面向对象程序设计系列教材

Java与面向对象程序设计教程

印旻

高等教育出版社

Java与面向对象程序设计教程
高等教育出版社

面向对象程序设计系列教材

Java 与面向对象程序设计教程

印 旻

高等教育出版社

（京）112 号

内 容 提 要

本书将 Java 语言编程技术和面向对象程序设计相结合，在讲授 Java 这个 Internet 上最流行的编程工具的同时，介绍了 Java 所采用的面向对象技术的基础理论、主要原则和思维方法。

书中首先介绍了 Java 语言的基础知识；然后阐述了面向对象程序设计的基本原则与特点，并借助于 Java 把这些原则和特点融入具体的程序中，帮助读者建立由感性到理性的认识；最后介绍了 Java 编程的必备知识和工具，包括类库、常用算法和数据结构的 Java 描述、GUI 和网络编程等。全书不仅介绍理论，更强调实际运用，特别注重提高读者运用 Java 语言和面向对象技术解决实际问题的能力。书中给出了大量经过调试运行的实例，便于初学者入门。

本书可作为高等学校计算机及相关专业程序设计课程的入门教材，也可作为各学校程序设计公共选修课的教材，本书也适用于职业教育或从事实际软件开发的读者学习使用。

本书配有教学辅助课件，可供教师和学生使用。需要者可与清华大学计算中心王行言老师联系

邮政编码：100084　电话：010 62782934　E-mail: xywang@tsinghua.edu.cn

图书在版编目(CIP)数据

Java 与面向对象程序设计教程/印旻. －北京 ：高等教育出版社，1999
ISBN 7-04-007501-6

Ⅰ.J…　Ⅱ.印…　Ⅲ.JAVA 语言-程序设计-教材　Ⅳ.TP 312

中国版本图书馆 CIP 数据核字(1999)第 68059 号

Java 与面向对象程序设计教程
印　旻

出版发行	高等教育出版社		
社　　址	北京市东城区沙滩后街 55 号	**邮政编码**	100009
电　　话	010－64054588	**传　　真**	010－64014048
网　　址	http://www.hep.edu.cn		
经　　销	新华书店北京发行所发行		
印　　刷	国防工业出版社印刷厂		
开　　本	787×1092　1/16	**版　　次**	1999 年 11 月第 1 版
印　　张	23.25	**印　　次**	1999 年 11 月第 1 版
字　　数	560 000	**定　　价**	24.00 元

前　言

能够把“Java 语言”和“面向对象的编程技术”这两种计算机应用领域中飞速崛起和普及的新兴技术结合在一起介绍给广大读者，是笔者的荣幸。面向对象技术被认为是程序设计方法学的一场革命，它已经逐步替代面向过程的程序设计技术，而成为计算机应用开发领域的主流趋势；这是因为这种技术所提倡的模拟现实世界的思维方式、数据与操作相捆绑的程序风格符合了现代大规模软件开发的要求和潮流。Java 语言就是面向对象技术成功应用的一个最著名范例之一。诞生于 1995 年的 Java 语言在短短的四五年间席卷全球，以 20 世纪末网络科技和网络经济所特有的令人瞠目结舌的速度迅速发展，有人预言，不久的将来全世界 90％的程序代码将用 Java 语言书写或改写，Java 的平台无关特性、安全机制、高可靠性和内嵌的网络支持，使之成为当前编写网络应用的首选工具之一。本书将以 Java 语言为载体，在介绍 Java 编程的同时讲解面向对象程序设计的主要原则和方法。

Java 语言的成功，相当大程度上归功于它顺应时代潮流的“网络计算”定位，同样，一本有价值的读物也需要一个合适的定位，本书的读者群体定位为大中专院校的低年级学生以及其他同等程度的对 Java 语言和面向对象编程技术感兴趣的读者。根据这个定位，笔者对书中的内容编排、剪裁和例题选择做了严格的控制，确保了一定的深度和广度，在难易程度上亦遵循由浅入深、循序渐进的原则。本书可以作为大中专院校的公共课教材（书中的内容相对于一个学期的课程略显偏多，教师可根据实际情况加以取舍），也可以作为感兴趣读者的自学用书。学习本书之前应该对计算机操作有一定的认识，但是不必有开发经验，接触过一门高级语言的读者会感到学习本书得心应手，从未编过程序的读者只要对于前三章多花些心思，一样可以获得良好的学习效果。

有别于一般的操作手册类丛书，本书并非仅仅以教授 Java 语言为终极使命。笔者认为，软件开发是触类旁通的技术，关键是要掌握其特定的思维方式方法，为此在详尽介绍 Java 语言程序设计的同时，尽个人能力把面向对象的思维方法和网络计算的全局观念贯穿于全书并加以强调。虽因水平有限，自感未能完善地表达出这种意图，但还是希望读者能在阅读过程中领会此用心，在学习的同时不忘思考和总结，争取在程序设计领域站得更高。另外根据试用期间同学们的反馈意见，在 Java 的语法规则之外增加了“常用算法和工具”一章，使本书更加实用，更好地满足读者用 Java 解决实际问题的需求。

