国外经典教材·计算机科学与技术

PEARSON Addison Wesley

Java完美编程

（第2版 影印版）

Absolute Java, Second Edition

- 双语教学推荐用书
- 原汁原味，真切体会原语精髓
- 浅显易懂，迅速吸取科技新知

（美） Walter Savitch 著

清华大学出版社

国外经典教材·计算机科学与技术

Java 完 美 编 程

（第2版 影印版）

（美）Walter Savitch 著

江苏工业学院图书馆
藏书章

清华大学出版社
北 京

ABSOLUTE Java

WALTER SAVITCH

University of California, San Diego

Boston San Francisco NewYork
London Toronto Sydney Tokyo Singapore Madrid
Mexico City Munich Paris Cape Town Hong Kong Montreal

English reprint edition copyright © 2006 by **PEARSON EDUCATION ASIA LIMITED and TSINGHUA UNIVERSITY PRESS.**

Original English language title from Proprietor's edition of the Work.
Original English language title: Absolute Java, Second Edition, **by** Walter Savitch, Copyright © 2006
All Rights Reserved.
Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Pearson Education, Inc.
This edition is authorized for sale and distribution only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong, Macao SAR and Taiwan).
本书影印版由 Pearson Education, Inc.授权给清华大学出版社出版发行。

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macao SAR).
仅限于中华人民共和国境内(不包括中国香港、澳门特别行政区和中国台湾地区)销售发行。

北京市版权局著作权合同登记号　图字：01-2005-5287

版权所有，翻印必究。举报电话：010-62782989　13501256678　13801310933
本书封面贴有 Pearson Education**(培生教育出版集团)激光防伪标签，无标签者不得销售。**

图书在版编目(CIP)数据
Java完美编程(第2版 影印版)= Absolute Java, Second Edition, /(美)萨维奇(Savitch, W.)著. —影印本. —北京：清华大学出版社，2006.9
国外经典教材 • 计算机科学与技术
ISBN 7-302-13210-0
Ⅰ. J… Ⅱ. 萨… Ⅲ. Java语言—程序设计—教材—英文 Ⅳ. TP312
中国版本图书馆CIP数据核字(2006)第064845号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
文稿编辑：文开棋
封面设计：久久度文化
印 刷 者：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×230　**印张**：78.25　**字数**：1001千字
版　　次：2006年9月第1版　2006年9月第1次印刷
书　　号：ISBN 7-302-13210-0/TP·8349
印　　数：1～3500
定　　价：109.00元(含1张光盘)

Acquisitions Editor	Matt Goldstein
Project Editor	Katherine Harutunian
Composition and Art	Argosy Publishing
Copyeditor	Ginny Kaczmarek
Proofreader	Kim Cofer
Indexer	Larry Sweazy
Text and Cover Design	Leslie Haimes/Joyce Cosentino Wells
Cover Photo	© 2005 Ryan McVay/Photodisk
Design Manager	Joyce Cosentino Wells
Prepress and Manufacturing	Caroline Fell
Production Assistant	Sarah Bartlett
Media Producer	Bethany Tidd

Access the latest information about Addison-Wesley titles from our World Wide Web site:
http://www.aw-bc.com/computing

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed in initial caps or all caps.

The programs and applications presented in this book have been included for their instructional value. They have been tested with care, but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs or applications.

Library of Congress Cataloging-in-Publication Data
Savitch, Walter J., 1943-
Absolute Java / Walter Savitch -- 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-321-33024-2
1. Java (Computer program language) I. Title.
QA76.73.J38S265 2005
005.13'3--dc22

2005002885

Copyright © 2006 by Pearson Education, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

ISBN 0-321-33024-2
345678910-QWT-080706

此为试读，需要完整PDF请访问：www.ertongbook.com

C++面向对象程序设计
（第5版）

Walter Savitch著　周靖译
ISBN 7-302-11818-3

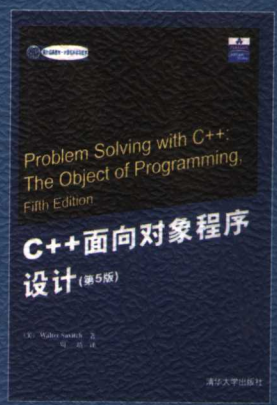
世界各地数十万读者可以证明，Walter Savitch教授的这本书是学习程序设计的理想入门教材。第5版沿袭以前通俗易懂、饶有趣味的写作风格，根据最新的ANSI/ISO标准进行修订，对命名空间的内容进行扩充，同时还新增一章的内容，专门介绍标准模板库。自测题和编程项目、编程提示、范例和编程陷阱，以及各章中突出强调关键主题的总结框，都有所增加。

