Java 经典教材译丛

Java 程序设计导论

著者：[美] Rick Decker
Stuart Hirshfield
译者：董庆霞

THOMSON
北京大学出版社
http://cbs.pku.edu.cn

Java 经典教材译丛

Java 程序设计导论

programming. java:
An Introduction to Programming Using Java
(Second Edition)

[美] Rick Decker　Stuart Hirshfield　著
董庆霞　译

北京大学出版社
·北　京·

内 容 简 介

本书适合于大中专院校计算机相关专业作为教材，也是Java初学者以及Java爱好者的理想参考用书。书中详细介绍了Java语言的发展、AWT基础、Java语言基础(包括Java语言特征、循环、数组、String类等)，使用AWT类进行可视化设计，编写交互式应用程序等内容。

本书结构严谨，内容全面，每节的最后都有问题回顾，并在每章最后提供答案，以便读者巩固所学知识。

Original English language title: programming.java: An Introduction to Programming Using Java, 2nd Edition by Rick Decker, Stuart Hirshfield.

EISBN: 0-534-37109-4

981-254-150-0

北京市版权局著作权合同登记号　图字 01-2003-8584 号

图书在版编目(CIP)数据

Java 程序设计导论/(美)德克尔(Decker,R.)等著；董庆霞译. —北京：北京大学出版社，2003.11
(Java 经典教材译丛)
ISBN 7-301-06641-4
Ⅰ.J… Ⅱ.①德… ②董… Ⅲ.JAVA 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2003)第 098000 号

书　　名：Java 程序设计导论
著作责任者：[美] Rick Decker　Stuart Hirshfield 著
译　　者：董庆霞
责任编辑：胡伟晔
标准书号：ISBN 7-301-06641-4/TP・0741
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
网　　址：http://cbs.pku.edu.cn　http://www.macrowin.net
电　　话：发行部 62750672　编辑部 62750581　邮购部 62752015
电子信箱：xxjs@pup.pku.edu.cn　macrowin@macrowin.net
排 版 者：北京东方人华北大彩印中心　电话：62754190
印 刷 者：河北涿县鑫华书刊印刷厂
发 行 者：北京大学出版社
经 销 者：新华书店
787 毫米×1092 毫米　16 开本　23.875 印张　739 千字
2003 年 12 月第 1 版　2003 年 12 月第 1 次印刷
定　　价：36.00 元

编 者 序

欢迎学习 Java 程序设计语言！

自计算机发明以来，无处不在显示着它对整个人类文明和社会进步产生着如此巨大、如此深刻的影响，推动着人类社会一日千里地向前发展

历经半个多世纪，计算机软件——计算机的重要组成部分，也发生了日新月异的变化。从机器语言到汇编语言再到高级语言，从结构化语言到面向对象语言，人们不断地探索和发明新的、功能强大又简单易学的计算机语言。

用计算机语言编程，有如我们用世界上的任何一种语言写作一样，两者都是创造性的工作，而且都要用到相应的技术。如果您掌握了一种语言，熟知它的词汇和语法规则，也许您能熟练地运用这种语言进行表达，但这并不能使您成为一位作家。写作是一种创造性的工作，它需要有创造性的人来做。用计算机语言编程也是同样的道理。也许您掌握了当今最流行的程序设计语言，而且也能迅速地写出程序语句，但是，程序的内容只能靠您的创造性来产生。

1. 为什么要学习 Java 程序设计语言？

工欲善其事，必先利其器。没有得心应手的工具又怎么能随心所欲地发挥创意？计算机高级语言众多，而 Java 语言的得天独厚的优点，使它从众多语言中脱颖而出，越来越受到人们的青睐。Java 是一种跨平台、适合于分布式计算环境的面向对象编程语言。具体来说，它具有如下特性：简单、面向对象、分布式、易解释、可靠、安全、与平台无关、体系结构中立、可移植、高性能、多线程、动态等。

鉴于此，我们从国外遴选了 8 本流行的 Java 书，组成了《Java 经典教材译丛》，以期读者能从中受益。

2. 您渴望从《Java 经典教材译丛》学到什么？

博大精深是我们选此套丛书的宗旨，从 Java 基本概念和技术到其高级主题，丛书的内容涵盖了 Java 的全部知识。原书多由国外教学经验丰富的教师编写，书中提供大量的实例和案例。为了给读者提供良好的服务，我们将大部分书中的实例中的源代码放在 http://cbs.pku.edu.cn 中的【下载专区】上，读者可从中下载。

