A **Class** Act

Tim Anderson investigates classes in Visual Basic 4.0, ODBC struggles and PACKing in Delphi.

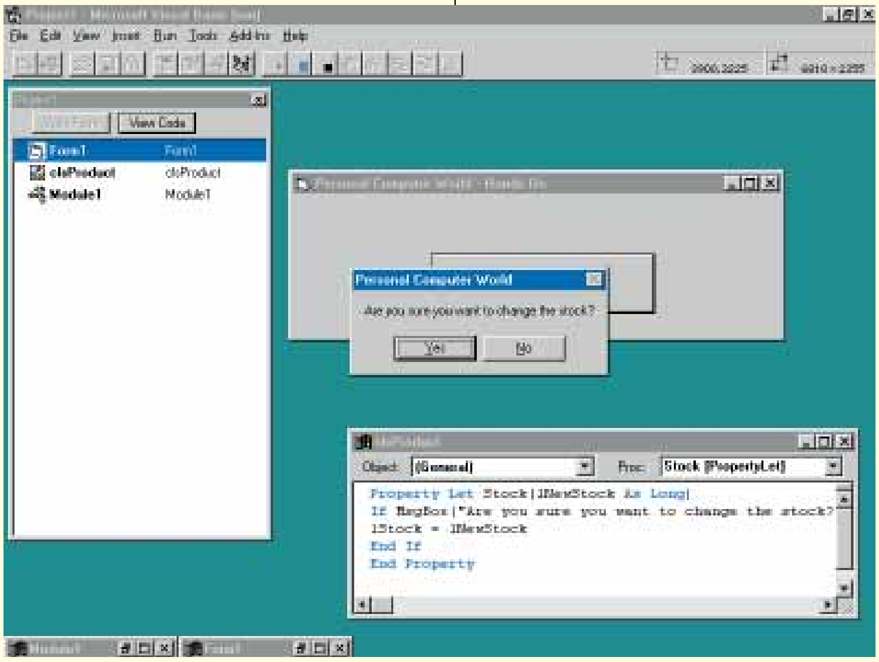
Visual Basic has not embraced all the features of object orientation. That's no reason not to take advantage of the substantial language improvements which it offers, not least the new class module. This enables you to define objects with their own properties and methods. For those unfamiliar with classes, here is a short introduction. In VB, you create a class module by choosing Insert — Class module. Then press F4 to show the property sheet for the class. For example, you might create a product class, and set the Name property to clsProduct. Next, define custom properties and methods for the class. For example:

```
Option Explicit
Public name as string  ‘ name of the
product
Public description as string ‘
describes the product
Public productID  as string  ‘ iden-
tifies the product
Public stock As Long ‘ number in
stock

Now you can use the class in your
code, for example:
Dim Widget as new clsProduct
Widget.name = “Widget”
Widget.Description = “A very handy
thing indeed”
```

But this is no more than the old user-defined type by another name. The difference is that classes also support methods and property procedures. For example, it is dangerous to expose Stock as a public property. It would be all too easy to set it to a wrong value by mistake. The answer is to rewrite clsProduct as follows:

