21世纪大学计算机专业教材

Java语言与面向对象程序设计

Object-Oriented Programming with Java

李建成 郝筱松 编著

邓良松 审

西安交通大学出版社

XI'AN JIAOTONG UNIVERSITY PRESS

21世纪大学计算机专业教材

Java语言

与面向对象程序设计

Object-Oriented Programming with Java

李建成 郝筱松 编著

邓良松 审

西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS
·西安·

内容简介

本书将面向对象程序设计的一般原理和Java语言的基本知识相结合，介绍了Java语言的基本知识及应用程序开发方法。

书中前半部分由浅入深地介绍了Java程序结构及运行过程、Java的语言基础、面向对象程序的开发方法，后半部分介绍了JDK常用类库及其应用包括图形界面、线程、网络、JDBC以及信息系统的开发方法。

本书可以作为高等学校计算机及相关专业面向对象程序设计课程的教材，也适合于各种职业教育学员以及从事软件开发人员自学之用。

图书在版编目(CIP)数据

Java语言与面向对象程序设计/李建成，郝筱松编著.
—西安：西安交通大学出版社，2004.9
(21世纪大学计算机专业教材)
ISBN 7-5605-1866-4

Ⅰ.J… Ⅱ.①李…②郝… Ⅲ.Java语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(2004)第073164号

书　　名	Java语言与面向对象程序设计
编　　著	李建成　郝筱松
责任编辑	屈晓燕　贺峰涛
文字编辑	葛赵青
出版发行	西安交通大学出版社
地　　址	西安市兴庆南路25号(邮编：710049)
电　　话	(029)82668315　82669096(总编办) (029)82668357　82667874(发行部)
电子邮件	eibooks@163.com
印　　刷	西安建筑科技大学印刷厂
版　　次	2004年10月第1版　2004年10月第1次印刷
开　　本	787mm×1092mm　1/16
印　　张	20
印　　数	0 001～3 000
字　　数	480千字
书　　号	ISBN 7-5605-1866-4/TP·383
定　　价	26.00元

前　言

面向对象技术和Java语言是目前软件开发领域中使用非常频繁的词汇。10年前，这两个名词对于一般程序员来说还是比较陌生的，只是技术先驱者的专利，而现在则是非常普及的软件开发技术。目前几乎所有的软件开发领域中都用到了面向对象技术，不懂面向对象技术和面向对象程序设计语言，几乎无法进行正常的软件开发工作，只有极少数的领域还在使用着面向过程的程序设计语言，如C语言。Java语言是一种纯面向对象程序设计语言，由于其简单性、开放性、与平台无关性，得到了绝大多数软件厂商和开发人员的认可，在近10中取得了突飞猛进的发展，其开发人员现在已经超过了300万，在今后两年中会猛增到1 000万，从而使其成为程序设计语言中的后起之秀。

作为大专院校的学生，除了要掌握一门面向过程的高级语言如C语言之外，还有必要熟练掌握至少一门面向对象程序设计语言，这是软件技术发展的大势所趋。目前比较流行的面向对象程序设计语言不下十几种，从语言系统本身的完备性、教学效果、普及和实用程度等方面来衡量，高校中首选的面向对象程序设计教学语言是C++和Java。Java是在C++基础上发展起来的，吸取了C++中的精华部分，抛弃了C++中复杂的、初学者容易出错的部分。由于是一门全新的语言，不需考虑和以前版本的兼容性，所以Java语言系统更加简单紧凑、合理有效。作者曾经教过多届C++和Java课程，并对教学效果进行了比较。实践证明，Java语言更适合于作为一门教学语言，其简单性和纯面向对象性更有利于读者对面向对象基本概念的理解，由于和C/C++的语法相似，也使其更容易被具有C语言基础的读者接受。Java语言的图形界面和网络编程具有相当的实用性，能引起读者浓厚的兴趣。Java语言的教学效率是非常高的，读者可以在短短的几十个学时里，不仅学到面向对象的基本概念，而且可以学习到复杂软件的开发方法和开发过程，充分体现了学以致用的教学方针，这些知识本身既可以在开发实践中应用，也可以作为进一步深入学习的基础。