本书的内容源自笔者在清华大学开设的一门深受欢迎的公共选修课“Java 语言与面向对象程序设计”，此课程自 1997 年秋季开设以来每每爆满，每次报名选课的人数都逾千人，一度成为清华大学选课人数最多的公共任选课。同学们对新技术的敏感和渴求在深深感动我们之余，也激励和敦促我们努力改进教学内容、方式和效果。本书就是这种努力的成果之一，它总结了笔者两年以来的大量教学经验和心得，书中的个别例题的灵感甚至直接来源于同学们的提问和作业，为此要在这里感谢所有关心和支持这门课程的领导、老师和同学，以及笔者的丈夫和家人，特别要感谢清华大学计算中心的王行言主任，正是他的远见和全力支持才促成了本书的付梓。

最后提醒广大读者，计算机学科是注重实践的学科，优秀的软件开发人员无不经过大量上机实践的磨炼；只有在学习书本内容的同时辅以相应的实际练习和实验环节，才能真正掌握书中介绍的知识和技能。为此本书中引入了大量的例题，并另配有一本专门的习题集和上机手册，只要读者能够按照书中的要求边学边练，一定能很快登堂入室，享受在 Java 语言和面向对象技术所构造的无限畅想空间中遨游的乐趣。好了，准备好一部能够联网的计算机了吗？愿本书成为你软件开发生涯起飞道路上的一块重要铺路石，希望它能伴你一段愉快的历程！

作者

1999 年 9 月

责任编辑　刘建元
封面设计　李卫青
责任印制　杨　明

目　录

第一章　软件开发基础与 Java 语言概述

1.1　软件开发基础

在具体学习 Java 程序设计语言之前，让我们先了解、回顾若干计算机软件开发的基础知识，明了软件开发在整个计算机系统中所处的位置和环境，它的目的和任务，以及软件开发的一般过程与原则。一名合格的软件开发人员必须首先建立这些正确的基本概念，才能在学习程序设计语言的过程中有的放矢，而不会产生“语言到底有什么用？”或者“为什么有那么多语言，我还要再学习其他的吗？”之类含糊的问题。

1.1.1　软件运行原理

计算机是人类 20 世纪最伟大、最重要的发明之一，这个发明的最伟大之处就在于它能够以惊人的效率和前所未有的智能化来辅助人们更好地完成认识自然和改造自然的工作，它是有史以来第一种能够完成真正意义上的复杂的“学习”功能的机器，这就使得它具有了某种更接近人类的“思考”能力；与此同时，计算机所特有的超人的计算能力可以把人们的工作效率和生产效率成千上万倍提升，从而把人从最直接、最原始的生产第一线上解放出来，转而从事使用和操纵计算机的工作。

正如人体是一个复杂的有机体一样，计算机也是由不同部分组成的非常复杂的系统，计算机正常、准确的工作将依赖于这些部分的正常工作和它们之间的相互配合。于是，在使用计算机的人们之中就引入了分工和专业细化，不同的专业将负责研究计算机系统的不同方面。例如，如果把计算机比作是一个工人，硬件工程师将负责其身体各部分健康、完好；软件工程师将教会他如何学习和工作；而计算机的操纵人员将向这个身体健康并学有所长的工人布置任务并监督其保质保量地工作。换句话说，计算机由硬件工程师为其赋予生命，由软件工程师为其注入灵魂，并最终在千千万万的操作人员手中发挥其威力和作用。本书就是写给那些有志于为计算机注入灵魂的人们的。

当然，要成为一个合格的软件开发人员，要学习的东西还很多，本书所提供的知识和技能仅仅是众多的铺路石中较为基础的一块——程序设计方法和程序设计语言。程序设计语言是软件开发人员与计算机进行沟通和交流的语言，是计算机这个工人能够明了和辨别的语言。只有掌握了程序设计语言，软件开发人员才能像熟识鸟语的公冶长号令百鸟一样，指挥计算机按照自己的意志完成种种复杂的工作。同时，作为一名软件开发人员，仅仅掌握程序设计语言是不够的，他（她）还需要对硬件知识有所了解，这就像是在对工人进行培训之前，首先必须对其身体和心理状况有所了解，才能够充分运用和调动其积极性来提高学习效率一样。

