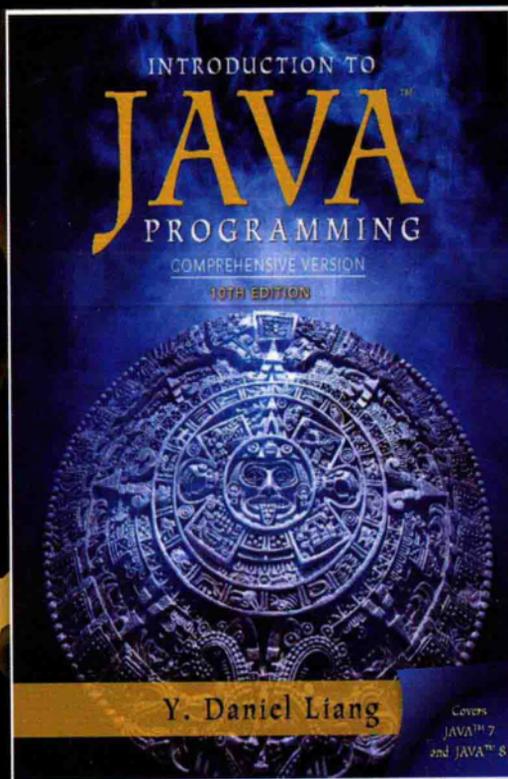


Java语言程序设计 (基础篇)

[美] 梁勇 (Y. Daniel Liang) 著 戴开宇 译
阿姆斯特朗亚特兰大州立大学 复旦大学

Introduction to Java Programming
Comprehensive Version Tenth Edition



提供各种书籍的pd电子版代找服务，如果你找不到自己想要的书的pdf电子版，我们可以帮您找到，如有需要，请联系QQ1779903665.

PDF代找说明：

本人可以帮助你找到你要的PDF电子书，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签索引和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我QQ1779903665。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ，大部分我都可以找到，而且每本100%带书签索引目录。因PDF电子书都有版权，请不要随意传播，如果您有经济购买能力，请尽量购买正版。

声明：本人只提供代找服务，每本100%索引书签和目录，因寻找pdf电子书有一定难度，仅收取代找费用。如因PDF产生的版权纠纷，与本人无关，我们仅仅只是帮助你寻找到你要的pdf而已。

Java语言程序设计 (基础篇) 原书第10版

Introduction to Java Programming Comprehensive Version Tenth Edition

本书是Java语言的经典教材,多年来畅销不衰。本书全面整合了Java 8的特性,采用“基础优先,问题驱动”的教学方式,循序渐进地介绍了程序设计基础、解决问题的方法、面向对象程序设计、图形用户界面设计、异常处理、I/O和递归等内容。此外,本书还全面且深入地覆盖了一些高级主题,包括算法和数据结构、多线程、网络、国际化、高级GUI等内容。

本书中文版由《Java语言程序设计 基础篇》和《Java语言程序设计 进阶篇》组成。基础篇对应原书的第1~18章,进阶篇对应原书的第19~33章。为满足对Web设计有浓厚兴趣的同学,本版在配套网站上增加了第34~42章的内容,以提供更多的相关信息。

本书特点

- 基础篇介绍基础内容,进阶篇介绍高级内容,便于教师按需选择理想的教材。
- 全面整合了Java 8的特性,对全书的内容进行了修订和更新,以反映Java程序设计的最新技术发展。
- 对面向对象程序设计进行了深入论述,包含GUI程序设计的基础和扩展。
- 提供的大量示例中都包括了对问题求解的详细步骤,很多示例都是随着Java技术的引入不断地进行增强,这种循序渐进的讲解方式更易于学生学习。
- 用JavaFX取代了Swing,极大地简化了GUI编程,比Swing更易于学习。
- 更多有趣示例和练习,激发学生兴趣。在配套网站上还为教师提供了100多道的编程练习题。

作者简介

梁勇 (Y. Daniel Liang) 现为阿姆斯特朗亚特兰大州立大学计算机科学系教授。之前曾是普度大学计算机科学系副教授,并两次获得普度大学杰出研究奖。他所编写的Java教程在美国大学Java课程中采用率极高,同时他还兼任Prentice Hall Java系列丛书的编辑。他是“Java Champion”荣誉得主,并在世界各地为在校学生和程序员做JAVA程序设计方法及技术方面的讲座。

译者简介

戴开宇 复旦大学软件学院教师,工程硕士导师,中国计算机协会会员。博士毕业于上海交通大学计算机应用专业,2011~2012年在美国佛罗里达大学作访问学者。承担多门本科专业课程、通识教育课程以及工程硕士课程,这些课程被评为校精品课程,上海市重点建设课程,IBM-教育部精品课程等。



PEARSON

投稿热线: (010) 88379604
客服热线: (010) 88378991 88361066
购书热线: (010) 68326294 88379649 68995259

华章网站: www.hzbook.com
网上购书: www.china-pub.com
数字阅读: www.hzmedia.com.cn



上架指导: 计算机程序设计

ISBN 978-7-111-50690-4



9 787111 506904 >

定价: 85.00元

计 算 机 科 学 丛 书

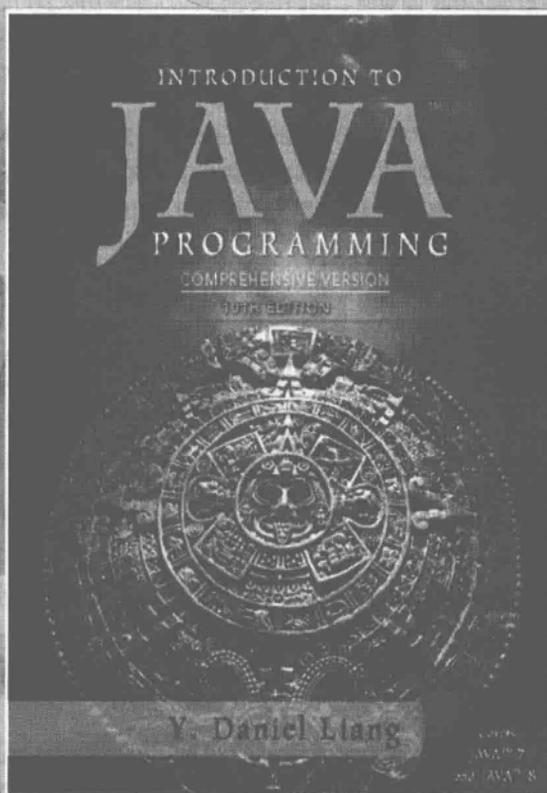
原书第10版

Java语言程序设计

(基础篇)

[美] 梁勇 (Y. Daniel Liang) 著 戴开宇 译
阿姆斯特朗亚特兰大州立大学 复旦大学

Introduction to Java Programming
Comprehensive Version Tenth Edition



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

Java 语言程序设计 (基础篇) (原书第 10 版) / (美) 梁勇 (Liang, Y. D.) 著, 戴开宇译.
—北京: 机械工业出版社, 2015.6

(计算机科学丛书)

书名原文: Introduction to Java Programming, Comprehension Version, Tenth Edition

ISBN 978-7-111-50690-4

I. J… II. ①梁… ②戴… III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2015) 第 131479 号

本书版权登记号: 图字: 01-2014-5466

Authorized translation from the English language edition, entitled *Introduction to Java Programming, Comprehension Version, Tenth Edition*, 978-0-13-376131-3 by Y. Daniel Liang, published by Pearson Education, Inc., Copyright © 2015, 2013, 2011.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Chinese simplified language edition published by Pearson Education Asia Ltd., and China Machine Press Copyright © 2015.

本书中文简体字版由 Pearson Education (培生教育出版集团) 授权机械工业出版社在中华人民共和国境内 (不包括中国台湾地区和中国香港、澳门特别行政区) 独家出版发行。未经出版者书面许可, 不得以任何方式抄袭、复制或节录本书中的任何部分。

本书封底贴有 Pearson Education (培生教育出版集团) 激光防伪标签, 无标签者不得销售。

本书是 Java 语言的经典教材, 中文版分为基础篇和进阶篇, 主要介绍程序设计基础、面向对象程序设计、GUI 程序设计、数据结构和算法、高级 Java 程序设计等内容。本书以示例讲解解决问题的技巧, 提供大量的程序清单, 每章配有大量复习题和编程练习题, 帮助读者掌握编程技术, 并应用所学技术解决实际应用开发中遇到的问题。

基础篇主要介绍基本程序设计、语法结构、面向对象程序设计、继承和多态、异常处理和文本 I/O、抽象类和接口等内容。

本书可作为高等院校相关专业程序设计课程的基础教材, 也可作为 Java 语言及编程爱好者的参考资料。

出版发行: 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: 李 艺

责任校对: 殷 虹

印 刷: 三河市宏图印务有限公司

版 次: 2015 年 7 月第 1 版第 1 次印刷

开 本: 185mm × 260mm 1/16

印 张: 42.25

书 号: ISBN 978-7-111-50690-4

定 价: 85.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

文艺复兴以来，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；也正是这样的优势，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，不仅肇划了研究的范畴，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公司较早意识到“出版要为教育服务”。自1998年开始，我们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相助，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，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，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。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方式如下：

华章网站：www.hzbook.com

电子邮件：hzsj@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



华章科技图书出版中心

中文版序

Introduction to Java Programming, Comprehension Version, Tenth Edition

Welcome to the Chinese translation of Introduction to Java Programming Tenth Edition. The first edition of the English version was published in 1998. Since then ten editions of the book have been published in the last seventeen years. Each new edition substantially improved the book in contents, presentation, organization, examples, and exercises. This book is now the #1 selling computer science textbook in the US. Hundreds and thousands of students around the world have learned programming and problem solving using this book.

I thank Dr. Kaiyu Dai of Fudan University for translating this latest edition. It is a great honor to reconnect with Fudan through this book. I personally benefited from teachings of many great professors at Fudan. Professor Meng Bin made Calculus easy with many insightful examples. Professor Liu Guangqi introduced multidimensional mathematic modeling in the Linear Algebra class. Professor Zhang Aizhu laid a solid mathematical foundation for computer science in the discrete mathematics class. Professor Xia Kuanli paid a great attention to small details in the PASCAL course. Professor Shi Bole showed many interesting sort algorithms in the data structures course. Professor Zhu Hong required an English text for the algorithm design and analysis course. Professor Lou Rongsheng taught the database course and later supervised my master's thesis.

My study at Fudan and teaching in the US prepared me to write the textbook. The Chinese teaching emphasizes on the fundamental concepts and basic skills, which is exactly I used to write this book. The book is fundamentals first by introducing basic programming concepts and techniques before designing custom classes. The fundamental-first approach is now widely adopted by the universities in the US. With the excellent translation from Dr. Dai, I hope more students will benefit from this book and excel in programming and problem solving.

欢迎阅读本书第 10 版的中文版。本书英文版的第 1 版于 1998 年出版。自那之后的 17 年中，本书共出版了 10 个版本。每个新的版本都在内容、表述、组织、示例以及练习题等方面进行了大量的改进。本书目前在美国计算机科学类教材中销量排名第一。全世界无数的学生通过本书学习程序设计以及问题求解。

感谢复旦大学的戴开宇博士翻译了这一最新版本。非常荣幸通过这本书和复旦大学重建联系，我本人曾经受益于复旦大学的许多杰出教授：孟斌教授采用许多富有洞察力的示例将微积分变得清晰易懂；刘光奇教授在线性代数课堂上介绍了多维度数学建模；张霁珠教授的离散数学课程为计算机科学的学习打下了坚实的数学基础；夏宽理教授在 Pascal 课程中对许多小的细节给予了极大的关注；施伯乐教授在数据结构课程中演示了许多有趣的排序算法；朱洪教授在算法设计和分析课程中使用了英文教材；楼荣生教授讲授了数据库课程，并且指导了我的硕士论文。

我在复旦大学的学习经历以及美国的授课经验为撰写本书奠定了基础。中国的教学重视基本概念和基础技能，这也是我写这本书所采用的方法。本书采用基础为先的方法，在设计自定义类之前首先介绍了基本的程序设计概念和方法。目前，基础为先的方法也被美国的大学广泛采用。我希望通过戴博士的优秀翻译，让更多的学生从中受益，并在程序设计和问题求解方面出类拔萃。

梁勇

y.daniel.liang@gmail.com

www.cs.armstrong.edu/liang

Java 是一门伟大的程序设计语言，同时，它还是基于 Java 语言从嵌入式开发到企业级开发的平台。在风起云涌的计算机技术发展历程中，Java 的身影随处可见，而且生命力极其强大。1995 年，Java Applet 使得 Web 网页可以表现精彩和互动的多媒体内容，促进了 Web 的蓬勃发展。之后随着 Web 的发展，应用 Web 成为大型应用开发的主流方式，Java 凭借其“一次编译，到处运行”的特性很好地支持了互联网应用所要求的跨平台能力，成为服务器端开发的主流语言。Java EE 至今依然是最重要的企业开发服务器端平台。2004 年再次产生了对 Web 客户端体验的强烈需求，促使富因特网应用技术广泛流行，从 Java Web Start 到现在的 JavaFX，都是重要的富因特网应用技术。现在我们进入了移动互联网时代，而 Java 依然是当之无愧的主角。从第一阶段移动互联网中的 J2ME，到目前移动操作系统中全球占据份额最大的 Android 系统上的 App 开发，都采用的是 Java 语言和平台。云计算、大数据、物联网、可穿戴设备等技术的应用，都需要可以跨平台、跨设备的分布式计算环境，我们依然会看到 Java 语言在其中的关键作用。除此之外，Java 还是一门非常优秀的教学语言。它是一门经典的面向对象编程语言，拥有优雅和尽量简明的语法以及丰富的实用类库，让编程人员可以尽可能地将精力集中在业务领域的问题求解上。许多开源项目和科研中的原型系统都是采用 Java 实现的。课堂教学采用的语言同时在工业界和学术领域具有如此广泛的应用，对于学生今后的科研和工作都有直接帮助。我曾经对美国计算机专业排名靠前的几十所大学的相关课程进行调研，这些著名大学的编程课程中绝大部分选用了 Java 语言进行教学。

在多年前机械工业出版社举办的一次教学研讨会上，我有幸认识了原书的作者梁勇（Y. Daniel Liang）教授并进行了交流。那次会议之后我开始在主讲的程序设计课程中采用本书英文版作为教材，在同行和学生中得到了良好反响。作为复旦校友，梁教授对中国学生的情况非常了解，书中没有过于晦涩的词汇和表达，所以本英文教材非常适合中国学生的英文基础。更重要的是，本书知识点全面，体系结构清晰，重点突出、文字准确，内容组织循序渐进，并有大量精选的示例和配套素材，比如精心设计的大量练习题，甚至在配套网站中有支持教学的大量动画演示。本书采用基础优先的方式，从编程基础开始，逐步引入面向对象思想，最后介绍应用框架，这样很适合程序设计入门的学生。另外，强调面向问题求解的教学方法是本书特色，这也是我在课堂上一直遵循的教学方法。通过生动实用的例子来引导学生学习程序设计课程，避免了枯燥的语法学习，让学生学以致用，并且可以举一反三。程序设计课堂最重要的是要培养学生的计算思维，这对学生综合素质的培养以及其他知识的学习，都是很有裨益的。掌握了程序设计的思维，可以很方便地学习和使用其他编程语言。该版本的另一特色是对最新 Java 语言特色的跟进，即基于 Java 最新版本 8 进行介绍。这是 Java 语言变动非常大的一个版本，比如对 JavaFX 的全面引入以及并行计算的支持等，都反映了最新的计算机技术和应用特点。相应地，教材也进行了大幅更新。我很荣幸成为本书第 10 版的译者，让中国的读者可以通过这一最新版本的中文版方便地学习程序设计相关知识。

在本书的翻译过程中，我得到了原书作者梁勇教授的大力支持。非常感谢他不仅对我邮件中的一些问题进行快速回复和详细解答，还拨冗写了中文版序，其一丝不苟的学术精神让人感动。感谢机械工业出版社的朱劼编辑，她在本书的整个翻译过程中提供了许多帮助。感谢李艺编辑等其他出版社工作人员以及本书前一版的译者，本书的出版也得益于他们的工作。最后要感谢我的家人在翻译过程中给予的支持和鼓励。由于经验不足和水平有限，书中一定会存在许多问题，敬请得到大家的指正。你们善意的指正，对我和阅读本书的许多读者是有益的。

戴开宇

2015年4月

许多读者就本书之前的版本给出了很多反馈。这些评论和建议极大地改进了本书。这一版从表述、组织、示例、练习题以及附录方面都进行了极大的增强，包括：

- 用 JavaFX 取代了 Swing。JavaFX 是一个用于开发 Java GUI 程序的新框架，它极大地简化了 GUI 程序设计，比 Swing 更易于学习。
- 在 GUI 程序设计之前介绍异常处理、抽象类和接口，若教师选择不教授 GUI 的内容，可以直接跳过第 14 ~ 16 章。
- 在第 4 章便开始介绍对象和字符串，从而使得学生可以较早地使用对象和字符串来开发有趣的程序。
- 包含更多新的有趣示例和练习题，用于激发学生兴趣。在配套网站 (www.cs.armstrong.edu/liang/intro10e/ 或 www.pearsonhighered.com/liang) 上还还为教师提供了 100 多道编程练习题。

本书采用基础优先的方法，在设计自定义类之前，首先介绍基本的程序设计概念和技术。选择语句、循环、方法和数组这样的基本概念和技术是程序设计的基础，它们为学生进一步学习面向对象程序设计和高级 Java 程序设计做好准备。

本书以问题驱动的方式来教授程序设计，将重点放在问题的解决而不是语法上。我们通过使用在各种应用情景中引发思考的问题，使得程序设计的介绍也变得更加有趣。前面章节的主线放在问题的解决上，引入合适的语法和库以支持编写解决问题的程序。为了支持以问题驱动的方式来教授程序设计，本书提供了大量不同难度的问题来激发学生的积极性。为了吸引各个专业的学生来学习，这些问题涉及很多应用领域，包括数学、科学、商业、金融、游戏、动画以及多媒体等。

本书将程序设计、数据结构和算法无缝集成在一起，采用一种实用性的方式来教授数据结构。首先介绍如何使用各种数据结构来开发高效的算法，然后演示如何实现这些数据结构。通过实现，学生获得关于数据结构效率，以及如何和何时使用某种数据结构的深入理解。最后，我们设计和实现了针对树和图的自定义数据结构。

本书广泛应用于全球各大学的程序设计入门、数据结构和算法课程中。完全版[⊖]包括程序设计基础、面向对象程序设计、GUI 程序设计、数据结构、算法、并行、网络、数据库和 Web 程序设计。这个版本旨在把学生培养成精通 Java 的程序员。基础篇可用于程序设计的第一门课程（通常称为 CS1）。基础篇包含完全版的前 18 章内容，前 13 章适合准备 AP 计算机科学考试（AP Computer Science Exam）的人员使用。

教授编程的最好途径是通过示例，而学习编程的唯一途径是通过动手练习。本书通过示例对基本概念进行了解释，提供了大量不同难度的练习题供学生进行实践。在我们的程序设计课程中，每次课后都布置了编程练习。

[⊖] 本书中文版将完全版分为基础篇和进阶篇出版，基础篇对应原书第 1 ~ 18 章，进阶篇对应原书第 19 ~ 33 章，您手中的这一本是基础篇。——编辑注

我们的目标是编写一本可以通过各种应用场景中的有趣示例来教授问题求解和程序设计的教材。如果您有任何关于如何改进本书的评论或建议，请通过以下方式与我联系。

Y. Daniel Liang

y.daniel.liang@gmail.com

www.cs.armstrong.edu/liang

www.pearsonhighered.com/liang

本版新增内容

本版对各个细节都进行了全面修订，以增强其清晰性、表述、内容、例子和练习题。本版主要的改进如下：

- 更新到 Java 8 版本。
- 由于 Swing 被 JavaFX 所替代，因此所有的 GUI 示例和练习题都使用 JavaFX 改写。
- 使用 lambda 表达式来简化 JavaFX 和线程中的编程。
- 在配套网站上为教师提供了 100 多道编程练习题，并给出了答案。这些练习题没有出现在教材中。
- 在第 4 章就引入了数学方法，使得学生可以使用数学函数编写代码。
- 在第 4 章就引入了字符串，使得学生可以早点使用对象和字符串开发有趣的程序。
- GUI 编程放在抽象类和接口之后介绍，若教师选择不教授 GUI 内容的话，可以直接跳过这些章节。
- 第 4、14、15 和 16 章是全新的章节。
- 第 28 和 29 章大幅改写，对最小生成树和最短路径使用更加简化的方法实现。

教学特色

本书使用以下要素组织素材：

- **教学目标** 在每章开始处列出学生应该掌握的内容，学完这章后，学生能够判断自己是否达到这个目标。
- **引言** 提出代表性的问题，以便学生对该章内容有一个概括了解。
- **要点提示** 突出每节中涵盖的重要概念。
- **复习题** 按节组织，帮助学生复习相关内容并评估掌握的程度。
- **示例学习** 通过精心挑选示例，以容易理解的方式教授问题求解和程序设计概念。本书使用多个小的、简单的、激发兴趣的例子来演示重要的概念。
- **本章小结** 回顾学生应该理解和记住的重要主题，有助于巩固该章所学的关键概念。
- **测试题** 测试题是在线的，让学生对编程概念和技术进行自我测试。
- **编程练习题** 为学生提供独立应用所学新技能的机会。练习题的难度分为容易（没有星号）、适中（*）、难（**）和具有挑战性（***）四个级别。学习程序设计的窍门就是实践、实践、再实践。所以，本书提供了大量的编程练习题。
- **注意、提示、警告和设计指南** 贯穿全书，对程序开发的重要方面提供有价值的建议和见解。
 - **注意** 提供学习主题的附加信息，巩固重要概念。

- **提示** 教授良好的程序设计风格和实践经验。
- **警告** 帮助学生避开程序设计错误的误区。
- **设计指南** 提供设计程序的指南。

灵活的章节顺序

本书提供灵活的章节顺序，使学生可以或早或晚地了解 GUI、异常处理、递归、泛型和 Java 集合框架等内容。下页的插图显示了各章之间的相关性。

本书的组织

所有的章节分为五部分，构成 Java 程序设计、数据结构和算法、数据库和 Web 程序设计的全面介绍。因为知识是循序渐进的，前面的章节介绍了程序设计的基本概念，并且通过简单的例子和练习题指导学生；后续的章节逐步详细地介绍 Java 程序设计，最后介绍开发综合的 Java 应用程序。附录包含各种主题，包含数系、位操作、正则表达式以及枚举类型。

第一部分 程序设计基础（第 1～8 章）

本书第一部分是基石，让你开始踏上 Java 学习之旅。你将开始了解 Java（第 1 章），还将学习像基本数据类型、变量、常量、赋值、表达式以及操作符这样的基本程序设计技术（第 2 章），选择语句（第 3 章），数学函数、字符和字符串（第 4 章），循环（第 5 章），方法（第 6 章），数组（第 7～8 章）。在第 7 章之后，可以跳到第 18 章去学习如何编写递归的方法来解决本身具有递归特性的问题。