"我正在修'程序设计入门'课程，《C++面向对象程序设计》使我对这门课程着了迷。它给我留下了深刻的印象。我喜欢它的写作风格，简直无法用言语来表达我对作者的感激之情。"

——Syed Ali，科罗拉多大学，圆石市

"Walter Savitch天赋秉异。他对一个复杂主题了如指掌，并知道如何用深入浅出的语言来解释它。"

——Ken Morris，大都会州立学院，丹佛市



Preface

This book is designed to serve as a textbook and reference for programming in the Java language. Although it does include programming techniques, it is organized around the features of the Java language rather than any particular curriculum of programming techniques. The main audience I had in mind when writing this book was undergraduate students who have not had extensive programming experience with the Java language. As such, it would be a suitable Java text or reference for either a first programming course or a later computer science course that uses Java. This book is designed to accommodate a wide range of users. The beginning chapters are written at a level that is accessible to beginners, while the boxed sections of those chapters serve to quickly introduce more experienced programmers to basic Java syntax. Later chapters are still designed to be accessible, but are written at a level suitable for students who have progressed to these more advanced topics.

The Java coverage in this book is very complete including extensive coverage of new version 5.0 features.

CHANGES IN THIS EDITION

If you have not used the first edition of this text, you can skip this subsection. If you have used the first edition, this subsection will tell you how this second edition differs from the first.

For instructors, the transition from the first to this second edition is easy. You can teach the same course, presenting basically the same topics in the same order with only very minor changes in the material covered. The largest required change is that this edition uses the `Scanner` class, new in Java version 5.0, for keyboard input. This edition does include a number of additional topics that offer an opportunity to add to an existing course. The most significant of the new topics is generic programming using type parameters. For the most part, the new topics are a direct result of the changes and additions in version 5.0 of Java.

This edition has added the following topics and updates, all new with Java version 5.0: the new Scanner class for keyboard and file input; extensive coverage of generic programming using type parameters; coverage of the Swing library has been updated to accommodate 5.0 changes; automatic boxing and unboxing; the new for-each loop (also called the enhanced for-loop); enumerated types; static imports; variable length argument lists for methods; covariant return types.

This edition also updates the programming projects and adds numerous new programming projects.

NO NONSTANDARD SOFTWARE

Only classes in the standard Java libraries are used. No nonstandard software is used anywhere in the book.

JAVA 5.0 COVERAGE

This edition has been updated and expanded to include full coverage of Java features new in version 5.0. Treatment of keyboard input and of file input has been updated to use the new Scanner class. A new chapter on generic programming has been added. Other chapters have been updated to use generic programming. In addition, the following topics and updates, all new with Java version 5.0, have been added: coverage of the Swing library has been updated to accommodate 5.0 changes; automatic boxing and unboxing; the new for-each loop (also called the enhanced for-loop); enumerated types; static imports; variable length argument lists for methods; covariant return types.

OBJECT-ORIENTED PROGRAMMING

This book gives extensive coverage of encapsulation, inheritance, and polymorphism as realized in the Java language. The chapters on Swing GUIs provide coverage of and extensive practice with event driven programming. A chapter on UML and patterns gives additional coverage of OOP-related material.

FLEXIBILITY IN TOPIC ORDERING

This book allows instructors wide latitude in reordering the material. This is important if a book is to serve as a reference. It is also in keeping with my philosophy of writing books that accommodate themselves to an instructor's style rather than tying the instructor to an author's personal preference of topic ordering. With this in mind, each chapter has a prerequisite section at the start of the chapter; it explains what material must be covered before doing each section of the chapter. Starred sections, which are explained next, further add to flexibility.

STARRED SECTIONS

Each chapter has a number of starred (☆) sections, which can be considered optional sections. These sections contain material that beginners might find difficult and that can be omitted or delayed without hurting the continuity of the text. It is hoped that eventually the reader would return and cover this material. For more advanced students, the starred sections should not be viewed as optional.

