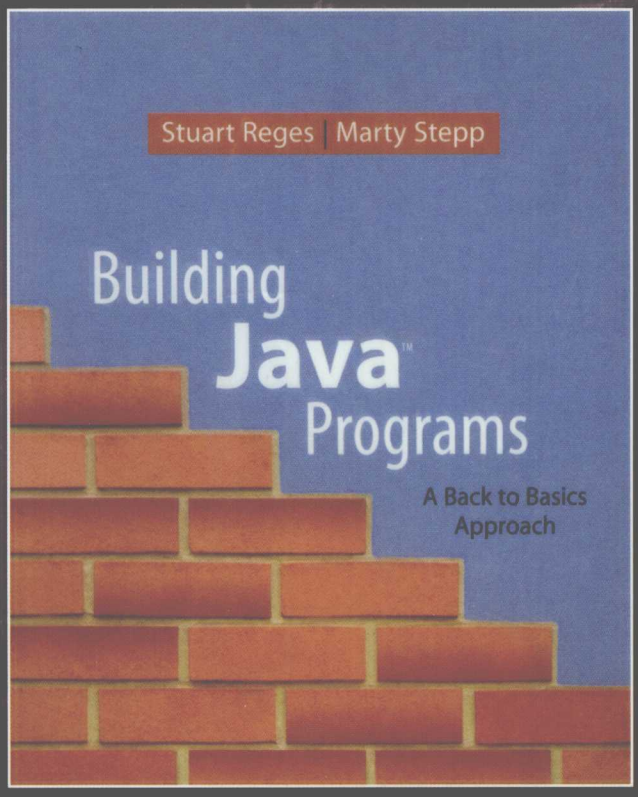
HZ BOOKS
华章教育

PEARSON
Addison Wesley

计 算 机 科 学 丛 书

Java程序设计教程

(美) Stuart Reges 华盛顿大学 Marty Stepp 著 陈志 南开大学 等译



Building Java Programs
A Back to Basics Approach

机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

Java程序设计教程

(美) Stuart Reges 华盛顿大学 Marty Stepp 著 陈志 南开大学 等译

Building Java Programs

A Back to Basics Approach

机械工业出版社
China Machine Press

本书采用了“从基础开始”的讲法，先介绍面向过程的程序设计方法，打下牢固的编程基础后，再讲述面向对象的程序设计方法。主要内容包括：程序设计基础、数组、定义类、递归、继承和接口、ArrayList、图形用户接口、排序和查找、集合等。本书在大部分章节都配有自测题和练习题，对理解和消化书中的概念极有帮助，非常适合学生和初学者自学参考。

本书是为计算机专业程序设计课程而编写的一本教材，也可以作为学习Java语言的入门读物。对于软件工程师、系统集成工程师以及应用和维护等相关人员来说，也不失为一本好的参考读物。

本书版权登记号：图字：01-2007-2444

图书在版编目（CIP）数据

Java程序设计教程 /（美）李杰斯（Reges, S.），（美）施特普（Stepp, M.）著；陈志等译. —北京：机械工业出版社，2008.9

（计算机科学丛书）

书名原文：Building Java Programs: A Back to Basics Approach

ISBN 978-7-111-24661-9

Ⅰ. J…　Ⅱ. ① 李…　② 施…　③ 陈…　Ⅲ. Java语言—程序设计—教材　Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第105201号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：王　璐

北京瑞德印刷有限公司印刷 · 新华书店北京发行所发行

2008年9月第1版第1次印刷

184mm × 260mm · 41.75印张

标准书号：ISBN 978-7-111-24661-9

定价：85.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com

电子邮件：hzjsj@hzbook.com

联系电话：(010) 88379604

联系地址：北京市西城区百万庄南街1号

邮政编码：100037

华章教育

译者序

程序设计是计算机专业的入门课也是必修课。从20世纪40年代计算机诞生以来，程序设计方法先后经历几代发展，从结构化程序设计发展到如今的面向对象程序设计。Java语言凭借完全面向对象的特性，及其特有的跨平台特性和强大的类库支持，逐渐成为程序设计课程的首选语言。但不可忽视的是，Java语言中引入了很多新概念，如对象、类等，对于基础比较薄弱的初学者而言，掌握这部分内容有一定困难。因此，如何能让学生顺利掌握面向对象的程序设计方法并设计程序解决实际问题，就成为教学的难点。

