中美通顾问/大华工作室

附范例磁片

陈宗兴 编著

Visual FoxPro 3.0 实用手册

——面向对象篇

科学出版社
龍門書局

Visual FoxPro 3.0 实用手册

——面向对象篇

陈宗兴　编著

彬　山　改编

科学出版社

龍門書局

1997

(京)新登字 092 号

内 容 简 介

FoxPro 是新一代关系数据库管理系统的杰出代表。本书对最新版 Visual FoxPro 3.0 面向对象程序设计作了全面系统的介绍。通过实例和操作步骤介绍新概念、新方法、新技术是本书的主要特色。

本书语言浅显，内容全面，图例丰富，可供广大管理人员和各类培训班使用。

需要得到有关本书的技术支持或购买本书的读者，请与(010)62562329，(010)62541992 联系，或传真至(010)62561057。

版 权 声 明

Visual FoxPro 3.0 实用手册——面向对象篇

陈宗兴 编著
彬 山 改编
责任编辑 汪亚文

科学出版社
龙门书局 出版

北京东黄城根北街 16 号
邮政编码：100717
北京广益印刷厂印刷
新华书店北京发行所发行 各地新华书店经售

*

1997 年 1 月第一版 开本：787×1092 1/16
1997 年 1 月第一次印刷 印张：26 1/2
印数：1～5000 册 字数：616000
ISBN 7-03-005611-6/TP.666
定价：39.00 元

序

1995 年 9 月笔者出版了《Visual FoxPro 3.0 实用手册——屏幕篇》一书。至今，在各种有关 VFP 培训班或研讨会上，总会有支持的读者询问《面向对象》何时推出？甚至有许多读者打电话到书店询问何时有售？这些热心读者的殷切盼望，让笔者非常感激。为此，感到有必要花更多的精力做最完善的整理，以回报读者。

本书是笔者自 1995 年 6 月至今，在从事项目辅导与顾问期间，针对使用者开发时所需要的解决方法，结合教学经验，采用逐步说明的方式，将 VFP 面向对象的精华——自定义 Subclass 的概念，配合实际开发所创建的范例，向读者说明如何设计自定义 Subclass，以及如何将 Subclass 直接堆积成应用系统。

本书在撰写期间，恰逢 Microsoft 对厂商进行辅导与培训，因此笔者对在辅导过程中所面对的若干应用系统开发问题，特别设计了一个完全使用 OOP 方法开发的应用系统，附在范例盘(详见书末订单)中，同时，对该应用系统范例还使用了 Local View 方法开发，让读者将来在开发 Client/Server 主从式系统时，也能使用同样的设计概念扩展自己的应用系统。

在辅导厂商期间，特别感谢“中坜资策会”蔡和穆、孙德华老师，以及“捷禾资讯”林启民经理，提出的若干开发上的问题。尽管身为顾问有责任提出相应的解决方案，但有时在集思广义的激励下，资策会项目组与捷禾资讯的同仁常常有令人激动的新构思，这在集体开发过程中，确实相当有益，也使笔者受到极大的鼓励。因此，本书是笔者在这半年系统辅导期间，配合培训班讲课经验所提出的一本面向对象程序设计的入门手册。

笔者深信，在可视化开发工具使用愈来愈广泛的时代，我们学习的习惯务必作重新调整，也就是说“无须再保留过程性的学习概念，即在熟悉所有的指令与函数之后，再从事系统开发”，应该是“在可视化工具辅助下，边开发边学习”，这样才可以节省学习时间。否则，若将 VFP 指令、函数、对象属性、方法弄懂之后，再来开发系统，实际上还得重头来，效益不大。

笔者著书的方向，也是笔者辅导厂商，让厂商快速上手的训练方向。因此，采用范例结合 VFP 特性，通过步骤说明，增加读者的学习兴趣，同时达到重点提示，让读者轻松学会设计的技巧与概念，快速进入面向对象程序设计领域。笔者仅希望这样的著书方向，真正能够为读者带来一点学习上的乐趣与整体上的学习效果。

最后感谢您的支持，与松岗同仁的协助，本书得以有其存在价值，也是笔者相对的成就，谢谢大家。

目　录

第一章　VFP 面向对象模块

Visual FoxPro 采用面向对象程序设计技术，让程序员以“对象”为中心来设计模块，而不是以“过程”为中心思考应用系统开发的结构。这种设计思想与结构，往往使以过程为中心的程序员无法“抓住”程序设计的方向，而且更严重的是：面对自己引以为傲的“过程性程序的逻辑与结构掌握较好”的有利因素，完全在“面向对象程序”配合“可视化工具”之下，变得毫无用处！

