

## Re: How to pass class as parameter?

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vb/2005-04/msg02611.html>

---

- *From:* "Larry Lard" <larrylard@xxxxxxxxxxxx>
  - *Date:* 14 Apr 2005 02:22:22 -0700
- 

Brett wrote:

> "Larry Lard" <larrylard@xxxxxxxxxxxx> wrote in message  
> [news:1113407173.231972.289250@xx](mailto:news:1113407173.231972.289250@xx)  
>>  
>> Brett wrote:  
>>> I have several classes that create arrays of data and have certain  
>>> properties. Call them A thru D classes, which means there are  
four.  
>> I can  
>>> call certain methods in each class and get back an array of data.  
>> All four  
>>> classes are the same except for the number of elements in their  
>> arrays and  
>>> the data.  
>>>  
>>> I have a MainClass (the form), which processes all of these  
arrays.  
>> The  
>>> MainClass makes use of third party objects to do this. There are  
>> three  
>>> methods in the MainClass. Currently, I pass a parameter to these  
>> three  
>>> methods such as "A as string". This means to process class A. I  
>> have case  
>>> statements that take this parameter in the MainClass methods to  
know  
>> know  
>>> which class needs processing. There is much redundant code here  
>> since the  
>>> same processing is done for each class. It would be nice if I  
could  
>> just  
>>> pass in the class name as a parameter and run the code. Problem  
is I  
>> have  
>>> four classes instead of one and can't just pass them in as type  
>> class.

## Re: How to pass class as parameter?

>>>  
>>> Should I place these four classes in a module? How else can I get  
>> around  
>>> using the case statements in the MainClass?  
>>  
>> If you mean that A B C D all implement the 'same' methods, define  
an  
>> interface IMyMethods containing these methods, and have A B C D all  
>> implement IMyMethods. Then where you currently pass "A" As String,  
>> instead pass MyABCD As IMyMethods, and polymorphically invoke the  
>> methods.  
>  
> I think this approach will work nicely. The method interfaces for  
the four  
> classes are the same. Any functions returning something are all the  
same  
> type. The only differences are inside of the methods, which is  
completely  
> encapsulated. Given this, can I still use the above technique?

Sure thing. However... as a general principle, you should only do this  
if the \*meaning\* of the method called and data returned are the same  
across all the classes that will be implementing the interface.

Example: if we have CSupplier, CCustomer, CEmployee that all implement  
IHaveAnAddress, and one of the interface methods is GetAddress  
returning an array of String, then it would be misleading if, say,  
CSupplier.GetAddress actually returned the last five order headers for  
that supplier, even though the type would be correct.

If the semantics of the calls are different, I would say you're better  
off checking the type with TypeOf ... Is and calling the different  
methods with different names.

--

Larry Lard  
Replies to group please

---

### • *References:*

- ◆ [\*How to pass class as parameter?\*](#)  
◇ From: Brett
- ◆ [\*Re: How to pass class as parameter?\*](#)  
◇ From: Larry Lard
- ◆ [\*Re: How to pass class as parameter?\*](#)  
◇ From: Brett

- Prev by Date: [\*Re: animation in menubar like in IE\*](#)
- Next by Date: [\*MS Chart\*](#)

Re: How to pass class as parameter?

- Previous by thread: [\*Re: How to pass class as parameter?\*](#)
- Next by thread: [\*Richtextbox\*](#)
- Index(es):
  - ◆ [\*Date\*](#)
  - ◆ [\*Thread\*](#)