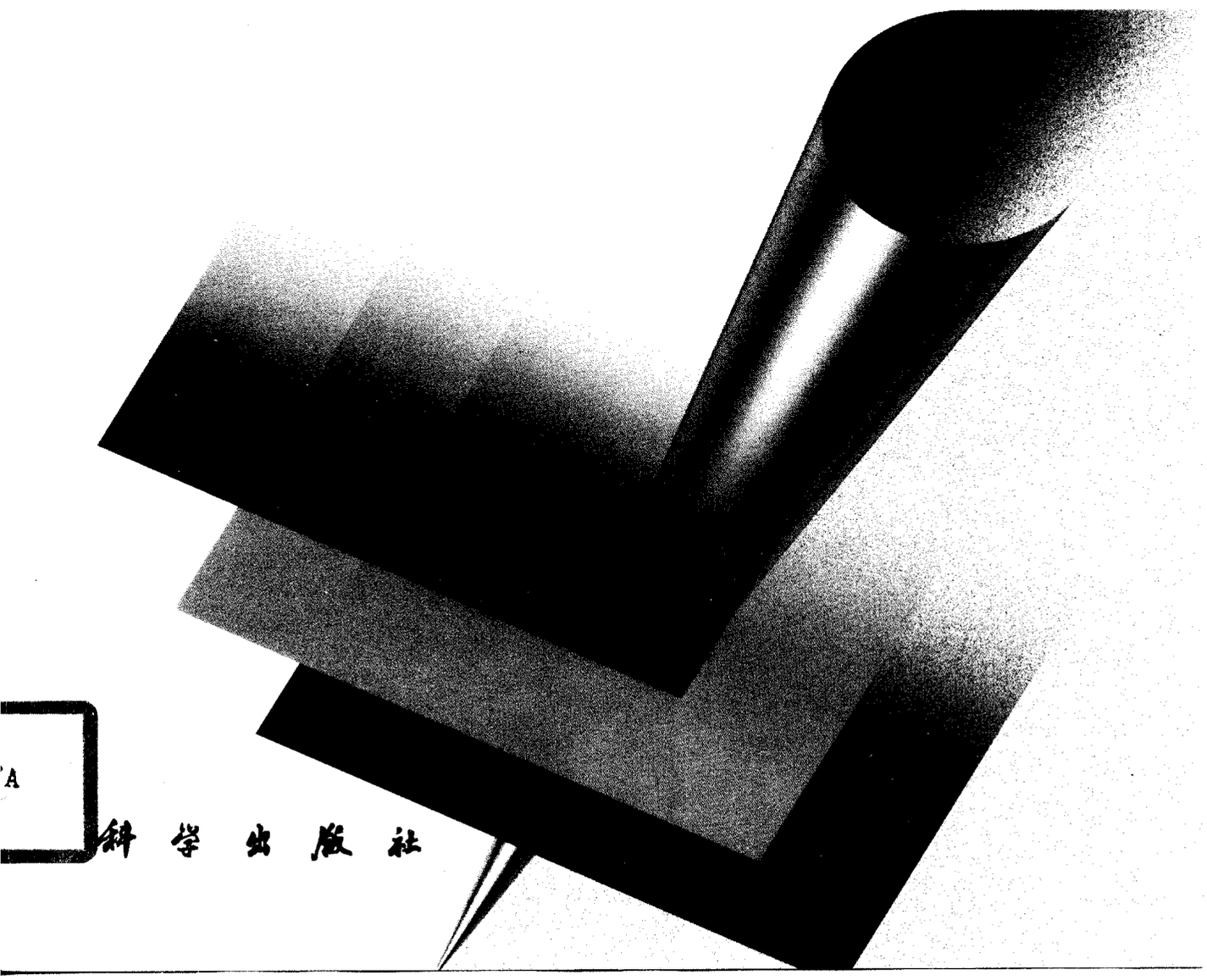
计算机软件工具应用与开发系列

MS Visual J++ 应用开发指南

鸿志创作组 编著



科学出版社

内 容 简 介

本书通过详细的图解和大量的实例介绍了怎样使用微软 Java 编程工具——Visual J++编写、调试、运行、开发 Java 程序。全书内容分为两个层面：Visual J++应用入门和 Visual J++编程与开发。第一个层面着重讲述 Visual J++开发环境以及各种开发工具的功能；第二个层面主要介绍使用 Visual J++编写 Java 程序，包括多线程、动画、数字签名和 ActiveX 控件等比较深入的应用。

