



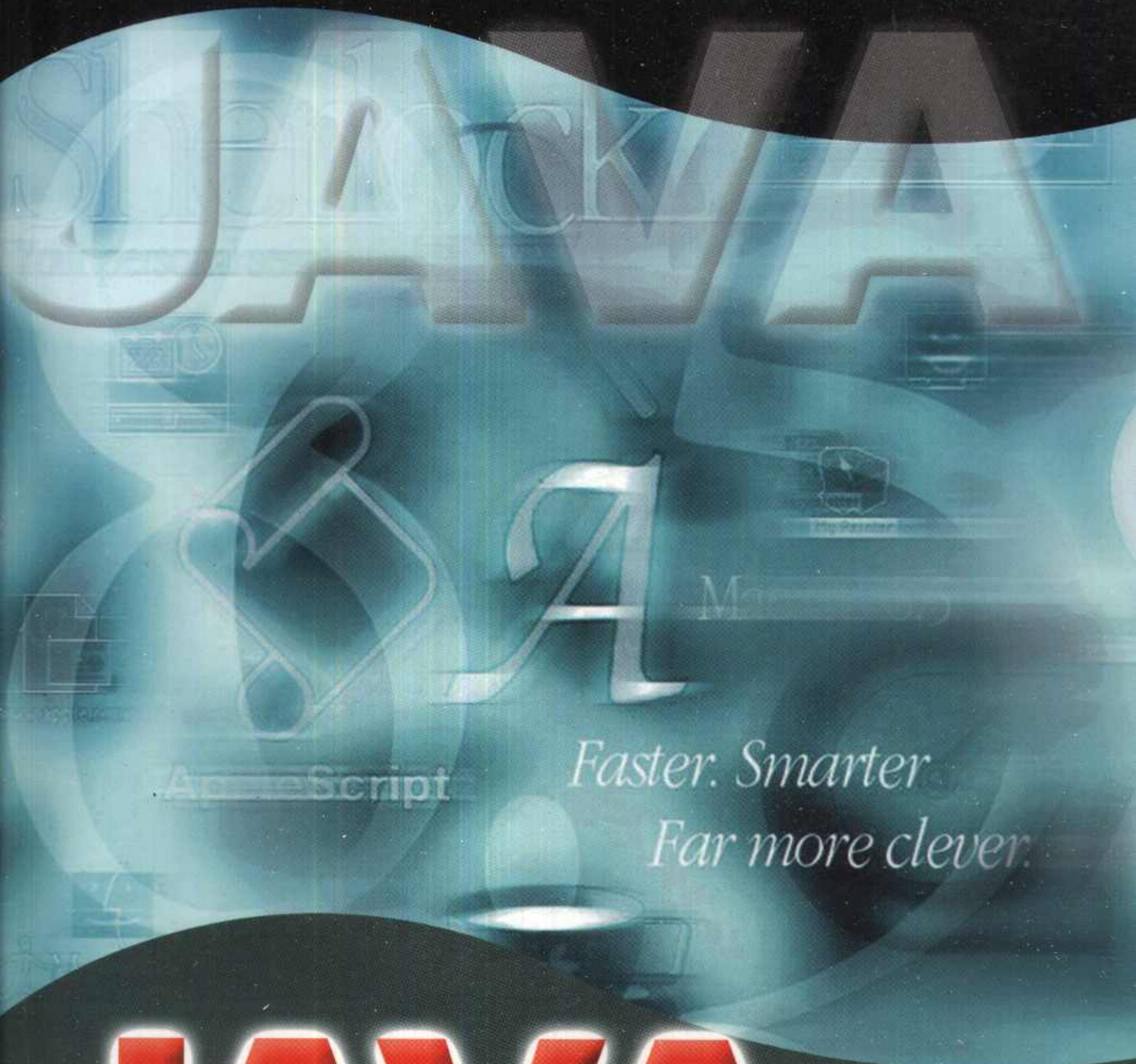
中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

第三波



内附光盘

柯温钊 编著



JAVA

例解教程

例解教程

01010101010101010101010101010101
10101010010101010
01010101010101010101010101

中国铁道出版社
2001年·北京

(京)新登字 063 号

北京市版权局著作权合同登记号: 01-2001- 0261 号

版 权 声 明

本书繁体字版由第三波资讯股份有限公司出版, 版权归第三波资讯股份有限公司所有。
本书简体字中文版由第三波资讯股份有限公司授权中国铁道出版社出版。专有出版权属中国铁道出版社所有, 未经本书原版出版者和本书出版者书面许可, 任何单位和个人不得以任何形式或任何手段复制或传播本书的一部分或全部。

