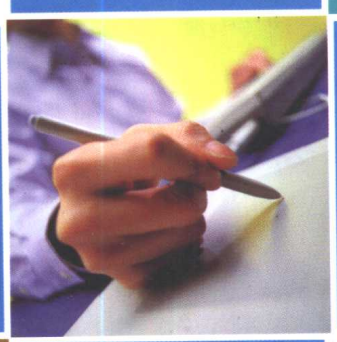
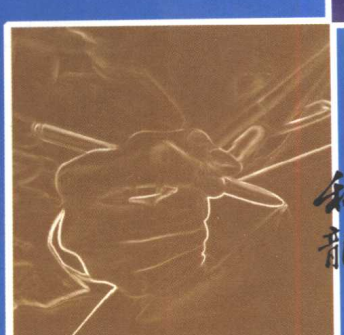
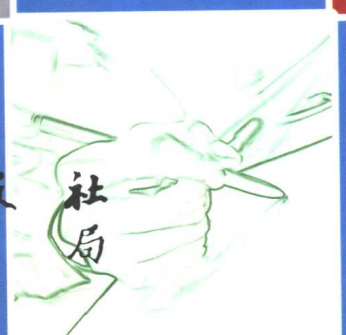
Java 编程手册

易文韬　陈颖平　编著
李权燕　改编



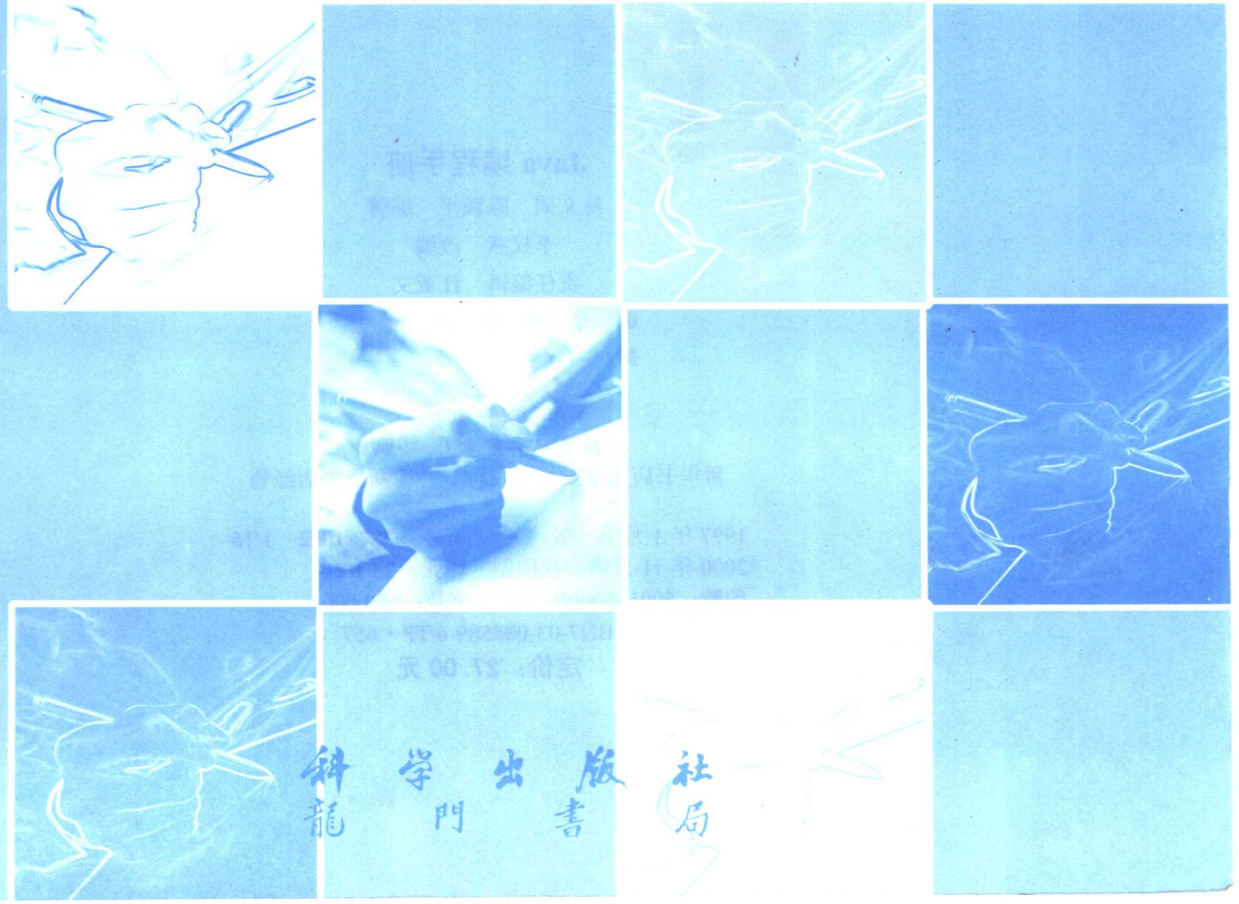


旗標
FLAG
科學出版社
龍門書局

Java 编程手册

易文韬　陈颖平　编著
李权燕　改编



科学出版社
龍門書局

（京）新登字 092 号

内 容 简 介

本书全面地介绍了 Java 的起源、影响和语言特性，以及基本概念和 Internet 上的各种资源。

本书通过大量实例论述了 Java 程序语言的语法和应用，全书共二十四章，内容分为四个部分：Java 介绍；Java 语言基础；如何在自己的 HomePage 上加入动画、影像、声音等小应用程序，配合上 AWT 和 Java 的网络功能，使 HomePage 不但可以生成多个窗口，也可以与 WWW Server 之间随时沟通信息；以及有关 Java 语言的更深层的知识。

本书适合于计算机用户、大专院校师生，以及所有对 Java 编程语言感兴趣的读者阅读参考。

需要购买本书或得到有关本书技术支持的读者可直接与中关村 083 信箱北京希望电子出版社联系，或电话 010-62562329，62531267，传真 010-62579874，邮政编码 100080。

版 权 声 明

Java 编程手册

易文韬　陈颖平　编著

李权燕　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

*

1997 年 1 月第一版　　开本：787×1092　1/16
2000 年 11 月第二次印刷　印张：24 1/2