本书适合于计算机用户、大专院校师生，以及 Visual J++和 Java 编程人员阅读参考。

图书在版编目(CIP)数据

MS Visual J++应用开发指南/鸿志创作组 编著 .-北京：科学出版社，1998.3

(计算机软件工具应用与开发系列)

ISBN 7-03-006246-9

Ⅰ.M… Ⅱ.鸿… Ⅲ.Java 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(97)第 25283 号

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100717

北京双青印刷厂 印刷

新华书店北京发行所发行 各地新华书店经售

*

1998 年 3 月第 一 版 开本：787×1092 1/16

1998 年 3 月第一次印刷 印张：25

印数：1—4 000 字数：581 000

定价：35.00 元

前　言

在我们第一次使用 Visual J++之前，就曾经接触过 Java 语言。那时候主要是因为从报纸和杂志上看到一轮又一轮关于 Java 的宣传，终于没有忍耐住动手一试的好奇心，便从网上下载了 Java JDK。我们迫不急待地编写了几个小程序之后，渐渐地感到有些失望。这倒不是因为 Java 语言自身的问题，而是因为 Java JDK 开发工具对用惯了集成开发环境的人来说实在是一种"折磨"。当时作者就想：怎么没有一个比较好的开发环境呢？

因为用不惯 JDK 单调而烦琐的行命令，Java 语言也就被丢在一边了。直到有一天拿到了 Visual J++，这一切才得到了改变。当 Visual J++在我们的机器上成功地运行时，我们看见了非常熟悉的工作界面，那种风格和品位绝对是微软的手笔。

有了 Visual J++之后，学习 Java 的兴趣立刻也跟着高涨起来。简单地阅读了一下联机帮助之后，就用 Java Applet Wizard 快速地生成了第一个程序，而且还具有动画功能。由此，极大地激发了我们的创作欲望，接着就一口气地学了下去。随着时间的推移，我们越来越感到 Visual J++功能的强大。从文本编辑器到编译器、调试器，就连 infoView 这样查看联机帮助的工具都做得精巧美观，更不要说还提供了简化界面设计、使用 ActiveX 构件、定制工具条和菜单、自动化日常工作等多种工具。

有了一些使用 Visual J++的体会和经验后，自然而然地产生了想把它介绍给大家的念头。因为 Visual J++是一种开发 Java 语言的优秀平台，可以使你更快更好地开发 Java 程序。正是基于这样一种想法，我们开始动手编写本书。

本书是为初次使用 Visual J++的读者而写的，全书结构分为两个部分：

第一部分——Visual J++应用入门　这一部分介绍 Java 语言及其基础知识，Visual J++的集成开发环境和各种工具。共分为十三章。

第 1 章简介 Java 语言；第 2 章和第 3 章介绍 Visual J++集成开发环境及其编译器和解释器；第 4 章和第 5 章介绍 Java 语言的基础知识。从第 6 章开始，我们介绍 Visual J++提供的各种工具。其中第 6 章介绍 Java Applet Wizard；第 7 章介绍项目和项目工作区；第 8 章介绍使用 ClassView 进行类的管理；第 9 章介绍调试工具的使用；第 10 章讲述资源编辑器和文本编辑器；第 11 章介绍 Java Resource Wizard；第 12 章介绍定制菜单和工具条；第 13 章介绍怎样利用 VBScript 宏实现工作自动化。

第二部分——Visual J++编程与开发　这一部分着重介绍使用 Visual J++，以及 Visual J++提供的一些比较复杂的工具进行 Java 程序编制和开发。共分为七章。

第 14 章介绍 Java 的 AWT 类库的编程；第 15 章介绍使用类 Graphics 进行图形的绘制和图象的载入显示；第 16 章介绍 Java 提供的多线程机制；第 17 章介绍动态和动画技术；第 18 章介绍例外处理；第 19 章介绍 Cabinet 文件的创建和数字签名技术；第 20 章介绍 Java 与 ActiveX 构件的共用。

