

JAVA

程序设计教程

■ 李明 李厚安 著



- 全部使用最新JDK 1.1.3
- 列举大量完整程序例题
- 免费提供完全配套的投影胶片、习题、考题和解答

科学出版社

1022
1021

Java 程序设计教程

李 明 李厚安 著

科学出版社

1997

JS195/30
内 容 简 介

本书是以最新的JDK1.1.1为准的Java程序设计教科书。介绍了从如何编简单的Java程序,直到用Java语言写漂亮的动画applet和网页的全部内容。论述由浅入深,流畅幽默。书中配有插图及大量范例程序,每章都附有习题,并在书后提供了关键字索引和英汉对照。

本书可作为大学生或培训班28个学时的教材,也可供计算机技术人员自学时使用。

图书在版编目(CIP)数据

Java 程序设计教程/李明,李厚安著.-北京:科学出版社,1997
ISBN 7-03-006064-4

I. J… I. ①李… ②李… III. JAVA 语言-程序设计-教材
IV. TP312JA

中国版本图书馆CIP数据核字(97)第13628号

科学出版社出版

北京东黄城根北街16号

邮政编码:100717

北京双青印刷厂印刷

*

1997年8月第一版 开本:787×1092 1/16

1997年8月第一次印刷 印张:16 1/2

印数:1—5 000 字数:300 000

定价:21.00元

前 言

六王毕，四海一；蜀山兀，阿房出
杜牧，《阿房宫赋》，公元 852 年

如果在二年前，你认得 Java 这个名字，那你一定是个咖啡爱好者或来自印尼，那儿有个岛叫 Java（爪哇）盛产咖啡。

在过去的二年里，Java 程序设计语言在计算机软件领域里引起的革命，使它一跃成为在全球信息网(www)上开发软件的主要工具，并正在迅速取代 Pascal，C 和 C++ 在大学讲堂的地位。Java 本身也在日新月异，不断发展。1997 年 2 月 18 日，JavaSoft 发布了 JDK 1.1，3 月 11 日发布了 JDK 1.1.1_Final 版，5 月底发布了 JDK 1.1.2，最近又发布 JDK 1.1.3，全面地改进了 JDK 1.0。JDK 1.0 以及相应的所有教科书全部过时了。这本书是根据作者之一在香港城市大学电脑科学系 1997 年 1~5 月讲授 Java (CS3381) 的讲稿整理而成的，以 JDK 1.1 为准，包含 JDK 1.1 里新的 AWT，JavaBeans，JAR，JFC 等最新内容。

“一劳永逸”是 Java 的口号。说成英文“Write Once, Run Anywhere”挺别扭，但总还算恰如其分，它也是 Java 的成功秘诀。除此之外，Java 简单、漂亮、功能齐全、面向实体、安全、易学易用、交互并发、编译成字节码后解释执行。Java 从 C++ 而来，去粗取精，升华提高，脱胎换骨。如果你已经是熟练的 C++ 程序员，那么你可以很轻松地翻过本书前一部分。假设你是个初学者或根本不会编程序，那么你应该庆幸你不像作者们这样走了许多弯路，学了一大堆正在被淘汰的语言，而是一来就碰上了 Java。学语言也像学打球、练武功一样，根基要正。C 和 FORTRAN 会给你带来坏习惯，而 Pascal 一类语言学了又没用。Java 正在取代 C 和 C++ 而成为软件工业的主要语言。Java 是有史以来唯一被工业界和学术界同时认可的语言。Java 应该是初学者的第一语言。在我们中国，国际互联网(Internet)正在迅速普及。我们已经看到 21 世纪信息社会的曙光。信息高速公路很快就会像电话一样进入每个家庭，随之而来的就是 Java。