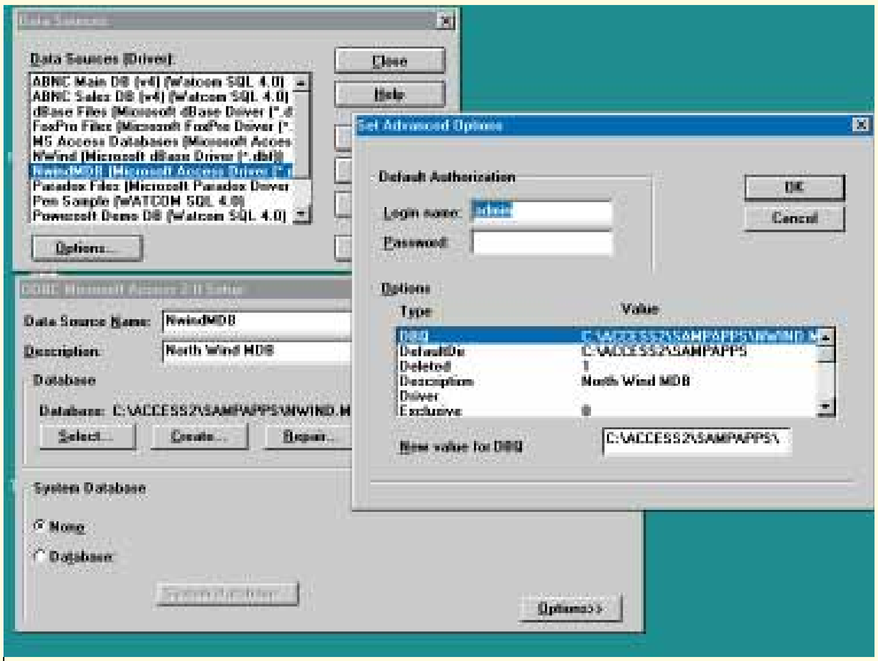
```
Private lstock as Long ‘ visible only
to class methods
Property Get Stock() as long
stock = lstock
end property

Property Let Stock(lNewStock) as long
if msgbox(“Are you sure you want to
change the stock?”, vbYesNo) = vbYes
then
lstock = lNewStock
endif
end property

You can test the new class as
follows:

Dim widget As New clsProduct
widget.Stock = 34
MsgBox widget.Stock
```

Note that when you set the Stock property, the confirming dialogue appears. In the real world, the confirming dialogue can be replaced by whatever code you require to ensure a valid stock update. You can prevent a negative value, for example; or check a User object for authorisation. Even better would be to replace the Property Set procedure with two methods, AddStock and ReduceStock. Gradually the code becomes more robust and easier to reuse (simply import the class module into another project).



```
Property Let Stock(lNewStock As Long)
If MsgBox("Are you sure you want to change the stock?
lStock = lNewStock
End If
End Property
```

Using VB 4.0's class modules, you can easily create applications that validate and protect key data


Data Sources (Driver):
ABNC Main DB (v4) (Watcom SQL 4.0)
ABNC Sales DB (v4) (Watcom SQL 4.0)
dBase Files (Microsoft dBase Driver (*.d
FoxPro Files (Microsoft FoxPro Driver (*.
MS Access Databases (Microsoft Acces
NWind (Microsoft dBase Driver (*.dbf))
NwindMDB (Microsoft Access Driver (*.
Paradox Files (Microsoft Paradox Driver
Pen Sample (WATCOM SQL 4.0)
Powersoft Demo DB (Watcom SQL 4.0)

ODBC Microsoft Access 2.0 Setup
Data Source Name: NwindMDB
Description: North Wind MDB
Database: C:\ACCESS2\SAMPAPPS\NWIND.M
System Database: (● None) (Database)

Set Advanced Options
Default Authorization
Login name: Admin
Password:

Options

Type	Value
DBQ	C:\ACCESS2\SAMPAPPS\NWIND.M
DefaultDir	C:\ACCESS2\SAMPAPPS
Deleted	1
Description	North Wind MDB
Driver	
Exclusive	0

New value for DBQ: C:\ACCESS2\SAMPAPPS\

Setting up an ODBC driver — do you really want your users to see this?

There is more to say about OO programming in VB, and future *Hands On* columns will return to this subject.

● How are you finding Visual Basic 4.0? Please contact me to say what you think of the new release and how you are using the new features.

Wrestling with ODBC

Steven Fletcher has written a database application using Visual C++ and ODBC. He writes: "I decided to use Microsoft Access as the Database builder and standard. My programs access the databases via CRecordset and CDatabase using a ODBC link to my Access Database files. This has been an effective method so far. The main problem I have using ODBC is the installation of the software on other PCs. Firstly, ODBC has to be installed, then the database files have to be copied across. ODBC has to be set up to recognise the installed database and it must be given the correct name for the program to read.

"Is there an easier way of installing my program together with ODBC? And since JET is a direct MS Access engine, can I use and link via CRecordSet with JET?"