第二部分 面向对象程序设计（第 9～13 章和第 17 章）

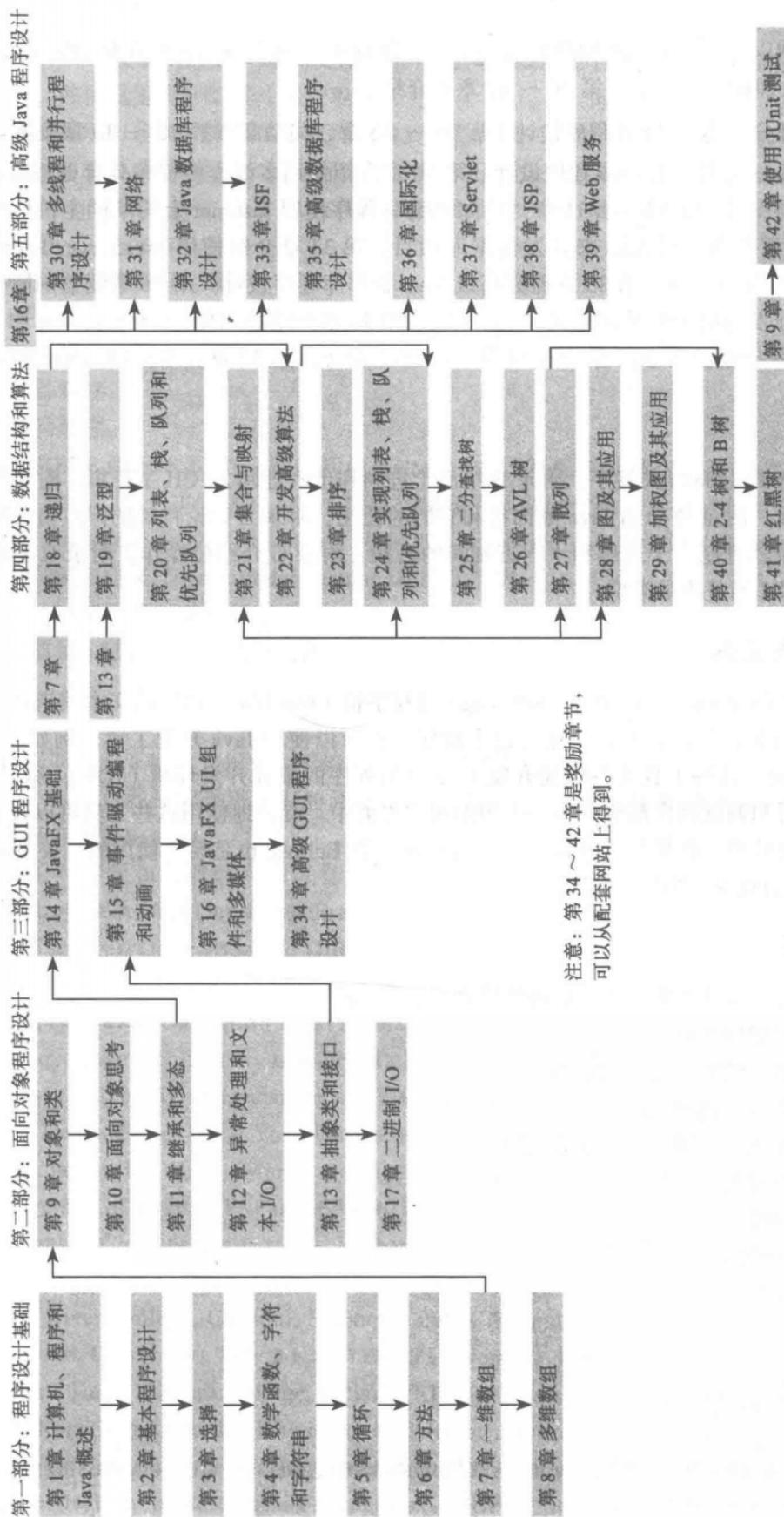
这一部分介绍面向对象程序设计。Java 是一种面向对象程序设计语言，它使用抽象、封装、继承和多态来提供开发软件的极大灵活性、模块化和可重用性。你将学习如何使用对象和类进行程序设计（第 9～10 章）、类的继承（第 11 章）、多态性（第 11 章）、异常处理（第 12 章）、抽象类（第 13 章）以及接口（第 13 章）。文本 I/O 将在第 12 章介绍，二进制 I/O 将在第 17 章介绍。

第三部分 GUI 程序设计（第 14～16 章和奖励章节第 34 章）

JavaFX 是一个开发 Java GUI 程序的新框架。它不仅对于开发 GUI 程序有用，还是一个用于学习面向对象程序设计的优秀教学工具。这一部分中在第 14～16 章介绍使用 JavaFX 的 Java GUI 程序设计。主要的主题包括 GUI 基础（第 14 章）、容器面板（第 14 章）、绘制形状（第 14 章）、事件驱动编程（第 15 章）、动画（第 15 章）、GUI 组件（第 16 章），以及播放音频和视频（第 16 章）。你将学习采用 JavaFX 的 GUI 程序设计的架构，并且使用组件、形状、面板、图像和视频来开发有用的应用程序。第 34 章涵盖 JavaFX 的高级特性。

第四部分 数据结构和算法（第 18～29 章和奖励章节第 40～41 章）

这一部分介绍经典数据结构和算法课程中的主要内容。第 18 章介绍递归来编写解决本身具有递归特性的问题的方法。第 19 章介绍泛型来提高软件的可靠性。第 20 和 21 章介绍 Java 集合框架，它为数据结构定义了一套有用的 API。第 22 章讨论算法效率的度量以便给应用程序选择合适的算法。第 23 章介绍经典的排序算法。你将在第 24 章中学到如何实现经典的数据结构，如列表、队列和优先队列。第 25 和 26 章介绍二分查找树和 AVL 树。第 27



注意：第 34 ~ 42 章是奖励章节，
可以从配套网站上得到。

章介绍散列以及通过散列实现映射 (map) 和集合 (set)。第 28 和 29 章介绍图的应用。2-4 树、B 树以及红黑树在奖励章节第 40 ~ 41 章中介绍。

第五部分 高级 Java 程序设计 (第 30 ~ 33 章、奖励章节第 35 ~ 39 章及第 42 章)

这一部分介绍高级 Java 程序设计。第 30 章介绍使用多线程使程序具有更好的响应和交互性, 并介绍并行编程。第 31 章讨论如何编写程序使得 Internet 上的不同主机能够相互对话。第 32 章介绍使用 Java 来开发数据库项目。第 33 章介绍使用 JavaServer Faces 进行现代 Web 应用程序开发。第 35 章探究高级 Java 数据库程序设计。第 36 章涵盖国际化支持的使用, 以开发面向全球使用者的项目。第 37 和 38 章介绍如何使用 Java servlet 和 JSP 创建来自 Web 服务器的动态内容。第 39 章讨论 Web 服务。第 42 章介绍使用 JUnit 测试 Java 程序。

附录

附录 A 列出 Java 关键字。附录 B 给出十进制和十六进制 ASCII 字符集。附录 C 给出操作符优先级。附录 D 总结 Java 修饰符和它们的使用。附录 E 讨论特殊的浮点值。附录 F 介绍数系以及二进制、十进制和十六进制间的转换。附录 G 介绍位操作。附录 H 介绍正则表达式。附录 I 涵盖枚举类型。

Java 开发工具

可以使用 Windows 记事本 (NotePad) 或写字板 (WordPad) 这样的文本编辑器创建 Java 程序, 然后从命令窗口编译、运行这个程序。也可以使用 Java 开发工具, 例如, NetBeans 或者 Eclipse。这些工具支持快速开发 Java 应用程序的集成开发环境 (IDE), 编辑、编译、构建、运行和调试程序都集成在一个图形用户界面中。有效地使用这些工具可以极大地提高编写程序的效率。如果按照教程学习, NetBeans 和 Eclipse 也是易于使用的。关于 NetBeans 和 Eclipse 的教程, 参见配套网站。

学生资源

学生资源可以从本书的配套网站得到, 具体包括:

- 复习题的答案。
- 偶数号编程练习题的解答。
- 本书例子的源代码。
- 交互式的自测题 (按章节组织)。
- 补充材料。
- 调试技巧。
- 算法动画。
- 勘误表。

教师资源[⊖]

教师资源包括:

⊖ 关于本书教辅资源, 用书教师可向培生教育出版集团北京代表处申请, 电话: 010-57355169/57355171, 电子邮件: service.cn@pearson.com。——编辑注

- PowerPoint 教学幻灯片，通过交互性的按钮可以观看彩色并且语法项高亮显示的源代码，并可以不离幻灯片运行程序。
- 所有编程练习题的答案。学生只可以得到偶数号练习题的答案。
- 100 多道编程练习题，按章节组织。这些练习题仅对教师开放，并提供答案。
- 基于 Web 的测试题生成器。(教师可以选择章节以从 2000 多个大型题库中生成测试题。)
- 样卷。大多数试卷包含 4 个部分：
 - 多选题或者简答题。
 - 改正编程错误。
 - 跟踪程序。
 - 编写程序。
- ACM/IEEE 课程体系 2013 版。新的 ACM/IEEE 计算机科学课程体系 2013 版将知识主体组织成 18 个知识领域。为了帮助教师基于本书设计课程，我们提供了示例教学大纲来确定知识领域和知识单元。示例教学大纲用于一个三学期的课程系列，作为一个学院自定义 (institutional customization) 示例。
- 具有 ABET 课程评价的样卷。
- 课程项目。通常，每个项目给出一个描述，并且要求学生分析、设计和实现该项目。

致谢

感谢阿姆斯特朗亚特兰大州立大学给我机会讲授我所写的内容，并支持我将所教的内容编写成教材。教学是我持续改进本书的灵感之源。感谢使用本书的教师和学生提出的评价、建议、错误报告和赞扬。

由于有了对本版和以前版本的富有见解的审阅，本书得到很大的改进。感谢以下审阅人员：Elizabeth Adams (James Madison University), Syed Ahmed (North Georgia College and State University), Omar Aldawud (Illinois Institute of Technology), Stefan Andrei (Lamar University), Yang Ang (University of Wollongong, Australia), Kevin Bierre (Rochester Institute of Technology), David Champion (DeVry Institute), James Chegwidden (Tarrant County College), Anup Dargar (University of North Dakota), Charles Dierbach (Towson University), Frank Ducrest (University of Louisiana at Lafayette), Erica Eddy (University of Wisconsin at Parkside), Deena Engel (New York University), Henry A Etlinger (Rochester Institute of Technology), James Ten Eyck (Marist College), Myers Foreman (Lamar University), Olac Fuentes (University of Texas at El Paso), Edward F. Gehringer (North Carolina State University), Harold Grossman (Clemson University), Barbara Guillot (Louisiana State University), Stuart hansen (University of Wisconsin, Parkside), Dan Harvey (Southern Oregon University), Ron Hofman (Red River College, Canada), Stephen Hughes (Roanoke College), Vladan Jovanovic (Georgia Southern University), Edwin Kay (Lehigh University), Larry King (University of Texas at Dallas), Nana Kofi (Langara College, Canada), George Koutsogiannakis (Illinois Institute of Technology), Roger Kraft (Purdue University at Calumet), Norman Krumpe (Miami University), Hong Lin (DeVry Institute), Dan Lipsa (Armstrong Atlantic State University), James Madison (Rensselaer Polytechnic Institute), Frank Malinowski (Darton College), Tim Margush (University

of Akron), Debbie Masada (Sun Microsystems), Blayne Mayfield (Oklahoma State University), John McGrath (J.P. McGrath Consulting), Hugh McGuire (Grand Valley State), Shyamal Mitra (University of Texas at Austin), Michel Mitri (James Madison University), Kenrick Mock (University of Alaska Anchorage), Frank Murgolo (California State University, Long Beach), Jun Ni (University of Iowa), Benjamin Nystuen (University of Colorado at Colorado Springs), Maureen Opkins (CA State University, Long Beach), Gavin Osborne (University of Saskatchewan), Kevin Parker (Idaho State University), Dale Parson (Kutztown University), Mark Pendergast (Florida Gulf Coast University), Richard Povinelli (Marquette University), Roger Priebe (University of Texas at Austin), Mary Ann Pumphrey (De Anza Junior College), Pat Roth (Southern Polytechnic State University), Amr Sabry (Indiana University), Ben Setzer (Kennesaw State University), Carolyn Schauble (Colorado State University), David Scuse (University of Manitoba), Ashraf Shirani (San Jose State University), Daniel Spiegel (Kutztown University), Joslyn A. Smith (Florida Atlantic University), Lixin Tao (Pace University), Ronald F. Taylor (Wright State University), Russ Tront (Simon Fraser University), Deborah Trytten (University of Oklahoma), Michael Verdicchio (Citadel), Kent Vidrine (George Washington University), Bahram Zartoshty (California State University at Northridge)。

能够与 Pearson 出版社一起工作，我感到非常愉快和荣幸。感谢 Tracy Johnson 和她的同事 Marcia Horton、Yez Alayan、Carole Snyder、Scott Disanno、Bob Engelhardt、Haseen Khan，感谢他们组织、开展和积极促进本项目。

一如既往，感谢我妻子 Samantha 的爱、支持和鼓励。

出版者的话	
中文版序	
译者序	
前言	
第1章 计算机、程序和Java概述	1
1.1 引言	1
1.2 什么是计算机	2
1.2.1 中央处理器	2
1.2.2 比特和字节	3
1.2.3 内存	3
1.2.4 存储设备	4
1.2.5 输入和输出设备	4
1.2.6 通信设备	5
1.3 编程语言	6
1.3.1 机器语言	6
1.3.2 汇编语言	6
1.3.3 高级语言	7
1.4 操作系统	8
1.4.1 控制和监视系统的活动	8
1.4.2 分配和调配系统资源	8
1.4.3 调度操作	8
1.5 Java、万维网以及其他	9
1.6 Java语言规范、API、JDK和IDE ..	10
1.7 一个简单的Java程序	11
1.8 创建、编译和执行Java程序	13
1.9 程序设计风格和文档	16
1.9.1 正确的注释和注释风格	16
1.9.2 正确的缩进和空白	16
1.9.3 块的风格	17
1.10 程序设计错误	17
1.10.1 语法错误	17
1.10.2 运行时错误	18
1.10.3 逻辑错误	18
1.10.4 常见错误	19
1.11 使用NetBeans开发Java程序	20
1.11.1 创建Java工程	20
1.11.2 创建Java类	21
1.11.3 编译和运行类	22
1.12 使用Eclipse开发Java程序	22
1.12.1 创建Java工程	22
1.12.2 创建Java类	24
1.12.3 编译和运行类	24
关键术语	25
本章小结	25
测试题	26
编程练习题	26
第2章 基本程序设计	28
2.1 引言	28
2.2 编写简单的程序	28
2.3 从控制台读取输入	31
2.4 标识符	34
2.5 变量	34
2.6 赋值语句和赋值表达式	36
2.7 命名常量	37
2.8 命名习惯	37
2.9 数值数据类型和操作	38
2.9.1 数值类型	38
2.9.2 从键盘读取数值	39
2.9.3 数值操作符	39
2.9.4 幂运算	41
2.10 数值型直接量	41
2.10.1 整型直接量	42
2.10.2 浮点型直接量	42
2.10.3 科学记数法	42
2.11 表达式求值以及操作符优先级 ..	43
2.12 示例学习：显示当前时间	44
2.13 增强赋值操作符	46
2.14 自增和自减操作符	47

2.15 数值类型转换	48	4.3 字符数据类型和操作	105
2.16 软件开发过程	50	4.3.1 Unicode和ASCII码	105
2.17 示例学习: 整钱兑零	54	4.3.2 特殊字符的转义序列	106
2.18 常见错误和陷阱	56	4.3.3 字符型数据与数值型数据 之间的转换	107
关键术语	58	4.3.4 字符的比较和测试	107
本章小结	58	4.4 String类型	109
测试题	59	4.4.1 求字符串长度	110
编程练习题	59	4.4.2 从字符串中获取字符	110
第3章 选择	64	4.4.3 连接字符串	111
3.1 引言	64	4.4.4 字符串的转换	111
3.2 boolean数据类型	64	4.4.5 从控制台读取字符串	112
3.3 if语句	66	4.4.6 从控制台读取字符	112
3.4 双分支if-else语句	68	4.4.7 字符串比较	112
3.5 嵌套的if语句和多分支if-else 语句	69	4.4.8 获得子字符串	114
3.6 常见错误和陷阱	71	4.4.9 获取字符串中的字符或者 子串	115
3.7 产生随机数	74	4.4.10 字符串和数字间的转换	116
3.8 示例学习: 计算身体质量指数	76	4.5 示例学习	117
3.9 示例学习: 计算税率	77	4.5.1 猜测生日	118
3.10 逻辑操作符	80	4.5.2 将十六进制数转换为 十进制数	121
3.11 示例学习: 判定闰年	83	4.5.3 使用字符串修改彩票程序	122
3.12 示例学习: 彩票	84	4.6 格式化控制台输出	123
3.13 switch语句	85	关键术语	126
3.14 条件表达式	88	本章小结	127
3.15 操作符的优先级和结合规则	89	测试题	127
3.16 调试	90	编程练习题	127
关键术语	91	第5章 循环	133
本章小结	91	5.1 引言	133
测试题	92	5.2 while循环	134
编程练习题	92	5.2.1 示例学习: 猜数字	136
第4章 数学函数、字符和字符串	100	5.2.2 循环设计策略	138
4.1 引言	100	5.2.3 示例学习: 多个减法测 试题	138
4.2 常用数学函数	101	5.2.4 使用标记值控制循环	140
4.2.1 三角函数方法	101	5.2.5 输入和输出重定向	141
4.2.2 指数函数方法	102	5.3 do-while循环	143
4.2.3 取整方法	102	5.4 for循环	144
4.2.4 min、max和abs方法	102	5.5 采用哪种循环	147
4.2.5 random方法	103		
4.2.6 示例学习: 计算三角形的 角度	103		

5.6 嵌套循环	149	7.2.2 创建数组	208
5.7 最小化数值错误	151	7.2.3 数组大小和默认值	209
5.8 示例学习	152	7.2.4 访问数组元素	209
5.8.1 求最大公约数	152	7.2.5 数组初始化语法	210
5.8.2 预测未来学费	154	7.2.6 处理数组	210
5.8.3 将十进制数转换为 十六进制数	155	7.2.7 foreach循环	212
5.9 关键字break和continue	156	7.3 示例学习: 分析数字	214
5.10 示例学习: 判断回文串	159	7.4 示例学习: 一副牌	215
5.11 示例学习: 显示素数	160	7.5 数组的复制	217
关键术语	162	7.6 将数组传递给方法	218
本章小结	163	7.7 从方法中返回数组	221
测试题	163	7.8 示例学习: 统计每个字母 出现的次数	221
编程练习题	163	7.9 可变长参数列表	224
第6章 方法	171	7.10 数组的查找	225
6.1 引言	171	7.10.1 线性查找法	225
6.2 定义方法	172	7.10.2 二分查找法	226
6.3 调用方法	173	7.11 数组的排序	228
6.4 void方法示例	175	7.12 Arrays类	230
6.5 通过传值进行参数传递	178	7.13 命令行参数	232
6.6 模块化代码	181	7.13.1 向main方法传递字符串	232
6.7 示例学习: 将十六进制数转换 为十进制数	183	7.13.2 示例学习: 计算器	232
6.8 重载方法	185	关键术语	234
6.9 变量的作用域	187	本章小结	235
6.10 示例学习: 生成随机字符	188	测试题	235
6.11 方法抽象和逐步求精	190	编程练习题	235
6.11.1 自顶向下的设计	191	第8章 多维数组	242
6.11.2 自顶向下和自底向上的 实现	192	8.1 引言	242
6.11.3 实现细节	193	8.2 二维数组的基础知识	242
6.11.4 逐步求精的优势	196	8.2.1 声明二维数组变量并创建 二维数组	243
关键术语	196	8.2.2 获取二维数组的长度	244
本章小结	197	8.2.3 锯齿数组	244
测试题	197	8.3 处理二维数组	245
编程练习题	197	8.4 将二维数组传递给方法	247
第7章 一维数组	207	8.5 示例学习: 多选题测验评分	248
7.1 引言	207	8.6 示例学习: 找出距离最近的 点对	249
7.2 数组的基础知识	207	8.7 示例学习: 数独	251
7.2.1 声明数组变量	208	8.8 多维数组	254

8.8.1 示例学习: 每日温度和湿度	255	10.4.1 关联	316
8.8.2 示例学习: 猜生日	256	10.4.2 聚集和组合	317
本章小结	258	10.5 示例学习: 设计Course类	318
测试题	258	10.6 示例学习: 设计栈类	320
编程练习题	258	10.7 将基本数据类型值作为对象处理	322
第9章 对象和类	270	10.8 基本类型和包装类类型之间的自动转换	325
9.1 引言	270	10.9 BigInteger和BigDecimal类	326
9.2 为对象定义类	270	10.10 String类	327
9.3 示例: 定义类和创建对象	272	10.10.1 构造字符串	327
9.4 使用构造方法构造对象	277	10.10.2 不可变字符串与限定字符串	328
9.5 通过引用变量访问对象	278	10.10.3 字符串的替换和分隔	329
9.5.1 引用变量和引用类型	278	10.10.4 依照模式匹配、替换和分隔	329
9.5.2 访问对象的数据和方法	279	10.10.5 字符串与数组之间的转换	330
9.5.3 引用数据域和null值	279	10.10.6 将字符和数值转换成字符串	331
9.5.4 基本类型变量和引用类型变量的区别	280	10.10.7 格式化字符串	331
9.6 使用Java库中的类	282	10.11 StringBuilder和StringBuffer类	333
9.6.1 Date类	282	10.11.1 修改StringBuilder中的字符串	334
9.6.2 Random类	283	10.11.2 toString、capacity、length、setLength和charAt方法	335
9.6.3 Point2D类	283	10.11.3 示例学习: 判断回文串时忽略既非字母又非数字的字符	336
9.7 静态变量、常量和方法	284	关键术语	338
9.8 可见性修饰符	289	本章小结	339
9.9 数据域封装	291	测试题	339
9.10 向方法传递对象参数	294	编程练习题	339
9.11 对象数组	297	第11章 继承和多态	347
9.12 不可变对象和类	299	11.1 引言	347
9.13 变量的作用域	301	11.2 父类和子类	347
9.14 this引用	302	11.3 使用super关键字	353
9.14.1 使用this引用隐藏数据域	302	11.3.1 调用父类的构造方法	353
9.14.2 使用this调用构造方法	303	11.3.2 构造方法链	354
关键术语	304		
本章小结	304		
测试题	305		
编程练习题	305		
第10章 面向对象思考	309		
10.1 引言	309		
10.2 类的抽象和封装	309		
10.3 面向对象的思考	313		
10.4 类的关系	315		