图书在版编目(CIP)数据

Java 例解教程/柯温钊编著. —北京: 中国铁道出版社, 2001. 3

ISBN 7-113-04098-5

I. J… II. 柯… III. Java 语言-程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2001)第 08001 号

书 名: Java 例解教程

作 者: 柯温钊

出版发行: 中国铁道出版社(100054, 北京市宣武区右安门西街 8 号)

策划编辑: 苏 茜

特邀编辑: 鲁 凌

封面设计: 冯龙彬

印 刷: 北京彩桥印刷厂

开 本: 787×1092 1/16 印张: 28.25 字数: 684 千

版 本: 2001 年 3 月第 1 版 2001 年 6 月第 2 次印刷

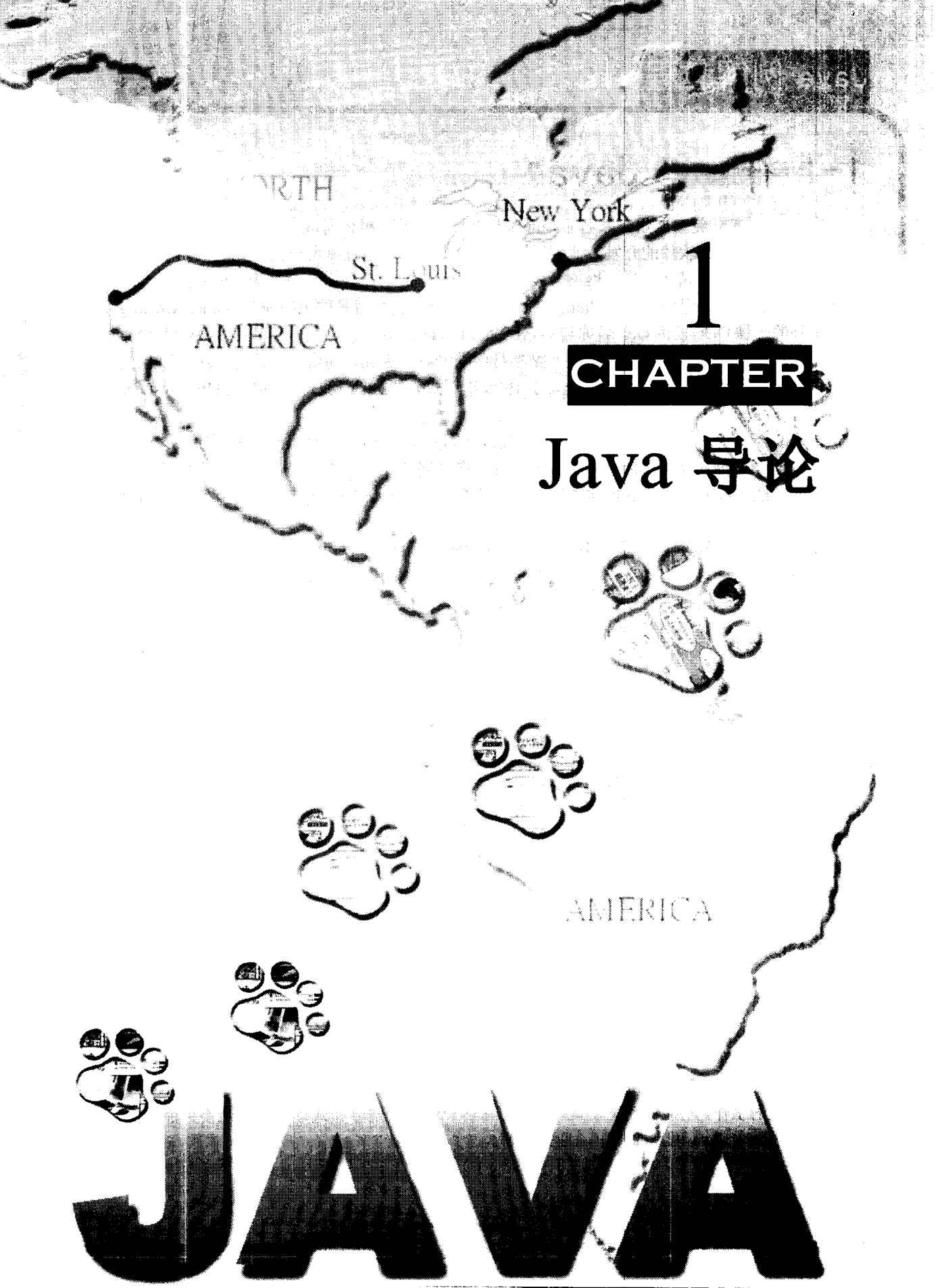
印 数: 5001~8000 册

书 号: ISBN 7-113-04098-5/TP·524

定 价: 47.00 元

版权所有 盗版必究

凡购买铁道版的图书, 如有缺页、倒页、脱页者, 请与本社计算机图书批销部调换。



1

CHAPTER

Java 导论

JAVA

1-1 什么是 Java?

