全国高等院校应用型**创新规划**教材 计算机系列

Java 核心技术

白文荣 主 编
王晓燕 副主编

- 以基础知识—实用技术—项目实训为主线
- 包含必要的理论基础知识和新知识，满足课程培养计划的要求
- 书中展示大量实际案例和习题，旨在提高本书的实践性
- 配备免费教学资源——电子课件、习题答案及书中源代码

清华大学出版社

全国高等院校应用型创新规划教材·计算机系列

Java 核心技术

白文荣　主　编

王晓燕　副主编

清华大学出版社
北京

内 容 简 介

本书是作者在多年从事 Java 程序设计、Java 核心技术课程教学实践基础上编写的。全书共分为 14 章，通过大量的可运行实例，系统地讲授了 Java 语言基本原理、Java 语言基本语法、Java 面向对象编程机制、异常处理及线程、Java I/O 流技术、GUI 界面设计、事件及事件处理、Java 常用类及集合、JSP 基本语法、JSP 内置对象、JavaBean 技术、JDBC 编程技术、Servlet 技术等相关知识。

本书紧密结合实际需求，从案例教学、项目式教学思路出发，根据需要安排了 Java 基础案例和综合案例，逐步阐述了 Java 各核心技术之间的联系。书后配有适量的思考题和练习题，使读者能够在学习过程中提高操作能力和实际应用能力。

本书可作为高等院校学生学习 Java 核心技术、Java 程序设计、面向对象编程与设计、软件项目实战等课程的教材，也可以作为读者自学的参考书。

图书在版编目(CIP)数据

Java 核心技术/白文荣主编. —北京：清华大学出版社，2018
(全国高等院校应用型创新规划教材 • 计算机系列)
ISBN 978-7-302-48380-9

Ⅰ. ①J…　Ⅱ. ①白…　Ⅲ. ①JAVA 语言—程序设计—教材　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2017)第 216514 号

责任编辑：秦　甲
封面设计：杨玉兰
责任校对：宋延清
责任印制：沈　露
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn, 010-62791865
印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　印　张：22　　字　数：535 千字
版　　次：2018 年 1 月第 1 版　　印　次：2018 年 1 月第 1 次印刷
印　　数：1～2000
定　　价：49.80 元

产品编号：071847-01

前　言

Java 是面向对象的、支持多线程的网络编程语言。它是目前最流行的编程语言之一，具有高度的安全性、可移植性和代码可重用性。本书立足于 Java 的各种核心技术，系统地讲解 Java 语言知识。学习本书无需任何基础，零起点学 Java 是本书的宗旨。本书实用性强，包含 Java 基础知识和高级编程方面的所有常用核心技术，由浅入深、通俗易懂、难易适度，可以增强读者成为 IT 精英的信心。

