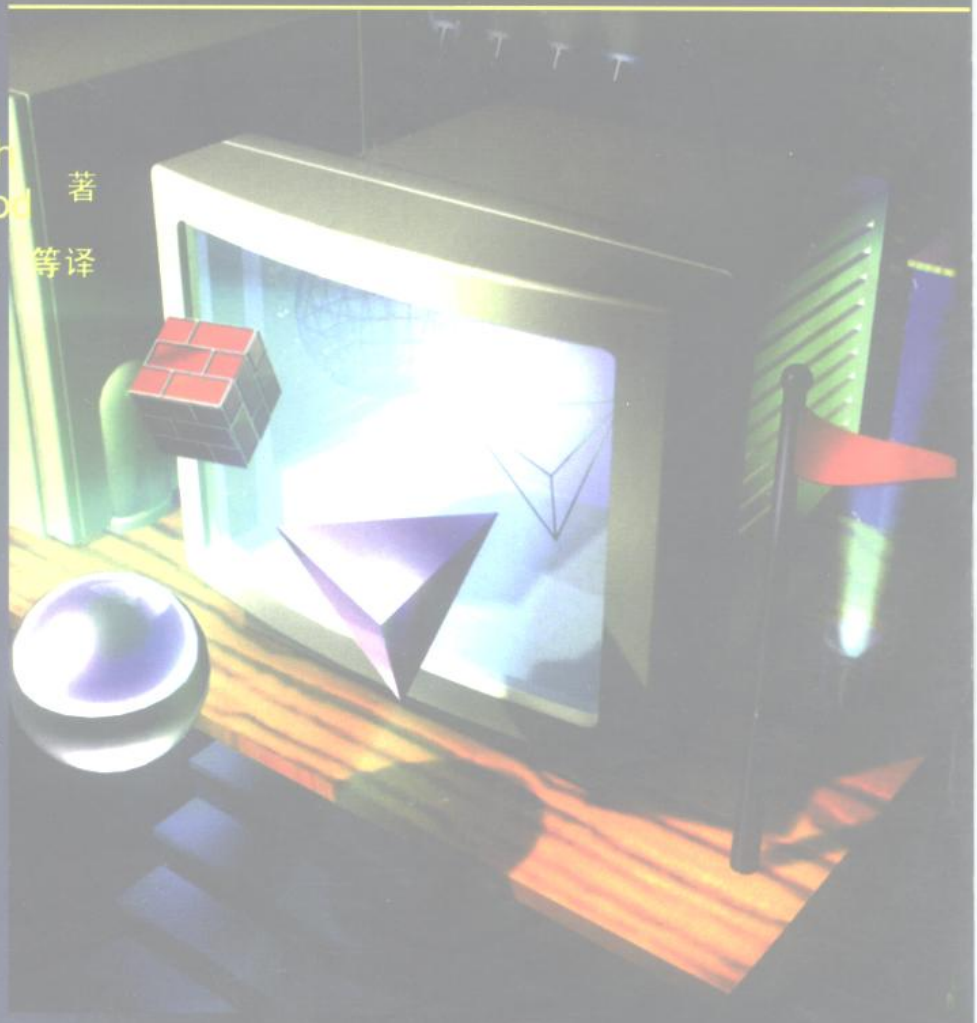
CMP

JAVA How-To

JAVA

开发人员指南

(美) Madhu Siddalingaiah
Stephen D. Lockwood 著
张录娥 等译



计算机软件开发
与程序设计
系列丛书

机械工业出版社
西蒙与舒斯特国际出版公司

计算机软件开发与程序设计系列丛书

JAVA开发人员指南

(美) Madhu Siddalingaiah
Stephen D. Lockwood 著

张录娥 等译

机 械 工 业 出 版 社
西蒙与舒斯特国际出版公司

本书是JAVA编程的指导书，全书共分10章，主要介绍利用JAVA语言进行基本图形、线程、事件和参数、用户接口、高级图形功能、多媒体与网络的编程技术。书中配有大量实例，读者通过对书中内容及实例的学习，可以快速掌握JAVA语言程序设计技术。

本书版权登记号：图字：01-97-0137

图书在版编目（CIP）数据：

JAVA开发人员指南/（美）史塔林埃伯（Siddalingaiah, M.），（美）洛克伍德（Lockwood, S.）著；张录娥等译.-北京：机械工业出版社，1997.5

