CX—2093

Java 基础教程

郑晓军 严望佳 朱嘉琳 编著

Java

基础教程

郑晓军
严望佳 编著
朱嘉琳

科学出版社
龍門書局

Java 基础教程

郑晓军
严望佳　朱嘉琳　编

燕卫华　审订

科学出版社
龍門書局
1997

(京)新登字 092 号

内 容 简 介

Java 是 Sun 公司开发的、不依赖于具体软硬件平台的、面向对象的编程语言环境，因此，很快便在 Internet 网络用户中流行开来。作为一本入门性的书籍，本书用浅显的语言说明了 Java 语言的特点，解释了 Java 语言所涉及的许多概念，介绍了 Java 语言的元素，具体包括对象、类、接口、包、字符串及字符串缓冲区类、程序属性、系统资源的使用、线程控制、异常处理和输入输出。另外，本书还用实例说明了 Java 的编程方法，并简单介绍了什么是 HotJava。

本书可供 Internet 用户和广大 Java 爱好者参考，尤其适于作为 Java 培训班的教材或自学教材。

欲购本书或技术咨询的用户，请直接与北京 010-62562329，010-62541992 或传真 010-62579874 联系。

Java 基础教程

郑晓军 朱嘉琳 编
严望佳

燕卫华 审订

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

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

1997 年 1 月第一次印刷 印张：6 3/8

印数：1～15000 字数：138 000

ISBN7-03-005613-2/TP·668

定价：10.00 元

序　言

简单、面向对象、分布式、解释性、高可靠、安全、结构中立、可移植、高性能、多线程和动态性，这就是 Java！难怪几乎每个 Internet 用户都在津津乐道地谈论着 Java，有这么多的优点，目前又没有其他任何语言能同时具备上述所有特征，它自然会广为流行。

为了满足目前国内广大用户掌握 Java 语言编程、访问 Internet 网上 Java 资源的需要，在近期将推出以下从不同侧面介绍 Java 语言的书籍：

●《Java 基础教程》

从初学者的角度出发，介绍 Java 语言的基本概念、特点和组成元素。

●《Java 使用手册》

是一本全面介绍 Java 语言的书，其中含有丰富的程序设计范例。

●《Java 资源图解》

全面列出 Internet 网上的 Java 资源。

作为一本入门性的书籍，本书用浅显的语言说明了 Java 语言的特点，解释了 Java 语言所涉及的许多概念，介绍了 Java 语言的元素，具体包括对象、类、接口、包、字符串及字符串缓冲区类、程序属性、系统资源的使用、线程控制、异常处理和输入输出，另外，还用实例说明了 Java 的编程方法，并简单介绍了什么是 HotJava。

本书可供 Internet 用户和广大 Java 爱好者参考，尤其适于作为 Java 培训班的教材或自学教材。

希望图书创作室

1996 年 6 月

目　录

第一章 概 述

Java 程序设计语言和环境是为了解决在 Internet 软件开发过程中所遇到的问题而设计的。最初，在 Sun 公司的一个电子购物项目中，人们需要在一些专用设备(如顶盒)中写入程序。这是一种小巧、可靠、可移植、分布式、实时的嵌入式系统。在项目开始时，工程师们首先想到的是 C++。但是人们马上就遇到了一系列问题，这些问题开始的时候只是一些编译技术方面的问题，但是随着时间的推移，所发现的问题之多使人们最终认为设计一种新的语言是最好的答案。于是，Java 诞生了。

Java 是一种简单的、面向对象、分布式、解释性、高可靠、安全、结构中立、可移植、高性能、多线程和动态性的语言。

罗列众多的定语是描述一个系统的拙劣的方法，我们不得已才用它来描述 Java。下面，我们将说明其中的每一种特点以及所能解决的问题。

1.1 简 单 性

我们需要建立一种编程简单的系统，无需专深的培训，却可利用现有标准的成果。目前，大多数软件开发人员使用的是 C 语言，同时绝大多数的面向对象程序设计使用的是 C++。因此，尽管我们发现 C++有许多地方不适合，但我们还是尽可能地将 Java 设计得与 C++相近，使之更容易理解和接受。

Java 舍弃了许多 C++中很少使用的、难于理解的、容易混淆的特性。在我们的经验中，它们所带来的麻烦大于好处。这些特性主要包括：操作符重载(虽然 Java 也支持方法的重载)、多重继承和自动强制。

