

刘彦明 主编

西安电子科技大学出版社



JAVA

JAVA

语言及其程序设计

TP312

L760

401619

JAVA 语言及其程序设计

刘彦明 刘立群
芮雨 赵俊 李鹏



西安电子科技大学出版社

1997

(陕)新登字 010 号

JS194/32
内 容 简 介

本书介绍了新一代计算机语言——Java。Java 语言是一种基于面向对象的网络编程语言，它特别适合于 Internet 网上的应用开发，同时具有常用计算机程序设计语言的特点。

本书从面向对象的基本概念开始，由浅入深、按部就班地介绍了 Java 语言的语言要素、编程方法以及一些常用的特殊功能，同时用较多的篇幅介绍了网络环境下的程序设计和用户界面编程。

本书不仅是一本 Java 语言的入门书，而且兼顾了具有一定 Java 语言基础的程序设计人员的需要，也是一本 Java 语言的提高书。

本书注重实用性，在书中给出了大量实例程序，使读者能够通过例子和上机实践，很快掌握 Java 语言。

本书适合于计算机和通信等相关专业的本科生、研究生、大学教师和工程技术人员使用，特别适合于从事网络和 Internet 开发的科技工作者使用。

JAVA 语言及其程序设计

刘彦明 主编

责任编辑 杨兵 云立实

西安电子科技大学出版社出版发行

空军电讯工程学院印刷厂印刷

陕西省新华书店发行 新华书店经售

开本 787×1092 1/16 印张 31 4/16 字数 741 千字

1997 年 3 月第 1 版 1997 年 3 月第 1 次印刷 印数 1—6 000

ISBN 7-5606-0508-7/TP·0243

定价：41.00 元

衷心感谢西安电子科技大学 ISN 国家重点实验室的朋友们

衷心感谢西安电子科技大学通信学院计算机通信教研室的朋友们

衷心感谢陕西海泰科技有限责任公司的大力支持

衷心感谢所有关心和支持我的朋友们

前 言

Java 语言是 Sun Microsystem 公司开发的新一代面向对象程序设计语言，特别适合于 Internet 网上的应用软件开发，因此有人也把 Java 语言称为新一代网络编程语言。Java 语言使得网络开发人员、WWW 应用开发人员可以设计具有高度交互能力和安全性的应用程序。

Java 的中文名字叫“爪哇”，实际上是印尼的一个小岛屿的名字，那里盛产咖啡。

Java 作为一种网络编程语言，在 Internet 网上是炙手可热的，大有“听取哇声一片”的气势。Java 作为软件开发的一种革命性技术，其地位已被确立，主要表现在以下几个方面：

(1) 计算机产业的许多大公司购买了 Java 的许可证，这一点说明 Java 已得到工业界的认可。

(2) 众多的软件开发商开始开发支持 Java 的软件产品。今天是以网络为中心的时代，不支持 Java 的应用是不敢想象的。

(3) Java 作为 Internet 和 Intranet 的主力开发语言，为信息产业提供了有力的支持。

因此，无论是做商业软件开发、企业信息系统，还是搞信息服务产业、科学研究，上 Internet，使用 Java 编程将是一个非常有前途的方向。可以这样说：Internet 是一个舞台，Java 语言是一种工具，企业家和科学家将在 Internet 舞台上，用 Java 导演一幕幕惊心动魄的故事。

鉴于此，我们把 Java 语言介绍给大家，以期帮助大家了解并掌握它，为我们的将来服务。

那末，为什么 Java 会如此成功？它有哪些特点呢？

有关 Java 语言的特点已有不少文章作了详细介绍，这里就不再重复了。这里给出几位科学家对 Java 的评价来说明它的成功所在。

Sun 公司副总裁、首席技术官员 Eric Schmidt：

“Java 的出现，正威胁着操作系统开发商。操作系统开发商不可能再通过“API 锁定”来把应用程序束定到特定的操作系统和 CPU 上，形成事实上的垄断，从而控制软硬件的销售。”

“Java 不仅是一种语言，而且是新的网络操作系统。Java 将会成为 90 年代的 DOS，虽然它不完善，但它会像以前的 DOS 一样无处不在。”

