21世纪应用型本科通用教材

上海市教育委员会高校重点教材建设项目

跨平台程序设计语言
——Java

Cross-Platform Programming Language—Java

组编　上海市教育委员会

主编　赵毅

西安电子科技大学出版社

http://www.xduph.com

21 世纪应用型本科通用教材

上海市教育委员会高校重点教材建设项目

跨平台程序设计语言——Java

Cross-Platform Programming Language—Java

组　编　上海市教育委员会

主　编　赵　毅

副主编　潘　勇

参　编　王明衍　黄　容　章颖芳

西安电子科技大学出版社

2006

内 容 简 介

本书根据普通大专院校学生的特点，结合先进的教学思路，以简洁明了的风格和大量实例，全面而详细地讲述了 Java 程序设计语言的基本概念与编程思想，使读者能快速理解和迅速掌握相应编程原理，并学以致用。

全书共 12 章，其中第 1～4 章讲解了程序设计语言的基本概念和过程设计的方法，第 5～8 章讲解了面向对象程序设计的原理，第 9 章和第 10 章介绍了编写图形界面的基本方法，第 11 章和第 12 章介绍了 I/O 技术和多线程技术。此外，书末还给出了三个附录，分别为常用类的使用方法、常用开发工具和常用词汇表。

本书可作为大专院校非计算机专业的计算机程序设计教材，也可作为编程爱好者的参考书及 Java 语言的自学教材，特别适合于缺乏程序设计经验的初学者。

★本书配有电子教案，需要者可与出版社联系，免费提供。

图书在版编目（CIP）数据

跨平台程序设计语言——Java / 赵毅主编. —西安：西安电子科技大学出版社，2006.5

ISBN 7-5606-1659-3

Ⅰ. 跨… Ⅱ. 赵… Ⅲ. Java 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2006）第 026308 号

策　　划　毛红兵

责任编辑　雷鸿俊　毛红兵

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)88242885　88201467　　邮　编　710071

http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印刷单位　陕西天意印务有限责任公司

版　　次　2006 年 5 月第 1 版　2006 年 5 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印张 21.625

字　　数　508 千字

印　　数　1～4000 册

定　　价　24.00 元

ISBN 7-5606-1659-3/TP・0401

XDUP 1951001-1

*****如有印装问题可调换*****

前　言

本书是基于 Java2 的基础教程。近几年 SUN 公司先后推出的 J2EE、J2SE、J2ME 三大平台开发工具已成为软件开发人员青睐的工具。

随着计算机技术在各个领域的广泛应用以及计算机系统平台和软件开发平台的不断更新，人们对程序设计语言的要求越来越高，希望程序设计语言能方便软件开发，所开发的软件能在各种系统平台上正常运行并能很好地应用于互联网。作为一种程序设计语言的 Java，它具备了一些独特的优点。例如，Java 是分布式的、面向对象的、不依赖于计算机系统的结构，这样所开发的软件就具备了跨平台、可移植的功能。随着互联网的快速发展及网络时代的到来，各种平台的系统相互连接，Java 与平台无关的优势得到了充分的体现，由 Java 编写的程序可在各种平台上应用，避免了软件重复开发。Java 的最新编译和运作软件几乎都可以从网上免费下载，这为广大编程人员学习和应用 Java 创造了有利条件。

国家计算机程序员职业标准(Java 版)要求，合格的程序员应具有编写 Java 一般应用程序和小应用程序的能力。本书就是根据这些要求编写的，主要内容包括 Java 语言基础、Java 面向对象的程序设计、Java 输入/输出流、图形用户界面、多线程程序设计、Java 小应用程序的开发和异常处理技术等。在掌握了这些知识后，读者将具有编写 Java 一般应用程序和小应用程序的能力。

本书通俗易懂，由浅至深。考虑到本书主要是面向非计算机专业的学生和编程爱好者的，因此我们从编程的最基本概念开始讲解，然后再介绍面向对象的编程原理和 Java 技术的应用。对开发环境的安装、配置及程序调试等技术都作了详细的介绍，习题也体现了全方位培养读者的编程思想，既注重基本知识的理解，又注意学习其他人的编程方法。需要与本书配套的授课课件或实验课件的人员可与本书主编联系。