简单地说，计算机由中央处理单元、数学逻辑单元、内存单元、输入单元、输出单元和

外存单元组成。其中数学逻辑单元（ALU）负责完成具体的数学和逻辑运算，如加、减、乘、除等，任何一种软件任务都最终被分解、细化成若干以数学或逻辑运算为核心的原子级子任务，这些子任务将按照一定的顺序传送到ALU中加以计算；输入单元和输出单元分别负责在计算机系统和外界之间接受和传送信息；内存单元负责暂时保存数据或即将运行的程序，以及运算过程中的中间结果；外存单元的速度较慢，但是容量很大，可以永久性地保存程序和数据，并在必要的时候与内存单元进行数据交换；中央处理单元就是大名鼎鼎的CPU，它负责调度、管理计算机的其他单元并保证它们和谐工作。例如，CPU可以指挥输入单元读入数据，送到ALU中运算，并把结果传到输出单元显示出来。

我们知道，一个计算机系统仅仅有硬件什么都做不成，只有当软件在其上运行的时候，它才是“活”的，才具有一定的功能。那么在上述的硬件环境下，软件是如何运行的呢？我们现在谈论的计算机系统，是目前占主导地位的冯·诺依曼式计算机系统，它的核心思想就是“存储＋运行”，即在运行一个软件之前，需要首先将其存储在计算机系统的内存单元之中。更具体地说，就是将这个软件所包含的程序和数据分别保存到内存的特定位置中。计算机软件是由程序、运行程序所需要的数据和文档三部分组成的，其中程序和数据是运行软件所必需的。软件运行之前将其保存到内存中的过程称为内存加载或调入内存，这个内存加载的步骤是由CPU执行的。加载成功之后，CPU将从内存中依次取出该软件程序的每一条指令顺序执行。在执行过程中，CPU可能需要内存中这个软件的或其他软件的数据，可能需要调动输入、输出单元完成输入、输出操作，也可能要调度其他的软件指令配合工作。这一切，都取决于开发人员事先编写好并已经加载在内存中的程序指令。这就是计算机系统中软件运行的基本过程和原理。

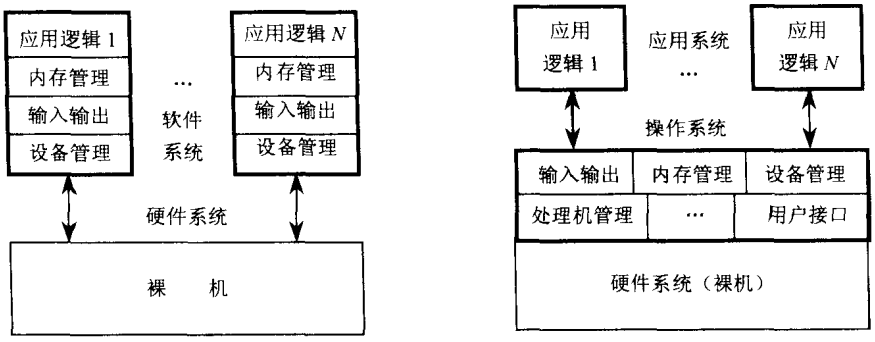
1.1.2 操作系统与计算结构的发展

1. 操作系统简介

没有安装任何软件而只有硬件的计算机系统被称为“裸机”。裸机只能接受二进制的指令，即它只能运行由单纯的0和1组成的二进制程序，所以使用起来非常不方便；而且由于裸机本身的功能非常原始和有限，故直接运行在裸机上的程序需要负担起各种各样的烦琐工作，更增加了编程和管理的难度。把一个复杂的问题分解成若干个较简单的子问题分别加以研究是一种常见的解决问题的思路，在对待上述问题时，人们就采用了这样的思路。正像复杂的计算机系统被分解为硬件、软件等部分一样，复杂的软件系统也被进一步划分为若干层次：软件中负责与裸机硬件直接打交道的功能（如设备管理）、一些常用的功能（如输入、输出、文件管理）以及自我管理以便提高效率的功能（如处理机管理、内存管理）被从程序中分解出来，专门编写成单独的软件，这部分软件就被称为操作系统。而软件中的其他剩余部分通常用来完成一些较特定的功能，可以称为应用系统。

由于操作系统所完成的功能是一个计算机系统所必备的、通用的和基础的功能，所以操作系统将被直接安装保存在裸机中。每当计算机开启时，操作系统将首先被自动调入内存运行并待命；当我们希望计算机系统完成某个特定的任务时，我们所编写的程序将运行在操作系统之上并接受操作系统提供的服务，而不是直接运行在裸机之上。这样，我们就可以不再为那些最基础的功能而费心，而只需调用操作系统业已提供的基础服务就可以了（如图 1.1

所示）。打一个简单的比喻，一名同学刚开始学习计算机操作的时候，老师要手把手地指导：如何连接电源，如何打开开关，如何使用键盘，如何使用文字编辑软件，如何发送电子邮件，等等。经过一段时间的学习之后，教师就可以把精力集中在更高层次的教学，如程序设计上面，而不再需要每每从头教起，因为训练有素的学生们已经能够明了并熟练地执行老师发出的诸如“开机”、“输入一段程序”和“用电子邮件上交作业”这样一些基本操作命令了。配备了操作系统的计算机，从某种程度上来说，就像是这些经过了训练的学生。