面向对象的分析问题方法不同于面向过程的分析问题方法，它们之间具有较大的差别。面向对象的分析方法基于类以及对象的交互，面向过程的分析方法则基于过程和调用。不仅分析方法相差很大，程序结构相差也很大：面向对象的程序结构更适合于开发大型和复杂的软件系统，也更适合于对程序进行维护和修改。只有将面向过程程序设计和面向对象程序设计的语言及方法进行比较，才能真正理解面向对象技术。从本质上讲，面向对象程序设计是以面向过程程序设计为基础的，是面向过程程序设计用于解决更复杂问题的一种合理和必然的进步，是面向过程方法更进一步的抽象。如果说面向过程程序设计主要以对现实世界或者问题域中的数据进行加工处理为目标，则面向对象就是对现实世界或者问题域的运动变化过程进行模拟。学习面向对象程序设计语言要比学习面向过程程序设计语言困难，因为面向对象语

言概念更多、更抽象，程序结构也更复杂，结果是其功能更强大，能解决的问题也更多、更复杂。哲学中关于结构和功能关系的一般原理在这里同样适用，那就是：只有更为复杂的结构，才能有更为完善复杂的功能，功能是以结构为基础的。

面向对象程序设计不仅仅从结构上对程序进行改进和优化，而且也对问题分析和程序设计方法进行了改进。现实世界或问题域和求解域或者程序域更加一致，问题域中的对象被映射成求解域中的对象，问题域中对象的交互被映射成求解域中对象的交互，无论是现实世界中的过程还是程序世界的过程都是对象交互的过程。当然在程序域中，只有现实世界或者问题域中的对象还不够，还需要加入软件系统中特有的对象，如界面对象、控制对象等。

如何学好这门语言呢？其实说到底只有两个字：实践，在正确方法指导下的实践。具体地说，就是要以对象和交互为中心，以改进程序结构为目标进行学习。对象的来源有两种：发现和发明。可以使用别人已经创建并满足自己要求的对象，也可以根据需要自己创建对象，自己创建的对象可以来自于现实世界，也可以来自于问题域或者软件系统本身。程序设计语言不同于某些学科如数学的学习，它是一门实践性很强的学科，更像学习游泳和骑自行车，必须在懂得一些基本概念和原理的前提下，通过不断地实践来积累经验，在实践中深入理解并融会贯通这些概念。当然，在学习的过程中还有一些具体的方法和技巧，比如说采用能力持续改进模型或者能力成熟度模型，这是适合于很多学习领域的一种模型，其方法原理是通过多次迭代，最终达到目标。先假定一个比较低的、合适的目标，在达到该目标后，再提出新的目标，再实现，周而复始，每次实现称为一次迭代，经过几轮的迭代，最终实现自己的目标。这种方法区别于一开始就给自己制定一个很高的、不切合实际的目标而无法实现，或者制定一个很低的目标而浅尝辄止。能力成熟度模型就像一个走向最终目标的阶梯，如图 1 所示，而面向对象方法由于其结构上的优势奠定了这种迭代式软件开发的基础，这种方法尤其适合于自学。

下面以使用 Java 语言编写一个网络游戏程序的开发过程为例，来说明成熟度模型的迭代过程。

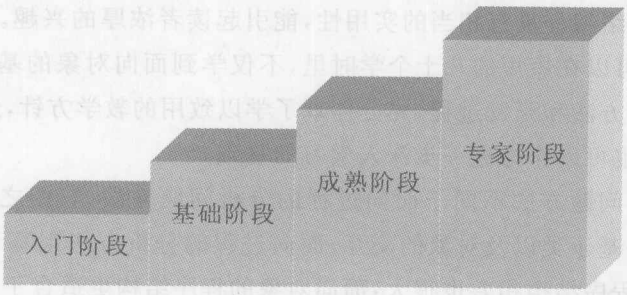


图 1　学习的成熟度模型——阶梯模型

该问题的最终目标是，开发可以在两台联网的计算机上进行联机的游戏程序，例如：下围棋程序。你可能是初学者，一点也不知道应该如何下手去做；你可能已经具有至少一门程序设计语言的基础，但是还没有多少实际开发能力和经验。不管是哪种情况，让我们从最简单的程序开发开始，把最终目标划分成一些小目标。