在 1991 年初,美国加州一家名为 Sun Microsystems 计算机公司成立一个研发小组,负责研究消费性电子产品软件的开发。他们以 C++ 语言为蓝本,并且参考一些较先进的语言,开发出分布式,安全性高而且适合网络开发环境的语言。最初,该语言命名为 Oak(橡树),这个名称来自于研发小组的主持人 James Gosling,开发新语言目录的灵感源自 James Gosling 办公室外的一棵 Oak。但因 Oak 名称与其它产品雷同,最后将开发的新语言取名为 Java(爪哇)。据说,研发小组成员经常在公司附近一家咖啡厅喝咖啡,因此,将咖啡原产地 Java(爪哇)当作新语言的名称。Sun Microsystems 公司于 1995 年正式介绍 Java 及 HotJava(Sun Microsystems 公司所开发的一种浏览器)。

由于 Java 语言有许多性质与 C++ 语言类似,因此 Java 语言具有面向对象程序设计语言的特点。Java 语言具有跨平台、分布式的特性,在网络上,Java 语言提供直接信息网相互传递信息的功能。再加上实时多媒体的出现,更增添了 Java 语言的魅力,这些是其它语言所望尘莫及的。

Java 语言具有解释型语言(如 BASIC 语言)和编译型语言(如 C 语言)的特性。解释型语言结构简单,容易学习,但如果程序变大时,程序架构显得复杂,阅读起来难懂。而编译型语言具有结构化程序语言的特性,原始程序经编译后,生成可执行程序,执行速度较解释型语言快。Java 语言兼具两者的优点,具有执行效率高、程序的可移植性及安全性好的特点。

1-2 互联网

在第一次世界大战和第二次世界大战期间,以摧毁敌方的通讯系统为首要任务,因为在战争中,一旦通讯设施被摧毁,这个国家就陷入孤立无援的境地。有鉴于此,美国国防部积极提高通讯能力,期望在战争中,敌人切断任何一条通讯系统,整个通讯系统还能畅行无阻。同时,与加州大学合作,建立世界上第一个计算机网络。后来,其它大学为了研究,纷纷建立所谓学术网络,使得计算机网络能够蓬勃开发。再加上一些企业机构预见计算机网络存在无限商机,大量投入人力和物力,促使网络开发发展到今日的繁荣景象。

在 Java 语言尚未推出之前,互联网的开发遇到一些问题。在计算机网络开发之初,只有几台同类型的计算机联机,彼此沟通尚不成问题。可是,随着联机计算机种类越来越多,它们之间的沟通出现了问题,例如 Apple 计算机与 PC 使用不同的操作系统,彼此无法兼容。所以,不同种类的使用不同语言的计算机,必须以相同的语言才能沟通。因此,计算机网络中,每一部计算机必须遵守的规则,称为通信协议(Protocol)。目前广泛使用的通信协议为 TCP/IP(TCP 是 Transmission Control Protocol【通信控制协议】的缩写,IP 是 Internet Protocol【互联网协议】的缩写)。

另外,使用 HTML(HyperText Markedup Language 超文本标记语言)所设计出的网页,只能做浏览的工作,不能做动态交谈,而且想在客户端做一些分布式计算工作是不可能的事。但这些问题随着 Java 语言的推出,迎刃而解。

1-3 Java 与互联网

互联网能受大众的喜爱，与浏览器开发有密切的关系；就像计算机如此普及与操作系统有着密不可分的关系一样。自从微软推出 Windows 操作系统以后，人们只要靠移动鼠标，即可操作自如，连三岁小孩子都可以轻易上手。所以，才会得到大众的青睐，以至于 Windows 操作系统一版接着一版出笼，使得 Windows 红透半边天，至今仍方兴未艾，微软俨然成为计算机界的新霸主。

早期最有名的浏览器是 Netscape 通讯公司所开发的 Netscape Navigator 浏览器。由于 Netscape Navigator 浏览器广受一般上网大众的肯定，使得互连网遍及全世界每一个角落，无形中将人与人的距离拉近，也使得 Netscape 公司成为家喻户晓的通讯公司。微软也不甘示弱，在其 Windows 操作系统加入 Internet Explorer 浏览器，微软企图以生产 Windows 操作系统的优势，独占浏览器市场，结果微软被告。违反托拉斯法案的诉讼，目前仍继续进行。

