Java 程序设计基础

吴晓东　编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

Java 是当今最流行也是最有前途的面向对象技术。本书将 Java 语言基础和面向对象程序设计方法结合起来，以大量实例详细介绍 Java 的编程方法和编程思想，进而引入 J2EE 技术基础，为读者学习 Java 提供了更加广阔的空间。

本书首先介绍了 Java 语言基础，并通过范例程序逐步加深读者对面向对象概念的理解，帮助读者建立面向对象的思维方式；然后深入介绍 Java 基本类库中比较实用的几个大模块，一方面，使读者通过对本书的学习，学到真正实用的技术，另一方面，为引入 J2EE 技术打下良好的基础；最后，通过电子商务方面的实例介绍 J2EE，尤其是 JSP(Java Server Pages)技术，使读者能够运用最新技术，解决最实际的问题。

本书以实用为主要导向，适于初学者学习，也可作为高等院校非计算机专业的教学参考书。

书　　名：Java 程序设计基础
作　　者：吴晓东　编著
责任编辑：马　丽
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
　　http://www.tup.tsinghua.edu.cn
印 刷 者：世界知识印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　印张：22　字数：512 千字
版　　次：2002 年 5 月第 1 版　2002 年 7 月第 2 次印刷
书　　号：ISBN 7-302-05437-1/TP・3204
印　　数：5001～8000
定　　价：25.00 元

前　言

随着人们计算机应用水平的迅速提高，社会生活中依赖于计算机解决的问题越来越多，当然，也越来越复杂。传统的面向过程的思维模式和编程技术越来越难以描述和解决面向商业化应用的复杂逻辑和复杂问题，网络的迅速发展更是对分布式计算及平台无关性提出了更高要求，能够满足这些要求的解决方案和程序语言才能在未来的信息社会中生存下去。

面向对象和网络计算的成功范例——Java 正是能够满足这种需求的一种技术。

本书将以 Java 语言为出发点，通过大量实例向读者详细介绍 Java 语言及其编程设计方法，进而引入当前最流行同时也是最有前途的 J2EE(Java 2 企业版)，它是目前 Java 的最新版本，也是最被看好的电子商务应用技术，在本书的最后我们将进行详细地介绍。希望能够为读者从一个普通计算机用户提高到具有 Java 程序员甚至系统设计员水平提供有益的帮助。

随着计算机应用技术的不断加深和电子商务时代的到来，现代程序员的主要任务集中在对商业逻辑和应用逻辑的处理上，而不需要在一些局部算法甚至底层处理上浪费时间。因此，本书的编写原则首先是深入浅出，保证一个普通计算机用户能够学懂 Java，其次就是要对一些实用技术有一定的了解，所以在这本书的最后还有一些 JSP(Java Server Pages)的介绍。

本书以 Java 提供的基础环境 JDK(Java Development Kit)为背景，让读者在学习许多实用程序的同时，逐渐培养自己的面向对象思维方式，进而了解 J2EE，直接步入网络编程以及电子商务应用的殿堂，加深读者对存在于自己身边的网络和电子商务的认识，使读者学习完这本书之后，不仅仅是学会了一种语言，而且能够在一定程度上掌握面向对象的思维方式，并且有能力编写真正有实际意义的应用程序。

本书第 1～4 章介绍了非面向对象语言也同样使用的基本编程知识；第 5、6 章介绍面向对象知识；第 7 章介绍例外处理，第 8～10 章介绍 Java 类库及某些库的使用；第 11、12 章分别介绍图形界面和多线程；第 13 章是 JSP 技术简介。

计算机是注重实践的学科，尤其是对编程语言的学习，更应注重练习。我们在这本书中精心设计了一定量的习题，供读者练习，在边学边练的过程中轻松地掌握这门语言，同时也为了增进学习者的兴趣。

如果您通过对本书的学习，成为一个 Java 爱好者，您不妨常去 Sun 公司的 Java 网站看看，网址为 http://java.sun.com，那里有太多的好东西会让您爱不释手。

