从入门到提高丛书

使用最广泛的网络编程语言

Java 实例教程

许平凡 等编著

PeP 浦东电子出版社

前　言

SUN MicroSystem 公司的总裁 Scott McNealy 认为 Java 为 Internet 和 WWW 开辟了一个崭新的时代。微软总裁比尔·盖茨曾不无感慨地说："Java 是长时间以来最卓越的程序设计语言"，并确定微软公司的整个软件开发战略将从 PC 单机时代向着以网络为中心的计算时代转移。而购买 Java 则是他的重大战略决策的实施部署。

Java 的重要性已经不再是一个需要讨论的问题了，我们现在需要做的事就是，怎样把 Java 这门语言学好，用好。在浏览网页的时候，Java Applet 程序所表现出的精彩效果令我们叹服，它可以为网页增添不少风采。我们也需要学会它，并且把它用到我们的网站上，使之更好地为我们所用。

应广大读者的要求，我们几个业界的朋友决定出一本书，书的内容就是教会朋友们掌握 Visual J++。其实，我们很早就有了这个想法，但是由于时间和配合的关系，一直不能让这本书与朋友们见面，实在是一大遗憾。今天，这本书终于出现在朋友们的眼前，我们这些计算机编程人员也是感到欣慰的。不敢说这本书写的水平是多么的高，也不敢说这本书的技术含量如何，但我们倾入了心血，把自己这些年的编程经验和教训尽量地融入了这本书中，并尽可能认真而又细致地编写这本书。我们希望朋友们在阅读这本书的时候能够尽可能多地学到知识。

在这本书里面，我们将向朋友们讨论关于 Java 的特点以及应用。首先，我们要看一看 Java 的特点，包括 Java 语言与其他语言的区别，以及它的优越性。然后，我们将讨论 Java 中比较重要的几个方面：线程、异常、输入输出以及远程方法调用。实际上，在这几章里，我们经常涉及到其他很多东西，包括图形、动画。在本书的附录里面，我们给大家一些例程，可以看作这本书的内容的综合应用。这也是我们业余时间的倾力之作。当然，大部分程序人员都有编娱乐节目的爱好，我们给的程序也不例外。

要说本书面对的读者层次实在不是很好说，要让一个对电脑很不熟悉的朋友来看这本书，我觉得难度比较大；要让一个专业的高水平的 Java 程序员阅读这本书，我觉得有些浪费时间，尽管这些朋友也能从这本书里面学到一些东西。所以，我们面向的读者就是那些对电脑有些理解，而对 Java 语言并不是特别精通的人，最好是对计算机软件有一点了解的朋友们。在本书中，我们并没有特别地讲解数据结构方面的知识，也没有特别提出面向对象的程序设计思想。如果朋友们要在计算机软件行业有所成就，这些知识是必须非常精通的，希望朋友们努力。

在阅读本书的同时，我们希望朋友们能够安装 Microsoft Visual J++ ，或者一边阅读本书，一边动手实践，"实践出真知"是永恒的真理。希望这本书对朋友们能够有所帮助。

本书主要由许平凡编著。参加本书编写的还有赫骞、邓淼、张云冰、陈满才、言金刚、俞涓、屈博勋、乔嵩、彭进展、曾增、李静、程雁玲、彭玉洪、许刚、肖辉、张永军等。由于作者水平有限，不足之处在所难免，欢迎广大读者批评指正。

2000 年 10 月

目　录

第 1 章　Java 的特点

Java 是广泛使用的网络编程语言，它是一种新的计算概念。

首先，作为一种程序设计语言，它简单，面向对象，不依赖于机器的结构，具有可移植性、鲁棒性、安全性并且提供了并发的机制，具有很高的性能。其次，它最大限度地利用了网络，Java 的小应用程序（applet）可在网络上传输而不受 CPU 和环境的限制。另外，Java 还提供了丰富的类库，使程序设计者可以很方便地建立自己的系统。

