Java 程序设计语言

〔美〕 Ken Arnold，James Gosling 著

杨承高　李健钧
刘清革　陈仕华　译

北京大学出版社
北　京

著作权合同登记号　图字：01-97-1094

图书在版编目(CIP)数据

Java 程序设计语言/ (美)阿诺德(Arnold, K .), (美)戈斯林(Gosling, J .)著 . - 北京:北京大学出版社,1997 .8
ISBN 7-301-03472-5

Ⅰ.J…　Ⅱ.①阿…　②戈…　Ⅲ.Java 语言　Ⅳ.TP312Ja

中国版本图书馆 CIP 数据核字(97)第 14148 号

书　　名: Java 程序设计语言
著作责任者: 〔美〕 Ken Arnold, James Gosling 著,杨承高等译
责 任 编 辑: 王　欢
标 准 书 号: ISBN 7-301-03472-5
出　版　者: 北京大学出版社
地　　址: 北京市海淀区中关村北京大学校内　100871
电　　话: 出版部 62752015　发行部 62559712　编辑部 62752032
排　版　者: 兴盛达激光照排中心
印　刷　者:
发　行　者: 北京大学出版社
经　销　者: 新华书店
787×1092　16 开本　16 印张　400 千字
1998 年 1 月第一版　1998 年 1 月第一次印刷
定　　价: 32 .00 元

This book is dedicated to the Java team
From whose hard work and vision
A mighty oak has grown

To Susan— *K .A .*

To Judy and Kate— *J .A .G .*

ISBN 0-201-63455-4
1 2 3 4 5 6 7 8 9-MA-99989796
First Printing, May 1996

前　言

美丽的建筑物并不仅仅是科学技术的结晶。它们是真正的有机体,富含精神内涵;它们是采用富有灵感的最佳技术所铸就的艺术品,而非由单一模式或按一般的工程设计堆积而成。

——Frank Lloyd Wright

Java 程序设计语言(以后简称 Java)已经被软件开发者和 Internet 服务提供者欣然接受。Internet 和万维网用户得到的益处是他们可以访问来自 Internet 上任意节点平台的独立的应用程序。而使用 Java 的软件开发者则只需一次编码,不必将应用程序移植到各种软件和硬件平台。

对多数人来说,Java 主要是一种创建万维网小应用程序(applet)的工具。Applet 是 Java 的一个术语,表示运行于 Web 页中的小应用程序。Applet 可以于下载后在浏览页中完成一个任务或与用户交互,而不使用 Web 服务器的资源。当然,有些 applet 会因为自身的目的与服务器交互,但那是它们自己的事。

Java 确实对于像 Web 这样的分布式网络环境很有价值。但是,Java 的好处远远超出了这一点,它又是一种通用的程序设计语言,适于创建不依赖于网络功能或者用于别的目的的各种应用程序。对许多组织而言,以安全方式运行远程主机程序的能力,是对 Java 的最关键的要求。

另外一些机构把 Java 作为通用程序设计语言,用于开发独立性不是那么重要的项目。Java 的易编程性和安全性可以帮助开发者迅速生成正确的代码。因为具有像垃圾收集和类型安全(type-safe)引用等特性,一些常见的程序错误不会再发生。Java 支持多线程的特性迎合了当今基于网络和图形用户界面的应用程序需要同时运行多任务的要求,同时异常处理(exception handling)机制使出错处理变得容易,因为它内置有功能强大的工具,Java 本身是一种简单的语言,所以程序员可以很快熟悉。

Java 的设计目的是最大的可移植性和尽可能少的实现依赖性。例如,int 在所有的 Java 实现中,无论运行 Java 程序的机器体系结构如何,都表示 32 位带符号整数。定义有关语言和运行期环境的各种可能情况允许用户在任何地方运行编译码,与任何具有 Java 环境的人共享代码。

Java 和当前使用的大多数程序设计语言具有许多共性。但是,与 C 和 C ++ 不同,Java 将自动垃圾管理、异常处理与对线程的支持结合起来。

有关本书

本书可以教会熟悉基本程序设计概念的人们用 Java 进行程序设计。它并没有形式地或完全地解释 Java 语言。它也不是面向对象程序设计的介绍,虽然涉及到了有些问题并建立了公用的术语。

对 C 和 C ++ 程序设计人员来说,Java 看起来应该是比较熟悉的,因为 Java 是利用 C 和 C ++ 中相似的构件设计的。本丛书中的其他书本,还有许多在线文档,主要与 Java 小应用程序有关。其他参考书,请见参考书目。