鉴于编者的水平有限，错误之处在所难免，敬请广大读者批评指正。

目　录

第 1 章　程序设计概述

本章主要针对学习程序设计语言之前需了解的一些概念进行介绍，继而引导大家对面向对象和 Java 有一个初步认识，同时介绍 Java 的开发和运行环境。

学习重点：

- 面向过程与面向对象
- 面向对象中的几个基本概念
- 建立 Java 的运行环境

1.1　程序设计语言的分类

程序设计语言是学习计算机技术的基础，它经历了较长的发展过程，也有许多不同的分类方法，下面将介绍几种对学习 Java 比较有帮助的分类。

1.1.1　按发展过程分类

按计算机系统发展的历程来看，程序设计语言大致可以分为以下 4 种。

1. 机器语言

机器语言是以二进制代码的形式组成的机器指令集合，不同的机器有不同的机器语言，存储安排也由语言本身控制。这种语言编制的程序运行效率极高，但程序很不直观，编写很简单的功能就需要大量代码，重用性差，而且编写起来效率比较低，很容易出现错误。

2. 汇编语言

汇编语言比机器语言直观，它将机器指令进行了符号化，并增加了一些功能，如宏、符号地址等，存储空间的安排由机器完成，编程工作相对机器语言有了极大的简化，使用起来方便了很多，错误也相对减少。但不同的指令集的机器仍有不同的汇编语言，程序重用性也很低。

3. 高级语言

高级语言是与机器不相关的一类程序设计语言，读写起来更接近人类的自然语言，因此，用高级语言开发的程序可读性较好，便于维护。同时，由于高级语言并不直接和硬件相关，其编制出来的程序可移植性和重用性也要好得多。常见的高级语言有 Pascal、C 和 Basic 等，现代应用程序设计多数都是使用高级语言。Java 就是高级语言的一种。

4. 第四代语言

一种还未成熟的语言。它具有一定的智能，更接近于日常语言，它对语言的概括更为抽象，从而使语言也更为简洁。

1.1.2 按执行方式分类

我们可以把程序按照程序的执行方式分成两大类。

1. 编译执行的语言

编译执行是在编写完程序之后，通过特定的工具软件将源代码经过目标代码转换成机器代码，即可执行程序，然后直接交操作系统执行，也就是说程序是作为一个整体来运行的。这类程序语言的优点是执行速度比较快，另外，编译链接之后可以独立在操作系统上运行，不需要其他应用程序的支持；缺点是不利于调试，每次修改后都要执行编译链接等步骤，才能看到其执行结果。当然，有些集成开发环境可以提供单步追踪等工具，方便程序员调试程序。另外，这类程序设计语言的编译器与机器之间存在一定的依赖性，不同操作系统需要的编译器可能不相同，因此，在一个系统上编译的程序到另外一个系统上并不一定能够运行。常见的编译执行的程序语言有 Pascal 和 C 等。

2. 解释执行的语言

解释执行是程序读入一句执行一句，而不需要整体编译链接，这样的语言与操作系统的相关性相对较小，但运行效率低，而且需要一定的软件环境来做源代码的解释器。当然，有些解释执行的程序并不是使用源代码来执行的，而是需要预先编译成一种解释器能够识别的格式，再解释执行。例如，我们在本书中要学习的 Java 就是这样的一种语言。不过，解释执行的语言相对来说调试比较方便。常见的解释执行的程序语言有 Basic 和 Java 等。

1.1.3 按思维模式分类

程序设计语言总是需要以某种思维方式进行设计和实现，因此不同的语言可能有不同的思维方式。目前存在两种思维方式。

1. 面向过程的程序设计语言

所谓面向过程就是以要解决的问题为思考的出发点和核心，并使用计算机逻辑描述需要解决的问题和解决的方法。针对这两个核心目标，面向过程的程序设计语言注重高质量的数据结构和算法，研究采用什么样的数据结构来描述问题，以及采用什么样的算法来高效地解决问题。在 20 世纪 70 年代和 80 年代，大多数流行的高级语言都是面向过程的程序设计语言，如 Basic、Fortran、Pascal 和 C 等。这类语言面向求解问题的过程，而不依赖于计算机硬件，可移植性相对较好，在计算机所要解决的问题还不是非常复杂、使用的范围还不是非常广泛的条件下，是非常有效的解决问题方法。但面向过程的程序设计语言有一

个致命的缺点，它极度面向过程，即使需要解决的问题发生微小的变化也会对程序本身产生很大的影响，也就是说需要程序员对程序做较大的改动。而且，不同的问题需要不同的程序解决，问题与解决几乎是一对一的，以往的成果很难直接利用。因此，其可维护性和可重用性都比较差。随着计算机应用范围的迅速扩大，面向过程的程序设计语言的缺点就更加明显地暴露出来。而解决这些问题的最有效方法就是另外一种面向对象的思维模式。