参与此书编写的作者如下：第 1～4 章及附录 B 由潘勇编写；第 5 章和第 8 章由王明衍编写；第 6 章和第 7 章由章颖芳编写；第 10 章及附录 A 由黄容编写；第 9、11、12 章及附录 C 由赵毅编写。赵毅组织和审阅了全书。另外，洪菁也为本书的出版做了很多工作，上海工程技术大学有关专家和教师也给予了大力支持，在此向他们表示衷心的感谢。

本教材的组织编写得到了上海市教委“高等学校优秀教材”项目基金的资助。

由于时间仓促、作者水平有限，书中难免有不妥之处，恳请广大读者批评指正。

编　者

2006 年 1 月

目　录

第 1 章　程序设计概述

本章首先简要介绍程序设计的基本概念和面向对象程序设计的基本思想，然后进一步介绍 Java 语言的基本特点和简单的程序设计。

1.1　程序设计语言的分类

计算机能够接受的、指示计算机完成特定功能的命令序列称为程序；编写程序的过程称为程序设计；用于描述程序中操作过程的命令、规则的符号集合称为程序设计语言。

程序设计语言是学习计算机技术的基础，是一种面向计算机的人工语言，它经历了较长的发展历程。程序设计语言的分类方法很多，下面介绍几种常见的分类方法。

1. 按照发展过程分类

从计算机诞生到今天信息技术的广泛应用，计算机程序设计语言经历了以下几个阶段。

1) 机器语言

机器语言是最初级的且依赖于硬件的计算机语言，是以二进制代码形式组成的机器指令集合，不同的机器有不同的机器语言。这种语言编制的程序运行效率非常高，但是程序很不直观，编写简单的功能就需要大量的代码，重用性差，很容易出现错误。

2) 汇编语言

汇编语言相对机器语言而言比较直观，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，编程工作相对机器语言有了较大的简化，使用起来方便了不少，错误也相对减少了。但是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程序设计语言，读/写接近人类的自然语言，因此，使用高级语言开发的程序可读性较好，便于维护。同时，由于高级语言不直接和硬件相关，因此用该语言编制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、Basic、Java 等，现代应用程序大多数都使用这些高级语言。

2. 按照程序执行方式分类

程序语言在计算机内通过编辑器输入完成后，就可以提交计算机执行了。按其在计算机内的执行方式，可将程序设计语言分为以下两种：

1) 编译执行方式的语言

编译执行是在编写、编辑完成程序之后，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来执行程序的。可执行程序可以在操作系统平台上运行。常见的编译执行

方式的语言有 Pascal、C 等。

2) 解释执行方式的语言

解释执行是对高级语言源程序进行逐句的翻译，翻译一句执行一句，不需要整体编译，这样的语言与操作系统的相关性比较小，但运行效率低。不过，解释执行的语言相对而言调试比较方便。常见的解释执行方式的语言有 Basic、Java 等。

3. 按照思维方式分类

程序设计语言总是需要按照某种思维方式进行设计和实现的，因此不同的语言可能有不同的思维方式。目前存在以下两种思维方式：

1) 面向过程的程序设计语言

面向过程(Process Oriented)就是要以解决的问题为思考的出发点和核心，并使用计算机语言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，主要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 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、Pascal、C 等。这类语言面向求解问题的过程，不依赖于计算机的硬件，可移植性较好，当计算机所要解决的问题不是非常复杂，适用范围不是非常广泛时，它们是非常有效的解决问题的方法。

但面向过程的程序设计语言的缺点也是十分明显的：该语言极度面向过程，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，需要程序员对程序做较大的修改。而且不同的问题需要不同的程序来解决，问题与解决几乎是一一对应的，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。随着近年来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了。而解决这些问题最有效的方法就是另外一种面向对象的思维方式。

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，而且是一种崭新的思维方式。面向对象的思想是以一种更接近人类一般思维的方式去看待世界，把世界上的任何个体都看作对象，每个对象都有自己的特点，以自己的方式行动，不同的对象之间存在着通信和交互，由此构成世界的运行。从程序员的角度来看，对象的特点就是它们的属性，而对象的行动就是它们的方法。常见的面向对象的程序设计语言有 C++、Java 等。

面向对象的方法大大提高了程序的重用性，而且显著降低了程序的复杂度，使得计算机程序设计能够适应越来越复杂的应用需求。其中最为突出的就是 Java 语言。在如今的电子商务时代，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　面向对象的相关概念