这本书不是翻译的书，你不用担心文字晦涩。这本书也不是一本 Java 手册，我们无意包罗万象。这是一本专为你写的教科书。它的内容恰好适合大学一个学期的课程。为了方便大学老师，我们还提供了与本书完全配套的一学期的讲课稿的英文投影胶片、课外作业及解答、期末考试及解答，你可以从 <http://www.cs.cityu.edu.hk/~mli/java.html> 免费下载。它也适合自学和作为参考书。我们在书后附录中提供了索引、中英文重要单词标准翻译对照、JDK 1.0 和 JDK 1.1 的 AWT 更名方法对照表，以及主要的 Java 网址和书籍等。我们只假设你有最基本的计算机知识。我们会由浅入深，用例子逐步引导你学会 Java 程序设计，而避免给你讲大道理、玩大名词儿。在每章结尾，我们还留了习题，它们是学习编程的关键。你如果不编程序，你就永远学不会 Java（或者是别的什么语言）。子曰：“知之者不如好之者，好之者不如乐之者。”，我们希望

你能很快发现编 Java 程序是一件很好玩儿的事情。

我们感谢香港城市大学电脑科学系提供写作及教授本书的条件和中国文化的环境。感谢吕虹、堵丁柱、贾小华、管惠维、贾维佳、曹建农、李京、钮建伟、李德明、孙成政、陈恭、束为、Paul Vitanyi, N.V. Balasubramanian, Lenore Blum, Manuel Blum, 张大鹏、黄琛芳及钟宝旋的支持与帮助。尤其感谢赵伟、刘燕虹和 John Tromp 提出了许多宝贵的意见及建议。没有这些朋友们的帮助，就没有本书。

作 者

1997 年 6 月 10 日

目 录

第一章 跟你聊聊计算机和全球信息网	1
1.1 计算机和全球信息网	1
1.2 计算机程序设计语言与 Java	2
1.3 Java 的结构与特点	4
第二章 你的第一个 Java 程序	7
2.1 JDK 的获得与安装	7
2.2 第一个简单的 Java 应用程序	8
2.3 第一个小应用	10
2.4 小应用与应用程序	11
习 题	12
第三章 变量 原始类型 表达式 语句	13
3.1 标识符和关键字	13
3.2 原始类型	14
3.2.1 整数类型	15
3.2.2 浮点数类型	15
3.2.3 布尔类型	15
3.2.4 字符类型	16
3.3 变 量	16
3.4 文字常数	17
3.4.1 整数常数	18
3.4.2 浮点数常数	18
3.4.3 布尔常数	18
3.4.4 字符常数	18
3.4.5 其它文字常数	19
3.5 表达式, 语句和块	19
3.6 运算符	20
3.6.1 赋值运算符	21
3.6.2 算术运算符	21
3.6.3 自增和自减运算符	23
3.6.4 关系运算符	24
3.6.5 位运算符和逻辑运算符	24
3.6.6 广义赋值运算符	25
3.7 运算符的优先级	26
3.8 类型转换	27
3.8.1 隐含转换	27
3.8.2 强制转换	27

3.9	Unicode	28
习 题		28
第四章	类	30
4.1	面向实体的程序设计	30
4.2	类的定义	32
4.3	实 体	32
4.3.1	生成实体	32
4.3.2	怎样使用实体	33
4.4	方 法	34
4.4.1	方法的定义与调用	34
4.4.2	按值调用	36
4.4.3	this 引用	37
4.4.4	方法重载	37
4.5	置初值与构造方法	38
4.6	数 组	39
4.7	static 变量与方法	41
4.8	包 装 类	43
4.9	内 嵌 类	45
习 题		45
第五章	子类与继承	47
5.1	子 类	47
5.2	继 承	48
5.3	数据隐藏与方法重写	50
5.4	super 关键字	50
5.5	多 态	51
5.6	子类的构造方法	52
5.7	final 修饰符	53
5.8	抽象方法与抽象类	54
5.9	使用权限: public, private, protected	55
5.10	实体的类型转换	56
习 题		56
第六章	控制语句	58
6.1	if-else 语句	58
6.2	switch 语句	60
6.3	循环语句	63
6.3.1	for 循环语句	63
6.3.2	while 循环语句	65
6.3.3	do...while 循环语句	66
6.4	标 号	67