2. 面向对象的程序设计语言

面向对象(Object Oriented)不仅仅是一种程序设计语言的概念，应该说是一种全新的思维方式。简单地说，面向对象的基本思想就是以一种更接近人类一般思维的方式去看待世界，把世界上的任何一个个体都看成是一个对象，每个对象都有自己的特点，并以自己的方式做事，不同对象之间存在着通讯和交互，以此构成世界的运转。用计算机专业的术语来说，对象的特点就是他们的属性，而能做的事就是他们的方法。关于面向对象中的各种概念我们在本书后面会有深入的讲解，需要在这里提醒大家的是，我们学习 Java，其语言规则只占很小的一部分，而要学习的主要内容是如何以面向对象的观点去分析问题和解决问题，这是一个程序员的基本功。常见的面向对象的程序设计语言包括 C++和 Java 等。

面向对象方法大大提高了程序的重用性，而且从相当程度上降低了程序的复杂度，使得计算机程序设计能够对付越来越复杂的应用需求。其中最为突出的是 Java 语言，以其严谨、可靠和跨平台性成为现代程序设计，尤其是网络应用程序的主流语言。广大的底层软件供应商、解决方案供应商等都纷纷推出支持 Java 的平台，使得 Java 语言得到更广泛的应用和发展。如今，在电子商务时代，以 Java 2 企业版(J2EE)为主的模型更是成为一种事实上的电子商务平台的服务标准。因此，学习和掌握 Java 语言和面向对象技术在电子商务时代将大有用武之地。

1.2　Java 简介

在所有的面向对象程序设计语言当中，Java 是最纯粹、结构最清晰的一种语言，它严格遵守着面向对象的绝大多数思想和理念，是学习面向对象思想最好最有效的一种语言。同时，Java 又具备构造非常复杂应用的能力，现在以及未来的很多应用都将基于 Java 来开发，它的发展前景是不可估量的。

1.2.1　Java 的历史和主要特点

Java 诞生于 20 世纪 90 年代，其前身是 Sun 微系统公司开发的一种智能化家电语言 Oak。到了 1993 年，万维网得到了迅速发展，但当时在浏览器中能够看到的页面都是静态的，内容是程序员事先写好的。Sun 公司发现可以利用 Oak 创造动态页面，便开始对 Oak 进行了大规模的改造，于 1995 年推出了 Java。同年，Netscape 公司推出支持 Java 的浏览器 Navigator 2.0，Java 便因此而获得了快速发展的大好契机。如今，Java 本身已经从一种程序设计语言上升成为一类技术，成为网络编程及电子商务系统开发不可缺少的有力工具和平台。Java 之所以能够在计算机和网络技术高度发达的今天占据这样关键性的地位，主要是由以下特点决定的。

- **面向对象**：毫无疑问，Java 语言是一种面向对象语言，而且与其他面向对象语言相比较，Java 严格遵循面向对象理论，是一种严谨、标准和可靠性高的面向对象语言。
- **可移植性**：Java 是一种解释执行的语言，但 Java 源程序并不是直接交给解释器执行，而是经过编译，生成为 Java 解释器能够识别的格式，然后由 Java 解释器解释执行。这样做的目的主要是为了去除语法错误，并引入程序中引用的其他程序(包)。这种 Java 解释器技术上通常称之为 Java 虚拟机(Java Virtual Machine)，它实际上是一组很小的程序，在安装 Java 环境时，就可以把它安装在 Java 运行的计算机系统上，或者已经存在于一些支持 Java 的底层软件上，如浏览器。解释器负责将不同计算机系统运行 Java 的差异隐藏在内部，因此，Java 程序可以运行在任何一种环境中，而不受底层系统的影响。所以，Java 成为当今网络编程的首选语言。
- **结构清晰**：Java 程序具有一个标准的面向对象结构，非常简单易懂，是学习面向对象技术最好的选择。
- **标准性好**：Java 程序所使用的基本类库和运行环境 JDK 是由 Sun 统一发布的，程序员在任何环境中使用的 Java 基本类都是相同的，因此通过 Java 基础的学习，大家就可以进一步深入到各种环境的应用中去，而不像 C++，Microsoft 公司和 Borland 公司的 C++相差万里，Visual C++专家未必能熟练运用 Borland C++，反之亦然。