印数：5001~10000　　字数：569 000

ISBN7-03-005589-6/TP・657

定价：27.00 元

目　录

第一篇

认识Java

第一章　全球信息网(WWW)及 Java

1.1　WWW 的发展史及对未来的影响

几年前曾经有人估计,认为世界上卖出的计算机已经达到了饱和点,今后可以卖出的硬件,大概就是旧计算机升级罢了!现在看来,这个预言显然是个笑话,但是为什么会有这样的说法呢?最大的理由就是,这个世界上该有计算机的人都有了……

虽然 Internet 走红是这一两年间的事,但 Internet 早已存在许多年了。在全球信息网(WWW:World-Wide Web)这个名词出现前,许许多多的计算机用户就通过世界上最大的网络——Internet 在世界各地浏览。只不过,用户多半只是用 FTP 等工具来传输文件,或者发送 e-mail 罢了。对于那些不熟悉计算机的人来说,即使网络上的资源再丰富,想到要学会用一些奇奇怪怪的传输协议来查找数据,就很容易因望而生畏而裹足不前了。

虽然当时的 Internet 资源已经很丰富,但再怎么说也不是一个对用户友好的网络环境。于是,出现了第一个综合性的工具,明尼苏达大学的研究人员创造出了 Gopher。使用 Gopher 并不比 WWW 复杂,在一个文字页的菜单上,用户只要在想要的标题上按下鼠标按钮或键盘上的键,即可连上你想要的资源,取得文件或者看到文件,也可能直接连到远地的某一台机器上。

Gopher 的出现确实让 Internet 的普及化向前迈进了一大步,但也因为 Gopher 的设计太简单了,很快人们就受不了它的一些限制。首先,Gopher 的文字页菜单设计,使得存放文件的人只能以很短的文字来当作全篇文件的说明,并且用作菜单中的选项。另外,纯文字化的环境也否定了声音、图片等多媒体出现的可能。