但是，面对程序生产力偏低与维护成本过高的情况，“面向对象程序”的高生产力与维护的方便性，是解决程序生产力与维护成本的一个好方案。

如图 1.1 所示，笔者直接拖曳一组按钮的 Container Class 放置在 Form 上，这样记录移动的功能模块已自动“构造”完毕。

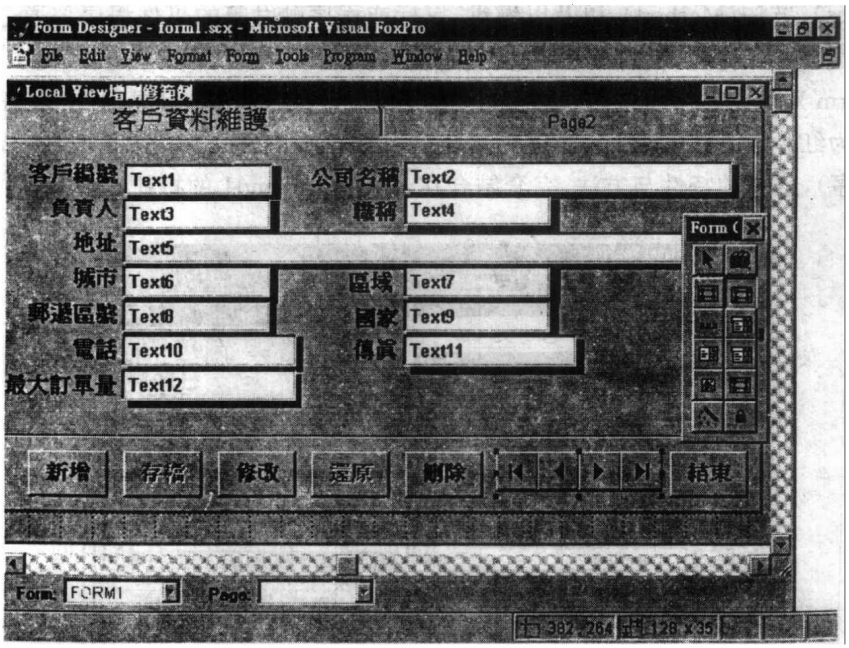


图 1.1　面向对象程序设计——以堆放方式“构造”应用系统

笔者强调的是“构造”，也就是说开发应用系统时，有许多的界面与功能在设计时，是通过堆放的方式进行设计的，即将所需要的功能（对象）逐一在应用系统中堆放起来。就拿上面的范例来看，其记录移动按钮组对象，是以堆放方式设计的。

这样的设计结构，也就是 VFP 完全面向对象的功能，可以让对象模块重复使用，快速建立起应用系统。

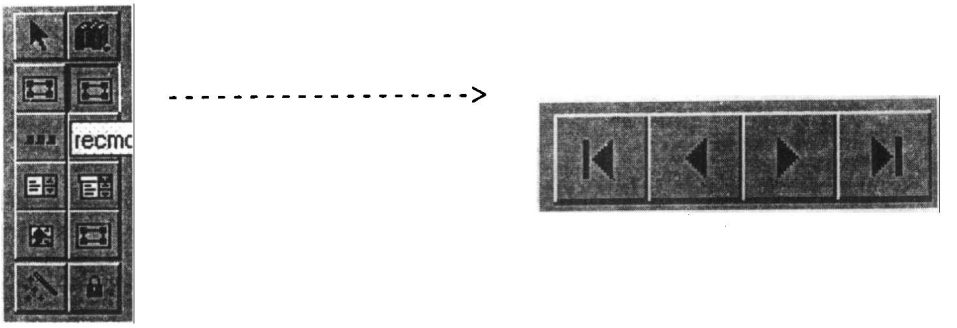


图 1.2 由 Controls ICON 变成对象

1.1 何谓对象

在 VFP 中，对象(Object)简单地说就是组成 VFP 程序的基本元素，该元素具有属性(Properties)、方法(Method)，以及以键盘、鼠标或程序触发等的事件程序。通过这么多的方法、属性、事件所组合而成的抽象实体，就称为对象。例如，在 VFP 开发系统中，如图 1.3 所示，在 Form 对象上放置一个按钮(CommandButton)，该按钮文字提示为“离开”、字形为楷体、颜色为红色(以上为属性)，当这个按钮被鼠标左键触发之后，会将 Form 从内存中清除(事件程序)，这样的特性与方法，完全结合在名为 Command1 的按钮对象上。