11.3.3 调用父类的方法	355	12.11.5 示例学习：替换文本	412
11.4 方法重写	356	12.12 从Web上读取数据	414
11.5 方法重写与重载	357	12.13 示例学习：Web爬虫	416
11.6 Object类及其toString()方法	359	关键术语	418
11.7 多态	359	本章小结	418
11.8 动态绑定	360	测试题	419
11.9 对象转换和instanceof运算符	363	编程练习	419
11.10 Object类的equals方法	367	第13章 抽象类和接口	424
11.11 ArrayList类	368	13.1 引言	424
11.12 对于列表有用的方法	374	13.2 抽象类	424
11.13 示例学习：自定义栈类	374	13.2.1 为何要使用抽象方法	427
11.14 protected数据和方法	376	13.2.2 抽象类的几点说明	428
11.15 防止扩展和重写	378	13.3 示例学习：抽象的Number类	429
关键术语	378	13.4 示例学习：Calendar和	
本章小结	379	GregorianCalendar	431
测试题	379	13.5 接口	434
编程练习题	380	13.6 Comparable接口	436
第12章 异常处理和文本I/O	384	13.7 Cloneable接口	440
12.1 引言	384	13.8 接口与抽象类	444
12.2 异常处理概述	385	13.9 示例学习：Rational类	447
12.3 异常类型	389	13.10 类的设计原则	452
12.4 关于异常处理的更多知识	391	13.10.1 内聚性	452
12.4.1 声明异常	392	13.10.2 一致性	452
12.4.2 抛出异常	392	13.10.3 封装性	452
12.4.3 捕获异常	393	13.10.4 清晰性	453
12.4.4 从异常中获取信息	394	13.10.5 完整性	453
12.4.5 示例学习：声明、抛出和		13.10.6 实例和静态	453
捕获异常	396	13.10.7 继承与聚合	454
12.5 finally子句	399	13.10.8 接口和抽象类	454
12.6 何时使用异常	400	关键术语	454
12.7 重新抛出异常	401	本章小结	455
12.8 链式异常	402	测试题	455
12.9 创建自定义异常类	403	编程练习题	455
12.10 File类	405	第14章 JavaFX基础	459
12.11 文件输入和输出	408	14.1 引言	459
12.11.1 使用PrintWriter写数据	408	14.2 JavaFX与Swing以及AWT的	
12.11.2 使用try-with-resources		比较	459
自动关闭资源	409	14.3 JavaFX程序的基本结构	460
12.11.3 使用Scanner读数据	410	14.4 面板、UI组件以及形状	462
12.11.4 Scanner如何工作	411	14.5 属性绑定	465

14.6	节点的通用属性和方法	468	测试题	536	
14.7	Color类	469	编程练习题	536	
14.8	Font类	470	第16章 JavaFX UI组件和多媒体	542	
14.9	Image和ImageView类	472	16.1	引言	542
14.10	布局面板	474	16.2	Labeled和Label	543
14.10.1	FlowPane	475	16.3	按钮	545
14.10.2	GridPane	477	16.4	复选框	547
14.10.3	BorderPane	478	16.5	单选按钮	549
14.10.4	HBox和VBox	480	16.6	文本域	551
14.11	形状	482	16.7	文本区域	553
14.11.1	Text	482	16.8	组合框	556
14.11.2	Line	484	16.9	列表视图	559
14.11.3	Rectangle	485	16.10	滚动条	562
14.11.4	Circle和Ellipse	487	16.11	滑动条	564
14.11.5	Arc	488	16.12	示例学习: 开发一个井字 游戏	567
14.11.6	Polygon和Polyline	491	16.13	视频和音频	572
14.12	示例学习: ClockPane类	493	16.14	示例学习: 国旗和国歌	575
关键术语		497	本章小结	577	
本章小结		498	测试题	578	
测试题		498	编程练习题	578	
编程练习题		498	第17章 二进制 I/O	584	
第15章 事件驱动编程和动画		504	17.1	引言	584
15.1	引言	504	17.2	在Java中如何处理文本I/O	584
15.2	事件和事件源	506	17.3	文本I/O与二进制I/O	585
15.3	注册处理器和处理事件	507	17.4	二进制I/O类	587
15.4	内部类	511	17.4.1	FileInputStream和 FileOutputStream	588
15.5	匿名内部类处理器	512	17.4.2	FilterInputStream和 FilterOutputStream	590
15.6	使用lambda表达式简化事件 处理	514	17.4.3	DataInputStream和 DataOutputStream	590
15.7	示例学习: 贷款计算器	517	17.4.4	BufferedInputStream和 BufferedOutputStream	594
15.8	鼠标事件	519	17.5	示例学习: 复制文件	596
15.9	键盘事件	520	17.6	对象I/O	598
15.10	可观察对象的监听器	523	17.6.1	Serializable接口	600
15.11	动画	525	17.6.2	序列化数组	601
15.11.1	PathTransition	525	17.7	随机访问文件	602
15.11.2	FadeTransition	528	关键术语	606	
15.11.3	Timeline	530			
15.12	示例学习: 弹球	532			
关键术语		535			
本章小结		535			

本章小结	606	18.10 尾递归	628
测试题	606	关键术语	629
编程练习题	606	本章小结	629
第18章 递归	609	测试题	630
18.1 引言	609	编程练习题	630
18.2 示例学习: 计算阶乘	610	附录A Java关键字	637
18.3 示例学习: 计算斐波那契数	613	附录B ASCII字符集	638
18.4 使用递归解决问题	615	附录C 操作符优先级表	639
18.5 递归辅助方法	617	附录D Java修饰符	640
18.5.1 递归选择排序	618	附录E 特殊浮点值	641
18.5.2 递归二分查找	618	附录F 数系	642
18.6 示例学习: 得到目录的大小	619	附录G 位操作	646
18.7 示例学习: 汉诺塔	621	附录H 正则表达式	647
18.8 示例学习: 分形	624	附录I 枚举类型	651
18.9 递归与迭代	627		

计算机、程序和 Java 概述

教学目标

- 理解计算机基础知识、程序和操作系统（1.2 ~ 1.4 节）。
- 阐述 Java 与万维网（World Wide Web）之间的关系（1.5 节）。
- 理解 Java 语言规范、API、JDK 和 IDE 的含义（1.6 节）。
- 编写一个简单的 Java 程序（1.7 节）。
- 在控制台上显示输出（1.7 节）。
- 解释 Java 程序的基本语法（1.7 节）。
- 创建、编译和运行 Java 程序（1.8 节）。
- 使用良好的 Java 程序设计风格和编写正确的程序文档（1.9 节）。
- 解释语法错误、运行时错误和逻辑错误的区别（1.10 节）。
- 使用 NetBeans 开发 Java 程序（1.11 节）。
- 使用 Eclipse 开发 Java 程序（1.12 节）。

1.1 引言

 **要点提示：**本书的主旨是学习如何通过编写程序来解决问题。

本书是关于程序设计（又称编程）的。那么，什么是程序设计呢？程序设计就是创建（或者开发）软件，软件也称为程序。简言之，软件包含了指令，告诉计算机（或者计算设备）做什么。

软件遍布我们的周围，甚至在一些你认为可能不需要软件的设备中。当然，你会希望在个人计算机上找到和使用软件；但软件在运行中的飞机、汽车、手机甚至烤面包机中同样起着作用。在个人计算机上，你会使用字处理程序编写文档，使用 Web 浏览器在互联网中冲浪，使用电子邮件程序收发电子邮件。这些程序都是软件的实例。软件开发人员在称为程序设计语言的强大工具的帮助下创建软件。

本书使用 Java 程序设计语言来教授如何创建程序。程序设计语言有很多种，有些语言已有几十年的历史。每种语言都是为了实现某个特定的目的而发明的，比如，构建在以前语言的长处之上，或者为程序员提供一套全新和独特的工具。当知道有如此多可用的程序设计语言后，你自然会困惑哪种程序设计语言是最好的。但是，事实上，没有“最好”的语言。每种语言有它自己的长处和短处。有经验的程序员知道某种语言可能在某种场景下工作得很好，但是在另外一个场景中可能另外一个语言会更加合适。因此，经验丰富的程序员将尽可能掌握各种不同的程序设计语言，从而利用各种强大的软件开发工具。

如果你掌握了一种程序设计语言，应该会很容易学会其他程序设计语言。关键是学习如何使用程序设计方法来解决实际问题，这是本书的主旨。

我们即将开始一段激动人心的旅程，学习如何进行程序设计。在开始之前，很有必要复习一下计算机基础、程序和操作系统等内容。如果你已经很熟悉 CPU、内存、磁盘、操作系统以及程序设计语言等术语，那么可以跳过 1.2 ~ 1.4 节中对这些内容的回顾。

1.2 什么是计算机

要点提示：计算机是存储和处理数据的电子设备。

计算机包括硬件 (hardware) 和软件 (software) 两部分。一般来说, 硬件包括计算机中可以看得见的物理部分, 而软件提供看不见的指令, 这些指令控制硬件并且使得硬件完成特定的任务。学习一种程序设计语言, 并不一定要了解计算机硬件知识, 但是如果你了解一些硬件知识的话, 它的确可以帮助你更好地理解程序中指令对于计算机及其组成部分的功效。本节介绍计算机硬件组件及其功能。

一台计算机是由以下几个主要的硬件组件构成的 (图 1-1):

- 中央处理器 (CPU)
- 内存 (主存)
- 存储设备 (例如, 磁盘和光盘)
- 输入设备 (例如, 鼠标和键盘)
- 输出设备 (例如, 显示器和打印机)
- 通信设备 (例如, 调制解调器和网卡)



图 1-1 计算机由中央处理器、内存、存储设备、输入设备、输出设备和通信设备组成

这些组件通过一个称为总线 (bus) 的子系统连接。你可以将总线想象成一个连接计算机组件的道路系统, 数据和电信号通过总线在计算机的各个部分之间传输。在个人计算机中, 总线搭建在主板上, 主板是一个连接计算机各个部分的电路板。

1.2.1 中央处理器

中央处理器 (Central Processing Unit, CPU) 是计算机的大脑。它从内存中获取指令, 然后执行这些指令。CPU 通常由两部分组成: 控制单元 (control unit) 和算术/逻辑单元 (arithmetic/logic unit)。控制单元用于控制和协调其他组件的动作。算术/逻辑单元用于完成数值运算 (加法、减法、乘法、除法) 和逻辑运算 (比较)。

现在的 CPU 都是构建在一块小小的硅半导体芯片上, 这块芯片上包含数百万称为晶体管的小电路开关, 用于处理信息。

每台计算机都有一个内部时钟, 该时钟以固定速度发射电子脉冲。这些脉冲用于控制和同步各种操作的步调。时钟速度越快, 在给定时间段内执行的指令就越多。时钟速度的计量单位是赫兹 (hertz, Hz), 1 赫兹相当于每秒 1 个脉冲。20 世纪 90 年代计算机的时钟速度通常是以兆赫 (MHz) 来表示的 (1MHz 就是 100 万 Hz)。随着 CPU 的速度不断提高, 目前计算机的时钟速度通常以千兆赫 (GHz) 来表述。Intel 公司最新处理器的运行速度大约是 3GHz。

最初被开发出来的 CPU 只有一个核 (core)。核是处理器中实现指令读取和执行的部分。

为了提高 CPU 的处理能力，芯片制造厂商现在生产包含多核的 CPU。一个多核 CPU 是一个具有两个或者更多独立核的组件。现在的消费类计算机一般具有两个、三个甚至四个独立的核。相信不久后，具有几十个甚至几百个核的 CPU 将普及。

1.2.2 比特和字节

在讨论内存前，让我们看下信息（数据和程序）是如何存储在计算机中的。

计算机就是一系列的电路开关。每个开关存在两种状态：关（off）和开（on）。简单而言，在计算机中存储信息就是将一系列的开关设置为开或者关。如果电路是开的，它的值是 1。如果电路是关的，它的值是 0。这些 0 和 1 被解释为二进制数字系统中的数，并且将它们称为比特（bit，二进制数）。

计算机中字节（byte）是最小的存储单元。每个字节由 8 个比特构成。像 3 这样的小数字就可以存储在单个字节中。为了存储单个字节放不下的大数字，计算机需要使用几个字节。

各种类型的数据（例如，数字和字符）都被编码为字节序列。程序员不需要关心数据的编码和解码，这些都是系统根据编码模式（schema）来自动完成的。编码模式是一系列的规则，控制计算机将字符、数字和符号翻译成计算机可以实际工作的数据。大多数模式将每个字符翻译成预先确定的一个比特串。例如，在流行的 ASCII 编码模式中，字符 C 是用一个字节 01000011 来表示的。

计算机的存储能力是以字节和多字节来衡量的，如下：

- 千字节（kilobyte, KB）大约是 1000 字节。
- 兆字节（megabyte, MB）大约是 100 万字节。
- 千兆字节（gigabyte, GB）大约是 10 亿字节。
- 万亿字节（terabyte, TB）大约是 1 万亿字节。

一页 Word 文档可能有 20KB。因此，1MB 可以存储 50 页的文档，1GB 可以存储 50 000 页的文档。一部两小时的高清电影可能有 8GB，因此将需要 160GB 来存储 20 部电影。

1.2.3 内存

计算机的内存由一个有序的字节序列组成，用于存储程序及程序需要的数据。你可以将内存想象成计算机执行程序的工作区域。一个程序和它的数据在被 CPU 执行前必须移到计算机的内存中。

每个字节都有一个唯一的地址，如图 1-2 所示。使用这个地址确定字节的位置，以便于存储和获取数据。因为可以按任意顺序存取字节，所以内存也被称为随机访问存储器（Random-Access Memory, RAM）。

现在的个人计算机通常至少有 4GB 的 RAM，但是它们一般装有 6 ~ 8GB 的内存。通常而言，一个计算机具有的 RAM 越多，它的运行速度越快，但是这条简单的经验法则是有限制的。

内存中字节的內容永远非空，但是它的原始内容可能对于你的程序来说是毫无意义的。一旦新的信息被放

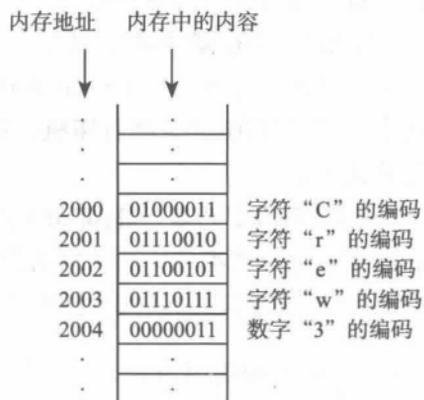


图 1-2 内存以唯一编码的内存位置来存储数据和程序指令

入内存，该字节的当前内容就会丢失。

同 CPU 一样，内存也是构建在一个表面上嵌有数百万晶体管的硅半导体芯片上。与 CPU 芯片相比，内存芯片更简单、更低速，也更便宜。

1.2.4 存储设备

计算机的内存 (RAM) 是一种易失的数据保存形式：断电时存储在内存中的信息就会丢失。程序和数据被永久地存放在存储设备上，当计算机确实要使用它们时再移入内存，因为从内存读取比从存储设备读取要快得多。

存储设备主要有以下三种类型：

- 磁盘驱动器
- 光盘驱动器 (CD 和 DVD)
- USB 闪存驱动器

驱动器 (drive) 是对存储介质进行操作的设备，例如，磁盘和光盘。存储介质物理地存储数据和程序指令。驱动器从介质读取数据并将数据写在介质上。

1. 磁盘

每台计算机至少有一个硬盘驱动器。硬盘 (hard disk) 用于永久地存储数据和程序。在较新的个人计算机上，硬盘容量一般为 500GB 到 1TB。磁盘驱动器通常安装在计算机内。此外，还有移动硬盘。

2. 光盘和数字化视频磁盘

CD 的全称是致密的盘片 (compact disc)。光盘驱动器的类型有两种：只读光盘 (CD-R) 和可读写光盘 (CD-RW)。只读光盘上的信息只能用于读取，内容一旦记录到光盘上，用户是不能修改它们的。可读写光盘可以像硬盘一样使用。也就是说，可以将数据写到光盘上，然后用新的数据覆盖掉这些数据。单张光盘的容量可以达到 700MB。大多数新型的个人计算机都安装了可读写光驱，它既支持只读光盘也支持可读写光盘。

DVD 的全称是数字化多功能碟片或者是数字化视频磁盘。DVD 和 CD 看起来很像，可以使用任意一种来存储数据。一张 DVD 上可以保存的信息要比一张 CD 上可以保存的信息多。一张标准 DVD 的存储容量是 4.7GB。如同 CD 一样，有两种类型的 DVD：DVD-R (只读) 和 DVD-RW (可重写)。

3. USB 闪存驱动器

通用串行总线 (Universal Serial Bus, USB) 接口允许用户将多种外部设备连接到计算机上。可以使用 USB 将打印机、数码相机、鼠标、外部硬盘驱动器，以及其他设备连接到计算机上。

USB 闪存驱动器 (flash drive) 是用于存储和传输数据的设备。闪存驱动器很小——大约就是一包口香糖的大小。它就像移动硬盘一样，可以插入计算机上的 USB 端口。USB 闪存驱动器目前可用的最大存储容量为 256GB。

1.2.5 输入和输出设备

输入设备和输出设备让用户可以和计算机进行通信。最常用的输入设备是键盘 (keyboard) 和鼠标 (mouse)，而最常用的输出设备是显示器 (monitor) 和打印机 (printer)。

1. 键盘

键盘是用于输入的设备。有一种便携式键盘，不带数字小键盘。

功能键 (function key) 位于键盘的最上边，而且都是以 F 为前缀。它们的功能取决于当前所使用的软件。

修饰符键 (modifier key) 是特殊键 (例如, Shift、Alt 和 Ctrl)，当它和另一个键同时按下时，会改变另一个键的常用功能。

数字小键盘 (numeric keypad) 位于键盘的右下角，是一套独立的类似计算器风格的按键集合，用于快速输入数字。

方向键 (arrow key) 位于主键盘和数字小键盘之间，在各种程序中用于上下左右地移动光标。

插入键 (Insert)、删除键 (Delete)、向上翻页键 (Page Up) 和向下翻页键 (Page Down) 分别用于在字处理和其他程序中完成插入文本和对象、删除文本和对象以及向上和向下翻页的功能。

2. 鼠标

鼠标 (mouse) 是定点设备，用来在屏幕上移动一个称为光标的图形化的指针 (通常以一个箭头的形状)，或者用于单击屏幕上的对象 (如一个按钮) 来触发它以执行动作。

3. 显示器

显示器 (monitor) 显示信息 (文本和图形)。屏幕分辨率和点距决定显示的质量。

屏幕分辨率 (screen resolution) 是指显示设备水平和垂直维度上的像素数。像素 (“图像元素”的简称) 就是构成屏幕上图像的小点。比如，对于一个 17 英寸的屏幕，分辨率一般为宽 1024 像素、高 768 像素。分辨率可以手工设置。分辨率越高，图像越锐化、越清晰。

点距 (dot pitch) 是指像素之间以毫米为单位的距离。点距越小，显示效果越好。

1.2.6 通信设备

计算机可以通过通信设备进行联网，例如，拨号调制解调器 (modulator/demodulator, 调制器 / 解调器)、DSL、电缆调制解调器、有线网络接口卡，或者无线适配器。

- 拨号调制解调器使用的是电话线，传输数据的速度可以高达 56 000bps (bps 表示每秒比特)。
- DSL (Digital Subscriber Line, 数字用户线) 使用的也是标准电话线，但是传输数据的速度比标准拨号调制解调器快 20 倍。
- 电缆调制解调器利用电缆公司维护的有线电视电缆进行数据传输，通常速度比 DSL 快。
- 网络接口卡 (NIC) 是将计算机接入局域网 (LAN) 的设备。局域网通常用于大学、商业组织和政府组织。一种称为 1000BaseT 的高速 NIC 能够以每秒 1000Mbps (Mbps 表示每秒百万比特) 的速度传输数据。
- 无线网络现在在家庭、商业和学校中极其流行。现在，每台笔记本电脑都配有无线适配器，计算机可以通过无线适配器连接到局域网和 Internet 上。

 **注意：**复习题问题的答案在配套网站上。

复习题

1.1 什么是硬件和软件？

- 1.2 列举计算机的5个主要硬件组件。
- 1.3 缩写“CPU”代表什么含义?
- 1.4 衡量CPU速度的单位是什么?
- 1.5 什么是比特? 什么是字节?
- 1.6 内存是用来做什么的? RAM代表什么? 为什么内存称为RAM?
- 1.7 用于衡量内存大小的单位是什么?
- 1.8 用于衡量磁盘大小的单位是什么?
- 1.9 内存和永久存储设备的主要不同是什么?

1.3 编程语言

要点提示: 计算机程序 (program) 称为软件 (software), 是告诉计算机该做什么的指令。

计算机不理解人类的语言, 所以, 计算机程序必须使用计算机可以使用的语言编写。现在有数百种编程语言, 对人们来说, 开发它们使编程过程更容易。但是, 所有的程序都必须转换成计算机可以执行的指令。

1.3.1 机器语言

计算机的原生语言因计算机类型的不同而有差异, 计算机的原生语言就是机器语言 (machine language), 即一套内嵌的原子指令集。因为这些指令都是以二进制代码的形式存在, 所以, 为了以机器原生语言的形式给计算机指令, 必须以二进制代码输入指令。例如, 为进行两数的相加, 可能必须写成如下的二进制形式:

```
1101101010011010
```

1.3.2 汇编语言

用机器语言进行程序设计是非常单调乏味的过程, 而且, 所编的程序也非常难以读懂和修改。为此, 在计算的早期就创建了汇编语言, 作为机器语言的替代品。汇编语言 (assembly language) 使用短的描述性单词 (称为助记符) 来表示每一条机器语言指令。例如, 助记符 add 一般表示数字相加, sub 表示数字相减。将数字 2 和数字 3 相加得到结果, 可以编写如下汇编代码:

```
add 2, 3, result
```

汇编语言的出现降低了程序设计的难度。然而, 由于计算机不理解汇编语言, 所以需要一种称为汇编器 (assembler) 的程序将汇编语言程序转换为机器代码, 如图 1-3 所示。

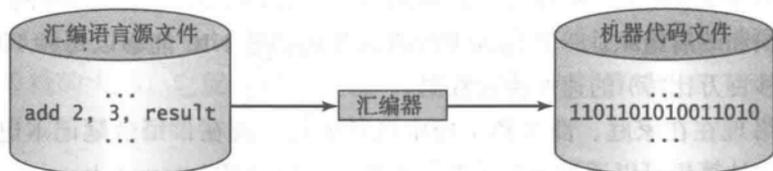


图 1-3 汇编器将汇编语言指令转换为机器代码

使用汇编语言编写代码比使用机器语言容易。然而, 用汇编语言编写代码依然很不方便。汇编语言中的一条指令对应机器代码中的一条指令。用汇编语言写代码需要知道 CPU

是如何工作的。汇编语言被认为是低级语言，因为汇编语言本质上非常接近机器语言，并且是机器相关的。

1.3.3 高级语言

20 世纪 50 年代，新一代编程语言即众所周知的高级语言出现了。它们是平台独立的，这意味着可以使用高级语言编程，然后在各种不同类型的机器上运行。高级语言很像英语，易于学习和使用。高级语言中的指令称为语句。例如，下面是计算半径为 5 的圆面积的高级语言语句：

```
area = 5 * 5 * 3.14159;
```

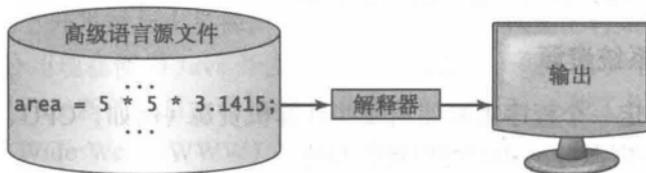
有许多高级编程语言，每种都为特定目的而设计。表 1-1 列出了一些流行的高级编程语言。

表 1-1 流行的高级编程语言

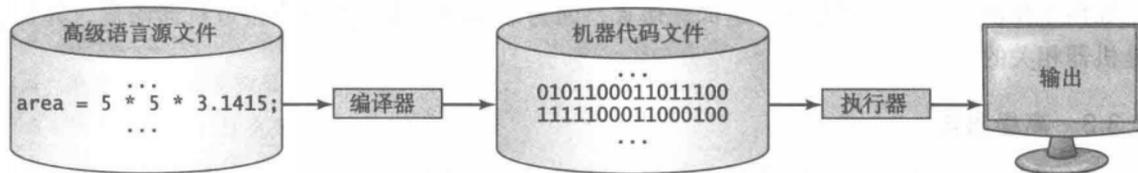
语言	描述
Ada	以 Ada Lovelace (她研究机械式的通用目的的计算机) 命名, Ada 是为美国国防部开发的, 主要用于国防项目
BASIC	初学者通用符号指令代码, 是为了让初学者易学易用而设计的
C	由贝尔实验室开发, C 语言具有汇编语言的强大功能以及高级语言的易学性和可移植性
C++	基于 C 语言开发, 是一种面向对象程序设计语言
C#	读为“C Sharp”, 由 Microsoft 公司开发的混合了 Java 和 C++ 特征的语言
COBOL	面向商业的通用语言, 是为商业应用而设计的
FORTRAN	公式翻译, 广泛用于科学和数学应用
Java	由 Sun 公司 (现在属于 Oracle) 开发, 广泛用于开发一些独立于平台的互联网应用程序
Pascal	以 Blaise Pascal (Blaise Pascal 是 17 世纪计算机器的先驱) 命名, Pascal 是一个简单的、结构化的、通用目的的语言, 主要用于编程教学
Python	一种简单的通用目的脚本语言, 适合编写小程序
Visual Basic	由 Microsoft 公司开发, 方便编程人员快速开发图形用户界面

用高级语言编写的程序称为源程序 (source program) 或源代码 (source code)。由于计算机不能运行源程序, 源程序必须被翻译成可执行的机器代码。翻译可以由另外一种称为解释器或者编译器的编程工具来完成。

- 解释器从源代码中读取一条语句, 将其翻译为机器代码或者虚拟机器代码, 然后立刻运行, 如图 1-4a 所示。请注意来自源代码的一条语句可能被翻译为多条机器指令。
- 编译器将整个源代码翻译为机器代码文件, 然后执行该机器代码文件, 如图 1-4b 所示。



a) 解释器一次翻译并且执行程序的一条语句



b) 编译器将整个源程序翻译为机器语言文件以运行

图 1-4 (续)

复习题

- 1.10 CPU 能理解什么语言?
- 1.11 什么是汇编语言?
- 1.12 什么是汇编器?
- 1.13 什么是高级编程语言?
- 1.14 什么是源程序?
- 1.15 什么是解释器?
- 1.16 什么是编译器?
- 1.17 解释语言和编译语言之间的区别是什么?

