21世纪 高等院校计算机系列教材

Java语言程序设计实验指导

杨 昭 孙 友 等编著

中国水利水电出版社
www.waterpub.com.cn

21世纪高等院校计算机系列教材

Java语言程序设计实验指导

杨昭　孙友　等编著

中国水利水电出版社

内 容 提 要

本书为《Java 语言程序设计教程》（雷学生主编）的配套教程，主要用于上机实验指导。全书共包括 11 个实验，涉及到 Java 语言程序设计最主要的内容。主要包括：Java 语言基础实验、数组与控制语句、类与面向对象程序设计、继承、多态与重载、包与接口的使用、异常处理、多线程编程、网络编程以及选做实验 JSP 编程等。

在本书附录部分，给出了 Java 的参考编程规范和 HTML 语法参考。本书主要适合于 Java 2 的初学者和具有一定 Java 基础的程序开发人员。

图书在版编目(CIP)数据

Java 语言程序设计实验指导 / 杨昭等编著. —北京：中国水利水电出版社，2004

（21 世纪高等院校计算机系列教材）

ISBN 7-5084-2362-3

I. J… II. 杨… III. JAVA 语言—程序设计—高等学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字（2004）第 092549 号

书　　名	Java 语言程序设计实验指导
作　　者	杨昭　孙友　等编著
出版　发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）63202266（总机）　68331835（营销中心）　82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787mm×1092mm　16 开本　11.25 印张　253 千字
版　　次	2004 年 9 月第 1 版　2004 年 9 月第 1 次印刷
印　　数	0001—5000 册
定　　价	16.00 元

前　言

在人类社会已经迈入21世纪的今天，信息技术的飞速发展和普及使得全社会对计算机应用技术的需求日益增强。时至今日，计算机已经深入到人们日常工作、学习、生活、娱乐的各个方面，而互联网络则是连接千千万万计算机的桥梁。在这种情形下，一种使用简单且功能强大的网络编程语言必不可少，而Java正是这样的一种语言。

自从1995年被正式推出后，Java语言就以其独特的优势迅猛发展，经过短短的七八年时间，成为迄今为止最为优秀的面向对象语言。Java也从当初的一种语言而逐渐形成一种产业，基于Java语言的J2EE架构已经成为微软.NET平台的强大竞争对手。环球信息网WWW的创始人Berners-Lee曾经说过：计算机产业发展的下一个浪潮就是Java，并且将会很快发生。现在看来，这一预言已成为不争的事实。比尔·盖茨也不无感慨地说过：Java是长时间以来最卓越的程序设计语言，并确定微软整个软件开发的战略从PC单机时代向着以网络为中心的计算时代转移，而毫不犹豫地购买Java使用权是其重大战略决策的实施部署。

Java最大的特点是跨平台性，它将是未来网络世界中的“世界语”。有人预言，今后所有用其他语言编写的软件统统都要用Java语言来改写。此外，Java语言还具有简单性、面向对象性、分布性、动态性、鲁棒性、多线程性、可移植性等诸多特点。目前Java产业在国内正如火如荼地发展着，基于Java的高级技术（如J2ME、J2EE、Java Web服务等）也开始大量进入实际应用领域。为了满足广大非计算机专业读者学习Java语言的需要，我们编写了这套介绍Java语言编程基础知识的图书。

本书为《Java语言程序设计教程》（雷学生主编）的配套教程，主要用于上机实验指导。全书共包括11个实验，涉及到Java编程的基本方法、数组与控制语句、类与面向对象程序设计、继承、多态与重载、包与接口、异常处理、多线程编程、网络编程以及选做实验JSP编程等内容。在附录部分，给出了Java的参考编程规范和HTML语法参考。

本书由杨昭主编，参加编写工作的还有孙友、杜维兴、黄立超、张晋宝、黄卓、童剑、郭强等。由于时间紧迫，加之作者水平有限，书中的缺漏与错误在所难免，恳请广大读者批评指正。如果您有任何批评、意见或建议，敬请访问：http://www.izozi.com/~java/。

