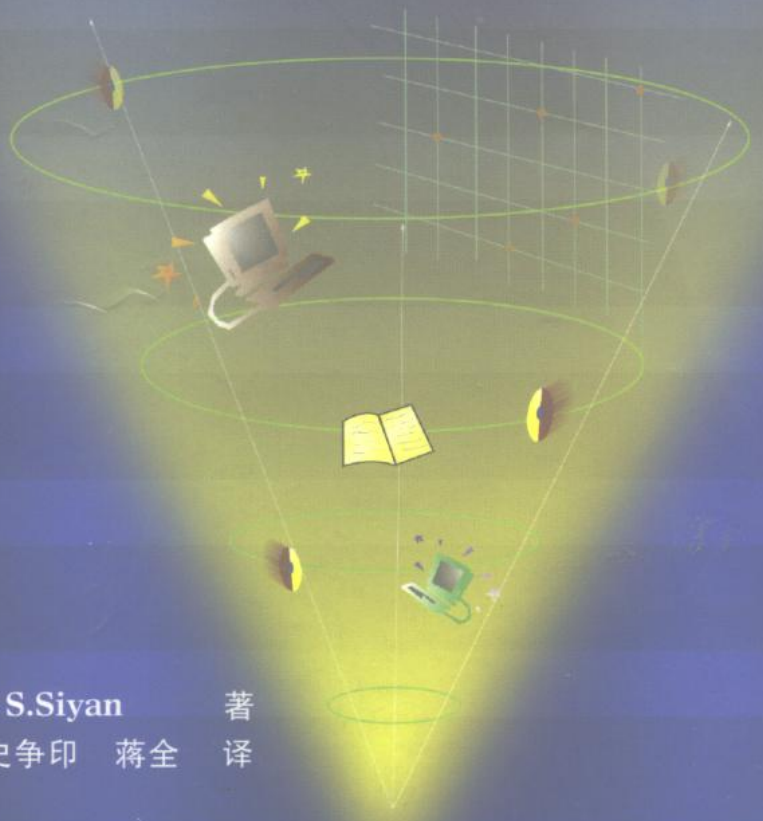
NRP

北京科海培训中心

Visual J++

实用大全



(美) Karanjit S.Siyan 著
丁一青 史争印 蒋全 译

清华大学出版社

NRP 北京科海培训中心

Visual J++实用大全

[美] Karanjit S. Siyan, PH. D 著

丁一青 史争印 蒋全 等译

清华大学出版社

(京)新登字 158 号

著作权合同登记号:01-97-2049

内 容 提 要

本书全面、系统地介绍了 Visual J++和面向解决方案的开发方法。

全书共分 15 章,分别介绍 Visual J++和 J++集成开发环境;通过各个实例逐步讲述编写跨平台的 Java 应用程序、Applets 程序和 ActiveX 控件的方法;介绍如何利用 J++开发多任务的应用程序;如何利用 J++的图形、图像、色彩和声音创建 Java 的动画;如何执行网络操作;如何开发 ActiveX 组件。

本书适合学习和使用 Visual J++语言的各层次人员、特别是高等院校学生和软件开发人员。

Inside Visual J++

书　名:Visual J++实用大全

原著者:Karanjit S. Siyan, PH. D

译　者:丁一青　史争印　蒋全

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

印刷者:北京门头沟胶印厂

发　行:新华书店总店北京科技发行所

开　本:16　印张:35.625　字数:875 千字

版　次:1998 年 5 月第 1 版　1998 年 5 月第 1 次印刷

印　数:00001～6000

书　号:ISBN 7-302-02992-X/TP・1587

定　价:59.00 元

目 录

第1章 Java及Visual J++简介

Java是Sun Microsystems研制的一种崭新的程序设计语言。Java有许多特点，这些特点使得Java特别适用于在像万维网(World Wide Web)这样的因特网(Internet)上开发应用程序，但Java并不仅限用于Internet。Java的跨平台可移植性是非常吸引人的，是许多公司内部网络的良好选择。Java语言已经用于大型软件系统的开发，这表明Java语言可以作为通用工具用于编写商业和科学应用程序。Java支持丰富的类和对象的继承，这就使得程序员可以重新使用现存的代码，而快速地写出复杂的应用程序。