在 Java 中加入了自动的空间回收处理，这使编程变得容易，但系统在某种程度上变得略微复杂。在许多 C 和 C++的应用程序中，复杂性的来源通常是存储管理，即申请和释放内存。Java 中自动的空间回收能力所带来的好处不仅是使得编程更加容易，而且显著地减少了错误。简单性的另一个方面是小巧。Java 的一个目标是使软件独立地运行在小机器上。基本的解释器和类支持模块的大小是 40K 字节左右，外加基本的标准库和线程支持模块(主要是一个自包含的微核心)约 175K 字节。

1.2 面 向 对 象

很不幸，这是一个被滥用的字眼。但是，面向对象的设计是强有力的，它清晰地定义了数据结构及相应的接口，并且使提供可重用的“软件集成块”成为可能。

简单地说，面向对象设计是一门技术，主要用于数据(对象)及其接口的设计。我们用木工做一个比方：一个“面向对象”的木匠首先关心的是他所制造的椅子，其次才是制造椅子所用的工具；而一个“非面向对象”的木匠首先考虑的则是他的工具。面向对象的设计也是定义

模块如何“即插即用”的机制。

1.3 分 布 式

Java 有一个扩展的例程库，可方便地采用 TCP/IP 协议，并像 HTTP 和 FTP 那样拷贝数据。Java 应用程序可使程序员如同访问本地文件那样方便地通过 URL 打开和访问网络上的远程对象。

1.4 可 靠 性

Java 通过各种方法保证其程序的高度可靠性。Java 将许多重点放在对可能发生问题的早期检查、后期(运行时)动态检查和减少可能发生的错误等方面。

强类型语言(如 C++)的一个优点是它允许编译时扩展检查，因此，错误可以被较早地发现。不幸的是，C++继承了从 C 语言编译检查带来的不严格的漏洞(尤其在方法/过程的说明部分)，Java 要求所有方法和过程必须声明，同时不支持 C 风格的隐含声明。

Java 和 C/C++的一个最大区别是，Java 拥有一个能够避免内存覆盖和数据冲突的指针模型。Java 用一个真正的数组类型取代了算术指针，这使得下标检查能够执行；另外，它不允许将一个任意的整数转变为一个指针值解决。

虽然 Java 并没有将 QA 问题全部消除，但使得这些问题变得十分容易。

非常动态的语言，如 Lisp，TCL 和 Smalltalk 经常被用于原型化。它们成功的原因之一是其非常可靠：您无需担心内存的释放和冲突。程序员可以相对自由地处理内存，因为不必担心内存冲突。Java 具有这个特性，而且非常自由。

动态语言适合于原型化的原因之一是您不需要过早地作出决定。Java 恰恰相反，它强迫您明确地作出选择。这些选择带来了一些好处：如果发生错误的话，您可以写一些方法激活，在编译时得到错误的通知。您不必担心方法的激活出错，您还可方便地应用接口来代替类。

1.5 安 全 性

设计 Java 的目的是被用在网络/分布式的环境下。为此，许多重点放在安全性上。Java 是防病毒、无损害的系统。真正的技术是基于公开密钥体系的。

可靠性和安全性之间相互影响。例如，指针语义的变化使应用程序不再可能非法访问数据结构或存取对象中的私有数据，从而关闭了绝大多数病毒活动的大门。

1.6 结 构 中 立

Java 是为支持网络上的应用程序而设计的。一般来讲，网络是由各种各样的 CPU 和操作系统结构所组成的系统。为了使 Java 的应用程序能够在网络上的任何地方运行，编译器生成一种结构中立的对象文件格式：编译后的代码在给出相应的 Java 运行系统后可在许多

处理器上运行。

这不仅对于网络，而且对于单个系统软件分布也是有用的。在当前的个人计算机市场中，应用程序的开发者不得不开发许多种与 IBM PC 及 Apple Macintosh 兼容的版本。随着 PC 市场通过 Windows NT 向许多CPU 结构的扩张，Apple 从 68000 向 PowerPC 的扩张，使得软件产品不可能在所有的平台上运行。有了 Java，应用程序的同一版本可以在所有的平台上运行。

