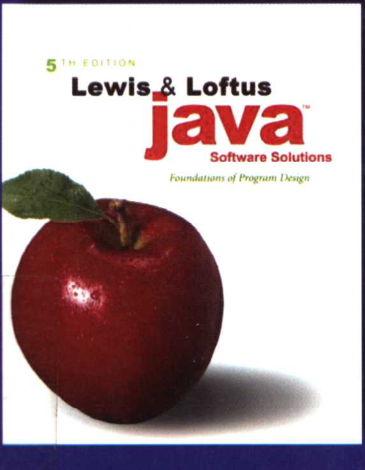
国外计算机科学教材系列

Java 程序设计教程

（第五版）

Java Software Solutions

Foundations of Program Design, Fifth Edition



英文版

[美] John Lewis
William Loftus 著

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

国外计算机科学教材系列

Java 程序设计教程

（第五版）

（英文版）

Java Software Solutions

Foundations of Program Design

Fifth Edition

[美] John Lewis
William Loftus 著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

本书是一本讲解Java原理与Java编程的畅销教材，书中的内容可为学习编程技术的读者打下坚实的基础，从而设计出良好的面向对象软件。通过讲解各种真实世界的编程实例，作者在书中强调了如何创建问题解决方案及如何应用设计技巧。利用早期的对象方法（使用与编写相关的类）及面向对象设计的过程，学生们可以在学习编写对象之前先学习如何使用它们。通过易于理解的和准确的写作风格，本书向读者展示了编程的各种概念，并覆盖了图形与GUI等最新主题。书中包含了大量的编程实例，同时在每章结尾给出了非常有价值的编程项目练习。

本书的概念清楚、逻辑性强、内容新颖，可作为大专院校计算机软件专业与计算机应用专业学生的教材和参考书，也可供计算机工程技术人员参考。

版权贸易合同登记号 图字：01-2007-3606

图书在版编目（CIP）数据

Java 程序设计教程（第五版）= Java Software Solutions: Foundations of Program Design, Fifth Edition /（美）刘易斯（Lewis, J.）等著. －北京：电子工业出版社，2007.8

（国外计算机科学教材系列）

ISBN 978-7-121-02774-1

Ⅰ. J… Ⅱ. 刘… 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（2007）第 064034 号

责任编辑：冯小贝

印　　刷：北京市天竺颖华印刷厂

装　　订：三河市金马印装有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱　邮编：100036

开　　本：787 × 980　1/16　　印张：48.75　　字数：1092 千字

印　　次：2007 年 8 月第 1 次印刷

定　　价：75.00 元

出 版 说 明

21世纪初的5至10年是我国国民经济和社会发展的重要时期，也是信息产业快速发展的关键时期。在我国加入WTO后的今天，培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡，是我国面对国际竞争时成败的关键因素。

当前，正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期，为使我国教育体制与国际化接轨，有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材，以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验，翻译出版了“国外计算机科学教材系列”丛书，这套教材覆盖学科范围广、领域宽、层次多，既有本科专业课程教材，也有研究生课程教材，以适应不同院系、不同专业、不同层次的师生对教材的需求，广大师生可自由选择和自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时，我们也适当引进了一些优秀英文原版教材，本着翻译版本和英文原版并重的原则，对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上，我们大都选择国外著名出版公司出版的高校教材，如Pearson Education培生教育出版集团、麦格劳—希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者，如道格拉斯·科默(Douglas E. Comer)、威廉·斯托林斯(William Stallings)、哈维·戴特尔（Harvey M. Deitel）、尤利斯·布莱克（Uyless Black）等。

为确保教材的选题质量和翻译质量，我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士，也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中，为提高教材质量，我们做了大量细致的工作，包括对所选教材进行全面论证；选择编辑时力求达到专业对口；对排版、印制质量进行严格把关。对于英文教材中出现的错误，我们通过与作者联络和网上下载勘误表等方式，逐一进行了修订。

此外，我们还将与国外著名出版公司合作，提供一些教材的教学支持资料，希望能为授课老师提供帮助。今后，我们将继续加强与各高校教师的密切联系，为广大师生引进更多的国外优秀教材和参考书，为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社

Preface

Welcome to the Fifth Edition of *Java Software Solutions, Foundations of Program Design*. We are pleased that this book has served the needs of so many students and faculty over the years. This edition is designed to further enhance the pedagogy of introductory computing, particularly with enhanced support for the instructor.