Visual J++是Microsoft研制的程序开发环境，它能用于开发跨平台的Java应用程序，它是Java应用程序开发平台中的一个，Visual J++独一无二的特点是它对Microsoft的ActiveX组件的支持。ActiveX是Microsoft的OLE(对象链接与嵌入)和OCX(对象链接与嵌入客户控制)控件的新名称，有了ActiveX和Java的结合，把Java代码嵌入到COM(Component Object Model，组件对象模型)成为可能。COM是Microsoft使用的基于对象的体系结构。在本书最后两章将会看到更多的关于ActiveX和Java结合的情况，在本章讲述的是Java的简要概述及Visual J++开发环境的特性。

1.1 Java语言

Java语言的起源可追踪到几年前软件工程师的尝试，他们想开发控制家用电器的可移植软件，这类电器如烤箱、烤面包机、电视、录像机、灯光设备、电话、呼机、无线接收机、个人数据助手等等。在1991年4月，一个Sun Microsystem的雇员组开始了代号“Green(绿色)”的项目研制。Green项目的目标就是研制一种开发家用电器逻辑的系统。Sun Microsystem的员工很快认识到在家用电器中实际上没有标准的处理器类型在使用，为了简化家用电器的开发，这些Sun的雇员需要一个独立环境的平台。

James Gosling(UNIX EMACS编辑器及NeWS窗口系统的创建者)在Sun Microsystems最初尝试扩展C++语言，但大家认为这样会付出太多并且不会产生最好的结果。因此，开始了研制Green项目的工作——一种被称为“Oak”的新语言。Oak是当James Gosling进入为新语言设置的目录结构时看到他窗外的橡树而联想到的。由于Oak已被一个原有的语言所用，因此后来不得不放弃这个名字。经过了很长时间的苦思瞑想，小组成员在参观了一家当地的咖啡馆后才突发灵感。因此，与流行的说法相反，Java并不是“Just Another Vague Acronym”的字首缩写。Oak开发平台有四个成分：Oak语言，名为GreenOS的操作系统代码，一个用户接口，以及一个称为*7的类似PDA的设备代码。名称*7出自电话序号，这个电话序号是Green项目成员办公室所在地，Sand Hill办公室用于回答振铃电话的电话序号。*7设想为一个普通的控制设备，它们可以卖给家用电器制造商，这样制造商可以把这种控制设备与他们的产品结合起来。用户界面认为是一个全彩色的人性化的家庭(或办公

室)场景,在那里使用者可以通过触摸屏幕来操纵应用设备。

在 1993 年,Green 项目组合并到 Sun Microsystems 的附属公司,这家附属公司称为"FirstPerson,Inc.",Green 项目组力求得到 Time-Warner TV 的试验,在那里很多家庭声称有试验所需的电视硬件。然而,在 1993 年 6 月之后,Time-Warner 选择了 Silicon Graphics 公司从而取代了 Green 组的建议。1994 年,与 3DO 公司的合作也失败了,并且新市场合伙人的前景暗谈,FirstPerson 的公众发布也被取消,并且新成立的公司也被解散了。Green 项目组的一半成员去为 Sun Interactive 工作,从事于数字视频服务器的研究,而另一半成员则回到母公司 Sun 从事多媒体和基于网络计算的工作。

正当 Green 项目组面临所有这一切困难的时候,Web 获得了广泛的支持,尤其是在给用户提供有相当灵活性和浏览能力的 GUI 客户端开发方面。正是此时 Internet 的增长被 World Wide Web 注入新的力量成为显然的事实。Sun Microsystems 认识到使用 Oak 技术的潜力,于是开发了后来称为"HotJava"的 WebRunner 浏览器。

*7 设备和视频点播(Video On Demand)VOD 技术的开发对 Java 语言的成熟有着巨大的帮助。Java 编译器,最先用 C 写成,后来由 Arthur Van Hoff 用 Java 完成。编译器由一些软件组合而成,Java 编译器用 Java 语言的重写表明了 Java 的通用特性。

1995 年 5 月 23 日,在 Sun World '95 发布会上 Sun Microsystems 发布了 Java Environment(Java 环境),此后人们对 Java 的兴趣及活动就是证明,并且表明了这项技术将主宰网络计算的未来。