6.5	break 语句	67
6.6	continue 语句	69
6.7	return 语句	70
6.8	递 归	70
	习 题	71
第七章	接 口	72
7.1	接口的声明	72
7.2	实现接口	73
7.3	扩展接口和接口的多重继承	75
7.4	关于菱形继承	76
7.5	使用别的类实现的方法	77
	习 题	79
第八章	包	80
8.1	Java 提供的包	80
8.2	定义自己的包	81
8.3	包的引用	84
8.4	包的结构	85
8.5	存取保护	85
	习 题	86
第九章	例外处理	87
9.1	Java 中的 Throwable 类及其子类	88
9.2	try-catch-finally 语句	89
9.3	throw 语句	92
9.4	设计自己的例外类	94
9.5	三大类例外	95
	习 题	96
第十章	线 程	97
10.1	线程的概念	97
10.2	线程的声明	98
10.3	线程的状态	99
10.4	线程状态的转换方法	100
10.5	扩展 Thread 类	102
10.6	使用 Runnable 接口	104
10.7	线程例子	105
10.8	Synchronization: 协调线程之间的冲突	106
10.8.1	synchronized 方法	107
10.8.2	synchronized 块	109
10.8.3	怎样使用 wait() 和 notify()	109
10.8.4	死 锁	111

习 题	112
第十一章 高级数据类	114
11.1 多维数组	114
11.2 字符串: String 类	115
11.2.1 字符串的长度	116
11.2.2 查找字符	116
11.2.3 字符串的比较	117
11.2.4 子 串	118
11.2.5 修改字符串	118
11.2.6 toString()方法	118
11.3 字符串: StringBuffer 类	119
11.4 向 量	120
11.4.1 向量的声明	121
11.4.2 Vector 类的方法	121
11.5 堆 栈	123
11.6 Dictionary 抽象类和它的散列表实现	125
11.6.1 Dictionary 抽象类	125
11.6.2 散列表	125
11.7 数据结构拾零	127
习 题	130
第十二章 杂 类	132
12.1 Math 类	132
12.2 Random 类	133
12.3 Number 抽象类	133
12.4 Date 类和 Calendar 类	134
12.5 System 类	135
12.6 输入输出	137
12.6.1 InputStream 和 OutputStream 抽象类	138
12.6.2 Reader 和 Writer 抽象类	139
12.6.3 FileInputStream 类和 FileOutputStream 类	139
12.6.4 FileReader 类和 FileWriter 类	140
12.6.5 ByteArrayInputStream 类和 ByteArrayOutputStream 类	141
12.6.6 CharArrayReader 类和 CharArrayWriter 类	142
12.6.7 PrintStream 类和 PrintWriter 类	142
12.6.8 File 类	143
习 题	145
第十三章 Applet 基础	146
13.1 HTML 里的 APPLET 标志	147
13.2 Applet 类和它的方法的调用次序	149

13.3	简单图形	151
13.4	字型与颜色	158
13.4.1	字 型	158
13.4.2	颜 色	160
13.5	图片的处理	163
13.5.1	载入图片	163
13.5.2	画出图片	164
13.6	HTML 简介	166
	习 题	169
第十四章	用户界面	170
14.1	设计用户界面的组件	170
14.2	屏幕布局	179
14.3	菜 单	185
14.3.1	下拉式菜单	185
14.3.2	弹出式菜单	187
14.4	响应事件	188
14.4.1	事件代表模型	189
14.4.2	两个例子	189
14.4.3	几个组件的事件处理	193
14.5	JDK 1.0 事件处理模型	194
	习 题	196
第十五章	音乐和动画	198
15.1	声音和音乐	198
15.2	动 画	202
15.3	程序例子	211
	习 题	218
第十六章	咖啡豆儿及其它	219
16.1	咖啡豆(JavaBeans)	219
16.1.1	什么是咖啡豆	220
16.1.2	咖啡豆,穿成串	220
16.1.3	怎么写 Java 的 Beans	224
16.1.4	咖啡豆和你	224
16.2	咖 啡 罐(JAR)	225
16.3	怎样使用 Native 程序	226
16.4	远程方法调用	228
16.5	JDBC API	232
16.6	Java 基础类库(JFC)和未来的 JDK 1.2	233
附录 A	怎样使用 ftp 方法	234
附录 B	怎样将 JDK 1.0 的程序转换成 JDK 1.1 的程序	235