为了让 Internet 上出现可以直接显示的多媒体文件,1989 年,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 的全新网络传输协议。除了保持 Gopher 原有的多种优点外,还提供了更新更强的功能。首先,用于连接的选项不再是单独存在,而是伴随整份文档一同出现,这就是你已经很熟悉的起始页上的链了。接着,你也可以很自然地在同一份文档上加入图片和声音,再配合一个好的 WWW 浏览器,多媒体文档就这样在网络上散播出来。

原本这样的传输协议和文档只在少数学术单位和研究人员之间流传,直到 1993 年伊利诺依大学的编程人员发表了 Mosaic——一个容易使用的 WWW 浏览器,WWW 才开始正式传播在整个 Internet 里。如今,对很多人来说,Internet 就等于是 WWW。

笔者不知道当初创造 WWW 的人是否曾预料到今天 WWW 和 Internet 的浪潮,但是 WWW 给这个世界最大的贡献就是让许许多多原来对计算机完全不懂的人,能够很方便地享受计算机所带来的便利。使用计算机的人愈来愈多,计算机也愈来愈像家用电器,世界上可以卖出的计算机,离饱和点恐怕是愈来愈远了。

1.2 Java 是什么

如果你原本就是一个常上国际网络，使用 WWW 浏览器，没事在网络上东晃晃西晃晃，到处看各地起始页的人，那么你可能会发现，最近的起始页似乎不太一样了。有时候有些起始页冒出了一个黑帽红鼻子的小人不停地向你挥手，又有些时候某些起始页会不经意地冒出一些声音来吓你，或者原来的起始页上静态的字词一时之间全都动起来了。是的，这正是 Java。

如果你是一个网络迷，可能早就在问："为什么这些起始页不一样了，为什么常看到'Java Support'的字眼出现，Java 到底是什么呢？"如果你是一个公司的经营者，你可能又会问，Java 到底有什么魔力，能获得包括 Netscape、Microsoft、Oracle 等多家厂商的支持，而 Sun 公司的股票也在几个月内涨到原来的三倍呢？

大家都在谈 Java，就连 TIME 杂志都将它列为 1995 年的十大科技产品之一。要知道在 1995 年时，它还是个测试版(Beta 2)。那么，到底什么是 Java 呢？

简单地说，Java 是一个由 Sun 公司开发而成的新一代编程语言。Java 的目标是为满足在一个充满各式各样不同种机器、不同种操作平台的网络环境中开发软件。这也正是为什么不论你使用的是哪一种 WWW 浏览器，哪一种计算机，哪一种操作系统；只要 WWW 浏览器上面注明了"支持 Java"，你就可以看到生动的起始页的原因。

利用 Java 编程语言，你可以在你的起始页中加入各式各样的动态效果，你可以放上一段动画，可以在起始页上建立霓虹灯式的看板，让你的名字在上面不停地转动。如果你愿意，就像一般的窗口程序一样，你还可以放上菜单和按钮，以及滚动条。只要使用 Java，没有什么你做不到的。

为什么 Java 可以做到这些传统起始页上做不到的动态演示呢？没有别的原因，在加入 Java 的支持后，你的起始页已不再只是一个冷冰冰的"文件"，而是配合 Java 程序的一个活生生的程序实体。正因为这一点，在你使用 Netscape Navigator 或者 Microsoft Internet Explorer 观看各地起始页时，你所用的 WWW 浏览器不但要负责将 HTML 格式的文件以正确的格式显示出来，同时也必须负责在你所使用的机器上，执行伴随而来的 Java 程序。

事实上，对这样一种伴随起始页而来的 Java 程序，我们取了一种特别的名称，叫 Applet，我们可以把它想像成是一个个小巧可爱的程序。Java 除了开发附在起始页上的小应用程序外，也具备有开发大型应用程序(Application)的能力，并且同样可以跨越不同种类的机器、不同种类的操作平台的限制，在各地执行。

1.3 为什么是 Java 呢

看了前一节对 Java 的简单说明，我想你一定会问，为什么是 Java，难道其他的编程语言办不到吗？我心爱的 C 及 C++语言不行吗？Assemble 不行吗？方便好用的 Visual Basic 或者 Delphi 不行吗？还有 Smalltalk、Lisp、Prolog、Ada……等种种编程语言或软件开发工具，它们都做不到 Java 能做的事吗？