Steven has hit on one of the least appealing aspects of ODBC, when used for applications that are to be distributed. ODBC consists of manager software, plus one or more database drivers, plus one or more data sources which use those drivers to access specific databases. All these are a shared, system-wide resource. If you want to create an ODBC application which can be delivered as a shrink-wrap and easily installed onto any Windows PC, there are several problems to solve:

1. The target PC may or may not have ODBC installed.

2. The drivers you require may or may not be installed.

3. The data source will presumably not be installed, although if you are unlucky the data source name (DSN) you chose may already be in use.

4. ODBC is a version control nightmare. If your application installs the latest versions of the manager and driver DLLs, there is always the chance that some other ODBC application will no longer work.

5. To add to the fun, there are now separate 16-bit and 32-bit ODBC versions, both of which may be installed on the same system.

The immediate conclusion is that ODBC is best managed in a corporate environment, where the IS department can control and configure the installations as required. But this is shrink-wrap software. That leaves two other possibilities. One is to ship the ODBC installation disks as supplied by Microsoft, and carefully explain to the user that they need to install ODBC from the separate setup disks, run the driver manager, and set up the data source name. That is asking a lot: end users should not have to face intimidating dialogues asking them to configure data sources.

That brings us to the alternative, which is to control ODBC programmatically from your own installation routine. To do this you need to obtain the ODBC 2.x SDK from Microsoft; it is on the MSDN level two disks. Your setup program will need to copy across the ODBC DLLS, and then call the ODBC API to configure it. For example, there is a function called SQL-

Christmas Wishes

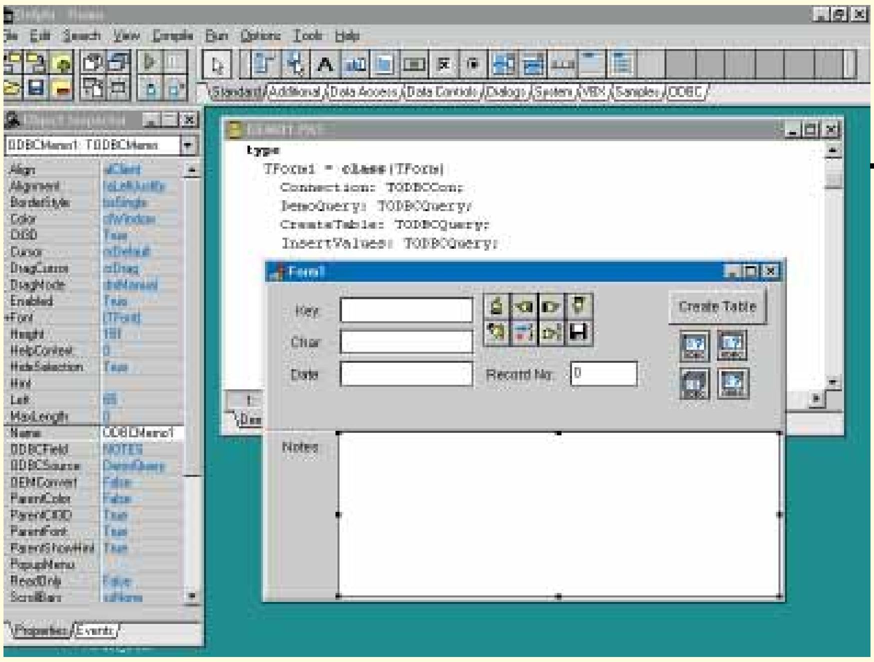
Dear Santa

It's that time of year again. And dear Father Christmas, I wrote to you last year with some very reasonable Visual Basic enhancement requests, few of which have been fulfilled. What did I ask for? Oh yes, a compiler. And object oriented extensions to the language — yes, I know we got the class module in VB 4.0, but somebody forgot inheritance. And decent error handling — what happened to that? And Visual Basic for the Mac, where are you?

Maybe I shouldn't complain. You did deliver three of the things I asked for. Version control was one, and VB 4.0 is miles better in this respect. A rich text edit control was another, although I had hoped it would run in Windows 3.x. And you excelled yourself with the third: I asked for more competition for VB, and now you can't move for VB lookalikes. Incidentally one of these, Borland's Delphi, delivers most of the other items on the list as well.