B.1 怎样将 JDK 1.0 的程序转换成 JDK 1.1 的程序	235
B.2 JDK 1.0 和 JDK 1.1 的 AWT 更名方法对照表	237
B.3 事件转换表	238
附录 C 本书使用的重要单词标准英汉对照	239
参考文献	241
索引	243

第一章 跟你聊聊计算机和全球信息网

The Network is the Computer
SUN Microsystems, 1982

1.1 计算机和全球信息网

如果有人将来写计算机历史，那他或许会这样写：计算机在 20 世纪出现，在 21 世纪消失在网络中。

从 1936 年德国人 Konrad Zuse 的第一台计算机(Z-1)到现在广泛普及的微机，计算机经历了四代产品。1946 年，“巨型机”ENIAC 每秒钟只能做 500 个乘法，今天一台普通的微机每秒能做上百万个乘法。在信息爆炸的今天，计算机的应用几乎遍布了人类的各个领域：生活、办公、医药、国防等等，无所不在。90 年代，全球信息网使计算机真正进入了我们每个人的日常生活。在国际互联网(Internet)上，你可以寄信，谈话，下棋，打牌，打电话^①，看电影，买东西，订机票，找对象，查信息，玩游戏，做生意，读小说，看报纸，发表文章。在 5~10 年内，20% 的商业将会转移到网上。电话、传真机、电视机、计算机将慢慢消失。网络会替你做一切事儿，你需要的只是一台网络终端(Network Computer)。

全球信息网的雏形最初是瑞士高能物理中心的 Tim Berners-Lee 在 1989 年提出的，我们将用到的 HTML 语言也是他同时提出的。1990 年 10 月，他的提议获得了批准，他将其命名为“World-Wide Web (全球信息网)”。这个网络基于 Internet。用最简单的话来说，全球信息网包括两个简单的部分：浏览器和服务器。所谓浏览器和服务器无非是两个不同的程序，其中服务器具有较高的性能。浏览器向服务器提出请求，服务器为浏览器提供相关信息。设计者没有想到让全球信息网遍及全球，但是事所难料，统计资料表明，全球信息网网点(服务器)个数每 3~6 个月加倍一次。表 1-1 列出了从 1993 年 6 月至 1997 年 1 月的 WWW 网点统计数。

表 1-1

日期	网点数
6/1993	130
12/1993	623
6/1994	2 738
12/1994	10 022
6/1995	23 500
1/1996	100 000
6/1996	230 000
1/1997	476 000

① 即便是今天，只要 100 元人民币就可以打六个小时的国际长途，如果你和对方都有合适的声音软件。

1997年1月, Alta Vista 登记了 476 000 个服务器全球信息网网点, 以及 3 100 百万个用户网页。诸如 Alta Vista, Yahoo, Opentext, Lycos 这些信息检索服务站每天都有几千万人次用它们查找信息。至于每天在国际互联网上发送 Email, 用 ftp 传送文件的人次数就难以统计了。全球信息网以迅雷不及掩耳之势向全球的每个角落扩张。Java 应运而生。“Write Once, Run Anywhere (一劳永逸)”的口号使得 Java 一出现就光芒四射, 用 Java 开发出来的软件可以不用修改或重新编译而直接用于任何计算机上, 不管它是 IBM PC, 苹果机, 或是 UNIX 工作站, 也不管它用什么操作系统或哪个版本, Java 程序都能安全运行。

在 Java 之前, 全球信息网上的网页基本上是静态的, 一个网点不能执行另一个网点上的程序, 因为这两个网点可能用的是不同的计算机、不同的操作系统, 或者只是不同版本的软件。即便这些全一样, 也还有个关键的安全性问题。Java 的出现迅速改变了这种状况。Java 的小应用(applet)可以被下载到任何用户机器上安全运行, 这使得用户与网页的互动对话、动画等等成为现实。用 Java 做成的动画优美流畅, 倾倒了无数用户。Java 为全球信息网带来了生命和一场真正意义上的技术革命。