在讨论这个问题之前，我们再回头看一下 Java 设计的目标和思路。Java 所想要适应的

程序设计环境,是一个充满各式各样不同种类的机器和软件的平台,并且用网络连接的复杂分散式环境,Java 的程序要能够通过网络在不同地方的不同机器上执行。

也就是说,Java 程序可能由你在家中的 PC 通过调制解调器连上国际网络后,从世界的某一角接收过来,在你 PC 上的 Windows 95 中执行;也可能会在公司中的高级工作站配合上 Sun 的 SunOS、Solaris 或者 IBM 的 AIX 来执行。写 Java 程序的人完全不需要去考虑他写出来的程序会如何被执行。

仔细想想,这其实是一个很大的挑战。可移植性(portability)一直是软件界一个大问题,为了在各种不同的平台上执行同一个程序,即使你的程序写得再好,你至少还要在各个平台上重做一次编译,而只要你写程序时稍微不小心,很可能在移植的过程中,就必须为许许多多当初的不小心付出大量的调试时间和精力。

别的不说,光是一个 16 位程序要移到 32 位的操作系统下执行,对 C 或 C++里的 int 整数类型就必须特别加以注意,因为在一般的 16 位操作系统中,int 所占用的空间是 2 个字节,而在 32 位操作系统中则占用 4 个字节。

如果你创建了一个属于你自己的起始页,并且放上能动态演示的程序,如果要先准备好在十来种操作系统下编译好的执行码,再由看你的起始页的用户选择不同的工作平台,那么想必这对你或者用户而言都是一件极其痛苦的事。这是 Java 成为网络上使用的编程语言的第一原因。

但这也同时带来了另一个严重问题,即虽然这样的网络语言使你在浏览各种起始页时增加了许多乐趣,但这同时意味着,只要你一使用 WWW 浏览器游览网络世界,一个个世界各地的程序就跳到你的机器上执行,请注意是"你的机器"。我怎么知道网络另一端的陌生人,附在他的 WWW 起始页上的程序是纯粹的动画程序,还是偷偷地埋了一个病毒。换句话说,这样的远端分布式执行环境,令人最重视并且最不放心就是所谓的安全问题(security)。

也正因为如此,自从 Java 出现以后,几乎每一个听到的人都会问,看 Java 的起始页真的安全吗?有没有可能用 Java 做出计算机病毒呢?很显然,要达到 Sun 公司开发 Java 时所定的目标,就不能不考虑这个问题,这也是当初开发 Java 语言时注意的一个重点。Java 的源代码是公开的,很幸运,一直到目前为止,还没有人发现那一段程序代码有真正的漏洞。也就是说,Java 到现在为止,都如同 Sun 公司所宣称的,是一个安全的网络编程语言。

其他编程语言或者软件开发工具办不到吗?我想答案是很明显的,如果想要达到上两个目标而成为网络上通用的编程语言,一定要对编程语言或开发工具本身做必要的修改。而这也正是目前 Microsoft 对 Visual Basic 的态度,希望新一代的 Visual Basic(VB Script)能取代 Java 成为大家使用的网络编程语言。另外,诸如由 Tcl 编程语言所衍生的 Safe-Tcl 语言,以及 General Magic 的 Telescript,还有由 Netscape 和 Sun 联合推出的 JavaScript 都具有类似的功能,不过从目前的市场趋势来看,Java 似乎已成为最大的赢家。

1.4 软件革命——谈 Java 对未来软件界的冲击

记得在 Microsoft Windows 95 推出不久以后,有一期 TIME 的封面人物是 Bill Gates,上面写着"All is mine"。是的,相信没有人会否认这句话,Microsoft 在这二十年中迅速成长,Bill Gates 高瞻远瞩,没有人敢不买 Microsoft 的帐。可是 Microsoft 一点也不庞大啊!为

什么 Microsoft 能够有这么大的影响力呢？Bill Gates 独到的眼光和卓越的领导固然是主要原因，但 Microsoft 控制了大部分 PC 的工作平台，似乎也是它无坚不摧的有力武器。

