Subject: Abstract Objects and Methods
Posted by Phillip & Suzanne on Mon, 22 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

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