3. 什么人使用《Java 经典教材译丛》？

无论您是一位计算机语言初学者，还是一位熟知计算机语言的编程人员，《Java 经典教材译丛》中必有一本适合您。当然，本套丛书主要针对在校大专院校的学生而编写，所以大多数读者都可以轻松上手，教师更可以从中找到教学用书和参考读物，部分书中带有星号(*)的章节则是为更高层次的读者而设计，为选学内容。

4. 对计算机的硬件和软件有什么需求？

- 软件　可以通过访问 Sun 的网站(http://java.sun.com)下载 JDK 编译程序，下载的软件大约为 20MB。
- 硬件　必须具备可以运行 Windows 操作系统的计算机或者 Windows NT 工作站，100MB 的空闲磁盘空间，最少 32MB 的内存(如果使用 Windows NT，建议准备 64MB 的内存)。

丛书编委会

2003 年 11 月

目　录

第 1 章　Java 简介

本章将从历史和技术的角度介绍 Java，简要描述什么是程序设计语言、在过去 50 年内它们是如何发展的、它们的工作原理以及早期 Java(Java 的历史和计算机短暂的历史一样长)、现代 Java 与其他程序设计语言的区别。同时，还将简单介绍图形用户界面的发展、描述 WWW 的萌芽，以及 WWW 如何促使现代计算机与日益受欢迎的 Java 语言紧密联系在一起。

本章将实现以下目标：

- 讨论从早期的程序设计语言到 Java 的程序设计语言发展史
- 学习 Internet 和 WWW 的发展史，并研究 Java 与 Web 页面是如何紧密联系在一起的
- 简要介绍创建 Java 程序的过程，并将其放在 Web 页面中运行
- 探讨 Java 的本质，并描述 Java 与其他程序设计语言的区别

1.1　程序设计的发展过程

用最简单的语言描述，程序设计就是设计计算机可以执行的用于解决某个问题的指令集的过程。如果把“计算机”用“人”来代替，那么这听起来像一项非常直接的练习。每次给路过的驾车人员提供如何到达邮局的信息时，都将进行这类事情，即向接收者提供明确的可以执行的命令信息：“沿着这条路走 3 英里，在第 3 个信号灯处向左转，然后直行，直到看到学校为止，邮局就在学校过去的路左边。”两者之间的区别就是接收命令的人和传达命令的人说同一种语言，而计算机却不是这样。事实上，计算机可以“理解”(即可以执行这种语言的指令)的惟一语言是像英语这样的自然语言指令。确切地说，计算机使用的语言非常复杂，人们几乎不能发明比它更难懂的语言，因此，把程序设计的整个发展史看成一系列试图推动与计算机交流的过程史是完全合情合理的。下面讨论与计算机交流的问题以及解决方案的发展过程。

1.1.1　并不辉煌的过去

电子计算机在二十世纪三四十年代由美国、德国和英国独立发明，第一台成功的计算机非常巨大，而且十分昂贵，计算能力基本上与现代电子表相同。尽管存在许多缺点，但是在第二次世界大战结束时可用的计算机(如图 1.1 所示)的运算速度仍比人类计算器快上千倍，许多研究中心、政府机构和私有企业的热衷者愿意支付上百万美元去购买这些“巨脑”机器。那时的程序设计与现在相比要困难得多，计算机指令是用机器本身的语言编写的，因此，惟一能够编写程序的是那些精通计算机语言的为数很少的人。

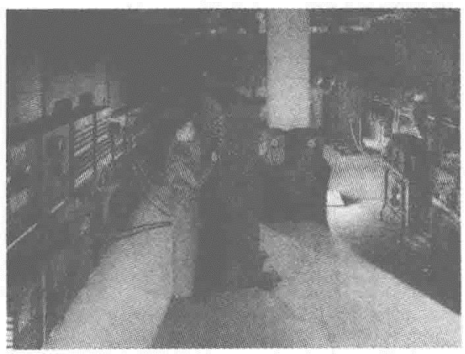

图 1.1　ENIAC，第一台电子计算机，1946 年

计算机语言由计算机的设计方式决定。从本质上说，计算机只不过是彼此间相互连接、处于开-关(on-off)状态的庞大开关集。计算机处理的任何事物，不管是数据还是指令，必须在计算机内部表示为开-关状态的电流集。例如，如果想设计一台可以对整数执行数学运算的计算机，那么进行操作的每个数字都必须可以用合适的开-关值的集合表示，这种二值信息编码称为二进制表示。