（计算机软件开发与程序设计系列丛书）

书名原文：Java How-To

ISBN 7-111-05462-8

Ⅰ.J… Ⅱ.①史… ②洛… ③张… Ⅲ.Java语言-程序设计-手册 Ⅳ..TP312-62

中国版本图书馆CIP数据核字（97）第04374号

出 版 人：马九荣（北京市百万庄南街1号 邮政编码100037）

责任编辑：傅豫波

三河永和印刷有限公司印刷·新华书店北京发行所发行

1997年5月第1版第1次印刷

787mm×1092mm 1/16· 37.5印张· 922千字

0001-7000册

定价：64.00元

前　言

在未来几年里，Java 将大大改变人们对计算机原有的观念。由于 Java 的产生，计算机的功能将达到一个新的高度。Java 的应用包括（applet）（包括 HTML 页）可以使用声音、图形进行实时用户交互，数据和程序也不必非要存储在某个特定的计算机上，它们可以驻留在 Internet 的任何地方，并且只要需要，这些数据和程序的可以被下载。实现以上功能只需编写 Java 代码就可以了。

我们编写《Java 开发人员指南》一书的目的就是要帮助读者了解 Java 那些比较难以掌握的内容，并介绍许多解决问题的编程方法。

1　“问题和解答”的风格

象 Waite Group 出版社出版的其他“开发人员指南”系列图书一样，本书也采取由浅入深的介绍方法，引导读者掌握解决疑难问题的技术和步骤，一般是先提出问题，然后介绍解决该问题的方法，接着进行总结评论并列出注意事项，最后再介绍几种其他可能的方法。本书既可作为教科书使用，也可作为参考书使用。

2　本书的读者对象

本书不是对 Java 语言的说明和介绍，而是面向程序设计人员的，以介绍 Java 编程技术为目的。因此，本书的读者范围很广，从专业程序设计人员到业余爱好者都适用。本书的结构是以教程序员如何使用 Java 语言编写特定程序而设计的。同时 Web 页开发人员也可从本书中获益，至于业余爱好者则可把本书作为学习 Java 的辅助教材。

为了充分利用本书，对编程员有个基本要求：就是必需具有 C 语言和面向对象编程技术的基本知识。如果读者对面向对象技术不是很熟悉的话，可先看本书中比较简单的问题（本书不介绍有关面向对象编程技术方面的内容）。Java 语言和 Java 应用编程接口 API（Application programming interface）的基本知识也只作粗略的介绍。

本书中所列的问题都来自实际应用，即来源于我们在 Internet 编程中遇到的问题或 Interntet 上常见的问题。

3　使用本书的环境

要使用本书，至少需要一台能运行 Windows 95 或 Windows NT 3.51（或更高版本）的计算机，或者拥有如下计算机系统：Macintoshes（System 7.5）、Sun Sparcstations（Solaris 2.3 或更高版本）；同时需要拥有 Java 开发包 JDK。其中 Java 开发包在对 Sun 系统的许多平台是免费的，请向 Javasoft（地址 www. javasoft. com）咨询，以便得到最新版本。JDK 中包括有：Java 解释程序（运行 Java 应用程序时必需的）和 Appletviewer（本地浏览应用程序包）。可能还需要购买几种商品化的 Java 开发环境。

4 Java 的发展历史

Java 最初只是一个研究项目的组成部分，该项目的任务是开发高级控制软件，这种软件最初命名为 Oak，是一个小型、可靠、易用、分布式、实时的操作环境。项目开始实施时，首选的是 C++，但是 C++ 在创建一个全新的语言时，碰到了困难。设计 Java 语言结构时，吸取了其他很多语言的特点，如 Eiffel、SmallTalk、C 和 Cedar/Mesa，这样就导致了一个十分出色的语言的产生，它可以用于开发安全的、分布式应用程序，这些应用程序适用于从网络设备到 WWW 和桌面。

5 内容介绍

本书共分十章，四个附录，以下为各章内容简介。

第 1 章：Java 简介。本章对 Java 语言进行了简单介绍。如基本语法、数据类型和控制语句等。另外，还讨论了 Stand－alone Java 应用程序和应用程序包。

第 2 章：基本知识。介绍如何使用 Java 应用程序完成一般的基本操作，例如：如何使用数据类型和数组、如何在字符串和数值之间转换、如何从键盘上输入命令参数、如何进行文件打印、文件 I/O 和如何使用数学函数等。