全书共分 14 章，各章的主要内容安排如下。

第 1 章 Java 语言的基本概念、编程技术的发展史和 Java 运行环境的搭建等。

第 2 章 Java 语言的基本语法，包括变量的定义、常量的定义、基本数据类型、复合数据类型、数组、编程语言控制流程等知识。

第 3 章 Java 的面向对象编程机制，如类与对象、接口及抽象类等。

第 4 章 面向对象程序设计语言 Java 的异常处理机制及线程的创建和使用方式。

第 5 章 Java 的输入输出处理机制，即 Java 的 IO 流机制。

第 6 章 Java 的图形用户界面 GUI 的设计，如 AWT 和 Swing 相关控件的应用。

第 7 章 Java 语言中的事件监听和事件处理机制。

第 8 章 Java 的常用类及集合的定义、创建和使用方式等基础知识。

第 9 章 JSP 的基本语法和 JSP 运行环境的搭建等。

第 10 章 JSP 常用的内置对象及应用方法。

第 11 章 Java 核心技术之一的 Java Bean 技术及其编程实践方式。

第 12 章 Java 连接数据库的知识——JDBC。

第 13 章 Java 核心技术之一的 Servlet 技术及其编程实践。

第 14 章 Java 核心技术的应用案例及程序分析思想。

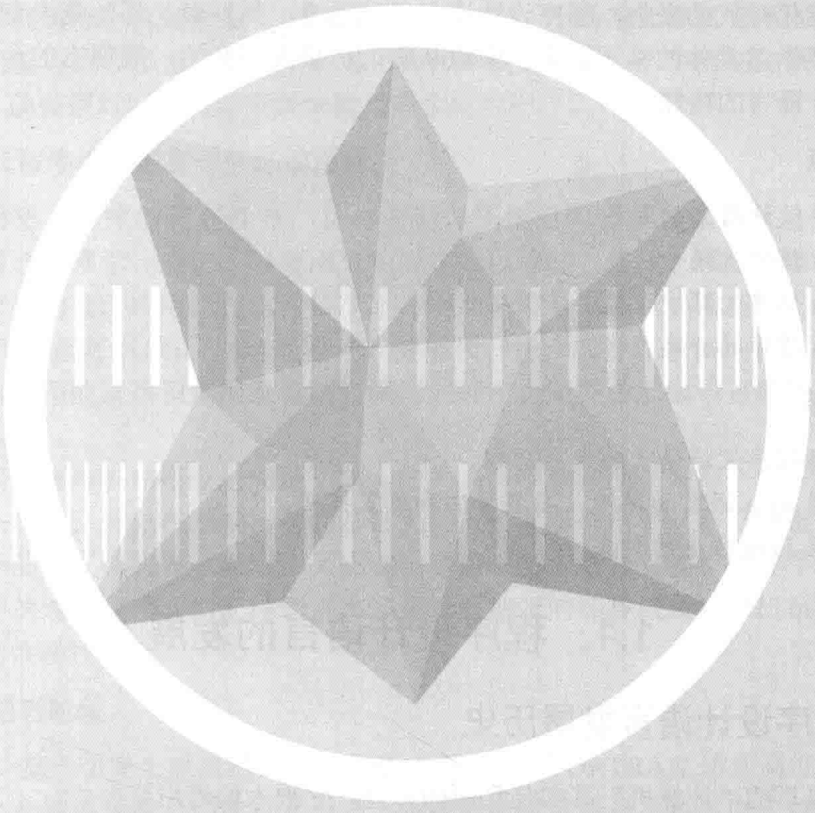
本书由白文荣主编，王晓燕为副主编。第 1、2、3、4、5、6、7、8、13、14 章由白文荣编写，第 9、10、11、12 章由王晓燕编写，在本书策划和编写的过程中，得到了清华大学出版社的支持，在此表示衷心的感谢。

由于作者水平有限，书中难免存在错误和不足之处，敬请广大读者批评指正。

为了方便教师教学和学生自主学习，本书配有电子教案、案例源代码、安装软件等，若有需要，可从清华大学出版社网站下载。

编　者

目　录



第 1 章

Java 语言简介

随着信息化时代的来临，程序设计语言发展迅速，到目前，还丝毫没有规范到统一语言的迹象，要学习具体的编程语言，应该从它的历史发展开始，展现它的全貌，从发展中了解程序设计语言的精髓。

本章要点

- 程序设计语言的发展。
- Java 语言的简介。
- Java 语言的环境配置。
- Java 程序的分类。

学习目标

- 了解程序设计语言的发展历程。
- 掌握 Java 语言的环境、安装方法和配置技巧。
- 掌握 Java 程序的种类。

1.1 程序设计语言的发展

1.1.1 程序设计语言发展历史

世界上最早的“计算机”其实是我国的算盘，它被人们沿用至今。

在 17 世纪，帕斯卡(Pascal)等人发明了一种以传动齿轮为基础的机械“计算机”，它以齿轮的转动来控制计算的累加与进位。19 世纪初，英国剑桥大学著名数学家查尔斯·巴贝奇(Charles Babbage)于 1822 年和 1848 年分别设计出了两种差分机，并于 1833 年制造出了有名的分析机。分析机在原理上与当今社会的计算机非常类似，它靠阅读穿孔卡片来对输入的数据进行算术运算，并给出结果。而且分析机可以随意重复运算序列。这些由阿达·洛芙莱斯(Ada Lovelace)设计的运算序列，可以解决许多计算方面的问题。实际上，这种运算序列就是程序的雏形，而这种设计思想一直沿用至今，因而 Ada Lovelace 被称为世界上第一个程序员(Ada 语言就是为纪念她而命名的)。1890 年，霍勒内斯(Hollerith)研制出一种同样使用穿孔卡片的统计机，被用于各种统计工作。此后，Hollerith 成立了一个公司，这个公司便是如今的 IBM。

