新一代编程语言
Java高级教程

乔建忠
林树宽
吕　立
郑　群
编　著

XINYIDAI BIANCHENG YUYAN
JAVA GAOJI JIAOCHENG

辽宁科学技术出版社

新一代编程语言

Java 高级教程

乔建忠　林树宽
吕　立　郑　群　编著

辽宁科学技术出版社
·沈阳·

图书在版编目(CIP)数据

新一代编程语言：Java高级教程 / 乔建忠等编著. —沈阳：辽宁科学技术出版社，1997.3
ISBN 7—5381—2456—X

Ⅰ.新… Ⅱ.乔… Ⅲ.Java语言 Ⅳ.TP312Ja

中国版本图书馆CIP数据核字(97)第01034号

辽宁科学技术出版社出版
(沈阳市和平区北一马路108号 邮政编码110001)
沈阳市第二印刷厂印刷 辽宁省新华书店发行
开本：787×1092 1/16 印张：11 5/8 字数：254,000
1997年3月第1版 1997年3月第1次印刷
责任编辑：刘绍山 韩延本 版式设计：于 浪
封面设计：庄庆芳 责任校对：刘 庶
印 数：1—4,000 定价：20.00元

前 言

这本书经过周密的策划和不懈的努力终于面世了。编者由衷地希望能把 Java 这个计算机业界的宠儿全面地展示给读者，使广大的计算机专业人员和爱好者能由浅入深地了解、掌握这门技术（我感觉也可称为“艺术”），赶上信息时代的浪潮。

本书共分十五章，从概论到 API 全面地介绍了 Java 的特点和具体的程序设计技术，可作为大学教材或自学使用。笔者曾是 C 语言的崇拜者，感觉过渡到 Java 不是一件难事。相信这本书会给读者以启迪和借鉴，帮助读者进入 Java 世界。

Java 语言是 Sun 公司开发的跨平台的高级程序设计语言。它是一种新概念，带来了自 PC 机以来计算机界的又一次技术革命。Java 是“世界语”，它将打破 Wintel（微软的视窗与英特的 CPU 体系）的垄断，在 21 世纪成为计算机业界的主导技术。

关于 Java 语言的优点在书中各章节均有论述，尤其是其结构无关性、安全性、多线程和简单性（相对于 C++而言），给我很深的印象。可就是这样一个优秀的语言，竟差一点被淘汰掉。这里有一段不平常的故事。

James Gosling 是 Sun 公司软件方面的大腕，是著名的 NeWS （ Networked/extensible Window System ）的作者。 NeWS 是一个类似于 X11 的一种透明的、设备无关的操作平台，其设计适应现代网络计算的要求，从技术上来说是一个相当优秀的作品。 Sun 公司发迹于 80 年代初，以 Standford 大学为背景。到 80 年代末， Sun 已脱掉了学院气息，在业界崭露头角，也成为众矢之的。 NeWS 的开发虽很成功，但由 MIT 、 IBM 、 DEC 、 HP 等联手推出的 X11 却成功地占领了市场，将 NeWS 挤到一边，使得 Sun 不得不将 NeWS 改装成 X11 的兼容产品，因此也需要增加技术人员。在新人中有一个叫 Patrick Naughton 的小伙子，刚毕业于纽约北部一个不出名的大学。因在学期间对 MS-DOS 、 X11 和 NeWS 均有研究，被 James 一眼看中，入选了 NeWS 的队伍（ Patrick 曾向微软求职遭拒绝，而转向 Sun ）。

此时 Sun 正忙于将 NeWS 与 X11 合并， Windows 开发组的一百多个技术人员同时要维持两个系统（ NeWS 和 X11 ）、两个操作系统、三套 Toolkit 、三个 Window 系统、三种硬件体系结构和两个图形用户界面，共 108 种不同的组合。两年多后 Patrick 已升至图形组项目负责人。这时他深感到 Sun 在 NeWS 上的失策，便提出了其建设性的设想，却无人响应，随起去意，拟转投 NeXT (由苹果公司的创始人 Steve Job 新组的公司)。就在他打点行装待发之时， Sun 准备着手解决他所提出的问题，并要提升他。但 Patrick 拒绝了这些条件，他要的是独立的开发环境，无约束的开发计划(通常开发各产品应考虑公司内部产品的互相支持)和精干的技术人员，来开发一个有划时代意义的产品。最后 Sun 的决策机构无条件地答应了他的要求，将他留下来组织这个尚无具体开发目的的项目(Green Project)，这时已是 1990 年末。

