March, 1999

Editor's View

Now What?

After you get your hands dirty with VFP, it's time to dig in and learn how OOP benefits you.

By Tamar E. Granor, Editor

Last month, I talked about the process of learning Visual FoxPro, especially for those who coming from earlier Xbase languages. I suspect that by the time you read this, I'll have heard from quite a few OOP purists distressed by my suggestion that you should jump right in and start writing without doing a careful analysis and design of the classes required. I stand by my advice, but now it's time to tell, as Paul Harvey says, "the rest of the story."

While it's okay to just figure out what classes and methods you need by the seat of your pants in the first application you write, and even the second or third, after that, it's time to get more structured. After writing a few applications in VFP, you should have some ideas about what things are candidates for truly reusable classes.

Again, it's okay to start small as long as you keep your eyes open for larger possibilities. Perhaps you've noticed that every application you write seems to include a form to allow users to enter an id and password, with the password hidden. After you've done this a few times, you may notice that two candidates for abstraction here. The obvious one is the textbox that allows input of the password. Instead of setting all the required properties each time, you can create a password textbox that knows what it needs to do.

A sign-on form that checks an id and a password against a master list is clearly another resuable item. You might create a form class that includes the two textboxes (or perhaps a combobox for the id and a textbox for the password) and a couple of buttons (OK and Cancel) and put logic to check the id and password in the OK button's Click method.

Certainly, such a form class is useful in lots of applications. But here's where you need to dig deeper. Does checking an id and password always have to come in the context of this kind of form? What if the physical interface for the system involves a card reader and a keypad, as with a credit or debit card? You still have an id and password to check, but there's no combobox, no textbox, no buttons.

This suggests that the code to perform the actual id/password check doesn't belong in the OK button's Click method. Then where does it belong? How about a method with a name like CheckIdAndPassword? That makes sense, but where does that method belong? It could be a form-level method, but then you'd still need to use a form when you have the card reader and keypad.

A better approach is to have an object (a class) whose sole reason for existence is security. Then, the button's Click method can call the security object's CheckIdAndPassword method. So can the code that interfaces with the card reader/keypad combination. Rather than putting the actual operation into the form, we separate interface from implementation to give us maximum flexibility.

This is where object orientation shines and where finding abstractions is the hardest while you're learning. The more you work at it, the easier it gets, of course. You can also improve your ability to find these abstractions by learning about Design Patterns. (See Y. Alan Griver's articles in the March, April and May '98 issues.) Design Patterns are general descriptions of problems that come up again and again. Categorizing and naming them provides a common vocabulary for discussing them. The naming process also helps to generalize problems rather than seeing each item as a new, unique situation. That's a key part of knowing what classes to create.

By the way, the password checking set-up (using a dedicated password class that interacts with one or more different input classes) is a "bridge," one of the simplest and most common patterns.

DevCon on the Horizon

No sooner had I written last month's column than I jumped into planning one of the biggest learning events for VFP developers, the annual Developers Conference (known far and wide as DevCon). This year's conference, to be held June 6-10 in Palm Springs, CA, will be the 10th FoxPro DevCon. (It'll be my eighth and my sixth time as a speaker. There are a few people in the FoxPro community who have attended all nine DevCons to date. We hope they'll keep their perfect attendance records intact.)

DevCon offers a unique mix of opportunities: sessions given by the leading FoxPro developers in the world, sessions by some of the people responsible for designing and developing Visual FoxPro, and a chance to meet your colleagues.

The sessions offer a mix of topics from VFP fundamentals to practical advice on using the language productively to VFP's interactions with other development tools to application design and analysis. Whatever your VFP interests are, you'll find something at DevCon.

Beyond the sessions, DevCon offers other benefits, both tangible and intangible. On the concrete side, you'll find people sitting around at all times of day and night (DevCon isn't known as a place to catch up on your sleep) solving VFP programs, talking about what they're working on, what they're learning, and where they're headed. The FoxPro community is incredibly open to newcomers, so just introduce yourself, and join the conversation.

As for the intangible, spending the better part of a week surrounded by people who do what you do is a real morale booster. I've never failed to go home from DevCon all charged up, filled with new ideas, and ready to take on the world.

You can find details of this year's DevCon at www.advisor.com. Plan now to join us in Palm Springs.