编者

2004年5月

目　　录

前言
实验 1　第一个 Java 程序 1
1.1　实验目的与要求 1
1.2　实验准备 1
1.2.1　Java 运行环境的安装与配置 1
1.2.2　Java 环境的有关工具 4
1.3　实验内容 9
1.3.1　第一个 Java 程序 10
1.3.2　Hello World!程序的 Applet 实现 10
1.3.3　关于 Java 程序的编写工具 12
1.4　补充练习 12
实验 2　Java 语言基础实验 13
2.1　实验目的与要求 13
2.2　内容概要 13
2.2.1　Java 标识符与关键字 13
2.2.2　变量、字面量、分隔符和注释 14
2.2.3　Java 基本数据类型 15
2.2.4　类型转换 17
2.2.5　Java 运算符 18
2.3　实验内容 21
2.3.1　输出 Unicode 字符 21
2.3.2　用单条语句输出多行文本 23
2.3.3　逻辑运算符的使用 24
2.4　补充练习 25
实验 3　数组与控制语句 28
3.1　实验目的与要求 28
3.2　内容概要 28
3.2.1　关于数组 28
3.2.2　流程控制语句 29
3.3　实验内容 31
3.3.1　创建并显示三维数组 31
3.3.2　练习 do-while 循环语句 32

3.3.3 练习跳转控制语句 34
3.4 补充练习 35
实验 4 类与面向对象程序设计 37
4.1 实验目的与要求 37
4.2 内容概要 37
4.2.1 类的一般格式 37
4.2.2 对象的声明 38
4.2.3 嵌套类与内部类 39
4.2.4 类的方法 39
4.2.5 参数传递 40
4.2.6 Java 中的访问控制 40
4.3 实验内容 41
4.3.1 创建一个 TimeClass 类 41
4.3.2 数组的对象化描述 45
4.3.3 Java 中的作用域 47
4.4 补充练习 48
实验 5 继承、多态与重载 50
5.1 实验目的与要求 50
5.2 内容概要 50
5.2.1 继承机制 50
5.2.2 多态与重载 52
5.2.3 方法的动态调用 53
5.3 实验内容 53
5.3.1 类的继承 53
5.3.2 练习使用 Super 关键字 55
5.3.3 创建多层次的类 57
5.3.4 实现方法的重载 60
5.4 补充练习 61
实验 6 Java 包的使用 65
6.1 实验目的与要求 65
6.2 内容概要 65
6.2.1 为什么要使用包 65
6.2.2 包的基本创建步骤 65
6.2.3 包的使用 68
6.2.4 包的成员 72
6.2.5 访问保护问题 73
6.3 实验内容 74

6.3.1 使用包的一个简单例子 74
6.3.2 访问多个包及其类 75
6.4 补充练习 77
实验 7 接口的使用 80
7.1 实验目的与要求 80
7.2 内容概要 80
7.2.1 抽象类与抽象方法 80
7.2.2 接口的相关知识 82
7.3 实验内容 86
7.3.1 使用抽象类和抽象方法 86
7.3.2 接口的使用实例 91
7.3.3 综合使用包和接口 94
7.4 补充练习 96
实验 8 异常处理 99
8.1 实验目的与要求 99
8.2 内容概要 99
8.3 实验内容 105
8.4 补充练习 109
实验 9 多线程编程 113
9.1 实验目的与要求 113
9.2 内容概要 113
9.3 实验内容 119
9.4 补充练习 130
实验 10 网络编程基础 133
10.1 实验目的与要求 133
10.2 内容概要 133
10.3 实验内容 141
10.4 补充练习 148
实验 11 JSP 编程初步* 150
11.1 实验目的与要求 150
11.2 实验内容与指导 150
11.3 补充练习 159
附录 A Java 参考编程规范 162
附录 B HTML 语法参考 170

实验 1　第一个 Java 程序

1.1　实验目的与要求

1．掌握 Java 运行环境的安装和配置。
2．了解 JDK 的各种类型与版本变化。
3．了解 Java 环境所提供的有关工具。
4．练习编写并运行一个简单的 Java 程序。

1.2　实验准备

由于 Java 是采用 Java 虚拟机进行解释执行的编程语言，它需要一定的软件支撑环境才能运行。本节首先介绍 Java 运行环境的安装与配置，然后附加介绍一些常用的 Java 环境工具。

