张卫民　廖湘科　编著

Java语言与WWW

人民邮电出版社

内 容 提 要

本书主要介绍面向对象的程序设计语言 Java 以及用 Java 语言进行程序设计的方法。全书共分九章，第一章是 Java 概述；第二章介绍 WWW、HTML 与 applet；第三章着重介绍 Netscape 的功能及使用方法；第四章是 Java 程序设计基础；第五章讲述对象、类和界面；第六章介绍 Java 语言的异常处理；第七章介绍 Java 语言的输入/输出流；第八章详细讲述 applet 程序的基础知识；第九章介绍 Java 在 WWW 中的应用。

本书内容丰富、深入浅出、可读性强，适合大专院校师生、程序设计人员和广大计算机爱好者阅读。

Java 语言与 WWW

◆ 编　　著　　张卫民　　廖湘科

◆ 人民邮电出版社出版发行　北京崇文区夕照寺街 14 号
北京北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：14.75
字数：363 千字　　1997 年 1 月第 1 版
印数：1— 11 000 册　　1997 年 1 月北京第 1 次印刷

ISBN7-115-06410-5/TP・409

定价：20.00 元

丛 书 前 言

世界上发达国家普遍重视 发展以计算机和通信为核心的信息技术、信息产业和信息技术的应用，一些经济发达国家信息产业发展迅速。

当前，我国处于国民经济高速发展时期。与此相伴随，必将有信息技术、信息产业和信息技术应用的高速发展。各行各业将面临信息技术应用研究与发展的大课题以及信息化技术改造的大任务、大工程。

为了适应信息技术应用大众化的趋势，提高应用水平，我们组织编写、出版了这套“计算机技术丛书”。这套丛书以实用化、系列化、大众化为特点，介绍实用计算机技术。

这套丛书采取开放式选题框架，即选题面向我国不断发展着的计算机技术应用的实际需要和国际上的实用新技术，选题不断增添又保持前后有序。

这套丛书中有的著作还拟配合出版软件版本，用软盘形式向读者提供著作中介绍的软件，以使读者方便地使用软件。

我们希望广大读者为这套丛书的出版多提意见和建议。

前　言

Java 语言是目前为止推广最快的一种计算机语言，从 1990 年 SUN 公司的 James Gosling 开始设计到现在迅速流行于全球，不过短短的 5 年多时间。Java 从 C++中衍生而来，它继承了 C++的大量语言成分，同时抛弃了 C++中冗余的和容易引起问题的功能，Java 集面向对象、平台无关、稳固性、安全性、多线程等诸多特性于一身，是一种适合于分布式计算的新型面向对象程序设计语言。

WWW 是 Internet 上提供的重要信息检索手段，它能将位于 Internet 不同地点上的相关信息资源以超文本、多媒体的方式有机地编制在一起，从而为 Internet 的用户提供了世界范围的多媒体信息服务。随着 Internet 的日益普及，WWW 将成为人们信息的主要来源之一，Web 页将是人们分布和收集信息的重要方式。

早期的 Web 页主要用来传送静态的用 HTML 语言编制的文档，而 SUN 公司最初设计 Java 的目的是为了开发消费类电子产品，由于 SUN 公司未能将这些产品推向市场，Java 语言也几乎夭折。后来由于 Internet 上 WWW 的迅速发展，SUN 公司审时度势地将 Java 定位到 Internet 中的 WWW 应用开发中，使得 Web 页可以方便地传送被称为 applet 的动态可执行内容。applet 是 Java 与 WWW 结合后引入的重要概念，它是用 Java 语言编写、连入到 Web 页中、由浏览器控制执行、用来产生特殊效果的 Java 程序，借助 applet，Web 页的作者可以设计出具有动画、声音、图像和其它特殊效果的 Web 页。Java 语言与 Internet 结合后，一方面，发挥了 Java 语言的强大优势，使 Java 走上了快速发展的轨道，使其在极短的时间内迅速流行起来；另一方面，Java 语言为 Internet 上的 WWW 带来了具有动态和交互特性的多媒体 Web 页，加速了 WWW 的发展。