面向对象技术的基本概念很多，这些概念与我们思考和解决问题的方式有着密切的一一对应的关系。下面介绍其中常见的几个基本概念。

1. 对象

对象代表现实世界的实体，这里的对象只有在某个特定的应用程序中才有意义。

每个对象都具有自身的属性或特征，通过该属性或特征来描述它是什么或者做什么。例如，“人”这一对象的属性包括姓名、年龄、体重等；“汽车”这一对象的属性包括颜色、重量、年份、发动机功率等。

此外，对象还可以进行一些活动。例如，汽车可以进行的活动包括发动、停车、加速、倒车等。

程序设计对象和现实世界对象之间的对应关系应该是属性和活动相结合的结果。

2. 类

类是相似对象的集合。类定义对象的特征，只有在创建了对象之后才可以给它赋值。

对象的特征用变量来表示，称为类的属性。例如，类中的每个人都有姓名、年龄、性别并可能从事某种工作，这些都是类中的所有人都具有的属性。

类中表示对象或实体的特征称为属性。类中表示的对象或实体所需的活动称为方法。

3. 抽象

数据抽象是识别与应用程序相关的特定实体的属性和方法的过程。

将对象组合到类中，实际上就是对问题进行数据抽象。

4. 继承

在生活中我们所使用的类可以划分成多个子类。例如，动物类可以划分成哺乳动物、两栖动物、昆虫和爬行动物等；交通工具类可以划分成汽车、卡车、公共汽车和摩托车等。

继承是重用现有类来生成新类的一种特性。

5. 封装

可以用黑盒子来解释面向对象编程的基本原则：作为类的使用者，永远不需要看到盒子内部。类有许多属性和方法，使用者不需要访问所有的属性和方法。

封装是允许有选择地隐藏类中的属性和方法的过程。

从编程的角度来看，由于无法看到对象的黑盒子内部，因此我们不能直接修改该对象。正确地封装了一些代码之后，可以实现两个目标：① 建起一堵墙，保护代码不会受到一些小错误的意外破坏；② 将错误隔离在小范围内，使得它们更易于查找和修改。

6. 可重用性

伴随数据和操作的抽象、封装一起出现的还有可重用性。通过可重用性，面向对象的开发可以实现两个目标：① 在应用程序中共享信息；② 在未来的项目中重用设计和代码。

所有面向对象的语言都努力使程序的各个部分可以更容易地进行重用和扩展。程序可以分解为可重用的对象，这些对象又可以重新组合起来，形成新的程序。程序员可以使用现有的类并添加附加的功能，而不用修改原来的类。有了重用代码的方式后，程序员就可以通过组装现有的程序片段，更容易地编写新程序。

面向对象编程的主要技巧在于如何将程序划分为一组简单的类。除了加快开发速度以外，类的恰当构建和重用还可以大大减少代码行数，这也意味着更少的缺陷和更低的维护费用。

7. 多态性

面向对象的语言试图使现有的代码更易于修改，而不是真正地更改代码。这是一种独特而非常强大的概念，因为要在不进行更改的情况下修改内容，看起来似乎是不可能的，然而使用继承和多态性将使其成为可能。

多态性是指相同的函数可以在不同的类中有不同的行为。现有对象保持不变，所做的任何更改只是对其进行添加。由于不需要更改原始对象，程序员可以在维护和修改代码时减少犯错误的机会。

多态性使得相同的函数在不同的类中有不同的行为。例如，在“多边形”类中，“绘制”是所有子类都共享的方法，但是，实现长方形的绘制方法和实现椭圆形的绘制方法就有很大的不同。

1.3 Java 简介

对于计算机语言的发展史，业界一般认为：B 语言导致了 C 语言的诞生，C 语言演变出了 C++语言，而 C++语言将让位于 Java 语言。要想更好地了解 Java 语言，就必须了解它产生的原因、推动它发展的动力，以及它对其他语言的继承。像以前其他成功的计算机语言一样，Java 继承了其他语言的先进原理，同时又因其独特的环境要求而提出了一些创新性的概念。在本书的其他各章中，将从实用的角度，对 Java 语言、库及应用程序包括语法在内的相关内容进行详细介绍。在本章中，我们将介绍 Java 语言产生的背景、发展过程，以及使它变得如此重要的原因。

