Subject: Re: Abstract Objects and Methods
Posted by Phillip & Suzanne on Tue, 23 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith wrote:
>
> Phillip David wrote:
>>
>> Does anyone know of any way to make sure a method has been overriden by a
>> child class? The only thought I have is to make the class return a
>> {structure/object/status} that contains a status code, with one specifying
>> that the abstract routine was invoked rather than the concrete subclass of the
>> abstract routine. Any better ideas?
>>
>> Phillip
>
> How about just:
>
> pro Abstract::aMethod, arg1, arg2
> message, 'This is an abstract method and must be overridden in class
> '+obj_class(self)
> end
>
> with the understanding that aMethod not be chained to in the overridding
> method of a subclass? Or am I missing something?

J.D. -- This is perfect. Thanks for the tip. BTW, this also works for making
the Init function abstract.

--- sample ---

```
pro abstract::method
  message, 'Method is abstract in class abstract'
end

function abstract::init
  message, 'Class abstract is abstract and cannot be instantiated'
end

pro abstract__define
  struct = {ABSTRACT, NULL:0b}
end
```

--- end of sample ---

By the way, I also learned another important lesson about objects. Be sure
you define both an init function (even if it only returns the value '1' (i.e.,
success)) and a cleanup routine (even with an empty body). When running IDL

5.1 under Win/NT, a class with an implicit Init and Cleanup takes about 4 seconds for each operation. When the operations are defined explicitly, even with empty bodies, the operations are almost instantaneous.

Phillip

Subject: Re: Abstract Objects and Methods
Posted by J.D. Smith on Tue, 23 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Phillip David wrote:
>
> While looking into using IDL's OO-based technologies, I came up with another
> interesting question. Is there any way in IDL to create an abstract method?
> An abstract method is a method common to all objects of a particular class
> whose implementation is not defined in the parent class. For example, if I
> had a class "Shape", and wanted it to have a method "Area", I would like to
> declare the method Area as abstract because no particular method makes sense.
> For a circle, area = pi * r^2; for a square, area = side^2; for a rectangle,
> area=l*w; .... An abstract method requires subclasses to define the method,
> but doesn't specify the implementation (other than perhaps the calling sequence).
>
> Any class that contains an abstract method must also be abstract. That's an
> easier problem to solve. Here's a basic abstract class:
>
> ----- Sample code -----
>
> function abstract::init
> ok = Widget_Message(/Error, $
> ['Class ABSTRACT cannot be instantiated.', $
> 'It is an abstract class.', $
> 'You must create a subclass of it.'])
> return, 0
> end
>
> pro abstract__define
> struct = {ABSTRACT, NULL:0b}
> end
>
> ----- End sample code -----
>
> When you create a concrete subclass, the init method doesn't call its parent's
> init method (or else it fails), but it returns success when it succeeds.
>
> Does anyone know of any way to make sure a method has been overriden by a
> child class? The only thought I have is to make the class return a
> {structure/object/status} that contains a status code, with one specifying

> that the abstract routine was invoked rather than the concrete subclass of the
> abstract routine. Any better ideas?
>
> Phillip

How about just:

```
pro Abstract::aMethod, arg1, arg2
 message, 'This is an abstract method and must be overridden in class
'+obj_class(self)
end
```

with the understanding that aMethod not be chained to in the overridding method of a subclass? Or am I missing something?

JD

--

J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-4083
206 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|