The overall vision of the book has not changed significantly from that of previous editions. Feedback from both instructors and students has made it clear that we are hitting the mark in that regard. The emphasis remains on presenting underlying core concepts. The Graphics Track sections in each chapter still segregate the coverage of graphics and graphical user interfaces, giving extreme flexibility in how that material gets covered. The casual writing style and entertaining examples still rule the day.

One of the significant enhancements in this edition is an improved set of end-of-chapter materials. Additional problem sets have been added to the Self-Review Questions, Exercises, and Programming Projects in each chapter. Furthermore, they have been carefully organized to present a nice flow given the topics they address and their level of challenge.

Some key additions and improvements to the text itself have also been made, including a new introductory section in Chapter 4. These additions were designed to strengthen the existing flow of discussion, rather than modifying it. In addition, we've made a complete pass through the text, making numerous minor adjustments to eliminate ambiguities and bolster understanding.

One other key change was made for this edition: we chose to remove the API reference material in Appendix M from the printed text. It is still available online as a supplement for those who'd like to use it. However, the main reason for removing it was to guide students instead to the official API documentation available from the java.sun.com Web site. That resource is much more complete than the abbreviated version we were able to include in the text. Furthermore, it represents the proper, state-of-the-practice technique for looking up API details that professional programmers use every day. We should encourage our students to become familiar with and actively use that official resource.

Cornerstones of the Text

This text is based on the following basic ideas that we believe make for a sound introductory text:

> *True object-orientation.* A text that really teaches a solid object-oriented approach must use what we call object-speak. That is, all processing should be discussed in object-oriented terms. That does not mean, however, that the first program a student sees must discuss the writing of multiple classes and methods. A student should learn to use objects before learning to write them. This text uses a natural progression that culminates in the ability to design real object-oriented solutions.

> *Sound programming practices.* Students should not be taught how to program; they should be taught how to write good software. There's a difference. Writing software is not a set of cookbook actions, and a good program is more than a collection of statements. This text integrates practices that serve as the foundation of good programming skills. These practices are used in all examples and are reinforced in the discussions. Students learn how to solve problems as well as how to implement solutions. We introduce and integrate basic software engineering techniques throughout the text.

> *Examples.* Students learn by example. This text is filled with fully implemented examples that demonstrate specific concepts. We have intertwined small, readily understandable examples with larger, more realistic ones. There is a balance between graphics and nongraphics programs.

> *Graphics and GUIs.* Graphics can be a great motivator for students, and their use can serve as excellent examples of object-orientation. As such, we use them throughout the text in a well-defined set of sections that we call the Graphics Track. This coverage includes the use of event processing and GUIs. Students learn to build GUIs in the appropriate way by using a natural progression of topics. The Graphics Track can be avoided entirely for those who do not choose to use graphics.

Chapter Breakdown

Chapter 1 (Introduction) introduces computer systems in general, including basic architecture and hardware, networking, programming, and language translation. Java is introduced in this chapter, and the basics of general program development, as well as object-oriented programming, are discussed. This chapter contains broad introductory material that can be covered while students become familiar with their development environment.

Chapter 2 (Data and Expressions) explores some of the basic types of data used in a Java program and the use of expressions to perform calculations. It discusses the conversion of data from one type to another, and how to read input interactively from the user with the help of the standard `Scanner` class.

Chapter 3 (Using Classes and Objects) explores the use of predefined classes and the objects that can be created from them. Classes and objects are used to manipulate character strings, produce random numbers, perform complex calculations, and format output. Enumerated types are also discussed.

Chapter 4 (Writing Classes) explores the basic issues related to writing classes and methods. Topics include instance data, visibility, scope, method parameters, and return types. Encapsulation and constructors are covered as well. Some of the more involved topics are deferred to or revisited in Chapter 6.

Chapter 5 (Conditionals and Loops) covers the use of boolean expressions to make decisions. All related statements for conditionals and loops are discussed, including the enhanced version of the `for` loop. The `Scanner` class is revisited for iterative input parsing and reading text files.

Chapter 6 (Object-Oriented Design) reinforces and extends the coverage of issues related to the design of classes. Techniques for identifying the classes and objects needed for a problem and the relationships among them are discussed. This chapter also covers static class members, interfaces, and the design of enumerated type classes. Method design issues and method overloading are also discussed.

Chapter 7 (Arrays) contains extensive coverage of arrays and array processing. Topics include command-line arguments, variable length parameter lists, and multidimensional arrays. The `ArrayList` class and its use as a generic type is explored as well.

Chapter 8 (Inheritance) covers class derivations and associated concepts such as class hierarchies, overriding, and visibility. Strong emphasis is put on the proper use of inheritance and its role in software design.

