Subject: Re: What about real polymorphism ??
Posted by on Fri, 10 Dec 2004 17:25:35 GMT
View Forum Message <> Reply to Message

very clear now, i agree with you saying that : "IDL is entirely type
agnostic, and doesn't require any pre-
declarations in code which uses objects, *everything* is completely
poly-morphic, i.e. all methods are pure virtual methods ", and the example
explain it quite well. Furthermore I have tried to add the same method
"AboutMe" to the <<super-class>> person like this:

```
pro Person::AboutMe
  print,FORMAT= $
        '(%"This doesn`t work and I am not a %d yr old woman, and I don`t
enjoy %s.")', $
        self.age, self.activity
end
```

and the result is the same:
> Dude, I am a 22 yr old man, and I like watching football.
> Pleased to meet you. I am a 31 yr old woman, and I enjoy gardening.

and it means that polymorphism is working, because is using the methos of
man and woman. I agree as well saying "There are no static or class
variables,
and no class methods", I mean IDL doesn't have many things of pure OO
programming so you can't do things like singleton pattern (desing patterns
by Erich Gamma, Richard Helm, ... for more information about this) i think
even IDL couldn't do many of the basics patterns of this paradigm but
anycase it didn't born to
do that, and works quite well in many other things (Visualization, matrix
operators, etc.).

A good reference for OO programming is the book thinking in C++, is for C++
programmers but the ideas of OO programming are very well explained, you can
find it in : http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html

Good thread and thanks all.
Esteban .

news:1102695714.17270.2.camel@turtle.as.arizona.edu...
> On Thu, 2004-12-09 at 18:32 +0100, Antonio Santiago wrote:
>>> Do you want:
>>>

```
>>>  c2->c1::datos
>>>
>>>  ???
>>>
>>>  This will invoke the datos method defined in c1.
>>>
>>>  Karl
>>>
>>
>>  Not exactly.
>>
>>  I want to view C2 as a C1 class object but when i invoque "datos" it
>>  will be execuete the method defined in C2.
>>  The problem is the point of view. I want to view like in java or other
>>  OO language but IDL cant accept this point of view :(
>>
>>  With PERSON, MAN and WOMAN is more clear. I want to view a MAN or a
>>  WOMAN only as a PERSON, something like a cast. Later when i execute
>>  methods over this PERSON it will execute the "datos" method of WOMEN or
>>  MAN depending of its subtype (that's polymorphism).
>>  But i supossed this point of view is not possible in IDL.
>
>  It's very possible:
>
>  ============================================================
>
>  pro Man::AboutMe
>    print,FORMAT='(%"Dude, I am a %d yr old man, and I like %s.")',self.age,
$
>          self.activity
>  end
>
>  function Man::Init,age
>    if ~self->Person::Init(age) then return,0
>    self.activity='watching football'
>    return,1
>  end
>
>  pro man__define
>    m={MAN, $
>       INHERITS PERSON, $
>       BELT_SIZE: 0.0, $
>       ACTIVITY: ''}
>  end
>
>
>  pro Woman::AboutMe
>    print,FORMAT= $
>         '(%"Pleased to meet you.  I am a %d yr old woman, and I enjoy
%s.")', $
>         self.age, self.activity
>  end
>
>  function Woman::Init,age
>    if ~self->Person::Init(age) then return,0
>    self.activity='gardening'
>    return,1
>  end
>
>  pro woman__define
>    m={WOMAN, $
>       INHERITS PERSON, $
>       SHOE_SIZE: 0.0, $
>       ACTIVITY: ''}
>  end
>
>  function Person::Init,age
>    self.age=age
>    return,1
>  end
>
>  pro person__define
>    p={PERSON, $
>       AGE: 0}
>  end
>
>  ============================================================
>
>  Compile, and then try:
>
>  IDL> people=[obj_new('man',22),obj_new('woman',31)]
>  IDL> for i=0,1 do people[i]->AboutMe
>  Dude, I am a 22 yr old man, and I like watching football.
>  Pleased to meet you.  I am a 31 yr old woman, and I enjoy gardening.
>
>  Sorry for the flagrant stereo-typing.
>
>  JD
>
>
```