(1) 首先在Java开发环境中开发并运行一个简单的程序，如在屏幕上显示一串字符，或者是原来用其他语言实现过的程序，如从键盘上输入一个字母，输出该字母的后续字母等。编写这些简单程序的目的主要是使你熟悉在新环境下开发程序的过程，并和以前的程序进行比较，包括程序结构和语法特点等，从而加深对目前开发环境和语言的理解。为了实现这个目标，你需要学习一些相关的知识，如Java程序的基本结构、Java开发环境、Java虚拟机，Java程序的开发过程等。也许你并不能完全理解这些概念，这没有关系，记住：我们是在不断成熟的，不可能一下把所有的问题都解决了，关键是要达到阶段目标，即运行这个简单程序，并得到预期的结果。

(2) 下面你可能要回到正题上。如何开发下围棋程序呢？很显然，需要先在计算机上画出一个棋盘，开始也可以没有棋子。这也可以作为一个目标，为了实现这个目标，你必须要学会使用Java程序画图，这在Java语言中是一件非常简单的事情。棋盘是由多条纵横交错的直线组成的，你可能要用到循环结构。如果该目标实现了，你就学会了在Java中如何使用循环语句和作图。

(3) 下一步目标，你可以给棋盘上添加棋子，这些棋子同样可以画出来，没有什么难度。如果你要实现更为生动美观的棋子，则可能要使用图像或者图片文件，这可以作为后话，现在不要考虑。记住：任何时候，先去实现系统最关键、最核心的功能，至于系统的豪华功能，如果实现代价很大，则可以放到下一版本去做，不要担心将来没有机会去做这件事情，优秀的程序结构对于今后的维护是非常方便的，面向对象技术最大的贡献就是优化了程序结构，使其更容易维护。

(4) 现在是需要考虑交互的时候了，不具备交互能力的程序几乎是无意义的。我们需要根据用户鼠标的指点在某个位置画上棋子，这需要对鼠标点击事件进行编程，其实图形编程从来就是和事件编程紧密相连的，没有理由将其分开。实现该步目标后，你可以随意在棋盘上下棋子了。试一下，是不是觉得挺有成就感？但是这还不能下棋，如果要能够下棋，必须由系统根据用户的操作，做出必要的响应，比如说：在另一个位置下一个另外颜色的棋子。在棋盘上画出该棋子是容易的，但是如何下、下在什么位置，则是非常复杂的，这需要很复杂的算法，好在我们不需要实现和机器下棋，而只是和另一台机器上的人来下棋。这样，剩下的问题就变成了如何把自己机器上的信息发送到另外一台机器上，并接收从另一台机器上传来的信息，这就是网络通讯问题，所以下一步需要针对网络进行编程。

(5) 首先完成两台机器最简单的通信编程，只要能够发送和接收信息即可，这需要使用Java语言的几个有关网络编程的类，等到能够实现两台机器简单的通信后，就可以考虑发送有关棋子的信息。其实这更简单，只需要接收和发送某台机器上用户所下棋子的坐标即可。另外，在通信的时候，还需要考虑多线程问题以及线程的等待和同步问题。到了这一阶段，Java语言的学习也到了后半部分，这时读者已具备了绝大多数的Java语言基础知识和一定的开发能力，而且通过自己的实践，发现并解决了许多问题，积累了很多经验，开发能力已经到了一个比较高的层次了。