20 世纪 30 年代，英国数学家图灵(Turing)提出了图灵机的概念，它是由一个控制块、一条存储带及一个读写头构成的能执行左移、右移、在存储带中清除或写入符号及进行条件转移等操作的机器。这种图灵机的结构虽然较为简单，但是却能完成现代计算机所能完成的几乎一切运算。随后，车尔赤(Church)发明了一种以逻辑公式中约束变量的代入为主要运算的 λ 演算，这种运算已经相当于一种语法和语义都非常简单的程序设计语言，被广泛应用于程序理论以及程序设计语言理论与实践的研究中。

1. 第一代程序设计语言——机器语言

机器语言是由二进制机器代码构成的代码序列，用来控制计算机执行规定的操作。其特点是能直接反映计算机的硬件结构，并且用机器语言编写的程序无须做任何处理，即可

直接输入计算机执行。机器语言与计算机是一对一的，不同的计算机有不同的指令系统，一种计算机上编写的程序无法直接搬到另一种计算机上运行。一个问题如果需要在多种计算机上求解，就必须对同一问题重复地编写多个应用程序。

2. 第二代程序设计语言——汇编语言

由于机器语言程序的直观性差，且与人们习惯使用的数学表达式及自然语言差距太大，导致机器语言难学、难记，编写出来的程序难以调试、修改、移植和维护，极大程度上限制了计算机的推广应用。在这种情况下，用助记符号来表示机器指令的操作符与操作数(亦称运算符和运算对象)，用地址符号或标号代替指令或操作数的地址的汇编语言出现了。但是，机器不能直接识别用汇编语言编写的程序，还要由汇编语言编译器转换成机器指令后才能运行。

由于汇编语言与机器指令之间是一对一的关系，导致即使是编写一个很简单的程序，也需要数百条指令。所以在汇编语言的基础上，人们又研制出了只需一条指令便可编译成多条机器指令的宏汇编语言。而后，又研制出了用于把多个独立编写的程序块连接组装成一个完整程序的连接程序。但汇编语言大多是针对特定的计算机或计算机系统设计的，所以它对机器的依赖性很强。

3. 高级语言阶段

1954 年，第一个完全脱离机器硬件的高级语言——FORTRAN 语言问世了。高级语言在不同平台上会被编译成不同的机器语言，使得程序设计语言不再过度依赖于某种特定的机器或者语言环境。1970 年，一个标志着结构化程序设计时期开始的语言问世了，它就是 Pascal 语言。这个标志性的语言拥有严格的结构化形式、丰富且完备的数据类型，运行效率高、查错能力强。同时，Pascal 语言还是一种自编译语言。这个以法国数学家 Pascal 命名的语言，在当时已成为使用最广泛的基于 DOS 的语言之一。

20 世纪 80 年代初，在程序设计的思想上又发生了一次大的革命。这个时期研制出的语言多为面向对象的程序设计，之后，高级语言的目标则是面向应用的程序设计。它侧重于描述程序“做什么”而不是“如何做”。

程序设计语言的发展是一个不断演变的过程。从最初的机器语言开始，到汇编语言，再到各种各样的高级语言，最后到支持面向对象技术的面向对象编程语言，甚至未来的面对应用的语言，其演化过程的根本推动力，就是对抽象机制的更高要求，以及对程序设计思想的更好支持。也就是说，把机器能够理解的语言提升到更容易让人类理解的程度。

1.1.2　程序设计语言的分类

程序设计语言可以从不同的角度进行分类，而且一种语言可以分在好几个类别中。对于程序设计语言来说，清楚地了解其所属类别，有利于我们根据项目的特征，选择正确的语言进行软件开发。

