

## Re: reduce redundant code

---

*Source:*

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2008-03/msg01197.html>

---

- *From:* "Ralph" <[nt\\_consulting64@xxxxxxxxxx](mailto:nt_consulting64@xxxxxxxxxx)>
  - *Date:* Tue, 11 Mar 2008 23:16:35 -0500
- 

"Schmidt" <[sss@xxxxxxxxxx](mailto:sss@xxxxxxxxxx)> wrote in message  
[news:uBfL2p%23gIHA.4196@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:uBfL2p%23gIHA.4196@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

"MP" <[NoSpam@xxxxxxxxxx](mailto:NoSpam@xxxxxxxxxx)> schrieb im Newsbeitrag  
[news:uTklkE%23gIHA.200@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:uTklkE%23gIHA.200@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

<snipped>.

I'd sure like to get a handle on the oo way...  
don't know why I'm so stupid that I cant'  
figure it out...other than I was born that way :-)

Maybe too many books...? ;)  
And BTW, if you can't figure it out, how to implement  
a solution "in OOP", then you probably don't need an  
OOP-approach for your scenario...<g>

No, seriously – I always try to implement something  
in the procedural way first, arrays instead of collections,  
Enums instead of large ParamLists, etc., because this  
normally results in faster code and also the code-  
volume is not large, ...

I have to walk a very fine line here. For while I have the highest respect  
for Olaf and am positive that whatever design methods he uses work well for  
him, I disagree with this advice.

Back in the day when I started out (Booch and UML 0.9) I blurted out pretty  
much the same frustration – "How do you know when you get it?" The reply I  
remember and pass on to my students to this day – "When you have trouble  
doing it any other way". <g>

## Re: reduce redundant code

If one is trying to learn OO/OOPL starting out with a procedural approach in order to implement something is not a good way to start. It tends to put the "how" in front of the "what" and leads to confusion.

But perhaps just as bad is trying to pick a "pattern" and apply it.

OO is simply the definition of objects and defining the collaboration and relationships between them that exist within your problem domain. So just define the objects and relationships. Keep it simple, who 'is a', who 'has a', who 'uses', ... Use a yellow pad, index cards, whiteboard (my favorite), tinker toys, ...

The implementation will come naturely – but later. <g>

–ralph

.