上面给出的开发过程没有涉及到具体的语言细节，只是提供了方法指导。成熟度模型是建立在一个不断改进的过程上的，你不能期望一下理解全部内容，而是需要逐步地理解并应用

于自己的问题中。实践证明，这种方法对初学者是很有效的，可以使用这种方法开发各种软件系统，也可以应用于一个团队进行合作开发。

不仅仅在软件开发和学习过程中有不同的成熟度，在任何一门知识的学习过程中，也具有不同的成熟度，因为人认识世界是不断成熟而不是一步到位的。比如说在程序设计语言的学习过程中，最开始我们可能把书上最简单的程序输入进去，调试并运行出来，但不一定理解或者完全理解，这是到了第一个层次。随着对知识的学习，我们可能对书上的程序做一些修改，以达到自己的目的，我们也可能自己编写出一些简单的程序，这又是一个层次。即使到了这个阶段，我们可能还是不能完全理解别人的程序，尤其是比较专业的程序，比如说由开发环境提供的示例程序、向导自动产生的程序框架等，这时候就需要认真阅读和理解，等到你将这种专业的程序理解了百分之六七十以上，并能够通过修改这种程序达到自己的目的，这又达到了一个能力层次。最后，如果你能够结合自己的知识和开发经验，独立设计出自己的程序结构和实现方法，并能够满足各方面的要求，如可维护性、可重用性等，那么你就到了最成熟的层次，以后的工作就是一种完全良性的循环了：学习、开发、积累，周而复始，螺旋式上升。读 Java 参考书同样也是有层次的，比如说，你一开始肯定读不懂那种大部头的国外原版书或者翻译书籍，这时候你可以选择一本简单的国内作者写的 Java 教材，作为入门层次，等对 Java 语言有了一定的理解后，就可以去读国外作者写的原版书或者翻译书籍，这是一个新的层次，最终的层次是读 SDK 的开发文档，这才是最原始、最权威的，其他的所有参考书都是作者对该开发文档的理解和应用举例而已。

本书强调面向对象程序设计语言的基本概念和基本方法的学习，并注重实用性，因为只有通过使用才能真正地理解并掌握面向对象的概念和方法。所以，本书在前 4 章讲述了面向对象的基本概念之后，就开始强调应用，通过各种应用程序开发的讲解和实践来巩固前面学习的基本概念。为了更容易地从 C 语言过渡到 Java 语言，我们一开始没有引入对象的概念，而是在 Java 基本程序结构的基础上进行结构化程序设计(第 2 章)，除了一些输入输出语句不一样外，这几乎和 C 语言是一样的。在读者对 Java 语言的语法特点、程序结构、开发和运行过程有了一定的了解，并能熟练使用 Java 语言实现过去 C 语言实现的基本功能后，开始讲授面向对象的核心概念——类和对象。这样的过渡，实践证明是有效的，比直接上来就讲授类和对象要容易一些，尤其适合于初学者。在比较系统地讲授了类、对象实例、属性、方法、继承、多态、接口、异常的概念后，我们开始比较系统地介绍 JDK 类库，并在应用中掌握如何查找和使用 JDK 的类，为今后进行复杂程序设计和开发打下基础。图形界面编程是当前软件开发中的一个非常重要的内容，在本书中，图形界面编程是一个重点内容，因为它不仅包含了许多图形界面类的使用方法和技巧，同时也包含了 GUI 编程的一般原理，如界面组件、容器、事件等，这些知识完全可以应用于其他语言的 GUI 编程中。在学习了基本的图形界面编程之后，我们又介绍了一些重要的扩展知识内容，如：流、线程、网络编程。这三者的内容是相辅相成的：流是 JDK 类库中非常有特色的部分，通过它可以很方便地对标准输入输出设备、文件、网络进行访问；线程则是提高 CPU 利用效率的现代软件技术；实现网络环境下的分布式程序设计，也是现代软件中非常重要的内容。在这部分的最后，实现了基于局域网的聊天室程序，这一部分综合使用了流、线程和网络的知识。最后一章针对一个简单的信息系统的开发实例，对信息系统的面向对

象开发过程和方法以及 JDBC 进行了介绍。

由于要在有限的篇幅中容纳更多的知识，如 JDK 类库、GUI 编程、流、线程、网络、JDBC、信息系统开发等，对于和 C 语言类似的 Java 语言基础和控制结构部分讲得就比较简略，这对已经有一门面向过程语言基础的读者来说，理解起来是容易的。所以，本书内容比较适合于那些已经有一门以上(尤其是 C 语言)语言基础的读者。当然，如果没有这种基础，也可以借助参考书来达到目的，因为毕竟一本书不可能把什么都包含殆尽。