因为几乎所有的软件都会支持 Microsoft 操作系统，而正如前面所说的，软件的移植并不是一件容易的事，其他操作系统上的应用程序相对来说也就比较少。也正因为如此，所有的应用程序由于必须在 Microsoft 的操作系统上运行，也就不得不受 Microsoft 的牵制，跟随 Microsoft 的脚步。因为操作环境的排他性太强，如果今天 Microsoft 推出一个比较差的操作系统，只要不出太大问题，应用程序的开发厂商恐怕还是得默默地接受。

也就是说，在这一点上，商业方面的影响力要比技术方面的影响力大得多。而对于用户来说，最沉重的负担还不是这个。如果你问一下周围的朋友，他们平常用得最多的软件是什么，相信十之八九会说 Word、Excel、Amipro 等所谓的 Office 软件，而对于大多数非计算机爱好者的用户来说，这样的软件只要学会一套，能够用来处理日常的事务，其实也就够了，所以学会 Word 的人大概就一直追随 Word 的脚步，而不会哪天突发奇想去看看 Amipro 怎么用。

而随着 Microsoft 不断地更新其软件，用户也就跟着 5.0、6.0、7.0 不断地升级，不断地买。更糟的还在后面，Office Suite 的软件套装销售方式，似乎也成了另一种趋势，一起买价钱看起来似乎更划算，买吧买吧！通通一起买下来吧！

你曾想过整套软件都是你会用到的东西吗？如果你是一般的上班族，回想一下，你曾经用 Microsoft Equation 编写过程序吗？一套 Office 中，到底有多少东西是你每天要用的？然而，大部分的人在安装软件时，总是不敢不全部安装，因为怕哪一天突然要用到某些特殊的功能，而又要重新做一次安装工作。

用户似乎每两三年就要大幅度更新他的计算机设备，每一次 Intel 推出更强大的 CPU，Microsoft 就立刻推出新一版的软件，将系统的资源用光，而卖硬盘的厂商总是笑嘻嘻地看你向他报到，因为现在的软件愈来愈大，以前 300MB 的硬盘绰绰有余，现在 1GB 的硬盘似乎才是标准的配置。就这样，软件、硬件的厂商联手出击，赚钱的速度似乎永远跟不上升级的脚步。

仔细想想，这并不是一件合理的事，每次去添购高级的设备，只是为了装一些很少用到的程序？而每一次使用新软件，都要安装一次，而安装软件并不是每次都很简单，常常需要调整一些参数，虽然我相信这样的工作对读者是轻车熟路，但是对那些完全没碰过计算机的人来说，可不是一件轻松的事。

由于 Internet 的风行和 Java 的出现，这样的僵局很有可能被打破。想想看，如果今天你想用一个电子表格软件来记记帐，做个小统计，你只需要打开你的计算机，接上 Internet，利用你习惯的 WWW 浏览器，连到某一家做电子表格的软件公司，用鼠标点出你想要的软件，接着你就可以在 WWW 浏览器上使用你选的软件，不需要安装，不需要准备大量的硬盘空间，使用完了关掉计算机即可，一切都不再存在，而只剩下数据。

是的，只有数据。我们使用计算机不就是为了要得到我们要的数据或信息吗？何必去储存处理程序呢？而且通过这样的方式，用户随时都可以使用最新的程序，不需要等到软件公司更新了好几个部分，推出新一版的应用程序以后，才能使用。

而软件可能也不再需要用户到店里花钱去买，取而代之的方式是通过网络向软件公司租用，或者按照使用的次数来计费。Internet 打通了程序数据等的流通渠道，Java 则利用方

便的 WWW,实现了远端执行的理想。

去年年底,Sun 公司的副总裁颜维伦博士在台湾大学应用力学馆发表了一场演说,回答了这样一个问题:"Sun 公司会在 Java 上得到什么好处?"他的答案很妙:"Sun 的目的只是想要打破目前软件市场被垄断的情况。"姑且不论将来软件界会不会真用上面预期的方式来经营,但几年以后的软件界肯定和现在有很大的差异,因为在 Internet 和 Java 下,什么事都可能发生!