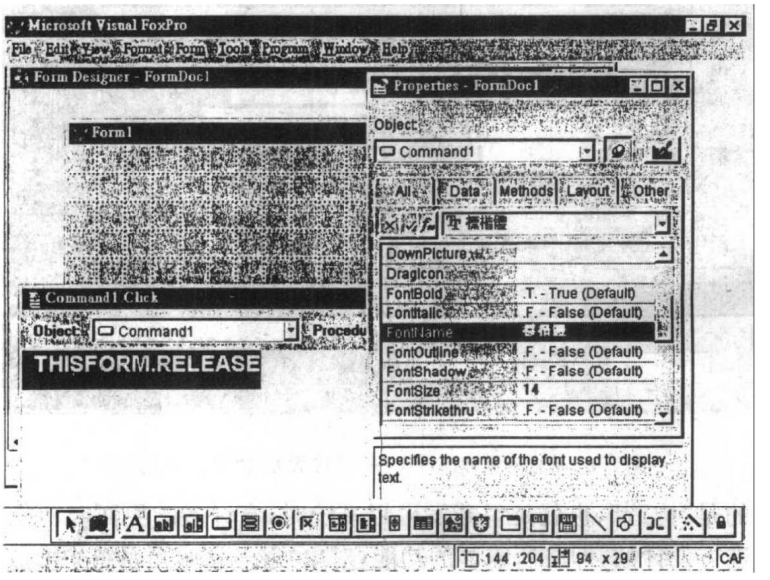


图 1.3 VFP 按钮对象的属性与方法设置

VFP 所提供的对象，对设计者的好处有：

(1)可以减少程序编码。

(2)在应用系统中无须考虑结构,容易组织应用系统。

(3)借助面向对象程序结构,易于软件重用(Reuse)。

(4)降低程序维护成本。

(5)可视化工具辅助,易于设计。

(6)易于将程序模块统一,避免维护冲突。

当然,VFP 对象不只是提供这些好处,笔者将会在本书中通过实际范例,让读者进一步了解 VFP 面向对象的超强功能。

1.1.1 对象减少程序编码

对象的确可以减少程序编码,例如要设计一个按钮,并将此按钮设计成被触发时可以对 Form 的窗口标题重新设置显示字符串。执行结果如图 1.4 所示,窗口标题通过触发按钮后,可重新设置。

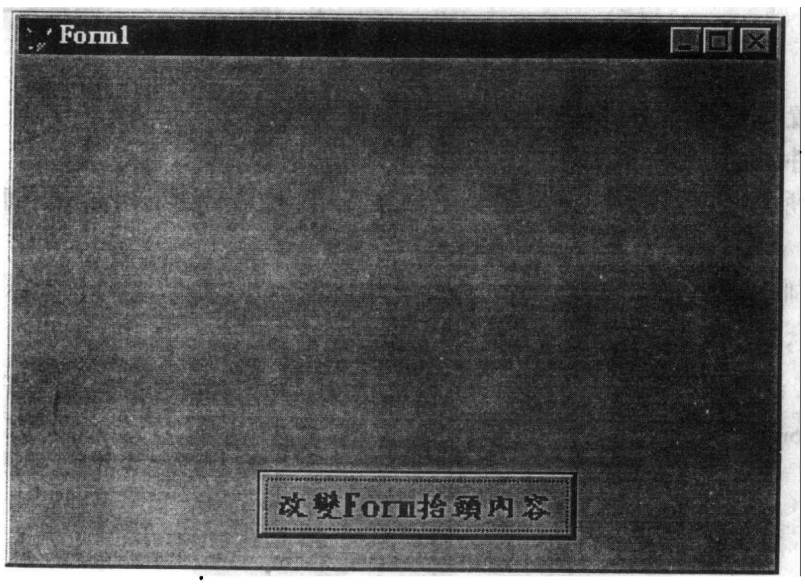


图 1.4 触发按钮后,可以改变 Form 标题内容

当用鼠标单击命令按钮后,立即将 Form 的标题内容改变,执行结果如图 1.5 所示。

如果我们使用过程设计方法,则必须使用下面所述的算法:

(1)Define Window 定义一个窗口(Form)。

(2)定义这个 Form 的标题内容。

(3)利用程序声明这个窗口的背景颜色填充(Fill)灰色。

(4)通过 Move Window Center 指令将窗口定义在屏幕中间显示。

(5)设置一个按钮在窗口相对位置上显示。

(6)设置这个按钮的提示(Prompt)。

图 1.5 触发按钮后改变 Form 的标题内容

(7)设置其 Click 事件程序。

(8)事件内容程序采用指令方式将 Form 的标题内容改变。

像上面所叙的算法,至少得编写几行以上的程序,才能将图 1.4/1.5 之间的变化结果表现出来。

如果我们采用面向对象方法设计,配合 VFP 可视化工具,则设计过程如下所示:

步骤一:启动一个新的 Form 对象并定义属性

如图 1.6 所示,在 Form Designer 工具中,启动一个 Form 对象并选择 Properties Window 定义 Form 的:

AutoCenter=.T.	Form 在屏幕中央显示
BackColor=RGB(192,192,192)	背景为灰色
Caption='Form1'	Form 窗口标题内容

步骤二:设计命令按钮类型与事件程序

接下来在 Form 上直接定义一个 CommandButton 命令按钮对象,并且定义如下按钮属性:

caption='改变 Form 标题内容'	按钮提示内容
FontName='楷体'	设置提示字形颜色
FontSize=12	设置字形大小

设置状态如图 1.7 所示。同时设置按钮的 Click Event 鼠标左键触发的程序内容:

Form1.Caption='改变 Form 标题内容'

在面向对象设计中结合了可视化工具,实际上只是定义对象的属性与方法,真正编程的地方,只有 Click 事件程序中的一行面向对象的语句,其余均属于画面展示的结果,直接交