人们预言 Java 还将带来更深刻的革命。它将改变我们使用软件的方式。你不用再去买专用软件, 然后没完没了地更新。你只要在需要时去访问专门网页, 交使用费, 就能使用最新版本的软件。这样的网络将面向用户, 给你的计算机“减肥”(减少 fatware); 而不是像现在的微机一样面向一个大公司, 经常要你去买新版本软件。

1.2 计算机程序设计语言与 Java

Java 是一个面向实体的通用程序设计语言。Java 不但适用于网上程序设计, 它也适用于一般的大规模软件工程项目。

计算机程序设计语言的发展与研究已经经历了半个世纪。我们知道, 计算机懂得的只是 0-1 序列。比如, 计算机看见:

```
11 0001 0010
```

如果 11 代表加法, 它就将 0010 单元里的数据加到 0001 单元里去。但是人们很快就发现, 这种机器码实在难写又难读。不如把它写成:

```
ADD A, B
```

然后再用一个翻译程序把 ADD 变成 11, A 变成 0001, B 变成 0010。这就是汇编语言。它的每一条命令恰好对应一条机器码。用汇编语言写程序虽说仍然很难, 但是它效率高。在 50 年代, 当计算机太慢且内存太小的时候, 汇编语言起到了它的作用。同时, 计算机科学家们开始追求计算机语言的表达力、与机器无关性、结构化、模块化, 以及更接近我们日常用的语言、思维和数学公式, 于是产生了高级程序设计语言。用高级语言 C (或者 Java), 前边那条命令就可以写成:

```
A = A + B;
```

或者

```
A += B;
```

世界上第一个高级语言并不是 FORTRAN 而是 K. Zuse 在 1944 年设计的 Plankalkul。这个语言有变量、过程与参数，它甚至强调我们 70 年代才开始重视的数据与程序结构。

从机器码，到汇编语言，到过程语言(如 FORTRAN, Algol 60, PL/1, COBOL, BASIC, Pascal, C)，到函数语言(如 Lisp, ML)，到逻辑语言(如 Prolog)，到面向实体的语言(如 C++, Eiffel, Smalltalk-80, Ada)，迄今为止，世界上已有 2000~4000 种程序设计语言。这些语言多数都只有它们的设计者和他们不情愿的学生们会用。但是最差劲儿的反而是最流行的。世界上最好的程序设计员 E.W. Dijkstra 在 1975 年发表过如下精辟见解(EWD498)：FORTRAN 得的是婴儿错乱症；…… PL/1 得了绝症；…… 用过 BASIC 的程序员再也不会编出好的程序；…… 使用 COBOL 会伤害你的大脑；…… APL 根本就是个错误。Dijkstra 没有提到 C，那是因为 C 在 1978 年才问世。长久以来，在大学讲堂里，我们经常面临着一个矛盾：是用 C 还是用 Pascal。Pascal 设计得好，但是太简单，工业界没用；C 最实用，但是用 C 作出的程序实在不怎么样。一个人用的工具影响他的习惯。不好的语言带来不好的编程习惯。50 年来的程序设计语言的研究带来了各种新思想以及正反面的经验。Java 的设计吸收了几乎所有的这些经验及新思想。

1990 年，SUN 公司上马了一个叫 Green 的项目。这个项目需要一种与平台无关、可靠性强、小而灵活的编程语言来控制有线电视和小型家用电器。这个以 James Gosling (UNIX emacs 的作者之一)为领导的项目几次险些被 SUN 公司取消掉。Java 组原始成员之一 P. Naughton 这样描写 1988 年第一次在 Gosling 的办公室见到他的情景：“他穿着一件不合身的 T 恤衫…… 没穿鞋，袜子也不凑对儿……” [N]。直到 1994 年，当全球信息网的热潮席卷全球时，在风雨飘摇中的 Green 项目时来运转，他们发现 Java 这种中性平台及可靠性强的语言恰恰是全球信息网在等待的编程语言。Java 组的开发人员基于网络对 Java 进行了一系列的改进，融合了 C 和 C++ 等传统程序语言的优点，形成了现在的这套与众不同的面向实体的通用程序设计语言。