1. 按照对机器的依赖程度分类

按照对机器的依赖程度，程序设计语言主要有以下几类。

(1) **低级语言**：面向机器，用机器直接提供的地址码、操作码语义概念编程。如机器语言、汇编语言和宏汇编等。

(2) **高级语言**：独立于机器，用语言提供的语义概念和支持的范型编程。如命令式(Pascal、C、Ada)、函数式(Lisp、M)、逻辑式(Prolog)、关系式(dBASE)、对象式(Smalltalk、C++)等。

(3) **中级语言**：可以编程操纵机器的硬件特征，但不涉及地址码和操作码。如字位运算、取地址、设中断、开辟空间、无用单元发回、用寄存器加速等。如高级汇编、C 语言、Forth 语言等。

2．按照程序设计语言的应用领域分类

根据程序设计语言的应用领域，主要分以下几种。

(1) **商用语言**：处理日常商业事务，有良好的文字表现、报表功能，拥有数据量大和与数据库密切相关等特点。代表语言是 Cobol、RPG、Ada 等。

(2) **科学计算**：数值计算量大，支持高精度、向量、矩阵运算。代表语言有 APL、FORTRAN、Ada 等。

(3) **系统程序设计**：支持与硬件相关的低级操作，是编写系统程序(操作系统，编译、解释器，数据库管理系统，网络接口程序)的语言，如 C、Ada、Bliss、Forth 等。

(4) **模拟语言**：模拟应用主要是以时间为进程，模拟客观世界的状态变化。代表语言有 GPSS、SLAM、SIMULA 等。

(5) **正文处理**：主要操作对象是自然语言中的字符，很方便产生报告、表格等，代表语言是 SMOBOL。

(6) **实时处理**：其特点是能根据外部信号控制不同的程序，做并发执行。此类语言有并发 Pascal、并发 C、Ada、Mesa、OCCAM、FORTHRAN-90、LINDA 等。此外，用于通信领域的具有实时功能的程序设计语言也属于该类，如 GYPSY、CHILL 等。

(7) **嵌入式应用**：在一个大型机器(宿主机)上为小机器(或单片机)开发程序，经调试后，将它译为小机器(目标机)的目标码，在小机器上运行的程序设计语言。这类程序一般都有实时要求，并近于系统设计。代表语言有 Ada。

(8) **人工智能应用**：这类程序是对人们的智力行为的仿真。包括自然语言理解、定理证明、模式识别、机器人、各种专家系统。这类语言能描述知识，并能够推断出合理的结论。在符号运算上做谓词演算或 λ 演算是其推理运算的基本方式。其代表语言有 Lisp 和 Prolog 等。

(9) **查询和命令语言**：这是一类新兴的语言，是各种早期系统程序简单的用户命令的发展。数据库语言有 dBASE、SQL 等。

(10) **教学语言**：为了培训程序员或使学生很快入门，人们设计了教学语言。例如，过程程序设计有 BASIC，结构化程序设计有 Pascal，青少年启蒙有 LOGO 等。

(11) **打印专用**：图文并用在各种打印机(包括激光打印机)上打印字体优美的报告、图形、图像等。代表语言有 PostScript、Tex、LaTeX 等。

(12) **专用于某类数据结构**：专用于处理正文字符串、抽取字符串、引用串函数、串形式匹配、回溯与穷举查找。代表语言有 Snobol、Icon 等。

3. 按照实现计算的方式分类

按照实现计算的方式划分，主要有以下几种。

(1) **编译型语言**：用户将源程序一次写好，提交编译，运行编译的目标码模块，再通过连接编辑，加载成为内存中的可执行目标码程序。再次运行目标码，读入数据，得出计算结果。大多数高级程序设计语言属于这一类。

(2) **解释型语言**：系统的解释程序对源程序直接加工。一边编写，一边执行。不形成再次调用它执行的目标码文件。大多数交互式语言、查询命令语言采用解释型实现。典型的例子有 BASIC、Lisp、Prolog、APL、Shell、SQL。它们的特点是占用空间小、反应快，但运行效率低。

1.1.3　程序设计方法的发展

1. 传统的程序设计方法

传统的编程方法主要是基于 DOS 操作系统下计算机程序的编程方法。用传统的编程方法编制完成特定功能的程序时，必须设计程序的算法，明晰数据的流程。传统编程方法的算法是变化多端的，同一问题可以有最优算法，也可以有一般算法，甚至可能存在劣等算法；它的数据流程是纷繁复杂的，数据的调用、控制方向等又是交叉变化的，而且这种编程方法一般依赖于操作平台、编译系统等，所以可移植性比较差，导致程序的设计也变得困难和繁琐。