本书第 1,4,6,7,8,9,10 章由李建成编写，第 2,3,5 章由郝筱松编写，郝筱松还为第 1,5 章编写了例题。

需要指出的是，由于时间紧迫，书中错误在所难免，希望读者提出批评和建议，作者在此不胜感激。

作 者

2004 年月 5 月

目　录

第 1 章　绪论

☞ 本章要点

- Java 语言特点
- Java 程序结构
- Java 程序的开发和运行过程
- Java 虚拟机(JVM)

本章从应用范围和程序结构角度介绍了 3 代程序设计语言的区别。通过介绍和分析 3 个简单的 Java 程序例子，说明了 3 种基本 Java 程序的结构以及 Java 程序的开发过程。

1.1　程序工作原理

计算机系统是由硬件(CPU、内存和输入输出设备)和运行于其上的软件组成的。软件即程序，是指挥计算机做事情的指令和数据序列。程序设计语言是书写程序的语言，目前的程序设计语言不下几百种，其功能和结构几乎是等价的，但语法有所区别。

利用 CPU、内存和代码数据进行工作的模型称为程序模型，如图 1－1 所示。

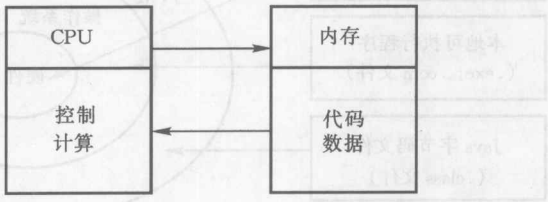


图 1－1　程序模型

程序是用程序设计语言书写的完成某些功能的指令和数据的序列。指令表示计算机对数据进行的处理，代表了机器的思维过程，数据则为计算机处理的对象，代表了机器思维的内容。自从人类第一台计算机产生以来，就出现了程序设计语言和程序，从最早的计算机可以直接理解的机器语言到现在高效的面向对象程序设计语言，程序设计语言经历了 3 个阶段：面向机

器，面向过程，面向对象。面向机器的程序设计语言主要是机器语言和汇编语言，机器语言即机器指令，汇编语言是符号化了的机器语言，这些语言的语句基本上都是和机器指令一一对应的。用面向机器的语言书写的程序和机器“思维”的方式一致，但和人类的语言和思维相去甚远，很难被人理解，因此称为低级语言。面向过程和面向对象语言则兼顾到人类的自然语言和自然思维，通过使用人类熟悉的语言、语句和表达方式来编写程序，这种语言一般被称为高级语言。高级语言不能被机器直接理解和执行，需要翻译程序将其翻译成机器指令，然后才能被执行。用高级语言写的程序一般称为源程序，对于高级语言的翻译有两种方式：解释和编译。前者是一边翻译一边执行，翻译一句，执行一句，此处的翻译程序称为解释程序；后者则将源程序一次性翻译好，然后一次性执行，这里的翻译程序称为编译程序，翻译出的程序称为目标程序，一般都是机器可以直接执行的代码，即面向机器的代码。这种目标程序的移植性比较差，换一种硬件类型的机器就不能运行了。解决程序移植性有两种方法：一是到另一种类型的机器上将高级语言源程序重新编译成适合于该台机器的机器代码，此时的高级语言程序相当于逻辑程序模型，而编译出的目标程序相当于物理模型，逻辑模型可以适合于任何机器，即与机器无关；另外一种方法是将源程序编译成一种与机器无关的中间代码（对 Java 程序来说即字节码），该中间代码程序不能被操作系统直接执行，需要由解释程序来解释和执行，这种方法实际上是编译和解释的结合，称为伪编译，在每台机器上安装解释该中间代码的解释程序，程序每次运行时都由该解释程序解释执行，这种方式相当于用解释程序扩展了该种机器的指令系统，被扩展了指令的机器就可以直接执行以中间代码形式存在的程序。Java 语言采用了后面的方式，而由解释程序扩展了指令系统的机器称为 Java 虚拟机，简称为 JVM（Java Virtual Machine）。