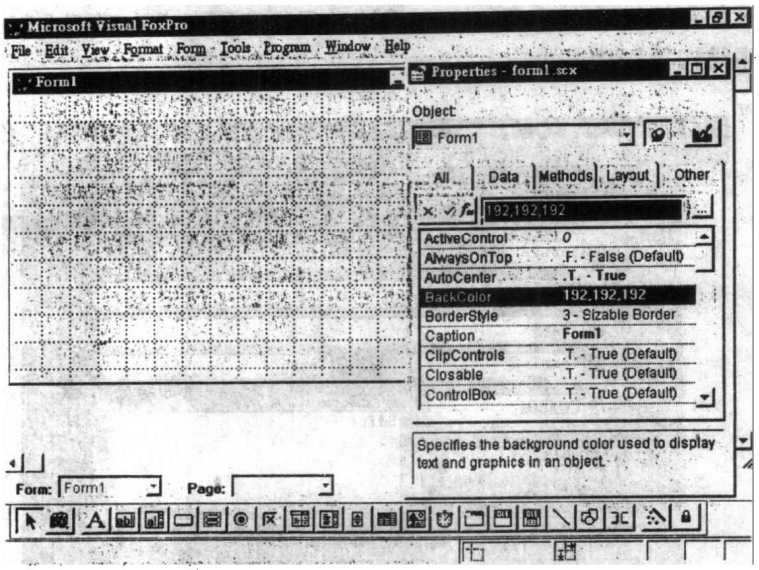


图 1.6　设置 Form 的属性

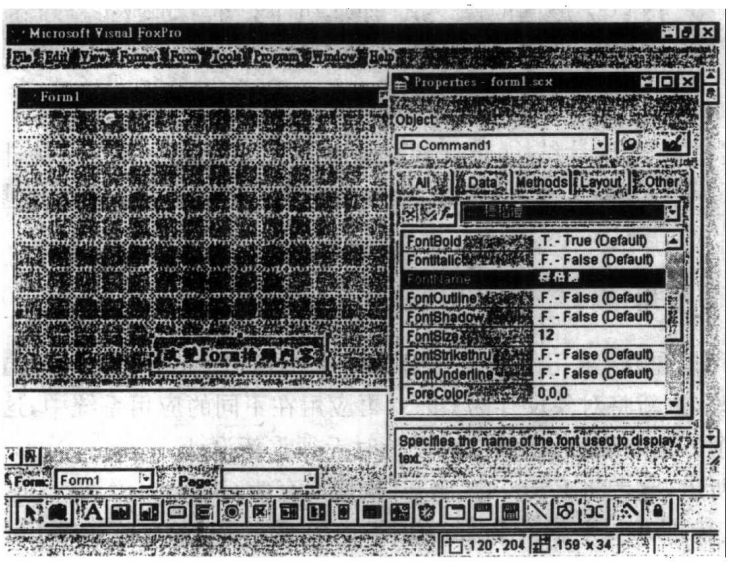


图 1.7　设置按钮的属性与事件程序

给对象本身属性定义即可。

使用对象为何可以减少程序编码，由上面范例可以说明：

“因为对象本身具有多个属性(Properties)与方法(Method)，配合分工较细的事件程序，可以使您在设计时，以对象为中心，直接定义对象属性与方法，这样即可以减少编程量。”

1.1.2　对象是最佳化的结构

对象本身是最佳化的结构(Structural)。如图 1.8 所示，笔者设计了一个维护客户主文

件的画面，把对象直接堆放在 Form 上，在设计过程中无须考虑在画面上新增加或减少一个对象时，对程序结构是否造成影响。

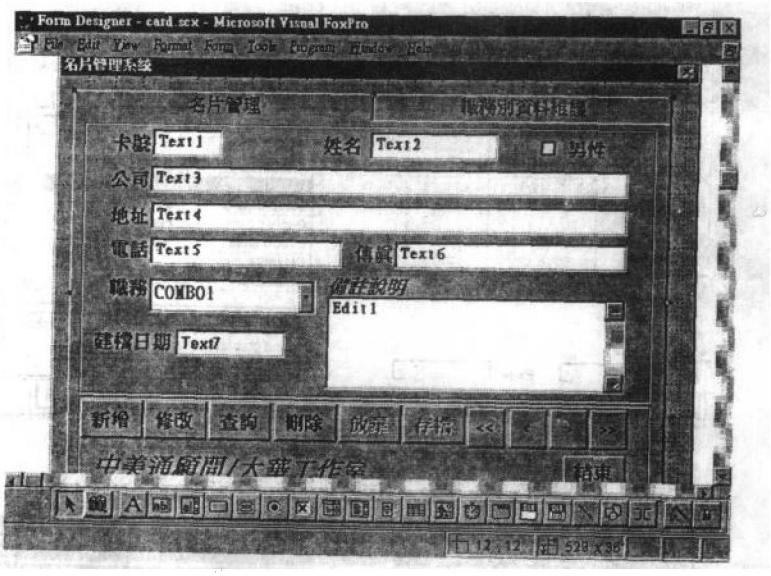


图 1.8　对象置放过程中并不影响程序结构

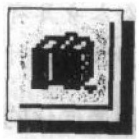
对象这种“独立”的设计结构，彼此相互关联的设计，的确让程序员能非常方便地进行设计，而不用担心“牵一发动全身”的困扰，程序员在设计时只须站在对象立场上考虑，无须担心程序结构。

1.1.3　对象可以重用

对象本身可以重用，重用的意义在于 Base Class(VFP Controls)在建立为对象之后，可以通过部分修改，重新将对象反存为 Class，再应用在不同的应用系统中，这样在第一次设计时所设置的各项方法与属性，在第二次设计时无须重新设计。