由 Patrick 、 Games 等几人组成的精干的小组开始了高强度的开发工作。经过几个月拼搏，终于完成了项目的概念设计，将目标定准在消费类电子产品，准备设计一种类似于 NeWS 的可下载的、跨平台的、解释型硬件需求低的环境。经过一年半的奋战，到 1992 年

初相关的设计都已完成，但这支队伍在体力上和精神上都已到了极限。到 1992 年夏末,由 Games 负责的 Oak(橡树)语言已可运行， Ed 和 Craig 负责的相应的硬件原型 Star7 和由 Patrick 负责的图形库和用户界面类库也告完成，并编写了一组应用: 一个电视程序管理器、一个共享消息板、一个无线寻呼机和一个脱放式 VCR 程序管理器。到此该项目纯技术部分告一段落。大家也都挺了过来。 Star7 是一个只有 15 × 10 × 10 cm 的小型 SPARC 工作站。它有 200Kbps 的无线接收机、红外 I/O 、 15cm 的 LCD 触摸屏、两个 PCMCIA 插槽、两个天堂 GameBoy 扬声器和 8 节 AA 电池。将 SunOS 、 Oak(Java)解释器、图形库、用户界面类、应用程序、图像、动画和声音全集成到 4M 快闪卡中,并能在 4M RAM 中正常运行。 Grean Progect 用原定时间的一半便成功了。

接下来 Sun 成立 FirstPerson 公司，旨在商品化该技术，但这便是走下坡路的开始。公司负责人选不当，开发主力人员纷纷离去， Patrick 和 Games 分别被封为 Chief Technologist(首席技术专家)和 Chief Scientist(首席科学家)，驾空起来。公司经营下滑不止，18 个月的艰苦奋斗付诸东流。虽后来经争取有望从 Time Warner(美国最大的有线电视公司)拿到一个大订单，但只因一步之差又输给了 SGI；而另一个从 3DO 来的大订单因 Oak 的使用权限争执而告吹。

就在 FirstPerson 难以为继、 Oak 面临流产危险之时， Sun 的 BillJoy 从各方予以关怀，使得 Oak 得以发育成熟。这时正赶上 WWW 在 Internet 上火了起来，遗憾的是开始由于工作紧张， Green 项目中竟无人见过 WWW 上的 Mosaic 。直到 1994 年末， Green 组员都已开始各找出路之时，才发现可将 Oak 用于 WWW 。经过一段时间的努力，于 1995 年初推出了 WebRunner （后改为 HotJava ）浏览器，为业界所认识。从此 Java 火起来了，而 Green 项目的骨干除 Games 外大都另起炉灶，“ run their own bussiness ”。

关于 Java 这段不寻常的历史，您做何感想呢?

编　者

1996年11月

目 录

第一章 Java 概论 1

1.1 什么是 Java 语言 1
1.2 设计 Java 语言的背景 1
1.3 Java 的现状和将来 2

第二章 Java 语言与 C/C++ 4

2.1 Java 语言的特点 4
2.1.1 简单性 4
2.1.2 面向对象 4
2.1.3 解释性 5
2.1.4 结构无关性 6
2.1.5 可移植性 6
2.1.6 动态性 7
2.1.7 健壮性 7
2.1.8 安全性 8
2.1.9 多线程 8
2.1.10 无用单元收集 8
2.1.11 高性能、易理解 9
2.2 Java 与 C/C++ 9
2.2.1 全局变量 9
2.2.2 Goto 无条件转移 9
2.2.3 指针 10
2.2.4 内存管理 10
2.2.5 数据类型和类型转换 11
2.2.6 头文件、结构和预处理 11