经过长时间的编写工作，本书终于完成了。在写作过程中，我们遇到了许多的问题和困难，也得到了许多人的帮助。首先要感谢出版社的编辑和工作人员，在他们的热心帮助

下才使本书得以问世;还要感谢李真文先生对本书总体构思上的建议;最后要感谢我们的家人和朋友,没有他们的支持,本书的写作是无法完成的。

本书由陆鑫主笔,创作组其他成员分别在各章参与写作。

一本书的优劣要由读者来判断,希望这本书能对读者的学习有所帮助。书中难免会有疏漏和错误,敬请读者批评指正。

鸿志创作组

1997年8月7日于北京

目　录

1 Java 语言

就像飞行员不会不知道波音飞机一样，喜爱计算机的朋友们中现在恐怕没有不知道Java语言的。Java 语言从发明到现在只不过两三年的时间，就一跃成为计算机界的“明星”，成为众多公司竞相推崇的编程语言。更有许多专家认为，Java 语言将对软件业带来冲击，甚至带来一次“软件革命”。

Java 语言到底有何“魔力”，能产生如此巨大的影响？它与传统的语言有哪些差别呢？看完本章后，你就会明白了。

1.1 Java 语言的产生

这一节虽然没有提到 Java 语言的技术问题，但是了解 Java 语言产生的背景，对掌握Java语言是有帮助的。

1.1.1 Java 的原型 Oak

Java 语言的成长经历与其他计算机语言不一样，它最初应用的对象并不是计算机，而是消费性电子产品(即 PDA、电视游戏机、电视置顶盒之类的产品)。为了进入消费性电子产品市场，Sun 公司专门成立了一个项目开发小组，其成员有 Gosling，Naughton 等人。他们的目标是要设计出能嵌在消费性电子产品中的小型分布系统软件，这种软件适用于异构网络、多主机体系结构，并且可以实现信息的安全传递。

开发小组把这个开发项目取名 Green，并考虑使用C + + 语言来完成这个目标，但是很快他们发现，C + + 语言的复杂性和不安全性，使它不能胜任这项工作。因此他们不得不自己开发出一种新的语言，这就是我们现在看到的 Oak 语言。

1.1.2 迁移到 Internet

Oak 语言作为 Green 项目开发组的产品虽然在技术上是成功的，但是在消费性电子产品市场上却没有获得青睐。在经过了几次失败后，这个新产品在原来定位的市场上似乎找不到用武之地。

但是 Oak 语言的遭遇正应验了中国的一句古话“塞翁失马，焉知非福”，在消费性电子产品市场上无所作为的 Oak 语言，却在 Internet 上找到大显身手的机会。

当然，这个机会的产生与 Internet 的飞速发展是分不开的。1989 年，Internet 上发生了一件大事，位于日内瓦的欧洲基本粒子物理实验室(CERN) 的科学家发明了 World Wide Web网络传输协议，这种协议可以在文本中插入图片和声音，从而使原本单调的Internet世界变得图文并茂了。1993 年，美国伊利诺依大学的开发人员发布了

Mosiac——第一个符合 WWW 协议的浏览器,从此人们终于可以通过浏览器来"漫游"丰富多彩的Internet世界了。

WWW 页面虽然已经拥有了图文和声音,但是仍然是静态的,不具备交互性。要让页面拥有动态的画面,并能进行交互,一种很自然的想法就是在 WWW 页面中嵌入一段程序。考虑到当时 Internet 是由数以千计不同种类的计算机组成的,所以编制该程序的语言必须具有平台无关性;由于网络传输带宽的限制,因此这种语言必须简练,其支撑环境必须很小;另外,由于是在网络环境中使用,对安全性也提出了很高的要求。而 Oak 语言恰好能够满足这些要求。看准了这一点后,Sun 公司决定把这种语言引入 Internet。

1.1.3 从 Oak 到 Java

在 Oak 语言正式应用于 Internet 之前,当然还需要进行一些完善工作。但是更重要的是要有一个能支持它的 WWW 浏览器,可以使人们真正地看到激动人心的充满动感的 WWW 页面。为达到这个目标,Green 开发小组的 Gosling 开始继续完善 Oak 语言,而 Naughton则着手用这种语言编写一个 WWW 浏览器。

这两位非凡的程序员不负众望,不久之后 Gosling 的 Oak 语言新版本诞生了,它被赋予了一个新的名字——Java。与此同时 Naughton 也完成了第一个支持 Java 语言的 WWW 浏览器——HotJava。