如图 1.9 所示，笔者将 Form 的“离开”按钮回存为 Class。

存储后，重新打开一个新的 Form，如图 1.10 所示，笔者直接换掉 Form Controls Toolbar 中的 Controls 组，选择“离开”Control，将其定义在 Form 上，这时在 Form 中的按钮对象即是图 1.9 所示的命令按钮对象。至于原先的方法，在这个按钮中都存在，无须特别设计任何一行程序或对象的属性与方法。

至于如何将“离开”Class 显示在 Form Controls Toolbar 上(From Controls Toolbar 如何应用请参考《Visual FoxPro 实用手册——屏幕篇》一书)，只要通过  View Classes 进行 Add Class Library 即可，这样就可以将 Class Library 中各个 Class 变成 Form Controls Toolbar 中的一个图象按钮，如图 1.11 所示。

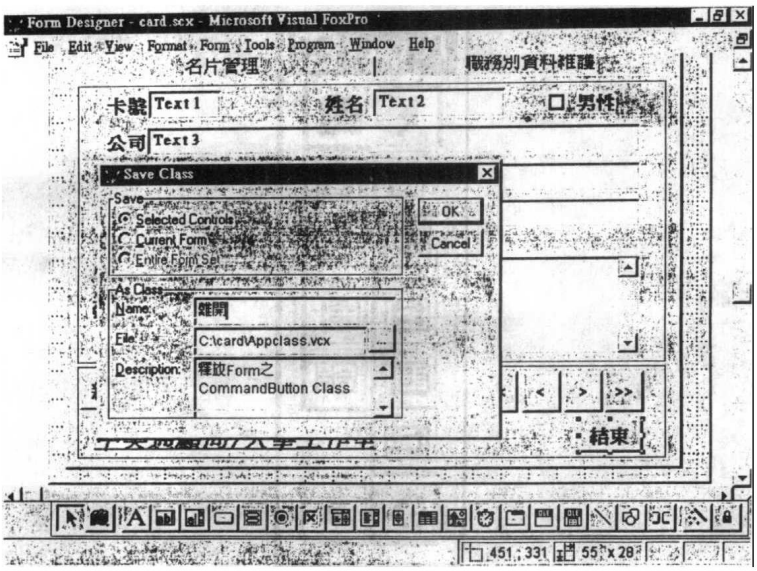


图 1.9　存储按钮对象为“离开”Class

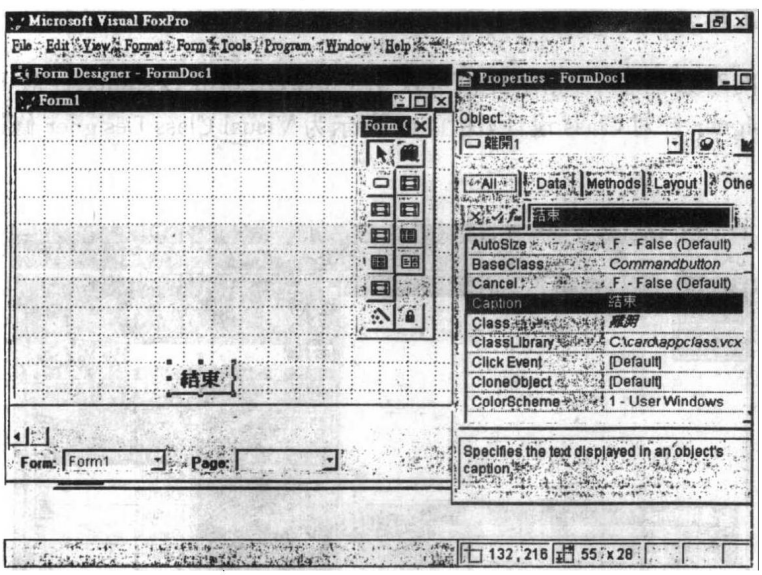


图 1.10　“离开”Class 变成对象的显示状态

这样当系统在开发过程中，面对重用的对象时，可以不必重新设计，直接采用：

Object→Class

Class→Object

的面向对象设计结构，系统开发可以变得非常简单。

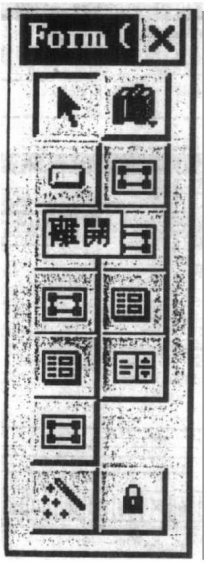


图 1.11 Form Controls Toolbar 换成自行开发的 Class

1.1.4 可视化工具

在 VFP 面向对象程序中，完全提供可视化工具，如 Form Designer 针对 Form 的设计，Visual Class Designer 针对 Class 设计，图 1.12 所示为 Visual Class Designer 修改“离开”Class 的画面。