下面我们分别从这三个方面来讨论 Java 的特点，然后通过比较 Java 与 C/C++，进一步指出它所具有的优点。

1.1　Java 语 言

Java 语言具有以下特点：简单、面向对象、分布式、解释执行、鲁棒、安全、体系结构中立、可移植、高性能、多线程以及动态性。

1.1.1　简单性

Java 语言是一种面向对象的语言，它通过提供最基本的方法来完成指定的任务。读者只需理解一些基本的概念，就可以用它编写出适合于各种情况的应用程序。Java 省去了运算符重载、多重继承等模糊的概念，并且通过实现自动垃圾收集大大简化了程序设计者的内存管理工作。另外，Java 也适合于在小型机上运行，它的基本解释器及类的支持只有 40KB 左右，加上标准类库和线程的支持也只有 215KB 左右。

1.1.2　面向对象

Java 语言的设计集中于对象及其接口，它提供了简单的类机制以及动态的接口模型。对象中封装了它的状态变量以及相应的方法，实现了模块化和信息隐藏；而类则提供了一类对象的原型，并且通过继承机制，子类可以使用父类所提供的方法，实现了代码的复用。

1.1.3　分布性

Java 是面向网络的语言。通过它提供的类库可以处理 TCP/IP 协议，用户可以通过 URL 地址在网络上方便地访问其他对象。

1.1.4　鲁棒性

Java 在编译和运行程序时，都要对可能出现的问题进行检查，以消除错误的产生。它提供自动垃圾收集来进行内存管理，防止程序员在管理内存时容易产生错误。通过集成面向对象的异常处理机制，在编译时，Java 提示可能出现但未被处理的异常，帮助程序员正确地进行选择以防止系统的崩溃。另外，Java 在编译时还可捕获类型声明中的许多常见错误，防止动态运行时不匹配问题的出现。

1.1.5　安全性

用于网络、分布环境下的 Java 必需防止病毒的入侵。Java 不支持指针，一切对内存的访问都必需通过对象的实例变量来实现，这样就防止了程序员使用“特洛伊”木马等欺骗手段访问对象的私有成员，同时也避免了指针操作中容易产生的错误。

1.1.6　体系结构中立

Java 解释器生成与体系结构无关的字节码指令，只要安装 Java 运行系统，Java 程序就可在任意的处理器上运行。这些字节码指令对应于 Java 虚拟机中的表示，Java 解释器得到字节码后，对它进行转换，使之能够在不同的平台上运行。

1.1.7　可移植性

与平台无关的特性使 Java 程序可以方便地被移植到网络中的不同机器上。同时，Java 的类库中也实现了与不同平台的接口，使这些类库可以移植。另外，Java 编译器是由 Java 语言实现的，Java 运行时系统由标准 C 实现，这使得 Java 系统本身也具有可移植性。

1.1.8　解释执行

Java 解释器直接对 Java 字节码进行解释执行。字节码本身携带了许多编译时信息，使得连接过程更加简单。

1.1.9　高性能

和其他解释执行的语言如 BASIC 或 TCL 不同，Java 字节码的设计使之很容易地直接转换成对应于特定 CPU 的机器码，从而得到较高的性能。

1.1.10　多线程

多线程机制使应用程序能够并行执行，而且同步机制保证了对共享数据的正确操作。通过使用多线程，程序设计者可以分别用不同的线程完成特定的行为，而不需要采用全局的事件循环机制，这样就很容易地实现网络上的实时交互行为。

1.1.11 动态性

Java 的设计使它适合于一个不断发展的环境。在类库中可以自由地加入新的方法和实例变量而不会影响用户程序的执行。并且 Java 通过接口来支持多重继承，使之比严格的类继承具有更灵活的形式和扩展性。