Java 和 HotJava 的出现,改变了 Internet。在此之前,Internet 只是充满了一页页文档的"信息库",你所能做的只是浏览,你永远只能被动地接受。现在,一切都改变了,有了Java语言的支持,你可以实现真正的交互。你可以在 Internet 上逛商店,看到喜欢的东西,就可以买下来;你也可以"走进"一家旅行社,这时 Office 小姐将热情地向你推荐最新的旅游热线,面对配有解说的美丽的风景画,甚至一段优美的录象片,相信你很难拒绝做一次浪漫的旅行。

1.1.4 Java 冲击波

一种新的技术要想迅速地占领市场,就要有天时、地利、人和。Java 语言产生时,Internet上可以应用的语言几乎都不成气候,这就为 Java 的发展创造了天时和地利。而要达到人和的目的,扩大用户的数量,最见成效的策略就是让用户免费使用自己的产品。Sun 公司正是采用了这个策略。

为了使 Java 迅速占领市场,Sun 公司免费发布了 Java 语言的完整技术规范和开发环境,用户可以从 Internet 上免费下载。这一招果然见效,以 Netscape 为首的众多 WWW 厂商立即宣布支持 Java 语言。许多著名的计算机公司也都不甘落后,纷纷宣布 Java 语言的支持,都想利用 Java 语言在 Internet 市场中占有一席之地。

Netscape 更是领先一步,首先在其浏览器 Navigator 2.0 的 32 位版本中实现了对 Java 语言的支持。Java 语言对 Internet 市场的强大冲击和飞速发展的美好前景使一直冷眼旁观的"软件巨人" Microsoft 再也坐不住了,不得不开始支持 Java 语言,并在其浏览器 Internet Explorer中加以实现。

总之,Java 语言的出现对软件业的冲击是巨大的,尤其是它与 Internet 日趋完美的结合,必将带来一次"软件革命"。

1.2 Java语言的特点

在前面一节中,我们介绍了Java语言的产生,其中也提到了Java语言的一些特点,如平台无关、安全性等。在这一节中,我们将重点阐述Java语言的特点。

1.2.1 Java语言概述

Java语言是适用于分布式计算环境的面向对象编程的语言,它的设计非常简单,而且类似于C和C++,所以熟悉C和C++的程序员可以很快掌握这门语言。

虽然Java和C,C++相关,但是也有很大的不同。它具有C和C++忽略了的许多特性并包括了少量其他语言的特性。它是一种商品化的语言,而不是研究性的语言,也就是说在设计上避免包含任何新的和未经验证的特性。

Java是强类型语言。这种类型语言的特点是明确地区分能够和必须被检测的编译时错,以及那些运行时产生的错误。编译时(compile-time)指的是把Java源程序翻译成与机器无关的字节码(bytecode)的过程。运行时(run-time)的活动有装载和链接需要执行的类、程序的动态优化和程序的执行。

Java语言是相对高级的语言,使用这种语言是无法利用机器的细节特性的。Java语言具有自动存储管理功能,典型的方法是使用垃圾收集器(garbage collector),以避免显示地撤消分配所引起的安全问题(就像C语言中的free和C++的delete)。其高性能的垃圾收集的实现限制了它对系统编程和实时应用程序的支持。Java语言不包含任何不安全的构造,例如:指针、没有下标检查的数组,因为任何不安全的构造都可能带来程序执行的不确定性。

1.2.2 Java语言的特点

Java语言的特点包括以下几点:面向对象、平台无关性、安全性、分布式计算、多线程、易学性等。下面我们就逐一介绍这些特点。

1. 面向对象

面向对象技术对于今天的软件开发人员来说,是最津津乐道的话题。Java语言也是一种面向对象的编程语言。虽然人们已经知道了许多面向对象的编程语言,如C++,Ada,SmallTalk等等,但是他们却都不是纯粹的面向对象编程语言,而是过程式语言加面向对象的扩展。

Java语言是"纯"面向对象的语言。它的所有数据类型,包括布尔类型、字符类型都有相应的类。整个语言都是基于对象的。

面向对象编程语言有三个特点:封装性、多态性和继承性。所谓封装性就是把数据及对数据的操作用类的形式集合起来,外界对类中数据进行操作受到一定的限制。多态性是指不同的对象对同一种信息,可以按照对象本身的性质进行响应。继承性是对象的一种层次关系的体现,下层对象是由上层对象派生出来的,它继承了上层对象的全部特性,同时又有一定的改变和扩充以适应于其自身的特点。Java语言很好地实现了这三种特