1.2.2 面向对象的几个基本概念

面向对象的概念不是一两句话就能够解释清楚的，读者的学习也是一个反复理解，逐渐深入的过程，但无论如何总要有一个开始。在这里，我们只简单提几个基本概念，为引入 Java 程序做一个铺垫。看过这一小节之后，读者可能会觉得一头雾水，但这是很正常的，很多东西需要从实践中逐步理解，等看完全书，再来温习这几个概念，才会真正体验到面向对象的“感觉”。

1. 类与对象

所谓对象(Object)就是真实世界中的实体，对象与实体是一一对应的，也就是说现实世界中每一个实体都是一个对象，它是一种具体的概念。而类(Class)是具备某些共同特征的实体的集合，它是一种抽象(Abstract)的概念，用程序设计的语言来说，类是一种抽象的数据类型，它是对所具有相同特征实体的抽象。例如，我们把“轿车”、“公交车”、“卡车”分别看成 3 个相互独立的对象，这 3 个对象有一些共同的特点，如它们都有轮子，都有控制方向的系统，而且它们都会向前、后、左、右 4 个方向移动等等，将这些特点抽象到一起，就得到了一个类——汽车，汽车是一种集合的概念，是抽象的，它可以被具体化。可见，类是对对象的抽象，而对象是对类的具体化，在专业术语中把这种具体化称为实例化(Instantiation)。

2. 抽象

抽象就是把事物共同点抽取出来，以统一的方式进行概要描述的一种过程。这种过程是提高程序重用性的根本原因，正是因为对许多类似的事物进行了抽象，过去已经产生的成果才可以在相同或类似的环境下重用。

抽象是存在不同层次的。例如，上面我们把汽车当成类，把卡车当成车的一个实例，而汽车实际上可以说是一种交通工具，那么如果把“交通工具”当做一个类，这时“汽车”可能就和船、飞机等事物一起成为交通工具的一个实例。因此，在不同的需求环境下，我们到底把什么抽象成类，把什么当做实例或者对象对待，是面向对象设计一开始最关键的一步。如果抽象过度，可能导致程序设计层次加重；但如果抽象不够，在极端的情况下可能会退化到面向过程设计的那种一段程序对应一个问题的情形。

在这里要提醒大家的是，Java 编程过程中，读者将不断地使用各种已经编译好的类，这些类能够实现许多程序设计的基本功能，如输入输出、界面设计等，或者能够实现某类应用的高级功能，如网络通讯、高级界面设计等。这些类都是以类库或者包的形式出现，我们在引用的时候，需要在程序的最开始加入一行“import *libname*”，其中 *libname* 指所引用的包的名称。另外，还可以把自己做的程序打包，创建自己的类库，此部分的内容我们将在后面进行介绍。

3. 属性与方法

我们前面提到，不同对象具有相同特点，就可能抽象为一定的类，那么这些特点基本上可以分为两类，一类是描述对象静态状态的，就是对象的属性(Attribute)，在程序设计中，可以称之为变量(Variable)；另一类是描述对象的动作，就是对象的方法(Method)，在程序设计中，可以称之为函数(Function)。属性和方法是一个对象所具备的两大基本要素，也是我们后面编程工作的核心。举例来说，类“车”或者说对象“汽车”具有一些基本的属性，包括颜色、功率、速度、位置等，而它们的方法则包括前进、后退、加速、减速等。每种方法都像函数一样，有一定的调用格式，利用这些方法，就可以驱动这些对象为我们工作。

4. 封装

那么属性和方法存在的意义又是什么呢？我们说无论一个对象方法的目的是什么，最终都表现为对对象属性的操作，包括把对象的状态告知外界以及改变对象的状态。例如，汽车加速，改变了汽车当前的速度和当前的位置，并可能将这两个值返回给引起汽车加速的对象。因此，只要有足够的方法，就没必要直接去操作对象属性，只要调用这些方法就可以实现要完成的任务，这种现象称为封装，它通过对象方法对其属性的操作把对象属性封装在一个对象内部，对象与外界打交道全部通过其自身的方法来实现，有效地把对象属性隐藏在对象内部。同时，对于外界来说，这些对象方法到底是怎么工作的，我们都不需要去关心，只要知道它能实现的功能以及如何去调用这种功能就可以了，这使得程序的设计和实现有效地分割开，使得方法(函数)之间的依赖性大大降低，对象之间的独立程度也大为提高，从而有效地提升了程序的可维护性。