第二章　Java 的发展史

Java 无疑已在市场上打下了非常好的基础，从商业的角度来看，Java 是一项成功而杰出的产品。大家可能会认为，Java 的背后必定有一套完整而目标明确的开发计划，来支持 Java 的开发工作。但是令人意外的是，开始的时候，Java 非但名称不叫 Java，而且就连最终的目的也不明确。虽然如此，整个小组的开发计划依然有个固定的大方向，经过五年左右的摸索和努力，终于有了今天的成功。

2.1　Java 的开始

Java 的产生最早可以追溯到 1991 年，当时 WWW 还没有正式出现在这个世界上。据 Java 的开发者之一 James Gosling 表示，当初 Sun 公司的这个开发小组的开发目标，只是希望能将 Sun 公司从传统起家的工作站市场，进一步扩展到消费性电子产品市场。而所谓消费性电子产品，是指我们在市场上所看到的，功能在一般计算机之下的电子产品，像个人数字助手(PDA)、电子翻译器、电视游戏机，以及交互式有线电视的电视控制盒等等。

如果我们拿 Sun 公司独霸一方的工作站市场，以及所擅长的微处理器技术和消费性电子产品做个比较的话，我们很容易就会发现购买这两类商品的顾客，他们的需求有非常大的差别。购买和使用 Sun 工作站的客户多半是学术单位，或者是工程师，他们非常在乎买进来的机器是否在运算速度上能够满足工作上的需要，对于机器本身相对高昂的售价则不是那么在意。而因为使用这些机器的人通常也是学有所长的计算机专家，机器是否操作简单并不是很重要，而公司也比较能忍受培训的高额花费，以及系统偶然出现的错误。因为高级机器如果运用得当，回收的价值是足以让人花钱去投资的。

消费性电子产品的需求则完全不同。消费性电子产品的顾客是一般消费大众，产品的价格绝对不能太高，同时产品的功能愈简单、愈容易操作愈好，用户并不是很在乎电子产品内部所使用的是什么 CPU，或者 CPU 的速度有多快，在意的重点是产品买来是否可以正常的运行，因为对非计算机专家的用户而言，机器一旦出错，就完全没有维修能力。

开发小组的方向是希望能够建立分布式的系统结构，同时将软件上的各种新技术移植到消费性电子产品上。

2.2　Java 曾经尝试过的应用局面

为了这个方向，在 1991 年 4 月 8 日，成立了“green”小组，正式展开整个研制计划。为了达到与软件平台无关的特性，以及让大部分的编程人员都能很快熟悉，Gosling 一开始试着从 C++入手，增加 C++编译器的功能，然而在经过一段时间的努力之后，由于 C++离 Green 小组的目标差距太大，最后只好自己重新定义一套全新的语言和系统。于是 Java 的前身——Oak 出现了。

为了能将整个系统作一次综合性的演示，Green 小组开始了“*7”计划，除了建立 Oak

语言的开发工具外，还有 Green OS，用户界面以及硬件部分，最后的目的是做出与现在的 PDA 类似的电子产品。

最后在向 Sun 公司内部演示产品时，由于 * 7 成功地表现了高效率的小程序代码技术，引起 Sun 公司高层主管的兴趣。但即使如此，如何将这项技术正式移植到合适的商业产品中，仍然是个大问题。

1992 年 10 月 1 日，Green 小组正式升级成 First Person 公司。与此同时，First Person 得知 Time-Warner 有线电视公司打算开发交互视式电视，认为 Java 非常适合应用到这个领域，于是全力投入了这项技术的开发。很可惜，最后因为商业上的某些原因，并未获得 Time-Warner 公司的青睐，Java 没有运用到交互式有线电视的电视控制盒上。

2.3 进入 WWW 的世界

大约在 1994 年年中，全球信息网(WWW)的势头在 Internet 上愈来愈猛，在积累了过去的开发经验，以及检查整个 WWW 的结构以后，开发小组产生了新的想法，过去 Java 的主要目标——与平台无关，系统的可靠性、安全性等，都非常适合 WWW 世界，而且与其他浏览器不一样的是，利用 Java 做出来的 WWW 浏览器，可以做到一般浏览器做不到的功能。于是第一个可以在 WWW 上执行 Java 程序的 WWW 浏览器出现了，这个 WWW 浏览器一开始被命名为 Web Runner，也就是后来的 HotJava。