MIT 副教授 Danny Hills：

“DNA 代表着 Java 的产生。用 DNA 可以生产出原始微生物和病毒。多细胞生物是由许多互相通信的细胞组成的集合。Internet 已成为巨大的电子生态系统，在这个电子生态系统中生活的‘生物’是用 Java 编写的应用程序。”

Sun 公司研究副总裁 Bill Joy：

“Java 是一种广泛使用的网络编程语言，它是一种新的计算概念。Java 是一种软件虚拟机，这种虚拟机器是网络加软件。Java 不依赖于机器结构，是面向对象的，可重用性好，

并有并发机制。”

从以上的言论我们可以得出：Java 将成为新的应用程序开发环境的标准。为此，我们把 Java 推荐给大家，希望能起到抛砖引玉的作用。

本书分六篇介绍了 Java 语言：

开篇 进入 Java。主要介绍了 Java 语言的开发环境。通过“Hello World”application 和“Hello world”applet 说明了 Java 程序设计的一般过程，包括 Java 程序的组成、编译和执行的整个过程。

第一篇 Java 编程。主要介绍了 Java 语言的编程要素，包括第 2~10 章的内容。

第二篇 编写 applet。主要介绍了 Java 环境下如何编写 applet 应用程序，包括第 11~14 章的内容。

第三篇 创建用户界面。主要介绍了在 Java 环境下如何创建应用程序的界面，包括第 15~18 章的内容。

第四篇 网络应用和安全。主要介绍了在 Java 环境下如何开发网络应用程序和开发符合安全要求的 Java 应用，包括第 19~22 章的内容。

第五篇 将本地方法集成到 Java 程序中。主要介绍了如何在 Java 程序中调用 C 语言函数，即把 C 函数作为 Java 程序中的方法，包括第 23、24 章的内容。

在本书的编写过程中，西安电子科技大学 ISN 重点实验室的硕士研究生高勍、马昕、伦观涛、朱兴全、张凯、付延增等同志做了大量的资料收集和整理工作，在此表示衷心感谢！

由于作者水平有限，书中难免有不妥之处，敬请读者批评指正。

刘彦明

谨识于西安电子科技大学通信学院计算机通信教研室

1996 年 8 月 1 日

目 录

开篇 进入 Java

0.1 Java 开发工具 JDK	1	0.2 “Hello World”application	4
0.1.1 javac——Java 语言编译器	1	0.3 “Hello World”applet	7
0.1.2 java——Java 语言解释器	2	0.3.1 创建目录	7
0.1.3 javah	2	0.3.2 创建 Java 的 applet 源文件	7
0.1.4 javap——Java 类文件反汇编程序	3	0.3.3 编译源文件	8
0.1.5 javaprof——Java 剖析工具	3	0.3.4 创建包含 applet 的 HTML 文件	8
0.1.6 hotjava——WWW 浏览器	3	0.3.5 加载 HTML 文件	8

第一篇 Java 编程

第 1 章 面向对象编程的基本概念	11	2.5.2 分支语句(switch)	29
1.1 什么是对象	11	2.5.3 循环语句	32
1.2 什么是消息	13	2.5.4 异常处理语句	33
1.3 什么是类	14	2.5.5 跳转语句(Branching Statement)	33
1.4 什么是继承	15	2.6 数组和字符串	33
第 2 章 Java 语言的基本要素	17	2.6.1 数组	33
2.1 运行 application	17	2.6.2 字符串	34
2.2 变量和数据类型	18	2.7 Java 环境的一些特征	35
2.2.1 变量类型	18	2.7.1 main()方法	35
2.2.2 变量名	19	2.7.2 异常	36
2.2.3 作用域	20	2.7.3 标准输入、输出流	37
2.2.4 变量初始化	20	第 3 章 对象、类和接口	38
2.3 操作符	22	3.1 对象的生命周期	38
2.3.1 算术操作符	22	3.1.1 创建对象	38
2.3.2 关系和条件操作符	23	3.1.2 使用对象	40
2.3.3 位操作符	24	3.1.3 销毁对象	42
2.3.4 赋值操作符	25	3.2 创建类	43
2.4 表达式	26	3.2.1 类的声明	43
2.5 流程控制语句	27	3.2.2 类的主体	45
2.5.1 if - else 语句	28	3.2.3 声明成员变量	46