为何使用者对 Windows 操作系统及窗口平台下的浏览器那么爱好？主要原因在于这些产品有亲和力，归功于可视化图标(Icon)控制，只要移动鼠标，按下图标即可上路。这种简易操作方式，当然为大众所接纳，这是与窗口有关的软件成功的地方。早期的 DOS，使用者必须熟记 DOS 指令，才能顺利操作计算机，这样的不便，造成初学者望而却步。

Java 的出现改善了过去互联网的一些缺点，例如动态文件、Web 的互动性、不同平台计算机之间的一致性，随着 Java 推出而迎刃而解。

下面介绍 Java 语言的特性：

一、简单易懂。

由于 Java 语言与 C++ 语言类似，只要学过 C 语言的人，很容易驾轻就熟。而 Java 语言本身撰写容易，语法简单，略有程序经验者，很快就能上手。

二、Java 执行环境与使用平台无关。

Java 执行环境由 Java 语言与 Java 虚拟机构成。登录互联网的计算机种类繁多，包括不同的 CPU 和操作系统。而 Java 语言编译器所生成的是一种中立性的 Bytecode(字节码)，此字节码可以在不同的平台上使用。换言之，Java 语言可以在不同平台上执行。

三、Java 语言提供分布式计算能力。

“**分布式计算(Distributed Computing)**”利用分布在各地的计算机分担计算工作。过去的观念是将数据处理及计算都集中到中央服务器主机上处理，这种集中计算模式下，随着网络的扩大，处理数据增加，造成服务器负担过大，因而会降低服务器工作效率。所以，九十年代，各大企业计算机网络提出分布式计算模式，将一些数据计算分散到远方客户端计算机处理。采用这种模式，可降低服务器的工作量，促使中央服务器主机处理及计算数据的效率提高，同时充分利用远方客户端计算机的计算能力。

由于 Java 语言的推出，使得过去不可能的任务(分布式计算)变得轻而易举。

四、Java 语言提供多线程(Thread)功能。

拥有多线程的程序可以充分利用系统资源，改善 CPU 闲置状态，缩短 CPU 闲置的时间。对同一时间显示多项应用程序提供较佳处理方式。

Java 语言支持多线程，因此，Java 语言设计的交互式应用程序有较快的执行速度。

五、Java 语言执行时具有编译程序和解释程序的优点。

Java 语言的原始程序代码由编译器编译成 Bytecode (字节码, 亦称为伪码), 而非机器码。此种伪码很容易在各种环境执行。再由 Java 解释器来执行字节码, 最后, 得到程序的结果。Java 语言的编译器为 javac, 解释器为 java。由编译器编译成字节码, 并将此字节码保存在扩展名为 class 的文件内, 再使用解释器 java 来执行字节码, 立刻可以得到结果。由于伪码很容易在各种环境执行, 适合各种不同的平台上使用。Java 原始程序代码受编译器编译, 是在 Java 虚拟机控制之下完成的, 能够防止计算机病毒或对程序的不当修改, 增加了 Java 语言的安全性。前面提过, 伪码适合在各种不同的平台上使用, 换言之, 伪码也适合于各种不同的 CPU 计算机, 因此, Java 语言程序可移植性高, 可以将 Java 程序移植到不同的操作系统上使用。

六、Java 语言的动态链接。

在微软 Windows 系统里, 如果多个应用软件同时使用相同的函数时, 计算机将加载这个函数到内存内, 然后, 这些应用软件共同使用内存内的函数。而每一个应用软件不需要在内存内复制相同的函数。这就是动态链接的概念。动态链接的优点是节省内存空间的使用。Java 语言的动态链接发生于程序执行时。Java 语言不会因为某一个类内容的更新, 而影响其它类必须重新编译。

1-4 Java 开发环境

现在介绍 Java 语言的使用工具。当您安装好 Java 软件时, 在 Java 目录内, 出现一些子目录及文件(见图 1-1), 并于 javabin 目录内, 可以看到所有 JDK(Java Developer's Kit)工具(见图 1-2)。