1.2 Java Applet

Java 语言的特性使它可以最大限度地利用网络。Applet 是 Java 的小应用程序，它是动态、安全和跨平台的网络应用程序。Java Applet 嵌入 HTML 语言，通过主页发布到 Internet。网络用户访问服务器的 Applet 时，这些 Applet 从网络上下载，然后在支持 Java 的浏览器中运行。由于 Java 语言的安全机制，用户一旦载入 Applet，就可以放心地来生成多媒体的界面或完成复杂的计算而不必担心病毒的入侵。虽然 Applet 可以和图像、声音、动画等一样从网络上下载，但它并不等同于这些多媒体的文件格式，它可以接收用户的输入，动态地进行改变，而不仅仅是动画的显示和声音的播放。

1.3 丰富的类库

Java 提供了大量的类以满足网络化、多线程、面向对象系统的需要。

（1）语言包提供字符串处理、多线程处理、异常处理、数学函数处理等，可以用它简单地实现 Java 程序的运行平台。

（2）实用程序包提供的支持包括哈希表、堆栈、可变数组、时间和日期等。

（3）输入输出包用统一的“流”模型来实现所有格式的 I/O，包括文件系统、网络、输入等。

（4）低级网络包用于实现 Socket 编程。

（5）抽象图形用户接口包实现了不同平台的计算机图形用户接口部件，包括窗口、菜单、滚动条、对话框等，使得 Java 可以移植到不同平台的机器上。

（6）网络包支持 Internet 的 TCP/IP 协议，提供了与 Internet 的接口。

它支持 URL 连接和 WWW 的即时访问，并且简化了客户机/服务器模型的程序设计。

1.4 Java 和 C/C++

对于变量声明、参数传递、操作符、流控制等，Java 使用了和 C/C++ 相同的传统，使得熟悉 C/C++ 的程序员能很方便地进行编程。同时，Java 为了实现其简单、鲁棒、安全等特性，也摒弃了 C 和 C++ 中许多不合理的内容。

1.4.1　全局变量

Java 程序中，不能在所有类之外定义全局变量，只能通过在一个类中定义公用、静态的变量来实现一个全局变量。例如：

```
Class GlobalVar{
public static global_var;
}
```

在类 GlobalVar 中定义变量 global_var 为 public static，使得其他类可以访问和修改该变量。

Java 对全局变量进行了更好的封装。而在 C 和 C++ 中，依赖于不加封装的全局变量常常造成系统的崩溃。

1.4.2　goto

Java 不支持 C/C++ 中的 goto 语句,而是通过异常处理语句 try、catch、final 等来代替 C/C++ 中用 goto 来处理遇到错误时跳转的情况，使程序更可读且更结构化。

1.4.3　指针

指针是 C/C++ 中最灵活，也是最容易产生错误的数据类型。由指针所进行的内存地址操作常会造成不可预知的错误，同时通过指针对某个内存地址进行显式类型转换后，可以访问一个 C++ 中的私有成员，从而破坏安全性，造成系统的崩溃。而 Java 对指针进行完全的控制，程序员不能直接进行任何指针操作，例如把整数转化为指针，或者通过指针释放某一内存地址等。同时，数组作为类在 Java 中实现，良好地解决了数组访问越界这一 C/C++ 中不作检查的错误。

1.4.4　内存管理

在 C 中，程序员通过库函数 malloc() 和 free() 来分配和释放内存，C++ 中则通过运算符 new 和 delete 来分配和释放内存。再次释放已释放的内存块或未被分配的内存块，会造成系统的崩溃；同样，忘记释放不再使用的内存块也会逐渐耗尽系统资源。而在 Java 中，所有的数据结构都是对象，通过运算符 new 为它们分配内存块。通过 new 得到对象的处理权，而实际分配给对象的内存可能随程序运行而改变，Java 对此自动地进行管理并且进行垃圾收集，有效防止了由于程序员的误操作而导致的错误，并且更好地利用了系统的资源。