这里只讲一些简单的概念，在后文中，我们会把更多复杂的面向对象概念和 Java 本身

一起介绍给大家，让大家能有形象、深刻的认识和理解。

1.2.3 Java 的运行及运行环境

这一节将帮助读者了解 Java 程序的编制及运行过程，并在自己的计算机上建立 Java 的运行环境。需要说明的是，Java 有两类程序，即 Java 应用程序(Java application)和 Java 小程序(Java applet)，前者是在命令行中运行的独立的应用程序，而后者需要嵌入 HTML(Hyper Text Mark Language，超文本标记语言)在浏览器中执行。本节我们将分别介绍这两类程序的编译执行方法，并将在后文中详细介绍这两类程序的意义和用法。

1. 编写并执行 Java 程序的步骤

编写并执行 Java 程序的步骤可以分为 3 个阶段。

- 编写源代码，在这个阶段，我们需要一个无格式的文本编写器，例如 Windows 的记事本等。程序编写好以后就可以保存到软盘或硬盘的某一目录下，这个文件称为源程序，存储时，其后缀名必须是.java。如果使用记事本，请注意存储时，先选择*.*的文件类型，然后，在文件名后加上.java 的后缀。建议读者把所有源文件都保存到一个目录下，便于运行。

  另外，向大家推荐一个非常好用的文本编辑工具 UltraEdit，该工具具有强大的行、列编辑功能，是一般的编辑器所无法比拟的，而且，在 UltraEdit 中可以使用自定义的配色文件，从而在编写 Java 程序时，系统自动把关键字、常量、变量等不同元素区分开，有助于程序员减少语法错误，这个软件可从网上下载，使用起来非常简单，这里不作具体介绍。另外，还可用 Java 的集成开发环境的编辑窗口，关于集成开发环境我们后面再进行介绍。在选择编辑器时，大家千万不要使用像 Word 这类带有格式的文本编辑工具，因为它会在看不到的情况下，给文本加入很多格式信息，这些信息是 Java 解释器所不能识别的。
- 编译源代码，虽然 Java 是解释型语言，但它仍有一个类似编译的过程，这和 Basic 不一样。在这个阶段，我们需要一个 Java 编译器：javac.exe。源文件经过编译后生成以 .class 为后缀的同名文件。编译有两个作用，一是可检查程序的语法错误，二是可在此阶段引入 Java 类库中的类。
- 解释执行程序，在这个阶段，我们需要一个 Java 解释器，如果编写的是 Java 小程序，它可以是浏览器，例如 Microsoft 公司的 Internet Explorer 或者网景公司的 Navigitor；如果编写的是应用程序，就需要 java.exe。

到此，我们知道了 Java 的运行过程，并看到第(2)、(3)阶段都需要一些工具，所以，在练习编程之前应在机器上建立起 Java 的运行环境。初学者所需的工具都可以从 Sun 公司提供的 JDK 中找到，并到www.sun.com网站上去下载最新的 JDK。本书以 J2SDK 为例，它是 JDK 的 1.3.1 版，将 JDK 安装到合适的目录下，如 d:\jdk1.3.1 目录下。然后在 Windows 的系统管理工具中设置好环境变量，以 Windows 2000 Professional 为例具体方法如下：

单击【开始】|【设置】|【控制面板】|【系统】|【高级】|【环境变量】命令，打开【环境变量】对话框，在下面的【系统变量】列表框设置系统变量，以便以任何用户身份登录

都能使用。选中 Path 一行，如图 1.1 所示。

单击【编辑】按钮，在已有内容的最后增加 JDK 下 bin 和 lib 这两个目录安装的位置，每一项用分号隔开，图 1.2 中最后两项是添加的。这样做的目的是让 Windows 在任何目录下都能运行 JDK 中的工具，读者可以把 Java 的源程序放在与 JDK 不同的目录下，如 e:\MyJavaProgram\目录下，并可在该目录下直接运行 JDK 中的工具。