本书的作者以自己的实际教学经验为依据，创新地采用了“从基础开始”、“从学习面向过程的程序设计开始”的方法，将程序设计的基本知识与面向对象程序设计方法划分为两个阶段并分别进行介绍。当读者具备了扎实的编程基础之后，再学习相对复杂的面向对象程序设计。对初学者来说，这种方式有效地分散了重点和难点，避免了因内容的难度过大而影响教学效果。而且本书在讲授基本编程知识的过程中充分利用了Java语言本身提供的类库支持，使学生在学习过程中可以利用这些预置功能编写出具有实际意义的程序，这也会增加学生学习的乐趣与成就感。总之，“先学习使用对象，再学习定义对象”是本书在安排和组织内容上的一大特色。

本书的第1～5章由李静翻译，第6～11章和附录由陈志翻译，第12～14章由何亮翻译，并由陈志负责全书翻译的组织工作。由于译者的专业水平和时间的双重限制，错误和不妥之处诚恳地希望读者批评指正。

本书可以作为计算机专业程序设计课程的教材，也可以作为学习Java语言的入门读物。对于软件工程师、系统集成工程师以及应用和维护等相关人员来说，也不失为一本好的参考读物。

译 者

2008年春于南开大学

前　言

本书是为计算机专业第一门程序设计课程而编写的一本教材，同样适用于非计算机专业学生学习程序设计。书中所使用的内容经过了2年半的实际教学检验，在此期间上千名华盛顿大学的本科生（大多数不是计算机专业的学生）使用了这本教材。

一直以来，很多学校计算机专业的第一门程序设计课程都是新生的梦魇。但是道格拉斯·亚当斯在《银河系漫游指南》中告诉大家“不要惊慌”。学生可以逐步掌握课程讲授的内容。

很多教师尝试了不同的方法来讲授入门课程。最典型的尝试包括“尽早介绍对象”这种方法。但我们感觉这些尝试的最终效果往往不尽如人意。尽管优等生掌握的效果不错，但更多的中等学生都在吃力地应付对象的概念。

本书采用了“从基础开始”的方法，强调应该先介绍面向过程的程序设计方法。实际的教学经验告诉我们，通过学习面向过程的方法，大多数学生可以更好地掌握程序设计的技能。一旦学生打下了牢固的编程基础，就可以转而学习面向对象的程序设计方法。所以，到课程结束时，学生可以掌握这两种程序设计方法。

下面列出了本书的一些主要特色：

- **先使用对象。**Java本身提供了很多功能强大的对象。通过使用这些对象，学生采用传统的面向过程的程序设计技术也能写出很多有趣的程序。尽可能早而且尽可能多地使用这些对象有利于学生们在后面学习如何设计并创建对象。
- **后定义对象。**如果让学生在学习基本的程序设计技巧的同时学习如何定义对象，他们往往会感觉难以消化吸收如此多的内容。我们将这两部分内容分开，先讲授编程的基本知识，然后再讲授如何定义对象。
- **关注如何解决问题。**很多教材将重点放在程序设计语言的各个语法细节上，而我们更关注如何用程序解决问题。如，语言的一种新的功能可以帮助我们解决哪些新的问题？初学者容易犯的错误有哪些？这种新功能的最典型用法是什么？
- **强调算法思维。**在使用面向过程的方法来解决问题时，我们强调如何通过使用算法来解决问题：将一个大问题分解为若干个小问题，用伪代码逐步细化解决问题的算法，并努力以算法的方式去描述问题。
- **层层推进。**采用Java语言编写程序需要同时涉及很多概念，学生不可能在一夜之间就掌握所有这些概念。教授初学者学习Java语言就好像搭积木一样，每一块都要小心翼翼地放置。如果一次放置的积木太多，反而会带来倒塌的危险。我们一步步地引入新概念，确保学生能够掌握每一步所介绍的内容。
- **全面介绍Java 5的特性。**本书介绍的内容以Java 5为基础，全面介绍了这一版引入的一些新特性，例如，`Scanner`类、泛型、for-each循环、装箱和拆箱等。Java 6是对Java 5的补充和升级，并不会影响本书所介绍的内容，本书的内容与Java 6完全兼容。
- **案例分析。**我们在每章的最后都安排了一个案例分析。通过这部分内容，学生可以了解如何分阶段开发并测试一个复杂的程序。而且我们可以用相对完整的程序代码来具体展示这一章所介绍的新的编程概念，这是书中那些短的代码示例所不能做到的。