例如如果使用两条电线代表数字，即 0 用“off,off”表示，1 用“off,on”表示，2 用“on,off”表示，3 用“on,on”表示；为简化描述，此处用 0 代表 off，用 1 代表 on(没有其他原因，只不过是为了节省输入时间)；那么使用两条电线表示法，就可以用 00 表示 0，用 01 表示 1，用 10 表示 2，用 11 表示 3。当然，这是一种十分有限的编码方案，因为两个开关的 on-off 状态只能表示 4 个数值。我们也可以用 8 个或 16 个开关设计计算机，其原理是相同的。

决定如何表示数字后，接下来将设计其他的开关和电线，从而可以执行加法等数学运算。我们把 4 条输入电线(表示两组数字的电线)和 3 条输出电线(表示和的两个数字，加上另一个用于传输的数字)组合成一个电路，如图 1.2 所示。在这里加法电路工作的细节并不重要，重要的是如何设计电路，使其可以提供所需的所有运算，如减法、乘法和比较运算等；同时还要预留开关集合，用于存储输入、中间和最后的数值，即作为计算机的内存使用。

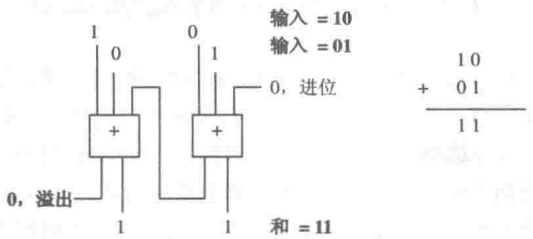

图 1.2 计算 1 + 2 = 3 的简单加法电路

为了完成设计，还需要包含控制开关，用于决定哪些数值在内存时应该激活哪些相应的运算电路。在当前比较简单的例子中，使用了 3 个控制开关，分别是“off,off,on”打开加法器，“off,on,off”打开减法器等。这些设计选项接下来会支配机器语言进行动作。机器语言指令的形式如图 1.3 所示，其中使用 3 位二进制数值决定运算类型、4 位二进制数值决定使用的内存位置，以及另外 4 位二进制数决定结果存储的内存位置。采用这种表示法，可以在 16 个内存位置上执行 8 条指令(你明白为什么吗？)。例如，机器指令 001 0000 0001 0010 将把内存位置 0(0000)和内存位置 1(0001)的数值加起来(001 代码表示)，结果存储在内存位置 2(0010)上。

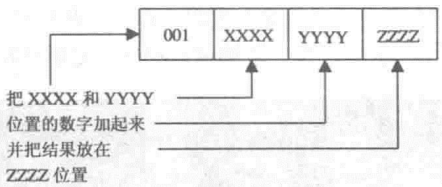

图 1.3 高度简化的机器语言指令格式

最后，就可以为计算机编写程序，程序代码片段如下所示。

```
001000000010010
110001000011010
011101000010000
111000100010000
001000010100010
```

很明显，没有人愿意选择用这种语言来给计算机下指令。首先存在这样一个问题，即计算机能够执行的运算集非常有限。在上面的样本计算机中，它只能执行 8 种可能的运算，而现代计算机可以执行的

基本运算达到几十种，这主要是因为这种计算机十分昂贵，所以难以设计并制造功能更多的计算机。这就意味着为了表达“把内存位置 0~1000 的数字进行排序”这样的问题，我们不得不使用计算机已经提供的有限的简单运算，例如 COMPARE 和 MOVE 等运算。

上述方法好像并不很坏，但实际上这种方法根本是不可能的，因为即使正确设计出使用的指令以及这些指令的顺序，也必须产生所需的上百条机器语言指令，并且一次也不能把 0 和 1 的位置放错，反之亦然。程序中不但不可避免地会包含一些小错误，每一个错误都可能导致程序失败，而且在无数的 0 和 1 中，要跟踪产生的错误也几乎是不可能的。

从长远来看，另一个非常重要的事实是机器语言由计算机设计时规定，这就意味着机器语言程序的“可移植性”并不比一套假牙好到哪里去。不同的机器使用不同的语言，我们对此无能为力。如果某个公司购买了一台新机器，那么为了能够在新机器上运行，所有旧程序都必须从 0 开始重新编写；如果 Des Moines 办公室有一台机器的型号与在费城使用的机器不同，则根本无法通过共享程序削减开支。这很好地描述了 20 世纪 50 年代的程序设计艺术，实际上是一种非常遗憾的状况，因此必须寻找一种更好的办法解决上述问题。

