Java 经典教材译丛

Java 程序设计基础

著者：[美] Gary J.Bronson

译者：赵德奎 林仕平

THOMSON

北京大学出版社

http://cbs.pku.edu.cn

Java 经典教材译丛

Java 程序设计基础

A First Book of Java

[美] Gary J. Bronson 著
赵德奎 林仕平 译

北京大学出版社
·北 京·

内 容 简 介

本书主要针对初学 Java 的读者，主要介绍了面向对象程序代码的基本概念；创建图形用户接口所要求的可视化对象；基于事件的编程等内容。本书的主要目标是：在合理的编程规则下以一种能够被初级程序员接受的方式介绍上述内容，以期提供给读者工具、技术以及创建并保持程序的必要理解，给更多的工作准备一个坚实的基础。

本书提供了非常清楚的定义，帮助读者获得和掌握 Java 知识，介绍概念的书写方式是本书最重要、最有特色的部分。本书的大量实例和习题都是笔者经验的结晶，这些例子非常适合对语言的基本介绍。

本书适合 Java 初学者做自学用书，也非常适合做大学 Java 程序设计的教材。

981-254-147-0

北京市版权局著作权合同登记号　图字：01-2003-8805 号

图书在版编目(CIP)数据

Java 程序设计基础/(美)布朗森（Bronson,G.j.）著；赵德奎，林仕平译. —北京：北京大学出版社，2005.1
(Java 经典教材译丛)
ISBN 7-301-08387-4
Ⅰ.J… Ⅱ. ①布… ②赵… ③林… Ⅲ. JAVA 语言—程序设计—教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2004)第 128027 号

书　　名：Java 程序设计基础
著作责任者：[美]Gary J. Bronson 著
译　　者：赵德奎　林仕平
责任编辑：黄庆生　汉　明
标准书号：ISBN 7-301-08387-4/TP·0772
出 版 者：北京大学出版社
地　　址：北京市海淀区成府路 205 号　100871
网　　址：http://cbs.pku.edu.cn
电　　话：邮购部 62752015　发行部 62750672　编辑部 62765013
电子信箱：xxjs@pup.pku.edu.cn
印 刷 者：河北滦县鑫华书刊印刷厂
发 行 者：北京大学出版社
经 销 者：新华书店
787 毫米×1092 毫米　16 开本　26.25 印张　752 千字
2005 年 1 月第 1 版　2005 年 1 月第 1 次印刷
定　　价：38.00 元

编 者 序

欢迎学习 Java 程序设计语言！

自计算机发明以来，无处不在显示着它对整个人类文明和社会进步产生着如此巨大、如此深刻的影响，推动着人类社会一日千里地向前发展

历经半个多世纪，计算机软件——计算机的重要组成部分，也发生了日新月异的变化。从机器语言到汇编语言再到高级语言，从结构化语言到面向对象语言，人们不断地探索和发明新的、功能强大又简单易学的计算机语言。

用计算机语言编程，有如我们用世界上的任何一种语言写作一样，两者都是创造性的工作，而且都要用到相应的技术。如果您掌握了一种语言，熟知它的词汇和语法规则，也许您能熟练地运用这种语言进行表达，但这并不能使您成为一位作家。写作是一种创造性的工作，它需要有创造性的人来做。用计算机语言编程也是同样的道理。也许您掌握了当今最流行的程序设计语言，而且也能迅速地写出程序语句，但是，程序的内容只能靠您的创造性来产生。

1. 为什么要学习 Java 程序设计语言？

工欲善其事，必先利其器。没有得心应手的工具又怎么能随心所欲地发挥创意？计算机高级语言众多，而 Java 语言的得天独厚的优点，使它从众多语言中脱颖而出，越来越受到人们的青睐。Java 是一种跨平台、适合于分布式计算环境的面向对象编程语言。具体来说，它具有如下特性：简单、面向对象、分布式、易解释、可靠、安全、与平台无关、体系结构中立、可移植、高性能、多线程、动态等。