3.2.4 实现方法	48	6.2.1 标准输入流	92
3.2.5 控制对类成员的访问	56	6.2.2 标准输出和错误流	92
3.2.6 实例成员和类成员	61	6.3 系统特性	93
3.2.7 构造方法	64	6.3.1 读取系统特性	94
3.2.8 编写 finalize()方法	65	6.3.2 修改系统特性	94
3.3 子类、超类和继承	66	6.4 强制终止和垃圾收集(garbage collection)	95
3.3.1 创建子类	66	6.4.1 终止对象	96
3.3.2 编写终止类和方法	69	6.4.2 运行垃圾收集器	96
3.3.3 编写抽象类和方法	70	6.5 加载动态库	96
3.3.4 java.lang.Object 类	71	6.6 杂类系统方法	97
3.4 创建和使用接口	73	6.6.1 拷贝数组	97
3.4.1 接口的定义	73	6.6.2 获取当前时间	97
3.4.2 定义接口	73	6.6.3 退出运行时环境	99
3.4.3 实现接口	75	6.6.4 设置和获取安全管理器	99
3.4.4 将接口作为一种数据类型	75	6.7 运行时对象	99
第4章 String 和 StringBuffer 类	77	第7章 控制线程	101
4.1 为什么有两个 String 类	77	7.1 线程的定义	102
4.2 创建 String 和 StringBuffer	78	7.2 一个简单的线程例子	102
4.3 访问者方法	78	7.3 线程属性	104
4.4 修改 StringBuffer	80	7.3.1 线程体	104
4.4.1 插入	80	7.3.2 线程状态	107
4.4.2 设置字符	81	7.3.3 线程优先权	110
4.5 将对象转化为字符串	81	7.3.4 守护线程	115
4.5.1 toString()方法	81	7.3.5 线程组	115
4.5.2 valueOf()方法	81	7.4 多线程程序	120
4.6 将字符串转化为数字	81	7.4.1 同步线程	120
4.7 字符串和 Java 编译器	82	7.4.2 死锁和哲学家会餐	127
4.7.1 字符串	82	附录 模拟三种排序方法的 applet 程序源代码	129
4.7.2 联结和+操作符	82	第8章 输入和输出流	139
4.8 Java 字符串是第一级类的对象	82	8.1 Java 中与 I/O 的首次相遇	139
4.8.1 不可预测的 C 字符串	83	8.2 Java.io 的输入和输出流概述	140
4.8.2 Java 字符串是可预测的	83	8.2.1 简单的输入和输出流	141
第5章 设置程序属性	85	8.2.2 过滤流	141
5.1 特性	85	8.2.3 其它流	142
5.1.1 设置 Properties 对象	85	8.3 输入和输出流	142
5.1.2 获取特性信息	86	8.3.1 使用管道化的流	142
5.2 命令行参数	86	8.3.2 使用流读写文件	145
5.2.1 空格符分隔命令行参数	88	8.3.3 使用流读写存储区	146
5.2.2 命令行参数的惯例	88	8.3.4 使用流联结文件	146
5.2.3 解析命令行参数	89	8.4 过滤流	148
第6章 使用系统资源	90	8.4.1 使用过滤流	148
6.1 使用 System 类	91		
6.2 标准 I/O 流	91		

8.4.2 使用 DataInputStream 和 DataOutputStream	149	9.3.1 捕获	165
8.4.3 写自己的过滤流	150	9.3.2 声明	165
8.5 随机存取文件	154	9.3.3 运行时异常	165
8.5.1 使用随机存取文件	154	9.3.4 可在方法范围内被引出的异常	166
8.5.2 为 RandomAccessFile 和 DataInput/ DataOutput 写过滤器	155	9.4 处理异常	166
第 9 章 利用异常处理错误	158	9.4.1 例子	166
9.1 Java 异常概述	158	9.4.2 捕获和处理异常	167
9.1.1 把错误处理代码从“常规” 代码中分离出来	159	9.4.3 声明异常	174
9.1.2 把错误传播给调用堆栈	160	9.5 如何引出异常	175
9.1.3 按错误类型和错误差别分组	162	9.5.1 throws 语句	175
9.2 第一次遇到 Java 异常	163	9.5.2 Throwable 类及其子类	176
9.3 Java 的捕获和声明要求	165	9.5.3 创建自己的 Exception 类	177
		9.6 运行时异常(RuntimeException) ——争论	179