第三章 Java 语言的程序结构 12

3.1 Java 语言程序结构 12
3.2 编译单元实例 12

第四章 Java 语言的词法结构 16

4.1 注释 16
4.2 标识符 16
4.3 关键字 17
4.4 字面量(literal) 17

4.4.1 整型字面量 17
4.4.2 浮点型字面量 18
4.4.3 布尔型字面量 18
4.4.4 字符字面量 18
4.4.5 串字面量 19
4.4.6 空字面量 19
4.5 分隔符及运算符 19

第五章 Java 的数据类型、值和变量 20

5.1 基本类型和值 20
5.1.1 整数类型 20
5.1.2 浮点类型 22
5.1.3 布尔类型 24
5.1.4 字符类型 24
5.2 复合类型 25
5.2.1 对象 25
5.2.2 类对象 26
5.3 变量 27
5.3.1 变量的种类 27
5.3.2 变量的初值 27
5.4 类型转换 28
5.4.1 同一类型转换 28
5.4.2 简单类型拓宽转换 28
5.4.3 简单类型削窄转换 29
5.4.4 复合类型拓宽转换 30
5.4.5 非法转换 31
5.4.6 转换规则 31

第六章 Java 的类数据结构 34

6.1 类的说明 34
6.1.1 类修饰符 34
6.1.2 类名 36
6.1.3 超类（ superclass ）和子类（ subclass ） 37
6.1.4 超界面（ superinterface ） 38
6.1.5 类体 39
6.2 域说明 39
6.2.1 域修饰符和域说明 39
6.2.2 域的初始化 42
6.2.3 域的隐藏和继承 42

6.3 方法说明 44
6.3.1 方法修饰符 44
6.3.2 方法原型 47
6.3.3 方法体 48
6.3.4 方法的覆盖和隐藏 48
6.3.5 方法的继承（ inherit ） 48
6.3.6 方法的重载（ overload ） 49
6.3.7 throws 子句 49
6.4 静态初始化 50
6.5 构造函数说明 51
6.6 类应用例子 52

第七章 数组 58

7.1 数组的定义 58
7.2 数组的创建、访问和初始化 59
7.3 数组类对象 61
7.4 例子 61

第八章 界面 71

8.1 界面说明 71
8.2 常数（域）说明 72
8.3 抽象方法说明 73
8.3.1 抽象方法的定义 73
8.3.2 继承和覆盖 74
8.3.3 重载 74
8.4 界面的实现 75

第九章 包 78

9.1 Java 包说明 78
9.2 包的存储方式 78
9.3 Import（引入）说明 79
9.4 包的例子 80

第十章 运算符和表达式 86

10.1 运算符 86
10.1.1 赋值运算符 86
10.1.2 算术运算符 88
10.1.3 关系运算符 90
10.1.4 位运算符 90

10.1.5 布尔逻辑运算符 93
10.1.6 其他运算符 96
10.1.7 运算符的优先级 96
10.2 表达式 97
10.3 运算符和表达式应用例子 99

第十一章 控制结构 110

11.1 复合语句 110
11.2 空语句 111
11.3 标号语句 111
11.4 表达式语句 112
11.5 选择结构语句 112
11.5.1 if 语句 112
11.5.2 switch 语句 113
11.5.3 break 语句 114
11.5.4 return 语句 116
11.6 循环结构 117
11.6.1 while 语句 117
11.6.2 do 语句 118
11.6.3 for 语句 120
11.6.4 continue 语句 122
11.7 异常处理语句 124
11.7.1 throw 语句 124
11.7.2 try 语句 125
11.8 synchronized 语句 127
11.9 注释语句 127
11.10 例子 127

第十二章 异常处理机制 133

12.1 异常的起因 133
12.2 异常处理机制的基本结构和异常类型 133
12.3 异常的处理方法 134

第十三章 线程和同步 138

13.1 Java 的线程 138
13.2 线程的操作 138
13.3 同步 140

第十四章 Java 编程规范介绍 143

14.1 Java 开发工具 JDK（ Java Development Kit ） 143
14.1.1 Java 语言编译器— javac 143
14.1.2 Java 语言解释器— java 144
14.1.3 C 头文件和剩余（ STUB ）文件生成器— javah 147
14.1.4 Java 类文件反汇编工具— javap 148
14.1.5 Java 调试工具— jdb 148
14.1.6 Java API 文件生成器— javadoc 149
14.1.7 Java 剖析工具— javaprof 150
14.2 Java 语言编程规范介绍 150
14.3 Java 应用小程序（ Java applet ） 153
14.3.1 简单的 Java applet 154
14.3.2 applet 的执行方法 156

