Subject: Re: Object-Oriented Programming Question
Posted by Peter Stoltz on Fri, 19 Dec 1997 08:00:00 GMT
View Forum Message <> Reply to Message

> There are mainly two ways to put toghether two (or more) classes:
>
> A) Composition: One component of a class is an Object.
> A 'Leg' is a part of an 'Animal':
> pro Animal__Define
> tmp = {ANIMAL, ..., LeftFrontLeg : OBJ_NEW(),...}
> ...
> end
> ...
> function Animal::Init
> ...
> self.LeftFrontLeg = OBJ_NEW('Leg')
> ...
> end
>
> B) Inheritance: One class is a special case of another one.
> A 'Reptil' is a kind of 'Animal':
> pro Reptil__Define
> tmp = {Reptil, ..., INHERITS ANIMAL}
> ...
> end
>
> [some stuff cut out]
>
> In your case, maybe what you need is to define Class A as a Subclass of
> Class B, i.e.,

But, using your example above, you would never inherit Animal from Leg.
You definitly want Leg as a member of the class Animal. An Animal
has a Leg, but an Animal is not a Leg, to use the "has a" versus
"is a" way of looking at it.

I don't think I want to tamper with this. In my case, I want class B
to be part of the member data of class A (not for A to inherit from B).
What I *really* want is from B to be public member data. You can define
public member data in C++, for instance. Then an instance of A would have
access to B's methods, and I wouldn't have to mess up the 'has a' versus
'is a' relationship.

So, my question is whether anyone has a good (i.e. not clumsy) way to
mock the technique of having public data members in a class when the
data member in question is another class. I did the obvious thing of
defining a member function in A that returns the class B, but then one has
to make an extra copy of the member class B every time you want to call

one of its methods via an instance of A. Plus, this way takes extra lines of code, and aesthetically speaking, is not great.

Thanks for the answer, by the way, Evilio...