鉴于此，我们从国外遴选了 7 本流行的 Java 书，组成了《Java 经典教材译丛》，以期读者能从中受益。

2. 您渴望从《Java 经典教材译丛》学到什么？

博大精深是我们选此套丛书的宗旨，从 Java 基本概念和技术到其高级主题，丛书的内容涵盖了 Java 的全部知识。原书多由国外教学经验丰富的教师编写，书中提供大量的实例和案例。为了给读者提供良好的服务，我们将大部分书中的实例中的源代码放在 http://cbs.pku.edu.cn 中的【下载专区】上，读者可从中下载。

3. 什么人使用《Java 经典教材译丛》？

无论您是一位计算机语言初学者，还是一位熟知计算机语言的编程人员，《Java 经典教材译丛》中必有一本适合您。当然，本套丛书主要针对在校大专院校的学生而编写，所以大多数读者都可以轻松上手，教师更可以从中找到教学用书和参考读物，部分书中带有星号(*)的章节则是为更高层次的读者而设计，为选学内容。

4. 对计算机的硬件和软件有什么需求？

- 软件　可以通过访问 Sun 的网站(http://java.sun.com)下载 JDK 编译程序，下载的软件大约为 20MB。
- 硬件　必须具备可以运行 Windows 操作系统的计算机或者 Windows NT 工作站，100MB 的空闲磁盘空间，最少 32MB 的内存(如果使用 Windows NT，建议准备 64MB 的内存)。

丛书编委会

2004 年 10 月

前　言

Sun Java 正在迅速地发展成为基于微软视窗系统的一门卓越的编程语言，其从小程序到应用程序语言的转变主要有两个原因：首先，大部分小程序已经能够被包含到网页中的脚本语言所代替；其次，Java 2.0 版本以及更高的版本提供了一系列完整的可视化对象，例如按钮、标签、文本框、单选钮和复选框等，它们可以很快地集合成一个图形用户接口并结合到一个多任务操作系统环境中。

1．本书目标

不管是从教还是学的方面，都要求读者对 Java 的如下三个部分非常熟悉（这三部分对于面向对象、基于图形的程序都是必须的）分别是：

（1）面向对象程序代码的基本概念。

（2）创建图形用户接口所要求的可视化对象。

（3）基于事件的编程。由用户，特别是由程序员决定操作的执行顺序。

本书的主要目标是：在合理的编程规则下以一种能够被初级程序员接受的方式介绍这三部分，以期提供给读者工具、技术以及创建并保持程序的必要理解，给更多的工作准备一个坚实的基础。

本书的大量例子和练习都是笔者经验的结晶，这些例子适合对语言的基本介绍。

2．写作风格

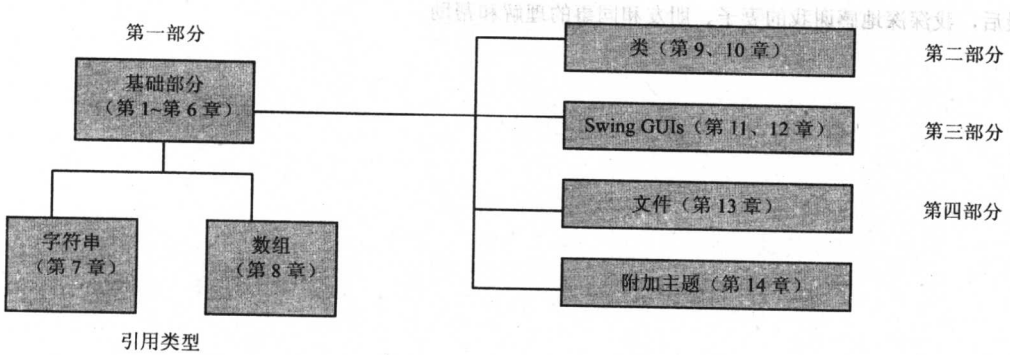
本书提供了非常清楚的定义，帮助读者获得和掌握 Java 知识。介绍概念的书写方式是本书最重要、最有特色的部分。