Java 的原名叫 Oak (橡树)。但申请注册商标时，律师发现 Oak 已经有人用了。所以他们必须重新命名。他们想了许多名字，诸如：DNA, SILK, RUBY, WRL, …… 其中也有 Java，提议者是在喝一杯 Java 咖啡时无意提到的。其它的名字后来都被淘汰掉了。

Java (JDK 1.0)正式发表于 1995 年 5 月。《时代》杂志把 Java 选为“1995 年最好的十大产品之一”。Java 是人选的唯一电子类产品。至 1996 年底，每个大计算机公司都购买了 Java 语言的使用权，宣布支持 Java；每个微机的操作系统都(将)在新版本中提供 Java 环境；每个主要的制造发展应用程序工具的公司都将出 Java 产品；每个计算机科学家都在谈论着 Java。到 1996 年 10 月，互联网上已经有超过八万个节点具有 Java 的内容。现在，已有 200 多种英文的 Java 书籍问世，还有许多种准备出版。世界上已有 30 万 Java 程序员，而目前 Windows 程序员才 40 万。预计全世界 Java 程序员的需要量是 110 万(香港，南华早报，1997 年 3 月 4 日)。在计算机的历史上，没有任何一个语言如此快地被学术界和工业界同时认可。

1997 年 2 月 18 日，JavaSoft 公布了 JDK 1.1，3 月 11 日又公布了 JDK 1.1.1。在

一个月的时间里，有近 22 万人从 SUN 站点上下载了 JDK 1.1。JDK 1.0.x($x < 3$)的许多功能被改进、增强、淘汰。新的 JDK 1.1 仍然暂时允许旧的 JDK 1.0 程序运行，但是，JDK 1.0 的许多东西将最终被淘汰掉。所以，读者应该尽量避免使用 JDK 1.0.x 来学习 Java，它已经过时了。JDK 1.1.1 去掉了 JDK 1.1 中的一些错误。JDK 1.1.2 以及 JDK 1.1.3 再次修改了一些错误。不过 JDK 1.1.x ($x = 1, 2, 3$)相对于 JDK 1.1 来说，并没有增加新的特性，其标准基本上没有改变。有了 JavaBeans 规范以及新的 AWT，JDK 1.1 将相对稳定。本书以 JDK 1.1 为准，使用 JDK 1.1.3 编译程序。当与 JDK 1.0 不一样时，我们也将告诉你 JDK 1.0 的程序设计方法(比如事件处理)，还会告诉你怎样把已有的 JDK 1.0 程序改写成 JDK 1.1 程序。

1.3 Java 的结构与特点

Java 是解释执行的。如图 1-1 所示，用户写好 Java 源程序后，通过 Java 编译器(javac)把源程序编译成字节码。字节码可以在任何装有 Java 虚拟机的平台上运行。Java 虚拟机的运行时系统对字节码进行解释执行。

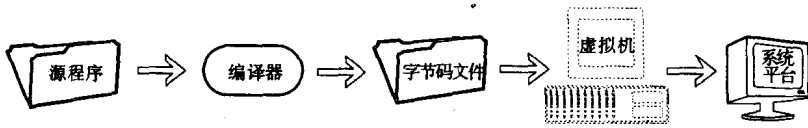


图 1-1

为了提高效率，SUN 公司最近推出了 Java 芯片(picoJava-I 微处理器)。新的 Java 芯片将比现在的 JIT 编译器快 5 倍。微软公司也推出了 Microsoft Visual J++。Borland 公司的 Open JBuilder，是 Delphi 的 Java 翻版，是写软件非常有效的工具。Java 有如下特点：

第一、一劳永逸(Write Once, Run Anywhere)

一般语言编写的应用程序如果换了一种平台系统，那么就必须要将源代码在新的平台上重新编译，以形成适合于该种 CPU 指令的代码。然后，如果你的运气好的话，它没准儿能运行。而 Java 编译器生成与机器无关的字节码指令，只要安装了 Java 虚拟机，你的应用程序不用重新编译就可以在所有平台上执行。这是其它程序设计语言所做不到的。这一点对网络应用至关重要。对其它软件应用系统，这种中立性也是不可忽视的关键问题。