Java 编译器产生与特定的计算机结构无关的字节代码，它们不仅被设计成易被任何机器解释，而且可以容易地转换成本地机器代码。

1.7 可移植性

结构中立是可以被移植的基础。不像C 和C++，Java 没有语言规格说明方面的应用依赖，原始数据类型的大小及其上的算术运算都是特定的。例如，int 总是意味着一个有符号的 32 位整数；float 总是意味着一个 32 位的浮点数。

那些属于系统一部分的库定义为可移植的接口。例如，有一个抽象的窗口类及其在 Unix，Windows 和 Macintosh 上的实现过程。

Java 系统本身是可移植的。新的编译器是用 Java 写的，运行时刻的解释器是用具有清晰的移植“边界”的 ANSI C 来写的，这种移植“边界”是至关重要的 POSIX。

1.8 解 释 性

Java 是一种解释性的语言，由 Java 解释器解释执行。Java 解释器可以在任何机器上直接运行 Java 字节代码，只要这些机器上移植了 Java 解释器。

作为字节代码流的一部分，更多的编译时刻信息在运行时装载和获得，RPC 协议就是基于此而产生的，这使得程序更易于调试 。

1.9 高 性 能

虽然被解释的字节代码的性能在一般情况下是足够的，但是，有时也有高性能的要求。对于应用程序所运行的特定的 CPU 来说，字节代码可以在运行时刻被翻译成机器代码。对于那些习惯于一般的编译器和动态装载的设计人员来说，这就像在动态装载里放进最后的机器代码生成器。

字节代码的格式是为了生成机器代码而设计的，生成机器代码的实际过程非常简单。字节代码的生成过程是：自动登记申请空间，编译器在生成字节代码时做一些优化。

Java 采用了许多技巧使解释了的字节代码执行速度相当令人满意。这些技巧包括：

(1) 内置多线程

(2) 经过优化的高效的字节代码

(3) just-in-time 编译

(4) 链入 C 代码的能力

Java 利用应用程序的空闲时间进行系统管理，如收集空间、一般系统维护等。

在 Sun Microsystems SPARCStation 10 上，每秒可以运行 300 000 个字节代码的方法调用，字节代码转换成机器代码的性能几乎与本地的 C 或 C++没有区别。

1.10 多 线 程

世界上许多事情同时发生，而写一个同时处理许多事情的程序比写一个传统的单线程的 C 或 C++程序要难得多。

Java 将普遍使用的管程和条件变量应用在同步机制中。将这些概念融合在一起放在 Java 语言中，它们变得更加容易使用和可靠。这种风格源自 Xerox 的 Ceda/Mesa 系统。

多线程的其他好处是更好的交互反应和实时的行为。然而，这一点是有限的，它取决于实际的平台：独立的 Java 运行环境具有良好的实时性；在运行其他的系统，如 Unix，Windows，Macintosh 或 Windows NT 时，这种实时性由于所在的系统受到了限制。

1.11 动 态 性

在许多方面，Java 是一种比 C 或 C++更为动态的语言。它是为适应变化的环境而设计的。

例如，C++的一个主要问题是在应用时总会产生副作用。假如 A 公司开发了一个类库（即插即用组件），B 公司购买了它并且用于自己的产品中（假设 A 公司是一个操作系统供应商，B 公司是一个应用软件供应商），那么一旦 A 公司改变了这个库，发布了一个新的版本，B 公司几乎一定要重新编译并发行他们的软件。当最终用户分别从 A 公司和 B 公司购买软件时，就会发生问题。

又如，A 公司发行了一个升级的库版本，则 B 公司的所有软件都会出现问题。在 C++中，这个问题可能避免，但非常困难，而且无法直接采用该语言提供的任何面向对象的特性。

通过把这些模块之间的连接“向后推移”，Java 完全避免了这些问题，并使得面向对象的范例更加直接了当 。库可以自由地添加新的方法和实例变量，并不影响任何客户。

Java 引入了接口，这是一个从 Objective C 借用的概念，它与类相似。接口是一个对象应答的方法集的简单规格说明。它不包括任何实例变量和实现过程。接口可以支持多重继承（它们与类不同），可被用于比固有的类继承结构更为灵活的环境。

总之，Java 提供了一个有力的工具，供开发者自己支配。Java 使编程更加容易，因为它是面向对象的，可以自动回收空间。另外，编译 Java 代码是结构中立的，Java 应用对于多样化的环境（如 Internet）是理想的。