Software development, they say, is the triumph of hope over experience. Here's seven modest requests for 1996. Over to you Santa.

1. A VB compiler, OO VB, better error handling (for details, see last year's list).
2. Slimmed-down Microsoft Office, broken into small OLE components that make Office automation a more realistic proposition.
3. Microsoft and Borland to agree on a single DBF standard (see Fig 1, page.330).
4. An end to the O/S wars. Last year my VB 3.0 program ran fine on Windows 3.x, NT and OS/2. Now 32-bit Windows has messed up everything.
5. A 12-month break in publicity for Windows 95. No, make that a permanent break.
6. Visual Basic for Applications in Word.
7. Working OpenDoc applications, and not just on the Mac, but on OS/2 and Windows as well. If this is a superior alternative to OLE, we need to see it in operation — soon.



```
type
  TForm1 = class(TForm)
    Connection: TODBCCon;
    DemoQuery: TODBCQuery;
    CreateTable: TODBCQuery;
    InsertValues: TODBCQuery;
```

ODBCMemo1: TODBCMemo

Align	alClient
Alignment	taLeftJustify
BorderStyle	bsSingle
Color	clWindow
Ctl3D	True
Cursor	crDefault
DragCursor	crDrag
DragMode	dmManual
Enabled	True
+Font	(TFont)
Height	181
HelpContext	0
HideSelection	True
Hint	
Left	65
MaxLength	0
Name	ODBCMemo1
ODBCField	NOTES
ODBCSource	DemoQuery
OEMConvert	False
ParentColor	False
ParentCtl3D	True
ParentFont	True
ParentShowHint	True
PopupMenu	
ReadOnly	False
ScrollBars	ssNone

Dan Daley has created a VCL wrapper for ODBC that bypasses the BDE entirely. The product is currently in beta, but can be downloaded from Compuserve

ConfigDataSource, which adds, modifies or deletes data sources. With care, it should be possible to build an installer application which silently installs and configures ODBC, complete with version control, and checks for name conflicts. But it is a delicate business, and in my opinion a good reason not to use ODBC for shrink-wrapped database applications.

Steven also asks about CRecordSet. This is an ODBC class so cannot use JET (except via a roundabout route if a Microsoft ODBC driver itself calls JET). Visual C++ version 4 introduces the CDaoRecordSet class. This is one of the new set of Data Access Objects (DAO) classes. Using DAO, Visual C++ programmers can use JET as easily as VB programmers. If you port your application to use DAO classes, and if (as Steven is) you are using a database supported directly by JET, then ODBC is bypassed entirely. In this case it looks like a wise move, except that the DAO classes do not exist in the 16-bit version of Visual C++. So if Steven needs to support Windows 3.x, it's back to ODBC with all its frustrations.

Getting Access from Delphi

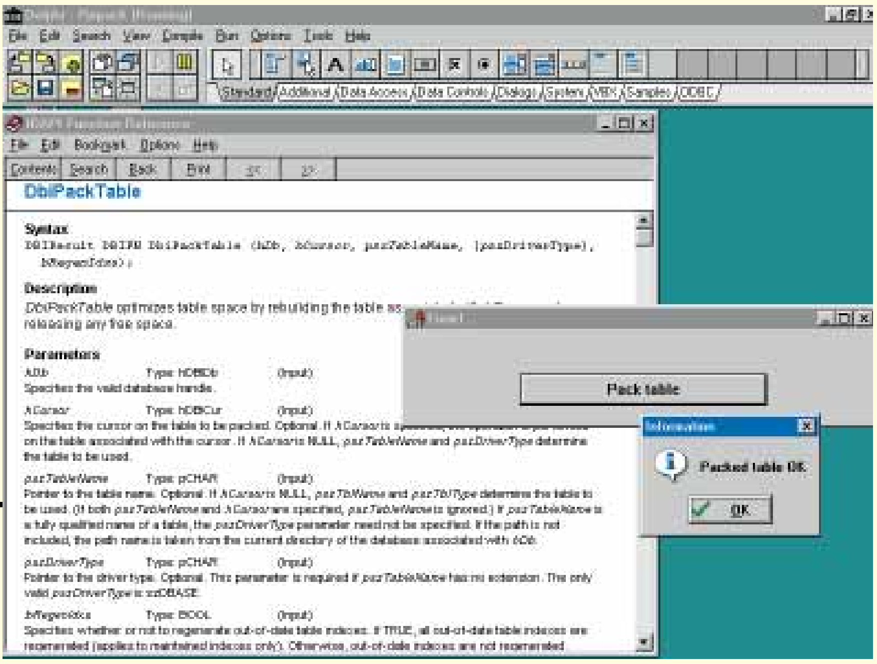
Colin Dow raises a common problem: "Do you have any tips for connecting to an Access database using Delphi. There must be a few people out there who use Access databases but like the speed of Delphi. I don't know if there are any specific settings you can set up in the ODBC/IDAPI to make the connection more reliable, but tables with counters don't seem to behave very well. Or even better, is there an Access VCL along the lines of the Apollo engine which you recommended in September?"