本书是针对那些想利用 Java 语言来开发自己的 Web 页的读者而编写，我们在详细介绍 Java 语言和超文本标签语言 HTML 的基础上，着重介绍了 applet 的程序设计方法，即如何将 Java 语言应用于 Web 页的制作中。通过对本书的学习，读者可以掌握 Java 语言的基础知识，applet 的程序设计方法和 Web 页的制作等内容。全书共分九章，第一章为概述，主要介绍 Java 语言的发展历程、Java 语言的特性、Java 语言的运行环境、Java 程序的结构和 JDK 环境等内容；第二章介绍超文本标签语言 HTML、applet 的基本概念以及如何将 applet 引入 WWW 以创建可动态执行和可交互的 Web 页等内容；第三章着重介绍 Netscape Navigator 2.0 的使用；第四章介绍 Java 语言的程序设计基础，主要介绍 Java 语言中与 C 相似的特性；第五章介绍 Java 语言的各种面向对象的特征，主要内容包括对象的使用、类的定义和实现、类的继承、界面的使用等；第六章介绍 Java 语言的异常处理；第七章介绍 Java 语言的各种输入/输出流；第八章介绍 applet 的基础知识，主要内容包括 applet 的生命周期与主方法、applet 程序的组成、Applet 类的方法概述以及 applet 的安全性等；第九章介绍 Java 在 WWW 中的应用，主要内容包括 applet 中的图形、图像、动画等的设计。

本书由张卫民、廖湘科编著，张卫民编著了其中的第一、二、五、六、七章，廖湘科编著了其中的第三、四、八、九章。在本书的编著过程中，得到了国防科技大学杨学军、黄瑞芳、卢宇彤等的大力帮助，在此一并致谢！

作　者

目　录

第一章 概 述

Java 是由 SUN 公司开发的、适合于 Internet 的新型面向对象程序设计语言。Java 将面向对象、平台无关、稳固性与安全性、多线程等诸多特性集于一身，为用户提供了一个良好的程序设计环境。Java 是 C++的衍生语言，它从 C++中继承了大量的语言成分，但抛弃了 C++中冗余和容易引起问题的功能。本章我们介绍 Java 语言的发展历程、Java 语言的特性、Java 语言的运行环境、Java 程序的结构和 JDK 环境等内容。

1.1 Java 语言的发展历程

Java 语言是目前为止推广最快的一种计算机语言，它从开始设计到广泛流行不过 5 年左右的时间。Java 的迅猛发展无疑是得益于它与 WWW 的成功结合，然而 Java 在开发的初期并不是针对 WWW，它最初是 SUN 公司在消费类电子产品开发计划中的一部分，由于 SUN 公司未能将这些产品推向市场，Java 语言也几乎夭折。Java 语言是在被定位到 WWW 上后，才真正焕发了生机，在极短的时间内迅速地流行起来。下面我们简要地回顾 Java 的发展历程。

Java 语言的设计开始于 1990 年。当时，SUN 公司由于感受到 PC 机对工作站市场的压力，想在工作站以外的消费类电子产品方面寻求市场，成立了由 James Gosling 领导的 Green 开发小组，其首要目标是实现一个对家用电器进行集中控制的装置，为此需要开发一系列软件。与其它软件开发项目一样，Green 开发小组开始时也想用 C 或 C++，但他们很快发现，诸如 C 或 C++等语言在可移植性、可靠性和安全性等方面并不能满足他们的要求。为此，在 Green 开发小组成立不久，Jame Gosling 就着手设计一种新的程序设计语言。该语言的设计目标是具有平台的独立性、高度的可靠性和安全性，考虑到 C 和 C++已得到了广泛应用，Jame Gosling 在设计新语言时主要以 C++为基础。最初 Jame Gosling 将这种语言称为 oak(橡树)，后来 Green 开发小组发现，oak 是 Sun 公司以前的一种编程语言的名字，最后被重新取名为 Java。