Java 和 HotJava 的出现改变了全世界人对 WWW 的看法。在 Java 出现前，人们对 WWW 上所传输的数据，只不过将它们想成是一页页的文档，而这份文档除了文字以外，还可以放上一些图片等多种数据，同时 WWW 浏览器则不过是一个看文档的工具程序罢了，但是在 Java 出现以后，WWW 所传输的数据由单纯的文件摇身一变成为一个个可执行的程序，并且 WWW 浏览器也成为执行这些程序的系统。

在经过 Sun 公司内部一系列的评估之后，在 1995 年 5 月 23 日，Sun 公司终于正式发表了 Java 和 HotJava 这两项产品，虽然当时出现的版本只是 Alpha 1 测试版，但很快就引起大家的注意。几个月后，当 Netscape 和 Sun 公司合作，在 Netscape Navigator WWW 浏览器中支持 Java 后，Java 在 WWW 上可说是站稳了脚跟。在得到 Netscpae 的支持以后，或许是达到了演示的效果，HotJava 这个 WWW 浏览器一直停留在 Alpha 3 版本上，而 Netscpae Navigator WWW 浏览器则不断地改版支持 Java Beta 1、Java Beta 2，现在 Java1.0 已完全定形，而 Netscape Navigator 2.0 也在 1996 年 2 月结束 Beta 测试，正式推出。

第三章　Java 语言的特点

3.1　面向对象

如果说 80 年代是 AI 的时代，那么 90 年代无疑地就是 OO 的年代了。计算机界常有一股接一股的浪潮，在这股浪潮中，开口闭口不忘提到面向对象(object-oriented)似乎不会错，但是你是否会回头想想面向对象到底是什么，是否怀疑过计算机界目前流行的所谓面向对象的编程语言或工具，真的能达到面向对象理论中所论述的神奇的效果吗？

面向对象的特点

和目前很多的软件开发工具一样，Java 的第一个特点就是它是一套面向对象的编程语言。但是如果我们以比较严谨的态度来看面向对象，那么一套面向对象的工具至少应该包括下面的四个特点：

□ 封装性(Encapsulation)：

必须有模块化(modularity)的性质以及信息隐藏(information hiding)的能力。

□ 多态性(Polymorphism)：

不同的对象对同一种信息，可以按照对象本身的性质加以回应。

□ 继承性(Inheritance)：

可以定义一套对象之间的层次关系，下层的对象继承了上层对象的特性，藉此可以实现程序代码重复利用，并且有效地组织整个程序。

□ 动态联编(Dynamic binding)：

一个对象一旦生成以后，要使用这个对象只需简单地把信息传递给它，不再需要去参考对象当初设计时的规格。只有在程序执行时，才会真正锁定需要的对象，这样的方式可以使程序设计具有最大的灵活性。

事实上，如果以上面四个特点去衡量现有的一些编程语言或工具，有很多都不能算是完全的面向对象。如 Visual Basic 缺乏数据的封装性，而 C++并没有办法做到动态联编。但是 Java 在这四点上都可以做得很好。

对象、类、信息和方法

面向对象的程序设计思路，就是希望能将生活中自然的概念，应用到程序设计上。现实生活中，什么是一个对象呢？是的，对象是你眼前所看到的任何物体，可能是一部计算机，可能是一部车，也可能是你手上的这本书。对于一个对象，它们都有两个基本的特点，一个是物体内部的状态，譬如说，车子现在是“停止的”，计算机现在是“开着的”等等；而另外一个特点则是对象对于外界给予的信息或操作方式，能以一套自己的方法来处理，譬如说，转了车子

上的钥匙，车子就会发动等等。面向对象的方法就是给程序中现实的对象定出模型。

一个程序中的对象具有自己的内部数据，可以把这些数据想像成现实对象内的状态，状态是不能随便改变的，必须由外界向对象传递信息，再由对象按照既定的方法加以改变后才会变成新的状态。同样的，对于一个程序对象，我们如果想要改变其内部的数据，也必须将外界的指令信息，传给这个程序对象原来定义的方法，由对象本身的方法来修正内部的数据，并给予外界适当的反应。

通过这种方式，我们就已经完成了上一小节中，面向对象程序设计的第一个特点——封装性。而对于多态性这个特点，只要在不同的对象中对相同的外界信息，设计出不同的处理方法，也就可以达到。