第二章　面向对象的程序设计

近几年来，面向对象这个词对我们来说是再熟悉不过了。几乎每个人都在谈论它。

然而，我们如何看待面向对象以及面向对象技术这股热潮呢？它是货真价实的，还是商家的炒作？事实上，两者兼而有之。面向对象技术确实为软件开发人员和软件产品提供了许多前所未有的好处，但在面向对象技术发展的历史上，围绕着这一概念和技术也存在着大量的不实之辞。面向对象的概念并不容易掌握，而且人们对这个概念的理解也随着时间的推移在发生变化。许多公司成了它的牺牲品，他们声称他们的软件产品是面向对象的，但事实上却不是。这种做法使不少客户产生了困惑，导致错误的信息广泛流传，同时也失去了对面向对象技术的信任。

如今，计算机界已开始克服这种滥用和误用面向对象概念的情况。同时，越来越多的人真正认识到面向对象技术及其优势。Java 的出现正是面向对象技术的杰出体现。如果说 Java 有什么划时代意义的话，那就是将网络从传统的数据传输工具扩展为对象的传输工具。

在本章，我们将讨论面向对象程序设计中的一些重要概念，它们与 Java 程序设计有着密切的关系。

从理论上讲，面向对象的核心技术包括对象、消息、类、继承等几个重要的概念。

2.1　对象

对象：是把数据及其相关方法或函数调用集合在一起的程序。经常被用来模拟现实世界中的物体对象。

现实世界中对象的两大特征是状态和行为。以自行车为例：

状态：如车的档位、脚踏板的位置、两个轮子、档的数目；

行为：如自行车的刹车、加速、减速、换档。

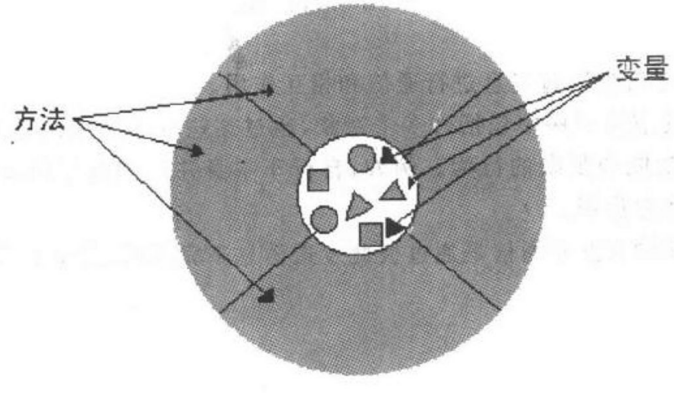

图 2.1　对象

软件中的对象模型是根据现实世界中物体对象的这两大特点来确定的，它们用变量来存放状态值，而用方法来实现其各种行为。所有软件对象所知道的(状态)和能做的(行为)都是通过该对象中的变量和方法来表达的；而所有该对象不知道的和不能做的都被排除在该对象表述范围之外。

对于自行车来说，一个状态可以表示如下：

当前速度是 10 mph，脚踏板的节奏是 90 rpm，档位是第 5 档。

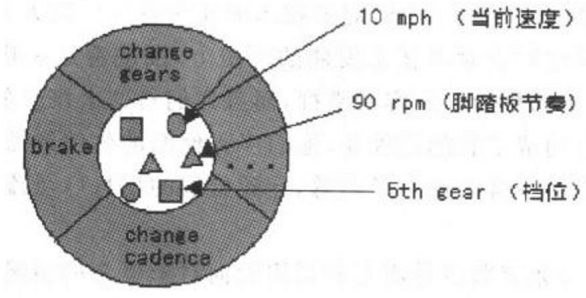

图 2.2 您的自行车

你能用这样的软件对象来表示现实世界中的物体。例如，一只狗可以是动画程序中的一个对象；而自行车可以是一辆电子锻炼用车。

一个抽象化的概念也可以是一个对象，如在 GUI 交互式窗口系统中，“事件”是一个普通的对象，用来表达用户按下某一鼠标或键盘上的键。

对象中的变量构成了该对象的核心，而方法好像是包围在外，将其变量和不重要的方法实现细节隐藏起来。要改变这些变量的值(也就是改变状态)，就必须调用在该对象中定义的方法，这就是封装。封装的两大优点如下：