Java 语言初步设计完后，就在 Green 开发组内部的软件开发项目中使用，首先是开发用于控制家庭环境中的电视机、灯、电话等设备的软件，之后是视频点播(VOD)系统。SUN 公司在这两个项目中使用 Java 语言开发了一些技术上非常成功的试验软件，但由于激烈的市场竞争和其它一些商业上的原因，SUN 公司没能将这些技术上成功的产品推向市场，Java 语言也几乎被束之高阁。

到了 1994 年，Internet 上的 WWW 由于从字符界面发展到了图形界面而获得了迅猛发展，此时 SUN 公司的创始人之一 Bill Joy 介入了 Green 小组的工作，他审时度势，将 Java 定位到 Internet 的 WWW 应用开发上，而且力排众议，让人们免费使用 Java，从而使 Java 走上了快速发展的轨道。Java 开发组一方面对 Java 语言进行改进和包装，使之能更适合于 Internet 的应用开发；另一方面用 Java 语言开发了一个起初被称为 WebRunner 的 Web 浏览器，该浏

览器后来由于商业上的因素被称为 Hotjava。Hotjava 是完全用 Java 语言写的，它是第一个支持 Java applet 的 Web 浏览器。

Hotjava 的开发对 Java 语言的发展是至关重要的。它的意义不仅仅在于使 Java 语言更加成熟，更重要的是向世人展示了 Java，使 Java 走向了世界。当其他程序员看到 Java 开发组正在做 Hotjava 时，许多人也想将这种新技术用到他们的软件开发中。

1995 年 5 月，SUN 公司在 Sunworld 会议上正式发布了 Java 技术。在那次会议上，Netscape 通信公司的创始人之一、现任的技术副总裁 Marc Andreessen 宣布了 Netscape 公司将在其 Web 浏览器产品中支持 Java。随后一些著名的计算机公司，如 IBM、Microsoft、Novell、Oracle、SGI、Borland 等，都纷纷宣布将支持 Java 语言，并购买 Java 的使用许可。Java 正式发布以后，很快就获得了计算机界的广泛好评，美国著名的计算机杂志 PC Magzine 将 Java 评为 1995 年十大优秀科技产品；WWW 的创始人 Berners Lee 说："计算机发展的下一个浪潮是 Java"；Microsoft 公司总裁 Bill Gates 在经过一段时间的观察后不无感慨地说："Java 是长时间以来最卓越的程序设计语言"，并由此调整了 Microsoft 的软件产品开发战略。

1.2 Java 语言的特点

Java 语言有许多突出的特点，本节我们介绍 Java 语言的主要特点。

1.2.1 简单性

一方面，Java 由 C++衍生而来，其基本概念、程序结构和语言风格等与 C++十分类似，如果你已经懂得一些 C 和 C++语言，则很快可以学会 Java 语言；另一方面 Java 又比 C++简单，Java 抛弃了 C++中的一些不是绝对必要的功能，如头文件、预处理器、指针、结构、联合和隐式的类型转换、运算符重载等，并且通过实现自动垃圾收集大大简化了内存管理的工作。因此 Java 比 C++更容易学习，其程序的可读性也更强。

1.2.2 面向对象

Java 是一种完全面向对象的程序设计语言，它除了数值、布尔和字符三个基本数据类型外的其它类型都是对象，Java 的程序代码以类的形式组织，由类来定义对象的各种状态和行为，Java 抛弃了 C++中的非面向对象特性，如结构和函数调用，Java 也不再支持全局变量。在 Java 中，如果不创建新类就无法创建程序，Java 程序在运行时必须先创建一个类的实例，然后才能提交运行。