ACCESSIBLE TO STUDENTS

It is not enough for a book to present the right topics in the right order. It is not even enough for it to be clear and correct when read by an instructor or other expert. The material needs to be presented in a way that is accessible to the person who does not yet know the material. Like my other textbooks that have proven to be very popular with students, this book was written to be friendly and accessible to the student.

SUMMARY BOXES

Each major point is summarized in a short boxed section. These boxed sections are spread throughout each chapter. They serve as summaries of the material, as a quick reference source, and as a way to quickly learn the Java syntax for features the reader knows about in general but for which he or she needs to know the Java particulars.

SELF-TEST EXERCISES

Each chapter contains numerous Self-Test Exercises at strategic points in the chapter. Complete answers for all the Self-Test Exercises are given at the end of each chapter.

OTHER FEATURES

Pitfall sections, programming tip sections, and examples of complete programs with sample I/O are given throughout each chapter. Each chapter ends with a summary section and a collection of programming projects suitable to assign to students.

CODEMATE ONLINE TUTORIAL RESOURCE

CodeMate is an online resource that provides tutorial help and evaluation of student work on programming projects. The code displays and selected programming projects in this edition have been fully integrated into CodeMate. Using CodeMate, a student can get hints on programming projects, write and compile the project, and receive feedback on how to address compiler errors messages, and all this can be done over the Internet from any computer with Internet access. Instructors can track each student's progress in the course's programming projects. A complimentary subscription is

offered when an access code is bundled with a new copy of this text. Subscriptions may also be purchased online. For more information on CodeMate, go to

```
http://www.aw-bc.com/codemate
```

SUPPORT MATERIAL

The following support materials are available to all users of this book:

- Self-check quizzes.
- Source code from the book.
- A free copy of the Java version 5.0 Software Development Kit, which includes a Java 5.0 compiler.

The following resources are available to qualified instructors only. Please contact your local sales representative or send e-mail to aw.cse@aw.com for access information:

- Instructor access to Addison-Wesley's CodeMate
- Instructor's Manual with Solutions
- Computerized Test Bank
- PowerPoint Slides

CD

The CD that accompanies this book contains:

- The source code for all the code displays in the book.
- A copy of the Java version 5.0 Software Development Kit, which includes a Java 5.0 compiler.
- A trial copy of the TextPad integrated development environment (IDE) for Windows.

OBTAINING JAVA

The CD that accompanies this book contains a free copy of the Java version 5.0 Software Development Kit, which includes a Java 5.0 compiler. You also need an IDE (integrated development environment) or at least an editor. There are a number of elaborate IDEs available. However, we've found that a simple IDE works best for beginning student since it presents fewer distractions from the task of designing programs. We recommend the TextPad environment or any similarly simple IDE. A free trial copy of TextPad for Windows is included on the CD that accompanies this book

ACKNOWLEDGMENTS

Numerous individuals have contributed invaluable help and support in making this book happen: My former editor Susan Hartman at Addison-Wesley first conceived of the idea for this book and worked with me on the first editions; My current editor Matt Goldstein provided support and inspiration for getting this second edition reviewed, revised, and out the door; Katherine Harutunian, Bethany Tidd, Sarah Bartlett, Joyce Wells, Michelle Brown, and the other fine people at Addison-Wesley also provided valuable support and encouragement.

Meghan James and Daniel Rausch at Argosy Publishing worked tirelessly, expertly, and cheerfully to get the book through production on schedule. Patty Mahtani headed the production team at Addison-Wesley. She was an inspiration as well as an indispensable help. Working with Patty transformed drudgery into joy.

The following reviewers provided corrections and suggestions for this second edition. Their contributions were a great help. I thank them all. In alphabetical order they are:

Kevin Bierre	Rochester Institute of Technology
Stephen Chandler	NW Shoals Community College
Massoud Ghyam	University of Southern California
Nigel Gwee	Louisiana State University
Sridhar P. Nerur	The University of Texas at Arlington
David Primeaux	VA Commonwealth University
Riyaz Sikora	The University of Texas at Arlington
Ronald F. Taylor	Wright State University

The following reviewers provided corrections and suggestions for the first edition of this book. Their contributions continue into this edition. I thank them all. In alphabetical order they are:

Jim Adams	Chandler-Gilbert Community College
Gerald W. Adkins	Georgia College & State University
Dr. Bay Arinze	Drexel University
Prof. Richard G. Baldwin	Austin Community College
Jon Bjornstad	Gavilan College
Adrienne Decker	University of Buffalo
Arthur Geis	College of DuPage
Judy Hankins	Middle Tennessee State University
Chris Howard	DeVry University

此为试读，需要完整PDF请访问：www.ertongbook.com

Eliot Jacobson	University of California, Santa Barbara
Balaji Janamanchi	Texas Tech University
Suresh Kalathur	Boston University
Dr. Clifford R. Kettemborough	IT Consultant and Professor
Frank Levey	Manatee Community College
Xia Lin	Drexel University
Mark M. Meysenburg	Doane College
Hoang M. Nguyen	Deanza College
Prof. Bryson R. Payne	North Georgia College & State University
W. Brent Seales	University of Kentucky
Jeff Six	University of Delaware
Xueqing (Clare) Tang	Governors State University
Natalie S. Wear	University of South Florida
Dale Welch	University of West Florida
Wook-Sung Yoo	Gannon University

Kenrick Mock (University of Alaska) updated the programming projects including many new projects and new solutions to old problems. I thank him for a truly excellent job.

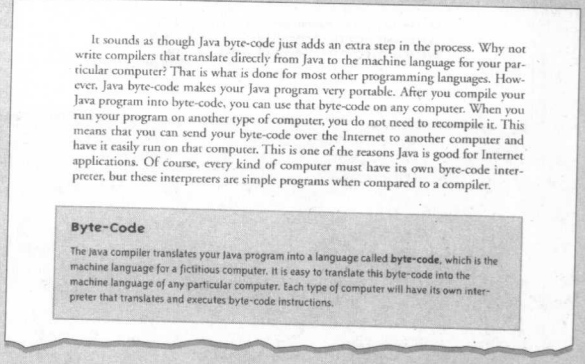
A special thanks to Rick Ord who reviewed the entire first edition and the updated sections of this second edition. He provided detailed and very helpful suggestions for improvements in the book. His insights contributed greatly to just about every part of the book.

Walter Savitch
http://www-cse.ucsd.edu/users/savitch/
wsavitch@ucsd.edu

Feature Walkthrough

Summary Boxes

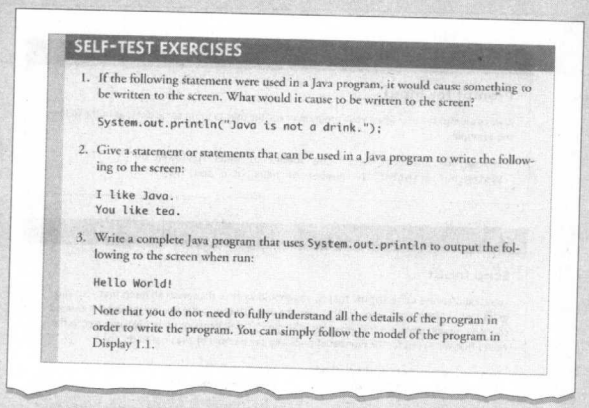
These boxes provide a brief synopsis of major points in each chapter, both highlighting and reinforcing core concepts throughout the book. Readers will find them to be a handy, quick reference for Java syntax and features.



It sounds as though Java byte-code just adds an extra step in the process. Why not write compilers that translate directly from Java to the machine language for your particular computer? That is what is done for most other programming languages. However, Java byte-code makes your Java program very portable. After you compile your Java program into byte-code, you can use that byte-code on any computer. When you run your program on another type of computer, you do not need to recompile it. This means that you can send your byte-code over the Internet to another computer and have it easily run on that computer. This is one of the reasons Java is good for Internet applications. Of course, every kind of computer must have its own byte-code interpreter, but these interpreters are simple programs when compared to a compiler.

Byte-Code

The Java compiler translates your Java program into a language called **byte-code**, which is the machine language for a fictitious computer. It is easy to translate this byte-code into the machine language of any particular computer. Each type of computer will have its own interpreter that translates and executes byte-code instructions.



