
Subject: object programming question

Posted by [markb77](#) on Tue, 04 Dec 2007 20:54:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

I've got a question about object programming technique in IDL. I've started writing objects that extend the functionality of object graphics.. for instance objects that are subclasses of IDLgrImage. These work well.

Now, I'm tempted to start organizing my data into objects as well. This is a widget program with a large amount of different types of data, and rather than keep everything in the info structure I could imagine organizing most of it into several objects that I could keep in the info structure.

Is this normal object programming technique? Do people do this?

thanks

Mark

Subject: Re: Object Programming

Posted by [David Fanning](#) on Wed, 11 May 2011 19:07:06 GMT

[View Forum Message](#) <> [Reply to Message](#)

David Fanning writes:

> Anyway, lesson for today: Don't make a mistake when you are
> writing IDL object programs!

This, by the way, in programs that have MASSIVE amounts of error handling code built into them! I wasn't even reaching my error handling routines! :-(

Cheers,

David

--

David Fanning, Ph.D.

Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: <http://www.idlcoyote.com/>

Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Object Programming

Posted by [Michael Galloy](#) on Wed, 11 May 2011 19:11:29 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 5/11/11 1:04 PM, David Fanning wrote:

> Folks,
>
> I guess it has been awhile since I did some extensive
> object programming, but goodness, IDL doesn't like it
> when you make mistakes!
>
> I suppose I crashed IDL 25-30 times already today, both
> IDL 7.1.1 and IDL 8.1. It seems as though if put a breakpoint
> in an object program and then do a .RESET from the IDL
> command line that IDL is certain to crash.
>
> The error I was tracking down looked something like this:
>
> FUNCTION foo, x, y
>
> obj = Obj_New('foo', x, y)
>
> RETURN, obj
>
> END
>
> Both routines are written in such a way that you can
> pass one or two parameters. If I pass a single parameter:
>
> IDL> obj = FOO(data)
>
> The object INIT method was reporting that I was
> passing *two* parameters (I was checking with N_Params())
> and my program was getting into trouble because I assumed
> y was defined in the code. It was actually passing an
> undefined variable into the object INIT method. Is this
> how IDL has always worked? I could have sworn I've done
> this many, many times in the past without difficulty, but
> perhaps I wasn't using N_Params() to check parameters
> in that code.

The strange problem with N_PARAMS could be related to the automatic wrapper routine creation in IDL 8.0+. Where you getting 2 parameters with 7.1.1 also? Are you sure you were getting your FOO function, and not an auto-created wrapper for the FOO class? That auto-created wrapper might do something weird with positional parameters.

> Anyway, lesson for today: Don't make a mistake when you are
> writing IDL object programs!

Mike

--

Michael Galloy

www.michaelgalloy.com

Modern IDL, A Guide to Learning IDL: <http://modernidl.idldev.com>

Research Mathematician

Tech-X Corporation

Subject: Re: Object Programming

Posted by [Michael Galloy](#) on Wed, 11 May 2011 19:12:45 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 5/11/11 1:04 PM, David Fanning wrote:

> Folks,

>

> I guess it has been awhile since I did some extensive
> object programming, but goodness, IDL doesn't like it
> when you make mistakes!

>

> I suppose I crashed IDL 25-30 times already today, both
> IDL 7.1.1 and IDL 8.1. It seems as though if put a breakpoint
> in an object program and then do a .RESET from the IDL
> command line that IDL is certain to crash.

>

> The error I was tracking down looked something like this:

>

> FUNCTION foo, x, y

>

> obj = Obj_New('foo', x, y)

>

> RETURN, obj

>

> END

Wait (never mind my other post), you are passing two parameters to
Foo::init!

> Both routines are written in such a way that you can
> pass one or two parameters. If I pass a single parameter:

>

> IDL> obj = FOO(data)

>

> The object INIT method was reporting that I was
> passing *two* parameters (I was checking with N_Params())
> and my program was getting into trouble because I assumed
> y was defined in the code. It was actually passing an

> undefined variable into the object INIT method. Is this
> how IDL has always worked? I could have sworn I've done
> this many, many times in the past without difficulty, but
> perhaps I wasn't using N_Params() to check parameters
> in that code.
>
> Anyway, lesson for today: Don't make a mistake when you are
> writing IDL object programs!

Mike

--

Michael Galloy
www.michaelgalloy.com
Modern IDL, A Guide to Learning IDL: <http://modernidl.idldev.com>
Research Mathematician
Tech-X Corporation

Subject: Re: Object Programming
Posted by [David Fanning](#) on Wed, 11 May 2011 19:23:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

Michael Galloy writes:

> Wait (never mind my other post), you are passing two parameters to
> Foo::init!

I guess that's right. I usually check to see if parameters
are undefined or not. In this case, because the actual programs
are wrappers to the PLOT command, I wanted to know how many
there were. Odd that I haven't run into this before, though. :-)

Cheers,

David

--

David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: <http://www.idlcoyote.com/>
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Object Programming
Posted by [David Fanning](#) on Wed, 11 May 2011 22:47:56 GMT
[View Forum Message](#) <> [Reply to Message](#)

David Fanning writes:

- > I guess it has been awhile since I did some extensive
- > object programming, but goodness, IDL doesn't like it
- > when you make mistakes!

Wow! You have to be an absolute masochist to write object programs in IDL. I'm beginning to think seriously about installing IDL 6.4 again. The two latest crashes haven't just taken down IDL (for the 50th time, at least!), but have taken down my entire machine.

I really don't know what this is. None of these errors are making it to my error handling code. I'm programming absolutely blind, but I am not doing anything particularly unusual. I know my way around IDL objects pretty well. I've never seen anything like this. :-(

Cheers,

David

--

David Fanning, Ph.D.

Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Sepore ma de ni thue. ("Perhaps thos speakest truth.")

Subject: Re: Object Programming

Posted by [penteado](#) on Thu, 12 May 2011 19:41:09 GMT

[View Forum Message](#) <> [Reply to Message](#)

On May 11, 4:23 pm, David Fanning <n...@idlcoyote.com> wrote:

- > I guess that's right. I usually check to see if parameters
- > are undefined or not. In this case, because the actual programs
- > are wrappers to the PLOT command, I wanted to know how many
- > there were. Odd that I haven't run into this before, though. :-)

Do you DG's plot (what I would guess by 'command')?

I ask because I know there are some peculiar pitfalls in NG's plot(): instead of (what I expected) every class inheriting Graphic, and being able to inherit from them, in the usual way, it is a very convoluted system. I suspect the cause was to make it work while minimizing the changes or additions to the way the iTools were organized.

The most confusing part is how the classes (like Plot) are created: The functions (like plot()) are not the usual init functions, they are separate functions that create a Graphic object (itself not created by a init function), informing it of what kind of graphic to make. The graphic() function creates the proper iTool, and an object of the proper class, which is just a wrapper and contains the Graphic object inside.

This is why I gave up on trying to inherit Window for a class, and contained it instead (you can see it at http://www.ppenteado.net/idl/pp_lib/doc/pp_multiplot__define.html). I do not remember whether it would be impossible or just complicated to inherit without changing the code in some of IDL's routines (which would be a very weird inheritance, if I had to change other code).