像 Netscape Navigator 和 Internet Explorer 这些流行的浏览器都溶入了基于 Java 的技术,这些浏览器可以运行从远程 Web 服务器上下载的 Java 程序。Web 上的 Java 程序称为 Java 小应用程序(applets)。Java 小应用程序的描述被嵌入到 HTML 页中作为 URL 地址。因为 URL 地址可以是指网络上的任何计算机,所以 HTTP 协议和 HTML 语言可用于在 Intranet 或 Internet 上以分布代码的方式执行。

1.1.1 独特的特性

Sun 把 Java 语言描述为简单的、分布式的、可解释的、安全的、独立于体系结构的、可移植的、高性能的、多线程的以及动态的语言,那么多的形容词,以至于有人把它们归为 buzzwords(在信息搜索中无意义的字或词)。

这些 buzzwords 有的也用于描述其他语言,但是 Java 独一无二的特点是第一个既能用于编写通用的程序,也可以专用于 Internet 和 Intranet 上的程序设计。当用于 Internet/Intranet 应用程序时,Java 程序一般用于与像 Web 浏览器这样的工具联接浏览 Web 页。那些设计为在 Web 浏览器内部运行的 Java 程序被称为 Applet(小应用程序)。在 Web 浏览器之外独立运行的通用 Java 程序被称为 Java 应用程序,在本书后面的部分将介绍更多的关于这两种程序在编写方法上的差别。

1.1.2 使用的简易性

设计 Java 的目的是能够简单而高效地表现 Web 网络计算的功能,这种简易性和高效性在开发 Intranet 的应用程序时也是有帮助的。为了达到简单化的目的,语言的设计者尽量把语言构造的数量变小,这样便于人们学习这种语言,同时也使编译器更小和更易于实现。

设计者把该语言建立在 C/C++语法的基础上，这样大部分使用 C/C++的程序员会发现这种语言很容易学而且易于转向使用这种语言。有意删除了一些 C/C++的特征使得 Java 语言简单而且安全。下面举例说明这个问题：Java 语言不支持 goto 语句，但它提供了 exceptional handling(例外处理)、labeled break(标号中止)、continue 和 finally 语句。C/C++语言的头文件也被删除，已经没有 #include 预处理语句；相反，Java 的 import 语句用于选择性地输入一个 Java 类或指定包(package)中的所有类。

为了支持模块风格的大型软件开发，Java 使用了包的概念，一个包就是按类似功能组合在一起的 Java 代码的集合。

与 C/C++其他的不同之处包括删除了对 struct 和 union 这种数据结构的支持，类似的概念由 class construct(类构造)来实现，另外 C++中操作符的重载及多继承的功能也被删除，这样保持了 Java 语言的简单。

从 C/C++中最大的变化之一就是删除了指针的直接使用。虽然在 C/C++中指针是非常强有力的机制，但它们需要正确地使用并且保证使用它们的代码是无错误的，但从使用 C/C++代码的现实经历表明，在 C/C++中指针的使用产生了面向程序的错误。Java 自动地处理语言对象的引用和撤消引用，这样把程序员从复杂易错的指针引用、无效的指针引用及存储丢失等问题中解脱出来。Java 环境动态创建对象，并且在后台自动实现垃圾回收。错误的内存分配是另一个在 C/C++程序中常见的错误根源，Java 通过其环境自动地完成这一功能，内存类型的错误就减少了。

虽然 Java 语言比 C/C++简单化了，而且带来了丰富的预定义的类库来完成 I/O、网络及图形操作，这样使 Java 语言非常容易学习而且功能强大，能够充分地开发网络支持的 Intranet 应用程序。

1.1.3　面向对象的性质

Java 是一种面向对象的语言，所有的程序和数据存在于对象的上下文中。一个对象就是数据及对数据进行操作的程序的集合。专门为数据编写的程序驻留在对象之中被称为对象方法(object method)。事实上，方法的参数之一就是对象本身(在 Java 语言中，它是一个特殊的保留字“this”)。正如名字“方法”所指，对象方法就是在对象内部用于操作数据的机制，数据和方法描述了对象的状态。

在 Java 中对象的概念由类构造(class construct)来实现。类的使用是 Java 语言中最基础的，因此想不使用类构造而去完成某些有意义的 Java 程序是不可能的。