性。

2. 平台无关性

平台无关性实际上就是可移植性。它是指程序不经修改就可以在不同的平台(特定的硬件及其操作系统)上运行的特性。可移植性好的语言可以很容易地实现跨平台的操作。

由于 Java 语言在设计时就重点考虑了可移植性,所以它真正实现了脱离平台独立地运行。Java 采用了多种机制来保证可移植性,其中最主要的设计思想是:定义一种虚拟机(Virtual Machine),以及这种虚拟机所使用的机器码,即 Java 字节码(Java Bytecode,是一种二进制代码)。在任何平台上,Java 源代码被 Java 编译器编译成虚拟机能够识别的字节码。这样,只要是有 Java 虚拟机的平台就能够通过解释的方法执行字节码,从而实现了平台无关性。另外,Java 还采用了基于国际标准的数据类型。在任何平台上,它的数据类型都是一致的。例如,数据类型 int 表示 32 位整数,那么无论是在 PC 机还是 SUN 工作站上,它都表示 32 位整数,而不会像 C 语言那样,随着平台的改变 int 所表示的位数也在改变。

平台无关性所带来的好处是巨大的。一种软件只要在一种平台上开发一次,就可以在所有的平台上通用。那么我们就不必再为 Windows NT 开发其 Alpha 版本或 Macintosh 版本。尤其对于 Internet 来说,更是如此。因为 Internet 上连接数千万台的计算机,这些计算机的种类更是五花八门,要想这些计算机都能执行同一个程序,那么编写该程序的语言就必须具有良好的可移植性,Java 语言就能作到这一点。

当然,Java 语言提高可移植性的代价是效率的降低。因为是一种解释执行的语言,所以在运行 Java 程序时,你就会感到它和 C 程序之间在执行速度上的差别。不过,为尽量弥补效率上的缺陷,Java 语言已经考虑了效率的问题。例如 Java 的字节码在设计上尽量地接近真正的汇编语言机器码,这样在转换成真正的机器指令时可以提高速度。

总的来说,以效率为代价提高可移植性还是值得的,而且随着计算机硬件性能的飞速发展,Java 在效率上的缺陷会很快被人遗忘。

3. 安全性

在许多影片中,我们经常可以看到计算机高手们通过网络轻易地突破重重口令,窃取情报。不论这么做是为了正义的事业还是进行罪恶的勾当,对计算机网络的安全性都是一种威胁。Java 既然是在网络环境中使用的编程语言,那么就必须考虑到程序的安全性问题。Java 对安全性做了详细的考虑,主要有以下几点:

设计时的安全防范:Java 取消了 C 和C++中非常重要的指针(pointer)类型,这样就避免了有意或无意中改变指针,访问程序之外内存空间的事件。提供了数组下标检查机制,禁止了恶意程序对内存的滥用。同时还提供了自动内存管理机制——垃圾收集机制(garbage collection),由系统进行无用内存的回收。

运行时的安全检查:对于由 Java 源程序编译生成的字节码,同样要在运行时进行安全检查。因为在生成字节码到解释执行时,字节码有可能被非法改动,所以在执行之前需要使用字节码校验器进行检查。然后 Java 解释器才决定程序中类的内存布局,这样就防

止了网络“黑客”利用预先知道的信息来破坏系统。随后类装载器把来自网络的类装载到单独的内存区域中,与其他程序不发生关系。最后,还可以在浏览器中限制 Java 程序对文件系统的访问。

4. 简单易学

Java语言之所以简单易学,主要是因为它源自 C 和C++语言。C 和C++是最广泛使用的语言,它们的语法和语义都为大家所熟悉。所以一个学过 C 和C++语言的人可以很快地学会 Java。

当然还不止于此,Java 语言取消了 C 和C++中一些复杂的、不易理解的特点,如多重继承、操作符重载、指针等。这就使 Java 更容易理解也更容易掌握。

5. 分布式计算

分布式包括数据分布和操作分布。数据分布指数据可以分散存放于网络上的不同主机,操作分布则指把计算分散由不同的主机完成,Java 支持 WWW 客户机/服务器计算模式。

对于数据分布,Java 提供了 URL 对象,利用此对象你可以打开并访问网络上的对象,其访问方式与访问本地文件系统几乎完全相同。对于操作分布,Java 的客户机/服务器模式可以把运算从服务器分散到客户一端,提高整个系统的执行效率,避免了瓶颈制约,增加了动态可扩充性。Java 的网络类库是对分布式编程的最好支持。Java 网络类库是支持 TCP/IP 协议的子例程库。