第 1 章是 Java 概述。不熟悉面向对象程序设计概念的程序设计人员应该读一下概述,已经熟悉面向对象程序设计规范的人可以发现概述是关于 Java 面向对象特性的有益介绍。

第 2、3、4 章介绍 Java 的面向对象核心特征,如用于定义程序组件的类的声明,以及根据类定义创建的对象。第 2 章是类和对象,描述 Java 语言的基础。第 3 章是继承类,描述已经存在的类如何通过继承或子类化创建具有新的数据和行为的新类。第 4 章是接口,描述如何声明接口类型。接口类型是为类设计者和实现者提供最大灵活性的行为的抽象描述。

第 5 和第 6 章介绍了与绝大部分语言相同的标准语言构件。第 5 章是语言符号、操作符和表达式,描述构成语句的语言符号,以及语言符号和操作符如何构成表达式,表达式如何计算等。第 6 章是控制流,描述控制语句如何控制语句的执行次序。

第 7 章是异常,描述 Java 功能强大的错误处理能力。

第 8 章是字符串,描述对字符串(String)对象的内置语言和运行时的支持。

第 9 章是线程,解释 Java 的多线程实现。许多应用程序,比如基于图形界面的软件,必须同时注意多个任务。这些任务为实现目标必须互相合作,而线程则要符合相互合作的多任务的要求。

第 10 章是包(Package),描述把 Java 类归并为包的机制。

第 11 章到第 14 章介绍 Java 核心类库包的主体。第 11 章是 I/O 包,描述 Java 的基于流(streams)的输入输出系统。第 12 章是标准工具,介绍 Java 的诸如向量和散列表等工具类。第 13 章是利用类型编程,描述 Java 类型相关类,包括描述每个类和接口的单个对象,以及把诸如整数和浮点数等基本数据类型包装成它们自己对象类型的类。第 14 章是系统编程,介绍提供底层平台特性访问的系统类。

附录 A 描述 Java 对本机方法(native methods) 的支持,本机方法是指一种访问以底层平台本机语言编写的代码的方法。

附录 B 列举了 Java 系统本身能够处理的运行中的异常和错误。

附录 C 是一些可以用作速查手段的表格。

最后,参考书目列举了有关面向对象,利用线程编程和其他主题的书籍。

实例和文档

本书中的所有程序实例都已在写作本书时最新的 Java 1.0.2 FCS 版上编译和运行。一般地讲,本书仅介绍 Java 1.0.2 的特性。但我们也介绍了编写可编译程序之外的话题——学习一种语言的原因之一是为了用好它。因为这个原因,我们尽量展示一些有关良好的编程风格和设计的原则。

有些地方我们提到在线文档。Java 开发环境提供一种利用文档说明从编译类中自动生成文档(通常是 HTML 文档)的方法。这种文档一般可以用 Web 浏览器阅读。

没有哪种技术书籍的写作是一种“孤岛”行为,我们的行为则更像是一片“大陆”,有许多人提供了技术帮助、细心复审、有用信息和写作建议。

Trilithon Software 的责任编辑 Henry McGilton 在本书成书过程中扮演了“主要编辑消防员”的角色。丛书编辑 Lisa 则以顽强的毅力给予支持。

很多复审人员在他们繁忙的日程中抽出时间阅读、编辑、建议、修改和删除材料，唯一的目的是让这本书成为一本好书。Kevin Coyle 完成了所有级别上最详细的编辑复审。Karen Bennet，Mike Burati，Patricia Giencke，Stephen Perelgut，R．Anders Schneiderman，Susan Sim，Bob Sproull，Guy Steele，Arthur Van Hoff，Jim Waldo，Greg Wilson，和 Ann Wollrath 作了深入的复审。Geoff Arnold，Tom cargill，Jimmy Torres，Arthut Van Hoff 和 Frank Yellin 在一些关键的地方提供了有用的评论和技术信息。

Alka Deshpande，Sharon Flank，Nassim Fotouhi，Betsy Halstead，Kee Hinckley，K．Kalyanasundaram 博士，Patrick Matrin，Paul Romagna，Susan Snyder 和 Nicole Yankelovich 合作完成了本书中 5 个 non-ISO-Latin-1 文本字。Jim Arnold 在 smoog 和 moorge 的正确拼写、用法和词源方面提供了研究性的帮助。Ed Moony 帮助准备文档。Herb 和 Joy Kaiser 是我们的克罗地亚语顾问。Cookie Callahan，Robert E．Pierce 和 Rita Tavilla 为避免本项目中途夭折提供了必要的支持。