(1) 模块化：可以独立于其他对象来进行开发、维护和修改该对象的源代码；

(2) 信息隐藏：一个对象有一个公共的接口为其他对象所使用，而每个对象也可以保留和修改私有的信息和方法，而不影响其他相关对象。

2.2 消息

软件中的多个对象通过消息进行通信和相互作用。

通常一个大的应用程序中包括许多个对象，通过这些成组对象的相互作用和合作共同完成更加高级的功能和复杂的行为。例如，自行车本身并不能做任何事，只有当另一对象(人)作用于它时才起作用。

软件对象之间的交互和通信是通过彼此发送消息来完成的，见图 2.3。

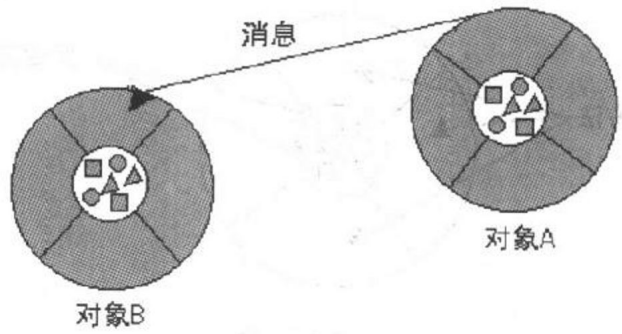

图 2.3　发送消息

例如，对象 A 需要对象 B 完成一件事情，则发送一条消息给对象 B 告诉它做什么。有时接收方需要更多的信息才能确切知道怎么做，消息中的参数就被用来传递这些信息(见图 2.4)。

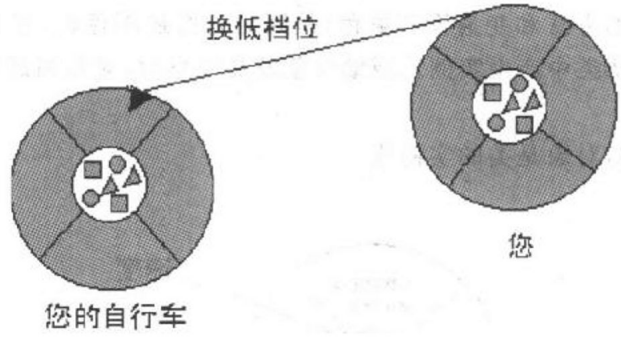

图 2.4　带有参数的消息

消息的三个组成部分如下：

(1) 指定接收方对象(自行车)。

(2) 需调用的方法的名字(换档)。

(3) 执行方法所需要的参数(换到第 5 档)。

消息的优点如下：

(1) 消息提供了对象交互的统一手段。

(2) 不同进程或不同机器上的对象也可以通过消息相互作用。

2.3　类

用来定义某一类对象共有的变量和方法的原型就是类。

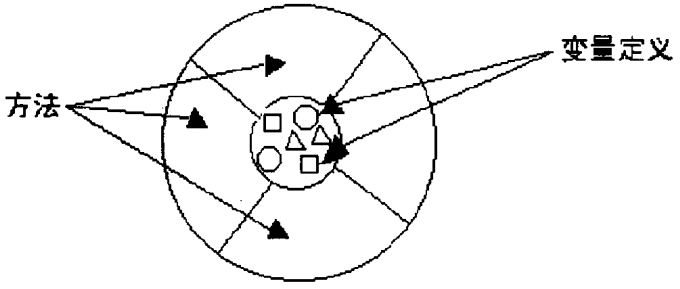

图 2.5 类

在现实世界中，通常有许多对象同属于某一种类。例如，你的自行车只是许多同类自行车中的一辆，我们称你的自行车为该自行车类中的一个“实例”。

类和对象非常相似，一些困惑经常源于它们之间的差别。在现实世界中，类本身很明显不是它所描述的对象，即自行车的蓝图并不是一辆自行车。但是，在软件中，区别类和对象有一些困难，部分原因是软件中的对象只是现实世界中的物体或抽象概念的电子模型，还与许多人在类和实例中不一致地使用“对象”这个词有关。

在图 2.5 中，类的方法和变量没有填色，是因为它们还不存在，在方法被调用和变量含任何值以前，必须先从类中产生实例。当给对象发送消息时，对象通过执行方法或改变变量的值来应答。