第 3 章：基本图形。介绍在 Java 中如何使用图形。例如如何画线、如何绘制各种基本图形、如何使用字体及颜色等。

第 4 章：线程。介绍如何使用线程。本章给出几个例子，用这些例子演示使用线程的几个问题。其中包括：多线程、线程优先级、同步机制和线程等待。

第 5 章：事件和参数。重点介绍了 Java 中不同事件的处理和应用包参数的使用。

第 6 章：用户接口。本章用一系列的例子演示用户界面的不同组合，如：按钮、检测框、列表框等。

第 7 章：高级图形功能。介绍一些复杂的图形管理技术，并且介绍了 image 及其双缓冲区的使用方法。

第 8 章：多媒体。介绍在 Java 中如何使用声音，并用具体例子作了演示。

第 9 章：网络应用。介绍 Java 中的许多联网功能。

第 10 章：其他技术和高级功能。介绍了几个有趣的高级功能。例如 Java 与其他代码的连接、应用包之间的通信和如何防止别人窃取应用包等。

附录 A：Java 类层次结构。包括了 Java API 中所有的类，是个比较实用的参考资料，一般在写说明书时会用到这些内容。

附录 B：Java 常见问题解答。

附录 C：Java 虚拟机规范说明。虚拟机操作简介。

附录 D：Java 应用包 Hall of Fame Applet Gallery。

目　录

第 1 章　Java 简介

1.1　Java 与 C/C++比较

Java 的外部特性有点象 C 和 C++，熟悉 C 和 C++或相类似语言的人，学习使用 Java 不会有什么困难，因为两者的流程控制语句和操作函数几乎没什么差别。

与 C 不同的是 Java 是面向对象的，并剔除了 C 和 C++中具有二义性的部分，使得 Java 相对来说更简洁些，因此使用 Java 开发应用程序能够减少代码量。

Java 还剔除了 C 和 C++中的许多冗余部分。C 在多年的发展过程中，形成了许多特有的技术，虽然 C++在 C 中增加了面向对象的特性，但也继承了 C 的固有缺点，其中最主要的问题就是 C/C++的预处理程序。C/C++的预处理程序将语言变成了一种难懂的东西，要弄懂预处理程序的宏代码和说明书往往要花费大量的时间。在 Java 编程中，不需要用包括头文件的方式来说明类型信息（类定义已编译成了二进制格式，本身包含了类型的信息）。另外 Java 用常量代替 #defines，用类代替 typedef。

在 C++中，“结构”和“类”只存在一些微小的差别，并且在大多数情况下，二者是相互冗 余的，因此 Java 只保留了对类的支持；大家都知道“联合”是依赖于平台的，因此 Java 不再支持“联合”。

面向对象的程序设计改变了函数和程序的设计方式。任何用过程实现的功能都可以通过定义一个类，并且创建该类的一些方法来实现。正因为如此，类之外的过程都没有必要存在了，因此 Java 中也不支持类之外的过程。

C++支持多重继承，这样在出现互为矛盾的条件时，会产生不可预料的结果。Java 支持的多重继承与 C++的多重继承不同，Java 使用接口完成多重继承，所谓“接口”是一组对象的方法的结合。接口只定义方法和常数，不定义变量，“接口”是完全多重继承和单个继承的一个折衷。

尽管操作符重载是个很重要的概念，但是 Java 中并没有包含操作符重载，学过 C++的人都知道操作符重载很繁琐。事实上，很多有经验的 C++程序员也不能很熟练的使用操作符重载这一功能。

虽然“指针”在 C++中的用处是很大的，但也有其固有的缺点，那就是：指针用错会带来无法预料的结果。即使是资深程序员编制的程序，而且这个程序已经过多年的正常运行，也会由于一个指针的误用而突然出错。Java 也保留了指针这个功能，但 Java 的运行支持系统会管理所有数组的下标指针，以确保下标不超出数组的边界。

1.2　语法

所有的程序设计语言在设计时都有其宗旨：C 的精华就在于它容易编译、运行速度快、便于系统之间的移植；C++着重于通过使用类库，便于代码复用，并尽可能保持与 C 一致；Java 则尽可能地保持与 C++相似，因此 Java 容易掌握，而且安全性增强了很多。这样 Java 的程序员就不必花费大量的时间去调试内存错误了。