在 Java 中通过使用对象继承(object inheritance)来支持代码重用。一个类可以由另一个类派生而来，这个过程称为继承(inheritance)或生成子类(subclassing)。Java 语言带有非常有用的类层次(class hierarchies)。在最顶层是一个特殊类——Object，这是根类。其他所有的类都是它的子类。缺省状态下，一个新创建的类总是继承于根类 Object，尽管不用明确地定义从哪里继承的类。预定义的 Java 类层次提供了丰富的 Java 环境。

事实上，在 Java 中最大的学习障碍并不来自于语言的语法或语言的语义，这两者相对来说都比较容易学，最大的挑战在于熟悉预定义的类层次和方法以及它们的作用。当然，初学者可能会打破并重新创建一个新的类层次的集合去代替现存的类层次，但是这样降低了效率。

通过使用继承,总是可以使用代码重用的概念及子类存在的类层次,在需要的类层次不充分或不可用时,尽管去创建自己的类层次。

1.1.4 分布式语言

Java 设计成可以支持网络上运行的应用程序,这种网络可以是 Internet 或一个企业的 Intranet。Java 使用了一个称为 Java.net 的预定义语言包提供网络能力。这个包包含有许多用于简化在网络上不同计算机中运行的应用程序之间进行网络通信的类。使用 Java 可以同样轻松地访问远程或本地文件。另外,该语言支持一个虚电路网络连接,这样可以用来为 Intranets 创建分布式 Client/Server(客户机/服务器)的软件。

1.1.5 解释语言

Java 编译器并不产生完全可执行 Java 程序的机器语言指令,相反,Java 编译器产生一种称为字节码(byte codes)的中间代码,Java 字节码被 Java 解释器读取并执行,这个过程是使用一个内部的抽象机器模型来完成。Java 解释器和这个抽象机器的实现,被称为 Java 虚拟机(Java Virtual Machine),简称 JVM。

Java 字节码从结构上看是中间代码型程序的代表,由于 Java 程序是通过解释 Java 字节码来执行的,因此 Java 语言是解释性的。在一个解释环境中标准的"link"阶段——即把目标模型连在一起形成一个二进制可执行机器指令的阶段消失了,连接阶段由把新类加载到 Java 环境的阶段而代替了。像 Java 这类的解释环境支持快速原型及程序的开发。

及时(Just In Time 简称 JIT)编译器对那些不能负担 Java 代码解释执行开销的应用程序来说仍是可用的。这些 JIT 编译器把字节码翻译成程序所运行的计算机上的机器语言指令,这样带来性能上的巨大提高。当然,JIT 编译器的输出不再是中间代码了。

1.1.6 稳固的语言

当程序员在一个语言中谈论稳固性时,他们认为是对减少编译和运行时的错误的支持。Java 是一种强类型的语言,这就是意味着在如何使用类的问题上是强约束的。例如,不能在不失精度的情况下把一个浮点数赋给一个整型数值。编译器在编译时将发现这些错误以及类似的错误,但是可以通过显式的类型转换或通过编写一个转换程序明确地把某对象的类型强制转换为其他类型,而这种显式的操作由程序员完成而不是由 Java 语言自动地实现的。

Java 存储模型实现自动垃圾回收,这样排除了回收内存类产生运行错误的机会。不支持直接的指针,这样排除了与指针有关的运行错误,诸如不当心的数据重写以及存储的破坏等。Java 解释器也实现运行时的检查,比如保证数组和字符串的存取限制在数组大小的边界内。

Java 支持语言中显式的例外处理,这就给了程序员提供了一个附加的工具去编写稳固的程序。除了已预定义好的指明潜在问题的例外情况,程序员也可以定义自己的例外情况。

1.1.7　安全的语言

由于 Java 要运行在带有未经确认的主机的网络化环境中，因此安全性是语言设计的重要目标之一，驻留在 Web 服务器中的 Java 代码能够被 Web 浏览器提供的 Java 虚拟机下载并且运行，没有任何措施阻止病毒和其他恶意的程序伪装为合法的代码并且破坏客户计算机或网络上其他计算机。

一项许多病毒程序用来导致破坏的技术是通过灵活地操纵像指针这类地址变量来占用机器资源，这就是 Java 的设计者决定不支持指针的原因之一，这样也排除了一个主要的安全上的危险性。