SELF-TEST EXERCISES

1. If the following statement were used in a Java program, it would cause something to be written to the screen. What would it cause to be written to the screen?

```
System.out.println("Java is not a drink.");
```

2. Give a statement or statements that can be used in a Java program to write the following to the screen:

```
I like Java.
You like tea.
```

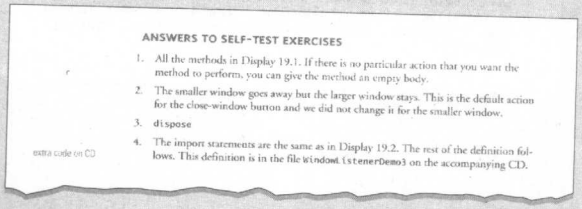
3. Write a complete Java program that uses `System.out.println` to output the following to the screen when run:

```
Hello World!
```

Note that you do not need to fully understand all the details of the program in order to write the program. You can simply follow the model of the program in Display 1.1.

Self-Test Exercises and Answers

Strategically placed within each chapter, Self-Test Exercises offer readers an opportunity to assess their mastery of key topics.



ANSWERS TO SELF-TEST EXERCISES

1. All the methods in Display 19.1. If there is no particular action that you want the method to perform, you can give the method an empty body.
2. The smaller window goes away but the larger window stays. This is the default action for the close-window button and we did not change it for the smaller window.
3. `dispose`
4. The import statements are the same as in Display 19.2. The rest of the definition follows. This definition is in the file `WindowListenerDemo3` on the accompanying CD.

extra code on CD

Detailed answers are provided at the end of the chapter.

Display 2.6 Keyboard Input Demonstration

```
1  import java.util.Scanner;
2  public class ScannerDemo
3  {
4     public static void main(String[] args)
5     {
6        Scanner keyboard = new Scanner(System.in);

7        System.out.println("Enter the number of pods followed by");
8        System.out.println("the number of peas in a pod:");
9        int numberOfPods = keyboard.nextInt();
10       int peasPerPod = keyboard.nextInt();

11       int totalNumberOfPeas = numberOfPods*peasPerPod;

12       System.out.print(numberOfPods + " pods and ");
13       System.out.println(peasPerPod + " peas per pod.");
14       System.out.println("The total number of peas = "
15                                          + totalNumberOfPeas);
16    }
17 }
```

Makes the `Scanner` *class available to your program.*

Creates an object of the class `Scanner` *and names the object* `keyboard`.

Each reads one `int` *from the keyboard.*

Code Displays

There are abundant code listings throughout the text. Informal comments that explain potentially confusing or difficult portions appear alongside the code. Key lines are highlighted and color-coding visually distinguishes comments (green) and reserved words (blue).

Tips

These helpful hints instruct readers on best programming practices. The author explains the rationale behind these practices and includes suggestions on how to execute them effectively.

TIP

Prompt for Input

Always prompt the user when your program needs the user to input some data, as in the following example:

```
System.out.println("Enter the number of pods followed by");
System.out.println("the number of peas in a pod:");
```

TIP

Echo Input

You should always echo input. That is, you should write to the screen all input that your program receives from the keyboard. This way, the user can check that the input has been entered correctly. For example, the following two statements from the program in Display 2.9 echo the values that were read for the number of pods and the number of peas per pod:

```
System.out.print(numberOfPods + " pods and ");
System.out.println(peasPerPod + " peas per pod.");
```

It might seem that there is no need to echo input, because the user's input is automatically displayed on the screen as the user enters it. Why bother to write it to the screen a second time? The input might be incorrect even though it looks correct. For example, the user might type a comma instead of a decimal point or the letter O in place of a zero. Echoing the input can expose such problems.

PITFALL

Dealing with the Line Terminator, '\n'

The method `nextLine` of the class `Scanner` reads the *remainder* of a line of text *starting wherever the last keyboard reading left off*. For example, suppose you create an object of the class `Scanner` as follows:

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

and suppose you continue with the following code:

```
int n = keyboard.nextInt();
String s1 = keyboard.nextLine();
String s2 = keyboard.nextLine();
```

Now, assume that the input typed on the keyboard is:

```
2 heads are
better than
1 head.
```

This sets the value of the variable `n` to 2, that of the variable `s1` to `" heads are"`, and that of the variable `s2` to `"better than"`.

So far there are no problems, but suppose the input were instead

```
2
heads are better than
1 head.
```

You might expect the value of `n` to be set to 2, the value of the variable `s1` to `"heads are better than"`, and that of the variable `s2` to `"1 head."`. But that is not what happens.

