Subject: Re: Copying (Duplicating) Objects
Posted by Mark Hadfield on Wed, 05 Jun 2002 21:40:13 GMT
View Forum Message <> Reply to Message

"M. Katz" <MKatz843@onebox.com> wrote in message
news:4a097d6a.0206050708.7dc1b091@posting.google.com...

> After 4 months away from IDL, I can't remember how to duplicate an
> object variable.

That's OK. After 10 years with it, I can't remember it either.

> I know that the simple command, b = a, only makes b and a have the
> same reference--they point to the same object.

Indeed.

> What's the command to make b an entirely new object but identical to
> a.

There isn't one, at least not built-in. The simplest & most robust
(though not foolproof) method of doing this is to SAVE the object to
disk then RESTORE it.

Try searching Google groups for "clone object group:comp.lang.idl-pvwave"

--
Mark Hadfield "Ka puwaha et tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Copying (Duplicating) Objects
Posted by btupper on Thu, 06 Jun 2002 01:08:34 GMT
View Forum Message <> Reply to Message

On Thu, 6 Jun 2002 09:40:13 +1200, "Mark Hadfield"
<m.hadfield@niwa.co.nz> wrote:

>
>> What's the command to make b an entirely new object but identical to
>> a.
>
> There isn't one, at least not built-in. The simplest & most robust
> (though not foolproof) method of doing this is to SAVE the object to
> disk then RESTORE it.
>
Hello,

In Martin Schultz's MGS_BASEOBJECT there is a COPY method that returns
a populated object of the same class as the original; the returned new
object has its own reference id. It's pretty handy for most object
coding I deal with. It's worth a peek.

http://www.mpimet.mpg.de/~schultz.martin/idl/index.html

Ben

Subject: Re: Copying (Duplicating) Objects
Posted by Randall Skelton on Thu, 06 Jun 2002 09:56:38 GMT
View Forum Message <> Reply to Message

On Thu, 6 Jun 2002, Ben Tupper wrote:

>>
>>> What's the command to make b an entirely new object but identical to
>>> a.
> Hello,
>
> In Martin Schultz's MGS_BASEOBJECT there is a COPY method that returns
> a populated object of the same class as the original; the returned new
> object has its own reference id. It's pretty handy for most object
> coding I deal with. It's worth a peek.

I agree with Ben... I too use Martin's copy method and it works very well.
It does have problems when you try and clone a class that has requried
positional parameters though as internally it relies on 'clone = obj_new(
obj_class(self))'. My 'I do not have time to think about this now'
solution was to make classes with no required positional parameters and
write an 'isValid' method that ensures the object is correctly populated
with 'quasi-required' parameters before it is used. Of course, failing to
use positional arguments for required parameters goes against some well
established IDL programming conventions and therefore tends to upset some
of the IDL Expert Programmer's Association Executive Committee members ;)

Subject: Re: Copying (Duplicating) Objects
Posted by Paul Van Delst[1] on Thu, 06 Jun 2002 13:44:38 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
>
> "M. Katz" <MKatz843@onebox.com> wrote in message
> news:4a097d6a.0206050708.7dc1b091@posting.google.com...

>
>> After 4 months away from IDL, I can't remember how to duplicate an
>> object variable.
>
> That's OK. After 10 years with it, I can't remember it either.
>
>> I know that the simple command, b = a, only makes b and a have the
>> same reference--they point to the same object.
>
> Indeed.
>
>> What's the command to make b an entirely new object but identical to
>> a.
>
> There isn't one, at least not built-in. The simplest & most robust
> (though not foolproof) method of doing this is to SAVE the object to
> disk then RESTORE it.

Really? Wouldn't writing your own copy/assign function be more, uh, OO-y? When I create definitions for structures with pointer components in Fortran the very next thing I do is write the Initialize() [nullifies the pointers], Destroy() [disassociates the pointers], Allocate() [associates the pointers], and Copy() [COPIES the entire structure, including the allocation of memory for the pointer components] functions (or methods, to use the obfuscatory OO terminology).

paulv

p.s. And in Fortran, if you wanted to, you could overload an Copy() subroutine with the "=" so that

```
  b = a
```

does the same thing as

```
  CALL Copy( b, a )
```

Some people have expressed a wish that IDL could do something similar to be truly OO-ey. The fact that fortran can do this (to some extent has *always* done it wrt intrinsic functions) has always made me wonder why this capability was thought of as an OO characteristic.

Anyway...

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP/EMC purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: Copying (Duplicating) Objects
Posted by Mark Hadfield on Thu, 06 Jun 2002 21:38:10 GMT
View Forum Message <> Reply to Message

"Randall Skelton" <rhskelto@atm.ox.ac.uk> wrote in message
news:Pine.LNX.4.33.0206061044560.10439-100000@mulligan.atm.o x.ac.uk...

> I agree with Ben... I too use Martin's copy method and it works very
> well. It does have problems when you try and clone a class that has
> requried positional parameters though as internally it relies on
> 'clone = obj_new(obj_class(self))'. My 'I do not have time to
> think about this now' solution was to make classes with no required
> positional parameters and write an 'isValid' method that ensures the
> object is correctly populated with 'quasi-required' parameters
> before it is used. Of course, failing to use positional arguments
> for required parameters goes against some well established IDL
> programming conventions and therefore tends to upset some of the IDL
> Expert Programmer's Association Executive Committee members ;)

Well, I can't comment on that since the IEPAEC has unaccountably failed to invite me to join. But just to spin this thread out a little longer, I would like to make a couple of observations:

- Few, if any, standard IDL classes require parameters during initialisation

- Where standard IDL classes have positional parameters, they tend to "echo" them with keyword parameters.

The first means that you can create a blank object then set its properties later. Allowing for this is good OO practice IMHO. The second lets you take advantage of the power of IDL's keyword handling (inheritance, keyword-structure equivalence).

--
Mark Hadfield "Ka puwaha et tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Copying (Duplicating) Objects
Posted by MKatz843 on Sat, 08 Jun 2002 18:49:41 GMT
View Forum Message <> Reply to Message

Thanks for all the great suggestions. Here's the solution I've cobbled together. It seems to work nicely.

I now define objects with the following specifications

(1) The GetProperty method has an all=all keyword that
  returns a structure containing all of the paramters and
  values of that object. (I know this works best when the
  values are simple and not pointers and containers...)

(2) The Init method has an all=all keyword. If the user provides
  an all structure when the object is first defined, then the
  contents are used to set the initial values of the object.

I defined the following obj_copy() function

```
function obj_copy, a
  a -> GetProperty, all=all  ;Read a structure with all prameters
  b = obj_new(obj_class(a), all=all)  ;Create new identical object
return, b
end
```

While not a completely generalizable function, this will work for me
for now.
I also like the suggestion to make a Copy() method part of all object
definitions.

M. Katz