下面列出 Java 的所有关键字：

abstract	boolean	break	byte	byvalue
case	cast	catch	char	class
const	continue	default	do	double
else	extends	final	finally	float
for	future	generic	goto	if
implements	import	inner	instanceof	int
interface	long	native	new	null
operator	outer	package	private	protected
public	rest	return	short	static
super	switch	synchronized	this	throw
throws	transient	try	var	void
volatile	while			

其中：byvalue, cast, const, future, generic, goto, inner, operator, outer, rest 和 var 在 Java 1.0 中没有用到。

注意：true 和 false 看起来象关键字，但从技术上来讲它们是布尔量。

表 1－1 为 Java 的操作符（按优先级排列）。

表 1-1 Java 的操作符

优先级	操作符	类型	完成操作
1	++	算术	先或后增一
	--	算术	先或后减一
	+, -	算术	单纯加、减
	~	整型	位取反
	!	布尔	逻辑取反
	(type)	任意	赋值
2	*, /, %	算术	乘，除，取余数
3	+, -	算术	加，减
	+	字符串	字符串连接
4	<<	整型	左移
	>>	整型	算术右移
	>>>	整型	逻辑右移

（续）

优先级	操作符	类型	完成操作
5	<，<=	算术	小于，小于等于
	>，>=	算术	大于，大于等于
	instanceof	对象	类型比较
6	==	Primitive	等于
	!=	Primitive	不等于
	==	对象	等于
	!=	对象	不等于
7	&	整型	位与
	&	布尔	布尔与
8	^	整型	位异或
	^	布尔	布尔异或
9	|	整型	位或
	|	布尔	布尔或
10	&&	布尔	条件与
11	||	布尔	条件或
12	?:	布尔，任意，任意	条件（ternary）
13	=，*=，/=，%=，+=，-=，<<=，>>=，>>>=，&=，^=，|=	变量，	任一赋值操作符

1.3 数据类型

现代计算机语言中所有的基本数据类型 Java 都支持，并且大部分数据类型和 C 和C++ 语言的数据类型相似，只有一小部分不同。Java 的数据类型都进行了严格的定义。如表 1－2 所列即为 Java 的基本数据类型。

Java 不允许两个类型截然不同的数据之间相互赋值，但允许数值型数据之间和相同对象子类和超类之间的赋值。如果两个类型长度不同（例如：将整型赋给字节型），则取长度较小的一个类型。

整型数值有 8 位字节、16 位短整数、32 位整数和 64 位长整数之分。整型数据类型由两部分组成：符号和整数。Java 没有无符号数据类型，因此，Java 增加了一个>>>操作符来完成无符号（逻辑）右移操作。

实型数值类型有 32 位浮点型和 64 位双倍型之分。这两个数据类型的算术操作按 IEEE 754 的规范进行。

Java 的字符型数据与 C 的有点不同，Java 使用的是 Unicode 字符集标准，该标准中字符（char）是 16 位无符号型的，而 C 的 char 是 8 位 ASCII 字符。字符型数据不能赋值给其他类型的变量，其他数据类型值也不能赋值给字符类型变量。

布尔型数据是 1 位数据，它的取值可以是 True 或 False。Java 的布尔量是一种特殊的数据类型，它与其他数据类型之间不能相互赋值。

Java 中的字符串是一种对象，这也与 C 或 C++ 不同。string 在 C 中是以 null 结束的字符数组，而在 Java 中却不是这样，它是 java.lang.String 类的一个实例。

Java 的数组是对象，而且是一维对象，多维数组用数组的数组实现。一个长度为 n 的数组由 n 部分组成，每部分分别用整数 0 到 n－1 来表示。

表 1-2　Java 的基本数据类型

类型	描　述	大小	最小值 最大值
boolean	True 或 False	1 bit	NA NA
char	有符号整数	16 bit	\ u0000 \ uFFFF
byte	有符号整数	8 bit	－128 127
short	有符号整数	16 bit	－32768 32767
int	有符号整数	32 bit	－2147483648 2147483647
long	有符号整数	64 bit	－9223372036854775808 －9223372036854775807
float	有符号整数	32 bit	+/－3.40282347E+38 +/－1.40239846E－45
double	有符号整数	64 bit	+/－1.79769313486231570E+308 +/－4.94065645841246544E－324

1.4　控制语句

Java 中的控制语句有：if，else，switch，for，while，do，break，continue 和 try－catch。这些语句与 C++ 中的控制语句十分相似，只有两处有明显的区别：第一是 Java 的 boolean 量是不能赋给其他类型的；第二是值 0 和 null 与 False 是不同的，非零和非空和 True 也不同。