2. 可视化编程方法

可视化编程可通过调用控件，并为对象设置属性，根据开发者的需要，直接在窗口中进行用户界面的布局设计。该技术的优点是编程简单、会自动生成程序代码、效率高，因此，在当代编程语言中被广泛采用。

3. 面向对象的编程方法

为了实现整体运算，要求每个对象都能够接收信息、处理数据和向其他对象发送信息，由此应运而生的面对对象的编程方法实现了软件工程的三个主要目标，即重用性、灵活性和扩展性。

面向对象设计是一种把面向对象的思想应用于软件开发过程中，指导开发活动的系统方法，是建立在“对象”概念基础上的方法学。

对象是由数据和允许的操作组成的封装体，与客观实体有直接的对应关系，一个对象类定义了具有相似性质的一组对象。而继承性是对具有层次关系的对象类的属性和操作进行共享的一种方式。

从传统的程序设计方法发展到可视化程序设计方法，进而发展到面向对象的程序设计方法的发展轨迹，是计算机程序设计方法发展的三个重要的阶段。在程序设计实践中，这三种方法即有区别，又相互交叉，彼此紧密联系。但面向对象的程序设计方法是如今应用最为普遍的方法。

1.2 Java 语言简介

1.2.1 Java 语言的历史

1991 年，美国的 Sun Microsystems(Sun)公司为了在消费类电子设备(现在称为智能家电)方面进行前沿研究，建立了以詹姆斯·高斯林(James Gosling)领导的 Green 小组进行软件方面的研究。该小组一开始选择当时已经很成熟的 C++语言进行设计和开发，但是，却发现执行 C++程序需要很多的设备内存，这样将增加硬件的成本，又不利于市场竞争，所以该小组在 C++语言的基础上，创建了一种新的语言。由于詹姆斯很喜欢自己办公室窗外的一棵橡树，所以把该语言的名字叫作 Oak，中文意思是橡树，这就是 Java 语言的前身。但是这个科研小组的成果最终没有转变成 Sun 公司的产品，也没有为 Sun 公司带来什么收益，像很多企业的科研项目一样，Oak 面临夭折的危险。但由于 Oak 专门为内存有限的消费类电子设备而设计，使其执行环境以及程序体积都很小，所以在 1994 年的互联网大潮中重新找到了自己的位置。

随着互联网的发展，以及 Oak 语言与浏览器的融合，产生了一种称作小应用程序(Applet)的技术，当然，该技术后来被 Flash 击败。Applet 是一种将小程序嵌入到网页中执行的技术，使互联网从静态网页过渡到动态网页，也使 Sun 公司的研发小组获得了新生。随后在 1995 年 3 月，Sun 公司正式向外界发布了 Java 语言，Java 语言正式诞生。

1998 年 12 月，JDK 1.2 发布，这是 Java 语言的重要里程碑，Java 也被首次划分为 J2SE/J2EE/J2ME 三个开发技术版本。

不久之后，Sun 公司将 Java 改称为 Java 2，Java 语言也开始被国内开发者学习和使用。2006 年 6 月，JDK 1.6 发布，也称为 Java SE 6.0。同时，Java 的各版本中去掉了 2 的称号，J2EE 改称为 Java EE，J2SE 改称为 Java SE，J2ME 改称为 Java ME。

1.2.2 Java 语言的特点

1. 简单

由于 Java 最初是为对家用电器进行集成控制而设计的一种语言，因此它必须简单明了。Java 语言的简单性主要体现在以下三个方面。

(1) Java 的风格类似于 C++，因而 C++程序员是非常熟悉的。从某种意义上讲，Java 语言是 C 及 C++语言的一个变种，因此，C++程序员可以很快就掌握 Java 编程技术。