Chapter 9 (Polymorphism) explores the concept of binding and how it relates to polymorphism. Then we examine how polymorphic references can be accomplished using either inheritance or interfaces. Sorting is used as an example of polymorphism. Design issues related to polymorphism are examined as well.

Chapter 10 (Exceptions) explores the class hierarchy from the Java standard library used to define exceptions, as well as the ability to define our own exception objects. We also discuss the use of exceptions when dealing with input and output, and examine an example that writes a text file.

Chapter 11 (Recursion) covers the concept, implementation, and proper use of recursion. Several examples from various domains are used to demonstrate how recursive techniques make certain types of processing elegant.

Chapter 12 (Data Structures) introduces the idea of a collection and its underlying data structure. Abstraction is revisited in this context and the classic data structures are explored. Generic types are introduced as well. This chapter serves as an introduction to a CS2 course.

Supplements

Student CD

This CD includes:

> Source code for all the programs in the text.
> Various Java development environments.

If a CD did not come with your book or you can't locate your CD, you can access most of these items at www.aw.com/cssupport

Other CDs Upon Request

Professors using this book in a course may want to order it with one of many other available Java development environments. Contact your campus Addison-Wesley representative for a list of current IDEs and their specific ISBNs to order.

MyCodeMate—Your Own T.A. Just a Click Away

Addison-Wesley's *MyCodeMate* is a book-specific Web resource that provides tutorial help and evaluation of student programs. Example programs throughout the book and selected Programming Projects from every chapter have been integrated into *MyCodeMate*. Using this tool, a student is able to write and compile programs from any computer with Internet access, and receive guidance and feedback on how to proceed and on how to address compiler error messages. Instructors can track each student's progress on Programming Projects from the text or can develop projects of their own. **A complementary subscription of *MyCodeMate* is offered when the access code is ordered in a package with a new copy of this text.** Subscriptions can also be purchased online. For more information visit www.mycodemate.com, or contact your campus Addison-Wesley representative.

Instructor Resources

The following supplements are available to qualified instructors only. Visit the Addison-Wesley Instructor Resource Center (www.aw.com/irc) or send an e-mail to computing@aw.com for information on how to access them:

> Presentation Slides—in PowerPoint.
> Solutions—includes solutions to exercises and programming projects.
> Test Bank with powerful test generator software—includes a wealth of free response, multiple-choice, and true/false type questions.
> Lab Manual—lab exercises are designed to accompany the topic progression in the text. A printed version of this manual is also available.

Acknowledgments

We are most grateful to the faculty and students from around the world who have provided their feedback on previous editions of this book. We are pleased to see the depth of the faculty's concern for their students and the students' thirst for knowledge. Your comments and questions are always welcome.

We continue to be amazed at the talent and effort demonstrated by the team at Addison-Wesley. Michael Hirsch, our editor, has amazing insight and commitment. His assistant, Lindsey Triebel, is a source of consistent and helpful support. Marketing Manager Michelle Brown makes sure that instructors understand the pedagogical advantages of the text. The cover and interior design were designed by the skilled talents of Joyce Wells. Marilyn Lloyd led the production effort. The Addison-Wesley folks are supported by a phenomenal team at Argosy Publishing, including Megan Schwenke and Edalin Michael. We thank all of these people for ensuring that this book meets the highest quality standards.

Special thanks go to the following people who provided valuable advice to us about this book via their participation in focus groups, interviews, and reviews: Robert Burton—Brigham Young University; John Chandler—Oklahoma State University; Dave Musicant—Carleton College; Patricia Roth—Southern Polytechnic State University; Saroja Kanchi—Kettering University; Elizabeth Adams—James Madison University; Stuart Steiner—Eastern Washington University; Laurie Murphy—Pacific Lutheran University; Dodi Coreson—Linn Benton Community College.

The reviewers of previous editions of this text, as well as many other instructors and friends, have provided valuable feedback. They include:

Lewis Barnett	University of Richmond
Tom Bennet	Mississippi College
Gian Mario Besana	DePaul University