1.5　小应用程序 Applet 与应用程序的比较

Java 的应用程序由一个包含 main（）方法的类和该类引用的其他类组成。main（）方法说明为 public static void，并接受字符串数组参数。应用程序的操作环境是由运行的操作

系统派生的。一个应用程序可以很简单，例如仅仅输出“Hello World”，也可能很复杂，如最先进的字处理器或编译程序（javac 编译程序本身就是一个 Java 应用程序）。Java 应用可以使用 Java 语言的所有功能和类库。

一个 Applet 至少应有一个 Java 公共类，并且该类是 java.awt.Applet 的子类（或者是 java.awt.Applet 子类的子类），它的操作环境是从支持 Java 的 Web 浏览器中派生出来的，如 Netscape Navigator 2.0 或 HotJava。

相对应用程序而言，对 Applet 的限制要多些。因为，Applet 是在不太安全的网络（如 Internet）上运行的。Applet 不允许在本地系统中读取文件，典型的限制是只能打开 Applet 下载的那个 host 的连接，而不能打开网络的其他连接。

1.6 平台无关性

仅用 System.in.readln（）进行输入和 System.out.println（）进行输出，就可编写一个简单的 Java 应用程序。它和 UNIX 或 MS-DOS 的简单命令行程序十分相象。对于许多应用来说，这已经足够了。但是，从另一方面来说，大多数现代计算机都使用了图形用户接口 GUI，例如 Macintosh，Windows 95/NT，UNIX 和 X Window。这样就要求用户有窗口、图标和鼠标控制。各种操作系统都有自己的 GUI，并且互不相同，因此大多数应用程序都是针对某一特定 GUI 编写的。这样要把一个程序从一个环境移到另一个环境中运行，需要花费很多精力和时间。而 Java 在这方面就有其独特的优越性，Java 代码只要编写一次，就可在支持 Java 虚拟机的任何环境中运行。因此，Java 在 Solaris，IRIX，Linux，Windows 95，Windows NT 和 Macintosh 环境中均可运行。OS/2，Windows 3.1 和掌上计算机的应用接口正在研制之中。这就是人们长期以来所梦寐以求的：一次编写，各处复用。

实现这种通用性的关键是抽象窗口工具包 AWT（Abstract Windowing Toolkit）。AWT 是 Java 类的集合，这些类完全适合所有 GUI 环境的基本功能，而不是针对某个特定 API 的。其中每个类都依赖 Java 虚拟机将它们的功能转换为适合特定 GUI 的功能。

编译 Java 程序时，是将程序转换为一种结构性的中间字节代码格式，这种字节代码可以在任何系统上运行，条件是：只要能在该系统中实现了 Java 虚拟机。由于 Java 程序字节代码，不是某种机器代码，Java 程序运行时，就需要一个解释程序，因此 Java 是一种解释语言，它不具有编译 C 所具有的性能。只有在 Java 的编译程序发布后，它的速度才可能与 C 和 C++ 的编译速度相当。

第2章 基本知识

本章将讨论 Java 中的基本操作，这些操作包括：打印字符、获取键盘输入、读写文件、使用数组和维护字符串和数值等。本章的内容与 C/C++ 的相关内容很相似，因此，对比较熟悉 C 语言的读者来说，这些内容是十分简单的。

本章中的所有例子都是“应用程序”，而不是 Applet。这是因为有些内容只适用于应用，而不适合 Applet，例如命令行参数（Applet 需要 applet 参数）、读写文件（Applet 不能读写本地硬盘）等，但是其他基本操作都既适合应用，也适合 Applet，例如字符串和数值之间的转换、字符串和数值的连接、字符串分解、数组的使用和数学函数的使用等。

下面是本章各节内容的简介：

第 2.1 节　打印字符。本节中的例子将介绍如何显示文字和数值。但它不象一般的计算机书籍那样打印“Hello World”。

第 2.2 节　接受键盘输入。本节演示如何获取键盘输入，例子提示用户输入一串字符，然后输出该字符串和字符串中的字符个数。

第 2.3 节　将字符串转换为数值。字符串和数值之间的转换是十分有用的操作。因为，Java 从键盘读入的信息都作为字符串存储的，如果需要从键盘上读取数值，就必需将所读入的内容转换为数值。本节中的例子将演示如何将键盘输入转换为各种数值类型。