补充了绘图内容。我们提供了一个自定义的`DrawingPanel`类，通过它可以使用Java中丰富的绘图功能。很多学生对输出复杂图形很感兴趣，但本书只将这一部分作为选学内容。

本书的组织结构和各章之间的依赖关系

很多计算机系的入门教材都是面向语言特性来进行介绍，但是我们这本书的前半部分采用了逐层推进的方式。例如，Java语言中有很多控制结构（例如，for循环、while循环、if/else语句），很多教材都把这些内容放到一章进行介绍。当然，这对于了解程序设计的人来说很正常，但对于初学者来说就会感觉很吃力。我们发现将这些控制结构分散到不同章节去介绍的效果很好，这样学生一次只需要专心学习一个内容，而不必一次将所有内容都塞进脑袋中。

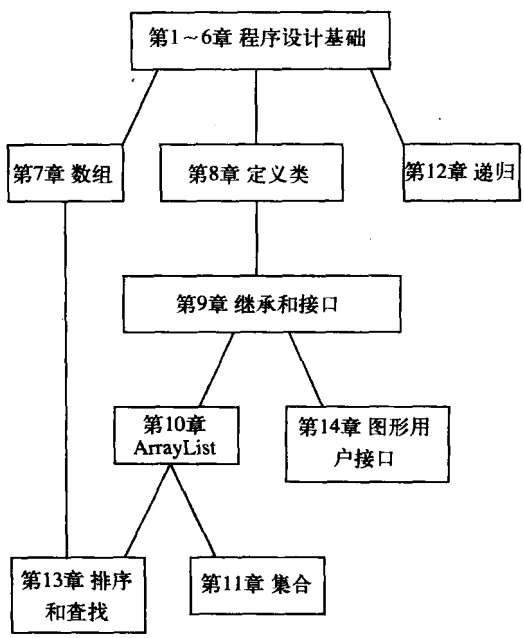
下表给出了本书前6章内容之间的顺序关系

章节	控制流程	数据	编程技术	输入/输出
1	方法	String源文本	过程分解	println
2	确定循环(for)	变量，表达式int,double	局部变量、类常量、伪代码	print
3	返回值	使用对象	参数	控制台输入 输出图形（选学）
4	条件执行(if/else)	char	前提/后置条件抛出异常	printf
5	不确定循环(while)	boolean	断言、程序健壮性	
6		Scanner	基于标记的处理 基于行的处理	文件输入/输出

下图是本书各章之间的依赖关系：

应该按照顺序学习本书的前1到6章，从第7章开始可以选择自己感兴趣的内容。

第6章专门讲述文件处理的内容，但是很多Java入门课程都去掉了这部分内容。最主要的原因还是在Java 5之前，用Java程序来处理文件是件很复杂的事情。自从Java 5引入了Scanner类，我们认为入门课程也完全可以把这部分内容包括在内。这样我们就有能力解决一些涉及到大量数据的程序。即便如此，很多教师也不愿意在这个主题上花费太多时间，所以我们将第6章的后半部分也设定为选学部分。



补充材料

我们为教师和学生准备了丰富的补充材料。附录A中提供了自测题的答案。学生还可以登录网站www.aw-bc.com/cssupport获得：

- 书中出现的源码。
- 案例分析中使用的数据文件和完整的程序代码。
- 第3章补充内容中介绍的用于绘制图形的`DrawingPanel`类。

除此之外，书中的源码和很多Java开发环境，可登录本书的配套网站下载。经过确认采用本书作为教材的教师还可以获得以下资源：

- PowerPoint课件
- 练习题和编程题目的答案
- 考试样卷
- 实验手册
- 补充编程习题（包括答案要点）

要获取这些教师资源，请按书后教学支持申请表中的联系方式联络培生教育出版集团北京代表处。

本书特点

- 术语定义

书中突出强调了所有重要的术语或概念，便于读者快速参考。

- “你知道吗？”

这部分提供了与书中介绍的主题或概念相关的背景知识或趣闻轶事。

- “常见编程错误”