Between them, Microsoft and Borland control the most popular desktop database formats on the PC. Microsoft have the FoxPro variant of the DBF, and their in-house invention, the Access MDB. Borland's equivalents are the dBase DBF variants, and the more sophisticated Paradox DB format. As you would expect, Borland's Delphi has native connections to dBase or Paradox data, but not FoxPro or Access. That means using ODBC, which is not Delphi's strongest suit, along with an MDB driver. A better option is to give in and use the Paradox format instead; but other factors may make that impossible. So it's ODBC; and here the main tip is you need the Microsoft Desktop Driver set along with ODBC 2.0 or higher. The drivers supplied with Microsoft Office 4.x do not work. The desktop drivers are part of the ODBC SDK which can be found on MSDN level 2, or on the Visual C++ 2.x CD, or can be obtained separately from Microsoft.

Apparently Microsoft refuse to release the MDB file format specifications, which makes it unlikely that a custom VCL will appear. The problem is that the advanced features of the MDB format are intimately linked with JET. But you can avoid going through the Borland Database Engine by calling the ODBC API directly. At least one person has created a VCL wrapper for ODBC, and a Beta version can be found in the Delphi conference on Compuserve. The package includes some data-aware controls for ODBC, and completely bypasses the BDE. email the author, Dan Daley (daley@scruz.net), for more details.

Get packing in Delphi

Finally, Andrew Richmond asks: "How do you pack a dBASE table from a Delphi



DbiPackTable

Syntax

```
DBIResult DBIFN DbiPackTable (hDb, hCursor, pszTableName, [pszDriverType],
  bRegenIdxs);
```

Description

DbiPackTable optimizes table space by rebuilding the table as [illegible] releasing any free space.

Parameters

hDb Type: HDBIDb (Input)
Specifies the valid database handle.

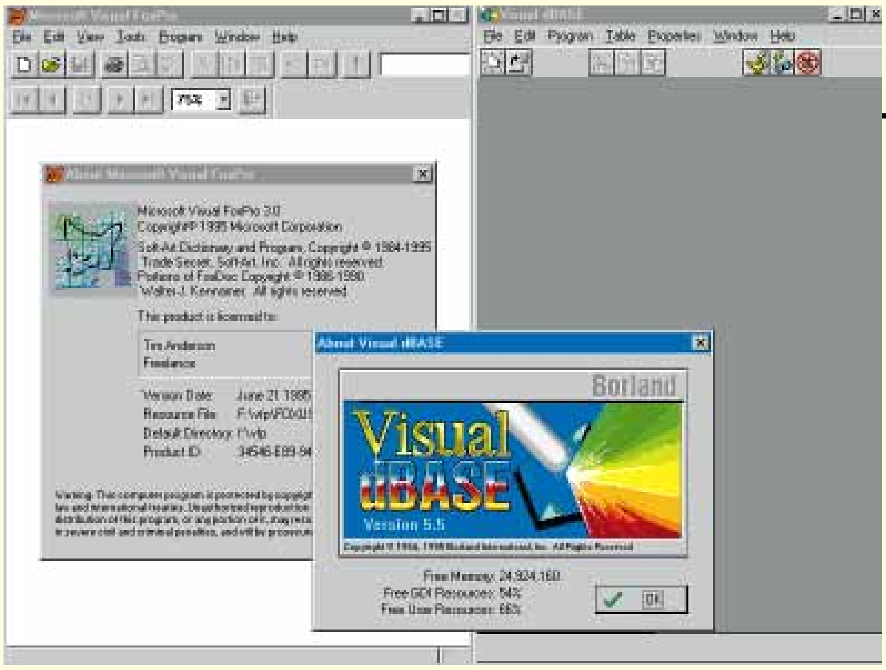
hCursor Type: HDBICur (Input)
Specifies the cursor on the table to be packed. Optional. If *hCursor* is [illegible] on the table associated with the cursor. If *hCursor* is NULL, *pszTableName* and *pszDriverType* determine the table to be used.