(2) Java 摒弃了 C++中容易引发程序错误的地方，如指针和内存管理等。

(3) Java 提供了丰富的类库，如 IO 类库、Exception 类库等。

2. 面向对象

面向对象可以说是 Java 最重要的特性。Java 语言的设计完全是面向对象的，它不支持类似 C 语言那样的面向过程的程序设计技术。Java 支持静态和动态风格的代码继承及重用。单从面向对象的特性来看，Java 类似于 Smalltalk，但其他特性，尤其是适用于分布式

计算环境的特性，远远超越了 Smalltalk。

3. 分布式

Java 包含一个支持 HTTP 和 FTP 等基于 TCP/IP 协议的子库。因此，Java 应用程序可凭借 URL 打开并访问网络上的对象，其访问方式与访问本地文件系统几乎完全相同。

为分布式环境尤其是 Internet 提供动态内容无疑是一项非常艰巨的任务，但 Java 的语法特性却使得这种任务很容易完成。

4. 健壮

Java 致力于检查程序在编译和运行时的错误。类型检查在开发的早期就可以帮助排除许多错误。Java 因自动操纵内存，从而减少了内存出错的可能性。Java 还实现了真数组，避免了覆盖数据的可能。这些功能特征大大缩短了开发 Java 应用程序的周期。Java 还提供 Null 指针检测、数组边界检测、异常检测、字节码校验等。

5. 结构中立

为了确立 Java 在网络中的整体地位，Java 将其程序编译成一种结构中立的中间文件格式。只要是拥有 Java 运行系统的机器，都能执行这种中间代码。例如，Java 可以运行在 Solaris(SPARC)、Win32 等系统中。Java 源程序被编译成一种高层次的与机器无关的 byte-code 格式语言，这种语言被设计在虚拟机上运行，由机器相关的运行调试器实现执行。

6. 安全

Java 的安全性可从两个方面得到保证。一方面，在 Java 语言中，像指针和释放内存等诸如此类的 C++功能已经被删除，从而可以避免非法内存操作。另一方面，用 Java 来创建浏览器时，语言功能和浏览器本身提供的功能结合起来，将会更安全。Java 语言在机器上执行前，要经过很多次的测试，如代码校验、检查代码段的格式、检测指针操作等。

7. 可移植性

可移植性一直是 Java 程序设计师们关注的重要指标，也是 Java 之所以能够受到程序设计师们喜爱的原因之一。这里最大的功臣就是 JVM 技术。

大多数编译器产生的目标代码只能运行在一种 CPU 上，即使那些能支持多种 CPU 的编译器也不能同时产生适合多种 CPU 的目标代码。如果需要在三种 CPU(如 x86、SPARC 和 MIPS)上运行同一程序，就必须编译三次。

但 Java 编译器就不同了。Java 编译器产生的目标代码(J-Code)针对一种并不存在的 CPU——Java 虚拟机(Java Virtual Machine)，而不是某一实际的 CPU。Java 虚拟机能掩盖不同 CPU 之间的差别，使 J-Code 能运行在任何具有 Java 虚拟机的机器上。

虚拟机的概念并不是 Java 所特有的，加州大学就提出过 Pascal 虚拟机的概念。但针对 Internet 应用而设计的 Java 虚拟机的特别之处在于，它能产生安全的、不受病毒威胁的目标代码。正是由于 Internet 对安全特性的特别要求，才使得 JVM 能够迅速被人们接受。

作为一种虚拟的 CPU，Java 虚拟机对于源代码(Source Code)来说是独立的。我们不仅可以用 Java 语言来生成 J-Code，也可以用 Ada 来生成。事实上，已经有了针对若干种源

代码的 J-Code 编译器，包括 Basic、Lisp 和 Forth。源代码一经转换成 J-Code 后，Java 虚拟机就能够执行，而不区分它是由哪种源代码生成的。将源程序编译为 J-Code 的好处在于可运行在各种计算机上，而缺点是，它不如本机代码运行速度快。与体系结构无关的特性使得 Java 应用程序可以在配备了 Java 解释器和运行环境的任何计算机系统上运行，这成为 Java 应用软件便于移植的良好基础。