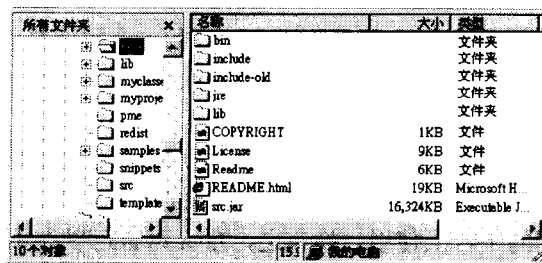


图 1-1

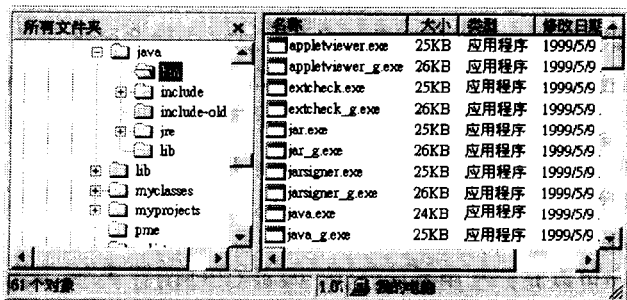


图 1-2

首先来看 Sun MicroSystem 所开发的 JDK 版本:

(一) JDK 1. 0—1995 年秋天发布。

(二) JDK 1. 1—1997 年春天发布。

(三) JDK 1. 2—1998 年 12 月发布。JDK 1. 2 版本就是我们一般所说的 Java 2。

据说, Sun MicroSystem 公司于 2000 年中秋前后推出 JDK 1. 3, 每次新版本的推出, 都将增添新类别以及对原有类别进行加强, 新版 JDK 内容改变颇多, 请拭目以待。

JDK 工具的内容:

一、javac: Java 语言的编译器。

这是 Java 语言非常重要的 JDK 工具, 将 Java 原始程序代码编译成字节码。另外, javac_g 也是一个编译器, 它是不进行优化的编译器, 通常, javac_g 用于 gdb 除错。

假设有一个 Java 原始程序代码 ko2_1. java, 现在使用 javac、javac_g 编译器编译 ko2_1. java, 编译方式如下:

```
javac <选择功能> ko2_1. java
```

及

```
javac_g <选择功能> ko2_1. java
```

其中<选择功能>可有可无, ko2_1. java 为原始程序代码。

下面来看一看<选择功能>:

路径	用来指定程序中类别的路径。其格式如下: .: <路径> 例如: .: /jbuilder/java/lib
-d	用来指定放置字节码的根目录。 例如: C: \jdk1. 2\bin>javac -d temp ko2_1. Java 按下 Enter 后, ko2_1. class 存放在 temp 子目录内, 请特别注意, temp 子目录与 bin 子目录都是属于 jdk1. 2 目录的子目录。否则, 将会出现 系统找不到指定路径 的错误信息。 如果-d 不使用时, 例如: C: \jdk1. 2\bin>javac ko2_1. Java 编译后, ko2_1. class 存放在 bin 子目录内。请读者注意这些差别。
-g	用于生成除错信息。
-nowarn	关闭编译警告信息。
-O	使程序最佳化。程序最佳化使执行程序效率增加, 速度加快, 但会阻碍除错功能或是使类别文件变大。
-verbose	用于生成有关编译器编译的输出信息。
-deprecation	用于反对使用 API 的输出资源位置。

二、java: Java 语言的解释器。

指令 java 用来执行 javac 编译成字节码。java_g 也是一个解释器, 它是不进行优化的解释器, 通常, java_g 用于 gdb 除错。

解释器指令 java 的用法如下:

```
java <选择功能> ko2_1
```

及

```
java_g <选择功能> ko2_1
```

其中<选择功能>请参考编译器 javac 的说明。

三、jdb: Java 语言的除错器。

jdb (jdb_g)是一个除错器，用于对 java 解释的类进行除错。

Jdb 的用法如下：

jdb <选择功能> <类名> <参数>

下面来看一看<选择功能>：

help	显示 help 信息并退出
version	显示建立的版本信息并退出
host <hostname>	依附解释器的主机
password <psswd>	依附解释器的密码
dbgtrace	显示除错信息

<类名>是指起始除错的类的名称。

<参数>是指传递给<类名> main()方法的参数。

例如：

D: \jtemp>jdb -dbgtrace ko2_1

按下 Enter 后，出现下列除错信息：

Initializing jdb...

[debugger: starting child: D: \JBUILDER\JAVA\JRE\bin\java -Xdebug -Djava.compiler=NONE

-Xbootclasspath/a: D: \JBUILDER\JAVA\lib\tools.jar sun.tools.agent.EmptyApp]

[debugger: password returned: i46k4]

[debugger: connected]

[debugger: loading classes ...]

[debugger: reading 174 classes...]