3．灵活性

本书分为四个部分，每部分中又包含若干章。

第一部分介绍了面向对象结构的基础知识和 Java 的程序元素；还介绍了键盘和基于对话框的数据输入，包含对可视化对象的 Swing 包的介绍，并为基本的 Java 开发工具集（JDK）技术的学习提供了坚实的基础。

第一部分结束后，第二、三、四部分的知识是可互换的，例如，如果按照编程的传统介绍归类，第 13 章应该包括在第一部分；然而，如果强调类的设计和开发，第二部分可以跟在第一部分后面；如果教学要求向可视化图形用户接口倾斜，第三部分可以跟在第一部分后面。在每一种选择中，都可以实现一个精选的课程结构，以下是各章标题的一种组合：



4. 软件工程

虽然本书主要是对 Java 知识的介绍，但和普通的编程书不同，本书将从一个过程和面向对象的观点来向读者介绍软件工程的基本知识。在许多例子中，很多编程都交叉在主要的语言组件介绍中。

5. 程序测试

本书中的每个 Java 程序都已经被成功输入并使用 Java 2.0(版本 1.2 和 1.3)执行。www.brookscole.com 上提供了本书包含的所有程序，以方便读者练习并扩展这些程序，并可以进一步地改进它们。

6. 教学功能

为了在第一阶段使 Java 更容易接受，本书包含了下列的教学内容：

- 每节练习。几乎在书中的每一节都包含了许多不同的编程技巧以及编程练习。另外，在 www.brookscole.com 上也提供了一些练习的答案。
- 常见编程错误和章回顾。除了第 14 章，每章都提供了常见的编程错误，同时每章的最后一节都对本章所介绍的主要内容进行了回顾。
- 章附录。在附录中列出了可以应用在 Java 教学中的许多不同内容，包含了许多基本主题。这些附录和把理解位和字节作为基本教学、把格式化和产生随机数作为实际教学、把内部和外部类作为理论教学有所不同，它们的目的是在选择那些主题介绍和在什么时候介绍它们之间提供了灵活性。
- 知识点。本书的阴影部分主要是对常用任务、快速引用、高专业编程技术提供了一个延伸，并提供了额外的概念知识。

7. 附录和补充

本书提供了一个扩展的附录。这包括了运算符优先级、Unicode 码、包含本地信息的包、格式化、一个键盘输入类和浮点小数存储。

8. 致谢

在本书的写作过程中，Brook/Cole 的许多人员给了我很大的帮助。首先，我要感谢 Kallie Swanson——Brook/Cole 的编辑，她对本书编写工作给予了巨大的支持和鼓励。

我也非常感谢 Carla Vera，他为我处理了很多计划和细节，使我能够专注于本书的实际写作中。

本书由赵德奎、林仕平主译，参加翻译和校对的还有杨章玉、范谦、李欣、胡巍、张红霞、孙闽、李军玲、侯军波、丛聪、栾琳等。

最后，我深深地感谢我的妻子、朋友和同事的理解和帮助。

目　录

第一部分

创建程序

第1章 基础知识

本章将介绍 Java 编程语言的主要背景以及在本书中构造 Java 程序所要使用的具体结构；本章还将介绍方法和类的概念，介绍 Java 中的两个具体方法：print()和 println()，在完整的程序环境中，这两个方法被用于在视频屏幕上显示数据；最后，介绍方法 showMessageDialog()，在完整的程序环境中，该方法被用来构造一个简单的图形用户界面（GUI）。

1.1 编程简介

计算机就像是汽车或割草机一样，是一种机器，首先必须将其启动，然后再驾驶或控制它来完成要做的工作，就好比汽车是由坐在里面并且指挥它的驾驶者来控制的一样，计算机则是由计算机程序来控制的，更准确地说，**计算机程序**是能够操纵计算机的数据和指令的结构化组合，计算机程序的另一个代名词是**软件**，我们将在本书中互换使用这两个术语[①]。