这部分提供了一些初学者常犯的错误。

- 代码

书中提供了大量代码，并且将代码中关键之处用高亮方式显示。代码中的保留字采用了不同的颜色。语法模板中的占位符放在一对尖括号内，而且用灰色高亮显示。

- 伪代码

书中的伪代码采用了与普通代码不同的字体。

- “拇指向下”错误代码图标

有些情况下，书中给出一些有缺陷或错误的代码来说明程序设计中常见的问题或错误。凡带有“拇指向下”标记的代码都含有某些缺陷或错误。

- 本章小结

每章的最后都给出了一个包含该章主要概念和内容的简单总结。

- 自测题

每章最后包含了一组自测题，用于检查读者对每一节重点内容的掌握情况。我们在本书的附录A中提供了自测题的答案。

- 练习题

这部分提供了一些难度适中的练习题，这些题目要求读者必须对本章内容有深入的了解。除此之外，还要进行一些计算、分析并编写代码。教师可以获得这部分习题的答案。

- 编程练习题

这一部分要求读者设计并编写实际的Java程序。这些题目难度不一，但总的来说，比练习

题的难度要大。教师可以获得这部分习题的参考答案。

致谢

首先，我们要感谢选过这门课程的学生和助教，他们为这本书的初稿提出了很多修改意见。没有他们的积极反馈就没有今天这本书。我们要特别感谢Helene Martin，他仔细阅读了本书的初稿并指出了书中存在的一些错误。

其次，我们要感谢华盛顿大学计算机科学与工程系的同事。学术委员会给了我们筹备和讲授这门课程的机会。我们要特别感谢Hank Levy（现任主席），David Notkin（前任主席）和Richard Anderson（负责教学的副主席）对我们的全力支持。我们还要感谢Steve Gribble和Carl Ebeling，他们最早在教学中使用了这本书，并给我们提供了有价值的反馈信息。

第三，我们要感谢参与了本书修订工作的同行，正是在他们的帮助下，我们才能够将最初的一份课件变为今天这样一本完整的教材：

Delroy A. Brinkerhoff，韦伯州立大学；Ed Brunjes，Miramar社团学院；Tom Cortina，卡内基一梅隆大学；H.E. Dunsmore，普度大学；Mary Anne Egan，Siena学院；Ahmad Ghafarian，北佐治亚学院和州立大学；Raj Gill，Anne Arundel社团学院；Michael Hostetler，Park大学；David Hovemeyer，宾夕法尼亚约克学院；Chenglie Hu，卡罗学院；Philip Isenhour，弗吉尼亚工学院；Andree Jacobson，新墨西哥大学；David C. Kamper Sr.，东北伊利诺伊大学；Simon G.M. Koo，圣地亚哥大学；Evan Korth，纽约大学；Joan Krone，Denison大学；Eric Matson，莱特州立大学；Kathryn S. McKinley，得克萨斯大学奥斯汀分校；Jerry Mead，巴科内尔大学；George Medelinskas，北Essex社团学院；John Neitzke，Truman州立大学；Richard E. Pattis，卡内基一梅隆大学；Frederick Pratter，东俄勒冈大学；Roger Priebe，得克萨斯大学奥斯汀分校；Dehu Qi，拉摩尔大学；Amala V S Rajan，Middlesex大学；Mike Scott，得克萨斯大学奥斯汀分校；Tom Stokke，北达科塔大学；Leigh Ann Sudol，Fox Lane高等学校；Ronald F. Taylor，莱特州立大学；Scott Thede，DePauw大学；Megan Thomas，加州州立大学Stanislaus分校；Jeannie Turner，Sayre学校；Tammy VanDeGrift，波特兰大学；Thomas John VanDrunen，Wheaton学院；Neal R. Wagner，得克萨斯大学圣安东尼奥分校；Jiangping Wang，韦伯斯特大学；Yang Wang，密苏里州立大学；Stephen Weiss，北卡罗来纳大学坎斐尔分校；Laurie Werner，迈阿密大学；Dianna Xu，Bryn Mawr学院；Carol Zander，华盛顿大学Bothell分校。

我们还要感谢本书的编辑Rachel Head，以及Argosy出版社的其他工作人员，包括Nancy Kotary和Kathleen Kenny。

最后，我们要感谢Addison-Wesley的工作人员，他们在过去两年中确保了本书能够最终完成：感谢Michelle Brown传授给我们很多表达方面的技巧，感谢Jeff Holcomb为本书完成了封面设计，感谢Maurene Goo帮助我们处理了很多细节工作，感谢Patty Mahtani把所有内容整理成册，最后要特别感谢本书的编辑Matt Goldstein从第一天开始就给予我们的支持和信任。没有这些人的帮助就不会有这本书。