感谢 Kim Polese 为我们提供的有关 Java 对计算机用户和编程人员重要性的提纲挈领式的总结。

Susan Jones，Bob Sproull，Jinm Waldo 和 Ann Wollrath 在关键时刻提供了支持和建议。我们也感谢我们的家庭，感谢他们充满爱心的支持。当我们一天到晚沉湎于我们所钟爱的工作中时，家人便会拉我们出去松弛一下。

感谢 Peets 咖啡店里的人们，在那里我们谈论世界上最好的 Java。

我们为写作本书已做了最大努力。书中存在的所有缺点和错误，将完全由作者负责。

目　　录

第 1 章　Java 概 述

本章是对 Java 程序设计语言的一个概述,可以让你迅速开始编写 Java 程序。我们扼要介绍语言的重点,而不涉及细枝末节。后续各章将详细讨论 Java 特性。

1.1　初 步 认 识

Java 程序是由类(class)构建的。从类定义开始,你可以创建任意多的对象(object),在此通常称为那个类的实例(instance)。可以将类想象成一个工厂,它具有制造零件的蓝图和指令,对象则是工厂制造的零件。

类包含两种成员(member),叫做域(field)和方法(method)。域是属于类本身或类对象的数据;它们构成对象或类的状态(state)。方法是在域上进行运算从而操纵状态的语句(statement)集合。

在许多语言中,首先碰到的例子是打印“Hello, world”。下面是 Java 版的相应程序:

```
Class HelloWorld {
    public static void main(String[ ] args) {
        System .out .println( "Hello, world");
    }
}
```

用你熟知的文本编辑器把上述源程序存为文件。然后利用 Java 编译器把源程序编译为 Java 字节码(bytecode),即 Java 虚拟机的“机器语言”。

编辑源程序和编译的细节因系统而异,这里不作介绍。详细信息可参考系统手册。当运行该程序时,它显示:

```
Hello, world
```

现在已经有了一个能做点事的小 Java 程序,但它是什么意思呢?

上述程序声明了一个叫做 HelloWorld 的类,它只有一个叫做 main 的方法。类成员出现在类名后的花括号{}内。HelloWorld 只有一个方法,没有域。

main 方法仅有的一个参数是 String 对象的数组,它是以命令行运行程序时的程序变量。数组、字符串以及 args 对 main 方法的意义将在以后介绍。

main 方法声明为 void,因为它不返回任何值。它是 Java 中几个特殊的方法名之一:类的 main 方法如果像上文那样声明,当我们把类作为应用程序运行时它将得以执行。运行时,main 会创建对象,为表达式求值,调用其他方法,完成定义一个应用程序的行为所需要的其他一切事情。

在上面的例子中,main 包含一个语句,调用了类 System 的 out 方法。我们可以利用对象引用 + 圆点(.) + 方法名的方式调用方法。HelloWorld 利用 out 对象的 println 方法在标准输

出流上打印以换行符终止的字符串。

练习 1.1

在你的系统中输入、编译、运行 HelloWorld。

练习 1.2

试着改变 HelloWorld，看看会出什么错误。

1.2 变　　量

下一个例子打印斐波那契序列，这个无穷序列的前几项是：

1
1
2
3
5
8
13
21
34

斐波那契序列的前两项是 1，1，后续各项是它前面两项之和。斐波那契序列的打印程序非常简单，由此可以让大家知道如何声明变量（variable），写简单的循环，执行基本的算术运算。以下就是斐波那契程序：

```
class Fibonacci {
    /** Print out the Fibonacci sequence for values < 50 */
    public static void main(String[] args) {
        int lo = 1;
        int hi = 1;

        System.out.println(lo);
        while (hi < 50) {
            System.out.println(hi);
            hi = lo + hi;                    // new hi
            lo = hi - lo;                    /* new lo is (sum - old lo)
                                                i.e., the old hi */
            }
        }
    }
```

这个程序声明了像 HelloWorld 那样的一个类 Fibonacci,它具有 main 方法。main 的前两行声明了两个变量,hi 和 lo。每个变量必须具有一种位于它名字前的类型(type)。hi 和 lo 的类型是 int,32 位带符号整数,取值范围是 -2^{32} 到 $2^{32}-1$。