编程是使用计算机能够响应、其他程序员能够理解的语言来编写计算机程序的过程。被用来构造程序的一系列指令、数据和规则被称为**“编程语言”**。

编程语言通常根据级别和面向对象进行分类。

使用类似于书面语言（例如，英语）结构的语言被称为**“高级语言”**，Pascal、Visual BASIC、C、C++和 Java 都是高级语言，使用这种语言编写的程序能够在不同类型的计算机上（例如，IBM、Apple 和 Compaq 等）运行；相反，**低级语言**使用与某一种类型的计算机直接相关的指令[②]，虽然使用低级语言编写的程序只能被限制在针对它们的计算机上运行，但是它们可以使用对应计算机的与众不同的特定功能。

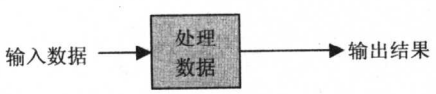


图 1.1 基本程序操作

基本上，几乎所有程序的目标都用来处理数据，以产生一个或多个特定的结果（如图 1.1 所示），直到 20 世纪 90 年代中期，用来执行这些基本任务的高级语言基本上都是面向过程的，**面向过程的语言**是使用一系列被称为“过程”的一组或多组指令构成程序的语言，这些指令分别通过一组输入来生成结果。实际上，面向过程的语言的重点是在如图 1.1 所示的过程上，每个过程都从输入的数据向得到最终所需的结果更进一步。自从 20 世纪 90 年代中期以来，面向对象的编程方式已经发展成为第二种编写应用程序的主要形式。

面向对象的语言发展的动机之一是发展和扩展图形化屏幕的使用，以及对图形用户界面（GUI）的支持，这些图形化屏幕能够显示多个窗口，以提供数据的输入和显示功能，而使用面向过程技术的程序是很难实现这些功能的。在图形环境下，屏幕上的每一个窗口都可以方便地被看作是一个对象，它们具有颜色、位置和大小等相关特性。使用这种对象面向性，程序首先定义它将要操作的对象，包括描述对

① 从本质来说，术语“软件”也被用来表示程序以及程序所要处理的数据。

② 通常，低级语言是针对组成计算机的处理器而定义的。这些处理器包括 IBM 个人计算机的 Intel 微处理器芯片、Apple 计算机的 Motorola 芯片和很多 Compaq 计算机的 Alpha 芯片。

象的常规特性及操纵对象的特定单元。Java 是一种面向对象的语言。

1991 年，Sun 公司的 James Gosling 最早设计了 Java 语言，它是一种能够被嵌入到电子消费产品中以创建智能电子设备的语言，当时，这种语言被称为“Oak”。1993 年，当开发 Oak 的项目面临终止时，Internet 却刚刚开始飞速地增长。Sun 的工程师们意识到，Oak 可以很容易地用于在 Internet 上创建动态的 Web 页面，这些页面将会包括作为整体的可执行的程序，而不只是包含静态的文本和图片，虽然通常从理论上也可以通过其他编程语言做到这些，但由于处理器类型的多种多样，这实际上是不可行的，而 Oak 程序将在所有的 Web 页面上执行相同的操作，而与显示页面的计算机无关，为了证明这种方式的可行性，Sun 创建一个名为“HotJava”的 Web 浏览器，这种浏览器本身就是使用重新命名的、被称为“Java”的 Oak 语言编写的。**Web 浏览器**是一种存在于用户计算机中，并在其中运行以显示 Web 页面的程序。虽然它最初是与 HotJava 浏览器相关的，但很快得到证明，Java 是一种功能强大的、面向对象的通用语言，从这方面来看，Java 的发展与 20 世纪 70 年代后期开发的用来编写 UNIX 操作系统程序的 C 语言是同步的。就像应用程序的开发者以前认为 C 语言可被用作独立于 UNIX 的通用过程语言一样，同样的认识过程很快就出现在 Java 身上。

对于 Java 来说，促进它作为一种通用的应用程序编程语言而受到欢迎和得到广泛接受，主要是因为它能够创建可以在任意计算机上执行、而与计算机所用的中央处理器无关的程序，不必重新编写或重新翻译程序，这种能力被称为“**跨平台的兼容性**”和“**一次编写，随处运行**”。