1.2.1　Java 运行环境的安装与配置

编写并运行 Java 程序需要 Java 开发工具包（JDK，Java Development Kit）的支持。因此在编写自己的第一个 Java 程序前，读者需要首先在自己的机器上安装 JDK。到目前为止，JDK 的发展经历了 JDK 1.0、JDK 1.1、JDK 1.2、JDK 1.3、JDK 1.4 等几个版本。其中，从 JDK 1.2 开始，又将其分为以下几种类型的版本：

- J2EE（Java 2 Enterprise Edition）
- J2SE（Java 2 Standard Edition）
- J2ME（Java 2 Micro Edition）

从本质原理上来看，Java 2 的这三种版本并没有特别大的区别，主要在于工具包和类库的规模大小。对于这三种版本的简要比较如图 1-1 所示。其中，J2EE 用于企业级的 Java 应用环境，而 KVM 和 CVM 都是 JVM 的子集，它们均可被看作是 J2SE JVM 的精简版本，并特定用于 J2ME。

J2SDK 目前较新的版本是 V1.4.2，本书中所涉及的 Java 程序代码都是基于此版本的。鉴于目前大多数读者使用的是 Windows 操作系统，下面就介绍一下在 Windows 操作系统中如何安装和配置 Java 运行环境。

首先，下载 JDK 的安装程序。例如，这里我们使用的是 j2sdk-1.4.2-01-windows-i586.exe 安装文件。双击此可执行文件，将出现如图 1-2 中所示的界面。

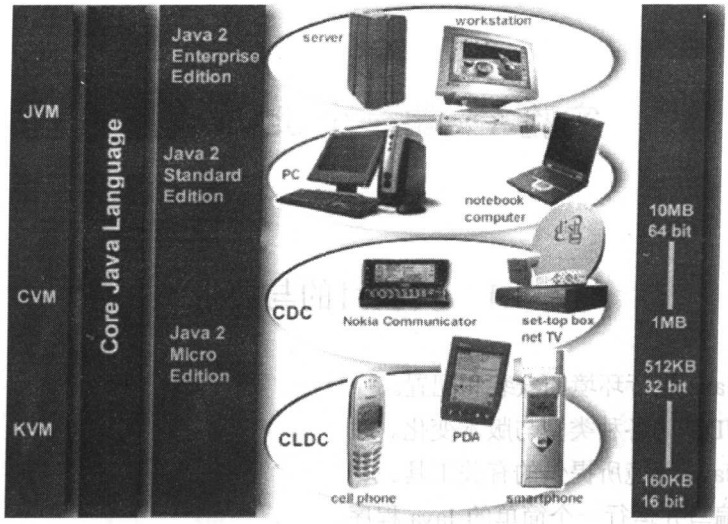

图 1-1 Java 2 的三种不同版本

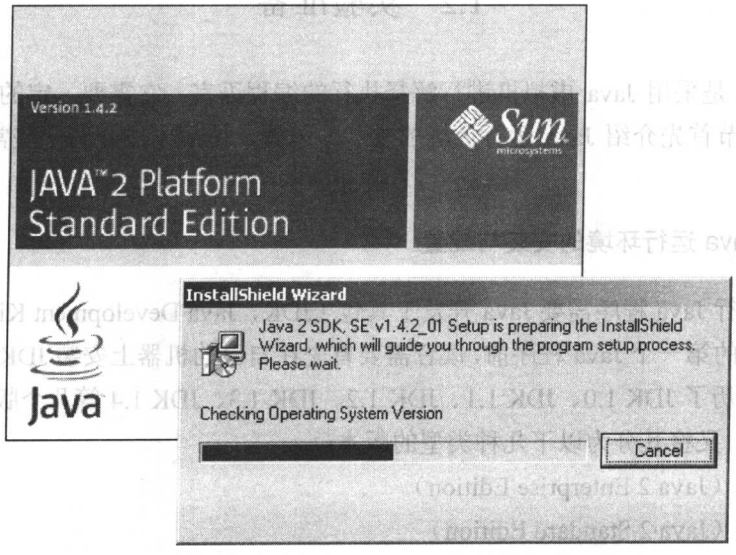

图 1-2 开始安装 Java 2 SDK

提示：如果读者需要获得 J2SDK 的最新版本及其相关文档，可以访问http://java.sun.com 查找并进行下载。

接着将出现许可协议对话框，如图 1-3 所示。这里只能选择“I accept the terms in the license agreement”，否则无法继续安装。然后，单击 Next 按钮。

