?*
> > *I don't believe even the debugger will show you a variable out of its*
> *scope.*
> *But there have to be a way to go over this and see what is happening with*
> *this all 'non-used' objects ? I have to go over garbage collector ?*

I'm really not sure what you're trying to do, to be honest. Please give a full description of what you're trying to do and in what context.

>> *Variables aren't "destroyed", but they will no longer be treated as
>> being alive by the garbage collector. In other words, just because a
>> variable comes to the end of its scope doesn't mean that an object it
>> refers to will immediately be garbage collected, but it means that the
>> variable will no longer stop if from being garbage collected. (In fact,
>> when not in debug mode, it's actually after the last use that it's no
>> longer "used".)*
>
> *So when I still try to do something like for-loop in which I use i =
> iterator, and after that I would like to use int i – new declaration. That
> could not be done, because of this object already could be somewhere in
> memory ? I don't get this idea of "A local variable named 'i' cannot be
> declared in this scope because it would give a different meaning to 'i',
> which is already used in a 'child' scope to denote something else"*

No, it's nothing to do with objects being in memory. It's just that the language is defined to prevent local variables from hiding each other.

>>> *– another good link (this one I have read before writing on group – and
> then
>>> I was wondering what: "Scopes can be nested, and an inner scope may
> redeclare the meaning of a
>>> name from an outer scope."
>> It means you can have, say, a local variable with the same name as an
>> instance variable.
>
> Ok, so why my int i; isn't instance variable ??*

Because it's a local variable. An instance variable is one which belongs to the instance; a local variable is one which belongs to the method.

>>> *mean ? I assume that when I write another for() loop with the same int i
>>> iterator I only redeclare it ?
>> No – you should read onto the next sentence:
>> <quote>
>> (This does not, however, remove the restriction imposed by Section 3.3
>> that within a nested block it is not possible to declare a local
>> variable with the same name as a local variable in an enclosing block.)
>> </quote>
>
> I was thinking about something like this
> for(int i=0;i<10;i++){ do sth A..}
> for(int i=10;i>0;i--){ do sth B..}
>
> And this works well, both 'child' scopes are using the same name for
> variable and there is no errors, so another time I ask –> why class variable
> (I think instance ?) isn't allowed as well ?*

It is. You just can't have another *local* variable called i in the scope which includes those two loops.

microsoft.public.dotnet.languages.csharp: Re: Local variables – quick question

> > > *What is the dipper (memory?) background of this 'redeclare' ?*
> > *Not sure what you're after here. Names are just names – by the time*
> > *everything's compiled there may still be a mapping of name to (eg)*
> > *member location or place in the stack for a local variable, but unless*
> > *you're using reflection, the runtime itself won't be using that map*
> > *much.*
> *I was wondering how my previous thing works, if I have two declarations of*
> *int i – there must be a way to forget about this first name and make another*
> *object for second 'i' – and why the same thing couldn't be done for int i –*
> *not in local for-loop scope but at class level.*
>
> *I really try to understand this... really..*

No objects are involved here in the first place, as we're dealing with integers.

You'll have to accept that you just can't have two local variables declared where one is declared in either the same scope or the child scope of the other.

--

Jon Skeet - <skeet@pobox.com>

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

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