Subject: How to re-use object methods
Posted by wlandsman on Sun, 08 Sep 2013 17:30:49 GMT
View Forum Message <> Reply to Message

I learned object programming late in my career and have come to appreciate its many benefits. But one thing I have not figured out is how to make the code re-usable.

I have access to a very large object widget display program.   Much of the widget is for a rather specific application, but some of the buttons, say, "Contrast" could be used in other widget display programs.   So when writing my own widget program, I add a "inherit largeobject" to the object definition so that it can inherit the methods of the large object widget.   But I immediately got conflicts between my object definition parameters and those in the large widget display.   I have to admit I did not try very hard to debug this, in part because I was hesitant for my 500 line program to be inheriting the 50,000 line largewidge object.   I ended up forgetting about inheritance and just cut and pasted the the portions of the largeobject methods I needed to create methods for my own object.

I wonder if the way to make the make the code more re-useable would have been for the author of the large widget program to have made each button in the large widget program into an object, rather than a method.

Thanks, --Wayne

Subject: Re: How to re-use object methods
Posted by David Fanning on Mon, 09 Sep 2013 12:44:46 GMT
View Forum Message <> Reply to Message

Wayne Landsman writes:

>
> I learned object programming late in my career and have come to appreciate its many benefits. But one thing I have not figured out is how to make the code re-usable.
>
> I have access to a very large object widget display program.   Much of the widget is for a rather specific application, but some of the buttons, say, "Contrast" could be used in other widget display programs.   So when writing my own widget program, I add a "inherit largeobject" to the object definition so that it can inherit the methods of the large object widget.   But I immediately got conflicts between my object definition parameters and those in the large
widget display.   I have to admit I did not try very hard to debug this, in part because I was hesitant for my 500 line program to be inheriting the 50,000 line largewidge object.   I ended up forgetting about inheritance and just cut and pasted the the portions of the largeobject methods I needed to create methods for my own object.
>
> I wonder if the way to make the make the code more re-useable would have been for the author of the large widget program to have made each button in the large widget program into an object, rather than a method.

The fact that IDL's objects are implemented as named structures really gets in the way of object inheritance. Foresight and a great deal of care is needed to be sure there are no overlapping field names is the final, combined structure. Since this is nearly impossible to do in practice, it limits IDL objects to "shallow" inheritance trees.

In your case, I think you were wise to pick the parts you needed for your new object and just paste them in, with field name changes, if needed, for the new object.

Once you become more familiar with objects, you will probably switch from writing one "very large object widget" program, to writing many much, much smaller ones that can be combined (but not necessarily inherited) into other large widget programs. That's when objects will start to make sense to you as "reusable" parts of your library.

Cheers,

David

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