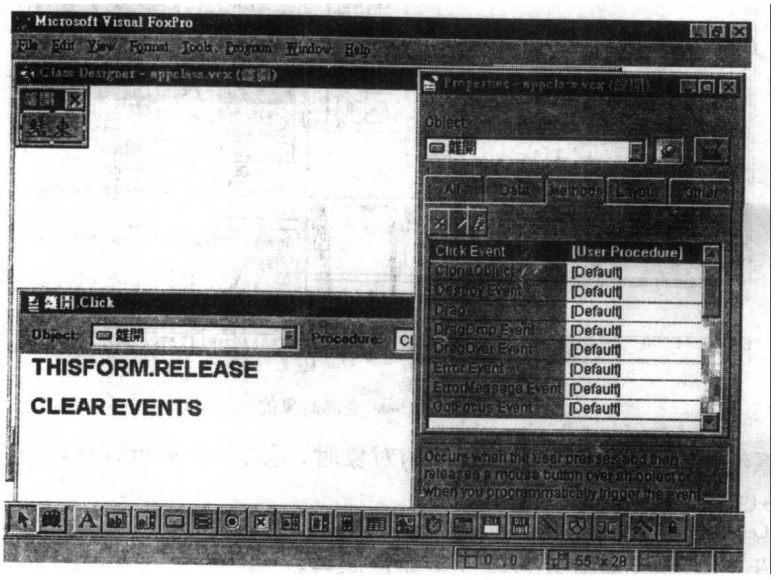


图 1.12 通过 Visual Class Designer 修改 Class

如图 1.12 所示，通过可视化工具新增或修改 Class 就变得非常容易。

1.1.5　方便程序维护

扩展图 1.12 所示的“离开”Class 结果，回到使用这个 Class 所设计的应用系统画面上来，不需要将每一个使用“离开”Class 所设计的模块逐个进行修改。图 1.13 所示，笔者直接执行了以前设计的 Form，不难发现，由“离开”Class 所产生的按钮对象，已经自动重新调整出输出结果，也就是说直接将 Class 修改过的属性与方法，立即反映在应用系统 Form 的命令按钮对象上。

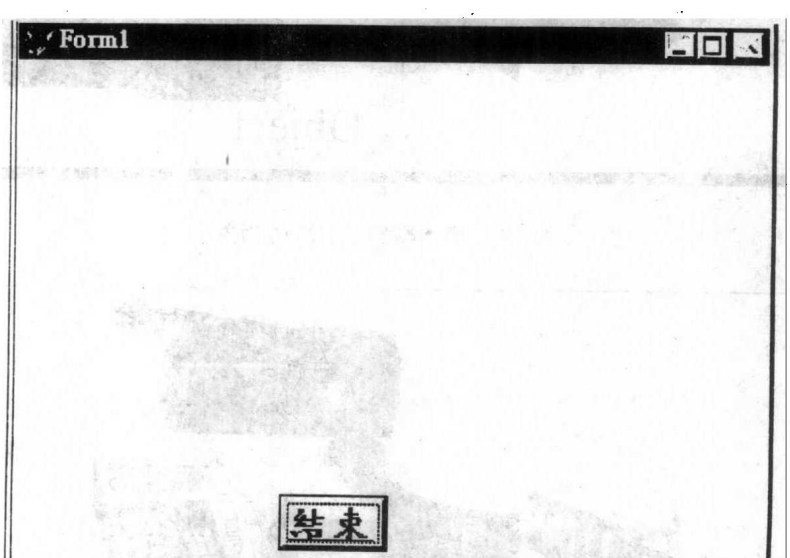


图 1.13　Form 执行结果，命令按钮同时调整

修改 Class 之后，所有使用这个 Class 设计的 Sub-Class 或应用系统，均自动调整，这样使系统维护变得很方便。

1.2　什么是 Class

Class 就字面上来说是：“类”或“分类”，在面向对象程序设计中的意义为“对象的分类就叫做 Class”，在 VFP 中称 Class 为蓝图(BluePrint)，但是笔者认为它是“产生某类对象的制作器”。这样的称呼应该是比较恰当的。