1.1.2 救星的到来

在早期，编写程序需要近乎野蛮的精确度，对细节的掌握要一丝无误并且能够抵抗工作上的厌烦和挫折。而现代的计算机具有超人的精确度，只要编写正确，那么它们在跟踪细节方面就毫无问题，而且它们天生就对厌烦和挫折有免疫力。不久，计算机科学家就提出寻求专门机器负责解决这种问题的想法。

因为组成程序的指令集和其他数据一样，可以存储在内存中，因此计算机可以像操纵数字、字符串、图片、声音和其他可以用二进制表示的信息一样操纵程序指令。洞察到这点后，人们就可以从结构支配的机器语言中解脱出来，设计一种程序设计人员可以非常容易操作的语言。

简化程序设计过程的最早期阶段就是对机器语言程序设计人员手工使用的技术进行简单扩展。假设此时要编写机器语言程序，那么就像用 0 和 1 代表 OFF 和 ON 开关一样，我们也可以开发一些简单的记忆方法代替二进制符号，例如“用 A、B 和 C 代表 0000、0001 和 0010 内存位置，为了实现‘ADD A 和 B(即 A + B)’并把结果存在 C 中具体应该怎样做”。实际上，我们可以把这些内容写下来，像下面形式那样对程序进行描述并注释。

```
ADD A B C ; C包含A + B
MUL C C D ; 现在D包含(A + B)*(A + B)
```

这比二进制版本理解起来要容易得多，现在要做的就是按照上面的形式编写字符格式的(机器语言)程序，然后一行行地把程序转换为相应的二进制指令。如果在这种汇程序设计序中出现 ADD 字符，那么就会产生 001 运算代码。当第一次发现 A 时，汇编程序会寻找没有使用的内存地址，例如 0000，并将之添加到已经产生的运算代码后，然后按照相似的办法依次执行后面的程序。汇编程序中分号到行尾的所有字符都认为是注释信息，在转换过程中都将被忽略。

现在程序设计工作对于人来说越来越简单，而对于机器来说则越来越复杂。因为程序设计人员可以用比较简单的汇编语言编写程序，把程序输入汇编器中，然后得到可执行的机器语言作为输出结果，以便计算机可以执行。虽然这个过程对程序设计人员来说或多或少是不可见的，但是实际上可以把汇编语言程序想象成在虚拟机中执行的程序，虚拟机专门用于执行汇编语言程序，而不是二进制程序。

程序设计语言发展的下一个阶段(目前仍在继续)是发明对人类来说更容易使用的语言，即高级程序设计语言。显然，人们比机器更乐意在更高级语言的环境中工作。例如，上面的例子可以用一条指令 D =(A + B)**2 来表示，此处代数符号“( )”和“+”有各自常规的定义，“**”表示二次幂，“=”解释为“计算右边的表达式，并把结果存在左边命名的位置上。

当然，从高级源代码转换为机器语言对象代码的过程是极其复杂的(如图 1.4 所示)，因为源代码中的一条语句可能会转换为对象代码中十几条甚至更多的指令。例如，1954 年提出的 FORTRAN(FORmula

TRANslator)语言是发明的第一批高级语言之一，它花费了 18 个人几年才完成了转换器程序。从那时起，人们就学到了许多语言设计和转换过程方面的知识，以至于为高级语言设计转换器目前成为计算机科学研究所长达半年的研究项目。

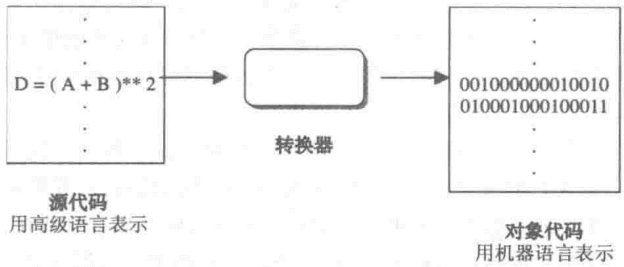

图 1.4 使用某个程序进行程序转换(根据现实情况进行了简化)