第十五章 Java API 简介 158

15.1 基本的 Java API 158
15.1.1 Java 语言包 java.lang 158
15.1.2 Java I/O 包（ java.io ） 160
15.1.3 Java 实用程序包（ java.util ） 161
15.1.4 Java 网络包（ java.net ） 162
15.1.5 抽象的（ Abstract ）窗口工具 (AWT) 包（ java.awt ） 162
15.1.6 AWT 映像包（ java.awt.image ） 164
15.1.7 AWT Peer 包（ java.awt.peer ） 164
15.1.8 Java Applet 包（ java.applet ） 165
15.2 Java API 功能介绍 165
15.2.1 Package java.lang 166
15.2.2 Package java.io 171
15.2.3 Package java.util 173
15.2.4 Package java.net 174

第一章　Java概论

1.1　什么是Java语言

Java语言是SUN公司于1995年推出的新一代面向对象的跨平台的高级程序设计语言，它的语法类似于C/C++，但没有C/C++复杂、混乱和不安全的语义。它具有简单、面向对象、分布式、安全、可靠、解释性、结构无关性、可移植、垃圾收集、高性能、多线程、动态、可扩展和易理解等特性，完全独立于机器。它将自己的源码翻译成中间字节代码，这些简短的指令经网络传递，在经过客户机的Java解释器时调用具体的方法或功能，即Java运行在Java解释器之上，Java解释器构成Java虚拟机。Java是一种可在网络上使用的语言，特别适合于网络应用软件的开发，尤其是Internet。

关于Java一词，人们有许多猜测。Java本意是“爪哇”，即印度尼西亚一个岛屿的名字，那里主要盛产咖啡。Java作为编程语言的名字，由于其本身所具有的特点，很多人认为它是首字母的缩略词，即"Just Another Vague Acronym"，但Java语言开发组否定了这种说法。在最初给Java语言命名时，Java语言开发组的领导人James Gosling称之为Oak，其灵感来自于在Sun Microsystem公司他的办公室窗外的一棵巨大的橡树。后来，Java语言开发组发现Oak已被注册为其他程序设计语言的名字，他们不得不另选新名。为了选一个合适的名字，他们可以说是绞尽脑汁，在偶然一次去咖啡店的途中，灵感出现，Java诞生。

1.2　设计Java语言的背景

近年来，由于C/C++本身所具有的特点，如:

- C/C++被定义为处于高级语言和汇编语言之间，允许程序员和计算机内部打交道到位、字节和控制CPU和外设的寄存器，为编程提供了高层次的灵活性.
- C/C++是可移植性语言，程序的编写是针对所给的操作系统和特定的计算机系统的，但只需做少量的修改就可移植到别的操作系统和计算机上。
- C/C++具有表达式的经济性和丰富的操作符集合.
- C代码高速高效，在一定程度上缓解了微机和小型机的硬件速度和存储限制.
- C++是面向对象的编程语言，界面的划分很容易。

C/C++已成为主导的软件开发工具，大多数程序设计人员在开发大型项目中都喜欢使用它。随着C/C++的广泛使用，人们也感到一些不足，如繁多的函数和一些难以掌握的功能，Java开发组在最初从事消费类电子产品设计、开发时就遇到了类似的问题。

Java语言开发组成立于1990年，其目的是开拓消费类电子产品市场，在具体设计时，他们很快发现已存在的高级语言像C和C++，对这类开发是不适合的，其具体表现在:

- 对于特定的计算机芯片，用C和C++编写的程序，必须编译。
- 当使用一个新的芯片时，大多数软件必须重新编译，以充分利用芯片的新特性。

❖ 要想适应软件库的变化，已编译的 C/C++程序必须重新编译。

而消费类电子产品要求可靠性高、费用低、标准化和使用简单，且大都有较长的生命期，但芯片经常更换，这就为使用传统的编程语言带来了极大的不便。为此，James Gosling 开始着手设计更适合于消费类电子产品的新型程序设计语言，以弥补 C/C++的不足。这就是 Java 语言，一种更快、更小、更可靠且可工作在所有 CPU 上的语言。