第二篇 编写 applet

第 10 章 applet 综述	183	10.7 在 HTML 页中加入 applet	201
10.1 applet 的生命周期	184	10.7.1 <APPLET>标记的最简形式	202
10.1.1 加载 applet	184	10.7.2 由 CODEBASE 指定 applet 的路径	202
10.1.2 离开和返回到 applet 页面	184	10.7.3 用<APPLET>标记指定参数	202
10.1.3 再次加载 applet	185	10.7.4 为不支持 Java 的浏览器提 供显示文本	203
10.1.4 退出浏览器	185	第 11 章 创建 applet 用户界面	205
10.2 重要事件的方法	185	11.1 创建一个 GUI	205
10.3 屏幕的绘制和事件处理的方法	186	11.1.1 一个 applet 是一个面板	205
10.4 使用 UI 部件	187	11.1.2 每个 applet 的尺寸是由用 户预定义的	205
10.4.1 预制 UI 部件的种类	187	11.1.3 applet 通过 getImage()方法 加载图像	206
10.4.2 在 applet 中使用 UI 部件的 方法	188	11.1.4 通过网络加载 Applet 类和 它们使用的文件	206
10.4.3 在 Simple 中增加一个不可 编辑的文本域	188	11.2 播放声音	206
10.5 applet 中的线程	189	11.2.1 播放声音的方法	206
10.5.1 概述	189	11.2.2 播放声音举例	207
10.5.2 利用线程处理重复性任务	191	11.3 定义和使用 applet 参数	212
10.5.3 利用线程执行一次性的初 始化	194	11.3.1 决定支持哪些参数	212
10.6 applet 的能力与限制	201	11.3.2 编写支持参数的代码	213
10.6.1 安全性限制	201		
10.6.2 applet 的功能	201		

11.3.3 给出参数信息	214	12.2 与浏览器通信	225
11.4 读系统参数	215	12.3 使用服务器一方的 application	229
11.5 显示短状态字符串	216	12.4 使用服务器克服 applet 的安全限制	236
第 12 章 与其它程序通信	218	第 13 章 完成 applet	254
12.1 给同一页上的其它 applet 发送消息	218	13.1 applet 的安全限制	254
12.1.1 概述	218	13.2 applet 的功能	254
12.1.2 通过名字来查找 applet—— getApplet()方法	219	13.2.1 applet 独具的功能	255
12.1.3 找到一页中所有的 applet—— getApplets()方法	223	13.2.2 更多的 applet 功能	255
		13.3 装载 applet 之前的工作	255
		13.4 完美的 applet	256

第三篇 创建用户界面

第 14 章 Java UI 概述	259	15.1.11 怎样使用 Panel	307
14.1 AWT 部件	259	15.1.12 怎样使用 Scrollbar	307
14.1.1 基本控制——按钮、检查盒、选项、 列表、菜单和文本域	266	15.1.13 怎样使用 TextArea 和 TextField	313
14.1.2 获得用户输入的其它方式——游 标、滚动条和文本区	267	15.2 部件结构的细节	315
14.1.3 创建定制部件——画布	267	15.2.1 如何创建同等物	316
14.1.4 标签	267	15.2.2 同等物如何处理事件	316
14.1.5 容器	267	15.3 常见的部件问题及其解决方法	316
14.2 其它 AWT 类	268	第 16 章 在容器中布置部件	318
14.3 基于 GUI 的程序的程序的分析	268	16.1 使用布局管理器	318
14.3.1 例程中的类	274	16.1.1 使用布局管理器的通用规则	318
14.3.2 部件层次	275	16.1.2 怎样使用 BorderLayout	319
14.3.3 绘制图形	277	16.1.3 怎样使用 CardLayout	325
14.3.4 事件处理	278	16.1.4 怎样使用 FlowLayout	328
第 15 章 使用部件——GUI 基础块	281	16.1.5 怎样使用 GridLayout	330
15.1 使用 AWT 部件	281	16.1.6 怎样使用 GridBagLayout	332
15.1.1 使用部件的通用规则	281	16.2 创建定制的布局管理器	338
15.1.2 怎样使用 Button	282	16.3 不用布局管理器	343
15.1.3 怎样使用 Canvas	284	第 17 章 使用图形	347
15.1.4 怎样使用 Checkbox	285	17.1 AWT 图形支持概述	347
15.1.5 怎样使用 Choice	287	17.2 使用图形原语	348
15.1.6 怎样使用 Dialog	289	17.2.1 绘制基本图元	348
15.1.7 怎样使用 Frame	293	17.2.2 使用文本	360
15.1.8 怎样使用 Label	295	17.3 使用图像	366
15.1.9 怎样使用 List	297	17.3.1 加载图像	367
15.1.10 怎样使用 Menu	301	17.3.2 显示图像	370
		17.3.3 处理图像	373