由于稍后将频繁地用到这些术语，因此此处首先介绍两种从高级语言转换为机器代码的方法。解释器把单条源代码语言转换成等价的机器代码后，执行此机器代码，然后移到下一条源代码语句并进行转换执行，依此类推。编译器对整个源代码程序进行转换并生成稍后可以执行的对象代码文件。解释型程序比等价的编译型程序运行速度要慢许多，因为一方面，程序执行过程中要穿插着解释过程；另一方面，编写解释型程序要比编译型程序容易许多，在解释型环境中调试程序(寻找和修复错误)要比在编译型环境中容易。LISP 和 BASIC 语言属于解释型语言，像 C++等更加复杂的语言属于编译型语言，稍后就会看到，Java 属于先编译后解释型语言。

高级语言把人类从底层机器指令中解脱出来，语言设计人员就可以集中设计尽可能容易使用的语言，惟一受到的约束就是如何把它转换为机器代码(至少在现在，这是不能用英语编写程序的原因)。研究发现，程序设计人员一天产生的调试代码行数与使用的语言是完全独立的，工作的语言级别越高，效率也就越高，这是因为 200 行的 Java 程序可以很好地转换为 1000 行或更多的机器代码指令。使用高级语言时还有另一个优点，即虽然 Sun 工作站、基于 Pentium 的 IBM PC 和 PowerPC Macintosh 机上使用的机器语言完全不同，但同样的 Java 源代码可以在这 3 种机器上运行，只要每种机器安装了合适的转换器程序即可。而对程序设计人员来说，所有这 3 种机器只不过是登录窗口不同的 Java 机器而已。

1.1.3 问题回顾

可以在每章的最后一节(本章为 1.9 节)找到问题的答案，例如 1.4 小标题后面的答案(1)指的是 1.4 节第 1 个问题的答案，依此类推。

(1) 信息的“二进制表示法”指的是什么？

(2) 3 个二进制位可以表示几个数值？

(3) 汇编程序是什么？为什么发明汇编程序？

(4) 解释编译器和解释器的区别。

1.2 Internet 和 WWW

20 世纪 60 年代，计算机在全美国的政府机构和科学实验室内已经非常普遍了。虽然计算机越来越强大，程序设计语言越来越复杂，但是人们的兴趣集中在另一个方面。意识到计算机在研究领域和行话所说的 C3 领域(指挥、控制和交通)的重要性后，美国国防部提出把政府机构、研究团体和大学的计算机连接在一起组成网络。因此，现在所称的 Internet 在 1969 年的劳动节诞生，它和 UCLA、Stanford 研究院所、位于圣巴巴拉的加利福尼亚大学，以及位于盐湖城的犹他大学的主机连接在一起。

Internet 开始设计时是从安全和稳定性方面考虑的，国防部要求此网络可以抵抗“突然袭击”，即如

果出现核攻击，几台主机的损坏不能危及网络其余部分的通信能力。这个要求具有非常重要的、无法预测的结果。它意味着网络不应该像现在的电话网那样，具有集中的交换中心，负责路由所有消息。相反，被提议的网络应该分散化布置，每台主机负责路由自己接收的信息，不管该信息是用于和主机相连的局域网的计算机，还是把信息传递到其他位置。

分散化布置意味着可以非常简单地向网络中添加其他站点，具有局域网的区域可以很简单地把某台计算机指定为 Internet 主机，安装所需的路由软件，并把这台主机与同意此连接的其他站点的一台或多台计算机连接在一起。这种连接对本地计算机来说或多或少是透明的，即使 Internet 是由路由计算机连接而成的局域网集合，但是对于本地机器来说，仿佛是直接与网络中的其他计算机连接在一起一样。

因为路由计算机负责管理网络流量，因此 Internet 上的每次通信实际上都是本地呼叫。从 Internet 开始之初，距离在 Internet 上就是毫无意义的概念。为了从旧金山向帕萨迪纳发送消息，路由计算机可能确定最有效的路径(按照当前网络流量)是从旧金山经卫星到达挪威，然后经过陆路到达伦敦，再经过卫星到达西弗吉尼亚，最后经过电缆和微波到达帕萨迪纳。

事实证明，不管身在何处，与 Internet 进行连接都是非常容易的，这导致 Internet 以惊人的、不可预料的速度发展。随着电子邮件的火爆(网络又一个没有预料的用途，原来是为工程合作而设计的)，每年主机的数目都会翻一番，1984 年达到 1024 台，1987 年达到 28 174 台，到 1992 年超过 100 万台。当写这本书时，已经有 800 万台主机为世界各地的 5000 万用户服务，每个月的总流量超过 30 万亿字节(1 字节等于 8 位，是为单个字符编码所需的位数)，这个流量大致相当于每月传送一次图书馆的全部内容。尤其是 1973 年考虑网络流量时，Internet 的设计者预期站点的总数最大为 256 个是给人印象最深的。