(a) 直接运行于裸机的软件系统　　(b) 操作系统与应用系统的划分

图 1.1

从图 1.1(b)中可以看出，操作系统界于裸机和应用系统之间，需要同时与两者打交道。这就决定了操作系统的两个特点：首先操作系统是直接运行在裸机上的，所以它对计算机硬件有很强的依赖性，在一种型号的计算机上能够正常工作的操作系统，到了另一种型号的计算机上就可能会有问题；如果计算机系统新增了什么设备，也需要在操作系统中添加相应的驱动软件才能够保证该设备的正常工作。其次操作系统是面向应用系统的，它需要为应用系统提供方便有效的支持和服务。操作系统对应用系统的服务有多种形式，有直接的，也有间接的，比较常见的有用户接口和功能调用两种：用户接口提供一系列的命令使用户本人直接使用操作系统的功能；功能调用则向运行在操作系统上的应用系统的程序提供可调用的功能模块，应用系统通过调用这些功能模块，可以方便地实现一些操作系统级的功能，获取操作系统的服务。目前较常见的台式机操作系统分为两大阵营：一是美国微软公司开发的 Windows 系列，最具代表性的是手持设备上的嵌入式操作系统 Windows CE、单机上的操作系统 Windows 98 以及网络操作系统 Windows NT 等；另一个是以 SUN、Novell 等厂商为代表的 UNIX 系列，新近较为流行的自由软件的代表 Linux 也属于这个系列。

正如计算机系统划分为硬件系统和软件系统之后，其中的软件系统可以再进一步地划分为操作系统和应用系统一样；软件系统划分之后，其中的应用系统也还可以进一步细分：通常应用系统是用来完成某种特定的任务的，这个任务一般都有特定的实用背景，如金融领域的应用、办公自动化领域的应用、实时控制领域的应用等。在某一个特定的领域中的所有应用系统，它们相互之间会有一些共同或相似之处；而即便是不同领域的应用系统，它们也可能需要一些相同的功能，比如金融领域的应用和办公领域的应用都需要用到数据库来管理和保存信息，文字处理、网络浏览和电子邮件收发等更是大多数计算机用户经常使用的功能。为此，应用系统被进一步划分为系统软件和应用软件两大类，如图 1.2 所示。其中系统软件

被抽取出来专用于实现那些通用性的功能（数据库系统就是最常见的系统软件之一），为工作在其上的应用软件提供服务。从这个角度来说，系统软件有点类似于操作系统，只是它并非直接作用于硬件系统，而是有其特定的服务领域，通用性不如操作系统软件。而应用软件则针对更加具体的问题，如在通用数据库系统上开发的某交易所的股票交易系统软件就属于应用软件。事实上系统软件和应用软件的划分并非绝对，有些应用软件也可以抛开系统软件直接工作在操作系统之上。

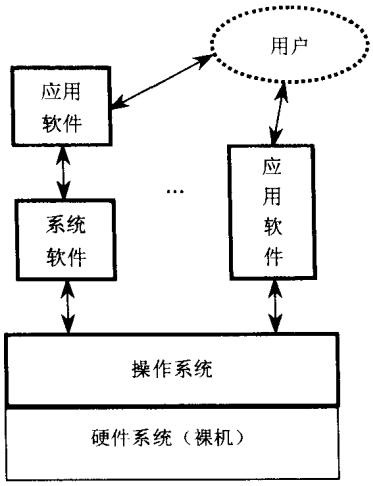


图 1.2　系统软件与应用软件

在图 1.2 中，操作系统、系统软件和应用软件之间的界限整齐划一，但事实上大多数情况并非如此简单。随着计算机技术的发展，操作系统和不同领域的系统软件越来越多，越来越成熟，实现的功能也越来越深入、丰富、强大和可靠，并逐渐渗透到应用软件的内部，使得二者之间的界面变得参差而模糊。所以我们在开发之前，首先要考虑可以利用哪些现成的系统软件或可以利用其中的哪些部分，在此基础之上我们还需要开发哪些模块，在最后形成的应用软件模块结构图中，系统软件功能模块可能出现在任何层次的任何位置上，这就要求我们对操作系统或系统软件向上提供的接口有充分的了解，这是我们从事软件开发的基础。