What actually happens is that the value of the variable `n` is set to 2, that of the variable `s1` is set to the empty string, and that of the variable `s2` to `"heads are better than"`. The method `nextInt` reads the 2 but does not read the end-of-line character `'\n'`. So the first `nextLine` invocation reads the rest of the line that contains the 2. There is nothing more on that line (except for `'\n'`), so `nextLine` returns the empty string. The second invocation of `nextLine` begins on the next line and reads `"heads are better than"`.

When combining different methods for reading from the keyboard, you sometimes have to include an extra invocation of `nextLine` to get rid of the end of a line (to get rid of a `'\n'`). This is illustrated in Display 2.8.

Pitfalls

These sections warn of common mistakes that can trip up beginning programmers and offer advice on how to avoid them.

Examples

These sections usually feature a complete program that solves a specific problem. The code examples are lengthier than in the standard code displays and highlight useful features of Java.

EXAMPLE

A GUI with a Label and Color

Display 17.6 shows a class for GUIs with a label and a background color. We have already discussed the use of color for this window. The label is used to display the text string `"Close-window button works."` The label is created as follows:

```
JLabel aLabel = new JLabel("Close-window button works.");
```

The label is added to the `JFrame` with the method `add` as shown in the following line from Display 17.6:

```
add(aLabel);
```

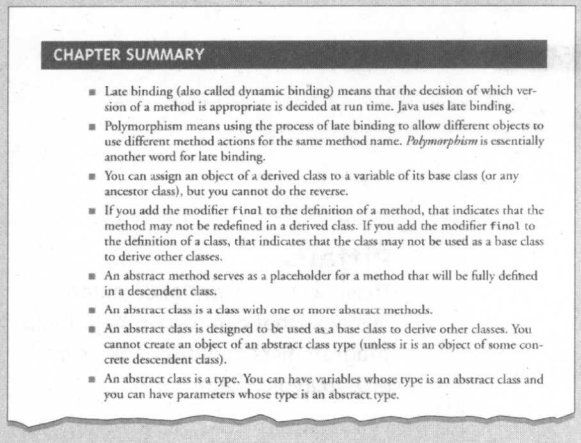
The GUI class `ColoredWindow` in Display 17.6 programs the close-window button as follows:

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

This way, when the user clicks the close-window button, the program ends. Note that if the program has more than one window, as it does in Display 17.6, and the user clicks the close-window button in any one window of the class `ColoredWindow`, then the entire program ends and all windows go away.

Note that we set the title of the `JFrame` by making it an argument to `super` rather than an argument to `setTitle`. This is another common way to set the title of a `JFrame`.

If you run the program `DemoColoredWindow` in Display 17.6, then the two windows will be placed one on top of the other. To see both windows, you need to use your mouse to move the top window.

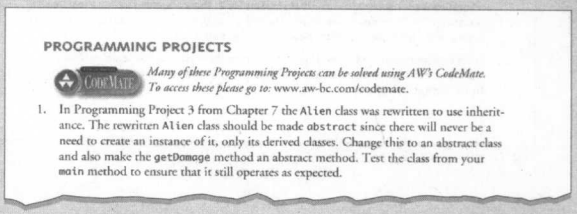


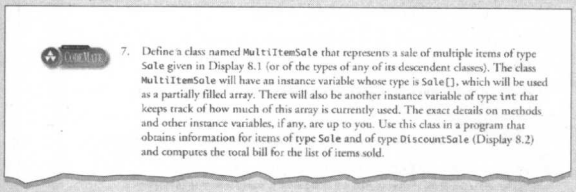
Chapter Summaries

This end-of-chapter tool provides a concise overview of the fundamental concepts presented in the chapter.

Programming Projects

Found at the end of each chapter, Programming Projects challenge readers to design and implement a Java program to solve a problem. Example solutions for all programming projects are available to instructors.





CodeMate

CodeMate brings end-of-chapter programming projects to life. Working online, students can view, compile, run, and edit select programming problems as well as all code listings from the textbook. Best of all, CodeMate's tutorial feedback helps students work through common programming errors, improving their programming skills. An automated gradebook allows instructors to assign CodeMate problems and track student progress online.

Brief Contents

此为试读,需要完整PDF请访问: www.ertongbook.com