Java 同样支持继承特性，Java 的类可以从其它类中继承行为，但 Java 只支持类的单重继承，即每个类只能从一个类中继承。其它一些面向对象的语言，如 C++支持多重继承，但这会引起混乱并使语言变得复杂。

Java 支持界面，界面允许程序员定义方法但又不立即实现，一个类可以实现多个界面，利用界面可以得到多重继承的许多优点而又没有多重继承的问题。Java 的界面十分类似于 IDL (Interface Definition Language)界面，构造 IDL 到 Java 的编译器是十分方便的，这就意味着可以在 CORBA 对象系统中使用 Java 来建立分布式对象系统。这种兼容性是十分重要的，因为 IDL 界面和 CORBA 对象系统已被广泛使用。

1.2.3 机器无关的字节码编译

Java 的编译器将 Java 程序编译为字节码，字节码十分类似于机器指令，但字节码又不是为某个特定的机器定义的，因此一般不能在某个具体的平台上执行，而需要由 Java 运行系统中的解释器来解释执行。Java 程序的执行需要经过两个步骤：首先由编译器将 Java 程序编译为字节码；然后由 Java 运行系统解释执行字节码。因此，从本质上讲 Java 语言是解释型语言，但 Java 通过预先将源代码编译为接近于机器指令的字节码，有效地克服了传统解释型语言的性能瓶颈，同时又保持了解释型语言的可移植性特点。

Java 源程序编译后生成的字节码中还包含了符号引用，即字节码中所有方法和变量的引用还都是名字，它们在执行时才被消解。

1.2.4 结构中立

Java 语言的设计不针对某种具体结构。Java 为了做到结构中立，除了上面提到的编译生成机器无关的字节码外，还制定了完全统一的语言文本，如 Java 的基本数据类型不会随目标机的变化而变化，一个整型总是 32 位，一个长整形总是 64 位。像 C 和 C++这样的现代程序设计语言并不满足这一点，不同的编译器和开发环境之间总会有一些细微的不同。

为了使 Java 的应用程序能不依赖于具体的系统，Java 语言环境还提供了用于访问底层操作系统功能的类组成的包，当程序使用这些包时，可以确保它能运行在各种支持 Java 的平台上。下面是 JDK 1.0 版本中提供的系统包：

java.lang	一般的语言包。其中包括用于支持字符串处理、多线程、异常处理和数学函数等的类，该包是实现 Java 程序运行平台的基本包。
java.util	实用工具包。其中包括哈希表、堆栈、时间和日期等。
java.io	基于流模型的输入/输出包。该包用统一的流模型实现了各种格式的输入/输出，包括文件系统、网络和设备的输入/输出等。
java.net	网络包。该包支持 TCP/IP 协议，其中提供了 socket、URL 和 WWW 的编程接口。
java.awt	抽象窗口工具集。其中实现了可以跨平台工作的图形用户界面组件，包括窗口、菜单、滚动条和对话框等。
java.applet	支持 applet 程序设计的基本包。

1.2.5 支持语言级多线程

Windows NT、OS/2 和 Windows95 这些新型操作系统都支持并发，这意味操作系统能同时进行多个任务的并发处理。Java 在语言级嵌入了对并发的支持功能，其支持的机制就是多线程。Java 程序可以有多个执行线程，如可以让一个线程进行复杂的计算，而让另一个线程与用户进行交互，这样用户可以在不中断计算线程的前提下与系统进行交互。

多线程程序设计通常是困难的，其原因在于同一时间内可能发生许多事件，而且它们的顺序是不确定的。在 Java 中提供了一套方便的同步机制，其基本原理是 C. A. R. Hoare 提出并在许多新型操作系统中得到广泛使用的管程和临界区保护规则。Jáva 将这些原理集成到了语言中，使这些规则的使用更加方便。