2. 计算结构的发展

所谓计算结构是指应用系统的系统体系结构，简单地说就是系统的层次、模块结构。前面我们介绍的概念可以理解为单机上的计算结构模式，即操作系统、系统软件和应用软件都保存并运行在同一台计算机上，但实际的系统可能要复杂得多。实际系统可能由多台计算机组成，每台机器有着不同的位置、扮演不同的角色，安装运行不同的软件，在系统中起到不同的作用。计算结构就是要描述清楚这种情况，它不但与软件有关，而且与硬件也有关系。按其发展顺序可以划分为以下四种：

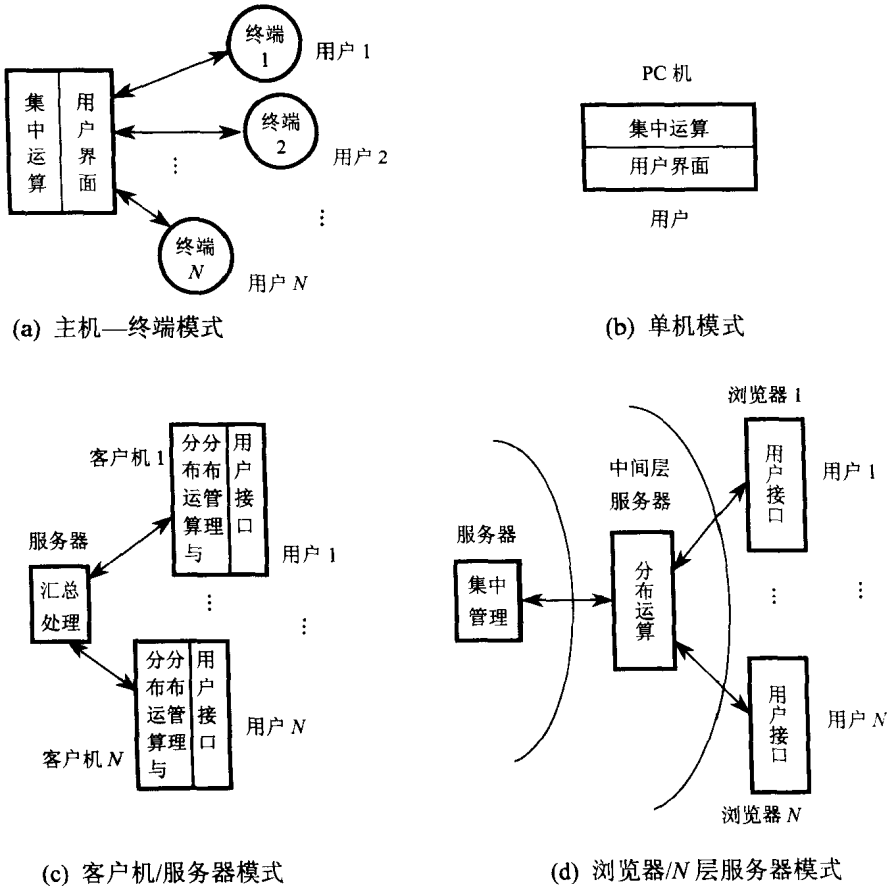
(1) 主机—终端模式

主机—终端模式在 20 世纪七八十年代较为盛行，它由一台功能强大的主机和若干台终端组成，所有的软件都保存在主机中，由主机独自完成所有的运算和处理任务。终端则负责与用户的交互和数据的传送，即用户通过终端提交任务，终端将该任务传送给另一个房间或者更远处的主机，主机运算或处理完毕之后，再将结果传回给终端，呈现在用户面前，如图 1.3(a) 所示。由于终端不具备运算能力，所以也被称为“哑终端”。

集中运算和集中管理是主机—终端模式的特点。其优点是系统管理简单、方便，只需管好一台主机便可；缺点是主机的负担太重，能承担的终端数目有限，而且终端由于没有运算能力，呈现给用户的只能是简单的字符界面，用户需要记背大量命令，界面不够友好。

(2) 单机模式

单机模式在 20 世纪 80 年代兴起，由于计算机硬件技术的发展，具有主机那种运算能力的计算机从体积到价格都可以被广大的机构甚至个人所接受，从而迅速发展为广受欢迎的个人计算机（PC）。单机模式下所有的程序、软件都保存并运行在 PC 机上，各 PC 机之间没有通信联系，成为孤立的个体，如图 1.3(b)所示。这种模式管理、运作起来都较为方便，但是单机处理有一定的局限性，当应用规模扩大时，就显得力不从心了。



(a) 主机—终端模式　　(b) 单机模式

(c) 客户机/服务器模式　　(d) 浏览器/*N* 层服务器模式

图 1.3　四种计算结构模式

(3) 客户机/服务器模式

客户机/服务器模式简称为 C/S 模式，是在 20 世纪 90 年代出现并迅速占据主导地位的一种计算结构，它实际上就是把主机—终端模式中原来全部集中在主机部分的任务一分为二，保留在主机上的部分负责集中处理和汇总运算，称为服务器；而下放到终端的部分负责为用户提供友好的交互界面，称为客户机，如图 1.3(c)所示。相对于以前的模式，C/S 模式最大的改进是不再把所有的软件一股脑都装进一台机器，而是把应用系统分成不同角色、不同地位的两个部分，一般在运算能力较强的机器上安装服务器程序，而在一般的 PC 机上安装客户