您是否看过小朋友玩积木，通过圆形、方形、三角形、堆积而成的房子或飞机模型。积木就是 Class，而房子、飞机就是对象(东西)。如图 1.14 所示。

在现实世界，可以是一个通用的 IC 元件，分别产生了不同的电子产品，如图 1.15 所示。IC 就是 Class，而组装的产品即是对象。

同样，在面向对象程序设计中，如图 1.16 所示，笔者设计的可视化 Class，命名为水压计

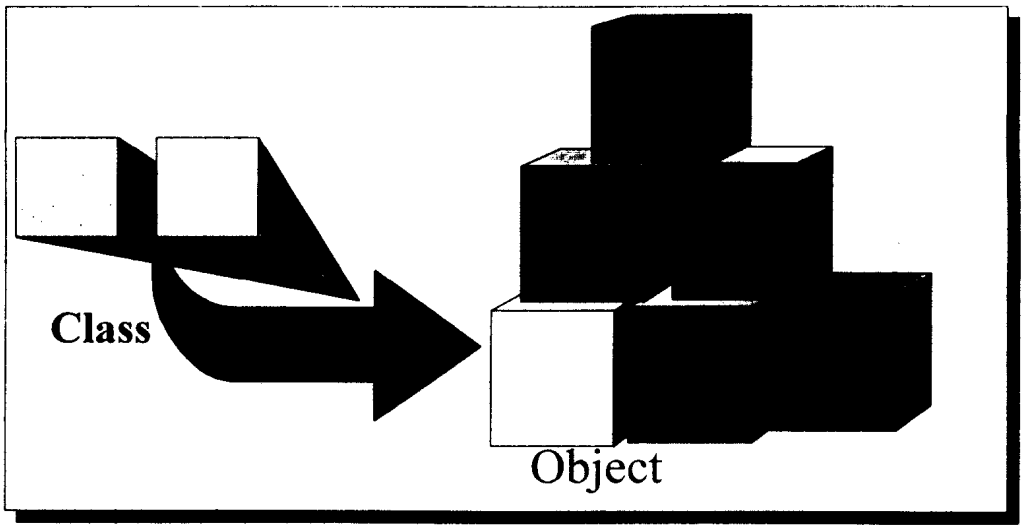


图 1.14 积木式的 Class 变成的对象

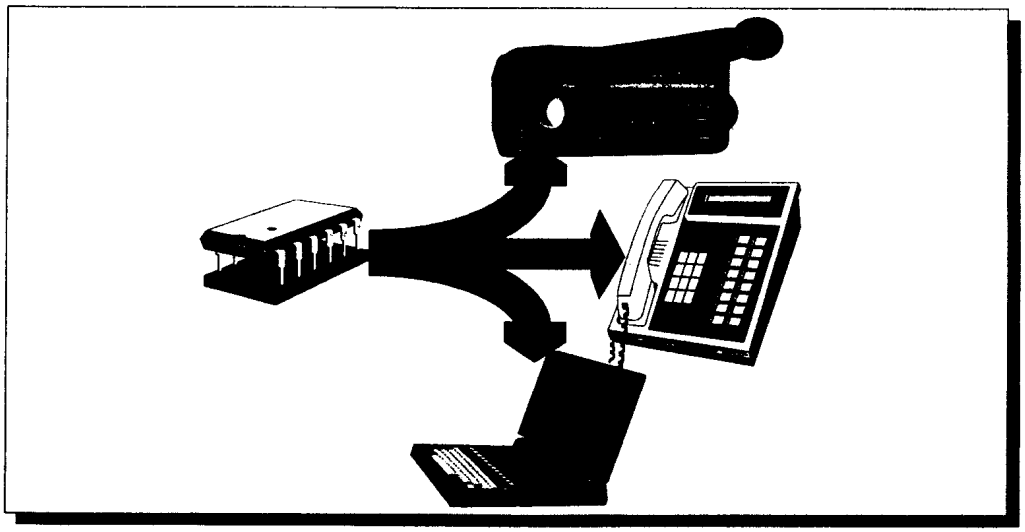


图 1.15 IC 元件可以组装成不同的电子产品

应用，其中包含一个 VB4.0 Gauge.OCX 与 VFP CommandButton Base Class 所组合而成的 Container Class，其方法均设计完整，但是并不是真正可执行，只是具有轮廓(Schema)。

如图 1.16 所示，水压计 Class 并非可执行的意义是，其方法与属性只有定义，但是无法立即执行。在通过使用 Form Designer 工具，将其放置在 Form 画面之后，如图 1.17 所示，执行的结果是在画面上显示出水压计特性，而且每个按钮的特性与功能，都可以通过鼠标触发来执行。这时可以发现，图 1.16 是一种 Container Class 的设计，只是定义某类对象(水压计)的“制作器”，而图 1.17 则是经过 Form Designer 制作完成后形成的一个对象，立即可以执