1.4 操作系统

要点提示: 操作系统 (Operating System, OS) 是运行在计算机上的最重要的程序, 它可以管理和控制计算机的活动。

流行的操作系统有 Microsoft Windows、Mac OS 以及 Linux。如果没有在计算机上安装和运行操作系统, 像 Web 浏览器或者字处理程序这样的应用程序就不能运行。硬件、操作系统、应用软件和用户之间的关系如图 1-5 所示。

操作系统的主要任务有:

- 控制和监视系统的活动
- 分配和调配系统资源
- 调度操作

1.4.1 控制和监视系统的活动

操作系统执行基本的任务, 例如, 识别来自键盘的输入, 向显示器发送输出结果, 跟踪存储设备中的文件和文件夹的动态, 控制类似硬盘驱动器和打印机这样的外部设备。操作系统还要确保不同的程序和用户同时使用计算机时不会相互干扰。另外, 操作系统还负责安全处理, 以确保未经授权的用户和程序无权访问系统。

1.4.2 分配和调配系统资源

操作系统负责确定一个程序需要使用哪些计算机资源 (例如, CPU、内存、磁盘、输入和输出设备), 并进行资源分配和调配以运行程序。

1.4.3 调度操作

操作系统负责调度程序的活动, 以便有效地利用系统资源。为了提高系统的性能, 目前

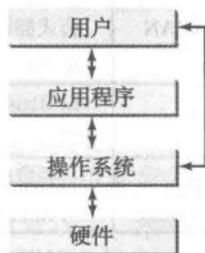


图 1-5 用户和应用程序通过操作系统访问计算机的硬件

许多操作系统都支持像多道程序设计 (multiprogramming)、多线程 (multithreading) 和多处理 (multiprocessing) 这样的技术。

多道程序设计允许多个程序通过共享 CPU 同时运行。CPU 的速度比其他组件快得多, 这样, 多数时间它都处于空闲状态, 例如, 在等待数据从磁盘或其他资源传入, 或者其他系统资源响应时。多道程序设计操作系统利用这一特点, 允许多个程序同时使用 CPU, 一旦 CPU 空闲就让别的程序使用它。例如, 在 Web 浏览器下载文件的同时, 可以用字处理程序来编辑文件。

多线程允许单个程序同时执行多个任务。例如, 字处理程序允许用户在编辑文本的同时, 将其保存到文件。在这个例子中, 编辑和保存是同一个应用程序的两个不同任务, 这两个任务可能并发运行。

多处理也称为并行处理 (parallel processing), 是指使用两个或多个处理器共同并行执行子任务, 然后将子任务的结果合并以得到整个任务的结果。它就像在外科手术中多名医生同时给一个病人做手术一样。

复习题

- 1.18 什么是操作系统? 列出一些流行的操作系统。
- 1.19 操作系统的主要任务是什么?
- 1.20 什么是多道程序设计、多线程以及多处理?

1.5 Java、万维网以及其他

要点提示: Java 是一种功能强大和多用途的编程语言, 可用于开发运行在移动设备、台式计算机以及服务器端的软件。

本书介绍 Java 程序设计。Java 是由 James Gosling 在 Sun 公司领导的小组开发的。(2010 年 Sun 公司被 Oracle 收购。) Java 最初被称为 Oak (橡树), 是 1991 年为消费类电子产品的嵌入式芯片而设计的。1995 年更名为 Java, 并重新设计用于开发 Web 应用程序。关于 Java 的历史, 参见 www.java.com/en/javahistory/index.jsp。

Java 已极其流行。Java 的快速发展以及被广泛接受都应归功于它的设计特性, 特别是它的承诺: 一次编写, 任何地方都可以运行。就像它的设计者声称的, Java 是简单的 (simple)、面向对象的 (object oriented)、分布式的 (distributed)、解释型的 (interpreted)、健壮的 (robust)、安全的 (secure)、体系结构中立的 (architecture neutral)、可移植的 (portable)、高性能的 (high performance)、多线程的 (multithreaded) 和动态的 (dynamic)。关于 Java 特性的剖析, 参见 www.cs.armstrong.edu/liang/JavaCharacteristics.pdf。

Java 是功能完善的通用程序设计语言, 可以用来开发健壮的任务关键的应用程序。现在, 它不仅用于 Web 程序设计, 而且用于在服务器、台式计算机和移动设备上开发跨平台的独立应用程序。用它开发过与火星探测器通信并控制其在火星上行走的代码。许多曾经认为 Java 言过其实的公司现在使用 Java 开发分布式应用程序, 便于客户和合作伙伴在 Internet 上访问。现在, 一旦开发新的项目, 公司都会考虑如何利用 Java 使工作变得更加容易。

万维网 (World Wide Web, WWW) 是从世界上任何地方的 Internet 都可以访问的电子信息宝库。Internet 作为万维网的基础架构已经问世四十多年。丰富多彩的万维网和设计精良的 Web 浏览器是 Internet 流行的主要原因。

Java 一开始富有吸引力是因为 Java 程序可以在 Web 浏览器中运行。这种能在 Web 浏览

器中运行的 Java 程序称为 Java 小程序 (applet)。applet 使用现代的图形用户界面与 Web 用户进行交互, 处理用户的要求, 界面中包括按钮、文本字段、文本域、单选按钮等。applet 使得 Web 更加具有响应性、交互性和趣味性。applet 内嵌在 HTML 文件中。HTML (Hypertext Markup Language) 是一种简单的脚本语言, 用于对文档布局, 链接因特网上的文档, 并且能够在万维网上提供生动的图像、声音和视频。现在, 你可以使用 Java 开发富因特网应用 (RIA)。富因特网应用作为一种 Web 应用, 被设计为可以提供通常桌面应用才具有的特性和功能。

现在, Java 广泛用于开发服务器端的应用程序。这些应用程序处理数据、执行计算, 并生成动态网页。许多商用网站后端都是采用 Java 进行开发的。

Java 是一个功能强大的程序设计语言, 可以用它来开发台式计算机、服务器以及小的手持设备上的应用程序。用于安卓手机的软件也是采用 Java 进行开发的。

复习题

- 1.21 Java 是由谁发明的? 哪个公司现在拥有 Java?
- 1.22 什么是 Java applet?
- 1.23 安卓使用的是什么编程语言?

1.6 Java 语言规范、API、JDK 和 IDE

要点提示: Java 语言规范定义了 Java 的语法, Java 库则在 Java API 中定义。JDK 是用于开发和运行 Java 程序的软件。IDE 是快速开发程序的集成开发环境。

计算机语言有严格的使用规范。如果编写程序时没有遵循这些规范, 计算机就不能理解程序。Java 语言规范和 Java API 定义 Java 的标准。

Java 语言规范 (Java language specification) 是对语言的技术定义, 包括 Java 程序设计语言的语法和语义。完整的 Java 语言规范可以在 <http://docs.oracle.com/javase/specs/> 上找到。

应用程序接口 (Application Program Interface, API) 也称为库, 包括为开发 Java 程序而预定义的类和接口。API 仍然在扩展, 在网站 <http://download.java.net/jdk8/docs/api/> 上, 可以查看和下载最新版的 Java API。

Java 是一个全面且功能强大的语言, 可用于多种用途。Java 有三个版本:

- Java 标准版 (Java Standard Edition, Java SE) 可以用来开发客户端的应用程序。应用程序可以独立运行或作为 applet 在 Web 浏览器中运行。
 - Java 企业版 (Java Enterprise Edition, Java EE) 可以用来开发服务器端的应用程序, 例如, Java servlet 和 Java Server Pages (JSP), 以及 Java Server Faces (JSF)。
 - Java 微型版 (Java Micro Edition, Java ME) 用来开发移动设备的应用程序, 例如手机。
- 本书使用 Java SE 介绍 Java 程序设计。Java SE 是基础, 其他 Java 技术都基于 Java SE。Java SE 也有很多版本, 本书采用最新的版本 Java SE 8。Oracle 发布 Java 的各个版本都带有 Java 开发工具包 (Java Development Toolkit, JDK)。Java SE 8 对应的 Java 开发工具包称为 JDK 1.8 (也称为 Java 8 或者 JDK 8)。

JDK 是由一套独立程序构成的集合, 每个程序都是从命令行调用的, 用于开发和测试 Java 程序。除了 JDK, 还可以使用某种 Java 开发工具 (例如, NetBeans、Eclipse 和 TextPad) —— 为了快速开发 Java 程序而提供集成开发环境 (Integrated Development Environment, IDE) 的软件。编辑、编译、链接、调试和在线帮助都集成在一个图形用户界

面中，这样，只需在一个窗口中输入源代码或在窗口中打开已有的文件，然后单击按钮、菜单选项或者使用功能键就可以编译和运行源代码。

☛ 复习题

- 1.24 什么是 Java 语言规范？
- 1.25 JDK 代表什么？
- 1.26 IDE 代表什么？
- 1.27 类似 NetBeans 和 Eclipse 的工具是和 Java 不同的语言吗？或者它们是 Java 的方言或者扩充？

1.7 一个简单的 Java 程序

🔑 要点提示：Java 是从类中的 main 方法开始执行的。

我们从一个简单的 Java 程序开始，该程序在控制台上显示消息“Welcome to Java!”。控制台 (console) 是一个老的计算机词汇，指计算机的文本输入和显示设备。控制台输入是指从键盘上接收输入，而控制台输出是指在显示器上显示输出。该程序如程序清单 1-1 所示。

程序清单 1-1 Welcome.java

```
1 public class Welcome {
2     public static void main(String[] args) {
3         // Display message Welcome to Java! on the console
4         System.out.println("Welcome to Java!");
5     }
6 }
```

Welcome to Java!

请注意，显示行号 (line number) 是为了引用方便，它们并不是程序的一部分。所以，不要在程序中敲入行号。

第 1 行定义了一个类。每个 Java 程序至少应该有一个类。每个类都有一个名字。按照惯例，类名都是以大写字母开头的。本例中，类名 (class name) 为 Welcome。

第 2 行定义主方法 (main method)。程序是从 main 方法开始执行的。一个类可以包含几个方法。main 方法是程序开始执行的入口。

方法是包含语句的结构体。本程序中的 main 方法包括了 System.out.println 语句。该语句在控制台上打印消息“Welcome to Java!”(第 4 行)。字符串 (string) 是一个编程术语，表示一个字符序列。一个字符串必须放入双引号中。Java 中的每条语句都以分号 (;) 结束，也称为语句结束符 (statement terminator)。

保留字 (reserved word) 或关键字 (keyword) 对编译器而言都是有特定含义的，所以不能在程序中用于其他目的。例如，当编译器看到字 class 时，它知道 class 后面的字就是这个类的名字。这个程序中的其他保留字还有 public、static 和 void。

第 3 行是注释 (comment)，它标注该程序是干什么的，以及它是如何构建的。注释帮助程序员进行相互沟通以及理解程序。注释不是程序设计语句，所以编译器编译程序时是忽略注释的。在 Java 中，在单行上用两个斜杠 (//) 引导注释，称为行注释 (line comment)；在一行或多行用 /* 和 */ 括住注释，称为块注释 (block comment)。当编译器看到 // 时，就会忽略本行 // 之后的所有文本。当看到 /* 时，它会搜索接下来的 */，并忽略掉 /* 与 */ 之间的文本。下面是这两种注释的例子：

```
// This application program displays Welcome to Java!
/* This application program displays Welcome to Java! */
/* This application program
   displays Welcome to Java! */
```

程序中的一对花括号将程序的一些组成部分组合起来，形成一个块 (block)。在 Java 中，每个块以左花括号 ({) 开始，以右花括号 (}) 结束。每个类都有一个将该类的数据和放在一块的类块 (class block)。每个方法都有一个将该方法中的语句放在一起的方法块 (method block)。块是可以嵌套的，即一个块可以放到另一个块内，如下面代码所示。

```
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

提示：一个左括号必须匹配一个右括号。任何时候，当输入一个左括号时，应该立即输入一个右括号来防止出现遗漏括号的错误。大多数 Java IDE 都会自动地为每个左括号插入一个右括号。

警告：Java 源程序是区分大小写的。如果在程序中将 main 替换成 Main，就会出错。

在这个程序中你可以注意到有些特殊的字符 (比如， { }、 //、 ;)，它们几乎在每个程序中都会使用。表 1-2 总结了它们的用途。

表 1-2 特殊字符

字符	名称	描述
{ }	左花括号和右花括号	表示一个包含语句的块
()	左圆括号和右圆括号	和方法一起使用
[]	左方括号和右方括号	表示一个数组
//	双斜杠	表示后面是一行注释
" "	左引用符号和右引用符号	包含一个字符串 (即一系列的字符)
;	分号	标识一个语句的结束

学习编程时最容易犯的错是语法错误。像其他任何一种程序设计语言一样，Java 也有自己的语法，而且你必须按照语法规则编写代码。如果你的程序违反了语法规则，例如，忘记了分号、忘记了花括号、忘记了引号，或者拼错了单词，Java 编译器会报告语法错误。可以尝试编译带有这些错误的程序，看看编译器会报告些什么。

注意：你可能想知道为什么 main 方法要以这样的方式定义，为什么使用 System.out.println(...) 在控制台上显示信息。在现阶段，你只需知道它们就是这么做的就可以。这一问题将在后续的章节中得到完整的解答。

程序清单 1-1 中的程序会显示一条信息。一旦你理解了 this 程序，很容易将该程序扩展为显示更多的信息。例如，可以改写该程序来显示三条信息，如程序清单 1-2 所示。

程序清单 1-2 WelcomeWithThreeMessages.java

```
1 public class WelcomeWithThreeMessages {
2     public static void main(String[] args) {
3         System.out.println("Programming is fun!");
4         System.out.println("Fundamentals First");
5         System.out.println("Problem Driven");
6     }
7 }
```

```
Programming is fun!  
Fundamentals First  
Problem Driven
```

更进一步，你可以进行数学计算，并将结果显示到控制台上。程序清单 1-3 给出一个计算 $\frac{10.5+2\times 3}{45-3.5}$ 的例子。

程序清单 1-3 ComputeExpression.java

```
1 public class ComputeExpression {  
2     public static void main(String[] args) {  
3         System.out.println((10.5 + 2 * 3) / (45 - 3.5));  
4     }  
5 }
```

```
0.39759036144578314
```

Java 中的乘法操作符是 *。如你所看到的，将一个数学表达式翻译成 Java 表达式是一个非常直观的过程，我们将在第 2 章进一步讨论 Java 表达式。

复习题

- 1.28 什么是关键字？列举一些 Java 关键字。
- 1.29 Java 是大小写敏感的吗？Java 关键字是大写还是小写？
- 1.30 什么是注释？编译器会忽略注释吗？如何标识一行注释以及一段注释？
- 1.31 在控制台上显示一个字符串的语句是什么？
- 1.32 给出以下代码的输出：

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("3.5 * 4 / 2 - 2.5 is ");  
        System.out.println(3.5 * 4 / 2 - 2.5);  
    }  
}
```

1.8 创建、编译和执行 Java 程序

 **要点提示：**Java 源程序保存为 .java 文件，编译为 .class 文件。.class 文件由 Java 虚拟机 (JVM) 执行。

在执行程序之前，必须创建程序并进行编译。这个过程是反复执行的，如图 1-6 所示。如果程序有编译错误，必须修改程序来纠正错误，然后重新编译它。如果程序有运行时错误或者不能产生正确的结果，必须修改这个程序，重新编译，然后重新执行。

可以使用任何一个文本编辑器或者集成开发环境来创建和编辑 Java 源代码文件。本节演示如何从命令窗口创建、编译和运行 Java 程序。1.10 节和 1.11 节将介绍使用 NetBeans 和 Eclipse 来开发 Java 程序。从命令窗口，可以使用文本编辑器比如记事本 (NotePad) 来创建 Java 源代码文件，如图 1-7 所示。

 **注意：**源文件的扩展名必须是 .java，而且文件名必须与公共类名完全相同。例如，程序清单 1-1 中源代码的文件必须命名为 Welcome.java，因为公共类的类名就是 Welcome。

Java 编译器将 Java 源文件翻译成 Java 字节码文件。下面的命令就是用来编译 Welcome.java 的：

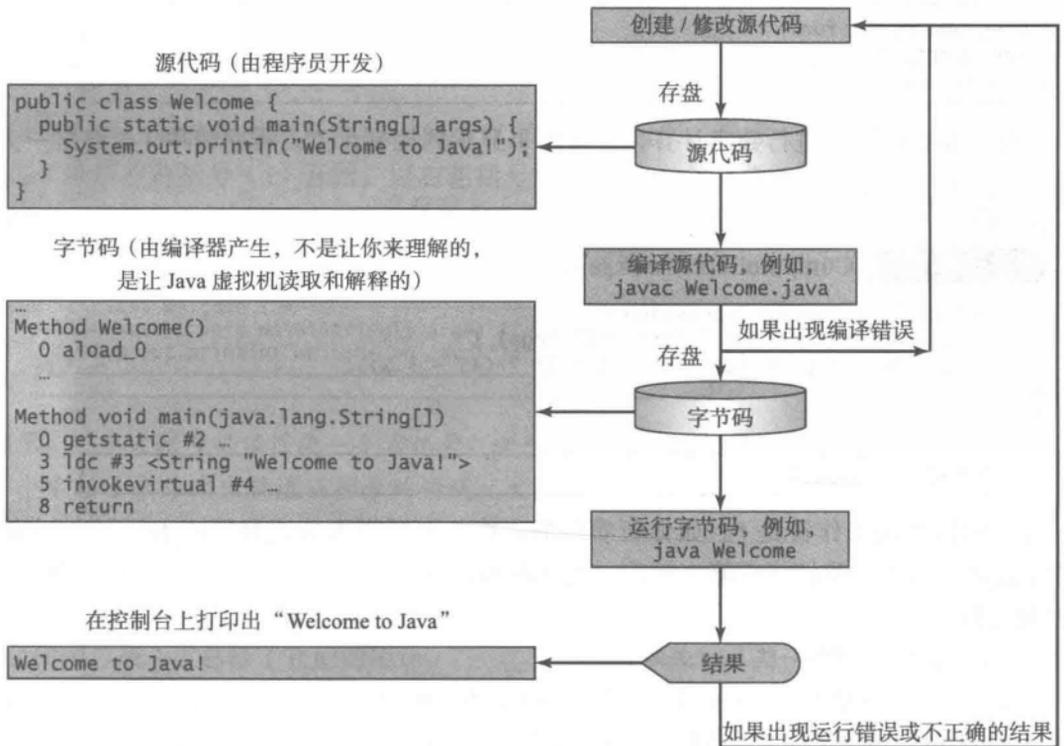


图 1-6 Java 程序开发过程就是重复地创建/修改源代码、编译和执行程序

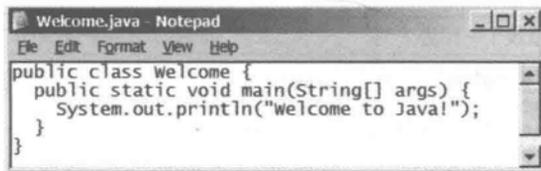


图 1-7 可以使用 Windows 记事本创建 Java 源程序文件

javac Welcome.java

注意: 在编译和运行程序前必须先安装和配置 JDK。补充材料 I.B 介绍如何安装 JDK 8 以及如何设置 Java 程序的编译和运行环境。如果你在编译和运行 Java 的过程中遇到问题, 请参考补充材料 I.C, 这个补充材料还解释了如何使用基本的 DOS 命令, 以及如何使用 Windows 记事本来创建和编辑文件。所有补充材料都在本书配套网站 www.cs.armstrong.edu/liang/intro10e/supplement.html 上。

如果没有语法错误, 编译器 (compiler) 就会生成一个扩展名为 .class 的字节码文件。所以, 前面的命令会生成一个名为 Welcome.class 的文件, 如图 1-8a 所示。Java 语言是高级语言, 而 Java 字节码是低级语言。字节码类似于机器指令, 但它是体系结构中立的, 是在任何带 Java 虚拟机 (JVM) 的平台上运行的, 如图 1-8b 所示。虚拟机不是物理机器, 而是一个解释 Java 字节码的程序。这正是 Java 的主要优点之一: Java 字节码可以在不同的硬件平台和操作系统上运行。Java 源代码编译成 Java 字节码, 然后 Java 字节码被 JVM 解释执行。你的 Java 代码可能要用到 Java 库中的代码。JVM 将执行你的程序代码以及库中的代码。

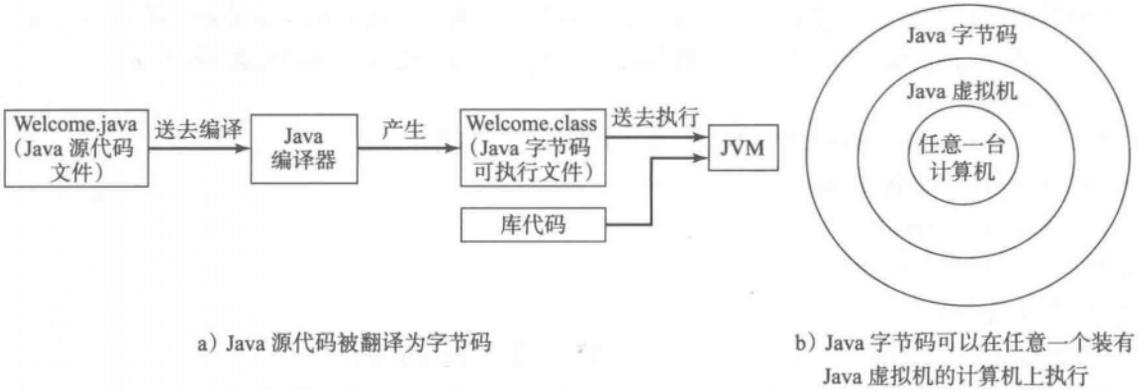


图 1-8

执行 Java 程序就是运行程序的字节码，可以在任何一个装有 JVM 的平台上运行字节码，解释 Java 字节码。解释的过程就是一次将字节码中单独的一步翻译为目标机器语言代码，而不是将整个程序翻译成单独的一块。翻译完一步之后就立即执行这一步。

下述命令用来运行程序清单 1-1 中的字节码：

```
java Welcome
```

图 1-9 显示了用于编译 Welcome.java 的命令 javac。编译器生成 Welcome.class 文件，使用命令 java 执行这个文件。

注意：为了简单性和一致性，除非特别指明，否则所有的源代码和类文件都放在 c:\book 下。

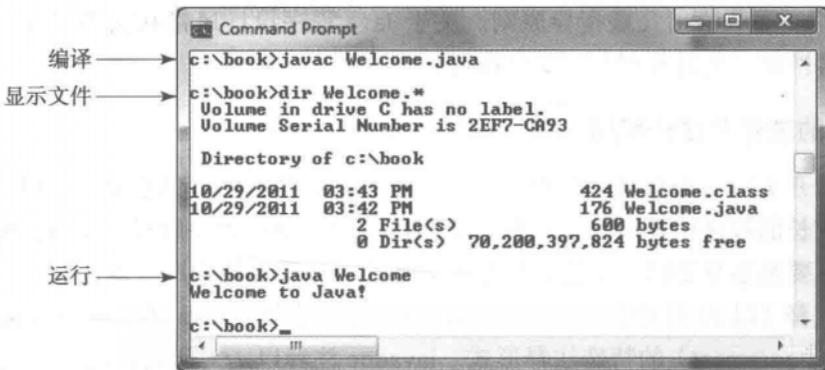


图 1-9 程序清单 1-1 的输出显示消息 “Welcome to Java!”