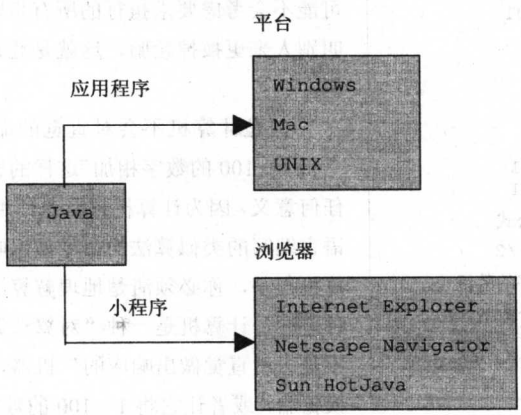


图 1.2　两种不同的 Java 环境

如图 1.2 所示，创建可以在用户计算机（正式名称为“本地计算机”和“Web 客户计算机”）上使用的 Java 程序，主要有两种方法[①]：

（1）作为嵌入到 Web 页面中的小程序，这种程序被称为“applet”。

（2）作为独立的程序，类似于使用像 C 和 C++这样的高级语言编写的程序，这种程序被称为“应用程序”。

虽然 Java 最初主要是用于创建 applet，但现在这种用途已经很小了，因为 applet 的功能可以很容易地使用像 JavaScript 这样的脚本语言来创建。虽然名称相似，但 JavaScript 与 Java 编程语言并没有多大的关系，因此，本书没有深入介绍 JavaScript[②]，而主要介绍的是以 Java 2 SDK（1.2 或 1.3 版本）规范（通

① 第三种也是更奇特的方法是，作为一种在特定 Web 页得到访问时自动在 Web 服务器计算机上运行的程序，这种程序被称为“servlets”。

② JavaScript 是由 NetScape 公司开发的，但只有名称是属于 Sun Microsystems 公司的。

常被称为“Java 2”）定义的 Java，并且将它用作一种功能齐全、通用的编程语言。

1.1.1 算法和方法

在编写任何过程或面向对象的程序之前，程序员必须清楚地理解将要使用的数据、需要的结果，以及将要产生这个结果的过程（所用的过程被称为“算法”）。准确地说，**算法**是一系列逐步执行的指令，它们描述了如何处理数据以产生所需的期望的输出结果。从本质上说，算法针对下面这个问题做出回答：你将使用什么方法来产生特定的结果？

方法 1——列：将数字从 1 到 100 排成一列，然后进行加操作

```
  1
  2
  3
  4
  .
  .
  .
 98
 99
+100
----
5050
```

方法 2——组：将和为 101 的两个数字分为一组，将组数乘以 101

```
1 + 100 = 101
2 +  99= 101
3 +  98= 101
4 +  97= 101
.      .
.      .
.      .
49 +  52= 101
50 +  51= 101
```

方法 3——公式：使用下面的公式

```
sum = n(a+b)/2
```

这里，n 是要进行加操作的数字的数目

a 是进行加操作的第一个数字（1）

b 是进行加操作的最后一个数字（100）

```
sum = 100(1+100)/2 = 5050
```

图 1.3 计算 1～100 的数字总和

为了说明算法，我们来看一个简单的问题。假设一个程序必须计算 1～100 的所有数的总和。图 1.3 列出了可以用来得到所需结果的 3 种方法。每种方法构成了一个算法。

显然，像图 1.3 那样先逐步详细地列出所有可能的选择，然后选择其中一个算法来解决问题并不是一件难事，但是，大多数人都不会考虑算法，而总是凭直觉考虑问题，例如，如果你不得不更换汽车的轮胎，那么你可能不会考虑要求执行的所有步骤；你可能只会自己或叫别人来更换掉轮胎，这就是凭直觉考虑问题的一个示例。