目 录

第1章 Java编程简介

引言

在本章开头我们首先要回顾一些与计算机和计算机编程相关的基本概念和术语。由于很多概念会在后边的章节中用到，所以我们在深入探讨Java编程之前，有必要重新温习一下这些概念。

我们以分析一些带有输出结果的简单的程序开启我们的Java探索之旅。这样我们能够在结构简单的程序中，熟悉Java编程中很多最常见、最基本的要素。

在重温了这些Java编程的基本要素之后，我们将学习如何把一个Java程序分解成若干个小的片段，并借此来介绍过程分解（procedural decomposition）技术。使用这种技术，我们可以将那些复杂的大任务分解成多个规模较小的、便于管理的子任务。同时，这样还可以避免因重复解决某一子任务而产生的冗余代码。

1.1 计算的基本概念

计算机已经融入了我们的日常生活，因特网给我们带来了几乎无限量的信息。有的信息是一些重要新闻，比如cnn.com上的头条新闻；有的则是无足轻重的垃圾信息：如果你想知道昨晚碰到的小伙子是否欺骗了他的女友，你可能会到dontdatehimgirl.com去找答案。计算机能够让我们和家庭成员分享照片，也能提供到最近的比萨饼店的路线。

很多现实中的问题都在借助计算机来寻求解决，然而所用的计算机不都是我们日常使用的计算机或笔记本电脑。人类基因测序和DNA片段比对用的是高性能计算机；新型汽车中会包含用来监测行驶状况的计算机；像iPod这样的数字音乐播放器，在它们小巧的外壳下面也是计算机；甚至连Roomba牌家用清洁机器人在清洁地板时，也是依靠内置的计算机来避开家具等障碍物。

那什么才是计算机呢？普通计算器是不是计算机？拿着纸和笔的人是不是也可以看作是计算机？下面几节我们就试着来回答这些问题，并让读者认识如何通过编程来控制计算机。

1.1.1 为什么要编程

在大学里，计算机系的第一门课往往都选择程序设计。这令很多计算机科学家感到不解，因为这会让人们产生一种印象：计算机科学＝编程。虽然那些训练有素的计算机科学家往往会花费很多时间来编程，但是计算机科学本身所涵盖的内容却远远不止程序设计。那为什么我们还要在一开始就学习程序设计呢？

斯坦福大学的著名计算机科学家Don Knuth回答了这个问题，他说计算机科学中的各个方面都或多或少与*算法*（algorithm）相关。

算法

按步骤描述如何完成一个任务的规则。

Knuth是算法领域的专家，他自然会倾向于把算法看作是计算机科学的核心。他认为最为重要的不是算法本身，而是计算机科学家处理问题时所用到的思考过程。Knuth说：

有人曾说一个人只有教给别人某件事时才会真正明白这件事。实际上一个人只有告诉计算机如何去做一件事时才会真正明白这件事，即用算法来表达这件事在计算机中如何实现。⊖

Knuth把计算机科学中具有共性的思考过程称为*算法性思考*（algorithmic thinking）。所以我们学习程序设计不仅仅因为它是计算机科学中最重要的一个方面，更因为它是阐述计算机科学家如何解决问题的最佳途径。

本书要讲的正是如何把算法写成计算机程序。但是在我们学习编程之前，回顾一下计算机的基本概念还是很有必要的。

1.1.2 硬件和软件

计算机（computer）就是一台可以操作数据并且执行一系列指令的机器，这些指令又被称为*程序*（program）。

程序

一系列被计算机执行的指令。

真正的计算机与类似于计算器的简单机器之间的最大不同点在于，计算机具有多功能性。一台计算机能完成多种不同的任务（玩游戏、计算所得税、与世界上其他计算机进行联网等），这完全取决于它在某一时刻所运行的是何种程序。计算机不仅可以运行本机上的现有程序，还可以运行那些将要编写的新程序。

组成计算机的物理部件称为*硬件*（hardware）。最重要的硬件是中央处理器，也称为CPU，它是计算机的“大脑”，计算机通过它来执行各种指令。同样重要的还有计算机的*内存*（memory）（通常也叫随机访问存储器或者RAM，因为计算机可以随时访问该存储体的任何部分）。计算机用内存来存储需要运行的程序，以及相关的数据。RAM的容量很有限，并且其中的内容会在计算机关闭后丢失。因此计算机通常用*硬盘*来作为更大的永久性存储区。