警告：在命令行执行程序时，不要使用扩展名 .class。使用 java ClassName 来运行程序。如果在命令行使用 java ClassName.class，系统就会尝试读取 ClassName.class.class。

提示：如果要运行一个不存在的类，就会出现 NoClassDefFoundError 的错误。如果执行的类文件中没有 main 方法或敲错了 main 方法（例如，将 main 错敲成 Main），则会出现提示 NoSuchMethodError。

注意：在执行一个 Java 程序时，JVM 首先会用一个称为类加载器（class loader）的程序将类的字节码加载到内存中。如果你的程序中使用其他类，类加载程序会在需要它们之前动态地加载它们。当加载该类后，JVM 使用一个称为字节码验证器（bytecode verifier）的程序来检验字节码的合法性，确保字节码不会违反 Java 的安全规范。Java 强制执行严格的安全规范，以确保来自网络的 Java 程序不会篡改和危害你的计算机。

🔑 **教学提示：**教师可能需要学生使用包来组织程序。例如，你可能将本章的所有程序都放在一个名为 `chapter1` 的包里。要得到如何使用包的指南，请参考教材补充材料 I.F。

🔑 复习题

- 1.33 什么是 Java 源程序的文件后缀名，什么是 Java 字节码文件后缀？
- 1.34 Java 编译器的输入和输出是什么？
- 1.35 编译 Java 程序的命令是什么？
- 1.36 运行 Java 程序的命令是什么？
- 1.37 什么是 JVM？
- 1.38 Java 可以运行在任何机器上吗？在一台计算机上运行 Java 需要什么？
- 1.39 如果运行程序的时候出现 `NoClassDefFoundError` 错误，是什么原因导致了这个错误？
- 1.40 如果运行程序的时候出现 `NoSuchMethodError` 错误，是什么原因导致了这个错误？

1.9 程序设计风格和文档

🔑 **要点提示：**良好的程序设计风格和正确的文档使程序更易阅读，并且能帮助程序员避免错误。

程序设计风格 (programming style) 决定程序的外观。如果把整个程序写在一行，它也会被正确地编译和运行，但是这是非常不好的程序设计风格，因为程序的可读性很差。文档 (documentation) 是关于程序的解释性评注和注释的一个结构体。程序设计风格与文档和编写代码的作用一样重要。良好的程序设计风格和适当的文档可以减少出错的机率，并且提高程序的可读性。本节给出几条指导原则。关于 Java 程序设计风格和文档更详细的指南，可以在本书配套网站上的补充材料 I.D 中找到。

1.9.1 正确的注释和注释风格

在程序的开头写一个总结，解释一下这个程序是做什么的、其主要特点以及所用到的独特技术。在较长的程序中还要加上注释，介绍每一个主要步骤并解释每个难以读懂之处。注释写得简明扼要是很重要的，不能让整个程序都充满注释而使程序很难读懂。

除了行注释 (以 `//` 开始) 和块注释 (以 `/*` 开始) 之外，Java 还支持一种称为 Java 文档注释 (javadoc comment) 的特殊注释形式。javadoc 注释以 `/**` 开始，以 `*/` 结尾。它们能使用 JDK 的 javadoc 命令提取成一个 HTML 文件。要获得更多信息，参见配套网站的补充材料 III.Y。

使用 javadoc 注释 (`/**...*/`) 来注释整个类或整个方法。为了将这些注释提取出来放在一个 javadoc HTML 文件中，这些注释必须放在类或者方法头的前面。要注释方法中的某一步骤，使用行注释 (`//`)。

可以从 www.cs.armstrong.edu/liang/javadoc/Exercise1.html 看到一个 javadoc HTML 文件的示例。相应的 Java 代码在 www.cs.armstrong.edu/liang/javadoc/Exercise1.java 中。

1.9.2 正确的缩进和空白

保持一致的缩进风格会使程序更加清晰、易读、易于调试和维护。缩进 (indentation) 用于描述程序中组成部分或语句之间的结构性关系。即使将程序的所有语句都写在一行中，Java 也可以读懂这样的程序，但是正确的对齐能够使人们更易读懂和维护代码。在嵌套结构

中，每个内层的组成部分或语句应该比外层缩进两格。

二元操作符的两边应该各加一个空格，如下面语句所示：

```
System.out.println(3+4*4);           不好的风格
System.out.println(3 + 4 * 4);       良好的风格
```

1.9.3 块的风格

块是由花括号围起来的一组语句。块的写法有两种常用方式：次行（next-line）风格和行尾（end-of-line）风格，如下所示。

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Styles");
    }
}
```

次行风格

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Block Styles");
    }
}
```

行尾风格

次行风格将括号垂直对齐，因而使程序容易阅读，而行尾风格更节省空间，并有助于避免犯一些细小的程序设计错误。这两种风格都是可以采纳的。选择哪一种完全依赖于个人或组织的偏好。应该统一采用一种风格，建议不要将这两种风格混合使用。本书与 Java API 源代码保持一致，都采用行尾风格。

☛ 复习题

1.41 使用行尾括号风格，将下面的程序根据程序设计风格 and 文档指南进行重新格式化。

```
public class Test
{
    // Main method
    public static void main(String[] args) {
        /** Display output */
        System.out.println("Welcome to Java");
    }
}
```

1.10 程序设计错误

🔑 要点提示：程序设计错误可以分为三类：语法错误、运行时错误和逻辑错误。

1.10.1 语法错误

在编译过程中出现的错误称为语法错误（syntax error）或编译错误（compile error）。语法错误是由创建代码时的错误引起的，例如：拼错关键字，忽略了一些必要的标点符号，或者左花括号没有对应的右花括号。这些错误通常很容易检测到，因为编译器会告诉你这些错误在哪儿，以及是什么原因造成的。例如：编译下面程序清单 1-4 中的程序会出现语法错误，如图 1-10 所示。

程序清单 1-4 ShowSyntaxErrors.java

```
1 public class ShowSyntaxErrors {
2     public static main(String[] args) {
3         System.out.println("Welcome to Java");
4     }
5 }
```

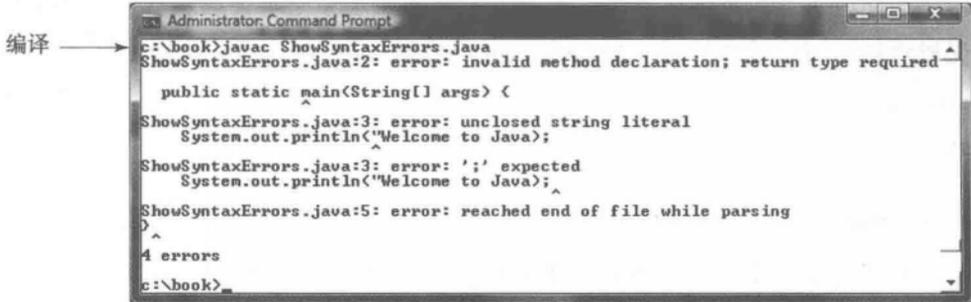


图 1-10 编译器报告语法错误

四个错误被报告，但实际上程序有两个错误：

- 第 2 行 main 方法前遗漏关键字 void。
- 第 3 行的字符串 Welcome to Java 应该加上引号。

由于一个错误常常会显示很多行的编译错误，因此，从最上面的行开始向下纠正错误是一个很好的习惯。解决程序前面出现的错误，可能就改正了程序后面出现的其他错误。

提示：如果你不知道如何纠正错误，将你的程序一个字符一个字符地仔细对照教材中的类似示例。在课程的前面几周，你可能要花许多时间纠正语法错误，但是很快你将熟悉 Java 语法，并快速纠正语法错误。

1.10.2 运行时错误

运行时错误 (runtime error) 是引起程序非正常中断的错误。运行应用程序时，当环境检测到一个不可能执行的操作时，就会出现运行时错误。输入错误是典型的运行时错误。当程序等待用户输入一个值，而用户输入了一个程序不能处理的值时，就会发生输入错误。例如：如果程序希望读入的是一个数值，而用户输入的却是一个字符串，就会导致程序出现数据类型错误。

另一个常见的运行时错误是 0 作除数。当整数除法中除数为 0 时可能引发这种情况。例如：下面程序清单 1-5 中的程序将会导致运行时错误，如图 1-11 所示。

程序清单 1-5 ShowRuntimeErrors.java

```

1 public class ShowRuntimeErrors {
2     public static void main(String[] args) {
3         System.out.println(1 / 0);
4     }
5 }

```

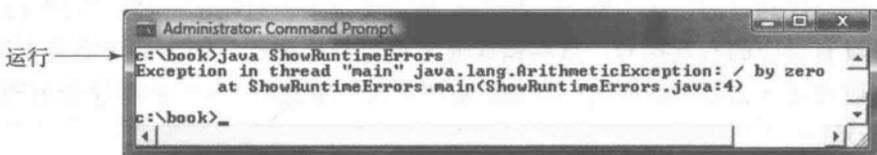


图 1-11 运行时错误导致程序非正常中断

1.10.3 逻辑错误

当程序没有按预期的方式执行时就会发生逻辑错误 (logic error)。这种错误发生的原因

有很多种。例如，假设你编写如程序清单 1-6 中的程序，将摄氏 35 度转换为华氏度：

程序清单 1-6 ShowLogicErrors.java

```
1 public class ShowLogicErrors {
2     public static void main(String[] args) {
3         System.out.println("Celsius 35 is Fahrenheit degree ");
4         System.out.println((9 / 5) * 35 + 32);
5     }
6 }
```

```
Celsius 35 is Fahrenheit degree
67
```

你将得到结果华氏 67 度，而这是错误的，结果应该是 95.0。Java 中，整数相除是返回除法的整数部分，即小数部分被截掉，因此 Java 中 9/5 的结果是 1。要得到正确的结果，需要使用 9.0/5，这样得到结果 1.8。

通常情况下，因为编译器可以明确指出错误的位置以及出错的原因，所以语法错误是很容易发现和纠正的。运行时错误也不难找，因为在程序异常中止时，错误的原因和位置都会显示在控制台上。然而，查找逻辑错误就很富有挑战性。在下面的章节中，我们将学习跟踪程序以及找到逻辑错误的技巧。

1.10.4 常见错误

对于编程新手来说，遗漏右括号、遗漏分号、遗漏字符串的引号、命名拼写错误，都是常见的错误。

常见错误 1：遗漏右括号

括号用来标识程序中的块。每个左括号必须有一个右括号匹配。常见的错误是遗漏右括号。为避免这个错误，任何时候输入左括号的时候就输入右括号，如下面的例子所示：

```
public class Welcome {
```

```
} ←—— 立刻输入右括号匹配左括号
```

如果使用 NetBeans 和 Eclipse 这样的 IDE，IDE 将自动为每个输入的左括号插入一个右括号。

常见错误 2：遗漏分号

每个语句都以一个语句结束符（；）结束。通常，编程入门者会忘了在一个块的最后一行语句后加上语句结束符，如下面例子所示：

```
public static void main(String[] args) {
    System.out.println("Programming is fun!");
    System.out.println("Fundamentals First");
    System.out.println("Problem Driven")
}
```

↑
遗漏一个分号

常见错误 3：遗漏引号

字符串必须放在引号中。通常，编程入门者会忘记在字符串结尾处加上一个引号，如下面例子所示：

```
System.out.println("Problem Driven");
```

↑
遗漏一个引号

如果使用 NetBeans 和 Eclipse 这样的 IDE，IDE 将自动为每个输入的左引号插入一个右引号。

常见错误 4：命名拼写错误

Java 是大小写敏感的。编程入门者常将名称拼写错误。例如，下面的代码中 main 错误拼写成 Main，String 错误拼写成 string。

```
1 public class Test {
2     public static void Main(string[] args) {
3         System.out.println((10.5 + 2 * 3) / (45 - 3.5));
4     }
5 }
```

复习题

- 1.42 什么是语法错误（编译错误）、运行时错误以及逻辑错误？
- 1.43 给出语法错误、运行时错误以及逻辑错误的示例。
- 1.44 如果忘记为字符串加引号了，将产生哪类错误？
- 1.45 如果程序需要读取整数，而用户输入了字符串，运行该程序的时候将产生什么错误？这是哪类错误？
- 1.46 假设编写一个计算矩形周长的程序，但是错误地写成了计算矩形面积的程序。这属于哪类错误？
- 1.47 指出和修改下面代码中的错误：

```
1 public class Welcome {
2     public void Main(String[] args) {
3         System.out.println('Welcome to Java!');
4     }
5 }
```

1.11 使用 NetBeans 开发 Java 程序

要点提示：可以使用 NetBeans 来编辑、编译、运行和调试 Java 程序。

NetBeans 和 Eclipse 是两个开发 Java 程序的免费的流行集成开发环境。如果按照简单的指南学习，可以很快掌握。我们建议你采用其中之一来开发 Java 程序。本节对于初学者给出基本的指南，在 NetBeans 环境中创建一个工程，创建类，以及编译和运行类。Eclipse 的使用将在下节中介绍。参考补充材料 II.B 以得到如何下载以及安装最新版本 NetBeans 的指南。

1.11.1 创建 Java 工程

创建 Java 程序前，首先需要创建一个工程。工程类似于一个文件夹，用于包含 Java 程序以及所有的支持文件。你只需要创建工程一次。这里是创建 Java 工程的步骤：

- 1) 选择 File → New Project 来显示 New Project 对话框，如图 1-12 所示。
- 2) 在 Categories 部分选择 Java，Projects 部分选择 Java Application，然后单击 Next 来显示 New Java Application 对话框，如图 1-13 所示。
- 3) 在 Project Name 域中输入 demo，在 Project Location 域中输入 c:\michael。去掉 Use Dedicated Folder for Storing Libraries 的勾选，并且去掉 Create Main Class 的勾选。
- 4) 单击 Finish 来创建工程，如图 1-14 所示。

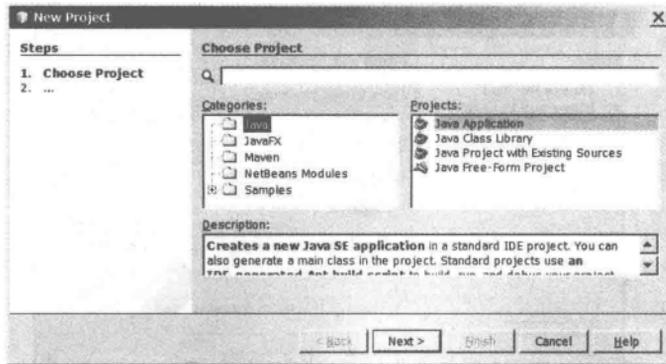


图 1-12 使用 New Project 对话框来创建一个新的工程，并且指定工程类别

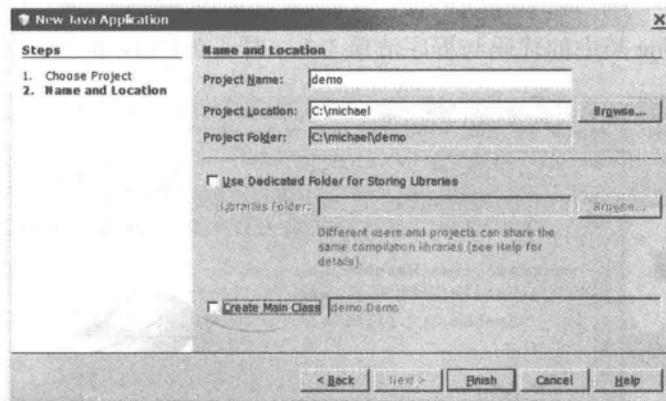


图 1-13 New Java Application 对话框用于确定工程名称和位置

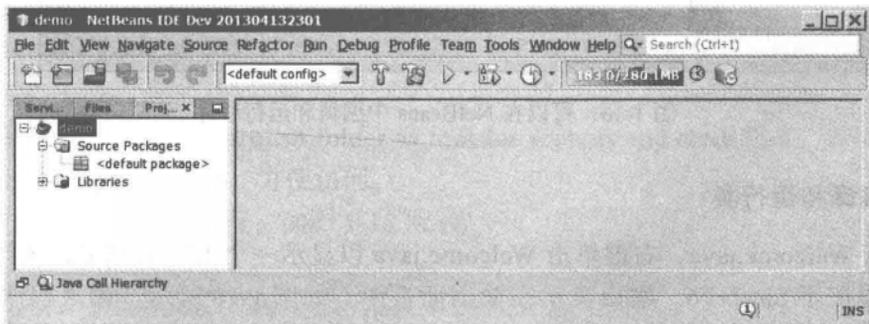


图 1-14 创建一个新的命名为 demo 的 Java 工程

1.11.2 创建 Java 类

工程创建后，可以采用以下步骤在工程中创建 Java 程序：

1) 右键单击工程面板的 demo 节点，显示一个上下文菜单。选择 New → Java Class 来显示 New Java Class 对话框，如图 1-15 所示。

2) 在 Class Name 域输入 Welcome，在 Location 域中选择 Source Packages。Package 域保留为空白，这样将在默认包中创建一个类。

3) 单击 Finish 来创建 Welcome 类。源代码文件 Welcome.java 放置在 <default package> 节点下面。

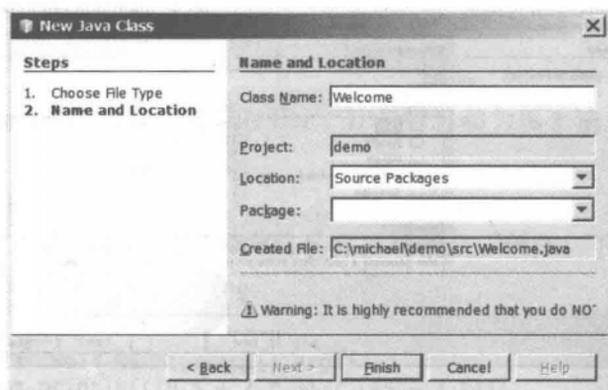


图 1-15 使用 New Java Class 对话框来创建一个新的 Java 类

4) 修改 Welcome 类中的代码与程序清单 1-1 一样, 如图 1-16 所示。

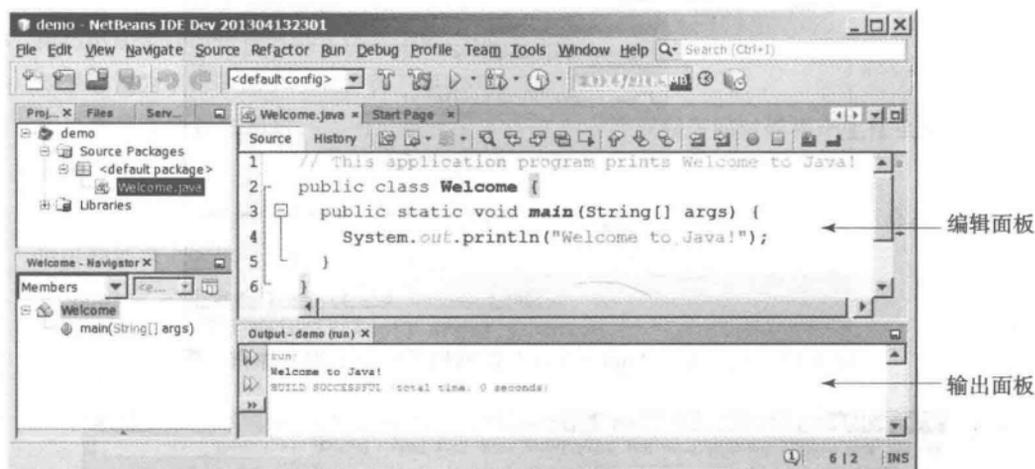


图 1-16 可以在 NetBeans 中编辑和运行程序

1.11.3 编译和运行类

要运行 Welcome.java, 右键单击 Welcome.java 以显示一个上下文菜单, 选择 Run File, 或者简单地按下 Shift+F6。输出显示在输出面板中, 如图 1-16 所示。如果程序被修改了, Run File 命令自动编译程序。

1.12 使用 Eclipse 开发 Java 程序

要点提示: 可以使用 Eclipse 来编辑、编译、运行和调试 Java 程序。

前一节介绍使用 NetBeans 开发 Java 程序, 也可以采用 Eclipse 开发 Java 程序。本节给出基本教程, 指导初学者在 Eclipse 下面创建工程, 创建类, 以及编译/运行类。参考补充材料 II.D 以得到如何下载以及安装最新版本 Eclipse 的指南。