第二、简单性

Java 来源于 C++，但却删除了 C++ 中很少被使用、不安全或容易令人混淆的功能：多继承，指针操作，GOTO 语句，全局变量及头文件，宏语句(macros)，struct 和 union 模块，模板(template)，自动类型转换，以及与 C 的(向后)兼容性。用户也不用释放内存空间，因为 Java 自动做垃圾收集。因此 Java 相对简单得多。不过这并没有使得 Java 的功能有所减弱，减掉的功能一般都可以用别的手段实现。

第三、面向实体

在国内，一般把 Object-Oriented 译成“面向对象”，有点儿像谈恋爱，而且意思也

不准。当我们说“生成一个 object”时，我们的意思是“生成一个实体”，而不是“生成一个对象”。有对象一定有主语：谁是对象的对象呢？所以我们用“面向实体”，更为客观。

Java 是一个面向实体的高级语言，但它不像 Smalltalk-80 [J] 那样纯。在 Java 里，有几个定义整数、浮点数、布尔值、字符的原始类型，它们不是类，这部分是 Java 的非面向实体的部分。除此之外，一切都是类。这样的处理使 Java 不像 Smalltalk-80 用起来那么别扭，连一个整数也要当成一个实体分配空间。

第四、健壮性和安全性

一个好的程序应该是一个不管在什么输入的情况下都不出错的程序。与 C++ 不同，Java 是一个强类型的语言。Java 在对程序进行编译程序时，就对类型和其它可能出现的问题进行检查。在运行时再进行动态的检查。所有的输入输出语句都要有例外处理程序。Java 的内存自动垃圾收集程序有效地防止程序员在管理内存时容易产生的错误，以及因此出现的内存丢失的情况。

Java 不支持指针类。指针操作是 C 语言中许多错误的根源。Java 剔除了指针类，这样可以有效地防止对内存进行非法访问等许多错误的产生。注意，Java 并非真的没有指针。对每一个生成的实体，Java 都要有个引用(reference)——也就是指针——指着它。Java 只是不准用户做指针操作罢了。

第五、其它

Java 支持多线程：多线程是指使多个应用程序能够并发执行。通过实行多线程，程序员可以容易地控制与协调并发程序的执行。多线程也使网上的交互行为更容易实现。

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类和方法以适应新的环境，而不会影响用户程序的执行。

分布性：Java 提供的类库可以容易地处理 TCP/IP 协议，用户可以通过 URL 开启和使用远程实体，比如其它小应用。新的 JDK 1.1 又加入了远程方法调用(Remote Method Invocation)。

可移植性：这个概念对 Java 来说已过时了。Java 根本不用像其它语言一样被“移植”。它与平台无关，连重新编译都不用，字节码可以直接被执行。

第六、对 Java 的批评

从程序设计理论上来说，Java 语言的所有概念全是 80 年代已经成熟的理论。最初的 Java 版本有许多的缺陷，特别是 PC 和 MAC 版的瑕疵要比 UNIX 版的多。还有一些安全漏洞，速度上也太慢。不过这些缺陷在新的版本中已经大部分被修改了。SUN 公司的 Java 芯片将使 Java 的速度可以与 C++ 不相上下，使得用 Java 做大型软件代替 C 或 C++ 成为可行。

因为安全性问题，Java 小应用下载到用户机器上之后，不能存取本地文件。这个问题如不解决，它将会影响编制许多应用程序。

Java 的安全性是在编译语言层次实现的。Lucent 公司(原 AT&T 的一部分，C 语言的家乡)最近推出了 Inferno 操作系统及与其配套的 Limbo 语言。Inferno 是一个网络操作系统，它在系统层次实现安全措施，比 Java 彻底。当然，即便系统保证 100% 的安全性，小应用的设计者仍可以写一个死循环的程序，使得下载这个小应用的用户的机器停

止工作。而且，可计算性理论告诉我们，不存在检查另一个程序是不是死循环的程序。Limbo 不是一个面向实体的语言。Inferno 也将支持 Java。

另外，由于 Java 小应用是下载到用户机器上的，用户很容易用 Java 反编译程序 (javap) 把 Java 的字节码变回源程序。这样，软件保护就成了问题。