各种计算机程序统称为*软件*（software）。计算机上最重要的软件是*操作系统*（operating system）提供了在同一时刻运行多个应用程序的环境，并且在应用程序、硬件和用户之间搭起了一座桥梁。运行在操作系统内部的程序通常称为*应用程序*（application）。

当用户要求操作系统运行一个程序时（例如通过双击桌面上某个应用程序的图标），会引发一系列处理动作。计算机将该程序的指令从硬盘装载到内存中，所需的内存由操作系统负责分配。然后CPU会不断地从内存中获取指令，并按顺序执行它们。

1.1.3 数字王国

在上一节我们已经看到计算机是一种通用设备，可以通过编程使其完成某些特定任务。你可能经常听到人们把现代计算机称为*数字*（digital）计算机，这是由它的工作方式决定的。⊜

数字

以离散（非连续）的方式增长的数字，比如整数（0，1，2，3，等等）。

因为计算机采用的工作方式是数字方式，所以计算机内的全部信息都是用整数序列来存储的，这包括所有程序和数据。在20世纪40年代，人们发明了第一台计算机，当时，这种用整数来表示任何事物的思想是一种前所未有的突破。然而今天，数字音乐、数字图像以及数

⊖ 摘自Knuth, Don. *Selected Papers on Computer Science*. Stanford. CA: Centre for the Study of Language and Information, 1996.

⊜ 与数字方式对应的是模拟方式。——译者注

字电影已经随处可见，这种思想变得再平常不过。例如，一个MP3文件只是一大串存储着音频编码信息的整数序列。

计算机不仅基于数字方式（即所有信息都用整数来表示），更进一步地说，它采用的是二进制（binary）的工作方式，这就意味着计算机中的整数序列都是以二进制数字（binary number）来存储的。

二进制数字

只有0和1组成的数字，采用逢二进位。

人们日常使用的都是十进制（逢十进位）的数字。既然人们习惯于十进制数字，那为什么非要让计算机使用二进制数字呢？人类使用十进制是由特定的生理情况（10个手指和10个脚趾）决定的。但对于计算机来说，我们的目标是构建一个简单而可靠的系统。事实证明，可以利用一些只有两种情况的事物（例如，电路的断开和闭合）来构建只有两种状态的系统，这要比构建一个需要区分10种状态（例如，10个不同的电压等级）的系统更加简单。

从算术的角度看，使用二进制还是十进制并没有差别。但是制造能够进行二进制运算的设备要容易得多，所以计算机系统普遍采用二进制。

然而，不熟悉计算机的人往往不会了解二进制。所以，花点时间来回顾一下二进制数字如何工作是值得的。在二进制里你还是从0开始往上数，就像你使用十进制计数一样，但是你会更快地遇到进位的情况。采用二进制计数时你数到：

```
0
1
```

这时就要进位了，这就像你在十进制计数时数到了9一样。向高位进位后，继续增加低位。所以紧跟着的两个数字是：

```
10
11
```

这时又要进位了，这类似于我们在十进制中数到了99。然后我们需要再增加一位，形成三位数100。在二进制中，当你遇到类似于111111这样的全“1”序列时，你会知道如果这个数字再加1的话，就需要增加新的一位，具体方法是将整个序列的每一位都置为0，并在这些“0”之前增加一个数字“1”。这一点和我们在十进制中遇到的情况完全一样：999999加1，得到的结果就是1000000。下边的表格说明我们怎样用二进制表示0到8。

十进制	二进制
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000

从表中我们可以看出二进制数的一些规律。注意在上边表格中的二进制数1，10，100，1000正好对应2的乘方（2^0，2^1，2^2，2^3）。就像我们在10进制中说到的个位、十位、百位等一

样，我们也可以说二进制中有个位、二位、四位、八位、十六位等等。[⊖]

很快，计算机科学家们发现需要一种衡量二进制数长度大小的方法，于是使用比特（bit）来代表一个二进制位，用字节（byte）来代表8比特。当描述大容量内存区域的大小时，则需要使用千字节（KB）、兆字节（MB），吉字节（GB）等等。很多人认为这里的“千”和普通计量系统中的“千”一样，都代表1000。但这并不准确，只是因为2^{10}（1024）和1000在数值上比较接近，所以才沿用了“千”的表示方法，但实际上一千字节应该是2^{10}字节（1024字节），一兆字节是2^{20}字节（1 048 576字节），一吉字节应该是2^{30}（1 073 741 824字节）等等。