1.12.1 创建 Java 工程

使用 Eclipse 创建 Java 程序前, 首先需要创建一个工程包含所有的文件。下面是 Eclipse

中创建 Java 工程的步骤：

1) 选择 File → New → Java Project 来显示 New Java Project 向导，如图 1-17 所示。

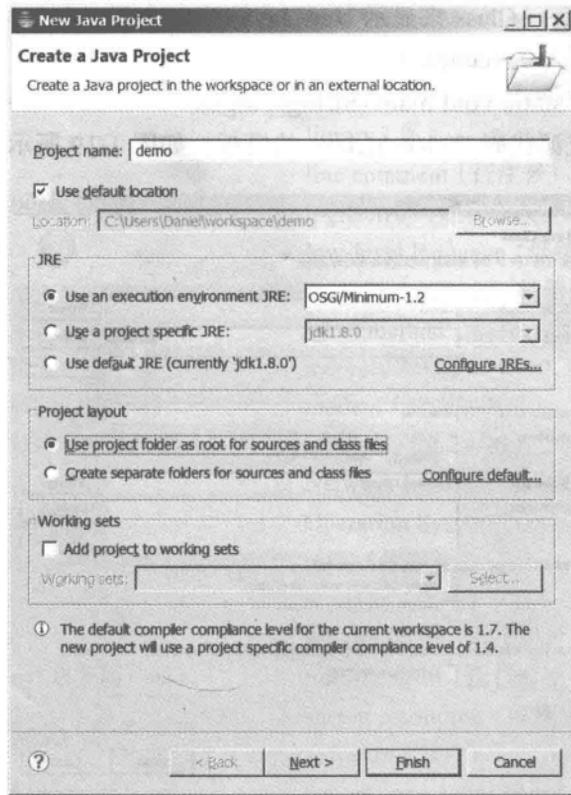


图 1-17 New Java Project 对话框用于确定工程名称和属性

2) 在 Project name 域中输入 demo，Location 域自动设置为默认。可以为你的工程自定义位置。

3) 确保选择了选项 Use project folder as root for sources and class files，从而 .java 文件和 .class 文件在同一个目录下，方便访问。

4) 单击 Finish 来创建工程，如图 1-18 所示。

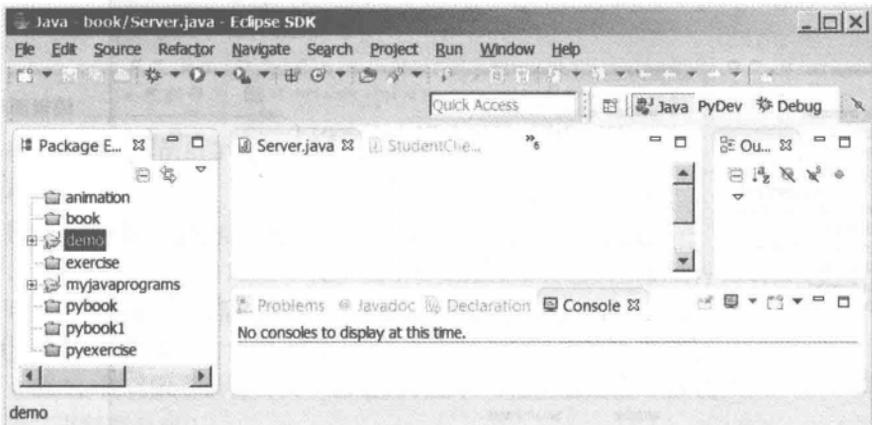


图 1-18 创建名为 demo 的新的 Java 工程

1.12.2 创建 Java 类

工程创建后，可以采用以下步骤在工程中创建 Java 程序：

- 1) 选择 File → New → Class 来显示 New Java Class 向导。
- 2) 在 Name 域中输入 Welcome。
- 3) 勾选选项 public static void main (String[] args)。
- 4) 单击 Finish 生成源代码 Welcome.java 的模板，如图 1-19 所示。

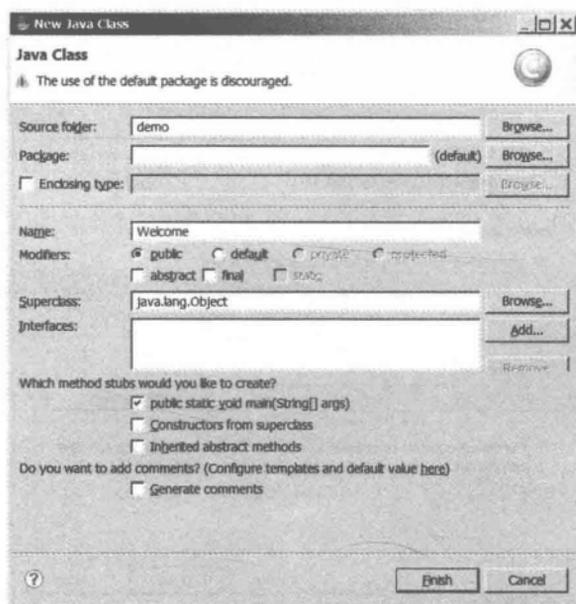


图 1-19 使用 New Java Class 对话框来创建一个新的 Java 类

1.12.3 编译和运行类

要运行程序，右键单击工程中的类，显示一个上下文菜单。在上下文菜单中选择 Run → Java Application 以运行类。输出显示在控制台面板中，如图 1-20 所示。

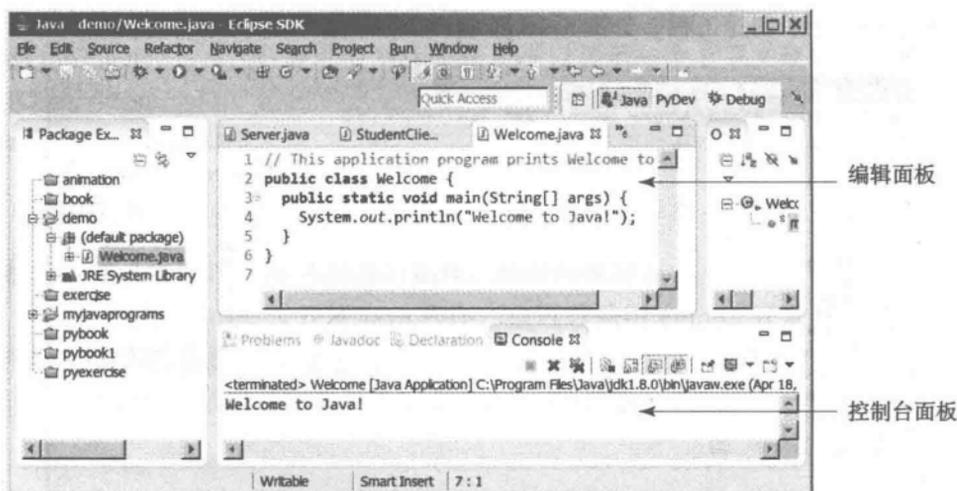


图 1-20 在 Eclipse 中编辑和运行程序

关键术语

Application Program Interface (API) (应用程序接口)	Java language specification (Java 语言规范)
assembler (汇编器)	Java Virtual Machine (JVM)(Java 虚拟机)
assembly language (汇编语言)	javac command (javac 命令)
bit (比特)	keyword or reserved word (关键字或保留字)
block (块)	library (库)
block comment (块注释)	line comment (行注释)
bus (总线)	logic error (逻辑错误)
byte (字节)	low-level language (低级语言)
bytecode (字节码)	machine language (机器语言)
bytecode verifier (字节码验证器)	main method (main 方法)
cable modem (电缆调制解调器)	memory (内存)
Central Processing Unit (CPU)(中央处理器)	modem (调制解调器)
class loader (类加载器)	motherboard (主板)
comment (注释)	Network Interface Card (NIC)(网络接口卡)
compiler (编译器)	Operation System (OS)(操作系统)
console (控制台)	pixel (像素)
dot pitch (点距)	program (程序)
DSL (Digital Subscriber Line)(数字用户线)	programming (程序设计)
encoding scheme (编码规范)	runtime error (运行时错误)
hardware (硬件)	screen resolution (屏幕分辨率)
high-level language (高级语言)	software (软件)
Integrated Development Environment (IDE) (集成开发环境)	source code (源代码)
interpreter (解释器)	source program (源程序)
java command (java 命令)	statement (语句)
Java Development Toolkit (JDK)(Java 开发工具包)	statement terminator (语句结束符)
	storage device (存储设备)
	syntax error (语法错误)

 **注意：**上面的术语都是本章所定义的。补充材料 I.A 按照章节顺序列出了本书所有的关键术语及其说明。

本章小结

1. 计算机是存储和处理数据的电子设备。
2. 计算机包括硬件和软件两部分。
3. 硬件是计算机中可以触摸到的物理部分。
4. 计算机程序，也就是通常所说的软件，是一些不可见的指令，它们控制硬件完成任务。
5. 计算机程序设计就是编写让计算机执行的指令（即代码）。
6. 中央处理器（CPU）是计算机的大脑。它从内存获取指令并且执行这些指令。
7. 计算机使用 0 或 1，因为数字设备有两个稳定的状态，习惯上就是指 0 和 1。
8. 一个比特是指二进制数 0 或 1。
9. 一个字节是指 8 比特的序列。

10. 千字节大约是 1000 字节, 兆字节大约是 100 万字节, 千兆字节大约是 10 亿字节, 万亿字节大约是 1 万亿字节。
11. 内存存储 CPU 要执行的数据和程序指令。
12. 内存单元是字节的有序序列。
13. 内存是不能长久保存数据的, 因为断电时信息就会丢失。
14. 程序和数据永久地保存在存储设备里, 当计算机确实需要使用它们时被移入内存。
15. 机器语言是一套内嵌在每台计算机的原始指令集。
16. 汇编语言是一种低级程序设计语言, 它用助记符表示每一条机器语言的指令。
17. 高级语言类似英语, 易于学习和编写程序。
18. 用高级语言编写的程序称为源程序。
19. 编译器是将源程序翻译成机器语言程序的软件。
20. 操作系统 (OS) 是管理和控制计算机活动的程序。
21. Java 是平台无关的, 这意味着只需编写一次程序, 就可以在任何计算机上运行。
22. Java 程序可以内嵌在 HTML 网页内, 通过 Web 浏览器下载, 给 Web 客户带来生动的动画和灵活的交互性。
23. Java 源程序文件名必须和程序中的公共类名一致, 并且以扩展名 .java 结束。
24. 每个类都被编译成一个独立的字节码文件, 该文件名与类名相同, 扩展名为 .class。
25. 使用 javac 命令可以从命令行编译 Java 源代码文件。
26. 使用 java 命令可以从命令行运行 Java 类。
27. 每个 Java 程序都是一套类的定义集合。关键字 class 引入类的定义, 类的内容包含在块内。
28. 一个块以左花括号 { } 开始, 以右花括号 } 结束。
29. 方法包含在类中。每个可执行的 Java 程序必须有一个 main 方法。main 方法是程序开始执行的入口。
30. Java 中的每条语句都是以分号 (;) 结束的, 也称该符号为语句结束符。
31. 保留字或者称关键字, 对编译器而言都有特殊含义, 在程序中不能用于其他目的。
32. 在 Java 中, 在单行上用两个斜杠 (//) 引导注释, 称为行注释; 在一行或多行用 /* 和 */ 包含注释, 称为块注释或者段注释。编译器会忽略注释。
33. Java 源程序是区分大小写的。
34. 编程错误可以分为三类: 语法错误、运行时错误和逻辑错误。编译器报告的错误称为语法错误或者编译错误。运行时错误指引起程序非正常结束的错误。当一个程序没有按照预期的方式执行时, 产生逻辑错误。

测试题

在线回答本章测试题, 地址为 www.cs.armstrong.edu/liang/intro10e/quiz.html。

编程练习题

 **注意:** 偶数题号的答案在配套网站上。所有编程练习题的答案在教师资源网站中。额外的编程练习题以及答案提供给教师。题目的难度等级分为容易 (没有星号)、适中 (*)、难 (**) 以及具有挑战性 (***)。

- 1.1 (显示三条消息) 编写程序, 显示 Welcome to Java、Welcome to Computer Science 和 Programming is fun。
- 1.2 (显示五条消息) 编写程序, 显示 Welcome to Java 五次。
- *1.3 (显示图案) 编写一个程序, 显示下面的图案:

```

      J   A   V   V   A
      J   A A   V   V   A A
      J J  AAAAA   V V   AAAAA
      J J  A   A   V   A   A

```

1.4 (打印表格) 编写程序, 显示以下表格:

a	a ²	a ³
1	1	1
2	4	8
3	9	27
4	16	64

1.5 (计算表达式) 编写程序, 显示以下公式的结果。

$$\frac{9.5 \times 4.5 - 2.5 \times 3}{45.5 - 3.5}$$

1.6 (数列求和) 编写程序, 显示 $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$ 的结果。

1.7 (近似求 π) 可以使用以下公式计算 π :

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \dots \right)$$

编写程序, 显示 $4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} \right)$ 和 $4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} \right)$ 的结果。在程序中用 1.0 代替 1。

1.8 (圆的面积和周长) 编写程序, 使用以下公式计算并显示半径为 5.5 的圆的面积和周长。

$$\text{周长} = 2 \times \text{半径} \times \pi$$

$$\text{面积} = \text{半径} \times \text{半径} \times \pi$$

1.9 (矩形的面积和周长) 编写程序, 使用以下公式计算并显示宽度为 4.5、高度为 7.9 的矩形的面积和周长。

$$\text{面积} = \text{宽} \times \text{高}$$

1.10 (以英里计的平均速度) 假设一个跑步者 45 分钟 30 秒内跑了 14 公里。编写一个程序显示以每小时多少英里为单位的平均速度值。(注意, 1 英里等于 1.6 公里。)

*1.11 (人口估算) 美国人口调查局基于以下假设进行人口估算:

- 每 7 秒有一个人诞生
- 每 13 秒有一个人死亡
- 每 45 秒有一个移民迁入

编写一个程序, 显示未来 5 年的每年的人口数。假设当前的人口是 312 032 486, 每年有 365 天。提示: Java 中, 两个整数相除, 结果还是整数, 小数部分被去掉。例如, $5/4$ 等于 1 (而不是 1.25), $10/4$ 等于 2 (而不是 2.5)。如果想得到有小数部分的精确结果, 进行除法运算的两个值之一必须是一个具有小数点的数值。例如, $5.0/4$ 等于 1.25, $10/4.0$ 等于 2.5。

1.12 (以公里计的平均速度) 假设一个跑步者 1 小时 40 分钟 35 秒内跑了 24 英里。编写一个程序显示以每小时多少公里为单位的平均速度值。(注意, 1 英里等于 1.6 公里。)

*1.13 (代数: 求解 2×2 线性方程) 可以使用 Cramer 规则解下面的 2×2 线性方程组:

$$\begin{cases} ax + by = e \\ cx + dy = f \end{cases} \quad x = \frac{ed - bf}{ad - bc} \quad y = \frac{af - ec}{ad - bc}$$

编写程序, 求解以下方程组并显示 x 和 y 的值。

$$3.4x + 50.2y = 44.5$$

$$2.1x + 0.55y = 5.9$$

基本程序设计

教学目标

- 编写 Java 程序完成简单的计算 (2.2 节)。
- 使用 Scanner 类从控制台获取输入 (2.3 节)。
- 使用标识符命名变量、常量、方法和类 (2.4 节)。
- 使用变量存储数据 (2.5 ~ 2.6 节)。
- 用赋值语句和赋值表达式编写程序 (2.6 节)。
- 使用常量存储永久数据 (2.7 节)。
- 按照命名习惯命名类, 方法, 变量和常量 (2.8 节)。
- 探索 Java 的基本数值类型: byte、short、int、long、float 和 double (2.9.1 节)。
- 从键盘读入一个 byte、short、int、long、float 或者 double 类型的值 (2.9.2 节)。
- 使用操作符 +、-、*、/ 和 % 来执行操作 (2.9.3 节)。
- 使用 Math.pow (a, b) 进行幂运算 (2.9.4 节)。
- 编写整数字面值、浮点数字面值, 以及科学表达式的字面值 (2.10 节)。
- 编写和计算数值表达式 (2.11 节)。
- 使用 System.currentTimeMillis() 获得当前系统时间 (2.12 节)。
- 使用增量赋值操作符 (2.13 节)。
- 区分后置递增和前置递增, 以及后置递减和前置递减 (2.14 节)。
- 将一种类型的值强制转换为另一种类型 (2.15 节)。
- 描述软件开发过程, 并将其应用于开发贷款支付额程序 (2.16 节)。
- 编写程序, 计算整钱兑零 (2.17 节)。
- 避免基础编程中常见错误和陷阱 (2.18 节)。

2.1 引言

 **要点提示:** 本章的重点是学习基础程序设计技术, 以进行问题求解。

在第 1 章里, 我们学习了如何创建、编译和运行非常基础的 Java 程序。现在, 将学习如何编程解决实际问题。通过这些问题, 你将学到如何利用基本数据类型、变量、常量、操作符、表达式以及输入/输出来进行基本的程序设计。

比如, 假设你需要计算一个学生贷款。给定贷款额度、贷款时间以及年利率, 你可以通过编写程序来计算每月支付以及整个支付额度吗? 本章将演示如何编写这样的程序。用这样的方法, 你将学习分析问题, 设计一个解决方案以及通过创建一个程序来实现这个解决方案的基本步骤。

2.2 编写简单的程序

 **要点提示:** 编写程序涉及如何设计解决问题的策略, 以及如何应用编程语言实现这个策略。

首先,我们来看一个计算圆面积的简单问题。该如何编写程序解决这个问题呢?

编写程序涉及如何设计算法以及如何将算法翻译成程序指令,即代码。算法描述的是:如果要解决问题,所需要执行的动作以及这些动作执行的顺序。算法可以帮助程序员在使用程序设计语言编写程序之前做一个规划。算法可以用自然语言或者伪代码(即自然语言和程序设计代码混在一起使用)描述。这个程序的算法描述如下:

1) 读入半径。

2) 利用下面的公式计算面积:

$$\text{面积} = \text{半径} \times \text{半径} \times \pi$$

3) 显示面积。

 **提示:** 在编写代码之前,以一种算法的形式来勾勒你的程序(或者它潜在的问题),是一个很好的做法。

当你编码,也就是当你编写一个程序时,你将一个算法翻译成程序。你已经知道每个 Java 程序都是以一个类的声明开始,在声明里类名紧跟在关键字 `class` 后面。假设你选择 `ComputeArea` 作为这个类的类名。这个程序的框架就如下所示:

```
public class ComputeArea {
    // Details to be given later
}
```

如你所知,每一个 Java 应用程序都必须有一个 `main` 方法,程序从该方法处开始执行。所以该程序被扩展为如下所示:

```
public class ComputeArea {
    public static void main(String[] args) {
        // Step 1: Read in radius

        // Step 2: Compute area
        // Step 3: Display the area
    }
}
```

这个程序需要读取用户从键盘输入的半径。这就产生了两个重要问题:

- 读取半径。
- 将半径存储在程序中。

我们先来解决第二个问题。为了存储半径,在程序中需要声明一个称作变量的符号。变量代表了存储在计算机内存中的一个值。

变量名应该尽量选择描述性的名字 (*descriptive name*),而不是用 `x` 和 `y` 这样的名字:在这里的例子中,用 `radius` 表示半径、用 `area` 表示面积。为了让编译器知道 `radius` 和 `area` 是什么,需要指明它们的数据类型,即存储在变量中的数据的类型,是整数、实数,或者其他。这称为声明变量。Java 提供简单数据类型来表示整数、实数、字符以及布尔类型。这些类型称为原始数据类型或基本类型。

实数(即带小数点的数字)在计算机中使用一种浮点的方法来表示。因此,实数也称为浮点数。Java 中,可以使用关键字 `double` 来声明一个浮点变量。将 `radius` 和 `area` 声明为 `double`。程序可被扩展为如下所示:

```
public class ComputeArea {
    public static void main(String[] args) {
        double radius;
        double area;
    }
}
```

```

// Step 1: Read in radius
// Step 2: Compute area
// Step 3: Display the area
}
}

```

程序将 `radius` 和 `area` 声明为变量。保留字 `double` 表明 `radius` 和 `area` 是以浮点数形式存储在计算机中的。

第一步是提示用户指定圆的半径。稍后我们会学习如何提示用户获得信息。现在为了学习变量是如何工作的，可以给程序中的 `radius` 赋一个固定值；之后，修改程序，以提示用户输入这个值。

第二步是计算 `area`，这是通过将表达式 `radius*radius*3.14159` 的值赋给 `area` 来实现的。在最后一步里，使用 `System.out.println` 方法在控制台上显示 `area` 的值。完整的程序如程序清单 2-1 所示。该程序的示例运行如图 2-1 所示。

程序清单 2-1 ComputeArea.java

```

1 public class ComputeArea {
2     public static void main(String[] args) {
3         double radius; // Declare radius
4         double area; // Declare area
5
6         // Assign a radius
7         radius = 20; // radius is now 20
8
9         // Compute area
10        area = radius * radius * 3.14159;
11
12        // Display results
13        System.out.println("The area for the circle of radius " +
14            radius + " is " + area);
15    }
16 }

```

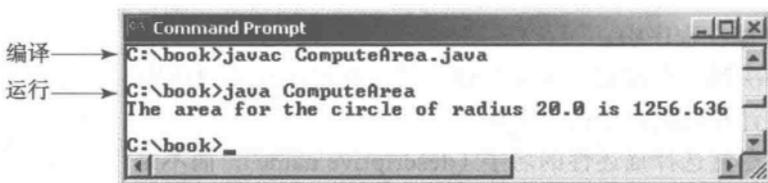


图 2-1 程序显示圆的面积

像 `radius` 和 `area` 这样的变量对应于它们在内存的位置。每个变量都有名字、类型、大小和值。第 3 行声明 `radius` 可以存储一个 `double` 型的数值。直到给它赋一个数值时，该变量才被定义。第 7 行将 `radius` 赋值为 20。类似地，第 4 行声明变量 `area`，第 10 行将 10 赋值给 `area`。下面的表格显示的是随着程序的执行，`area` 和 `radius` 在内存中的值。该表中的每一行显示的是程序中对应的每行语句执行之后变量的值。这种审查程序如何工作的方法称为跟踪一个程序。跟踪程序有助于理解一个程序是如何工作的，而且它也是一个查找程序错误的非常有用的工具。

行号	半径	面积
3	无值	
4		无值
7	20	
10		1256.636

加号 (+) 有两种意义：一种用途是做加法，另一种用途是做字符串的连接（合并）。第 13 ~ 14 行中的加号 (+) 称为字符串连接符。它把两个字符串合并为一个。如果一个字符串和一个数值连接，数值将转化为字符串然后再和另外一个字符串连接。所以，第 13 ~ 14 行的加号 (+) 会将几个字符串连成一个更长的字符串，然后将它在输出结果中显示出来。关于字符串以及字符串连接的更多内容将在第 4 章中讨论。

 **警告：**在源代码中，字符串常量不能跨行。因此，下面的语句会造成编译错误：

```
System.out.println("Introduction to Java Programming,
by Y. Daniel Liang");
```

为了改正错误，将该字符串分成几个单独的子串，然后再用连接符 (+) 将它们组合起来：

```
System.out.println("Introduction to Java Programming, " +
"by Y. Daniel Liang");
```

复习题

2.1 指出并修改以下代码中的错误：

```
1 public class Test {
2     public void main(string[] args) {
3         double i = 50.0;
4         double k = i + 50.0;
5         double j = k + 1;
6
7         System.out.println("j is " + j + " and
8             k is " + k);
9     }
10 }
```