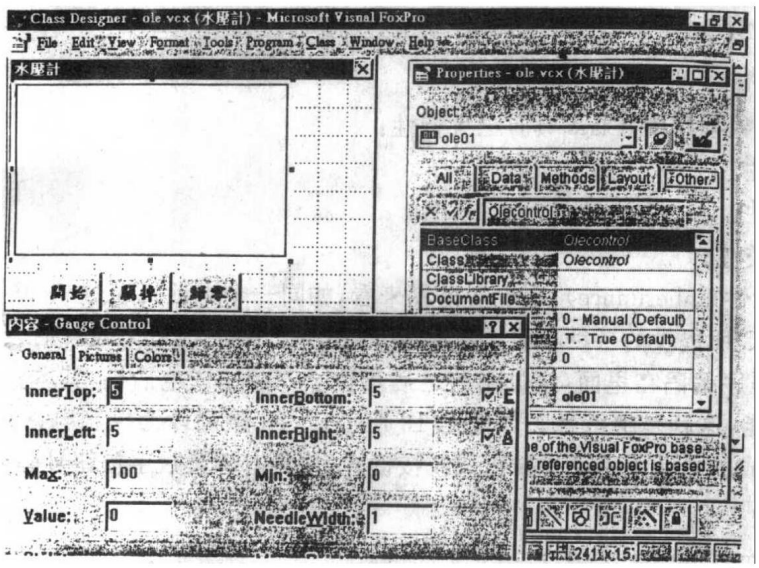


图 1.16　水压计 Container Class

行。

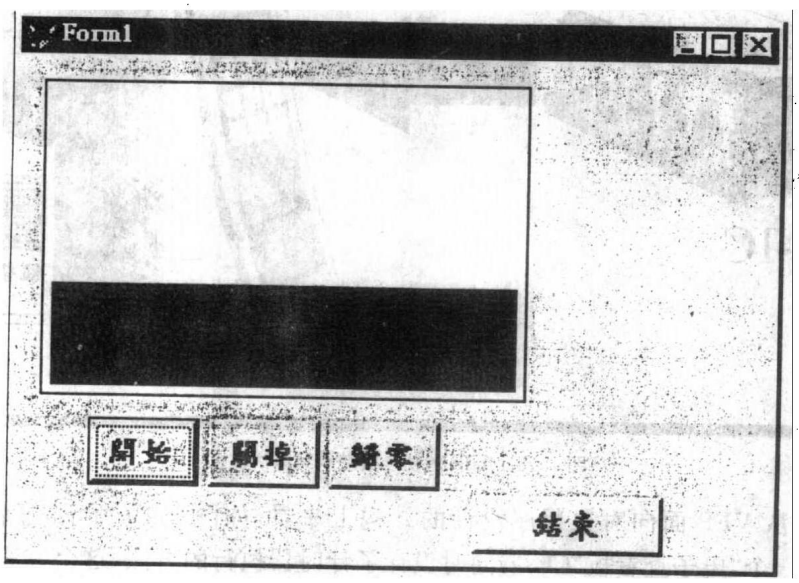


图 1.17　水压计 Class 变成对象之后的执行结果

笔者不涉及很深难懂的面向对象的名词，仅希望借助现实世界与 VFP 可视化工具所表现的可视效果，简单地定义 Class，至于 VFP Class 结构与种类，将会在后面章节中进一步说明。

1.3 Class 的特性

在面向对象程序中，Class 具有三大特性：

- 继承性。
- 封装性。
- 再分类性。

所谓继承性(Inheritance)，在现实世界来看，如同一个电话机通过 A-IC 组合而成，其本身就具有 A-IC 所提供的各种功能。因为 A-IC 在经过 FAE 工程师设计后，已经“烧录”若干功能的处理程序编码在里面，经过组装之后产生了产品——电话机，其本身当然继承原 A-IC 的基本功能，这就是继承性。如图 1.18 所示，A-IC 被设计为控制电话号码数据转换功能，在被组装成的电话中不论是转盘式或按键式，均具有解读数字按钮号码或转盘号码的功能。

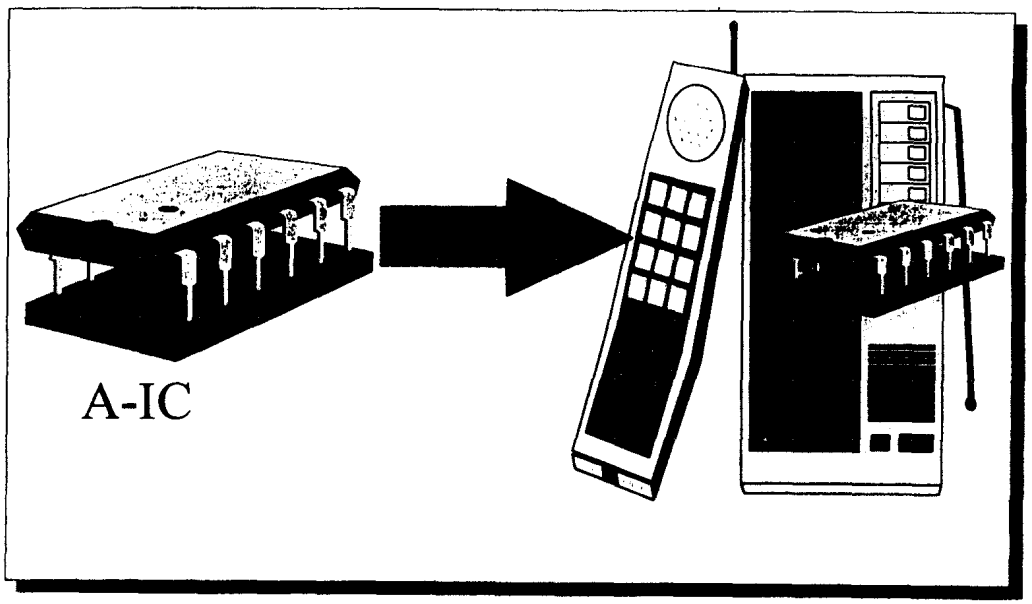


图 1.18 A-IC 的功能被继承到电话机上

同样地，从 VFP 面向对象程序设计的结构上来看，如图 1.19 所示，笔者利用 Visual Class 设计了一个“电子查看板”Class，同时设计了计时器执行的方法，这个方法就如同 A-IC 内所“烧录”的程序编码一样，内容如下：

```
If This.Parent.Label1.Left+This.Parent.Label1.Width>0
    This.Parent.Label1.Left=This.Parent.Label1.Left-20
Else
    This.Parent.Label1.Left=This.Parent.Shape1.Width
Endif
```

通过这样简短的程序，文本串会在设置的时间间隔(Interval)内进行移动，产生电子查