1.2.1 WWW

早期的 Internet 主要用于电子邮件、新闻组、远程登录以及访问存储在其他站点文件中的信息。当时使用 Internet 需要相当的技术，而且也是非常令人气馁的过程。把信息从一台计算机传送到另一台计算机不但需要掌握一种或多种软件，甚至在设计用于搜索所需数据的程序帮助下，定位第一点信息也决不容易。Internet 无序、分散的本质意味着不存在可以查找诸如“疣猴”的信息的集中目录。

为了解决这些问题，CERN(欧洲粒子物理研究所)的研究学者在 1989 年提出了分布式超媒体的概念。超媒体系统是把文本、图像、动画和声音组合在一个文档中的系统。有关“疣猴”的超文本文档可能包含其生活环境、物理特征和社会行为的文本描述，以及它们群居的图片、呼叫的声音和威胁行为的视频剪辑。如果你已经看过计算机方面的百科全书，那么或许熟悉超媒体。当然，超媒体的概念早在 1989 年以前就提出来了，CERN 提出的超媒体变得如此重要是因为它在超文本文档中提出了链接到其他文档的思想，这些文档甚至可以存储在其他计算机中。人们使用分布式超媒体术语的原因是：网页或许看起来只是一个虚拟的实体，但是它包含的信息可以来自分散在全球计算机中的许多文件。

这项工程很明显是一项非常重要的事业，对于二进制表示的文本、图片和声音，不仅需要标准支持，而且还需要称为浏览器的程序，此浏览器支持所有这些信息的显示以及处理 Internet 上其他计算机调用超文本文档的过程。截止到 1993 年，第一个浏览器得到发展并诞生了 WWW，到 1993 年 1 月大约有 50 个站点支持超文档链接；只不过 9 个月时间，Web 信息就占到所有 Internet 流量的 1%；到 1994 年 6 月已经有 1500 多个站点，现在 Web 站点的数量已经超过 100 万。

有一点需要重点指出，与 Internet 或纯粹的内联网(称为 intranet)不同，WWW 不是物理实体，它只是一个概念，指的是文档、链接以及用于查看这些内容并与之进行交互的浏览器。从某种意义上说，Web 位于 Internet 等计算机网络之上。

我们很难高估 WWW 的影响力，因为它出其不意地使每个用户都变成信息源。用户需要的就是访问 Web 服务器(具有合适软件和 Internet 连接的计算机)，世界各地的人都可以访问用户自己的 Web 页面。生成和访问 Web 页面是如此轻松，使得穿过全球网络的页面达到上百万，而且所有的网页都紧密地连接在一起。随着搜索引擎的发明，信息访问变得更加容易。Yahoo、Lycos 和 Excite 等程序不停地从一个链接

跳到另一个链接，利用空闲时间记录在数据库中发现的索引信息，帮助用户处理查询。为了寻找“疣猴”的信息，用户不需要知道信息位于何处，而只需要定位到某个搜索引擎，告诉它寻找“疣猴”关键字。搜索引擎接着就会在数据库中寻找，并返回包含“疣猴”单词的所有网页的 Net 地址。为了轻松一下，我们在几个搜索引擎中查找“疣猴”，结果发现包含“疣猴”的网页数目从 50~709 个站点不等。在短短几分钟的时间内就找到了足够的文本、图片、视频剪辑和声音，这样可以满足任何人对“疣猴”的好奇心。

1.2.2 HTML

WWW 最优秀的特征之一就是，为了生成 Web 页面，用户所需的只不过是一台计算机、一个文本编辑器以及一个浏览器(如果不在意创建的 Web 页面显示时具体是什么样子，那么可以省略浏览器)。设计 Web 浏览器的目的就是让其识别称为 HTML(Hypertext Markup Language，超文本标记语言)的纯文本语言，HTML 的要素称为标记，其中包含一些关键字和其他包含在尖括号“<>”中的信息，不是标记的其他任何文本都看成纯文本，它在浏览器中将适当显示。

HTML 的重要特征之一就是学习起来非常容易，其中只有为数不多的标记需要学习，大多数标记非常简单，不需要加以说明。例如，如果把文本包含在<B>和</B>标记之间，那么包含的文本将显示为黑体。如果书写下面的 HTML 代码。

```
This is <B>really</B> easy.
```