2.3 从控制台读取输入

 **要点提示：**从控制台读取输入，使得程序可以从用户那里获得输入。

在程序清单 2-1 中，源代码中的半径是固定的。为了能使用不同的半径，必须修改源代码然后重新编译它。很显然，这是非常不方便的。可以使用 Scanner 类从控制台输入。

Java 使用 System.out 来表示标准输出设备，而用 System.in 来表示标准输入设备。默认情况下，输出设备是显示器，而输入设备是键盘。为了完成控制台输出，只需使用 println 方法就可以在控制台上显示基本值或字符串。Java 并不直接支持控制台输入，但是可以使用 Scanner 类创建它的对象，以读取来自 System.in 的输入，如下所示：

```
Scanner input = new Scanner(System.in);
```

语法 new Scanner (System.in) 表明创建了一个 Scanner 类型的对象。语法 Scanner input 声明 input 是一个 Scanner 类型的变量。整行的 Scanner input=new Scanner (System.in) 表明创建了一个 Scanner 对象，并且将它的引用值赋值给变量 input。对象可以调用它自己的方法。调用对象的方法就是让这个对象完成某个任务。可以调用

nextDouble() 方法来读取一个 double 值，如下所示：

```
double radius = input.nextDouble();
```

该语句从键盘读入一个数值，并且将该数值赋给 radius。

程序清单 2-2 重写程序清单 2-1，提示用户输入一个半径。

程序清单 2-2 ComputeAreaWithConsoleInput.java

```
1 import java.util.Scanner; // Scanner is in the java.util package
2
3 public class ComputeAreaWithConsoleInput {
4     public static void main(String[] args) {
5         // Create a Scanner object
6         Scanner input = new Scanner(System.in);
7
8         // Prompt the user to enter a radius
9         System.out.print("Enter a number for radius: ");
10        double radius = input.nextDouble();
11
12        // Compute area
13        double area = radius * radius * 3.14159;
14
15        // Display results
16        System.out.println("The area for the circle of radius " +
17            radius + " is " + area);
18    }
19 }
```

```
Enter a number for radius: 2.5 ↵
The area for the circle of radius 2.5 is 19.6349375
```

```
Enter a number for radius: 23 ↵
The area for the circle of radius 23.0 is 1661.90111
```

第 9 行的语句在控制台显示一个字符串 "Enter a number for radius:"，这称为一个提示，因为它指导用户键入输入。你的程序应该在希望得到键盘输入的时候，告知用户输入什么。

第 9 行的 print 方法

```
System.out.print("Enter a number for radius: ");
```

和 println 方法很类似，两者的不同之处在于：当显示完字符串之后，println 会将光标移到下一行，而 print 不会将光标移到下一行。

第 6 行创建一个 Scanner 对象。第 10 行的语句从键盘读入一个输入。

```
double radius = input.nextDouble();
```

在用户键入一个数值然后单击回车键之后，该数值就被读入并赋值给 radius。

更多关于对象的细节将在第 9 章中介绍。目前，只要知道这是如何从控制台获取输入的方式就可以了。

Scanner 类在包 java.util 里。它在第 1 行被导入。import 语句有两种类型：明确导入 (specific import) 和通配符导入 (wildcard import)。明确导入是在 import 语句中指定单个的类。例如，下面的语句就是从包 java.util 中导入 Scanner。

```
import java.util.Scanner;
```

通配符导入是指通过使用星号作为通配符，导入一个包中所有的类。例如，下面的语句导入包 `java.util` 中所有的类。

```
import java.util.*;
```

除非要在程序中使用某个类，否则关于被导入包中的这些类的信息在编译时或运行时是不被读入的。导入语句只是告诉编译器在什么地方能找到这些类。声明明确导入和声明通配符导入在性能上是没有什么差别的。

程序清单 2-3 给出从键盘读取多个输入的例子。这个例子读取三个数值，然后显示它们的平均值。

程序清单 2-3 ComputeAverage.java

```
1 import java.util.Scanner; // Scanner is in the java.util package
2
3 public class ComputeAverage {
4     public static void main(String[] args) {
5         // Create a Scanner object
6         Scanner input = new Scanner(System.in);
7
8         // Prompt the user to enter three numbers
9         System.out.print("Enter three numbers: ");
10        double number1 = input.nextDouble();
11        double number2 = input.nextDouble();
12        double number3 = input.nextDouble();
13
14        // Compute average
15        double average = (number1 + number2 + number3) / 3;
16
17        // Display results
18        System.out.println("The average of " + number1 + " " + number2
19            + " " + number3 + " is " + average);
20    }
21 }
```

```
Enter three numbers: 1 2 3 ↵
The average of 1.0 2.0 3.0 is 2.0
```

```
Enter three numbers: 10.5 ↵
11 ↵
11.5 ↵
The average of 10.5 11.0 11.5 is 11.0
```

导入 `Scanner` 类的代码（第 1 行）以及创建 `Scanner` 对象的代码（第 6 行）都是和前一个例子一样的，而且在你将编写的所有新程序中，这两行也都是一样的。

第 9 行提示用户输入三个数值。这些数值在第 10 ~ 12 行被读取。可以输入三个用空格符分隔开的数值，然后按回车键，或者每输入一个数值之后就按一次回车键，如该程序的示例运行所示。

如果输入了一个非数值的值，一个运行时错误将产生。在第 12 章中，我们将学习如何处理异常，保证程序可以继续运行下去。

注意：本书前面章节中的大多数程序分三个步骤执行，即输入、处理和输出，这被称为 IPO。输入是从用户那里获得输入，处理是使用输入产生结果，而输出是显示结果。

复习题

2.2 如何编写一条语句，让用户从键盘输入一个双精度值？在执行下面代码的时候，如果你输入 5a，将发生什么？

```
double radius = input.nextDouble();
```

2.3 下面两个 import 语句之间有什么执行的不同吗？

```
import java.util.Scanner;
import java.util.*;
```

2.4 标识符

要点提示：标识符是为了标识程序中诸如类、方法和变量的元素而采用的命名。

正如在程序清单 2-3 中看到的，ComputeAverage、main、input、number1、number2、number3 等都是出现在程序中事物的名字。在程序设计术语中，这样的名字称为标识符 (identifier)。所有的标识符必须遵从以下规则：

- 标识符是由字母、数字、下划线 (_) 和美元符号 (\$) 构成的字符序列。
- 标识符必须以字母、下划线 (_) 或美元符号 (\$) 开头，不能以数字开头。
- 标识符不能是保留字 (参见附录 A 中的保留字列表)。
- 标识符不能是 true、false 或 null。
- 标识符可以为任意长度。

例如，\$2、ComputeArea、area、radius 和 print 都是合法的标识符，而 2A 和 d+4 都是非法的，因为它们不符合标识符的命名规则。Java 编译器会检测出非法标识符，并且报语法错误。

注意：由于 Java 是区分大小写的，所以 area、Area 和 AREA 都是不同的标识符。

提示：标识符是用于命名程序中的变量、方法、类和其他项。具有描述性的标识符可提高程序的可读性。避免采用缩写作为标识符，使用完整的词汇会更具有描述性。比如，numberOfStudents 比 numStuds、numOfStuds 或者 numofStudents 要好。本教材中我们在完整的程序采用描述性的命名。然而，为了简明，我们也会偶尔在一些代码片段中采用诸如 i、j、k、x 和 y 之类的变量名。这样的命名在代码片段中也是具有一定普遍性的做法。

提示：不要用字符 \$ 命名标识符。习惯上，字符 \$ 只用在机器自动产生的源代码中。

复习题

2.4 以下标识符哪些是合法的？哪些是 Java 的关键字？

```
miles, Test, a++, --a, 4#R, $4, #44, apps
class, public, int, x, y, radius
```

2.5 变量

要点提示：变量用于表示在程序中可能被改变的值。

正如在前几节的程序中看到的，变量被用于存储程序中后面要用到的值。它们被称为变量是因为它们的值可以被改变。在程序清单 2-2 中，radius 和 area 都是双精度浮点型变量。可以将任意数值赋给 radius 和 area，并且可以对它们重新赋值。例如，在下面的代码中，radius 初始为 1.0 (第 2 行)，然后被改为 2.0 (第 7 行)，面积被设为 3.14159 (第 3 行)，然

后被重新设置为 12.56636 (第 8 行)。

```

1 // Compute the first area
2 radius = 1.0;                                radius: 1.0
3 area = radius * radius * 3.14159;           area: 3.14159
4 System.out.println("The area is " + area + " for radius " + radius);
5
6 // Compute the second area
7 radius = 2.0;                                radius: 2.0
8 area = radius * radius * 3.14159;           area: 12.56636
9 System.out.println("The area is " + area + " for radius " + radius);

```

变量用于表示特定类型的数据。为了使用变量，可以通过告诉编译器变量的名字及其可以存储的数据类型来声明该变量。变量声明告知编译器根据数据类型为变量分配合适的内存空间。声明变量的语法如下：

```
datatype variableName;
```

下面是一些变量声明的例子：

```

int count; // Declare count to be an integer variable
double radius; // Declare radius to be a double variable
double interestRate; // Declare interestRate to be a double variable

```

这个例子中使用了数据类型 `int` 和 `double`。后面还将介绍更多的数据类型，例如，`byte`、`short`、`long`、`float`、`char` 和 `boolean`。

如果几个变量为同一类型，允许一起声明它们，如下所示：

```
datatype variable1, variable2, ..., variablen;
```

变量之间用逗号分隔开。例如：

```
int i, j, k; // Declare i, j, and k as int variables
```

变量通常都有初始值。可以一步完成变量的声明和初始化。例如，考虑下面的代码：

```
int count = 1;
```

它等同于下面的两条语句：

```

int count;
count = 1;

```

也可以使用简捷的方式来同时声明和初始化同一类型的变量。例如：

```
int i = 1, j = 2;
```

提示：在赋值给变量之前，必须声明变量。方法中声明的变量在使用之前必须被赋值。

任何时候，都要尽可能一步完成变量的声明和赋初值。这会使得程序易读，同时避免程序设计错误。

每个变量都有使用范围。变量的使用范围是指变量可以被引用到的程序的部分。定义变量使用范围的规则将在本书后面逐步介绍。目前，你需要知道的是，一个变量在可以使用前，必须被声明和初始化。

复习题

2.5 请指出并修改下面代码中的错误：

```

1 public class Test {
2     public static void main(String[] args) {

```

```

3     int i = k + 2;
4     System.out.println(i);
5 }
6 }

```

2.6 赋值语句和赋值表达式

要点提示：赋值语句将一个值指定给一个变量。在 Java 中赋值语句可以作为一个表达式。

声明变量之后，可以使用赋值语句（assignment statement）给它赋一个值。在 Java 中，将等号（=）作为赋值操作符（assignment operator）。赋值语句的语法如下所示：

```
variable = expression; (变量 = 表达式;)
```

表达式（expression）表示涉及值、变量和操作符的一个运算，它们组合在一起计算出一个新值。例如，考虑下面的代码：

```

int y = 1;           // Assign 1 to variable y
double radius = 1.0; // Assign 1.0 to variable radius
int x = 5 * (3 / 2); // Assign the value of the expression to x
x = y + 1;          // Assign the addition of y and 1 to x
double area = radius * radius * 3.14159; // Compute area

```

变量也可用在表达式中。变量也可以用于 = 操作符的两边，例如：

```
x = x + 1;
```

在这个赋值语句中， $x+1$ 的结果赋值给 x 。假设在语句执行前 x 为 1，那么语句执行后它就变成了 2。

要给一个变量赋值，变量名必须在赋值操作符的左边。因此，下面的语句是错误的。

```
1 = x; // Wrong
```

注意：在数学运算中， $x=2*x+1$ 表示一个等式。但是，在 Java 中， $x=2*x+1$ 是一个赋值语句，它计算表达式 $2*x+1$ ，并且将结果赋值给 x 。

在 Java 中，赋值语句本质上就是计算出一个值并将它赋给操作符左边变量的一个表达式。由于这个原因，赋值语句常常称作赋值表达式（assignment expression）。例如，下面的语句是正确的：

```
System.out.println(x = 1);
```

它等价于语句：

```

x = 1;
System.out.println(x);

```

如果一个值要赋给多个变量，可以采用以下语法：

```
i = j = k = 1;
```

它等价于：

```

k = 1;
j = k;
i = j;

```

注意：在赋值语句中，左边变量的数据类型必须与右边值的数据类型兼容。例如，`int x=1.0` 是非法的，因为 x 的数据类型是整型 `int`。在不使用类型转换的情况下，是不能把

double 值 (1.0) 赋给 int 变量的。类型转换将在 2.15 节介绍。

复习题

2.6 请指出并修改下面代码中的错误：

```
1 public class Test {
2     public static void main(String[] args) {
3         int i = j = k = 2;
4         System.out.println(i + " " + j + " " + k);
5     }
6 }
```

2.7 命名常量

要点提示：命名常量是一个代表不变值的标识符。

一个变量的值在程序执行过程中可能会发生变化，但是命名常量 (named constant) 或简称常量，则表示从不改变的永久数据。在程序清单 2-1 中， π 是一个常量。如果频繁使用它，但又不想重复地输入 3.14159，代替的方式就是声明一个常量 π 。下面就是声明常量的语法：

```
final datatype CONSTANTNAME = value;
```

常量必须在同一条语句中声明和赋值。单词 final 是声明常量的 Java 关键字。例如，可以将 π 声明为常量，然后将程序清单 2-1 改写为程序清单 2-4：

程序清单 2-4 ComputeAreaWithConstant.java

```
1 import java.util.Scanner; // Scanner is in the java.util package
2
3 public class ComputeAreaWithConstant {
4     public static void main(String[] args) {
5         final double PI = 3.14159; // Declare a constant
6
7         // Create a Scanner object
8         Scanner input = new Scanner(System.in);
9
10        // Prompt the user to enter a radius
11        System.out.print("Enter a number for radius: ");
12        double radius = input.nextDouble();
13
14        // Compute area
15        double area = radius * radius * PI;
16
17        // Display result
18        System.out.println("The area for the circle of radius " +
19            radius + " is " + area);
20    }
21 }
```

使用常量有三个好处：1) 不必重复输入同一个值；2) 如果必须修改常量值 (例如，将 PI 的值从 3.14 改为 3.14159)，只需在源代码中的一个地方做改动；3) 给常量赋一个描述性名字会提高程序易读性。

2.8 命名习惯

要点提示：严格遵循 Java 的命名习惯可以让你的程序易于理解，以及避免错误。

应该确保程序中为变量、常量、类和方法所选择的描述性名字是直观易懂的。如前所述，命名是区分大小写的。下面列出变量、常量、方法和类的命名习惯。

- 使用小写字母命名变量和方法。如果一个名字包含多个单词，就将它们连在一起，第一个单词的字母小写，而后面的每个单词的首字母大写，例如，变量 `radius` 和 `area` 以及方法 `print`。
- 类名中的每个单词的首字母大写，例如，类名 `ComputeArea` 和 `System`。
- 大写常量中的所有字母，两个单词间用下划线连接，例如，常量 `PI` 和常量 `MAX_VALUE`。

严格遵循 Java 的命名习惯是非常重要的，这样可以让你的程序易于理解。

 **警告：**对类命名时不要选择 Java 库中已经使用的名称。例如，因为 Java 已定义了 `System` 类，就不要用 `System` 来命名自己的类。

复习题

- 2.7 使用常量的好处是什么？声明一个 `int` 类型的常量 `SIZE`，并且值为 20。
- 2.8 类名、方法名、常量和变量的命名习惯是什么？按照 Java 的命名习惯，以下哪些项可以作为常量，方法、变量或者一个类？

`MAX_VALUE`, `Test`, `read`, `readDouble`

- 2.9 将以下算法翻译成 Java 代码。

第一步：声明一个双精度型变量 `miles`，初始值为 100。

第二步：声明一个双精度型常量 `KILOMETERS_PER_MILE`，初始值为 1.609。

第三步：声明一个双精度型变量 `kilometers`，将 `miles` 和 `KILOMETERS_PER_MILE` 相乘，并且将结果赋值给 `kilometers`。

第四步：在控制台显示 `kilometers`。

第四步之后，`kilometers` 是多少？

2.9 数值数据类型和操作

 **要点提示：**Java 针对整数和浮点数有六种数值类型，以及 `+`、`-`、`*`、`/`、和 `%` 等操作符。

2.9.1 数值类型

每个数据类型都有它的取值范围。编译器会根据每个变量或常量的数据类型为其分配内存空间。Java 为数值、字符值和布尔值数据提供了八种基本数据类型。本节介绍数值数据类型和操作符。

表 2-1 列出了六种数值数据类型、它们的范围以及所占存储空间。

表 2-1 数值数据类型

类型名	范围	存储大小
<code>byte</code>	-2^7 (-128) \sim 2^7-1 (127)	8 位带符号数
<code>short</code>	-2^{15} (-32 768) \sim $2^{15}-1$ (32 767)	16 位带符号数
<code>int</code>	-2^{31} (-2 147 483 648) \sim $2^{31}-1$ (2 147 483 647)	32 位带符号数
<code>long</code>	$-2^{63} \sim 2^{63}-1$ (即 -9 223 372 036 854 775 808 \sim 9 223 372 036 854 775 807)	64 位带符号数
<code>float</code>	负数范围: $-3.4028235E+38 \sim -1.4E-45$ 正数范围: $1.4E-45 \sim 3.4028235E+38$	32 位, 标准 IEEE 754

(续)

类型名	范围	存储大小
double	负数范围: $-1.7976931348623157E+308 \sim -4.9E-324$ 正数范围: $4.9E-324 \sim 1.7976931348623157E+308$	64 位, 标准 IEEE 754

注意: IEEE 754 是美国电气电子工程师协会通过的一个标准, 用于在计算机上表示浮点数。该标准已被广泛采用。Java 采用 32 位 IEEE 754 表示 float 型, 64 位 IEEE 754 表示 double 型。IEEE 754 标准还定义了一些特殊浮点值, 这些值都在附录 E 中列出。

Java 使用四种类型的整数: byte、short、int 和 long。应该为变量选择最适合的数据类型。例如: 如果知道存储在变量中的整数是在字节范围内, 将该变量声明为 byte 型。为了简单和一致性, 我们在本书的大部分内容中都使用 int 来表示整数。

Java 使用两种类型的浮点数: float 和 double。double 型是 float 型的两倍。所以, double 型又称为双精度 (double precision), 而 float 称为单精度 (single precision)。通常情况下, 应该使用 double 型, 因为它比 float 型更精确。

2.9.2 从键盘读取数值

你知道如何使用 Scanner 类中的 nextDouble() 方法来从键盘读取一个 double 数值。你也可以使用列在表 2-2 中的方法来读取 byte、short、int、long 以及 float 类型的数值。

下面是从键盘上读取各种类型数值的例子:

```

1 Scanner input = new Scanner(System.in);
2 System.out.print("Enter a byte value: ");
3 byte byteValue = input.nextByte();
4
5 System.out.print("Enter a short value: ");
6 short shortValue = input.nextShort();
7
8 System.out.print("Enter an int value: ");
9 int intValue = input.nextInt();
10
11 System.out.print("Enter a long value: ");
12 long longValue = input.nextLong();
13
14 System.out.print("Enter a float value: ");
15 float floatValue = input.nextFloat();

```

如果你输入了一个不正确范围或者格式的值, 将产生一个运行时错误。比如, 为第 3 行你输入了 128, 将产生一个错误, 因为 128 已经超过了 byte 类型整数的范围。

2.9.3 数值操作符

数值数据类型的操作符包括标准的算术操作符: 加号 (+)、减号 (-)、乘号 (*)、除号 (/) 和求余号 (%), 如表 2-3 所示。操作数是被操作符操作的值。

表 2-3 数值操作符

运算符	名字	示例	运算结果	运算符	名字	示例	运算结果
+	加	34 + 1	35	/	除	1.0 / 2.0	0.5
-	减	34.0 - 0.1	33.9	%	求余	20 % 3	2
*	乘	300 * 30	9000				

表 2-2 Scanner 对象的方法

方法	描述
nextByte()	读取一个 byte 类型的整数
nextShort()	读取一个 short 类型的整数
nextInt()	读取一个 int 类型的整数
nextLong()	读取一个 long 类型的整数
nextFloat()	读取一个 float 类型的数
nextDouble()	读取一个 double 类型的数

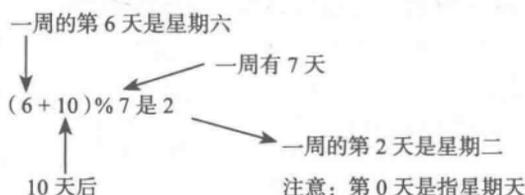
当除法的操作数都是整数时，除法的结果就是整数，小数部分被舍去。例如：5/2 的结果是 2 而不是 2.5，而 -5/2 的结果是 -2 而不是 -2.5。为了实现浮点数的除法，其中一个操作数必须是浮点数。例如：5.0/2 的结果是 2.5。

操作符 %，被称为求余或者取模操作符，可以求得除法的余数。左边的操作数是被除数，右边的操作数是除数。因此，7%3 的结果是 1，3%7 的结果是 3，12%4 的结果是 0，26%8 的结果是 2，20%13 的结果是 7。

$$\begin{array}{r}
 2 \\
 3 \overline{)7} \\
 \underline{6} \\
 1
 \end{array}
 \quad
 \begin{array}{r}
 0 \\
 7 \overline{)3} \\
 \underline{0} \\
 3
 \end{array}
 \quad
 \begin{array}{r}
 3 \\
 4 \overline{)12} \\
 \underline{12} \\
 0
 \end{array}
 \quad
 \begin{array}{r}
 0 \\
 8 \overline{)26} \\
 \underline{24} \\
 2
 \end{array}
 \quad
 \begin{array}{r}
 \text{被除数} \longrightarrow \\
 13 \overline{)20} \\
 \underline{13} \\
 7 \\
 \longleftarrow \text{商} \\
 \longleftarrow \text{除数} \\
 \longleftarrow \text{余数}
 \end{array}$$

操作符 % 通常用在正整数上，实际上，它也可用于负整数和浮点值。只有当被除数是负数时，余数才是负的。例如：-7%3 结果是 -1，-12%4 结果是 0，-26%-8 结果是 -2，20%-13 结果是 7。

在程序设计中余数是非常有用的。例如：偶数 %2 的结果总是 0 而正奇数 %2 的结果总是 1。所以，可以利用这一特性来判定一个数是偶数还是奇数。如果今天是星期六，7 天之后就又会星期六。假设你和你的朋友计划 10 天之后见面，那么 10 天之后是星期几呢？使用下面的表达式你就能够发现那天是星期二：



程序清单 2-5 计算以秒为单位的时间量所包含的分钟数和余下的秒数。例如，500 秒就是 8 分钟 20 秒。

程序清单 2-5 DisplayTime.java

```

1 import java.util.Scanner;
2
3 public class DisplayTime {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         // Prompt the user for input
7         System.out.print("Enter an integer for seconds: ");
8         int seconds = input.nextInt();
9
10        int minutes = seconds / 60; // Find minutes in seconds
11        int remainingSeconds = seconds % 60; // Seconds remaining
12        System.out.println(seconds + " seconds is " + minutes +
13            " minutes and " + remainingSeconds + " seconds");
14    }
15 }

```

Enter an integer for seconds: 500
500 seconds is 8 minutes and 20 seconds

line#	seconds	minutes	remainingSeconds
8	500		
10		8	
11			20

nextInt() 方法 (第 8 行) 读取 seconds 的整数值。第 10 行使用 seconds/60 获取分钟数。第 11 行 (seconds%60) 获取在减去分钟数之后得到的剩余秒数。

操作符 + 和 - 可以是一元的也可以是二元的。一元操作符仅有一个操作数; 而二元操作符有两个操作数。例如, 在 -5 中, 负号 (-) 可以认为是一元操作符, 是对正数 5 取负数值, 而在表达式 4-5 中, 负号 (-) 是二元操作符, 是从 4 中减去 5。

2.9.4 幂运算

使用方法 Math.pow(a,b) 来计算 a^b 。pow 方法定义在 Java API 的 Math 类中。运用语法 Math.pow(a,b) 可以调用 (比如, Math.pow(2,3)) 该方法, 并将返回结果 a^b (2^3)。这里, a 和 b 是 pow 方法的参数, 而数值 2 和 3 是调用方法时的真实值。比如:

```
System.out.println(Math.pow(2, 3)); // Displays 8.0
System.out.println(Math.pow(4, 0.5)); // Displays 2.0
System.out.println(Math.pow(2.5, 2)); // Displays 6.25
System.out.println(Math.pow(2.5, -2)); // Displays 0.16
```

第 5 章将介绍关于方法的更多细节。现在, 你只需要知道如何通过调用 pow 方法来执行幂运算。

复习题

2.10 找到最大和最小的 byte、short、int、long、float 以及 double。这些数据类型中, 哪个需要的内存最小?