在 Custom Setup 对话框中可以自定义所要安装的特性，默认情况下将全部予以安装。此外，还可以自行指定安装路径，如图 1-4 所示。

在后面的安装中，按照提示进行即可。安装成功后，j2sdk1.4.2 目录下将出现 bin、demo、include、jre、lib 五个文件夹和 COPYRIGHT、LICENSE、readme.html、README.txt、src.zip 五个文件。

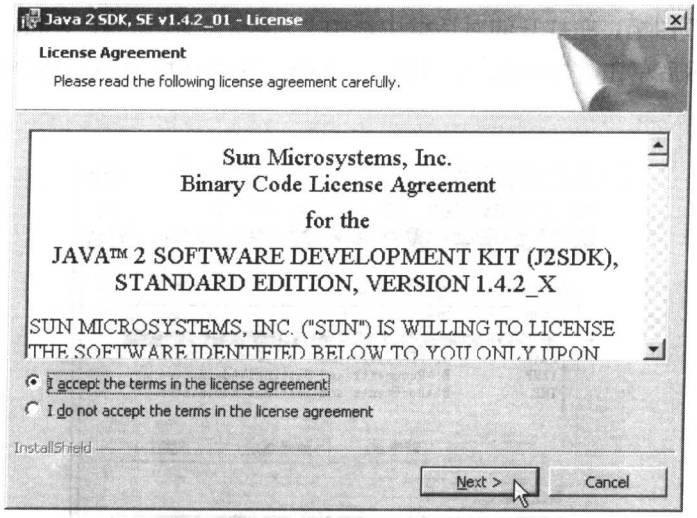

图 1-3　License Agreement 对话框

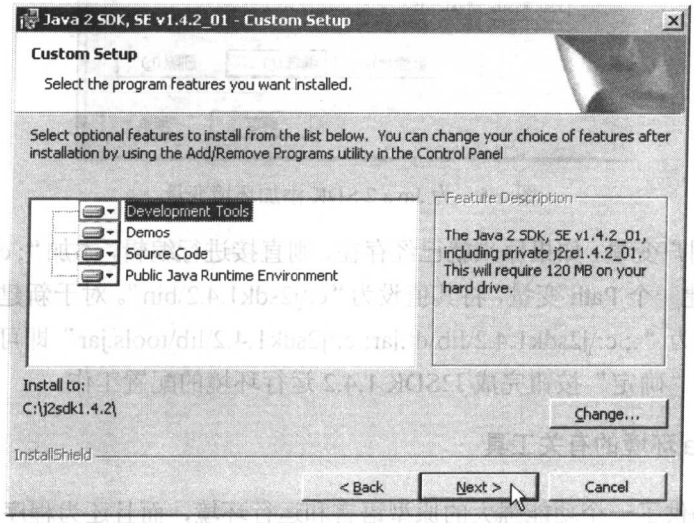

图 1-4　Custom Setup 对话框

安装完毕后，还需要对系统环境进行一下相关配置，JDK 才能正常工作。配置的中心环节是实现以下两条参数设置：

```
Path = %path%; c:\j2sdk1.4.2\bin
CLASSPATH = .; c:\j2sdk1.4.2\lib
```

如果读者所用的操作系统是 Windows 98，则在系统根目录（一般是 C 盘根目录）下找到 autoexec.bat 文件，打开该文件，向其中修改/添加如下两条语句：

```
set Path = [...]; c:\j2sdk1.4.2\bin   // [...]表示原来可能已经存在的设置
set CLASSPATH = .; c:\j2sdk1.4.2\lib\dt.jar; c:\j2sdk1.4.2\lib\tools.jar
```

然后，将文件保存，重新启动机器，这样就完成了 J2SDK 1.4.2 的安装。

如果读者机器的运行环境是 Windows NT/2000/XP/2003，则需要启动“控制面板”，双

击其中的“系统”图标，在打开的对话框中选中“高级”选项卡（在 Windows NT 中是“环境”选项卡），然后单击“环境变量”按钮，在弹出的对话框中添加两个用户变量，如图 1-5 所示。