0xb1: class(ko2_1)

>_ ←jdb 提示符号

在 jdb 提示符下，可键入下面的指令：

- ★ threads [threadgroup]: 列出线程。
- ★ thread <thread id>: 设定预设线程。
- ★ suspend [thread id(s)]: 暂停线程。
- ★ resume [thread id(s)]: 恢复线程。
- ★ where [thread id] | all: 清除线程堆栈。
- ★ wherei [thread id] | all: 清除线程堆栈及计算机信息。
- ★ threadgroups: 列出线程群组。
- ★ threadgroup <name>: 设定目前线程群组。
- ★ print <id> [id(s)]: 显示对象或数据域。
- ★ dump <id> [id(s)]: 显示所有对象信息。
- ★ locals: 列出目前堆栈所有局部变量(local variables)。

- ★ classes: 列出目前已知的类。
- ★ methods <class id>: 列出类的方法。
- ★ stop in <class id>. <method>[(argument_type, ...)] : 在方法中设定断点。
- ★ stop at <class id>: <line>: 在行中设定断点。
- ★ up [n frames]: 上移线程堆栈。
- ★ down [n frames]: 下移线程堆栈。
- ★ clear <class id>. <method>[(argument_type, ...)] : 清除方法中的断点。
- ★ clear <class id>: <line>清除行中的断点。
- ★ step: 执行目前的行。
- ★ step up: 执行到当前方法返回。
- ★ stepi: 执行目前的指令。
- ★ next: 下一行。
- ★ cont: 从断点处继续执行。
- ★ catch <class id>: 中止指定异常。
- ★ ignore <class id>: 忽略指定异常。
- ★ list [line number|method]: 显示原始代码。
- ★ use [source file path]: 显示或改变原始路径。
- ★ memory: 说明内存的用法。
- ★ gc: 释放没有用的对象。
- ★ load classname: 加载出错的 Java 类。
- ★ run <class> [args]: 加载的 Java 类开始执行。
- ★ !! : 重复最后指令。
- ★ help (or ?): 列出指令。
- ★ exit (or quit): 离开 jdb 系统, 回到 DOS。

例如:

```
D: \jtemp>jdb ←键入 jdb
Initializing jdb. . .
> run ko2_1 ←键入 run ko2_1
欢迎进入 Java 世界!!running . . .
main[1]

Current thread "main" died. Execution continuing. . .
>
The application exited
```

四、appletviewer: Java Applet 的浏览器。

前面介绍 javac 编译 Java 程序代码成为字节码, 然后, 使用 java 执行字节码类, 这种程序称为 Java Application 程序。如果在 javac 编译 Java 程序代码成为字节码后, 再建立一个 HTML 文档, 使用 appletviewer(或 appletviewer_g)执行此 HTML 文档, 这种程序称为 Java Applet 程序。请参考第 2 章 Java Application 程序与 Java Applet 程序的说明。

五、javadoc: Java API 文件管理器。

javadoc(javadoc_g)会使用 Java 程序代码生成一组 HTML 格式的相关文件。例如:

D: \jtemp>javadoc ko2_1. Java

按下 Enter 后, 生成如下的信息:

```

Loading source file ko2_1. java. . .
Constructing Javadoc information. . .
Building tree for all the packages and classes. . .
Building index for all the packages and classes. . .
Generating overview-tree. html. . . ←生成文档
Generating index-all. html. . . ←生成文档
Generating deprecated-list. html. . . ←生成文档
Building index for all classes. . .
Generating allclasses-frame. html. . . ←生成文档
Generating index. html. . . ←生成文档
Generating packages. html. . . ←生成文档
Generating ko2_1. html. . . ←生成文档
Generating serialized-form. html. . . ←生成文档
Generating package-list. . . ←生成文档
Generating help-doc. html. . . ←生成文档
Generating stylesheet. css. . . ←生成文档

```

六、javah: C/C++头文件生成器。

javah(javah_g)用来生成 C/C++头文件。javah(javah_g)用法如下:

javah [options] <classes>

下面介绍[options]包括的指令:

help	显示 javah 的 help 信息。
classpath <path>	加载类的路径。
d <dir>	输出目录。
o <file>	输出文档。
jni	建立 JNI 样式文件。
old	建立 JDK1.0 样式文件。
stubs	建立定义文档。
version	显示版本信息。
verbose	显示输出信息。
force	写入输出文档。

七、javap: Java 反汇编器。

javap(javap_g)用来反汇编 Java 文件。其用法如下:

javap <options> <classes> . . .