17.4 实现动画	381	17.4.4 移动图像穿过屏幕	400
17.4.1 创建动画循环	381	17.4.5 显示图像序列	401
17.4.2 动画图形	386	17.4.6 改善图像动画的外观与性能	405
17.4.3 消除闪烁	390		

第四篇 网络应用和安全

第 18 章 Java 网络功能入门	413	20.2 对 Socket 的读写	426
18.1 Java 网络功能	413	20.3 编写 Socket 的服务器方程序	429
18.2 从网络加载 applet	413	20.3.1 KnockKnockServer 类	429
18.3 从一个 URL 加载图像	413	20.3.2 KKProtocol 类	432
第 19 章 使用 URL	414	20.3.3 “Knock Knock”客户机	434
19.1 什么是 URL	414	20.3.4 运行程序	436
19.2 创建 URL	415	20.4 支持多客户	437
19.2.1 创建绝对 URL	415	第 21 章 数据报	440
19.2.2 创建一个相对 URL 对象	415	21.1 什么是一个数据报	440
19.2.3 其它 URL 构造函数	416	21.2 编写数据报客户和服务程序	441
19.2.4 MalformedURLException	416	21.2.1 QuoteServer 类	441
19.3 URL 的语法分析	417	21.2.2 QuoteServerThread 类	441
19.3.1 URL 的语法分析方法	417	21.2.3 QuoteClient 类	445
19.3.2 getRef()方法使用注意事项	418	21.2.4 运行服务器	446
19.4 直接读 URL	418	21.2.5 运行客户机	447
19.5 连接到 URL	419	21.2.6 安全性考虑	447
19.6 读写 URLConnection	420	第 22 章 提供自己的安全管理器	448
19.6.1 从 URLConnection 读	420	22.1 编写一个安全管理器	448
19.6.2 写 URLConnection	421	22.2 安装你的安全管理器	450
第 20 章 Socket	425	22.2.1 SecurityManagerTest	450
20.1 什么是 Socket	425	22.2.2 运行测试程序	451

第五篇 将本地方法集成到 Java 程序中

第 23 章 集成本地方法的步骤	455	23.3 第三步: 创建.h 文件	456
23.1 第一步: 编写 Java 代码	455	23.3.1 类结构	457
23.1.1 定义本地方法	455	23.3.2 函数定义	457
23.1.2 加载库	455	23.3.3 运行 javah	457
23.1.3 创建主程序	456	23.4 第四步: 创建 Stub 文件	458
23.2 第二步: 编译 Java 代码	456	23.5 第五步: 编写 C 函数	458