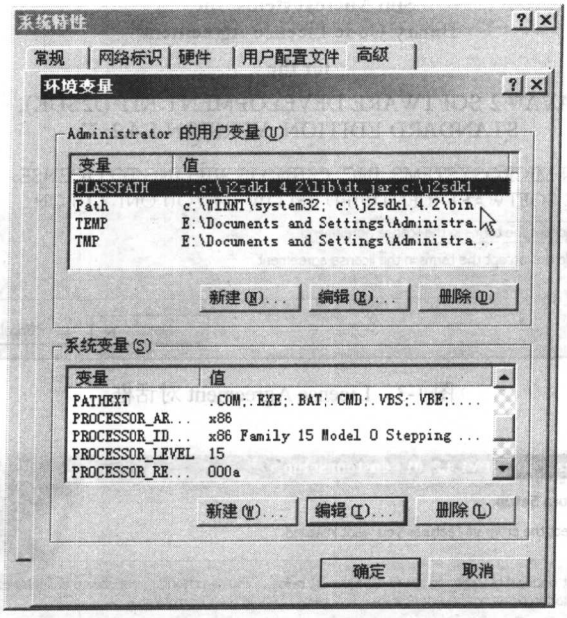

图 1-5　为 Java 2 SDK 添加环境变量

对于 Path 用户变量，如果原来就已经存在，则直接进行编辑，添加“; c:\j2sdk1.4.2\bin”即可；否则，创建一个 Path 变量，将其值设为“c:\j2sdk1.4.2\bin”。对于新建的 CLASSPATH 变量，将其值设为“.; c:\j2sdk1.4.2\lib\dt.jar; c:\j2sdk1.4.2\lib\tools.jar”即可。

最后，单击“确定”按钮完成 J2SDK 1.4.2 运行环境的配置工作。

1.2.2　Java 环境的有关工具

Java 不但提供了一个功能强大的原型语言和运行环境，而且还为程序员和最终用户提供了一些开发和使用 Java 的工具。Java 提供了程序员可用来更好地制作并更快地运行其程序的扩展类库。Java 类库是一组预先开发的程序码，可以与单独的应用程序相链接。Java 类库为程序员提供了一个彻底经过测试的健壮性很好的类集合。通过使用这些类库，程序员就不需要重新编写这些代码了，从而节约了时间，提高了效率。

下面将介绍一些主要 Java 工具的使用，这些工具有：

- java：解释器。
- javac：编译器。
- appletviewer：小应用程序浏览器。
- javah：头文件生成器。
- javadoc：API 文档生成器。
- javap：类文件反汇编器。

- jdb：Java 语言调试器。

这些文件在/java/bin/目录中，并可以在任何目录中运行，前提是设置了运行程序的相应系统路径。

1. Java 解释器

Java 解释器可以用来直接解释执行 Java 字节代码，具体命令行格式如下：

```
C:\>java options className arguments
```

其中，className 必须包括所有软件包信息。不仅有类名本身，还有 Java 解释器所期望的类名（注意：不是 Java 字节代码的文件名），所有在解释器环境下运行的类都必须包括解释器第一次调用时所需的 main 成员函数，用以传递命令所带的变量。

```
public static void main(string args[])
{
    ...
}
```

表 1-1 给出了 Java 解释器的所有参数选项。

表 1-1　Java 解释器命令的参数选项

参数选项	功能说明
-cs -checksource	让解释器重编译 Java 源文件已更新的类，即重编译已改变过了的类
-classpath path	重写 Classpath 环境变量，告诉 Java 在哪里能找到类库。如果其中用冒号分开，则可能包含多个目录
-mx x	设置内存分配池的最大值。所指定的池必须大于 1000 字节。另外，K、M 可以附加在数字上指定是千字节还是兆字节。默认值为 16MB
-ms x	设置内存分配池的最小值。所指定的池必须大于 1000 字节。另外，K、M 可以附加在数字上指定是千字节还是兆字节。默认值为 1MB
-noasyncgc	关闭异步无用单元收集功能，只有在程序中调用它或内存溢出时，无用单元收集才会被激活
-ss x	将 C 线程栈的最大值设置为 x，x 必须大于 1KB，其设定方式同-ms
-oss x	设定 Java 堆栈的最大值为 x
-v，-verbose	告知 Java 每当类被调用时，向标准输出设备输出信息
-verify	告知 Java 在所有代码上使用校验
-verifyremote	告知 Java 仅仅对类载入器所载入的类进行校验
-noverify	告知 Java 不进行校验
-verbosegc	告知 Java 让无用单元收集器在它释放内存时显示一条信息
-debug	允许 Java 调试器与本次 Java 解释器会话相联接。当它运行时，Java 会显示一个密码，用于启动这次调试会话
-D propName=newVal	允许用户在运行时改变属性值