图 1.1　Windows 2000 Professional 下的环境变量设置

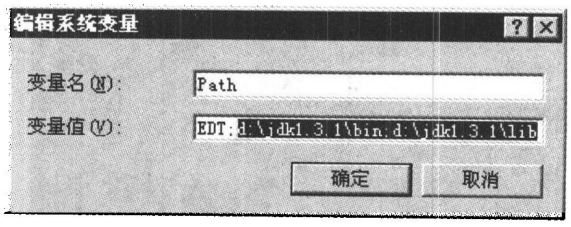

图 1.2　Path 变量的设置

2. Java 应用程序示例

下面是一个以 Windows 下的记事本编辑的 Java 应用程序，并在 JDK 下运行的过程。程序内容将在以后章节中予以解释，读者只需熟悉编译及运行过程。

(1) 编写程序：打开记事本，编辑例 1.1 的程序，保存到某一目录。需要注意的是：Java 是区分大小的，保存的文件名与程序类名相同，其扩展名为.java。

注意： 在记事本中编写 Java 文件，在保存时一定要把文件名和扩展名用双引号括起来，否则将默认保存为文本文件，如果要保存的 Java 文件名为 Program1.java，则在保存时在【文件名】文本框中一定要输入“Program1. java”。

例 1.1　JavaProgram1.java

JavaProgram1.java 程序的内容如下：

```
public class JavaProgram1{
public static void main(String args[ ]){
System.out.println("I have been a programmer!");
}
}
```

(2) 编译程序：单击【开始】|【运行】命令，在命令行上输入“cmd”，按回车键(在 Windows 98 中输入“command”，按回车键)，即可打开一个命令窗口，将目录转换到编写的 Java 源程序所在的目录，输入“javac *filename*.java”，本例应输入“javac JavaProgram1.java”，如果编译正确的话，显示如图 1.3 的结果。

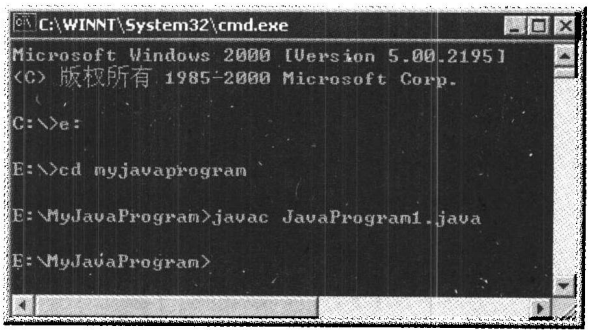

图 1.3 编译 Java 程序

(3) 执行程序：同样在命令窗口中输入“java *filename*”，本例应输入“java JavaProgram1”，运行结果如图 1.4 所示。

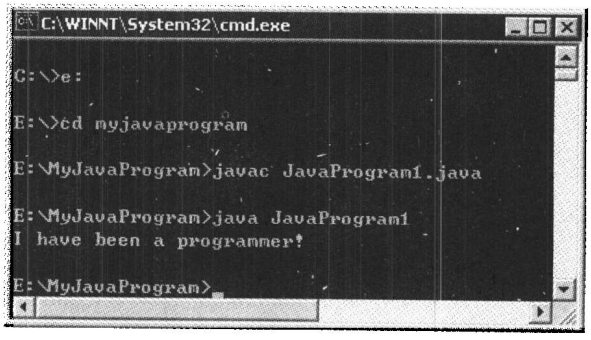

图 1.4 运行 Java 程序结果

3. Java 小程序示例

例 1.2

首先，编辑一个 Java 程序文件。

JavaProgram2.java 程序的内容如下：

```
import java.applet.Applet;
import java.awt.*;

public class JavaProgram2 extends Applet {
    public void paint(Graphics g) {
```

```
        g.drawString("I have been a programmer of Java Applet!",10,20);
    }
}
```

然后，再用编辑器编辑一个 HTML 文件来调用该 Java 小程序。

Applet.html 文件的内容如下：

```
<html>
<title>Applet Test Page</title>
<h1>Applet Test Page</h1>
<applet
    code="JavaProgram2.class"
    width=250
    height=250
    name="MyApplet">
</applet>
</html>
```

使用与例 1.1 中同样的方法编译 AppletProgram2.java 之后，再输入“appletviewer *filename*.html”，在本例中输入“appletviewer applet.html”，即得到如图 1.5 和图 1.6 的运行结果。