但是计算机不会对直觉的命令做出响应，一个像“将 1～100 的数字相加”这样的普通语句对计算机没有任何意义，因为计算机只能对使用像 Java 这样的可接受语言编写的类似算法的命令做出响应。要成功地编写计算机程序，你必须清楚地理解算法和直觉命令之间的这种差异，计算机是一种“对算法做出响应”的机器，而不是“对直觉做出响应的”机器，你不能让计算机去更换轮胎，或者让它将 1～100 的数字相加起来，相反，你必须向计算机提供详细的、逐步顺序的指令，这些指令合在一起就组成了一种算法，例如，下面的一系列指令

```
Set n equal to 100
Set a = 1
Set b equal to 100
Calculate sum = n×(a+b) / 2
```

组成了一个详细的方法（或算法），来计算 1～100 的数字的总和。请注意，这些指令并不是计算机程序。与程序必须使用一种计算机能够做出响应的语言来编写不同，算法可以采用不同的方式来编写或描述；当使用类似英语短语的词来描述算法（处理步骤）时，这种描述被称为**“伪代码”（Pseudocode）**；当使用数学等式来描述时，这种描述被称为**“公式”（Formula）**。当使用下面列出的符号图来描述时，这种描述被称**“流程图”（Flowchart）**。

- “终止”符号，表示程序的开始和结束。
- “输入/输出”符号，表示输入或输出操作。

“过程”符号，表示计算或数据操纵。

“流程线”符号，用来连接其他流程符号并表示逻辑流程。

“决定”符号，表示程序的分支点。

“循环”符号，表示循环的初始值、限制和步长值。

预先定义的过程，表示预先定义的过程，例如调用函数。

连接点，表示流程图其他部分的入口点或退出点，或者表示连接点。

“报告”符号，表示所写的输出报告。

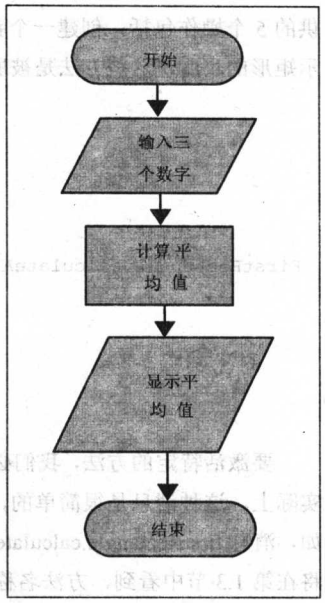


图 1.4　计算平均值的流程图

计算 3 个数字的平均值的流程图如图 1.4 所示。因为流程图很难修改，并且可以很容易地支持非结构化的编程做法，所以除了以可视化方式描述基本的编程结构之外，专业的程序员已经不再习惯使用流程图，他们更愿意接受伪代码。在使用伪代码描述算法时，可以使用简短的英语短语，例如，下面是一个描述计算 3 个数的平均值的步骤的伪代码：

```
Input the three numbers into the computer.
Calculate the average by adding the numbers and dividing the sum by 3.
Display the average.
```

只有在选定了算法，并且程序员理解了所需的步骤之后，才可以使用计算机语言的语句来编写算法。使用计算机语言的语句编写算法被称为“算法的编码化”，如图 1.5 所示。

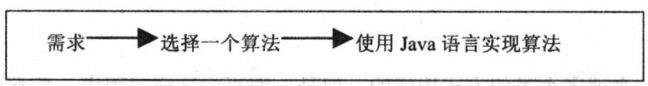


图 1.5　算法的编码化

1.1.2　类和对象的概述

除了将要用来操纵数据以产生输出结果的算法之外，类也是面向对象的语言的重点，这是因为在像 Java 这样的面向对象的语言中，类是一个同时定义将要使用的数据和可以对数据执行的操作的单元。可以执行的操作被正式称为“**方法**”（**Method**），它们也被非正式地称为“**过程**”（**Procedure**）和“**函数**”（**Function**），我们将在讨论类时互换使用这 3 个术语。个别地方，方法就是指使用适当的 Java 代码编写的算法。

一旦定义了类，就可以创建被称为“对象”的单元。“**对象**”（**Object**）就是来自于类的特定项目。对象与类的关系类似于特定几何形状与一类形状之间的关系，例如，如果我们将 Rectangles 定义为一类四边形的图形，其对边的长度相等，邻边相互垂直，那么一个 2 英尺×3 英尺的特定矩形就是类 Rectangles 的一个对象，这个特殊的对象就是所定义的类的一个特定情况或实例。