2.11 给出以下求余计算的结果。

```
56 % 6
78 % -4
-34 % 5
-34 % -5
5 % 1
1 % 5
```

2.12 假设今天是周二, 100 天后将是周几?

2.13 25/4 的结果是多少? 如果你希望得到浮点数结果, 如何重写表达式?

2.14 给出以下代码的结果:

```
System.out.println(2 * (5 / 2 + 5 / 2));
System.out.println(2 * 5 / 2 + 2 * 5 / 2);
System.out.println(2 * (5 / 2));
System.out.println(2 * 5 / 2);
```

2.15 下面的语句正确吗? 如果正确的话, 给出输出。

```
System.out.println("25 / 4 is " + 25 / 4);
System.out.println("25 / 4.0 is " + 25 / 4.0);
System.out.println("3 * 2 / 4 is " + 3 * 2 / 4);
System.out.println("3.0 * 2 / 4 is " + 3.0 * 2 / 4);
```

2.16 写一个显示 2^{3^5} 的计算结果的语句。

2.17 假设 m 和 r 是整数。编写一个 Java 表达式, 使得 mr^2 可以得到一个浮点数类型的结果。

2.10 数值型直接量

 要点提示: 一个直接量 (literal) 是一个程序中直接出现的常量值。

例如, 下面的语句中 34 和 0.305 都是直接量:

```
int numberOfYears = 34;
double weight = 0.305;
```

2.10.1 整型直接量

只要整型直接量与整型变量相匹配，就可以将整型直接量赋值给该整型变量。如果直接量太大，超出该变量的存储范围，就会出现编译错误。例如：语句 `byte b=128` 就会造成一个编译错误，因为 `byte` 型变量存放不下 128（注意：`byte` 型变量的范围是 $-128 \sim 127$ ）。

整型直接量默认认为是 `int` 型的，它的值在 $-2^{31}(-2\ 147\ 483\ 648) \sim 2^{31}-1(2\ 147\ 483\ 647)$ 。为了表示一个 `long` 型的整型直接量，需要在其后追加字母 `L` 或 `l`（例如：`2147483648L`）。为了在 `Java` 程序中表示整数 `2147483648`，必须将它写成 `2147483648L` 或者 `2147483648l`，因为 `2147483648` 超出了 `int` 型的范围。推荐使用 `L`，因为 `l` (`L` 的小写) 很容易与 `1` (数字 `1`) 混淆。

注意：默认情况下，整型直接量是一个十进制整数。要表示一个二进制整数直接量，使用 `0b` 或者 `0B` (零 `B`) 开头；表示一个八进制整数直接量，就用 `0` (零) 开头，而要表示一个十六进制整数直接量，就用 `0x` 或 `0X` (零 `x`) 开头。例如，

```
System.out.println(0B1111); // Displays 15
System.out.println(07777); // Displays 4095
System.out.println(0xFFFF); // Displays 65535
```

十六进制数、二进制数和八进制数都将在附录 F 中介绍。

2.10.2 浮点型直接量

浮点型直接量带小数点，默认情况下是 `double` 型的。例如：`5.0` 被认为是 `double` 型而不是 `float` 型。可以通过在数字后面加字母 `f` 或 `F` 表示该数为 `float` 型直接量，也可以在数字后面加 `d` 或 `D` 表示该数为 `double` 型直接量。例如：可以使用 `100.2f` 或 `100.2F` 表示 `float` 型值，用 `100.2d` 或 `100.2D` 表示 `double` 型值。

注意：`double` 型值比 `float` 型值更精确。例如：

```
System.out.println("1.0 / 3.0 is " + 1.0 / 3.0);
```

显示结果为：`1.0/3.0 is 0.3333333333333333` (小数点后 16 位)。

```
System.out.println("1.0F / 3.0F is " + 1.0F / 3.0F);
```

显示结果为：`1.0F/3.0F is 0.33333334` (小数点后 8 位)。

一个 `float` 值有 7 到 8 位小数位，一个 `double` 值有 15 到 17 位小数位。

2.10.3 科学记数法

浮点型直接量也可以用 $a \times 10^b$ 形式的科学记数法表示，例如，`123.456` 的科学记数法形式是 1.23456×10^2 ，`0.012 345 6` 的科学记数法是 1.23456×10^{-2} 。一种特定的语法可以用于表示科学记数法的数值。例如， 1.23456×10^2 可以写成 `1.23456E2` 或者 `1.23456E+2`，而 1.23456×10^{-2} 等于 `1.23456E-2`。`E` (或 `e`) 表示指数，既可以是上写的也可以是小写的。

注意：`float` 型和 `double` 型都是用来表示带有小数点的数。为什么把它们称为浮点数呢？因为这些数都是以科学记数法的形式进行内部存储的。当一个像 `50.534` 的数被转换成科学记数法的形式时，它就是 `5.0534E+1`，它的小数点就移到 (即浮动到) 一个新的位置。

注意：为了提高可读性，`Java` 允许在数值直接量的两个数字间使用下划线。例如，下面的直接量是正确的：

```
long ssn = 232_45_4519;
long creditCardNumber = 2324_4545_4519_3415L;
```

然而, 45_ 和 _45 是不正确的。下划线必须置于两个数字间。

复习题

2.18 在 float 和 double 类型的变量中保存了多少个精确位?

2.19 以下哪些是正确的浮点数类型直接量?

12.3, 12.3e+2, 23.4e-2, -334.4, 20.5, 39F, 40D

2.20 以下哪些和 52.534 是等价的?

5.2534e+1, 0.52534e+2, 525.34e-1, 5.2534e+0

2.21 以下哪些是正确的直接量?

5_2534e+1, _2534, 5_2, 5_

2.11 表达式求值以及操作符优先级

要点提示: Java 表达式的求值和数学表达式求值是一样的。

用 Java 编写数值表达式就是使用 Java 操作符对算术表达式进行直接的翻译。例如, 下述算术表达式:

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

可以翻译成如下所示的 Java 表达式:

```
(3 + 4 * x) / 5 - 10 * (y - 5) * (a + b + c) / x +
9 * (4 / x + (9 + x) / y)
```

尽管 Java 有自己在后台计算表达式的方法, 但是, Java 表达式的结果和它对应的算术表达式的结果是一样的。因此, 可以放心地将算术运算规则应用在计算 Java 表达式上。首先执行的是包括在圆括号里的运算。圆括号可以嵌套, 嵌套时先计算内层括号。当一个表达式中有多于一个操作符时, 以下操作符的优先级规则用于确定计算的次序:

- 乘法、除法和求余运算首先计算。如果表达式中包含若干个乘法、除法和求余操作符, 可按照从左到右的顺序执行。
- 最后执行加法和减法运算。如果表达式中包含若干个加法和减法操作符, 则按照从左到右的顺序执行。

下面是一个如何计算表达式的例子:

```
3 + 4 * 4 + 5 * (4 + 3) - 1
      ↑
      (1) 圆括号里的先计算
3 + 4 * 4 + 5 * 7 - 1
      ↑
      (2) 乘法
3 + 16 + 5 * 7 - 1
      ↑
      (3) 乘法
3 + 16 + 35 - 1
      ↑
      (4) 加法
19 + 35 - 1
      ↑
      (5) 加法
54 - 1
      ↑
      (6) 减法
53
```

程序清单 2-6 给出了利用公式 $celsius = \left(\frac{5}{9}\right) (fahrenheit - 32)$ 将华氏温度转换成摄氏温度的程序。

程序清单 2-6 FahrenheitToCelsius.java

```

1 import java.util.Scanner;
2
3 public class FahrenheitToCelsius {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6
7         System.out.print("Enter a degree in Fahrenheit: ");
8         double fahrenheit = input.nextDouble();
9
10        // Convert Fahrenheit to Celsius
11        double celsius = (5.0 / 9) * (fahrenheit - 32);
12        System.out.println("Fahrenheit " + fahrenheit + " is " +
13            celsius + " in Celsius");
14    }
15 }

```

Enter a degree in Fahrenheit: 100
 Fahrenheit 100.0 is 37.7777777777778 in Celsius

line#	fahrenheit	celsius
8	100	
11		37.7777777777778

使用除法时要特别小心。在 Java 中，两个整数相除商为整数。在第 11 行，将 $\frac{5}{9}$ 转换为 $5.0/9$ 而不再是 $5/9$ ，因为在 Java 中 $5/9$ 的结果是 0。

复习题

2.22 如何在 Java 中表达以下算术表达式？

- $\frac{4}{3(r+34)} - 9(a+bc) + \frac{3+d(2+a)}{a+bd}$
- $5.5 \times (r + 2.5)^{2.5+r}$

2.12 示例学习：显示当前时间

要点提示：可以通过调用 `System.currentTimeMillis()` 返回当前时间。

本节的问题是开发一个以 GMT（格林威治标准时间）来显示当前时间的程序，以小时：分钟：秒的格式来显示，例如 13:19:8。

`System` 类中的方法 `currentTimeMillis` 返回从 GMT 1970 年 1 月 1 日 00:00:00 开始到当前时刻的毫秒数，如图 2-2 所示。时间戳是时间开始计时的点，因为 1970 年是 UNIX 操作系统正式发布的时间，所以这一时间也称为 UNIX 时间戳（UNIX epoch）。

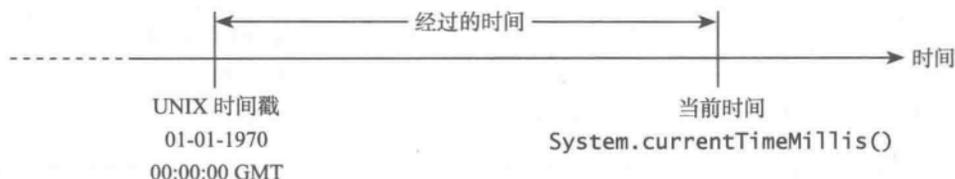


图 2-2 `System.currentTimeMillis()` 返回自 UNIX 时间戳以来的毫秒数

可以使用这个方法获取当前时间，然后按照如下步骤计算出当前的秒数、分钟数和小时数：

1) 调用 `System.currentTimeMillis()` 方法获取 1970 年 1 月 1 日午夜到现在的毫秒数 (例如：1203183086328 毫秒)，并存放在变量 `totalMilliseconds` 中。

2) 通过将总毫秒数 `totalMilliseconds` 除以 1000 得到总秒数 `totalSeconds` (例如：1203183086328 毫秒 / 1000 = 1203183068 秒)。

3) 通过 `totalSeconds % 60` 得到当前的秒数 (例如：1203183068 秒 % 60 = 8，这个值就是当前秒数)。

4) 通过将 `totalSeconds` 除以 60 得到总的分钟数 `totalMinutes` (例如：1203183068 秒 / 60 = 20053051 分钟)。

5) 通过 `totalMinutes % 60` 得到当前分钟数 (例如：20053051 分钟 % 60 = 31，这个值就是当前分钟数)。

6) 通过将总分钟数 `totalMinutes` 除以 60 获得总的小时数 `totalHours` (例如：20053051 分钟 / 60 = 334217 小时)。

7) 通过 `totalHours % 24` 得到当前的小时数 (例如：334217 小时 % 24 = 17，该值就是当前小时数)。

程序清单 2-7 给出完整的程序。

程序清单 2-7 ShowCurrentTime.java

```
1 public class ShowCurrentTime {
2     public static void main(String[] args) {
3         // Obtain the total milliseconds since midnight, Jan 1, 1970
4         long totalMilliseconds = System.currentTimeMillis();
5
6         // Obtain the total seconds since midnight, Jan 1, 1970
7         long totalSeconds = totalMilliseconds / 1000;
8
9         // Compute the current second in the minute in the hour
10        long currentSecond = totalSeconds % 60;
11
12        // Obtain the total minutes
13        long totalMinutes = totalSeconds / 60;
14
15        // Compute the current minute in the hour
16        long currentMinute = totalMinutes % 60;
17
18        // Obtain the total hours
19        long totalHours = totalMinutes / 60;
20
21        // Compute the current hour
22        long currentHour = totalHours % 24;
23
24        // Display results
25        System.out.println("Current time is " + currentHour + ":"
26            + currentMinute + ":" + currentSecond + " GMT");
27    }
28 }
```

Current time is 17:31:8 GMT

line#	4	7	10	13	16	19	22
variables							
totalMilliseconds	1203183068328						
totalSeconds		1203183068					
currentSecond			8				
totalMinutes				20053051			
currentMinute					31		
totalHours						334217	
currentHour							17

第4行调用 `System.currentTimeMillis()` 获得一个 `long` 类型的以毫秒为单位的当前时间值。因此，在该程序中的所有变量都被声明为 `long` 型。秒、分钟、小时数都从当期时间中运用 `/` 和 `%` 操作符抽取得到（第6~22行）。

在一个示例运行中，仅一位的数字8显示为秒数，而希望的输出是08。这可以运用一个方法来修复，该方法可以将单位数字格式化为加一位前缀0（参见编程练习题6.37）。

复习题

2.23 如何获得当前的秒、分钟以及小时数？

2.13 增强赋值操作符

要点提示：操作符 `+`、`-`、`*`、`/`、`%` 可以结合赋值操作符形成增强操作符。

经常会出现变量的当前值被使用、修改，然后再重新赋值给该变量的情况。例如，下面语句将变量 `count` 加1。

```
count = count + 1;
```

Java 允许使用增强赋值操作符来结合赋值和加法操作符的功能。例如，上面的语句可以写成：

```
count += 1;
```

符号 `+=` 称为加法赋值操作符（addition assignment operator）。其他简捷赋值操作符如表2-4中所示。

表 2-4 简捷赋值操作符

操作符	名称	示例	等价于
<code>+=</code>	加法赋值操作符	<code>i += 8</code>	<code>i = i + 8</code>
<code>-=</code>	减法赋值操作符	<code>i -= 8</code>	<code>i = i - 8</code>
<code>*=</code>	乘法赋值操作符	<code>i * = 8</code>	<code>i = i * 8</code>
<code>/=</code>	除法赋值操作符	<code>i / = 8</code>	<code>i = i / 8</code>
<code>%=</code>	求余赋值操作符	<code>i % = 8</code>	<code>i = i % 8</code>

增强赋值操作符在表达式中所有其他操作符计算完成后执行。例如：

```
x /= 4 + 5.5 * 1.5;
```

等同于

```
x = x / (4 + 5.5 * 1.5);
```

 **警告：**在增强操作符中是没有空格的。例如：`+ =`应该是`+=`。

 **注意：**就像赋值操作符(=)一样，操作符(+=、-=、*=、/=、%=)既可以构成赋值语句，也可以构成赋值表达式。例如，在下面的代码中，第1行的`x+=2`是一条语句，而在第2行中它就是一个表达式：

```
x += 2; // Statement
System.out.println(x += 2); // Expression
```

复习题

2.24 给出以下代码运行的结果：

```
double a = 6.5;
a += a + 1;
System.out.println(a);
a = 6;
a /= 2;
System.out.println(a);
```

2.14 自增和自减操作符

 **要点提示：**自增操作符(++)和自减操作符(--)是对变量进行加1和减1的操作。

`++`和`--`是对变量进行自增1和自减1的简写操作符。由于这是许多编程任务中经常需要改变的值，所以这两个操作符使用起来很方便。例如，下面的代码是对`i`自增1，而对`j`自减1：

```
int i = 3, j = 3;
i++; // i becomes 4
j--; // j becomes 2
```

`i++`读为`i`加加，`i--`读为`i`减减。这些操作符分别称为后置自增操作符和后置自减操作符，因为操作符`++`和`--`放在变量后面。这些操作符也可以放在变量前面，比如：

```
int i = 3, j = 3;
++i; // i becomes 4
--j; // j becomes 2
```

`++i`将`i`增加1，`--j`将`j`减去1。这些操作符称为前置自增操作符和前置自减操作符。

如你所见，前面的例子中，`i++`和`++i`的效果，或者`i--`和`--i`的效果是一样的。然而，当用在表达式中不单纯只进行自增和自减时，它们就会产生不同的效果。表2-5描述了它们的不同，并且给出了示例。

表 2-5 自增和自减操作符

操作符	名称	说明	示例(假设 i=1)
<code>++var</code>	前置自增操作符	变量 var 的值加 1 且使用 var 增加后的新值	<code>int j=++i;</code>
<code>var++</code>	后置自增操作符	变量 var 的值加 1 但使用 var 原来的值	<code>int j=i++;</code>
<code>--var</code>	前置自减操作符	变量 var 的值减 1 且使用 var 减少后的新值	<code>int j=--i;</code>
<code>var--</code>	后置自减操作符	变量 var 的值减 1 但使用 var 原来的值	<code>int j=i--;</code>

下面是展示前置形式的 ++ (或者 --) 和后置形式的 ++ (或者 --) 的补充示例。考虑以下代码:

```
int i = 10;
int newNum = 10 * i++;      效果等同于
System.out.print("i is " + i
    + ", newNum is " + newNum);
```

```
int newNum = 10 * i;
i = i + 1;
```

```
i is 11, newNum is 100
```

在此例中, 首先对 i 自增 1, 然后返回 i 的旧值来参与乘法运算。这样, $newNum$ 的值就成为 100。如果如下所示将 $i++$ 换为 $++i$:

```
int i = 10;
int newNum = 10 * (++i);   效果等同于
System.out.print("i is " + i
    + ", newNum is " + newNum);
```

```
i = i + 1;
int newNum = 10 * i;
```

```
i is 11, newNum is 110
```

i 自增 1, 然后返回 i 的新值, 并参与乘法运算。这样, $newNum$ 的值就成为 110。

下面是另一个例子:

```
double x = 1.0;
double y = 5.0;
double z = x-- + (++y);
```

在这三行程序执行完之后, y 的值为 6.0, z 的值为 7.0, 而 x 的值为 0.0。

提示: 使用自增操作符和自减操作符可以使表达式更加简短, 但也会使它们比较复杂且难以读懂。应该避免在同一个表达式中使用这些操作符修改多个变量或多次修改同一个变量, 如: `int k = ++i + i`。

复习题

2.25 下面的说法哪个为真?

- 任何表达式都可以用作一个语句。
- 表达式 `x++` 可以用作一个语句。
- 语句 `x = x + 5` 也是一个表达式。
- `x = y = x = 0` 是非法的。

2.26 给出以下代码的输出:

```
int a = 6;
int b = a++;
System.out.println(a);
System.out.println(b);
a = 6;
b = ++a;
System.out.println(a);
System.out.println(b);
```

2.15 数值类型转换

要点提示: 通过显式转换, 浮点数可以被转换为整数。

可以完成两个不同类型操作数的二元运算吗? 当然可以。如果在一个二元运算中, 其中

一个操作数是整数，而另一个操作数是浮点数，Java 会自动地将整数转换为浮点值。这样， $3*4.5$ 就成了 $3.0*4.5$ 。

总是可以将一个数值赋给支持更大数值范围类型的变量，例如，可以将 `long` 型的值赋给 `float` 型变量。但是，如果不进行类型转换，就不能将一个值赋给范围较小类型的变量。类型转换是一种将一种数据类型的值转换成另一种数据类型的操作。将一个小范围类型的变量转换为大范围类型的变量称为拓宽类型 (`widening a type`)，把大范围类型的变量转换为小范围类型的变量称为缩窄类型 (`narrowing a type`)。Java 将自动拓宽一个类型，但是，缩窄类型必须显式完成。

类型转换的语法要求目标类型放在括号内，紧跟其后的是要转换的变量名或值。例如，下面的语句：

```
System.out.println((int)1.7);
```

显示结果为 1。当 `double` 型值被转换为 `int` 型值时，小数部分被截去。

下面的语句：

```
System.out.println((double)1 / 2);
```

显示结果为 0.5，因为 1 首先被转换为 1.0，然后用 2 除 1.0。但是，语句

```
System.out.println(1 / 2);
```

显示结果为 0，因为 1 和 2 都是整数，结果也应该是整数。

🔧 **警告：** 如果要将一个值赋给一个范围较小类型的变量，例如：将 `double` 型的值赋给 `int` 型变量，就必须进行类型转换。如果在这种情况下没有使用类型转换，就会出现编译错误。使用类型转换时必须小心，丢失的信息也许会导致不精确的结果。

🔧 **注意：** 类型转换不改变被转换的变量。例如，下面代码中的 `d` 在类型转换之后值不变：

```
double d = 4.5;
int i = (int)d; // i becomes 4, but d is still 4.5
```

🔧 **注意：** Java 中，`x1 op= x2` 形式的增强赋值表达式，执行为 `x1 = (T)(x1 op x2)`，这里 `T` 是 `x1` 的类型。因此，下面代码是正确的。

```
int sum = 0;
sum += 4.5; // sum becomes 4 after this statement
sum += 4.5 等价于 sum = (int)(sum + 4.5).
```

🔧 **注意：** 将一个 `int` 型变量赋值给 `short` 型或 `byte` 型变量，必须显式地使用类型转换。例如，下述语句就会有一个编译错误：

```
int i = 1;
byte b = i; // Error because explicit casting is required
```

然而，只要整型直接量是在目标变量允许的范围內，那么将整型直接量赋给 `short` 型或 `byte` 型变量时，就不需要显式的类型转换（请参考 2.10 节）。

程序清单 2-8 中的程序将营业税值显示为小数点后两位。

程序清单 2-8 SalesTax.java

```
1 import java.util.Scanner;
2
3 public class SalesTax {
4     public static void main(String[] args) {
```

提供各种书籍的pd电子版代找服务，如果你找不到自己想要的书的pdf电子版，我们可以帮您找到，如有需要，请联系QQ1779903665.

PDF代找说明：

本人可以帮助你找到你要的PDF电子书，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签索引和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我QQ1779903665。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ，大部分我都可以找到，而且每本100%带书签索引目录。因PDF电子书都有版权，请不要随意传播，如果您有经济购买能力，请尽量购买正版。

声明：本人只提供代找服务，每本100%索引书签和目录，因寻找pdf电子书有一定难度，仅收取代找费用。如因PDF产生的版权纠纷，与本人无关，我们仅仅只是帮助你寻找到你要的pdf而已。