下面介绍<options>包括的指令:

c	反汇编码。
classpath <pathlist>	搜寻用户的类别文档。
help	显示 javap 用法信息。
J<flag>	直接传递<flag>到执行系统。
l	显示行号及局部变量表。
public	显示 public 类与成员。
protected	显示 protected/public 类与成员。
package	显示 package/protected/public 类与成员。
private	显示所有类与成员。
s	显示内部类型识别标志。
verbose	显示堆栈大小, 方法的参数数量。

例如:

D: \jtemp>javap -c ko2_1

按下 **Enter** 后, 出现下列信息:

Compiled from ko2_1. java

```
public class ko2_1 extends java.lang.Object {
    public ko2_1();
    public static void main(java.lang.String[] );
}
```

Method ko2_1()

0 aload_0

1 invokespecial #6 <Method java.lang.Object()>

4 return

Method void main(java.lang.String[])

0 getstatic #7 <Field java.io.PrintStream out>

3 ldc #1 <String "欢迎进入 Java 世界!!">

5 invokevirtual #8 <Method void println(java.lang.String)>

8 return

1-5 JDK 的安装及使用

首先取得有关 Java 语言的安装软件, 如: IBM Java 软件、Borland Jbuilder 3.0 软件以及 Sun Microsystems JDK1.2 等软件, 安装完成后, 在 java 目录内可以找到 JDK1.2 编译程序的执行文件。

假如手边没有这些软件, 可以进入 Internet, 到网址为 <http://java.sun.com/products/jdk/1.2/>

下载 JDK1. 2 软件。

若安装完成后，最好在 Autoexec. Bat 文件内设定 JDK 路径。否则必须将 Java 原始程序代码放置于与 Java 编译器相同的目录内，不然，无法顺利执行 Java 原始程序代码。

如果您的 JDK1. 2 软件安装于 C 磁盘驱动器内，请在 Autoexec. Bat 文件内加入 `PATH=C: \java\bin;`。

倘若您使用 Borland Jbuilder 3. 0 软件，且将软件安装于 D 磁盘驱动器内，请在 Autoexec. bat 内加入 `PATH=D: \jbuilder\java\ bin;`。

设定 JDK 路径完毕后，任何地方皆可执行 Java 原始程序代码。至于在何处撰写程序，如何执行程序，第 2 章将有详细的说明。

2

CHAPTER

Java 程序简介

JAVA

一般 Java 语言原始程序代码的编写都是利用文本编辑软件完成,例如: Word、附属应用程序的 WordPad 及记事本,或是 DOS 下的 Edit. com 编辑器等,都可以用来编写 Java 程序代码。Java 原始程序代码由解释器(Interpreter) 译成 Java Byte-Code,就可以在浏览器上执行。

图 2-1 是利用记事本编写 Java 程序代码的例子。

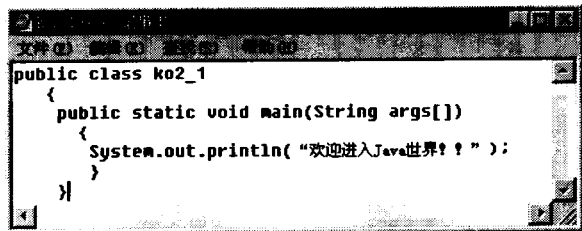


图 2-1

2-1 Java 程序种类

Java 程序分为两种: **Java Application** 和 **Java Applet**。Java Application 必须在所处的操作系统下执行,而 Java Applet 必须与**超文本标记语言**(Hypertext Markup Language, 简称 HTML) 配合使用,并在 WWW 上执行程序。

2-2 Java Application

Java Application 的程序可以在 MS-DOS、UNIX 等操作系统下执行。下列的程序是属于 Java Application 类的程序架构:

```
public class ko2_1
{
    public static void main(String args[ ])
    {
        System. out. println("欢迎进入 Java 世界!!");
    }
}
```

Java Application 程序架构

现在以 ko2_1 程序说明 Java Application 程序架构:

★ public class ko2_1 :

我们声明名称为“ko2_1”的 class(类)。public class ko2_1 将“ko2_1”这个 class 属性声明为 public。一般而言,Java Application 程序架构,开始声明 class 类时,public 属性可加可不加。例如:

public class ko2_1 或 class ko2_1

基本上,上述两种声明方式都会被接受,所不同的是使用“`public class ko2_1`”声明,存盘时文件名称必须与“`ko2_1`”名称一致(存盘文件名称为 `ko2_1.java`);倘若名称不一致,执行程序时,将出现下列错误的信息:

```
ko2_1a.java: 1: Public class ko2_1 must be defined in a file called "ko2_1.java"
```