作为开发消费类电子产品的 Java 语言，在最初阶段并没引起人们的注意，直到 1994 年下半年，Java 开发部开发了一个 Internet 浏览器 HotJava，使 WWW 活泼起来，才使 Java 红火起来。随着 Internet 的迅猛发展，环球信息网 WWW 的快速增长，Java 已成为长时间以来最卓越的程序设计语言。

1.3 Java 的现状和将来

Java 的出现虽只是近一年多的事，但现在已成为全球计算机界的一大热门话题，是大多数媒体关注的中心。最初设计 Java 的目的是应用于消费类电子产品开发，如交互式电视、电话和烤面包箱等。随着 WWW 的诞生，Java 开发组成员用 Java 语言开发了一个 Internet 浏览器 HotJava，并正式出台。HotJava 使基于正文的、静态的 WWW 页面生动、有趣，并引起巨大轰动，它不仅使 Java 语言成熟，也向人们显示了 Java 的魅力。

现在，Java 代码仍处于 1.0 的开发阶段，要达到当前业界标准 C/C++语言的成熟水平，Java 还有大量的工作要做。尽管如此，由于 Java 所具有的优良特性和已取得的成绩，许多公司竞相和 Sun 签定 Java 使用许可协议，发布与 Java 相关的产品，如 MicroSoft、IBM、Netscape、Novell、DEC、Oracle 等。IBM 在和 Sun 公司签定合同仅半年多的时间内，就在为它的四种平台上嵌入 Java 支持取得成效，如宣布了它的 AIX 4.2 的 Java 接口等，并进行 Windows 3.1 的 Java 接口研究。在 Internet World 上，IBM 还展示了一些新的 Java 能力,这包括它的 Visual Age 应用开发产品的 Java 版本和 CISC 客户机程序的 Java 版本,并正在开发到 DB2 的 Java 数据库的连接（IDBC）。3Com 公司也计划推出采用 Java 管理网络硬件的软件产品 Transcend，以实现了网络设备的图形化监控、报告和配置。

同时，Sun 公司的 Java 开发部也在对 Java 进行一系列的升级和增强，并计划推出 Java 1.1 版本，以解决原存在的缺欠，提高 Java 性能。Java 开发组提出的增强功能包括提供一种轻便的“自动化视窗工具包”，这是 Java 开发工具包的关键编辑工具。此外，Java 开发组还在努力提高运行速度，扩充基本的 API，并计划推出下面一系列实用的 API：

- ❖ Java Enterprise API，以支持现有应用程序与企业数据库之间的连接，使得开发者能够用 Java 开发能在任意的操作平台上运行的分布式 Applet 和应用程序.
- ❖ Java Commerce API，以支持使用网络进行安全的电子贸易和金融管理。
- ❖ Java Management API，为在 Internet 上管理商业网络提供丰富的派生 Java 对象和方法。
- ❖ Java Server API，提供对服务器和管理系统资源进行统一的、稳定的访问.
- ❖ Java Media API，允许用户方便、灵活地利用网络上丰富的交互式媒体。
- ❖ Java Security API，增强开发的 Applet 应用程序安全机能。

Sun 的目的是使得软件开发者能够编写基于 Java 的协作应用程序、电子商务应用程序，

提高程序的安全性，以解决缺乏企业应用类库的问题。

1995 年 12 月 4 日， Netscape 通信公司与 Sun Microsystem 联合推出一种开放式、跨平台的对象描述语言 Java Script，以解决 Java 语言使用上的缺欠. 该语言面向非编程人员，容易掌握，用户可以用它在企业网络和 Internet 网上开发应用系统。该语言目前以得到 28 家主要信息公司的支持，如 Borland 、 DEC 、 Novell 等。

在 1996 年 5 月底的 Java One 大会上， Sun Microsystem 发布了 Java OS，为运行 Java 应用提供了最小、最快的系统环境，使得只需 512K ROM 和 128K RAM 就可运行 Java OS，还可把 Java OS 嵌入到控制器中. 同时，还宣布了支持 Java OS 的一些厂商，如 Oracle 公司计划在网络计算机中支持 Java OS 、宏基准备在用户设备中支持 Java OS 、东芝在 Internet 终端上及 Nokia 在电话方面支持 Java OS， 这就使得在一年内， Java OS 就可在多种硬件系统上运行。