Java 的基本数据类型支持整型、浮点型、布尔型和字符型值。Java 可以直接理解这些基本类型的数据,这和程序员定义的对象类型不同。Java 没有缺省的类型;每个变量的类型必须显式定义。Java 的基本类型有:

boolean	true 或 false
char	16 位 Unicode 1 .1 字符
byte	8 位整数(带符号)
short	16 位整数(带符号)
int	32 位整数(带符号)
long	64 位整数(带符号)
float	32 位浮点数(IEEE 754-1985)
double	64 位浮点数(IEEE 754-1985)

在 Fibonacci 程序中,我们声明 hi 和 lo 初值为 1。初值是在变量声明时利用 = 运算符在初始化表达式中设定的。= 运算符将它左边的变量的值设定为它右边表达式的值。在这个程序中 hi 是序列中的最后一项,lo 则是前一项。

在初始化之前,变量未定义。如果在给变量赋值之前就使用它,Java 编译器拒绝编译,直到得以改正。

程序中的 while 语句提供了 Java 中的一种循环方式。计算 while 语句中的表达式值,如果为真,执行循环体再验算表达式值。当表达式值变为假时,while 循环结束。如果它永不为假,程序将不断运行,除非循环中有 break 语句或异常发生。

while 验算的表达式是值为真或假的布尔型表达式。上述布尔型表达式验算序列中当前最大值是否小于 50。如果最大值小于 50,则打印它的值并计算下一项。如果最大值等于或大于 50,控制交给 while 循环体后的第一条语句。在这个程序中就是 main 方法的结尾,所以程序结束。

注意 println 方法在上列的 Fibonacci 程序中接受一个整型变量,而在 HelloWorld 中接受一个字符串型变量。println 是许多重载的方法之一,经过重载方法可以接受不同类型的变量。

练习 1 .3

给打印序列增加一个标题。

练习 1 .4

编写一个生成另一个序列的程序,例如平方表(乘法由 * 执行,例如 i * i)。

1 .3 程序中的注释

注释夹杂在程序中。如实例所示,Java 有三种形式的注释。

从/ / 开始到行末的文本,/ * 和 */ 之间的文本,都会被编译器忽略。

注释允许程序员在程序外书写描述性的文本,为将来可能读这些程序的程序员作注释。将来的程序员有可能是若干年月之后的你自己。另外,你可以在书写注释时发现许多错误,因为解释程序的功能会迫使你仔细思考。

第三种注释出现在最上面,位于/ ** 和 */ 之间。以两个星号起头的注释叫做文档注释(简称 doc comment)。文档注释描述紧跟其后的声明部分。上面例子中的注释解释 main 方法。有一种叫作 javadoc 的工具可以把注释析取出来生成 HTML 文档。

1.4 有名常量

常量是指如 12,17,9,“Strings Like This”这样的一些值。常量是你指定的、不必运算的、在程序生命期内保持不变的值。

程序员有两个原因需要用到命名常量。一个原因是常量的名字是一种形式的文档。名字可以(而且应当)描述特定的值用于什么。另一个原因是在程序的一个位置定义常量的值,它只能在这个位置修改,方便了程序维护。在 Java 中创建命名常量的方法是把变量声明为 static 和 final,并在声明中给它赋初值:

```
class CircleStuff {
    static final double π = 3.1416;
}
```

如果五位精度不够,只在一个地方改变 π 的值即可。我们把 π 声明为双精度的 64 位浮点数。现在可以把 π 改为更精确的值,如 3.14159265358979323846。

可以将相关的常量组合在一个类中。举个例子,纸牌游戏可能需要这些常量:

```
class Suit {
    final static int CLUBS    = 1;
    final static int DIAMONDS = 2;
    final static int HEARTS   = 3;
    final static int SPADES   = 4;
}
```

这样声明后,程序中的牌就可以通过 Suit.HEARTS, Suit.SPADES 等访问,这样就将所有的牌名组合到一个类中。

1.4.1 Unicode 字符

稍稍注意一下,在上个例子中我们把符号 π 作为一个常量的名字。在绝大多数程序语言中,标识符仅限于 ASCII 字符集中的字母和数字。

Java 将你带入国际化的软件世界:以 Unicode 这种国际字符集标准来编写 Java 程序。Unicode 是 16 位的字符集,它的字符足够覆盖世界上使用的主要语言,因此 π 可以用作上例中的常量名。π 是 Unicode 希腊语区的一个合法字符,所以也是 Java 的有效字符。绝大部分 Java 程序用 ASCII(一种 7 位标准字符集),或者 ISO-Latin-1(一种 8 位标准字符集,通常称为