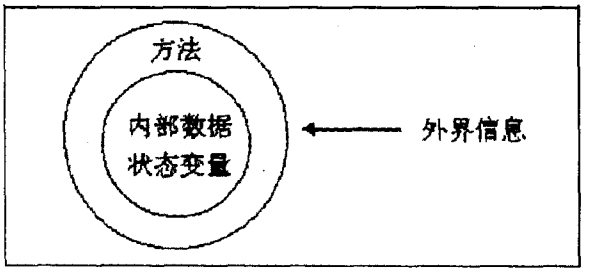


图 3-1 程序对象示意图

可以把类(class)想像成是对象的基本原型，同一种类的对象有着同样的方法和性质，譬如说，你可以定义一个“汽车”的类，这个类中定义一车子必须有颜色、有车门等，对于踩油门的反应都是增快车子的速度，那么，由这个“汽车”的类原型，就可以生产出许多汽车的对象实体(instance)。这些汽车的对象实体都有自己的颜色(虽然这些颜色可能都不一样)，也都有自己的车门(3 门、4 门不一定)，但是这些性质都存在，更不用说对于踩油门踏板的反应了。

事实上，在我们编写程序时，实际定义的都是一个个的类，而对象的实体则是在程序执行时，才由系统一一产生。

Java 语言对类的定义方式和 C++的方式基本上是相同的，我们看下面的一小段例子即可得知：

程序 3-1

```
// Java 的类实例

class Car {
    int color_number;      //车的颜色编号
    int door_number;       //有多少车门
    int speed;             //车速
    ...
    push_break (){         //踩煞车的反应
      ...
    }
    Add_oil() {            //加油的反应
```

```
        ...
    }
}
```

类的另一个最大作用，就是它提供了“继承性”的实现方法。在面向对象程序设计中，所谓继承，是指被继承的类中，原定义的性质和方法，都直接被后来的类所承接，同时后来的类还可以在原类中定义的性质和方法之外，加入自己新的定义。就拿一上段中的“汽车”类来做说明，我们可以定义出一种新的类叫“垃圾车”，“垃圾车”类直接继承自“汽车”类，所以原来“汽车”类中已经定义过的性质和对外界信息的处理方法，在“垃圾车”类上同样有效。也就是垃圾车也有颜色，也有车门，踩了油门一样会跑。但是垃圾车的特殊用途——装垃圾，则必须再独立定义。

在这里我们就可以看出继承的最大好处——程序代码的复用。原先我们对于“汽车”类里写过的定义，写过的程序，在“垃圾车”类中都不需要再次重写。另外一个好处则是可以帮助编程人员在写程序时，很自然地运用面向对象的概念去实现所要处理的数据，使程序和数据的组织更加完善。

程序 3-2

```
// Java 的类继承实例
class Trash_Car extends Car {
    double amount;          //垃圾数量

    fill_trash(){           //装垃圾的方法
       ...
    }
}
```

3.2 操作平台无关性

如果要让我们的程序可以在各种不同的机器及操作系统上执行，那么首先在编写程序源代码时，必须让自己的程序不使用编程语言中由编译器或平台本身定义的功能，譬如说在 C 语言中的 int 整数类型并没有实际定义它的长度是 2 个字节或 4 个字节，必须由在那种机器或操作系统下执行才能决定。如果我们在使用时，只把它想成是 2 个字节长度的数据空间，那么当这个程序移植到别的平台时，就很难保证不会发生问题。因此，编程语言严格的定义，是确保程序可以在各种平台上工作的第一步。在 Java 的语言定义，我们看不到任何取决于工作平台或编译器的功能或特性。

另外一个最大的问题是，各个机器都有不同的低级汇编语言，在这个机器上编译好的机器码，肯定不能直接拿到其他机器上使用，即使在同一种机器上的机器码(binary code)，由于与程序的执行过程和操作系统信息相关，只要是不同的操作系统，程序编译好的机器码往往也有很大的差异，而不能互用，而必须重做一次编译工作。

譬如说在同一台 PC 下，Linux 操作系统的程序机器码，并不能直接给 DOS 使用。为了解决这个难题，Java 的策略是采取半编译、半解释的方式，定义出 Java 自己的虚拟机