为把开发者吸引到 Java， Sun 公司推出了完整的 Java 开发环境 Java Workshop1.0 及它的升级版本，为用户提供了用 Java 语言开发应用小程序所需的所有工具，尤其是在图形界面方面。 JavaSoft 还宣布了 Java Beans 计划，使用户可以在其他面向对象环境中（如 OpenDoc 、 Microsoft 的 Active X 和 Netscape 的 Live Connect ）使用 Java 组件，把所喜欢的各种对象模型统一起来，丰富使用的工具. 同时， Sun 和 Oracle 公司还设立了 Java 杯竞赛，奖金高达 100 万美元，加强 Java 的宣传力度。

目前， Sun 公司已完成 Java 的移植工作，使其可运行在 Macintosh 、 Windows 95 、 Sun 的 Solaris 和 Windows NT 等系统上，独立软件开发商也已将 Java 移植到 HPUX 、 OS/2 以及 NextStep 上。 Java 的诞生已对整个计算机产业发生了深远的影响，使用 Java 已成大势所趋。

第二章　Java 语言与 C/C++

2.1　Java 语言的特点

2.1.1　简单性

评价一种编程语言的简单性，既要考虑编程语言的易理解、易掌握程度，又要考虑它的具体功能实现．我们说 Java 简单，是把 Java 与目前最流行的高级编程语言 C/C++、Pascal 等相比较而言的．

为了使程序员很容易的掌握 Java 语言的编程技巧，Java 开发部在进行 Java 编程语言的开发时，是以 C/C++为基础的。Java 的语法类似于 C/C++、Pascal 等高级语言，这样就从程序结构上（例 2.1 给出了一个简单的 Java 程序）使得已熟练掌握 C/C++、Pascal 等高级语言编程规范的程序员，很容易了解 Java 语言的奥秘和编程技巧，掌握 Java 的开发规范。在保持 Java 与 C/C++等高级语言的许多相似之处之外，在具体设计 Java 语言时，Java 开发部摒弃了使 C/C++复杂、混乱和不安全的语义及很少用到的功能，如操作符重载、多重继承和隐含类型转换等一些自动强制性，避免了属性与操作名的冲突。同时，Java 增加了内存空间自动垃圾收集的功能，使得在 Java 具体应用过程中，程序员不必考虑如在 C/C++程序中因存储器管理而造成程序复杂的问题，极大地减少了编程错误。

另外，Java 简单化的另一方面是 Java 系统非常小，其基本解释程序和类支持功能约占 40K 字节，附加的基本标准库和线程支持功能占 175K 字节，这样就使 Java 程序能独立工作在非常小的系统上。

例 2.1　简单的 Java 程序

```
public class GoodJava
    {
        public static void main(String argv[])
            {
                System.out.print("Good Java");
            }
    }
```

2.1.2　面向对象

“面向对象”这一术语指的是把软件系统看成一系列离散的对象的集合，这些对象中既包括数据结构也包括行为。与传统的程序设计思想相比，传统的程序设计中的数据结构与行为之间只维持一种松散的联系。面向对象思想基本的特性有四种：标识唯一性、分类性、多态性和继承性。

❖　标识唯一性是指数据量化进入离散的、可区分的称之为对象的实体中。对象可以是

具体的，如一个文件；对象也可以是概念化的，如操作系统中的一种算法．每个对象都有自身的唯一标识．

- ❖ 分类性是指将具有一致的数据结构（属性）和行为（操作）的对象抽象成类，以反映与应用有关的重要性质而忽略其他一些无关的内容．任何类的划分都是主观的，但必须与具体应用相关．
- ❖ 多态性是指同一操作可以有多个不同的类行为．
- ❖ 继承性是指对具有层次关系的类的属性和操作进行共享的一种方式，它可以大大地减少设计和程序的重复性．

面向对象的开发过程通常由四个步骤组成:

- ❖ 标识系统中的对象以及对象的类．
- ❖ 提取每个对象的行为特征．
- ❖ 指定系统中对象之间的关系．
- ❖ 实现类．

采用面向对象的方法进行开发，开发重心放在分析阶段，这样就使得设计结果更为清晰和更为灵活，对以后的修改也容易实现．同时，采用面向对象的方法强调数据结构而不是功能．在整个开发过程中允许使用统一的概念：对象的概念；其他概念围绕对象来组成，使开发过程更为稳定，减少了不一致性和出错的可能性．

由于语言的构造与设计的构造是一致的，且面向对象的语言支持对象、运行多态性和继承，采用面向对象的语言来实现面向对象的设计是非常容易的．

Java 是面向对象的程序设计语言，除像数字和布尔量等简单类型外，Java 的很多方面都是用对象表示．Java 代码按类组织，每一个类定义构成一个对象行为的一系列方法，一个类可以继承另一个类的一些行为．虽然 Java 中的数字和布尔量等简单类型不是对象，但 Java 为所有的简单类型提供了封装（wrapper）对象，使得这些简单类型可以作为类来实现．

同时，Java 也支持作为抽象类的接口（interface），它使得程序员在为接口定义方法时，不必考虑方法的具体实现．一个类可以实现多个接口，一个对象也可实现多个接口．

2.1.3 解释性

程序设计语言是人类和计算机通信，并控制其工作的人工语言，虽目前编写程序有各种各样的程序设计语言，但计算机只懂机器语言写的程序．为使程序设计语言写的程序能被计算机所接受，必须将其翻译成机器语言，通常的翻译方法有两种，编译和解释．

- ❖ 编译是将源语言写的程序翻译成机器语言．
- ❖ 解释是将源语言写的程序翻译成一种介于源语言和机器语言之间的中间语言，然后再解释执行这种中间语言．

它们的区别在于解释程序不产生将被执行的目标程序，而是直接执行源程序．

解释系统的优点是:

- ❖ 直观易懂．
- ❖ 结构简单，易于实现．
- ❖ 易于实现人机会话，帮助用户发现和解释问题．
- ❖ 便于改进和扩充．

❖ 易于移植。

缺点是处理速度慢。

现在，纯粹的解释系统并不多见，通常的做法是把编译和解释做某种程度的结合，或先编译后解释，或先解释后编译，以弥补纯解释的缺欠。

Java 是一个解释性的运行系统， Java 解释器可以在任何机器上直接运行 Java 字节码（ byte code ）。当运行 Java 应用程序时， Java 应用程序首先被编译成非常接近机器指令的字节码，然后由 Java 解释器解释执行，其链接过程非常简单，开发过程也很快，并且在编译期间产生的各种信息可被视为字节码的一部分，以提供后期类型检查。同时，编译后产生的字节码不依赖任何特定的机器，它运行在 Java 解释器之上。这些字节码经网络传递，在经过客户机的 Java 解释器时在调用具体的方法或功能，实现具体的运作。 Java 解释器构成 Java 虚拟机。

2.1.4 结构无关性

Java 的设计目标是支持通用应用软件，尤其是 Internet 网络。在执行 Java 应用程序时，应用程序首先被编译成字节码，这些字节码独立于机器，且不能在任何计算机结构上直接运行，需要由 Java 解释器对其解释执行。

Java 的这种特性，使得 Java 应用软件不再依赖于任何计算机结构，编译好的 Java 程序可在任何支持 Java 的平台上运行，而不需重新编译。 Java 的结构无关性，不仅对网络十分有用，对软件的移植也非常有用，它解决了软件的移植和兼容问题. 使用 Java ，同一版本的应用软件就可运行于所有的平台。

随着 Internet 的发展，以网络为中心计算的普及，人们就愈来愈迫切需要一种独立于平台、代码可移植的计算技术。通过将 Java 解释器嵌入各个平台，使得 Java 应用程序不依赖于机器， Java 达到了软件开发平台的统一。

2.1.5 可移植性

自 60 年代起，如何解决程序的可移植性问题就一直成为计算机系统设计者和应用者非常关心的一个方面。至今，人们仍在这一方面做不懈的努力。

所谓程序的可移植性，就是指一个程序可以不加修改地由一台机器搬到另一台机器上，即同一个程序可以应用于不同的环境。如果程序具有可移植性或计算机系统设计成能实现程序的可移植性，可靠的软件就可长期应用，不必随着机器的更新而重新编写，从而能大大减少编制软件的工作量，尤其是可大大节省各个计算机系统为编制功能相近或甚至相同的软件所耗费的重复工作量。同时，新的硬件技术也能很快地被采用，使新的计算机系统能迅速发挥作用。