机程序。正是 PC 机的出现使客户机/服务器模式的实现成为可能，因为 PC 机具有一定的运算能力，用它替代哑终端后，就可以把主机端的一部分工作放在客户机端完成，从而减轻了主机的负担，也增加了系统对用户的响应速度和响应能力。

分布运算和分布管理是客户机/服务器模式的特点。其优点是客户机端能够提供丰富友好的图形界面，缺点是分布管理较为烦琐。由于每台客户机上都要安装软件，当需要软件升级或维护的时候，相当于把工作量放大若干倍，而且作为独立计算机的客户机，很容易染上病毒，更加大了管理工作的难度。

(4) 浏览器/*N* 层服务器模式

浏览器/*N* 层服务器模式是 20 世纪 90 年代后期随着 Internet 的兴起而出现的新型计算结构，浏览器/服务器/服务器模式简称 B/S/S 模式，是浏览器/*N* 层服务器模式中最简单的一种。相对于 C/S 模式，它主要进行了如下的改进：把客户机程序的功能进一步一分为二，其中负责用户界面的一部分搬到 Internet 上，利用 Internet 上通用浏览器的通用功能来实现，而把剩余的另一部分放在独立的中间层服务器上，成为浏览器/中间层服务器/最高层服务器的三层体系结构，如图 1.3(d)所示。

分布运算和集中管理是这种模式的主要特点，它的优点在于采用了通用浏览器提供统一的用户界面，使得任何会使用浏览器的用户都可以方便地使用该模式下的系统；而且由于这种系统利用世界上最大的网络 Internet 为其组成框架的一部分，故而在任何 Internet 可及的地方都可以获取该系统的服务，从而极大地拓展了其外延和用户群体，而不必像 C/S 模式那样还要专门建设价格高昂、范围有限的局域网。同时系统的维护和升级等工作集中在数目较少的中间层服务器和最高层服务器上进行，大大降低了管理费用。关于 Internet 和浏览器我们这里就不再赘述，仅仅打一个不太恰当的比喻：如果说操作系统是软件与硬件的接口处一组用来完成公共功能的通用程序，那么浏览器就是软件与 Internet 用户的接口处一组用来完成公共功能的通用程序。

本节我们了解了操作系统、系统软件、应用软件的划分和计算结构的演进，这些知识对于我们的软件开发是非常重要的。如果我们要设计、研制一个较大规模的软件系统，在明确了用户需求之后，首先需要为该系统选择一种合适的计算结构并加以详细设计。如在 C/S 模式中，需要划分客户机和服务器两种不同角色的不同功能和任务，分别为它们选择操作系统和必要的系统软件，然后再做针对性的开发工作。即使仅仅做局部的开发，也需要先了解所处的环境和支撑系统，这是对软件开发人员的基本要求。

1.1.3 软件开发过程与程序设计语言

1. 软件开发过程

由于软件系统被划分成了操作系统、系统软件和应用软件，从事软件开发的人员也就自然进行了分工，有的专门研究如何把操作系统设计得更加严谨、高效、安全和方便，是操作系统开发人员；有的则负责研制在操作系统之上具有一定通用用途的系统软件；还有的将根据本单位、本部门或特定用户的具体需要来设计、开发专用的应用软件。由于操作系统软件是硬件裸机和其他软件或用户之间的必由接口，它的性能将决定整个计算机系统的性能，所以开发要求很高，需要具有精深的专业知识与技能，从业人员也最少，对操作系统的研究不

在本书的范围之内。系统软件是操作系统和应用软件之间的接口，从事系统软件开发一方面需要对操作系统有足够、深入的了解，以便能充分利用操作系统提供的服务；另一方面系统软件自身也需要为其上的应用软件提供方便、充分的服务，使得应用软件可以不必了解操作系统的细节知识而满足于使用系统软件的功能。从事系统软件开发的人员也较少。应用软件由于是针对某个具体问题或某个具体实体，所以功能的专用性最强，软件间的差异性最大，开发的需求量最大，从业人员的人数也最多。开发操作系统或系统软件多注重于软件的性能、效率，而开发应用软件最重要的一点是要充分研究用户的需求，即充分研究应用软件的最终用户和操作者希望这个软件具有何种功能，能解决何种问题，在明确了需求的基础之上，再去寻找一个能满足这个需求的解决方案：是直接将系统建筑在操作系统之上，还是寻找一个合适的系统软件作为基础，选择何种计算结构等。无论何种情况，开发人员都需要对即将研制的应用软件所立足的基础层次有足够的了解，并掌握这个层次的相应开发工具。