第 2.4 节　将数值转换为字符串。本例计算当前的国家债务，因为数值很大，因此输出显示时，需要将该数值转换为字符串插入一些逗号以便阅读，同时还讨论了 Date 的构造函数和 getTime 的使用方法。

第 2.5 节　存取命令行参数。在编程过程中，经常要用到命令行参数，本节将介绍在 Java 中如何使用命令行参数。本节的例子用月和年作为参数，输出指定月的日历。本例也用到了 Date 类中的方法。

第 2.6 节　读文件。本节介绍在 Java 中如何读文件。该例将要读的文件名作为命令行参数传递给应用，读入的文件内容显示在屏幕上，显示方式象 UNIX 中的 more 工具软件采用的方式。

第 2.7 节　写文件。学会了如何读文件后，还要学习如何写文件。本节的例子是将一个文本文件转换为 DOS，UNIX 和 Macintosh 系统中的文本文件格式。工作原理是读取文件并把行结束符转换为命令行参数指定的格式。

第 2.8 节　字符串和数值之间的连接。字符串和数值之间的连接是一个很有用的特性。有了这个功能，就可以很容易地在一个 print 语句打印各种值了。本节的例子是将字符串和日期连接起来。

第 2.9 节　使用数组。本节介绍如何使用数组。本节例将一个文件的内容读入到一个数组中，并用快速排序法进行排序。

第 2.10 节　字符串分解。字符串分解是将一个字符串分解为由任意字符分隔的子串，是一个非常有用的功能。本例是根据命令行输入的姓名，在文件中寻找对应的电话号码。

第 2.11 节　数学函数的使用。Java 中包含了很多数学函数，它们可以完成各种计算任务。本节的例子通过计算贷款的月利息演示几个数学函数的使用。

2.1　打印字符

2.1.1　问题

现在需要在屏幕上打印一些常用的转换系数，要求同时打印文字说明和双精度浮点数，不要求有严格的格式，但要求每个转换系数占一行，最简单的方法是什么呢？

2.1.2　技术

Java 的运行库提供了几个打印字符串和基本数值类型的类，本例用到的方法有 print 和 println。这些方法完成向屏幕输出操作（标准输出），它们包含在类 System.out 中。

2.1.3　步骤

1. 创建应用程序源文件。建立一个名为 convert.java 的新文件，输入如下内容：

```
/*
 * Convert.class prints a short table of common conversion factors.
 */
class Convert {
/*
 * Note that the main method must to be defined as
 * public static void main (String []) to be called from
 * the Java interpreter.
 */

public static void main (String args []) {

  double mi_to_km = 1.609;
  double gals_to_l = 3.79;
  double kg_to_lbs = 2.2;

  System.out.print ("1 Mile equals \t");          //No newline at end here
  System.out.print (mi_to_km);
  System.out.println ("\tKilometers");            //Print newline at end.

  System.out.print ("1 Gallon equals \t");
  System.out.print (gals_to_l);
  System.out.print ("\tLiters \n");               //\n works as newline also.

  System.out.print ("1 Kilogram equals \t");
  System.out.print (kg_to_lbs);
  System.out.println ("\tPounds");
```

```
}
}
```

2. 编译并运行该应用程序。编译时用 javac 或所提供的 makefile 文件，运行时键入：

java Convert

程序运行后，以下转换系数就显示在了屏幕上：英里-千米，加仑-公升，磅-千克。输出结果如图 2-1 所示。

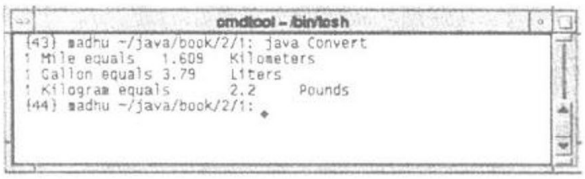


图 2-1 Convert.java 的输出样本

2.1.4 工作原理

Convert 类只包含了一个方法：main。这个方法的作用相当于 C/C++ 中的 main 函数。Java 解释程序运行应用程序的第一件事就是调用 main 方法。

程序中还说明了三个双精度类型的数据代表三个转换系数，并赋了初值。打印是通过调用 System.out.print 和 System.out.println 方法实现的，它们的区别是：后者在打印后换行，而前者打印完成后不换行。print 和 println 方法有好几种，它们分别接受不同的参数，本例使用的方法接受的参数是字符串和双精度型数据。