尽管 Java 语言已和 Internet 的在线环境密不可分，但首先应该注意到的最重要的一点是：它是一种程序语言。计算机语言的革新和发展需要两个基本因素的驱动：① 适应正在变化的环境和需求；② 实现编程艺术的完善与提高。

Java 总是和 C++联系在一起，而 C++则是从 C 语言派生而来的，所以 Java 语言继承了这两种语言的大部分特性。Java 的语法是从 C 继承的，Java 许多面向对象的特性受到了 C++的影响。事实上，Java 中几个自定义的特性都来自于或可以追溯到它的前驱。而且，Java 语言的产生与过去 30 年中计算机语言的细致改进和不断发展密切相关。基于这些原因，本节将按顺序回顾促使 Java 产生的事件和推动力。每一次语言设计的革新都是因为先前的语言不能解决目前遇到的基本问题而引起的，Java 也不例外。

1.3.1 Java 的由来

1. 现代编程语言的诞生——C 语言

C 语言的产生震撼了整个计算机界，它从根本上改变了人们编程的方法和思路。C 语言的产生是人们追求结构化、高效率、高级语言的直接结果，可用它替代汇编语言开发系统程序。当设计一种计算机语言时，经常要从易用性与功能、安全性和效率性、稳定性和可扩展性等几方面进行权衡。

C 语言出现以前，程序员们不得不经常在有优点但在某些方面又有欠缺的语言之间做出选择。例如，尽管公认 FORTRAN 在科学计算应用方面可以编写出相当高效的程序，但

它不适于编写系统程序；Basic 虽然容易学习，但功能不够强大，并且谈不上结构化，这使它应用到大程序的有效性受到怀疑；汇编语言虽能编写出高效率的程序，但是学习或有效地使用它却是不容易的，而且调试汇编程序也相当困难。

另一个复杂的问题是，早期设计的计算机语言(如 Basic，COBOL，FORTRAN 等)没有考虑结构化设计原则，使用 goto 语句作为对程序进行控制的一种主要方法。这样做的结果是，用这些语言编写的程序往往成了“意大利面条式的程序代码”，一大堆混乱的跳转语句和条件分支语句使得程序几乎不可能被读懂。Pascal 虽然是结构化语言，但它的设计效率比较低，也难以在大规模的编程中使用。

因此，在 C 语言产生以前，没有任何一种语言能完全满足人们的需要，而人们对这样一种语言的需要是迫切的。在 20 世纪 70 年代初期，计算机革命开始了，对软件的需求量日益增加，使用早期的计算机语言进行软件开发根本无法满足这种需要。学术界付出了很多努力，尝试创造一种更好的计算机语言。但是，促使 C 语言诞生的另一个，也是最重要的因素是计算机硬件资源的富余带来了机遇。计算机不再像以前那样被紧锁在门里，程序员们可以随意使用计算机，可以随意进行自由尝试，因而也就有了可以开发适合自己使用的工具的机会。

在 Dennis Ritchie 第一个发明和实现在 DEC PDP-11 上运行 UNIX 操作系统时，一种更古老的由 Martin Richards 设计的 BCPL 语言导致了 C 语言的产生。受 BCPL 语言的影响，由 Ken Thompson 发明的 B 语言，在 20 世纪 70 年代逐渐向 C 语言发展演变。在此后的许多年里，由 Brian Kernighan 和 Dennis Ritchie 编写的《The C Programming Language》(Prentice-Hall，1978)被认为是事实上的 C 语言标准，该书认为 C 只是支持 UNIX 操作系统的一种语言。1989 年 12 月，美国国家标准化组织(ANSI)制定了 C 语言的标准，C 语言被正式标准化。

许多人认为，C 语言的产生标志着现代计算机语言时代的开始。它成功地综合处理了长期困扰早期语言的矛盾。C 语言是功能强大、高效的结构化语言，简单易学，而且它还包括一个无形的方面：它是程序员自己的语言。它的设计、实现、开发由真正从事编程工作的程序员来完成，反映了现实编程工作的方法。它的特性经由实际运用该语言的人们不断地提炼、测试、思考、再思考，使得 C 语言成为程序员们喜欢使用的语言。