1.4.5　数据类型的支持

在 C/C++ 中，对于不同的平台，编译器对于简单数据类型如 int、float 等分别分配不同长度的字节数，例如：int 在 IBM PC 中为 16 位，在 VAX-11 中为 32 位，这导致了代码的不可移植性。但在 Java 中，对于这些数据类型总是分配固定长度的位数，如对 int 型，

它总占 32 比特，这就保证了 Java 的平台无关性。

1.4.6 类型转换

在 C/C++ 中，可以通过指针进行任意的类型转换，这样会经常带来不安全性。而在 Java 中，运行时系统对于对象的处理要进行类型相容性检查，以防止不安全的转换。

1.4.7 头文件

C/C++ 中用头文件来声明类的原型以及全局变量、库函数等，在大的系统中，维护这些头文件是很困难的。而 Java 不支持头文件，类成员的类型和访问权限都封装在一个类中，运行时系统对访问进行控制，防止对私有成员的操作。同时，Java 中用 import 语句与其他类进行通讯，以便使用它们的方法。

1.4.8 结构和联合

C/C++ 中的结构和联合中所有成员均为公有，这就带来了安全性问题。Java 中不包含结构和联合，所有的内容都封装在类中。

1.4.9 预处理

C/C++ 中用宏定义实现的代码给程序的可读性带来了困难。在 Java 中不支持宏，它通过关键字 final 来声明一个常量，以实现宏定义中广泛使用的常量定义。

1.4.10 简单的例程分析

下面先介绍两个简单的 Java 程序，并对其进行分析。首先让我们看下面的例子。

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

本程序的作用是输出下面一行信息：

```
Hello World!
```

程序中，首先用保留字 class 来声明一个新的类，其类名为 HelloWorldApp，它是一个公共类（public）。整个类定义由大括号{}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 则指明 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，而且必需按照如上的格式来定义。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，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(大括号)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 cout<<语句具有相同的功能。另外，双斜杠“//”后的内容为注释。

现在我们可以运行该程序。首先把它放到一个名为 HelloWorldApp.java 的文件中，这里，文件名应和类名相同，因为 Java 解释器要求公共类必需放在与其同名的文件中。然后对它进行编译：

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

编译的结果是生成字节码文件 HelloWorldApp.class。最后用 Java 解释器来运行该字节码文件：

```
C:\>java HelloWorldApp
```

结果在屏幕上显示 Hello World!

我们再来看下面的一个例子：

```
import java.awr.*;
import java.applet.*;
public class HelloWorldApplet extends Applet
{
        //an applet
        public void paint(Graphics g)
        {
                g.drawString ("Hello World!",20,20);
    }
}
```

这是一个简单的 Applet（小应用程序）。程序中，首先用 import 语句输入 java.awr 和 java.applet 下所有的包，使得该程序可以使用这些包中所定义的类，它类似于 C 中的#include 语句。然后声明一个公共类 HelloWorldApplet，用 extends 指明它是 Applet 的子类。在类中，我们重写父类 Applet 的 paint() 方法，其中参数 g 为 Graphics 类，它表明当前作画的上下文。在 paint() 方法中，调用 g 的方法 drawString()，在坐标（20，20）处输出字符串“"Hello World!"”，其中坐标用像素点来表示。

这个程序中没有实现 main() 方法，这是 Applet 与应用程序 Application（如前一例）的区别之一。为了运行该程序，首先我们要把它放在文件 HelloWorldApplet.java 中，然后对它进行编译：

```
C:\>javac HelloWorldApplet.java
```

得到字节码文件 HelloWorldApplet.class。由于 Applet 中没有 main() 方法作为 Java 解

释器的入口，我们必需编写 HTML 文件，把该 Applet 嵌入其中，然后用 appletviewer 来运行，或在支持 Java 的浏览器上运行。它的<HTML>文件如下：

```
<HTML>
<HEAD>
<TITLE> An Applet </TITLE>
</HEAD>
<BODY>
<applet code="HelloWorldApplet.class" width=200 height=40>
</applet>
</BODY>
</HTML>
```

其中用<applet>标记来启动 HelloWorldApplet，code 指明字节码所在的文件，width 和 height 指明 applet 所占的大小，我们把这个 HTML 文件存入 Example.html，然后运行：

```
C:\>appletviewer Example.html
```

这时屏幕上弹出一个窗口，其中显示 Hello World!