如果底层的操作系统支持多线程，Java 的线程通常被映射到实际的操作系统线程中，这意味着在多机环境下，用 Java 写的程序可以并行执行。

1.2.6 自动内存管理

用 C 和 C++写软件时，程序员需要仔细地处理内存的使用。当一个内存块不再使用时，必须释放它，这样程序员需要仔细地留意内存的使用情况，在大的项目中这是十分困难的，而内存管理不当通常是造成系统故障和存储空间浪费的主要原因之一。

在 Java 中，程序员不需要关心内存管理的问题，Java 系统内嵌了自动垃圾收集功能，通过扫描内存，能自动地释放不再使用的内存块，这就使 Java 程序的编写变得简单了，而且减少了程序中因内存管理问题而出错的可能性。

1.2.7 稳固性

在 PC 机上编写程序经常会遇到由于程序中的错误而使计算机崩溃的情况。Java 程序也会产生错误，但 Java 在发生错误时，程序并不中断，而是抛出一个异常，并自动寻找相应的异常处理方法。这种异常处理是新型操作系统的错误处理方法，Java 将异常处理引入到语言中，使程序员能用统一的方法处理各种错误。

传统的程序可以访问计算机中的所有内存，程序可以任意改变内存的值，这种方式会引起一系列问题。Java 引入了内存保护机制，Java 程序只能修改被许可部分的内存的值。

Java 取消了指针操作，从而消除了复写内存单元或破坏有用数据的可能性。Java 具有真正的数组和串的概念，即解释器能够检查数组或串的索引值以防越界，而且 Java 程序员也不能将任一整数通过强制类型转换的方法转换成对某一对象的引用。

1.2.8 分布性

Java 是面向网络的程序设计语言。一方面，Java 提供了适合于 Internet 环境的对象连接机制、程序组织方式和名字空间；另一方面，Java 通过提供支持 TCP/IP、WWW 等的网络包，使用户能方便地访问其它 URL 地址上的资源。

1.2.9 安全性

在分布式环境中，安全性是一个十分重要的问题。Java 为了使自己适合于分布式应用的开发，在语言的设计之中考虑了安全性问题。Java 在语言和运行环境中引入了多级安全措施，其采用的主要安全机制有如下几个方面。

1. 内存分配及布局由 Java 运行系统决定

Java 编译器为安全性而提供的一个主要防御线是它的内存分配和引用模型。首先，内存布局不像 C 和 C++一样由 Java 编译器决定，而是延迟到运行时由 Java 运行系统决定，内存布局依赖于 Java 运行系统所在的软硬件平台的特性；其次，Java 并没有传统 C 和 C++意义上的内存单元指针。Java 编译器是通过符号指针来引用内存的，符号指针是由 Java 运行系统在运行时具体解释为实际的内存地址。Java 程序员不能强制引用内存指针，这样 Java 的内存分配和引用模型对于程序员是透明的，它完全由底层的运行系统控制，程序员不能通过修改代码而直接指向一个物理内存布局。通过放弃 C 和 C++的内存布局和指针模型，Java 语言已去掉了程序员直接进行内存分配及布局的能力。Java 的这个特点使 Java 的应用更安全、更可

靠。

2. 字节码验证

Java 编译器虽然保证了 Java 的源代码不违背安全规则，但是像 Hotjava 这样的应用需要从其它地方引入代码段，而 Java 运行系统并不知道这些代码是否满足 Java 语言的安全规则。网络病毒和其它形式的入侵者可以绕过 Java 编译器生成有危险的字节代码，因此 Java 运行系统并不应该信任引入进来的代码，需要对它们进行字节码验证。Java 运行系统中引入了一个字节码验证器，在字节码执行之前，由它进行代码验证。字节码验证器起了一个守门员的作用。具体地说，字节码验证器将每个输入代码段送给一个简单的规则验证程序，以确保代码段遵循如下规则：