那么浏览器发现此标签后，将显示如下的代码。

```
This is really easy.
```

统计数字显示，大部分用户都已经使用过 Netscape Navigator 或微软的 Internet Explorer 浏览器上网浏览信息。当浏览信息时，其中发生的过程既简单又复杂。如果让浏览器去如下网址(我们的主页之一)，

```
http://www.cs.hamilton.edu/~rdecker/
```

那么浏览器就会通过网络向 Web 服务器计算机 WWW 发送信息，它位于所有教育机构 edu 域的 cs.hamilton 子域中。此信息会告诉主机在~rdecker 目录中寻找 HTML 文档 index.html(因为在网址中没有指定此文档，这是浏览器假定的 HTML 文档)，然后把文档拷贝发送回用户计算机。一旦找到文档，浏览器就会读取文档，解释标记，并在显示器上按照文档规定的格式显示所有信息，如图 1.5 所示。从理论上说，上述过程是非常简单的。

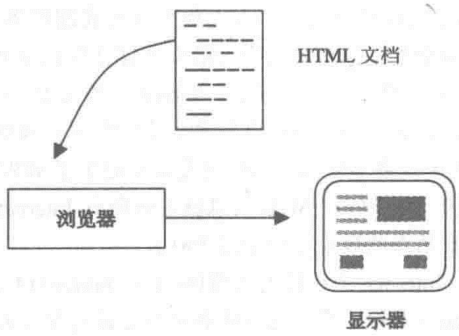

图 1.5 Web 浏览器显示的 HTML 文档

但是实际上取得文档、解释标记并显示结果的过程是相当复杂的。例如，决定使用哪种字体显示纯文本，或者设置页面大小使其能够满足显示器的要求，所有这些都要由编写 Web 浏览器的程序设计人员决定。这种复杂性一部分是因为 HTML 是独立于平台的标准，只要用户计算机中有专门设计的浏览器，那么谁制造的计算机根本不是问题。HTML 标准可以保证 Web 页面按同样的信息进行交流，即使在不同

的浏览器中也仅会有很小的差别。

多数浏览器都为用户提供了检查任何文档底层 HTML 代码的功能，用户可以使用此功能查看生成 Web 页面的 HTML 源代码。用户可以尝试使用 Web 浏览器打开 Rick 页面，查看源代码。

尤其要注意下面 4 行标记。

```
<IMG
    src = "common/graph.gif"
    align = right
    width = 258
    height = 169>
```

这几行指示浏览器定位到最初存储 HTML 文档的计算机，在 common 目录下寻找 graph.gif，并使此图片与浏览器窗口的右边对齐。如果用户喜欢，还可以把这幅图片拷贝下来，用于自己的 HTML 文档中。所有这些都是为了说明，通过浏览器在 WWW 中寻找其他计算机中的信息并下载信息是非常容易的。在以后的章节中将大量使用此功能，用于寻找在线的 Java 信息，以及寻找其他人已经编写好的 Java 程序。

1.2.3 问题回顾

(1) 您的年龄比 Internet 大还是小？与 WWW 相比呢？
(2) 什么是浏览器？
(3) 什么是“平台独立性”，它为什么非常重要？
(4) 什么是 HTML？

1.3 Java 的诞生

对于所有视觉和听觉上的吸引力，除了为用户提供单击链接，查看存储在世界各地的文档这一重要功能外，HTML 文档在浏览器中看起来仍然是功能有限的、相对静止的对象。WWW 提供了对难以想象的巨大信息库的访问，但是至少从当初设想来看，Web 页面没有提供过多与用户交互的方式。现在，所有这些都发生了巨大的变化，很大程度上是因为与 Web 同时诞生的一项工程。

1.3.1 智能烤面包机

1990 年在加利福尼亚州的 Sun Microsystems 公司，James Gosling 正在开发一种新的程序设计语言。与其他传统语言不同，这种语言主要用于烤面包机、微波炉和电视机等家用电器的控制系统。由于这类家电的本质以及涉及到的经济学原因，原来用于数据处理和科学使用的标准语言，例如 C++ 语言并不适合在此处使用。对解决这类问题来说，这些语言实在太大了，当时的多数语言原本就是为工业程序设计而设计的，因此主要目的是为了使程序员的程序设计更加容易，而根本不考虑其中涉及的计算成本；但是对于许多语言来说，运行语言所需的内存和处理芯片的花费，已经超过了家电本身的花费。尽管使用这类语言可以处理下面的任务，例如设置微波炉以全功率加热 3 分钟，然后以半功率加热直到温度探测器测量的温度达到 185°F(华氏温度)，然而这种方式实际上等价于租一条核动力航空母舰进行为期一天的钓鱼旅途，即大材小用。因此，我们需要的语言应该从零开始设计，应该快速、小巧、可靠，并且普遍。