23.6	第六步：创建动态可加载库	459	24.3.3	传递引用数据类型	474
23.7	第七步：运行程序	460	24.3.4	警告	475
23.7.1	运行 Hello World 程序	460	24.4	从本地方法返回值	476
23.7.2	设置库路径	460	24.4.1	返回基本类型	476
23.7.3	改变库路径	461	24.4.2	返回引用类型	476
第 24 章	实现本地方法	462	24.4.3	通过参数返回数据	477
24.1	例子	462	24.5	在本地方法中使用 Java 对象	477
24.1.1	源文件	462	24.5.1	去除对句柄的引用	477
24.1.2	javah 生成的文件	469	24.5.2	存取对象的成员变量	477
24.1.3	指令步骤	471	24.5.3	从本地方法调用 Java 方法	478
24.2	方法标识和函数标识	472	24.5.4	调用 Java 构造函数	481
24.2.1	Java 一方	472	24.5.5	调用实例方法	481
24.2.2	本地语言一方	472	24.5.6	调用类方法	482
24.3	向本地方法传递数据	472	24.6	使用字符串	482
24.3.1	自动参数	473	24.7	本地方法和线程同步	483
24.3.2	传递基本数据类型	473	24.8	从本地方法引出异常	487

开篇 进入 Java

0.1 Java 开发工具 JDK

Sun 公司提供的 Java 开发工具 JDK 有 Windows 95 版、Windows NT 版和 Solaris 2.x 版，它们都可以从 java.sun.com 处免费获得，国内的 Sun 镜像服务器及各大网络节点的 FTP 服务器上也有。

Windows95 版的 JDK 由下述六种工具组成。

0.1.1 javac——Java 语言编译器

Java 是一种编译语言，与其它语言不同的是：它编译后产生一种字节代码，这种字节代码与机器代码类似，但又不针对于某一具体机器，它对所有变量和方法的引用都使用名字而不是地址。在任何机器上，Java 语言的语义都相同，如 int 总是 32 位，long 总是 64 位，这也是 Java 语言移植性好的主要原因。Java 语言编译器的编译格式为：

```
javac [选项]文件名.java
```

```
javac -g [选项]文件名.java
```

javac 将扩展名为 .java 的源程序编译成字节代码。传递给 javac 的文件中的每一个定义好的类，都产生各自的字节代码，并单独保存在各自的类名.class 文件中。javac 都把它们放在与 .java 文件相同的子目录下。如果在编译的过程中，javac 找不到出现的某个类的定义，则按 -classpath 选项提供的路径去查找。

javac -g 是 javac 的一个未优化版本，它适合于像 gdb 或 dbx 之类的调试器调试。

javac 中可使用的选项及含义如下：

-classpath <路径;路径;...>——定义 javac 去查找用户定义的某些类时的路径，各路径间以“;”分隔；

-d <目录;目录;...>——定义产生的 class 层次的根目录；

-g——产生调试信息，使字节代码包含行号和局部变量信息，以供调试用。这是缺省方式；

-ng——不产生调试信息，字节代码不能用于调试；

-nowarn——不产生警告信息；

-O——代码优化；

-verbose——使编译和链接程序打印出源文件及字节代码文件的信息。

0.1.2 java——Java 语言解释器

编译后的 Java 字节代码是无法在机器上直接运行的，需要由解释器对其解释执行。Java 语言解释器的解释格式为：

```
java      [选项] 类名 <参数>
```

```
java -g   [选项] 类名 <参数>
```

类名是要运行的类的名字，它必须包含所在的组件(package)的名字(例如 java.lang.String)，只有在未定义组件名，即类缺省地放在一个无名的组件内时，才可直接写类的名字。

类名后的参数是传递给类而不是传递给解释器的。

Java 对扩展名为.class 的字节代码解释执行。类名.class 文件中必须包含一个 main() 方法，执行即从 main() 开始。由于 main() 在一个类内定义，但是却不能由这个类的任一实例来调用，因此必须将其定义为静态的。如果在 main() 中或其它方法中创建了线程，则只有当 main() 及所有线程都结束后，程序才终止运行。否则，main() 退出后，程序即终止。

当在执行过程中，用到其他用户自己定义的类时，必须用 -classpath 选项说明它们的路径。当用户未指明时，系统将自动把系统类的路径加到其后部。

一般，我们都使用 javac 将 Java 的源程序编译成字节代码，然后用 java 对其解释执行。然而，若使用 -cs 选项，则用 java 即可对源程序编译并解释执行。每个类被加载后，它的修改数据与源文件中的修改数据相比较，如果源文件最近修改过，则需重新编译并加载新的字节代码。