- 不存在伪造的指针。
- 未违反访问权限。
- 严格遵循对象规范来访问对象。
- 用合适的参数调用方法。
- 没有栈溢出。

通过语言的内在安全机制，再加上字节码的验证过程，Java 建立了一套严密的安全体系。在字节码检查过程中，字节码验证器遍历字节码，构造类型的状态信息，并且验证所有字节指令的参数类型。

1.2.10 动态特性

Java 的动态特性是其面向对象设计的延伸。Java 程序的基本组成单元是类，而 Java 的类又是运行时动态装载的，这使得 Java 可以在分布环境中动态地维护应用程序及其支持类库之间的一致性，而不用像 C++那样，每当其支持类库升级之后，相应的应用程序都必须重新编译。

1.2.11 高性能

一般情况下，可移植性、稳定性和安全性总是以牺牲性能为代价，解释型语言的执行效率远远低于编译型语言的速度，但 Java 语言很好地解决了这个问题。它在实现了可移植性、稳定性和安全性等特性的同时，又有高性能。Java 编译生成的字节码与机器码十分接近，而且 Java 还有两种提高性能的措施，即及时编译和连入 C 代码。及时编译是指在执行前将字节码编译为机器码。

1.3 Java 运行系统与 Java 虚拟机

Java 的目标代码是字节码，字节码不是直接针对某个具体的平台，在执行之前，需要将字节码转化为本机代码。为了实现语言的动态性与安全性，Java 编译器没有将变量和方法的引用直接编译为具体的内存引用，也没有确定程序执行过程中的内存布局，而是将符号引用信息保留在字节码中，这样类的装载和符号的消解都要在运行时进行。为此，Java 专门为字节码的运行引入了运行系统，由运行系统装载程序运行时需要的类，安排程序运行时的内存布局以及确定方法调用所需要的实际地址，最后也是由运行系统来控制执行代码。

从运行机制来看，Java 中有两类应用程序。一类是 Java 应用，Java 应用是有自己的运行入口点的独立程序；另一类是 applet，applet 是被嵌入在 Web 页面中由 Web 浏览器控制运行的 Java 小程序。无论是 Java 应用还是 applet，在执行时都需要 Java 运行系统的支持。对于 Java 应用，Java 运行系统一般是 Java 解释器；而对于 applet，Java 运行系统一般就是指 Java 兼容的 Web 浏览器。并不是所有的 Web 浏览器都能运行 applet，我们将能运行 applet 的 Web 浏览器称为 Java 兼容浏览器，Java 兼容浏览器中包含了支持 applet 运行的环境。

Java 运行系统一般需要包括以下几部分：类装配器、字节码验证器、解释器、代码生成器和运行支持库。其组成情况如图 1.1 所示。

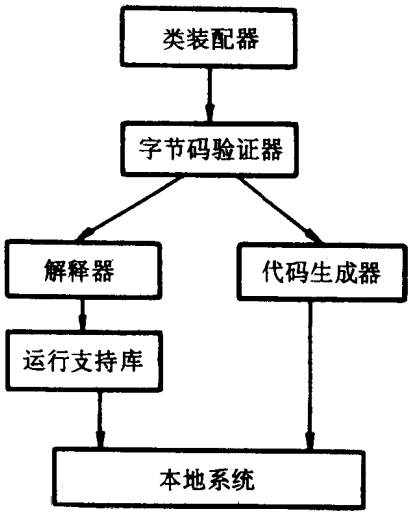

图 1.1 Java 运行系统的组成

Java 运行系统执行字节码的过程可分为三步：代码的装入、代码的验证和代码的执行。代码的装入工作由类装配器完成，类装配器负责装入程序运行时需要的所有代码，其中包括程序代码中调用到的所有类。当类装配器装入一个类时，该类被放在自己的名字空间中，除了通过符号来引用其它类外，该类不能影响其它类所在的空间。本地计算机上的所有类都被安排在同一个地址空间内，而所有从外部引进的类都有一个自己独立的名字空间，这使得本地类通过共享相同的名字空间获得较高的运行效率，同时又保证它与外部引进的类之间不会相互影响。