Hans-Peter Bischof	Rochester Institute of Technology
Robert Burton	Brigham Young University
James Cross	Auburn University
Robert Cohen	University of Massachusetts, Boston
Eman El-Sheikh	University of West Florida
Christopher Eliot	University of Massachusetts, Amherst
Matt Evett	Eastern Michigan University
John Gauch	University of Kansas
Chris Haynes	Indiana University
Laurie Hendren	McGill University
James Heliotis	Rochester Institute of Technology
Mike Higgs	Austin College
Karen Kluge	Dartmouth College
Jason Levy	University of Hawaii
Peter MacKenzie	McGill University
Blayne Mayfield	Oklahoma State University
Faye Navabi-Tadayon	Arizona State University
Lawrence Osborne	Lamar University
Barry Pollack	City College of San Francisco
B. Ravikumar	University of Rhode Island
David Riley	University of Wisconsin (La Crosse)
Jerry Ross	Lane Community College
Carolyn Schauble	Colorado State University
Arjit Sengupta	Georgia State University
Vijay Srinivasan	JavaSoft, Sun Microsystems, Inc.
Katherine St. John	Lehman College, CUNY
Ed Timmerman	University of Maryland, University College
Shengru Tu	University of New Orleans
Paul Tymann	Rochester Institute of Technology
John J. Wegis	JavaSoft, Sun Microsystems, Inc.
Linda Wilson	Dartmouth College
David Wittenberg	Brandeis University
Wang-Chan Wong	California State University (Dominguez Hills)

Thanks also go to my colleagues at Villanova University who have provided so much wonderful feedback. They include Bob Beck, Cathy Helwig, Dan Joyce, Anany Levitin, Najib Nadi, Beth Taddei, and Barbara Zimmerman.

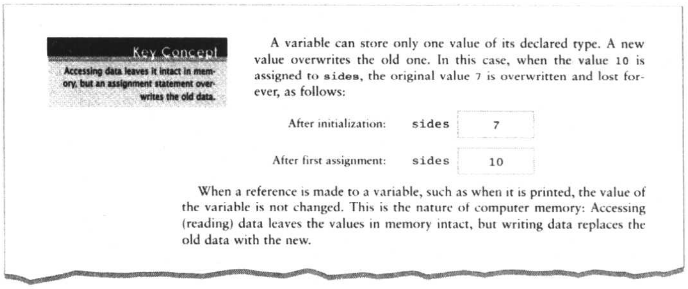
Special thanks go to Pete DePasquale of The College of New Jersey for the design and evolution of the PaintBox project, as well as the original Java Class Library appendix.

Many other people have helped in various ways. They include Ken Arnold, Mike Czepiel, John Loftus, Sebastian Niezgoda, and Sammy Perugini. Our apologies to anyone we may have forgotten.

The ACM Special Interest Group on Computer Science Education (SIGCSE) is a tremendous resource. Their conferences provide an opportunity for educators from all levels and all types of schools to share ideas and materials. If you are an educator in any area of computing and are not involved with SIGCSE, you're missing out.

Feature Walkthrough

Key Concepts. Throughout the text, the Key Concept boxes highlight fundamental ideas and important guidelines. These concepts are summarized at the end of each chapter.



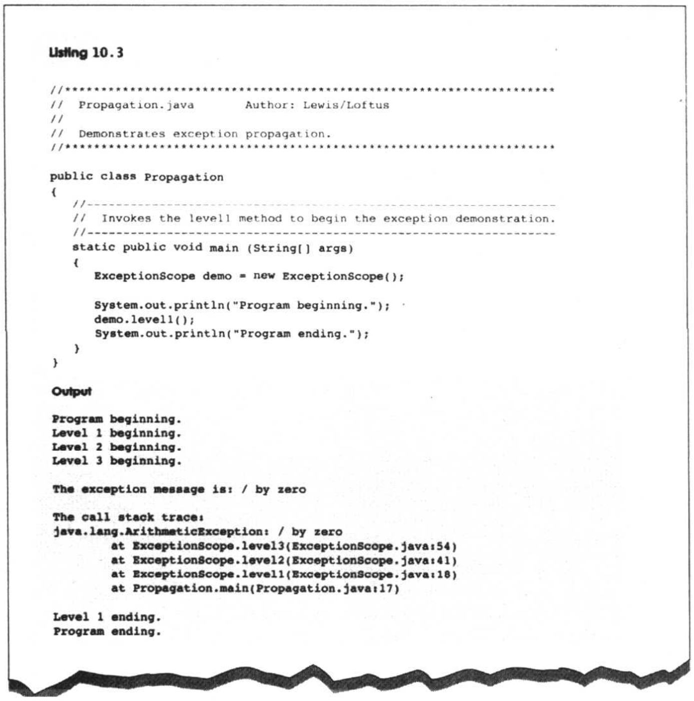
Key Concept

Accessing data leaves it intact in memory, but an assignment statement overwrites the old data.