内存分配以及类的设计是由 Java 环境透明地完成的。由于程序员没有权限去进行内存的设计，因此程序员就不知道他所使用的实际内存设计。这样使病毒程序很难访问 Java 程序的内部数据结构。

在一个 Java 程序被解释之前，Java 运行期间系统完成字节码验证（byte-code verification）。字节码验证是正式的处理，在处理过程中使用数学算法保证程序不破坏系统的完整性。另外，通过网络加载的程序被装到一个与本地类不同的名字空间内。于是防止了被篡改的 Java 小应用程序代替标准的 Java 类。

Java 反对使用会引起程序不正常运行的传统技术。虽然它不能保证或声明 100%的不会失败的或无差错的环境，但是很多恶意的、狡诈的“黑客”将看到在 Java 中有更多障碍。而当这些将要发生的时候，Sun Microsystems 将先于这些“黑客”一步堵上安全漏洞。

虽然在 Internet 环境中安全性是重要的，但在 Intranet 环境中安全性更为重要，主要原因是 Intranet 比 Internet 更少暴露给恶意的用户，由于 Java 的设计是在 Internet 上提供一个安全的执行环境，当然它能够满足大多数 Intranet 的需要。

1.1.8　体系结构的中立性

一个编译后的 Java 程序产生在 Java 虚拟机解释的字节码，这种字节码独立于任何特定的处理器类型和机器的体系结构，使用字节码使得 Java 代码可以在任何支持 Java 虚拟机和 Java 解释器的机器上运行。

Java 结构的中立性对于 Intranet 是非常重要的，因为它使得在 Java 中的程序代码仅编写一次，同样的代码则可以在不同客户机或服务器的平台上运行。图 1.1 说明了同一个 Java 小应用程序代码下载到不同的客户机平台。

当前，许多团体和机构花费了很多的人力、物力来努力满足他们需要支持的每一种可能平台。对于不同版本的 UNIX、Windows NT、Windows 95、OS/2 以及 Macintosh 计算机，要编写出可以在所有这些平台上运行的软件是一个挑战，而 Java 在对付这个挑战方面已走了很长的路。

1.1.9　可移植的平台

由于 Java 代码是独立于体系结构的，它能够运行在任何平台上，这种代码只需编写一次，则字节码可以分布到不同平台上运行而不需要改变。在其他语言中可移植性问题不变的原因在于整型、字符型和浮点数这些基本类型大小在具体实现中存在着不同。

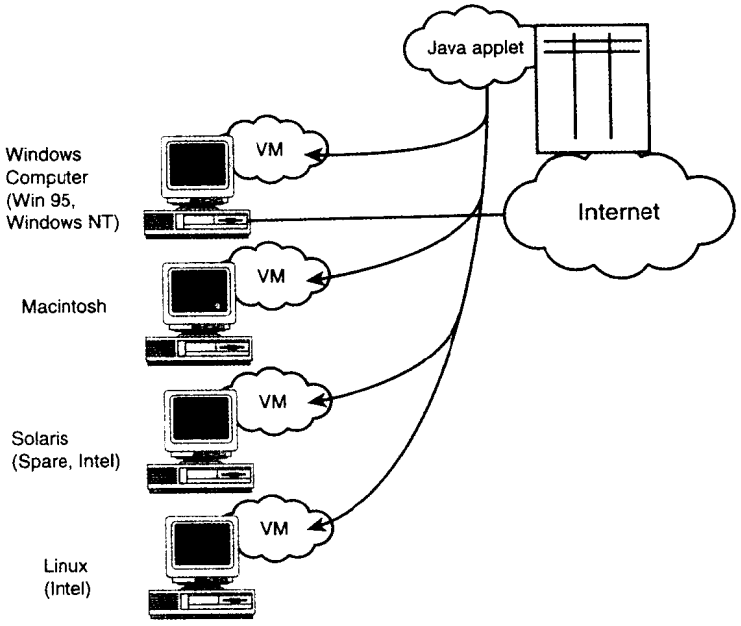


图 1.1 Java 代码在独立的环境的平台上运行

例如，程序员需要知道一个基本的整数类型是 16 位长还是 32 位长。在 Java 中，所有的基本类型不论它运行的平台是什么都是同样大小的。例如，Java 中的整数类型总是有符号的 32 位值，不论它运行在 UNIX 机器上还是运行在 OS/2 机器上。使用一个标准的整数类型定义能够避免诸如上溢或下溢这样的错误，这类错误一般是由于假定了不正确的基本类型大小。