类是对象的原型，对象是类的实例化。

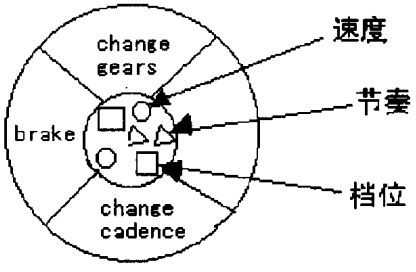

图 2.6 自行车类

类的优点主要表现在可重复使用性。

2.4 继承

类继承它们的超类的状态和行为。继承提供了一种有力的内在的机制来组织和使软件程序结构化。

类的层次结构可用下面的例子说明：山地车、赛车、双人自行车是不同种类的自行车，都是自行车类中的子类；相反，自行车是山地车、赛车、双人自行车的超类。

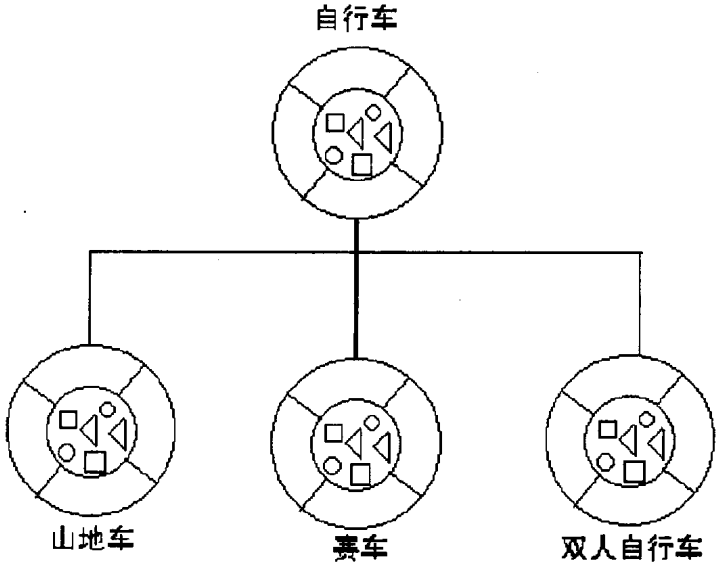

图 2.7　子类与超类

每个子类从超类处继承状态(变量定义)和行为(方法)。但是,子类并不局限于其超类所拥有的变量和方法的定义,它们可以在继承的基础上扩展自己的状态和行为。

继承可以是多层次的。

继承的优点主要表现在:

(1) 可重复使用性。

(2) 超类的最主要的部分被定义以后,其他的实现细节可以留待以后再实例化。

第三章　Java 语言简介

Java 是 Sun 公司开发的一种适合 Internet 上各种平台的新一代面向对象程序设计语言。它与 C++语言有许多相似之处，如果有一定的 C++基础，掌握 Java 是非常容易的。下面，我们从一个具体的例子开始全面介绍这种语言的特性。

下面是一个用 Java 编写的计算字符个数的应用程序：

```
class Count {
    public static void main(String args[ ])
      throws java. io. IOException
    {
      int count = 0;
      while (System. in. read( ) ! = -1)
            count ++;
      System. out. println("Input has " + count + " chars.");
    }
}
```

该程序的功能是统计输入的字符个数并显示出来。

3.1　程序的执行

Java 的运行方式有两种。第一种是将 Java 程序嵌入到 HTML 文件中，由远程客户端的 WWW 浏览器运行。这是 Java 语言独特的一种运行方式，也是设计该语言的初衷。我们将在后面的章节中介绍如何将 Java 程序嵌入到 HTML 文件中。运行 Java 程序的第二种方式是通过 Java 解释器。Java 解释器也叫 Java 的运行环境，在多种操作系统中均有实现版本，即 Sun 公司为多种操作系统平台提供了 Java 解释器。这种运行方式将 Java 程序作为一个独立的应用，体现了 Java 作为一种通用的程序设计语言的功能。事实上，Java 程序可在 WWW 浏览器中运行的原因也是因为在浏览器中包含了 Java 解释器。

在此，我们先介绍如何通过解释器运行 Java 程序，例如：

(1) UNIX 平台

```
%java Count
This is a test.
^D
Input has 16 chars.
```

(2) DOS 平台

```
C:> java Count
This is a test.
^Z
Input has 17 chars.
```

从表面上看，“This is a test.”只有 15 个字符(含空格)，由于换行符占一个字符，因此输