由此看出，任何软件开发都具有一定的层次和位置，在开发之前首先应明确自己的位置所在，明确自己的开发工作以谁为基础，从谁那里获得哪些支持和服务，同时又为谁而负责。从这个意义上来说，任何软件开发都是站在巨人的肩膀之上进行的，问题在于如何选择一个合适的巨人；一切从底层开始自己动手是非常得不偿失和没有意义的，甚至对于我们所要立足的这个巨人（对于应用软件开发来说，它就是操作系统或系统软件），我们也不需要从头到尾透彻研究，我们只需要充分了解它的那对支撑我们的肩膀就足够了。

在这里，我们反复强调重申的是一种整体概念，只有拥有了清晰的整体概念，在实际的开发过程中才能够做到事半功倍、游刃有余。具体到本书，我们将力图通过对一种具体高级语言程序设计的介绍来帮助读者提高应用软件开发能力。应用软件开发能力一般包括：

(1) 了解应用软件的背景知识

由于应用软件一般都有针对性很强的应用背景，所以开发之前有必要在一定程度上了解这个背景，以便更好地把握开发的重点。事实上，有不少应用开发人员本身就具有较强的专业知识背景，能够帮助他们准确地界定应用开发的范围和功能。

(2) 分析和了解应用软件的功能和需求

在着手进行具体开发工作之前，首先应该花一定的时间分析和了解应用软件的功能和需求，这就是著名的软件工程中的需求分析。需求分析通常是开发人员和用户共同努力、互相沟通、达成一致的过程，需求分析的结果是双方都认可的需求文档。需求分析的具体方法可以参看有关软件工程的书籍。

(3) 确定应用软件的计算结构

根据软件规模、范围和预算很容易确定其合适的计算结构。需要注意的是，如果选择 C/S 或其他分布式计算结构，应该考虑到未来系统的响应速度、吞吐量等硬件参数是否符合用户要求，有条件时最好能在实验室里先进行小规模的测试。

(4) 熟悉应用软件的工作运行环境

任何一个应用软件都是工作运行在某一类或几类具体的环境中的，称为基于某种环境的应用软件，如基于 Windows 98 环境的财务软件，基于 Unix 的电子邮件系统等。这是因为应用软件需要依赖其环境提供的各种服务，当环境发生变化时，应用软件的工作也有可能变得不正常。所以在开发软件之前，应该明确其工作环境，并且熟悉了解这个环境。

(5) 掌握开发应用软件的工具

具体的应用软件开发中，在明确了用户的需求之后，就可以进入设计与实现阶段。设计阶段主要完成应用于实际问题的物理、数学建模和应用软件的系统设计、模块设计、算法设计等工作。实现阶段的主要工作是选择一种或几种计算机高级编程语言编写程序，完成设计阶段提出的任务。

无论是设计阶段或实现阶段，其工作都是非常复杂和专业化的。为了提高软件开发效率，有必要掌握有关工具的使用方法，利用这些工具帮助我们化繁为简，提高软件开发的生产率和质量。设计阶段的工具由于涉及到人工智能技术较多，所以大都局限于一定的领域且价格较高，如数据库设计工具、CASE 工具等，它们的使用范围相对较窄。相比之下，实现阶段的工具要丰富多样得多，如快速应用开发工具（RAD）、各种高级语言编译生成系统等。一般说来，最终实现应用软件要依赖于某种具体语言编写的程序，所以实现工具中应用最广的当属高级语言的集成开发环境（IDE）。如果读者接触过 TC、BC、VC、VB、VJ＋＋等工具产品，对于 IDE 就不会感到陌生，它是集高级语言程序的源代码编辑、目标代码的编译生成、可执行代码的调试和执行于一体的综合性工具，使用它将大大提高开发效率和正确性。

从上述分析中可以看出，完成一个应用软件的开发，首先应该明确最终用户的需求，分析和设计用户逻辑；其次应该了解该软件所运行的系统软件（如操作系统）的有关功能和要求；最后，应该熟练地掌握至少一种高级编程语言和应用软件开发工具。本书的重点将放在这最后一个方面。

2. 程序设计语言及其发展

程序设计语言是能够为计算机和编程人员双方所理解和认可的交流工具。当软件开发人员希望计算机完成一件工作，或解决一个问题时，他（她）首先需要把这个问题的实质彻底研究清楚，确定解决问题的方法和步骤；然后再把这个方法和步骤用计算机能够理解和执行的程序设计语言表述出来，形成一组语句的集合，称为程序。当计算机接收、理解并记住了一个程序的时候，它就具有了这个程序所蕴含的技能，就能够解决这个程序旨在解决的问题，从某种意义上来说，也就拥有了程序员希望它拥有的灵魂。所以一个合格软件开发人员必须了解并熟练运用至少一种程序设计语言。