当装入了运行程序需要的所有类后，运行系统就可以确定整个可执行程序的内存布局，即建立符号引用与特定的内存地址之间的查找表。通过运行时确定代码的内存布局，Java 很好地解决了代码重用中的一系列问题（如当超类改变时，若子类不重新编译，子类就不能感受超类的变化），同时也可以有效地防止代码对地址的非法访问，避免病毒的感染。

随后，被装入的代码由字节码验证器进行安全性检查，以确保代码并不违反 Java 的安全性规则，同时字节码验证器还可发现操作数栈溢出、非法数据类型转化等多种错误。通过校验后，代码便可以提交运行了。

Java 字节码的运行可以有两种方式：

（1）即时编译方式：由代码生成器先将字节码转化为本机代码，然后再全速执行本机代码。

（2）解释执行方式：由解释器通过每次翻译并执行一小段代码来完成 Java 字节码程序的

所有操作。

现在的 Java 运行系统通常采用的是第二种方法。将字节码转化为机器代码具有较高的效率，对于那些对运行速度要求较高的应用程序，Java 运行系统可采用即时编译的方式，从而很好地保证 Java 代码的高性能。

Java 的目标代码在执行时需要有 Java 运行系统的支持。虽然 Java 运行系统被建立在各种不同的平台上，但为了做到 Java 的可移植性，它们的功能要求是统一的，为此 Java 引入了 Java 虚拟机(JVM)的概念。从概念上看，Java 虚拟机是一个想象中的、能运行 Java 字节码的操作平台，它实际上是一组有关指令系统、字节码格式等的规格说明。具体地说 Java 虚拟机定义了一组抽象的逻辑组件，这些组件包括指令集、寄存器组、栈结构、垃圾收集器和存储区五部分。其中，寄存器组包括程序计数器、栈顶指针、用于指向当前执行方法的执行环境的指针和用于指向当前执行方法的局部变量的指针；栈用于提供操作参数、返回运行结果和为方法传递参数等；垃圾收集器用来收集不用的数据堆；存储区用于存放字节码的方法代码、符号表等。Java 虚拟机规定了这些逻辑组件的规格，尤其是对字节码的格式作了明确的规定，但它没有预先规定这些逻辑组件的具体实现技术。这些逻辑组件可以采用任何一种技术实现，如字节码解释、将字节码转化为本地目标机代码甚至于用芯片实现。无论采用什么具体的实现技术，Java 虚拟机的功能必须是统一的，Java 虚拟机也只能执行统一格式的字节码。

在本书中，我们将 Java 的各种运行系统统称为 Java 虚拟机。

1.4 Java 程序的组成

Java 有两类应用程序：一类是由 Java 解释器控制执行的 Java 应用；另一类是被连入到 Web 页中由 Java 兼容浏览器控制执行的 Java applet(以下简称为 applet)。这两类 Java 应用程序在程序组成方面是不同的。

1.4.1 Java 应用的组成

Java 应用是可以独立运行的 Java 程序，正如 C 程序是由函数组成的一样，Java 应用是由一个或多个类组成，并且其中必须有一个类中定义了 main()方法，main()方法就像 C 语言的 main 函数一样是 Java 应用运行时的起始点。下面这个名为 HelloWorld.java 的源程序是一个简单的 Java 应用：

```
/* file:HelloWorld.java */
Public class HelloWorld {
    public static void main(String arg[]){
        System.out.println ("Hello World!");
    }
}
```

该程序的功能是在标准的输出设备(一般是在显示器屏幕)上显示字符串“Hello World!”。熟悉 C++的读者一定不会对这个 Java 程序感到陌生，正如我们在前一章中已经指