程序设计的过程

“code”这个词可以作为名词，表示程序的片段（如“四行的代码（code）”），也可以作为动词，表示编程动作（如“让我们用Java来编程（code）”）。一旦程序写完了，就可以执行（execute）。

程序的执行

运行程序中指令的过程。

程序的执行过程叫运行（running）。当你说“我的程序运行起来有点奇怪”时，它就是一个动词，而当你说“这是我的程序上次运行的结果”时，它又是一个名词。

计算机程序以一系列特殊二进制数字的形式存储在计算机内部，我们称这些特殊二进制数为机器语言（machine language）。在早期，程序员需要直接输入这样的数字（即，使用机器语言）来进行编程。显然这种方式过于烦琐而且很不直观，所以人们又提出了很多方法来简化这个过程。

现代程序员使用高级语言进行程序设计，例如Java。用高级语言书写的程序不能在计算机上直接运行。它们要被一个叫编译器（compiler）的特殊程序翻译成另一种形式才能运行。

编译器

一个可以将计算机程序从一种语言翻译成另一种语言的程序（通常是翻译成机器语言，但也有例外）。

编译器翻译成机器语言后就可以在计算机上直接运行，我们把这样的程序称为可执行程序，把可以将代码编译到最低等级的编译器称为本地编译器（native compiler）。

当你事先知道程序将要在哪台机器上运行时这种方法十分有效，但是如果你想在不同类型的计算机上运行相同的程序该怎么办呢？一种解决办法是使用一种特殊的编译器，它会针对不同计算机将高级语言翻译为不同的机器语言。Java的设计者却使用了另外一种方法。如果想让用一种语言编写的程序可以在互联网上顺利运行，前提就是要保证这些程序能够在很多不同的计算机上运行。编写Java小程序（内嵌在网页中的Java程序）的人们就是要让他们的程序可以在不同的计算机上运行。

与直接编译成机器语言不同的是，Java程序被编译成Java字节码（Java bytecode）。这些字节码既不同于Java程序这样的高级语言，也不同于机器语言这样的低级语言，而是介于这二者之间。关键是这种字节码可以在不同的计算机上运行。实际上，它们是Java虚拟机（Java Virtual Machine，JVM）的机器语言。

Java Virtual Machine（JVM）

一种理论上的计算机，它的机器语言是Java字节码集合。

⊖ 这里是根据每一位的权重来命名该位，例如，二位上的1表示2，八位上的1表示8，等等。因此二进制数1010可以分解为：八位和二位上是1，四位和个位是0，相当于8+0+2+0=10，所以1010对应的十进制数就是10。——译者注

Java虚拟机不是真实的计算机，但与实际的计算机类似。在编译到这个水平之后，再将Java字节码转换成实际的机器指令的工作量就所剩无几了。

在Java编程语言中，任何东西都必须归结为类（class）。

类

是构成Java程序的基本代码单元。

在第8章中我们会对类进行更为详细的介绍，现在我们只需知道每个Java程序都要保存到类中。

为了执行Java类文件，你还需要另个一执行Java字节码的程序。这个程序通常被称为Java运行时（Java runtime），它是Sun公司制定的标准Java运行时环境（Java Runtime Envirment，JRE）。

Java运行时环境

一种可以执行Java字节码的程序。

大部分人可能没有感觉到自己的机器上有Java运行时环境。例如，苹果公司的OS X操作系统自带了Java运行时环境，微软公司的Windows操作系统也预装了Java运行时环境。

为什么选择Java

当Sun公司在1995年发布Java时，还同时发布了描述Java语言的“白皮书”。以下是这篇文章中关键的句子：

> *Java是一种简单的、面向对象的、适用于网络应用的、可解释的、健壮的、安全的、结构自然的、可移植的、高性能的、多线程的、动态的语言。*[⊖]

这句话从多个角度解释了为什么Java可以成为一种优秀的编程语言。对初学者而言，Java是简单易学的，它采用了面向对象的方式，这种方法已经被证明可以成功地编写复杂的大型软件。