从上述例子中可以看出，Java 程序是由类构成的。对于一个应用程序来说，必需在一个类中定义 main() 方法，而对 applet 来说，它必需作为 Applet 的一个子类。在类的定义中，应包含类变量的声明和类中方法的实现。Java 在基本数据类型、运算符、表达式、控制语句等方面与 C/C++ 基本是相同的，但它同时也增加了一些新的内容，在以后的各章中我们会详细介绍。本节中，只是使用户对 Java 程序有一个初步的了解。

1.5 本 章 小 结

1.5.1 主要内容

本章的目的是使读者对于 Java 有一定程度的了解。知道 Java 到底是什么东西，有什么功能，并且有什么好处。另外还对 Java 语言和 C/C++ 语言的不同之处进行了分析，对于了解、熟悉 C/C++ 的读者会有一定的帮助。

1.5.2 课后习题

（1）Java 语言有哪些特点？
（2）Java 语言为什么会有大量的类库？
（3）Java 语言在语法上与 C/C++ 有什么不同？
（4）Java 语言和 C/C++ 语言到底有哪些不同？
（5）Java 到底是不是 Microsoft 的产品？
（6）Microsoft 为什么会推出 VJ 6.0?

第2章 Java 概 述

Java 是由 Sun 公司开发而成的新一代编程语言。可以使用它在各式各样不同类型机器、不同类型操作平台的网络环境中开发软件。

Java 虽出现的时间不长，但已被业界接受，IBM、Apple、DEC、Adobe、Silicon Graphics、HP、Oracle、Toshiba、Netscap 和 Microsoft 等大公司已经购买了 Java 的许可证。Microsoft 还在其 Web 浏览器 Explorer 3.0 版中增加了对 Java 的支持。

本章主要讲解 Java 的发展史和 Java 所带来的影响。

2.1 Java 的发展史

2.1.1 什么是 Java

美国硅谷有一句行话，每 10 年到 15 年有一次轮回。最近的一次轮回就是从 Java 开始。

Java 是由 Sun 公司开发而成的新一代编程语言。可以使用它在各式各样不同类型机器、不同类型操作平台的网络环境中开发软件。不论用户使用的是哪一种 WWW 浏览器，哪一种计算机，哪一种操作系统，只要 WWW 浏览器上面注明了“支持 Java”，就可以看到生动的主页。Java 正在逐步成为 Internet 应用的主要开发语言。它彻底改变了应用软件的开发模式，带来了自 PC 机以来又一次技术革命，为迅速发展的信息世界增添了新的活力。

Sun 的 Java 语言开发小组成立于 1991 年，其目的是开拓消费类电子产品市场，例如，交互式电视、烤面包箱等。Sun 内部人员把这个项目称为 Green，那时 World Wide Web 还在图纸上呢。该小组的领导人 James Gosling 是一位非常杰出的程序员。他出生于 1957 年，于 1984 年加盟 Sun Microsystem 公司，之前在 IBM 一家研究机构工作。他是 Sun NeWs 窗口系统的总设计师。也是第一个用 C 实现 EMACS 文本编辑器 COSMACS 的开发者。

在研究开发过程中，Gosling 深刻体会到消费类电子产品和工作站产品在开发哲学上的差异：消费类电子产品要求可靠性高、费用低、标准化、使用简单，用户并不关心 CPU 的型号，也不欣赏专用昂贵的 RISC 处理器，他们需要建立在一个标准基础之上，具有一系列可选的方案，从 8086 到 Pentium 都可以选取。