正像操作系统扩展了机器硬件的功能，使计算机更容易使用一样，JVM 同样扩展了计算机的功能，使得计算机可以认识以字节码形式存在的程序，这种程序既可以是本地的，也可以是通过网页从网络上下载的。JVM 就相当于一个字节码程序解释器，不仅如此，JVM 还担负有对网络上下载的程序进行代码校验、安全检查等职责。

计算机硬件、操作系统、JVM 以及各种可执行程序之间的关系如图 1-2 所示。

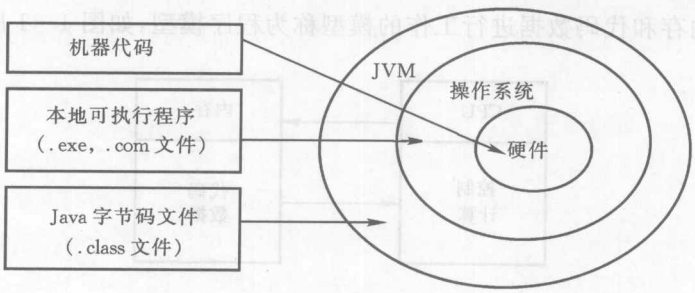


图 1-2 JVM 工作原理

1.2 程序设计语言的分类

正像自然语言是人类交流的工具一样，程序设计语言是人类和计算机之间通信的工具，人

类可以把自己的想法和要求通过程序设计语言描述的程序通知计算机，计算机则将处理结果以某种形式显示给人类。从最早的机器语言发展到现在，程序设计语言不下几百种，但总的可以分为两大类：面向机器的语言；与机器无关的语言。前者一般称为低级语言，后者称为高级语言。从目前情况来看，高级语言又可以分为两大类：面向过程的语言和面向对象的语言。下面分别对这几种语言从程序结构、开发过程、适用领域等方面做一简单介绍。

1.2.1　面向机器的语言

众所周知，计算机内部的工作语言是二进制代码，即二进制数，使用这种语言编制的程序即为机器程序，这种以二进制代码为语法形式的语言称为机器语言。程序的内容包括指令和数据。在机器语言中，指令和数据都使用二进制数来表示，如何区分和执行是计算机的工作，编写程序却是程序员的工作，由于这些二进制代码表示的指令和数据非常难懂，给程序编制和调试带来了很大的困难，因此编制程序的效率很低。后来人们发明了使用符号来代替机器指令的方法，一般把这种符号化的机器语言称为汇编语言，用该语言书写的程序称为汇编源程序，而将汇编源程序翻译成机器指令的程序称为汇编程序。这样的符号程序虽然要比直接使用二进制代码表示的程序好读得多，但是由于符号指令仍然是和机器指令一一对应的，而机器指令都是很基本的机器操作，如加、减、乘、除、移位等，所以采用汇编语言编写程序的过程和人类的自然思维过程仍然相差很大，并且需要翻译成机器代码，因此开发效率还是很低，不适合去解决各种复杂的现实世界中的问题，但由于和机器代码一一对应，其运行效率是最高的，目前主要用于计算机控制领域中。