解释器可以通过检查机制确定一个类是否合法，这就使得解释执行的字节代码不会破坏语言的一些约定。

除了上面提到过的 -cs 选项及 -classpath 选项外，一般用到的选项还有：

- ms x——设置内存分配池的大小为 x；
- prof——使 Java 解释器产生剖析数据，在当前目录下产生一个“java.prof”的文件；
- v——在每次加载一个类文件后，向标准输出打印信息；
- verbosegc——在每次自动回收无用单元后，向标准输出打印信息；
- verify——检查所有代码；
- verifyremote——对通过类加载器加载到系统的代码实行检查，这是缺省方式；
- noverify——关闭检查。

java -g 是 Java 的一个未优化版本，它还有一个专用的选项 -t，用以打印跟踪信息。

0.1.3 javah

jvah 实现从 Java 的类文件中产生 C 的头文件及 C 剩余文件，其格式为：

```
jvah      [选项] 类名
```

```
jvah -g   [选项] 类名
```

jvah 主要用于为实现类中本地的(native)方法而产生 C 的头文件及其它相关文件。其中头文件(.h)包含一个结构的定义，这个结构中的域与类中的实例变量相对应。这个头文件用于 C 程序中对类中变量的引用。

在省缺的情况下, javah 创建一个“CClassHeaders”子目录, 并将产生的头文件放在这个子目录下。如果使用了 - stubs 选项, javah 会创建一个“ stubs ”子目录并将剩余(stubs)文件放在这个子目录下。

除上面提到的 - stubs 选项外, 一般用到的选项还有:

- d <目录;目录;...> —— 覆盖缺省的存放头文件的目录“CClassHeaders”;
- td <目录> —— 覆盖缺省的存放临时文件的目录“\tmp”;
- verbose —— 向标准输出打印产生的文件的状态;
- classpath <路径;路径;...> —— 定义 javah 用以查找类文件的路径, 以“;”分隔。

javah - g 是 javah 的一个未优化版本, 它适合于像 gdb 或 dbx 之类的调试。

0.1.4 javap —— Java 类文件反汇编程序

javap 用以对类文件进行反汇编。它的输出依赖于所使用的选项, 若不使用任何选项, javap 仅打印出传递给它的类内的公共域及方法, javap 向标准输出打印信息。Java 类文件反汇编程序的格式为:

javap [选项] 类...

各选项含义如下:

- p —— 除打印公共的域及方法外, 亦打印出私有的和受保护的域及方法;
- c —— 打印出反汇编码;
- classpath <路径;路径;...> —— 定义 javap 用以查找类文件的路径, 以“;”分隔。

0.1.5 javaprof —— Java 剖析工具

javaprof 用于对由 Java 语言解释器产生的剖析数据格式输出。在用 Java 解释器解释执行.class 程序时, 若加 - prof 选项, 则产生一个 java. prof 文件, 它包含了 javaprof 用以进行大量统计的信息。javaprof 对其格式化后向标准输出打印, 包括三方面的信息:

- (1) 每个方法调用的次数和时间。
- (2) 每个类调用的次数和时间。
- (3) 每种数据类型所占的内存空间。

应注意, 在多线程 application 中, 剖析数据很不正确, 剖析只能用于单线程 application。Java 剖析工具的格式为:

javaprof [选项] java. prof

选项 - v 使 javaprof 打印一些额外的剖析信息。

0.1.6 hotjava —— WWW 浏览器

hotjava 允许在 WWW 网上浏览, 并且提供通过平台的现场交互功能, 交互内容以 applet 的形式存在。applet 是用 Java 语言编写的一类小程序, 它只能在支持 Java 的 WWW 浏览器上执行。

通过编写 applet 程序, 将其嵌入到 hotjava 的 Web 页中, 便可集声音、动画等多种技术于一体, 产生生动的页面。在翻阅页面的同时, applet 执行, 从而实现交互。hotjava 使网络功能进一步扩大, 充分体现了 Java 语言的强大功能。

0.2 “Hello World”application

如果你对编写独立运行的 application(能不依赖于任何浏览器而执行的 Java 程序)感兴趣,那么这里就是你开始的起点。遵循本节中的步骤,就能创建和使用独立运行的 Java application。