程序设计语言并不唯一，在计算机技术发展的 50 年中先后形成了不下数百种不同的程序设计语言。按照发展历史，它们大致可分为如下三种类型：

(1) 面向机器的语言

最早的计算机程序被称为是面向机器的程序，因为这些程序与具体硬件的结合非常紧密，通常是针对某一种类型的计算机或其他设备而专门编写的，所以这类程序一般可以充分发掘硬件的潜力、扬长避短，拥有非常高的运行效率，这是面向机器程序的最大优点。

面向机器的程序设计语言主要有各种机器语言和汇编语言，它们在计算机技术发展的早期曾起过很重要的作用。但是这种方法本身也存在着固有的缺陷：一是由于程序是针对机器编写的，与人类的自然语言相差较大，所以面向机器的程序的可读性很差，给后期程序的维护和修改带来很大的困难；二是面向机器的程序面向的是具体某一种型号的计算机或设备，这样使得程序的移植几乎没有可能实现，妨碍和限制了面向机器的程序设计方法在大规模、广泛使用的应用开发中发挥作用。

以上的缺点随着计算机技术的飞速发展和普及越来越成为软件发展的障碍，因此在本世纪六七十年代，一种新的面向过程的程序设计方法被提出和使用。

(2) 面向过程的语言

面向过程的语言把注意力从完成某一任务或功能的机器转移到了问题本身，它致力于用计算机能够理解的逻辑来描述需要解决的问题和解决问题的具体方法、步骤。

面向过程的程序设计的核心是数据结构和算法，其中数据结构用来量化描述需要解决的问题，算法则研究如何用更快捷、高效的方法来组织解决问题的具体过程。例如当我们用面向过程的方法来处理一些数据时，通常都是先考虑如何用计算机可以接受的数据结构来描述这些数据，然后再用一定的算法把这些数据的处理过程一步步地表述出来。

面向过程的程序设计语言主要有BASIC、FORTRAN、Pascal、C等，它们一般与人类的自然语言比较相近，理解起来比机器语言容易得多，从而改善了程序的可读性和可维护性；更重要的，由于面向过程语言着重的是问题的求解过程而不依赖于具体型号的计算机，使得程序的移植、推广成为可能。

正是由于上述优点，面向过程的程序设计方法自提出之后，就得到了广泛的肯定和应用，在软件技术的发展史上有着重要的一席之地，至今也仍然继续发挥着它的作用。但是随着软硬件系统规模的飞速发展，面向过程的程序设计语言也暴露出管理、维护困难，可重用性低等难于克服的弱点。

(3) 面向对象的语言

面向对象的语言，相对于以前的程序设计语言，代表了一种全新的思维模式，这种全新的思维模式能够方便、有效地实现以往方法所不能企及的软件扩展、软件管理和软件复用，使大型软件的高效率、高质量地开发、维护和升级成为可能，从而为软件开发技术拓展了一片新天地。

面向对象的方法早在20世纪60年代就在实验室中提出了，最早的面向对象的软件是1966年开发的 Simula I，它首次提出模拟人类的思维方法，把数据和相关的操作集成在一起的思想。但是受限于当时的硬件条件和方法本身的不成熟，这种技术没有得到推广和使用。随着软件危机的出现和过程化开发方法固有局限性的暴露，人们把目光重新转回到面向对象的方法上来。1980年提出的 Smalltalk-80 语言已经是一种比较成熟、有效的面向对象的工具了，利用 Smalltalk-80 也确实实现了一些面向对象的应用，但是这个语言更重要的作用是提出了一种新的思想观念和解决问题的新方法，它向人们展示了面向对象这个虽然稚嫩，但却充满希望的发展方向。其后，先后产生了多种面向对象的语言，这中间最有影响、对面向对象技术的普及推动最大的当属C++。

C++语言在兼容当时最流行的C语言的基础之上，加入了面向对象的有关内容和规则。由于它的很多语法规则与C语言相近，所以很容易为广大的C程序员所接受；同时C++所具有的面向对象的功能简化了应用软件的开发、设计和维护，为开发大型软件提供了很大的方便。C++的广泛推广和成功应用证明了新兴的面向对象技术的实力和前景，C++也正在逐渐取代C而成为主流编程语言。

Java是20世纪90年代新出现的面向对象的编程语言。相对于C++，Java去除了其中为了兼容C语言而保留的非面向对象的内容，使程序更加严谨、可靠、易懂。尤其是Java所特有的“一次编写、多次使用”的跨平台优点，使得它特别适合在网络应用开发中使用，成为面向对象开发工具中极具潜力的一员。

面向对象的技术正在成为20世纪90年代中后期以来的最重要的程序设计方法，而Java