pszTableName Type: pCHAR (Input)
Pointer to the table name. Optional. If *hCursor* is NULL, *pszTblName* and *pszTblType* determine the table to be used. (If both *pszTableName* and *hCursor* are specified, *pszTableName* is ignored.) If *pszTableName* is a fully qualified name of a table, the *pszDriverType* parameter need not be specified. If the path is not included, the path name is taken from the current directory of the database associated with *hDb*.

pszDriverType Type: pCHAR (Input)
Pointer to the driver type. Optional. This parameter is required if *pszTableName* has no extension. The only valid *pszDriverType* is szDBASE.

bRegenIdxs Type: BOOL (Input)
Specifies whether or not to regenerate out-of-date table indexes. If TRUE, all out-of-date table indexes are regenerated (applies to maintained indexes only). Otherwise, out-of-date indexes are not regenerated.

You can pack dBase tables from Delphi. It's not documented with the product, but requires the BDE help file (also shown), obtainable from Borland


Microsoft Visual FoxPro 3.0
Copyright© 1995 Microsoft Corporation
Soft-Art Dictionary and Program, Copyright © 1984-1995
Trade Secret, Soft-Art, Inc. All rights reserved.
Portions of FoxDoc Copyright © 1988-1990
Walter J. Kennamer. All rights reserved.
This product is licensed to:
Tim Anderson
Freelance
Version Date: June 21 1995
Resource File: F:\vfp\FOXU
Default Directory: f:\vfp
Product ID: 34546-E89-94

About Visual dBASE
Borland
Visual dBASE
Version 5.5
Free Memory: 24,924,160
Free GDI Resources: 54%
Free User Resources: 66%

Fig 1 *Two xBase databases — two incompatible data formats. Why?*

program?” Strangely, although Delphi uses the same database engine as dBase itself, table objects have no Pack method. The only way round this is to use lower-level BDE functions, which Borland have documented in a file called BDE.HLP, available for download from Compuserve and no doubt elsewhere. Using these functions is much harder than programming the usual database components, and not often necessary. The following skeleton routine successfully packs a table on my system. Note that for this to work, the table must not be in use elsewhere. To use this in earnest, you need the full description of the parameters and possible error messages, contained in the Borland help file:

```
uses
... DBiTypes, DBiProcs, DBiErrs;

implementation

procedure PackTable;

var
hMyDb : hDBIDb;
iResult: Word;

begin
dbiInit(nil);

iResult := DbiOpenDatabase(nil,’STAN-
DARD’,
dbiREADWRITE,dbiOPENEXCL,’’,0,nil,ni
l,hMyDb);
{note database must be opened exclu-
sive}

if iResult <> DBIERR_NONE then
MessageDlg(‘Error opening database’,
mtError,[mbOk], 0)
else
iResult :=
dbiPackTable(hMyDb,nil,’C:\MYTABLE.D
BF’,’DBASE’,False);
{table must not be in use elsewhere}

if iResult = DBIERR_NONE then
MessageDlg(‘Packed table OK’, mtIn-
formation,[mbOk], 0);

iResult := dbiCloseDatabase(hMyDb);
iResult := dbiExit;
```

PCW *Contacts*

Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted via *PCW* at the usual address, or
freer@cix.compulink.co.uk