从源程序结构上来看，汇编源程序的数据和代码基本上没有分离，没有很好的预定义的程序结构。控制结构主要是一些顺序、转向和过程结构，但没有局部变量。

从开发方法和适用问题领域上来看，汇编源程序主要是通过过程分析的方法，来解决一些简单的数据处理和控制问题。

1.2.2　面向过程的语言

面向过程有两种含义，一是需要通过描述处理问题的详细过程来达到解决问题的目的，从这一点来看，上面讲的面向机器语言和下节将要介绍的面向对象语言都是面向过程的，因为都需要描述解决问题的详细处理过程。只有面向问题或者面向目标的语言才是不需要描述过程的，只需要给出要求解的问题或者目标即可（如代数推导语言）。实际上任何语言都需要描述问题的处理过程，只是描述的程度有所不同而已。面向过程的另外一层含义是指源程序的结构是由一些过程（也叫子程序或者函数等）组成的。在本书中，面向过程指的就是这种含义。如 C 语言的源程序至少应该由一个 main 函数组成，也可以定义多个函数，由主函数或者其他函数调用。面向过程的程序结构如图 1-3 和图 1-4 所示。图 1-3 中给出了 Pascal 语言和 C 语言程序的静态组成结构，图 1-4 中给出了 C 语言中函数的调用结构。面向过程的语言根据其控制结构的特性，又可以分为非结构化的程序设计语言（如早期的 Basic 语言、Fortran 语言）和结构化的程序设计语言（如 Pascal 语言、C 语言）。非结构化的程序设计语言属于第一代高级语言，这种语言提供了一些简单的控制结构，如转向结构（goto 语句）、子程序结构等。尤其是转向结构的应用给程序带来了很大的灵活性，但有可能使程序的控制结构变得非常复杂，以致于难以理解和维护。后来出现了以 Pascal 语言、C 语言为代表的结构化程序设计语