6. 多线程

线程是现代操作系统中提出的新概念。线程,也叫“轻负荷”(lightweight)进程,是比传统进程更小的一种可并发执行的执行单位。

在进程的概念中,每个进程都有自己独立的内存空间和资源,因此当多个进程协同工作时,它们就可能需要进行大量的数据复制和数据交换,这就增加了系统的负担。而在支持线程的操作系统中,可以使多个需要进行通信和使用相同数据的线程共用一块内存,这样做就避免了过多的通信开销,从而减轻了系统的负担。所以线程的概念提高了程序的执行效率,从而提高了系统的效率。Java 语言提供了对多线程的支持。

Java 对多线程的支持体现在:Java 环境本身就是多线程的,它可利用系统的空闲事件来执行诸如必要的垃圾清除和一般性的系统维护等操作。Java 还提供了对多线程的语言级的支持。利用多线程编程接口,编程人员可以很方便地编写出支持多线程的应用程序。

另外要注意的是,从 Java 语言的规范中可以看出,Java 的多线程支持在一定程度上会受到其运行的支撑平台的限制,并且依赖于其他一些与平台相关的特性。例如,如果操作系统本身不支持多线程,则 Java 的多线程可能就会是不完全的多线程。

1.3 Java 与C++的差异

在前面我们提到过,Java 语言与C++有着密切的关系。可以说 Java 语言是从C++

演变而来的。这其中的一个重要原因就是因为C++是使用最广泛，被最大多数程序员所掌握的语言。为了使程序设计人员能够很快的掌握这种编程语言，Java在语法上基本沿用了C++的语法，而且许多关键字也相同。

但是Java语言在语言设计的侧重点上与C++有着根本的差别，这就造成了两者之间的本质上的不同。下面我们从宏观的角度分析一下它们的区别，见表1.1。

表1.1 Java和C++的区别

特　性	Java	C++
应用范围	分布式计算环境，尤其是Internet环境	通用性很强的语言，可用于各种类型的应用
面向对象	是一种100%的面向对象编程语言	既具备了面向对象编程的特性，又保留了对过程式编程语言C的兼容性
安全性	多种安全机制保证了程序执行的安全性	在语言的设计上对安全性的考虑不够
可移植性	实现了二进制字节码的跨平台可执行	理论上有一定的可移植性，实际上由于种类繁多(如Visual C++、Borland C++)，可移植性差
易学性	对于熟悉C语言的程序员来说，很容易学习	许多特性如多重继承、操作符重载、指针均不易掌握
执行效率	解释执行，效率较低	执行效率高
开发环境	相对较为缺乏，但正在不断增多	有众多优秀的开发环境
语言标准	不断发展中	基本成熟

1.3.1 Java取消的C++特性

从上表中，我们可以发现Java在面向对象、可移植性、安全性、易学性等方面与C++相比都有一定的优势，这是因为Java取消了一些C++的特性，并增加了一些新的特性，包括：

1. 取消了#define，#include语句

#define语句虽然在某些时候有用，但是过多的该语句会使得程序的可读性变差，因此Java语言中取消了该语句。而#include语句的工作则由import语句完成。

2. 取消了structure，union及typedef

structure，union和typedef是C++为了与C语言保持兼容性而保留下来的。其实在C++中，完全可以用类来完成它们的功能。所以在Java语言中，就没有它们的“身影”了。

3. 不再有多重继承(Multiple Inheritance)

多重继承,即一个类可以有多个父类,可以同时继承多个父类的特性。它虽然很有用,但是却增加的程序的复杂性。Java使用接口(Interface)作为替代多重继承的机制,一个Java类可以实现多个接口。接口实际上是一种抽象类,它只是设计了实现接口的方法(method)和变量、常量,而没有具体的实现代码。所以,用实现多个接口的方法就比继承多个父类的方法简单。

4. 不再使用函数(function)

Java语言中,取消了结构化语言中很重要的函数,取而代之的是方法(method)。其实方法从结构上看与函数并没有不同,但是所有的方法都是封装在类中的。这就使Java更符合面向对象的标准。