除了定义用来创建特定对象的通用特性之外，类还可以提供应用于该类对象的方法或操作，例如，如果我们使用刚才描述的 Rectangles 类，那么可能会提供适当的方法，来计算特定矩形的面积和周长，以及显示和更改矩形对象的尺度。特定的方法是通过向特定对象发送消息来运行所需的方法来激活（面向对象的术语）的，该消息即为：“针对这个特定对象运行该方法”。

图 1.6 展示了两个“Rectangle”类型的对象。请注意，所示的每一个对象都不是实际的矩形，而只是包含两块数据，其中第一块是特定矩形的长度，第二块是特定矩形的宽度。此外，如图 1.6 所示，提

供的 5 个操作包括：创建一个新的矩形、计算矩形的面积、计算矩形的周长、更改矩形的尺度，以及显示矩形的尺度，这些方法是被所有对象共享的。

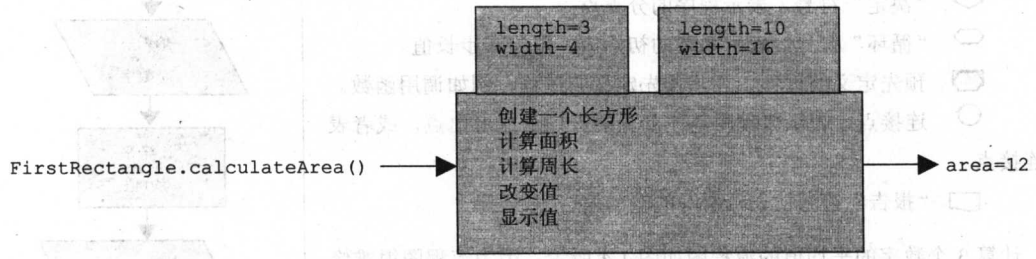


图 1.6 一组 Rectangle 对象

要激活特定的方法，我们必须发送一条相当于“针对这个特定的对象使用这个特定的方法”的消息。实际上，这种消息是很简单的，通过两块使用句点分隔的信息来提供对象的名称和方法的名称即可，例如，消息 firstRectangle.calculateArea()足以告诉 calcuateArea 方法来计算对象 firstRectangle 的面积。我们将在第 1.3 节中看到，方法名称后的括号被用来向该方法提供完成它的操作所需的其他数据。

虽然在熟练掌握 Java 后我们将会定义自己的类并创建对象，但是，只要我们知道如何激活适应预先定义的方法的正确方式，就可以使用 Java 所提供的任何对象和类。因为 Java 提供了很多内置类，并且是以这些内置类为基础的，所以使用 Java 编写程序要求我们至少非常熟悉它的可用类的一部分、它们各自的方法，以及任何已经为我们的使用创建的现有对象。

1.1.3 程序翻译

Java 程序是以一个或多个类的方式构成的，但是，编写完这样的程序，若不进行进一步的翻译，程序仍不能在计算机上得到执行，这是因为所有计算机的内部语言都是由一系列的数字 1 和 0 组成的，这种语言被称为计算机的**“机器语言”**（**Machine Language**）。要生成可以由计算机执行的机器语言程序，需要将 Java 程序——被称为**“源程序”**（**Source Program**）——翻译成计算机的机器语言。在这种翻译过程中，可以实际地体会到 Java 的跨平台能力（在计算机之间传递的能力，这意味着在一台机器上编写并翻译的程序不需要进行重新翻译就可以在另一台机器上得到执行）。

在所有其他主要的计算机语言中（从 1957 年产生的 Fortran 开始，到连续经历的 Pascal、C、C++，以及 Visual BASIC），从源程序到机器语言的翻译过程通常都是采用两种完全不同的方法之一来完成的，这两种方法分别被称为“解释”和“编译”。Java 通过一种经过修改的方式同时使用了这两种方法。