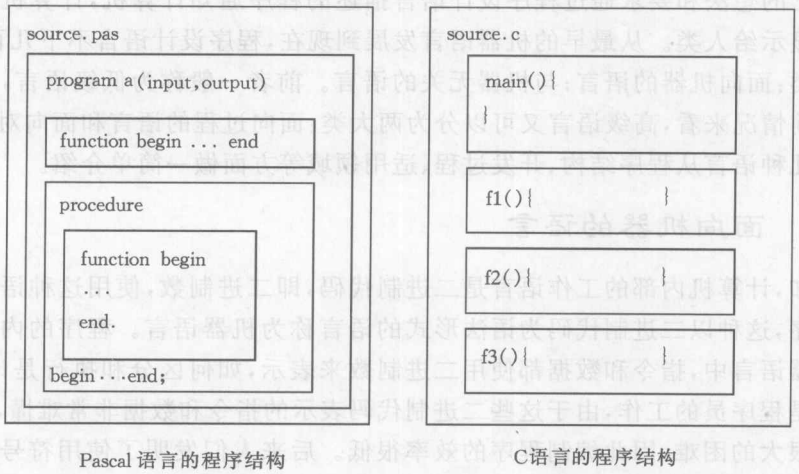


图 1-3 面向过程的程序结构

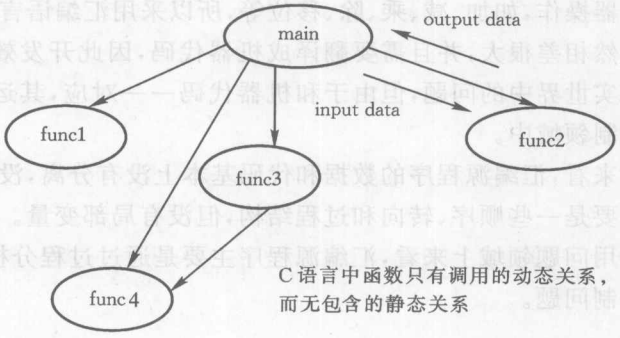


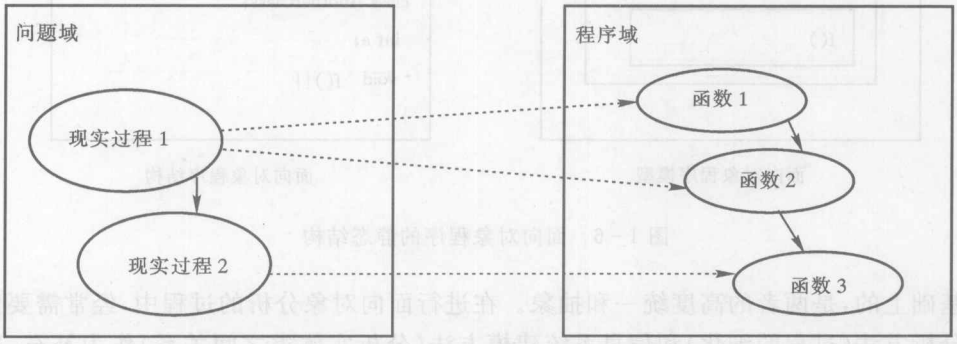
图 1-4 C 语言的动态调用结构

言，这种语言可以称为第二代高级语言，在这种语言中严格限制转向语句的使用，规定了 3 种基本程序控制结构——顺序结构、选择结构、循环结构，每种结构都必须是单入口和单出口，这种程序很好地规范了程序的控制结构，程序的可读性和可维护性都得到了较大的提高，也减少了发生错误的机会。这是从程序结构的角度来对程序设计语言的功能进行改进，从而可以解决更复杂的问题。

面向过程程序的静态程序结构主要以自定义数据结构、过程或者函数结构为主，这两种结构都是预定义结构。数据结构定义是静态的，是数据的模板，必须定义变量后才得以实施。函数也是一种控制结构，包括静态和动态两方面的含义，函数定义是静态的，而函数调用是动态的，如果程序中没有函数调用语句，它是不会被自动执行的，这一点和其他控制语句不一样。Pascal 语言是一种严格的结构化程序设计语言，允许过程或者函数嵌套，允许定义局部数据结构等，不提供破坏结构化的控制语句如 return 或 break 等；C 语言是注重实用效率的准结构化程序，它的函数结构都是并列的，即不允许函数嵌套，有限度地定义了一些旨在提高开发和实现效率但破坏结构化的语句如 return 和 break。每种结构化程序设计语言都允许定义局部变

量。

面向过程程序设计中，分析问题的主要方法是以实际问题或问题域中的处理过程为中心，同时考虑适用于实际问题的数据结构和程序的输入输出，程序是由多个具有输入输出的处理过程组成的。面向过程语言一般提供多种基本数据类型和自定义结构数据类型，可以自定义复杂的数据结构。总的说来，面向过程的程序设计无论是程序结构还是分析方法，都是以处理过程为主、数据结构为辅的。程序的分析和处理模式是：输入—处理—输出。面向过程的程序设计支持功能分解和细化的层次分析方法，如自顶向下、逐步求精的方法。问题域中的处理过程和程序域中的过程之间的对应关系如图 1－5 所示。



问题域中的过程可能对应程序域中的多个过程或者函数

图 1－5　面向过程程序设计中问题域和程序域之间的对应关系

1.2.3　面向对象的语言

面向对象程序设计语言可称为第三代高级语言，和面向过程的语言有较大的区别。除了保留原来的 3 种基本控制结构和函数结构外，新增加了一种“类”结构。“类”结构是一种更为高级的程序结构，它综合了数据结构和函数结构，不仅可以包含复杂的数据属性和数据结构，还包含着对这些数据的操作。“类”结构是数据和操作高度统一的整体，具有更高的内聚性、安全性、稳定性和可重用性。不仅如此，类与类之间还可以建立更为深刻的继承和包含关系，使得程序结构更加复杂、更加合理，同时也更加稳定。从哲学的观点来看，只有层次丰富完整的复杂物质结构，才能完成复杂完善的功能。事实正是如此，利用面向对象技术和面向对象程序设计语言，人们实现了比原来面向过程更为复杂完善的程序，也实现了更多更可靠、更易用、功能更强大、结构更合理的类库，极大地提高了开发效率。以 Turbo C 为例，该开发环境只提供了一百多个函数，而 JDK(Java Develop Kit)则提供了近万个类，每个类都有几个到上百个方法。面向对象程序的结构如图 1－6 所示。

“类”的引入除了可以改进程序结构之外，还可以和现实世界或者问题域中的概念或者实体对应起来，是现实世界中实体属性和行为的抽象。这为解决实际问题提供了一种更为直观的分析方法，即以类或者对象为中心的分析方法。在此之前还有一种以数据为中心的分析问题的方法，即信息系统建模方法，该方法以现实世界和问题域中的概念或者实体及其数据属性、实体之间关系为中心来分析和建模，该方法主要用于信息系统的数据建模。面向对象方法则以数据和对数据的操作为中心来分析实际问题，是建立在面向过程分析方法和信息系统建