而用“`class ko2_1`”声明,存盘时文件名称不一定要与“`ko2_1`”名称一致,其声明方式较有弹性。

★ { :

上述“`public class ko2_1`”声明中,紧接着一个大引号“{“,代表“`ko2_1`”类由此开始,最后在“}”处结束。

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

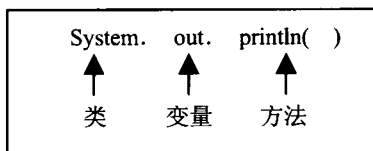
这是 java 程序的主函数。`main()`函数与 C、C++语言主函数 `main()`的用法一样,当程序执行时,解释器会寻找 `main()`,开始程序执行,若找不着 `main()`,解释器会显示错误信息。`Public`表示 `main()`函数可以被其它对象直接调用和使用。`static`用于将 `main()`函数声明为 `static`(静态)。`void`表示 `main()`函数不会传回任何数值、字符或字符串。

`args[]`用来传递命令行的自变量,同时,`args[]`也用来存储字符串。虽然,在这个程序没有用这个参数,不能将此参数删除,否则程序执行时,将会出现下列错误信息:

```
Exception in thread "main" java. Lang. NoSuchMethodError: main
```

★ `System. out. println("欢迎进入 Java 世界!!")` :

这一行语句的功能是将“欢迎进入 Java 世界!!”显示到屏幕上。下列是 `System. out. println()`的介绍:



`System. out. println()`表示在 `System` 类中有一个 `out` 的变量,`println()`是 `out` 变量的一个方法。

值得一提的是 `println()`方法: `println()`用来显示“欢迎进入 Java 世界!!”字符串。也有另一种与 `println()`类似的方法: `print()`。`print()`的用法与 `println()`大同小异,所不同的是 `println()`表示若有两个输出,它们会显示在不同的行上;而 `print()`则显示在同一行上。例如:

```
System. out. println("欢迎进入 Java 世界!!");
```

```
System. out. println("开始 Java 世界!!");
```

执行结果为:

```
欢迎进入 Java 世界!!
```

```
开始 Java 世界!!
```

如果:

```
System. out. print("欢迎进入 Java 世界!!");
```

```
System.out.print("开始 Java 世界!!");
```

执行结果为:

欢迎进入 Java 世界!! 开始 Java 世界!!

Java Application 程序的执行

前面已经介绍过 Java Application 程序的编写,程序编写完成后,必须以 java 扩展名存盘,例如:以 ko2_1.java 文件名存盘。

接着按下“开始”按钮,选取“程序”中的“MS-DOS 模式”,进入 MS-DOS 模式,如图 2-2 所示。

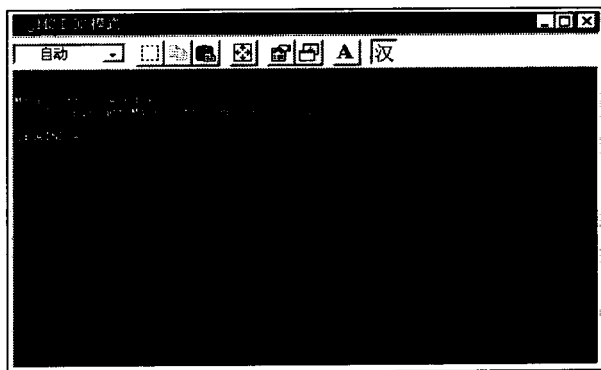


图 2-2

记住!要将图 2-2 的目录转换到存储 Java Application 程序的目录内,例如:Java Application 程序存放于 D 盘的 jtemp 目录内。转换目录操作方法如下。

Microsoft(R) Windows 98

(C)Copyright Microsoft Corp 1981-1998.

C: \WINDOWS>D: ☒ (表示按下 Enter 键)

D: \>cd jtemp ☒

D: \jtemp>_

接着开始编辑程序,键入

D: \jtemp>javac ko2_1.java ☒

如果程序没有错误,屏幕出现 D: \jtemp>_。而 javac 将 ko2_1.java 原始程序编译成字节码(bytecode),此时,字节码的文件名称为 ko2_1.class。ko2_1.class 为一种类别码,它能被解译程序 java 执行。

最后,执行 Java Application 程序,其操作方法如下:

D: \jtemp>java ko2_1 ☒

按下 Enter 键后,立刻出现执行的结果“欢迎进入 Java 世界!!”。

图 2-3 为在 MS-DOS 模式下的操作过程。