- 创建 Java 源程序

创建一个名为 HelloWorldApp.java 的文件,Java 代码是:

```
class HelloWorldApp {  
    public static void main (String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

- 编译源文件

用 Java 编译器编译源文件:

```
javac HelloWorldApp.java
```

如果编译成功,编译器则创建名为 HelloWorldApp.class 的文件。如果编译失败,则需确认一下你是否输入并命名了完全如上面所示的程序。

平台特定细节:

在 UNIX 下:

```
javac HelloWorldApp.java
```

在 DOS Shell(Windows 95/NT)下:

```
javac HelloWorldApp.java
```

- 运行 application HelloWorldApp.java

用 Java 解释器运行程序:

```
java HelloWorldApp
```

应该看到“Hello World!”显示在标准输出上。

平台特定细节:

在 UNIX 下:

```
java HelloWorldApp
```

在 DOS Shell(Windows 95/NT)下:

```
java HelloWorldApp
```

注意:Java 解释器的参数是类名,而不是文件名。

你已经看到了一个 Java application(也许甚至编译并运行了它),你可能想知道它是怎样工作的以及它与其它 Java application 在多大程度上类似。记住,Java application 是独立运行的 Java 程序(用 Java 语言写的),是不依赖于任何浏览器运行的程序。

一、定义类

在 Java 语言中,所有方法(函数)和变量都存在于类和对象(类的实例)中。Java 语言不

支持全局变量和杂散变量，因此任何 Java 程序的基干都是类定义。

在上面所示的代码中，class HelloWorldApp 开始了一个类定义块。类，作为面向对象语言(例如 Java)的基本结构单元，是描述与类的实例相关联的数据与行为的模板。当你实例化一个类时，就创建了一个对象，外观和感觉上很像同一个类的其它对象。与类或对象相关联的数据存储在变量中，与类或对象相关联的行为由方法执行。

在编译世界中，一个传统例子是表示矩形的类。这个类包含表示矩形的原点、宽度和高度的变量。一个矩形类的实例包含一个特殊的矩形(如这一页纸)的维数信息。

在 Java 语言中，类定义的最简单形式是：

```
class name {  
    . . .  
}
```

关键字 class 为名为 name 的类开始类定义的标志。这个类的变量和方法用开始和结束类定义块的花括号包含。“Hello World”application 没有变量，只有一个 main()方法。

为了获得更多有关面向对象的信息，请参见第 1 章“面向对象编程的基本概念”。为了学习有关用 Java 语言实现面向对象概念，请参见第 3 章“对象、类和接口”。

二、main()方法

每个 Java application 的首脑是 main()方法。当你用 Java 解释器运行一个 application 时，就确定了你想运行的类名。解释器执行在类中定义的 main()方法。main()方法控制程序流程，分配所需资源，运行其它任何为 application 提供功能的方法。

每个 Java application 必须包含一个 main()方法，其标识是这样的：

```
public static void main(String args[])
```

main()方法的方法标记包含三个修饰符：

- public 表明 main()方法能被任何对象调用。3.2.5 节“控制对类成员的访问”中讲述了 Java 语言支持的访问限制符 public、private、protected 和隐式 friendly 的来龙去脉。
- static 表示 main()方法是一个类方法。3.2.6 节“实例成员和类成员”讨论了类方法和变量。
- void 表示 main()方法不返回任何值。

1. 调用 main()方法

在 Java 语言中，main()方法与 C 语言和 C++语言中的 main()函数类似。执行 C 或 C++程序时，系统通过首先调用它的 main()函数来启动程序，然后 main()函数调用程序中的其它函数。类似地，当 Java 解释器执行 application(通过在 application 的控制类上请求)时，通过调用类的 main()方法开始。main()方法能调用所有其它方法。

如果试图在没有 main()方法的类上运行 Java application，解释器则会显示一条与之类似的错误信息并拒绝编译程序。提示信息如下：

```
In class classname: void main(String argv[]) is not defined
```

其中，classname 是你试图运行的类名。

2. 传递给 main()方法的参数

如下面的代码段所示，main()方法接收一个参数——String 类型数组：

```
public static void main(String args[])
```