2．对 C++的需要

在 20 世纪 70 年代末和 80 年代初，C 成为了主流的计算机编程语言，至今仍被广泛使用。你也许会问，既然 C 是一种成功且有用的语言，为什么还需要新的计算机语言？这是由程序的复杂性(Complexity)决定的。程序越来越复杂这一事实贯穿于编程语言的历史，C++正是适应复杂性这一需求而产生的。

自从计算机发明以来，编程方法经历了戏剧性的变化。例如，当计算机刚发明出来时，编程是通过面板触发器用人工打孔的办法输入二进制机器指令来实现的。对于只有几百行的程序，这种办法是可行的。随着程序不断增大，人们发明了汇编语言，它通过使用符号来代替机器指令，这样程序员就能处理更大、更复杂的程序。随着程序的进一步增大，高级语言产生了，它给程序员提供了更多的工具来处理某些复杂的问题。

第一个被广泛使用的高级语言是 FORTRAN。尽管 FORTRAN 最初给人留下了深刻的印象，但人们利用它无法开发出条理清楚、易于理解的程序。20 世纪 60 年代，人们提出了结构化编程方法。这种结构化的编程思想被像 C 这样的语言所应用，第一次使程序员可以相对轻松地编写适度复杂的程序。然而，当一个工程项目达到一定规模后，即使使用结构化编程方法，编程人员也无法对它的复杂性进行有效管理。20 世纪 80 年代初期，许多工程项目的复杂性都超过了结构化方法的极限。为解决这个问题，面向对象编程(Object-Oriented Programming，OOP)新方法诞生了。面向对象的编程是通过使用继承性、封装性和多态性来帮助组织复杂程序的编程方法。本书后面将对该方法进行详细讨论。

总之，尽管 C 是世界上伟大的编程语言之一，但它处理复杂性的能力有限。一旦一个程序的代码超过 25 000～100 000 行，就很难从总体上把握它的复杂性了。C++突破了这个限制，可帮助程序员理解并且管理更大的程序。

1979 年，当 Bjarne Stroustrup 在美国新泽西州的 Murray Hill 实验室工作时，发明了 C++。Stroustrup 最初把这种新语言称为“带类的 C”，1983 年将其改名为 C++。C++ 通过增加面向对象的特性扩充了 C。由于 C++产生在 C 的基础之上，因此它包括了 C 所有的特征、属性和优点。这是 C++成功的一个关键原因。C++的发明不是企图创造一种全新的编程语言，而是对一个已经高度成功的语言的改进。C++在 1997 年 11 月被标准化，目前的标准是 ANSI/ISO。

3. Java 应运而生

在 20 世纪 80 年代末和 90 年代初，使用面向对象编程的 C++语言占据了主导地位。曾经有一段时间一些程序员似乎都认为已经找到了一种完美的语言。因为 C++ 有面向对象的特征，又有 C 语言高效和格式上的优点，所以人们认为它是一种可以被广泛应用的编程语言。然而，就像过去一样，推动计算机语言进化的力量始终在酝酿。在随后的几年里，万维网(WWW)和因特网(Internet)的迅猛发展促成了编程的另一场革命，Java 由此应运而生。

1.3.2 Java 的历史和现状

Java 是由 James Gosling、Patrick Naughton、Chris Warth、Ed Frank 和 Mike Sheridan 于 1991 年在 SUN Microsystems 公司设计出来的。他们开发第一个版本时用了 18 个月。该语言开始名叫“Oak”，于 1995 年更名为“Java”。从 1992 年秋 Oak 问世到 1995 年春公开发布 Java 语言，许多人对 Java 的设计和改进都做出了贡献。

事实上，Java 的最初推动力并不是因特网，而是源于对独立于平台(也就是体系结构中立)语言的需要，这种语言可创建能够嵌入微波炉、遥控器等各种家用电器设备的软件。用作控制器的 CPU 芯片是多种多样的，但 C 和 C++以及其他多数语言的缺点是只能对特定目标进行编译。尽管为任何类型的 CPU 芯片编译 C++程序是可能的，但这样做需要一个完整的以该 CPU 为目标的 C++编译器，而创建编译器是一项既耗资巨大又耗时较长的工作。因此需要一种简单且经济的解决方案。为了找到这样一种方案，Gosling 和其他人开始一起致力于开发一种可移植、跨平台的语言，该语言能够生成运行于不同环境、不同 CPU 芯片上的代码。他们的努力最终促成了 Java 的诞生。