2. Javac 编译器

Javac 编译器读取 Java 源代码，并将其编译成字节代码，调用 Javac 的命令行如下：

```
C:\>javac options filename.java
```

注意：与 Java 解释器不同，Javac 编译器希望它正在编译的文件具有.java 扩展名。其命令行参数选项如表 1-2 所示。

表 1-2 Javac 编译器命令的参数选项

参数选项	功能说明
-classpath path	用于设定路径，在该路径上 Javac 寻找需要被调用的类。该路径是一个用分号分开的目录列表
-d directory	指定一个根目录，该目录用来创建反映软件包继承关系的目录数
-g	在代码产生器中打开调试表，以后可以凭此调试产生字节代码
-nowarn	禁止编译器产生警告
-o	告诉 Javac 优化由内联的 static、final、private 成员函数所产生的代码
-verbose	告知 Java 显示出有关被编译的源文件和任何被调用类库的信息

3. Appletviewer 小应用程序浏览器

小应用程序浏览器 Appletviewer 提供了一个 Java 运行环境，在其中可以运行和测试小应用程序 Applet。Appletviewer 读取包含小应用程序的 HTML 文件，并在一个窗口中运行它们。

在小应用程序浏览器窗口的 Applet 菜单中，主要有如下的菜单命令：

- 重新启动（Restart）：重新启动运行小应用程序。
- 重新载入（Reload）：重新调取小应用程序。如果.class 文件在读取后被改变，此命令将是很有必要的。
- 停止（Stop）：停止运行小应用程序。
- 保存（Save）：将小应用程序序列化为文件保存（.ser 格式）。
- 启动（Start）：开始运行小应用程序。
- 复制（Clone）：按照 HTML 文件的命令行变量打开一个新的 Appletviewer 窗口。
- 标记（Tag）：打开一个对话框，显示用在 HTML 文件中的<applet>标签。
- 信息（Info）：提供关于小应用程序的有用信息。
- 属性(Properties)：此命令将打开 AppletViewer 属性对话框，允许为 Appletviewer 设置不同的网络和安全配置，如图 1-6 所示。此对话框允许 Appletviewer 指定 HTTP proxy 代理服务者及 firewall proxy 代理服务者运行，这需要知道上述两个 proxy 代理的地址和端口号。用户可以从站点管理者那里得到这些信息。网络访问安全性控制有不同级别的安全性，包括禁止网络访问、只允许访问小应用程序的主机、无限制访问等。类访问选择使你能够指定在机器上是否对类访问加以限制。

Appletviewer 是一个基本的工具，和 HTML 相似，它处理并显示了 Java 小应用程序。Java 小应用程序只是全部 WWW 页面中的一部分，观察小应用程序在其余 HTML 文件中的适应情况是十分重要的。因此，必须有一个全功能浏览器（如 Netscape 或 HotJava）。Appletviewer 惟一的可选项就是 debug，它将在 Java 语言调试器 jdb 中启动 Appletviewer。

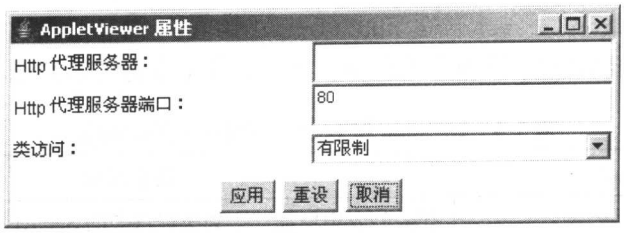

图 1-6　Appletviewer 属性对话框

4. Javah 头文件生成器

头文件生成器 Javah 程序创建 C 头文件和存根文件，这些是把本地 C 成员函数包入 java 所需要的。被创建的头文件给出了有关 java 类的信息，这些信息是 C 成员函数与 java 类交换数据所必需的。存根文件将用来创建定义 java 对象的结构及与 java 对象本身数据相联系的 C 文件。调用 Javah 的命令行格式如下：

```
C:\>javah options classname additionalClasses
```