Java 也支持 C/C++ 程序员熟悉的许多 ESC 序列，如表 2－1 所列即 3 为 Java 的 ESC 序列及功能说明。在本例中使用了制表符 \ t 和换行符 \ n。

表 2-1 Java 的 ESC 序列

ESC 序列	UNICode 值	作用
\ b	\ u0008	退格删除 BS
\ t	\ u0009	水平制表 HT
\ n	\ u000a	换行 LF
\ f	\ u000c	走纸 FF
\ r	\ u000d	回车 CR
\ "	\ u0022	双引号 "
\ '	\ u0027	单引号 '
\ \	\ u005c	反斜杠 \
八进制 ESC 序列	\ u0000 到 \ u00ff	八进制 ESC 序列

注意：main 方法必须定义为 * i (String [])，否则，Java 解释程序就识别不出它，而输出一个出错信息。另外源程序的文件名要与包含 main 方法的类名一致，例如，类 myMainClass 应该在名为 MyMainClass.java 的文件中。

和 C/C++一样，从 main 返回将终止程序的运行，main 不能返回值，因为它是 void 型的。要返回应用完成的状态，可以使用类似于 C/C++的 exit（）函数的机制，我们将在以后的例子中介绍如何返回程序的执行状态。

2.1.5 总结

Java 中，没有与 C/C++中的 printf 相类似的函数。C 的 printf 可以严格地定义数据的格式并带有参数表，而 Java 的 print 没有那样复杂，它不允许程序员指定详细的数值格式，和其他 Java 方法一样，它只接受固定的参数数目和类型。

因为 Java 总要检查方法的参数类型的一致性，因此不能使用有可变参数个数和可变参数类型的方法。

2.2 接收键盘输入

2.2.1 问题

如果想读取键盘的输入并显示输入字符的个数，该怎样做呢?

2.2.2 技术

方法 readLine 的作用是从 DataInputStream 中读取输入，并返回一个字符串。从键盘接受数据也必须使用该方法。输入的字符个数可以用 String 类中的 length 方法确定，字符串和它的长度可以用 print 或 println 输出。

2.2.3 步骤

1. 创建应用程序源文件。建立一个名为 Input.java 的新文件，输入如下内容：

```
import java.io.*;
/*
 * Input.class reads a line of text from standard input,
 * determines the length of the line, and prints it.
 */
class Input {
public static void main (String args []) {
    String input = "  ";
    boolean error;

/*
 * DataInputStream contains the readLine method.
 * Create a new instance for standard input System.in
 */
  DataInputStream in = new DataInputStream (System.in);
/*
 * This loop is used to catch IO exceptions that may occur.
 */
    do {
      error =false;
```

```
            System.out.print ("Enter the string>");

    /*
     * We need to flush the output because there is no newline at the end.
     */

          System.out.flush ();
            try {
                input = in.readLine ();              //Read the input.
            } catch (IOException e) {
                System.out.println (e);              // Print exception
                System.out.println ("An input error was caught");
                error = true;
            }
        } while (error);
            System.out.print ("You entered \" ");
            System.out.print (input);        //readLine does NOT keep the \n or \r.
            System.out.println ("\" ");
            System.out.print ("The length is ");
            System.out.println (input.length ()); //Print the length.
    } //end of main ()
    }
```

2. 编译并运行该应用。编译用 javae 或所提供的 makefile 文件，运行时键入：

```
java Input
```

应用程序开始运行后，提示输入一个字符串，用户输入字符串并按回车后，所输入的字符串及其长度就显示在屏幕上，如图 2-2 所示。

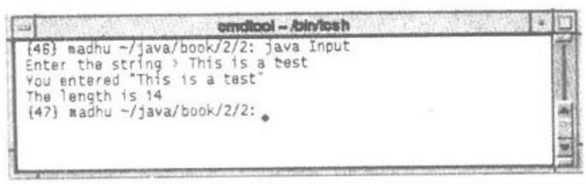


图 2-2　Input.java 的输出

2.2.4　工作原理

为了从键盘输入数据，必须先创建一个 DataInputStream，这个工作是通过调用 DataInputStream 的构造函数来完成的，给它指定的参数是 System.in（标准输入）。readLine 释放一个 IO 异常码，do-whilw 循环中的 try-catch 用来捕捉该异常码。readLine 返回的字符串赋给字符串变量 input。

然后用 print 将 input 打印出来，input 的长度用 String 类中的 length 方法确定。