8. 解释性

Java 解释器(运行系统)能直接运行目标代码指令。连接程序通常比编译程序所需资源少，所以程序员可以在创建源程序上花更多的时间。

9. 高性能

如果解释器速度不慢，Java 可以在运行时直接将目标代码翻译成机器指令。

Sun 用直接解释器一秒钟内可调用三十万个过程。翻译目标代码的速度与 C/C++的性能没什么区别。

10. 多线程

多线程功能使得在一个程序里可同时执行多个小任务。线程有时也称小进程，是一个大进程里分出来的小的独立的进程。因 Java 支持多线程技术，所以比 C 和 C++更健壮。

多线程带来的更大的好处，是更好的交互性能和实时控制性能。当然，实时控制性能还取决于系统本身(Unix、Windows、Macintosh 等)，在开发难易程度和性能上都比单线程要好。任何用过当前浏览器的人，都可感觉到为查看一幅较大的图片而等待是一件很令人烦恼的事情。但在 Java 中，我们可以用一个单线程来调用一幅图片，同时，其他线程可以访问 HTML 里的其他信息，而不用相互等待任务的完成，即实现了 Java 的多线程机制。

11. 动态性

Java 的动态特性是其面向对象设计方法的发展，它允许程序动态地装入运行过程中所需要的类，这是以 C++语言进行面向对象程序设计时所无法实现的。在 C++程序设计过程中，每当在类中增加一个实例变量或一种成员函数后，引用该类的所有子类都必须重新编译，否则将导致程序崩溃。Java 则解决了这个问题。Java 编译器不是将对实例变量和成员函数的引用编译为数值引用，而是将符号引用信息保存在字节码中，然后传递给解释器，再由解释器在完成动态连接后，将符号引用信息转换为数值偏移量。这样，一个在存储器生成的对象不在编译过程中决定，而是延迟到运行时由解释器确定，从而对类中的变量和方法进行更新时就不至于影响现存的代码。解释执行字节码时，这种符号信息的查找和转换过程仅在一个新的名字出现时才进行一次，随后代码便可以全速执行。在运行时确定引用的好处是，可以使用已被更新的类，而不必担心会影响原有的代码。如果程序连接了网络中另一系统中的某一类，该类的所有者也可以自由地对该类进行更新，而不会使任何引用该类的程序崩溃。

Java 还简化了使用一个升级的或全新的协议的方法。如果系统运行 Java 程序时遇到了不知怎样处理的程序，Java 能自动下载所需要的功能程序。

12. 跨平台性

Java 使用 Unicode 作为它的标准字符集，这项特性使得 Java 的程序在不同语言的平台上都能撰写和执行。简单地说，既便把程序中的变量、类别名称使用中文来表示，当程序移植到其他语言平台时，还是可以正常执行。Java 也是目前所有计算机语言中唯一天生就使用 Unicode 的语言。

1.3　Java 运行环境的配置

1.3.1　JDK 的安装

若想正常运行 Java 程序，必须先安装 JDK 环境 jdk-6u10-rc2-bin-b32-windows-i586-p-12_sep_2008。具体安装步骤如下。

(1) 双击安装文件，进入“许可证协议”界面，界面为英文模式，如图 1-1 所示。

(2) 单击“接受”按钮，进入“自定义安装”界面，确定软件安装路径，如图 1-2 所示，也可以单击“更改”按钮改变安装路径，弹出如图 1-3 所示的对话框。

(3) 单击“确定”按钮，进入安装界面，这可能需要等待几分钟时间，如图 1-4 所示。

图 1-1　“许可证协议”界面

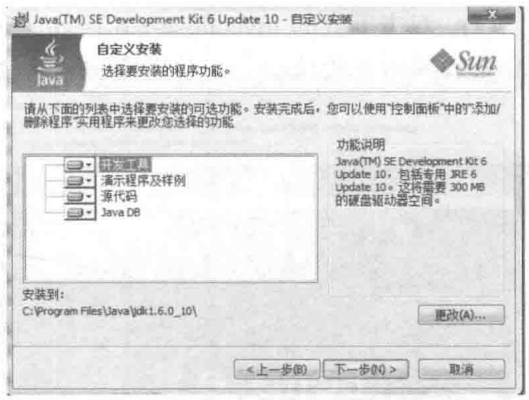


图 1-2　安装路径选择

图 1-3　更改安装路径

图 1-4　安装界面