5. 取消了Goto语句

Goto语句既能在很多情况下减少程序的代码量,又会造成程序结构的混乱。通常Goto语句大都是用在循环需要提前结束的情况,C语言中可以使用break或continue跳出单层循环,但是在多重循环的情况下,就需要增加许多代码实现这个功能。Java语言扩大了break和continue的功能,使其可以在多层循环中终止循环流程。

6. 不再有指针

指针是C和C++中最有用的"武器",对于一个C和C++语言的编程高手来说,指针简直就是无所不能的。它给编程带来了极大的灵活性。

但是,有利就有弊。这种灵活性的代价是对安全性的巨大威胁,尤其是对要应用于分布式计算环境的Java语言来说,这是不能容忍的。另外,指针虽然功能强大,但是却不易掌握和学习,对程序员来说也是最容易出错的地方。因为这些原因,Java取消了指针。

7. 不再有操作符重载

操作符重载也就是程序员可以按照自己的习惯,把+,-,×,÷,>,<等运算符号重新定义,完成新的运算操作。例如下面关于复数的操作,可以使复数的运算看起来更符合习惯。

```
class Complex
{
  private:
  float m_fRealNum; //复数的实部
  float m_fImageNum; //复数的虚部
  public:
  Complex(floate real, float image) //构造函数
  {
    m_fRealNum = real;
    m_fImageNum = image;
  }
```

```
    Complex operator×（Complex&c)
    {
      return Complex(c.m_fRealNum×m_fRealNum-c.m_fImageNum×m_fImageNum,c.m_fRealNum×
      m_fImageNum+c.m_fImageNum×m_fRealNum);
    }
    ....
}
```

上面的例子实现的是复数类的乘法操作，通过操作符重载使复数的乘法就像整数乘法一样简单明了，从这一点来说操作符重载是很有用的。

但是，由于这一点也使程序员可以任意地自定义自己的运算符，使程序的可读性变差。而且操作符重载不是必须的，我们完全可以使用其他的办法实现同样的功能。总之，Java语言取消的操作符重载，使Java语言少了一项功能，同时也使它更简单易学。

8. 取消自动类型转换

在C++中，允许类型的自动转换；而Java是强类型检查的编程语言，在进行类型转换时必须明确地指出。

例如下面的程序：

```
long l;
double d=3.1415926;
l = d;
```

在C++中编译该段程序，编译器可能会给出警告信息，但是不会报错。用Java编译器编译它，则会报错，因为把double型变量的值赋给long型变量，必须如下进行强制类型转换：

```
l = (long)d;
```

强制类型转换使编译器的检查更严格，这样可以防止一些人为的程序错误。

1.3.2 Java语言增加的特性

与C++相比，Java语言在内存管理上更加严格。在Java语言中，无用内存的“回收”是通过一个名为“垃圾收集”（垃圾指的是无用内存）的机制完成的。其特点是由系统来完成无用空间的回收。这种机制与C++由程序员通过指针完成回收有何不同呢？

让我们先了解两种垃圾回收策略：一种是程序员负责回收，另一种是系统负责回收。使用第一种方法，程序员请求并分配到的内存由程序员管理和使用，使用完成后交还给系统（也就是通常所说的释放内存）。C++，C都是用这种方法。分配给程序员的内存一般是由一个指针变量指定的，也就是说程序员是用指针管理内存的。一旦该指针的使用发生了问题（这种情况经常会发生），如指针被错误地赋值或程序员忘记了释放，那么这块内存也就无法回收了。所以这种方法对程序员要求很高，而且程序中的错误也经常出现在这个方面。

第二种方法是由系统负责不定期地检查内存使用情况，发现不用的内存后，自动收回。程序员管理内存的工具不是指针，而是类似指针的引用（reference）。当发现某块内存没有任何引用指向它，就意味着程序没有使用它，那么系统将自动回收这块内存。这种方法减轻了程序员的负担。

1.4 小 结

本章简单地介绍了近来非常流行的Java语言。Java语言是这几年才发展起来的编程语言。作为一种编程语言,它能够在很短的时间内流行起来,固然有一些商业的原因,但是不可否认Java本身技术上的优势和对“火爆的”Internet的迎合是其为大家亲睐的关键因素。

在这一章中,我们从Java语言产生的背景分析了Java语言的特点:面向对象、平台无关性、安全性等等。之后我们把Java与最流行的面向对象编程语言C++作了比较,通过比较了解了Java在某些方面所具备的优点,为进一步学习Java做好准备。