为了解决这个问题，传统的解决方案在数字表和电子计算器上已经徘徊了 20 年。例如微波炉制造商决定使用廉价的处理器芯片，并将其内置于家用电器里，雇人编写机器代码形式的程序，但是这种方法存在几个问题。首先，机器语言就像前面提到的那样，机器不同代码也不同。如果制造商想更换控制处理器的牌子，就不得不重新购买新的机器代码，原有的代码当然与新的处理器不再兼容。这也使得更新或修理旧家电变得非常困难，因为制造商将不得不储存一些旧的控制处理器，而且微处理器被新的型号取代的速度要比家电磨损的速度快。

Gosling 和他的同事认为，解决方案应该是发明一种比较小的语言，这种语言不应该比机器语言小或

者难以使用，可以在任何计算机芯片上运行。就像 1.1 节中讲到的，惟一的方法就是发明一种容易编译的新语言，使用这种语言时，硬件就不再是个问题。制造商如果想更新控制处理器，那么可以使用手头上的同一个程序，惟一需要的就是购买或设计新的编译器，将程序编译成机器代码，以便用于新型号芯片。实际上，程序员可以只编写控制代码一次，而根本无须考虑此程序最终运行在何种芯片上。

1.3.2 相得益彰的 Web 和 Java

这种新语言第一次用于电视机、电灯和电话等家用电器的主控制器中。此类产品并没有将这种语言束之高阁，不久设计组又提出了能够很好地与 Java 速度、简单性和稳定性相匹配的“虚拟家电”的概念，即 Web 页面。为什么不把 Java 代码嵌入到 HTML 文档中呢？因为 Web 页面是独立于设备的，Java 也独立于设备(在基于 Intel 的 PC 机、 Macintosh 机以及 Sun 工作站上浏览 Web 页面，所需的只不过是浏览器而已)，而且 Java 是如此简单，以致于可以把 Java 解释器嵌入到浏览器中，而不需要太多额外的程序设计，所有这些只需要得到浏览器开发人员的许可即可。1995 年当 Netscape Navigator 2.0 宣布支持 Java 时，这变成了现实。为了评价由此造成的争论，可以参考以下网址。

```
http://www.acm.uiuc.edu/webmonkeys/juggling/index.html
```

使用 Java 语言，Web 页面由于动画和用户交互而变得生动起来。只要正确设计，网页的 Java 部分可以发出指示、通知，可以发笑、进行解释、自娱自乐以及进行重点强调。当然，也可以使 Java 程序令人讨厌、使人愤怒或者激怒观众。我们自然鼓励选择前面的选项，但是对于后者也会充分指出，使得用户可以选择如何处理这类程序。

1.3.3 应用程序和 Java Applet

Java 程序根据最终用途的不同有两种风格，分别是应用程序和 Java Applet。应用程序是传统的独立型程序，先经过编译然后才能运行，例如像下面的应用程序一样。

```
public class Hello
// Just about the simplest Java application anyone can write.
{
     public static void main(String argv[])
   {
       System.out.println("Hi!");
   }
}
```

此程序的核心是包含 main 的程序行，这一行和下面的 3 行程序成为方法。方法只不过是一个有名字的程序块，其中包含指示计算机采取动作的代码指令。本例中，动作指令是下面一行语句：

```
System.out.println("Hi!");
```

这条语句将显示“Hi!”消息。每个 Java 运行程序必须包含主方法，一般称为 main 方法。程序的其他部分非常容易理解，对编译器来说，大括号起的作用就像标点符号对人所起的作用一样，它们负责把代码组织起来。

以“//”开始的程序行是注释行，编译器在进行程序编译时将忽略此注释行，它们的惟一作用是便于用户理解程序代码。最后看一下最上面的一行程序，就会发现这行文字是类的一部分。类是 Java 的基本组成部分。在任何语言中，不管是人类语言还是计算机语言，首次碰到的有些语法规则是毫无意义的。在 1.4 节将重点讨论类的概念。

对于应用程序来说，Java 只不过是众多选择中的一个，上面的程序也可以用其他语言编写。下面是用 Pascal 语言编写的程序。

```
program Hello(output);
{Just about the simplest Pascal program anyone can write.}
begin
    writeln("Hi!");
```