2.1.2 从 C 开始

为了使整个系统与平台无关，Gosling 首先从改写 C 编译器着手。但是 Gosling 在改写过程中感到仅仅 C 是无法满足需要的。于是在 1991 年 6 月份开始准备开发一个新的语言，那么给它起一个什么名字呢？Gosling 回首向窗外望去，看见一棵老橡树，于是建了一个目录叫 Oak，这就是 Java 语言的前身（后来发现 Oak 已是 Sun 公司另一个语言的注册商标，

才改名为 Java，即太平洋上一个盛产咖啡的岛屿的名字）。

Gosling 在开始写 Java 时，并不局限于扩充语言机制本身，更注重于语言所运行的软硬件环境。他要建立一个系统，这个系统运行于一个巨大的、分布式的、异构的网格环境中，完成各电子设备之间的通信与协同工作。Gosling 在设计中采用了虚拟机器码（Virtual Machine Code）方式，即 Java 语言编译后产生的是虚拟机，虚拟机运行在一个解释器上，每一个操作系统均有一个解释器。这样一来，Java 就成了平台无关语言。这和 Gosling 设计的 Sun NeWs 窗口系统有着相同的技术味道。在 NeWs 中，用户界面统一用 Postscript 描述，不同的显示器有不同的 Postscript 解释器，这样便保证了用户界面良好的可移植性。

Patrick Naughton 也是 Sun 公司的技术骨干，曾经是 Open Windows 项目的负责人。当 Naughton 加入该小组后，整个工作进展神速。经过 17 个月的奋战，整个系统胜利完成。它是由一个操作系统、一种语言（Java）、一个用户界面、一个新的硬件平台、三块专用芯片构成的。通常情况下，这样的项目在 Sun 公司要 75 个人干三年。项目完成后，在 Sun 公司内部做了一次展示和鉴定，观众的反应是：在各方面都采用了崭新的、非常大胆的技术。许多参观者对 Java 留下了非常深刻的印象，特别是得到了 Sun 的两位领导人 Scott McNealy 和 Bill Joy 的关注，但 Java 的前途未卜。

2.1.3 Java 语言的转折点

到了 1994 年，WWW 已如火如荼地发展起来。Gosling 意识到 WWW 需要一个中性的浏览器，它不依赖于任何硬件平台和软件平台，它应是一种实时性较高、可靠安全、有交互功能的浏览器。于是 Gosling 决定用 Java 开发一个新的 Web 浏览器。

这项工作由 Naughton 和 Jonathan Payne 负责，到 1994 年秋天，完成了 WebRunner 的开发工作。WebRunner 是 HotJava 的前身，这个原型系统展示了 Java 可能带来的广阔市场前景。WebRunner 改名为 HotJava，于 1995 年 5 月 23 日发表后，在产业界引起了巨大的轰动，Java 的地位也随之而得到肯定。又经过一年的试用和改进，Java 2.0 版终于在 1996 年年初正式发表。

2.2 Java 带来的影响

Java 虽然出现的时间不长，但已被业界接受，IBM、Apple、DEC、Adobe、Silicon Graphics、HP、Oracle、Toshiba、Netscap 和 Microsoft 等大公司已经购买了 Java 的许可证。Microsoft 还在其 Web 浏览器 Explorer 3.0 版中增加了对 Java 的支持。

另外，众多的软件开发商也开发了许多支持 Java 的软件产品。如：Borland 公司的基于 Java 的快速应用程序开发环境 Latte；Metrowerks 公司和 Natural Intelligence 公司分别开发的基于 Machintosh 的 Java 开发工具；Sun 公司的 Java 开发环境 Java Workshop；Microsoft 也开发出系列 Java 产品。数据库厂商如 Illustra、Sybase、Versant、Oracle 都在开发支持 HTML 和 Java 的 CGI（Common Gateway Interface）。在以网络为中心的计算时代，不支持 HTML 和 Java，就意味着应用程序的应用范围只能局限于有限的环境。