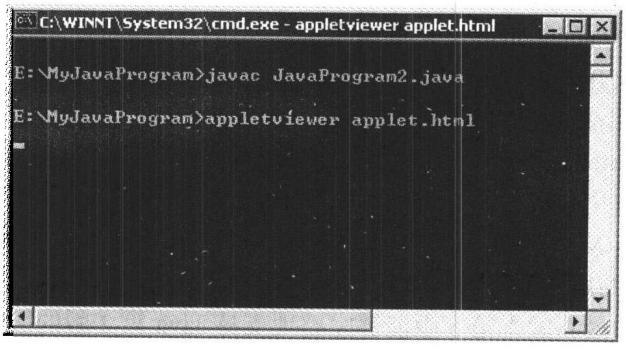

图 1.5　Java Applet 的执行结果

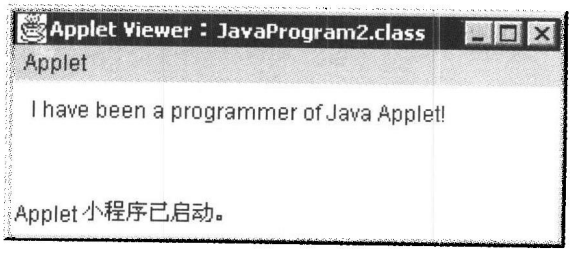

图 1.6　Java Applet 示例结果

假如读者的机器上安装了支持 Java 的浏览器，可以直接使用浏览器调用这个 HTML 文件，也可以直接用鼠标双击 HTML 文件的名字。如果读者使用的是 Microsoft 的 Internet Explorer，可以打开一个新的 IE 窗口，在地址栏中输入 HTML 文件的全路径名，如图 1.7 所示。

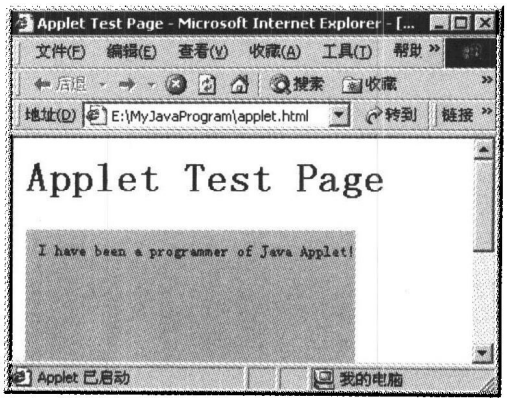

图 1.7 在浏览器中执行 Java Applet

1.2.4 常见的 Java 集成开发环境

如果有一个 Java 集成开发环境，就不需要做上面这些繁琐的事情了。集成开发环境不仅可以提供一个能够识别 Java 语法的编辑器，使读者简单地编译和执行 Java 程序，还可以建立应用程序或者小程序的框架，并调试程序，将做好的程序打包成可执行文件等等。在这里，我们只对集成开发环境进行简单的介绍。

Symantec 公司出品的 Visual Café，据说这是最早出现的 Java 集成开发环境，同时里面还包含了一些 Symantec 自己的实用类库。Visual Café 应该说是一种非常好的 Java 开发环境，惟一的缺点是占用系统资源稍大，如果机器配置很好，就不需要担心这类问题了。

除此之外，还有 Microsoft 出品的 Visual J++以及 Borland 公司出品的 Jbuilder，前者继承了 Microsoft 一贯的风格，内容比较混乱，不太适合初学者使用；后者应该说也是一种不错的工具，而且它的最新版本第 5 版能够完整地支持 Java 第 2 版的所有开发工作，但对系统资源的侵占非常大，通常建议用户有 256 MB 内存。

我们目前只是初学阶段，不需要编写很复杂的 Java 程序，对调试方面的需求并不迫切，使用文本编辑器和 JDK 就足够了。感兴趣的读者可以自己找一些相关软件来试用。

1.3 练 习 题

1. 选择题

(1) 操作系统的主要功能包括：

A. 程序编译

B. 内存管理

C. CPU 管理

D. 数据的组织与维护

(2) 面向过程程序设计的核心是：

A. 商业逻辑

B. 客观实体
C. 算法
D. 要解决的问题

(3) 常见的面向对象的程序设计语言包括：
A. Pascal
B. Fortran
B. C++
D. Java

(4) 符合对象和类关系的是：
A. 人和老虎
B. 书和汽车
C. 楼和建筑物
D. 汽车和交通工具

2. 试验题

在你自己的计算机上安装并配置 Java 运行环境，并编辑运行本书中的两个例程。