

Re: need some realworld examples

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vb/2005-02/1241.html>

From: Don (*unknown_at_oblivion.com*)

Date: 01/31/05

Date: Mon, 31 Jan 2005 19:49:41 GMT

"mattie" <mattie@discussions.microsoft.com> wrote in message
news:F654E1FD-F88F-4C69-8A2B-98E704105F05@microsoft.com...

> hi,

>

> *i was just reading an article on interface-based programming and the
example*

> *they used was pretty good to get the concept across.*

>

> *dim dog as IDog*

> *dog=new CBeagle*

> *dog.bark*

>

> *ok, i kinda understand. but i can't imagine what to relate this to in the*

> *business world. could you give some quick examples and explanations of*

> *business related topics where this would be useful. please, the more
examples*

> *and analogies the better. very new to this.*

For one, instantiating objects using interfaces allows you to be more
specific later on your code. e.g.:

```
Dim dog as IDog
```

```
If userWantsANiceDog then
```

```
    dog = new GermanShepherd
```

```
Else
```

```
    dog = new Chihuahua
```

```
End If
```

```
dog.bark
```

One example with which you might use interfaces in variable declarations
might be if you had some of contact management program in which a contact
might be a person or a business. You might do something like this:

(pseudocode)

IContact interface

Function PrintDetails()

PatientContact class

Property FirstName

Property LastName

Implements PrintDetails()

Print FirstName & " " & LastName

BusinessContact class

Property CompanyName

Implements PrintDetails()

Print CompanyName

Here you have an interface (IContact) with one method: PrintDetails. You also have two classes (PatientContact and BusinessContact), both of which implement IContact. As you can see, each one implements the PrintDetails method differently, though, because they store different information. When you have something like the above, you can do neat things like this:

```
Public Function PrintContactDetails(ByVal contact as IContact)
```

```
    contact.PrintDetails
```

```
End Function
```

This function will take either PatientContact or BusinessContact objects and tell them to print their details.

Imagine an inventory control system. You might have an interface called IProduct. Every product class could implement this class, and it could provide methods for doing general tasks like retrieving a product description or whatever. Then your business logic would just work with IProduct objects whenever it doesn't care about what specific kind of product it might be (for example, you might just be displaying a list of product names).

I recently stumbled onto this stuff. It comes in really handy sometimes.

– Don