由于 Java 小应用程序是基于 GUI 的并且可被多线程化，因此 Java 程序只能在具有这种特征的环境中运行。例如，让 Java 代码运行在不支持 GUI 或多任务的 MS-DOS 机器中则是非常困难的。

1.1.10 高性能的工具

Java 与其他的 script 语言，比如 Basic 和 Visual Basic语言的多样性、外壳 Scripts 以及 Perl 等相比是高性能的。然而，它比 C 语言慢 20 倍，但对很多交互式应用程序，Java 的速度已足够了。

如果想把 Java 的速度与 C 语言相比较，则不得不使用 Just In Time 编译器，JIT 的实例可以从很多厂家得到，比如 Sun、Borland 和 Symantec 等等。Web 浏览器允许 JIT 编译器提供加速器来提高其小应用程序的性能。Java 的字节码可以提供一种快速从字节码格式到处理器机器指令的翻译。因此字节码转变到机器指令的性能是可以与 C/C++程序相比的。

1.1.11 多线程语言

Java 是几个为数不多的可以在语言自身内部用多线程的方式支持多任务的语言之一，这就意味着可以编写出有多个线程“同时”被执行的 Java 程序，一个线程就是穿过程序的一

条独立的执行线。线程是可用在程序中实现的一层并行机制。每个线程完成一个特定的功能，随着几个线程的执行，就发生了几个并行的活动。一个单处理器的机器，线程不得不轮流在 CPU 上执行，因此它只能表现为同时执行。事实上，每个线程在拥有 CPU 的运行权被抢占之前执行一小段时间，然后另一个线程才能得到机会运行。在多处理器机器上，线程可以同时在不同处理器上执行。为了防止线程在出现互斥操作时互相干扰，可以通过使用同步关键字指明临界区(critical regions)。

许多应用某形式的多任务或多线程的程序都是运用低层系统服务调用来操纵它们运行的系统。由于系统服务调用上的不同，这些程序对不同的操作系统类型而言是不能移植的。然而，运用 Java 语言可以编写出一个能运行不同计算机和操作系统上的多线程程序。

1.1.12　动态的语言

Java 在这种意义上是动态的语言：即当需要的时候就去加载类。可以通过检查与类相关的运行类型信息决定在运行期间一个对象属于哪一个类。

1.2　Visual J＋＋概述

Visual J＋＋是开发 Java 应用程序和小应用程序的强有力的工具，Visual J＋＋目前运行于 Windows 95 和 Windows NT 计算机平台，但是 Java 代码可以在任何支持 Java 运行时环境的平台上运行。Visual J＋＋主要包含如下特点：

- 集成化开发环境(Integrated Development Environment)
- AppWizards 用来简化 Java 程序设计的细节
- 基于项目的开发系统
- 符号化的调试能力

Visual J＋＋围绕 Developer Studio 建立，Developer Studio 是常见的 Microsoft Development 环境。这个常见的开发环境也用于其他程序设计工具中，比如 Microsoft 的 Visual C＋＋。

1.2.1　Visual J＋＋集成化开发环境

Microsoft Developer Studio 提供了开发 Java 应用程序和小应用程序(见图 1.2)的集成化开发环境。它由一组集成化的运行于 Windows 95 或 Windows NT 的可视化工具集组成，通过使用这些工具，可以编写、测试/调试以及在同一个环境下优化 Java 应用程序。

术语“集成化”在这里是指具有统一格式界面的工具集，这种统一格式的界面可以用来在 Java 程序的开发中协调合作。例如，这个集成化工具包括一个文本编辑器、资源编辑器、项目管理器、一个优化的编译器、一个增量链接器、一个源代码浏览窗口、一个集成的调试器和联机手册。

这些工具的操作可以由单一的应用程序控制。例如，可以在文本编辑器中选择一个变量的名字(见图 1.3)，然后把它拖放到调试器中的“Watch”(监视)窗口中(见图 1.4)。监视窗口可以显示当前变量的值。即使它发生改变，当变量被拖入监视窗口，调试器立即计算它并