Subject: Cleaning up inherited object classes
Posted by MKatz843 on Wed, 03 Dec 2003 19:28:27 GMT
View Forum Message <> Reply to Message

Cleaning up is my least favorite activity. Were my living room an IDL object I'm sure it'd be full of dangling pointer references. Here's a question regarding objects' Cleanup methods and inheritance.

When an object inherits another object, methods can be overridden. So what happens to the CleanUp method? It is special.

If my House object inherits the Living_Room and Bathroom object classes, will a call to HOUSE::CleanUp also call Living_Room::CleanUp and Bathroom::Cleanup when obj_destroy, self is called?

Let me put that another way. Suppose an object class, A, has pointer fields. Unless someone tells me otherwise, I assume it's a good idea two specifically free the pointers in that object's Cleanup routine. Now, suppose another object class, B, inherits A. B has its own pointers to clean up as well, so I write that into its cleanup routine.

It is sufficient to write the Cleanup methods like this?

```
pro Bobj::CleanUp
  ptr_free, self.Bpointer
  obj_destroy, self
end

pro Aobj::CleanUp
  ptr_free, self.Apointer
  obj_destroy, self
end
```

Will Bobj::CleanUp's call to "obj_destroy, self" also call Aobj::Cleanup so that self.Apointer can be freed as the object is destroyed?

Also, does the destruction of an object that contains a pointer field also inherently free the pointer? or is it necessary to specifically ask for that in the Cleanup?

Now if I could only get the House::TakeOutTheTrash method to work reliably my wife would be thrilled.

M. Katz

Subject: Re: Cleaning up inherited object classes
Posted by JD Smith on Wed, 03 Dec 2003 22:12:29 GMT
View Forum Message <> Reply to Message

On Wed, 03 Dec 2003 12:28:27 -0700, M. Katz wrote:

> Cleaning up is my least favorite activity. Were my living room an IDL
> object I'm sure it'd be full of dangling pointer references. Here's a
> question regarding objects' Cleanup methods and inheritance.
>
> When an object inherits another object, methods can be overridden. So
> what happens to the CleanUp method? It is special.
>
> If my House object inherits the Living_Room and Bathroom object classes,
> will a call to HOUSE::CleanUp also call Living_Room::CleanUp and
> Bathroom::Cleanup when obj_destroy, self is called?
>
> Let me put that another way. Suppose an object class, A, has pointer
> fields. Unless someone tells me otherwise, I assume it's a good idea two
> specifically free the pointers in that object's Cleanup routine. Now,
> suppose another object class, B, inherits A. B has its own pointers to
> clean up as well, so I write that into its cleanup routine.
>
> It is sufficient to write the Cleanup methods like this?
>
> pro Bobj::CleanUp
> ptr_free, self.Bpointer
> obj_destroy, self
> end
>
> pro Aobj::CleanUp
> ptr_free, self.Apointer
> obj_destroy, self
> end
>
> Will Bobj::CleanUp's call to "obj_destroy, self" also call Aobj::Cleanup
> so that self.Apointer can be freed as the object is destroyed?
>
> Also, does the destruction of an object that contains a pointer field
> also inherently free the pointer? or is it necessary to specifically ask
> for that in the Cleanup?
>
> Now if I could only get the House::TakeOutTheTrash method to work
> reliably my wife would be thrilled.

You need to clean up dynamic memory in each object which contains any,
which means chaining your calls to Cleanup to superclass(es) --- IDL
never chains for you automatically (unlike some languages you may
know), and OBJ_DESTROY is specifically trapped inside Cleanup (it

seems) to avoid recursive calls.

On the plus side, the lovely HEAP_FREE routine RSI gave us with IDL5.3 does a very nice job of cleaning up large data structures with lots of dynamic data (pointers/objects) tucked into them. Though the manual warns of inefficiencies, in the few cases I've tested, HEAP_FREE is actually faster than the explicit alternative, even with thousands of variables on the heap. I often write a cleanup method as simple as:

```
pro FooClass::Cleanup
   heap_free,self.data
   self->SuperClass::Cleanup
end
```

with the intention of fixing it later. Since I've found it to be just as fast, and far less error prone, I generally just leave it, unless I want to preserve some parts of the data (e.g. shared objects/pointers).

JD