常用的移植技术是模拟和仿真，还有系列机的办法，如早年 DEC 公司的 PDP 和 VAX 系列，不过，系列机的办法所能实现的程序移植只局限在具有相同结构的各机器之间。目前，采用模拟和仿真技术虽可做到具有不同结构的机器间的相互移植，但效率一般都很低。

Java 是可移植性的编程语言，它的结构无关性可说明这一点。在 Java 语言中，只要基本数据类型确定，其运算特性也就确定了，不存在由于具体实现系统的不同而造成的差异，并且它的类型规定，符合所有主要 CPU 的特性。同时， Java 的各种库也规定了可移植的界面，保证了 Java 应用软件可移植到各种不同的系统。此外， Java 系统自身也是可移植的:

新的 Java 编译器用 Java 编写；运行库用 ANSI C 语言在可移植界面 POSIX 下编写。Java 应用程序，经编译后形成独立于机器的字节码，该字节码可由 Java 解释器解释执行。只要为各个计算机系统配置 Java 解释器，就可实现 Java 应用程序的平台无关性。

2.1.6 动态性

软件的动态性，体现了该软件适应新环境的能力，它直接影响该软件的生命期。我们在具体编程过程中，经常遇到这样的问题，同一应用程序在不同版本的相同语言环境下，会出现许多错误，其原因是由于支撑软件和应用软件更新不一致。

Java 采用后期建立模块间互连的方法，不仅很好地解决了这类问题，而且可以很方便地使用各种面向对象的功能，并使库能自由地增加新的方法和范例变量，而不影响它们的使用者。同时，Java 还引用 Objective C 中“界面”的概念，以弥补由于规定类不能多重继承而导致的不灵活性。一个界面就是一个对象所对应的一套方法规范，但不涉及任何具体的范例变量和实现方法。与类不同，界面有多级继承性，它比类的继承结构更灵活。

为了增强 Java 的动态性，Java 的类均被设计成具有运行标识码，使得同一类中的各个范例都具有相应的运行类定义。因此，Java 在运行期间很容易确认要用哪个类，并找到相应的类库。同时，Java 还提供用类名中所含字符串查找类定义的方法，可以根据字符串将数据类型名计算出来，并将其动态地链接到运行系统中。

2.1.7 健壮性

“健壮”一词来源于英文“Robust”，很多书也直接采用其直译“鲁棒”。我们评价一种语言或系统的健壮性，主要是考虑它对各种错误的处理程度，如果一种语言或系统在接收了错误的参数时，并不引起系统的崩溃，称该语言或系统具有良好的健壮性。为了保证软件具有很好的健壮性，在进行软件设计时应做到：

- ❖ 防止出错。软件应该具备排除用户输入错误的能力，不正确的用户输入参数不应该引起软件的崩溃，而是给出良好的对话提示。
- ❖ 避免先定义的限制条件。只要有可能，就使用动态内存分配来创建不具有先定义的限制条件的数据结构，各种限制应放到后期阶段去定义。
- ❖ 严格规定应用程序对内存的访问，以防止内存数据被错误的改写。

Java 语言为了确保用户编写的 Java 应用软件具有良好的健壮性，采取了很多有效的措施，如：

- ❖ 在 Java 语言中没有隐含说明，防止了 C++由于具有隐含说明而在编译期间留下许多漏洞的情况在 Java 中出现。同时，Java 链接程序对编译程序已做过的许多类型检查再次核查。做到了在编译期间对可能存在的问题进行早期检测，在后期进行动态检查。
- ❖ 在 Java 中提供了一个数组类型的指示器模型，以保证存储器不被错误地访问，数据不被错误地改写，同时，还可对指示器数组进行下标检查，不允许将任意整数值赋予指示器。

2.1.8 安全性

随着计算机网络技术，尤其是 Internet 的迅猛发展，计算机的脆弱性越来越突出，计算