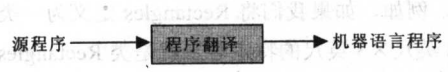


图 1.7 传统的翻译过程

当源程序的每一条语句被分别翻译并立即执行时，这种编程语言被称为**“解释性语言”**（**Interpreted Language**），而执行翻译操作的程序被称为**“解释器”**（**Interpreter**）。另一种翻译源程序的方法是在执行任何语句之前完整地翻译源程序中的所有语句，当采用这种方法时，这种编程语言被称为**“编译性语言”**（**Complied Language**），而执行翻译操作的程序被称为**“编译器”**（**Compiler**）。由编译器产生的输出结果被称为“对象程序”，**对象程序**（**Object Program**）就是源程序经过翻译的版本，它可以通过计算机系统的多个执行步骤来得到执行，下面我们来看一看这样做的原因。

对于输入和输出，以及查找像平方根、绝对值及其他常见的数学计算的等式，大多数计算机程序都

包含一些使用预先编写的例程的语句；此外，较大的程序还可能采用两个独立的程序文件来存储，在这种情况下，可以分别编译每个文件，然而在执行该程序之前，最终必须将两个文件合并成一个程序。在上述两种情况下，都是由链接程序（常常是由编译器自动调用的）来将所有预先编写的例程和个别对象文件组合为一个准备执行的程序，这个得到的最终的程序被称为**“可执行程序”（Executable Program）**。图 1.7 展示了翻译者程序（编译器或解释器）在传统上是如何将源程序翻译成对应的可执行程序的。

Java 程序的翻译过程是一个在同时使用编译器和解释器的传统过程的基础上修改的过程。如图 1.8 所示，编译步骤的输出结果是字节编码格式的程序，这种字节码格式是一种机器码，它不是针对特定计算机的内部处理器的，而是针对被称为“Java 虚拟机”（Java Virtual Machine，JVM）的计算机的。“Java 虚拟机”计算机不是一台实际的计算机，而是一种可以读取并执行由编译器产生的字节码的软件程序。

图 1.8　编译和执行 Java 程序

在其中运行 JVM 的计算机被称为**“主机”（Host Computer）**。如图 1.8 所示，在主机的 JVM 中，字节码最终被翻译成适合于主机的机器语言码。具体来说，JVM 是一种翻译器，当它遇到每个字节码指令时，将它们翻译成特定的、由计算机立即执行的机器码。

这种两阶段的翻译过程提供了 Java 的跨平台能力，它使每台计算机（不管它们的内部处理器类型）都能执行相同的 Java 程序，把最终翻译步骤的具体细节放在主机的 JVM 中实现，而不是放在编译源代码的计算机中。

1.1.4　练习题

1．定义下面的术语：

计算机语言、编程语言、编程、算法、伪代码、流程图、过程、对象、方法、类、源程序、编译器、对象程序、可执行程序、解释器。

2．问题如下：

（1）确定绘制如图 1.9 所示的花的可能的 6 种过程（列出步骤），要求每种颜色必须在开始绘制新的颜色之前完成（提示：其中一种算法是，先使用黄色，再使用绿色，最后使用黑色）。

（2）如果限制只能使用一个画笔，并且无法清洗画笔，那么在这 6 种绘图算法中，哪一种是最好的？

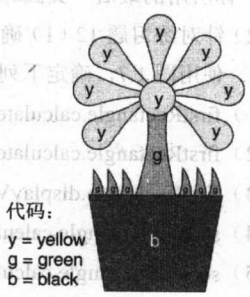


图 1.9　一个简单的按编号绘制的插图

3．确定执行下列任务的逐步过程，即列出步骤（注意：这些任务没有单一的正确答案。该题旨在让读者练习如何将直觉命令转换成相应的算法，并理解这两种响应方式所涉及的思考过程的差异）。

（1）使用备用轮胎更换磨平的轮胎。

（2）拨打电话。

（3）去商店购买面包。

（4）烘烤火鸡。

4．确定并编写一个算法（列出步骤）来互换两杯液体的内容。假设可以使用第三个杯子来临时盛放其中一杯液体。在倒入新的液体之前，必须清洗每个杯子。