Javah 程序有些像 Java 解释器，它只需要类名而不需要写.class 扩展名。Javah 程序可以接受多个类名以产生文件头和存根文件。表 1-3 列出了 javah 程序命令行的参数选项。

表 1-3　Javah 命令行的参数选项

参数选项	功能说明
-o outputfile	告诉 javah 将产生的所有文件头或存根文件都放到一个单独的文件 outputfile 中
-d directory	告知 javah 把产生的文件头或存根文件放入给定目录
-td directory	告知 javah 把临时文件放入目录 directory 而不是 tmp 目录中
-stubs	告知 javah 将产生存根文件而不是头文件
-verbose	告知 javah 将产生文件的状态输出到标准输出设备
-classpath path	告知 javah 用 path 目录寻找类文件，多个目录之间应该用分号分开

5. Javadoc API 文件产生器

Javadoc API 文件产生器程序读取一个 Java 类文件并自动创建一组 HTML 文件，这些 HTML 文件描述了 Java 类文件的类、变量、成员函数，所有 Java 类库的 APIHTML 文件都可以由此程序创建。Javadoc 把软件包名或源文件列表当作一个变量。Javadoc 依靠以@打头的备注标记来创建 HTML 文件。表 1-4 给出了标记列表，它们被 Javadoc 用于在 HTML 文件中创建链接。

除此之外，Javadoc 中也可以使用如下两个命令行参数：

- -classpath path：指定寻找 Java 文件的目录。
- -d directory：指定用来存放最终 HTML 文件的目录路径。通过对它的使用，可以使用户在源代码中加入更多的注释，这对提高程序的可读性是非常有用的。

6. Javap 反汇编器

反汇编器 Javap 用来反汇编一个 java 字节代码文件，返回有关可变部分和成员函数的信息，其命令行形式如下：

```
C:\>javap options classname additionalClasses
```

Javap 的标准输出是公有变量和类的成员函数，其命令行的参数选项如表 1-5 所示。

表 1-4 Javadoc API 文件产生器中所用的标记

标记名称	功能说明
@ see classname	在类列表中增加一个到所提供类的 See Also 条目
@ see classname # methodname	创建一个到特定成员函数的 See Also 条目
@ version text	在 HTML 文件中加入一个版本信息条目
@ author text	在 HTML 文件中加入一个作者信息条目
@ param name description	用成员函数备注来描述一个成员函数所带的变量
@ return description	用成员函数备注来描述返回值
@ exception classname	用成员函数备注来连接成员函数产生的异常出口

表 1-5 Javap 反汇编器命令的参数选项

参数选项	功能说明
-h	建立能够放入 C 头文件中的信息
-p	使 javap 输出私有和公有的成员函数和变量
-c	使 javap 为各成员函数输出实际已编译过的字节代码
-classpath path	使得 javap 在路径 path 中寻找 Java 类
-v	输出所有的信息
-verify	运行校验器以验证并显示出调试信息
-version	输出 javap 的版本信息

7. Java 调试器 jdb

Java 调度器为 Java 程序提供了一个命令行调试环境。它既可以在本地执行，也可以在与远程解释器的一次对话中执行。

Jdb 在本地机器中可以用如下命令启动：

```
C:\>jdb classname
```

当使用-debug 选项开始一个 Java 例程时，必须提供给 jdb 一个密码，这样 jdb 才能开始运转。表 1-6 中给出了所有常见的 jdb 命令。

表 1-6 Jdb 调试器命令一览表

Jdb 命令	功能说明
catch calssID	为特定异常出口而中断
classes	列出当前已知的类
clear classID:line	清除一个断点
cont	从断点处继续执行
down[n frames]	下移一个线程的堆栈
dump ID[ID...]	显示所有对象信息