Java包含了很多预先写好的软件组件，程序员可以使用这些组件来提高程序的质量。这些现成的软件组件通常叫做库。例如，如果你想要编写一个用来连接因特网上的某个站点的程序，那么Java已经为你准备了一个库用于简化连接过程。Java中有很多库，有些用来绘制图形用户接口（GUI），也有一些库可以从数据库中提取数据，有些则可以进行复杂的数学运算，还有很多具备其他功能的库。这些库统称为Java类库（Java class library）。

Java类库

Java中预置的可以用来解决通用问题的代码集合。

Java中丰富的类库也是它得以广泛流行的一个重要因素。Java 1.5版本中的类库包括3 200项。

另一个使用Java的原因是已经存在一个高人气的程序员社区。程序员可以从网上获得大量在线文档和教程来学习新的技能。这些文档有很多是Sun公司自己编写的，其中有一个API说明，它是详实的Java类库参考资料（“API”代表应用程序接口）。

不像那些用其他语言编写的程序，Java绝对是平台无关的，同一个Java程序可以运行在很多不同的操作系统上，例如Windows、Linux和Mac OS X等。

Java被广泛应用于科研和商业领域，这意味着一个熟练的Java程序员具有广泛的就业前景。举一个例子，在Google上搜索关键字“Java jobs”可以获得124 000 000条返回结果。

⊖ 摘自http://java.sun.com/docs/overviews/java/java-overview-1.html。

Java编程环境

在开始学习编程之前你应该熟悉你的计算机设置。每台计算机都为程序开发提供了不同的环境，但是有些共性需要说明一下。无论使用何种环境，你都应该遵从以下三个步骤：

1. 敲入一段Java程序。
2. 编译程序文件。
3. 运行编译后的程序。

对于大多数计算机而言，最基本的存储单元是文件（file）。每个文件都有一个文件名。文件名以一个扩展名（extension）作为结尾，它紧随着文件的名字用一个点字符作为分隔。文件的扩展名表明了这个文件中存储的是何种类型的数据。例如，以.doc作为扩展名的文件是Word文档，以.mp3作为扩展名的则是MP3音频文件。

编写好的Java程序文件必须以.java作为扩展名。当你编译一个Java程序时，最终的编译好的Java字节码被存储在相同的文件名下，只是扩展名变成了.class。

大多数程序员使用集成开发环境（Integrated Development Environment，IDE，该环境是一个集创建、编辑、编译、执行程序等多种功能为一体的开发环境。现在导论性的计算机课程中比较流行的开发环境是Eclipse、DrJava、BlueJ和TextPad。你的老师将会告诉你应该使用何种开发环境。

试着在你的IDE中敲入以下的简单程序：

```
1  public class Hello {
2      public static void main(String[] args) {
3          System.out.println("Hello, world!");
4      }
5  }
```

目前，先不要深究这个程序的细节。在下一节我们会加以探讨。

编好了你的程序后，进入步骤2——编译程序。在不同的开发环境中编译命令是不同的，但是处理过程却是相同的（典型的命令是“complie”或者“build”）。如果编译报错，那就要回到编辑状态下，修改好之后重新编译。（在本章稍后我们会详细地讨论出现的错误。）

当你成功地编译好你的程序之后，就该到步骤3——运行程序。同样，运行的命令也会随着开发环境的不同而不同，但是处理过程也是大同小异的（典型的命令是“run”）。

图1-1概括了从创建到执行一个名为Hello.java程序的过程。

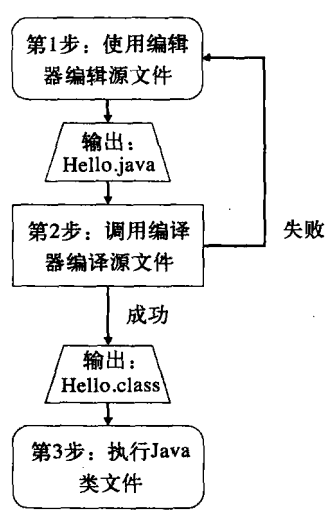


图1-1 Java程序的创建和执行

在一些IDE中（特别是Eclipse），前两个步骤是在一起的。在这些IDE中编译过程是循序渐进的；当你输错代码，就会被开发环境警告。通常在这种环境下，没有必要特意去执行编译，因为在你输代码时编译过程就已经开始了。

程序执行时，会与用户以某种方式进行交互。Hello.java程序在一个叫做控制台（console）的屏幕窗口下执行。

控制台窗口

一种只有文字的特殊窗口，用于Java程序与用户交互。

控制台窗口是一种典型的交互机制，在这里计算机