续表

Jdb 命令	功能说明
exit（或 quit）	退出调试器
help（或?）	列出所有命令
ignore classID	忽略特定的异常出口
list[line number]	显示源代码
load classbame	载入要调试的 Java 类
locals	在当前堆栈帧中显示所有局部变量
memory	报告内存使用情况
methods classID	列出一个类的成员函数集
print ID[ID...]	列出对象或域
resume [threadID...]	恢复线程（默认情况恢复所有线程）
run class [args]	开始执行已下载的 Java 类
step	执行当前行
stop in classID:method	在一成员函数中设置一断点
stop at classID:line	在一行中设置一断点
suspend[threadID...]	停止一个线程（默认情况停止所有线程）
threads threadgroup	列出线程
thread threadID	设置当前线程
threadgroups	列出线程组
threadgroup name	设置当前线程组
up [n frames]	上移一个线程堆栈
use [path]	显示或改变源程序路径
where [threadID] or all !!	使一线程的堆线置空重复上一次命令

除了调试期间可用的命令，还有两个可选的命令行变量，它们可以用于远程调试。具体用法如下：

- -host hostname：告诉 Jdb 调试器到哪里去建立远程运行的 Java 解释器对话过程。
- -password password：告诉 Jdb 调试器使用哪个密码去与远程运行的 Java 对话进程相连接。密码 password 是由运行带有-debug 选项的 Java 解释器所提供的。

1.3　实验内容

本次实验的内容比较简单，就是使用两种方式编写“Hello World!”程序，使读者对 Java 编程有一个初步的认识。

1.3.1 第一个 Java 程序

现在，就可以开始编写并运行比较简单的 Java 程序了。下面是一个经典的 Java 入门程序，虽然只有短短几行代码，但其中的内容却很丰富，后面将进行具体说明：

```
// Code1-1: Hello World! - application
public class HelloWorldApp{
    public static void main (String args[])
    {   // Output the first sentence:
        System.out.println("Hello World!");
    }
}
```

此程序的作用是输出如下一行信息：

```
Hello World!
```

在程序代码中，首先用保留字 class 来声明一个新的类，类名为 HelloWorldApp，它是一个公共类（public）。整个类定义由大括号对“{}”括起来。在该类中，定义了一个 main()方法，其中 public 表示访问权限，指明所有的类都可以使用这一方法；static 指明该方法是一个类方法，它可以通过类名直接调用；void 指明 main()方法不返回任何值。对于一个应用程序来说，main()方法是必需的，而且必须按照上面的格式来定义。Java 解释器在没有生成任何实例的情况下，以 main()作为入口来执行程序。Java 程序中可以定义多个类，每个类中可以定义多个方法，但是最多只能有一个公共类，main()方法也只能有一个，作为程序的入口。在 main()方法定义中的，括号中的 String args[]是传递给 main()方法的参数，参数名为 args，它是类 String 的一个实例，参数可以为 0 个或多个，每个参数用“类名 参数名”来指定，多个参数间用逗号分隔。在 main()方法的实现中，只有一条语句：

```
System.out.println("Hello World!");
```

它用来实现字符串的输出，这条语句实现与 C 语言中的 printf 语句和 C++中的 cout<< 语句相同的功能。另外，//后的内容为注释。在编辑程序代码时，可以使用任何文本编辑器，例如记事本、UltraEdit、写字板等。这里推荐使用 UltraEdit，此工具支持 Java 程序的关键字多颜色显示。

现在就可以编译运行该程序了。首先，将其保存到一个名为 HelloWorldApp.java 的文件中。在这里，文件名应该与类名相同，因为 Java 解释器要求公共类必须放在与其同名的文件中。然后，对它进行编译：

```
C:\JavaBook>chap01>javac HelloWorldApp.java
```

编译的结果是生成字节码文件 HelloWorldApp.class。最后，使用 java 命令来运行该字节码文件：

```
C:\JavaBook>chap01>java HelloWorldApp
```

其结果就是在屏幕上显示出“Hello World!”这行文字。

1.3.2 Hello World!程序的 Applet 实现

除了上面的方法外，还可以使用另外一种方式来实现这个“Hello World!”程序。下面