A variable can store only one value of its declared type. A new value overwrites the old one. In this case, when the value `10` is assigned to `sides`, the original value `7` is overwritten and lost forever, as follows:

After initialization:	`sides`	7
After first assignment:	`sides`	10

When a reference is made to a variable, such as when it is printed, the value of the variable is not changed. This is the nature of computer memory: Accessing (reading) data leaves the values in memory intact, but writing data replaces the old data with the new.



Listing 10.3

```
//********************************************************************
//  Propagation.java       Author: Lewis/Loftus
//
//  Demonstrates exception propagation.
//********************************************************************

public class Propagation
{
   //-----------------------------------------------------------------
   //  Invokes the level1 method to begin the exception demonstration.
   //-----------------------------------------------------------------
   static public void main (String[] args)
   {
      ExceptionScope demo = new ExceptionScope();

      System.out.println("Program beginning.");
      demo.level1();
      System.out.println("Program ending.");
   }
}
```

Output

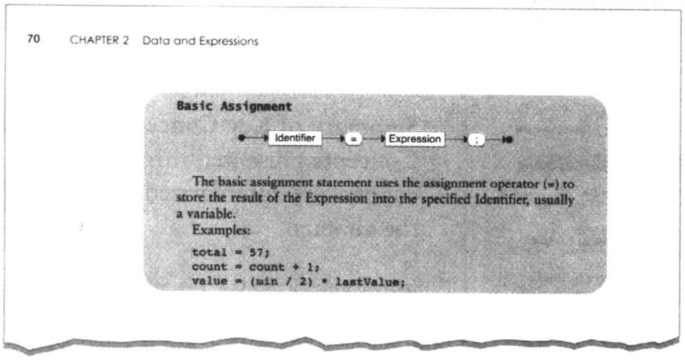
```
Program beginning.
Level 1 beginning.
Level 2 beginning.
Level 3 beginning.

The exception message is: / by zero

The call stack trace:
java.lang.ArithmeticException: / by zero
        at ExceptionScope.level3(ExceptionScope.java:54)
        at ExceptionScope.level2(ExceptionScope.java:41)
        at ExceptionScope.level1(ExceptionScope.java:18)
        at Propagation.main(Propagation.java:17)

Level 1 ending.
Program ending.
```

Listings. All programming examples are presented in clearly labeled listings, followed by the program output, a sample run, or screen shot display as appropriate. The code is colored to visually distinguish comments and reserved words.


70 CHAPTER 2 Data and Expressions

Basic Assignment

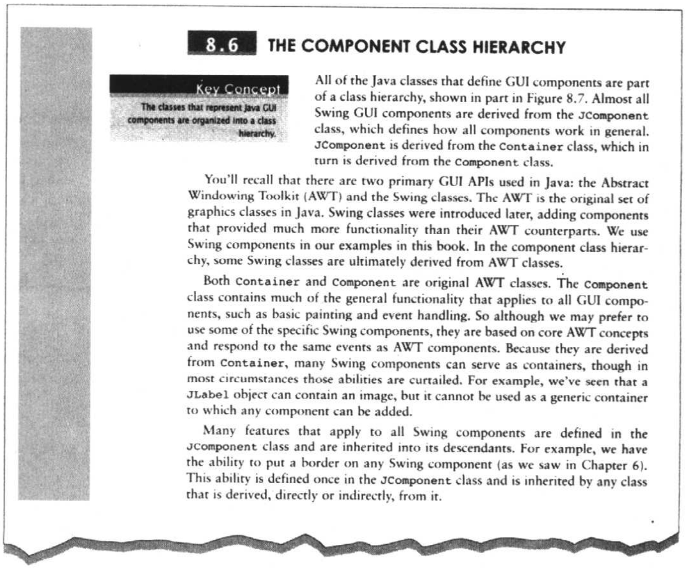
The basic assignment statement uses the assignment operator (=) to store the result of the Expression into the specified Identifier, usually a variable.

Examples:

```
total = 57;
count = count + 1;
value = (min / 2) * lastValue;
```

Syntax Diagrams. At appropriate points in the text, syntactic elements of the Java language are discussed in special highlighted sections with diagrams that clearly identify the valid forms for a statement or construct. Syntax diagrams for the entire Java language are presented in Appendix L.

Graphics Track. All processing that involves graphics and graphical user interfaces is discussed in one or two sections at the end of each chapter that we collectively refer to as the Graphics Track. This material can be skipped without loss of continuity, or focused on specifically as desired. The material in any Graphics Track section relates to the main topics of the chapter in which it is found. Graphics Track sections are indicated by a patterned border on the edge of the page.


8.6 THE COMPONENT CLASS HIERARCHY

Key Concept

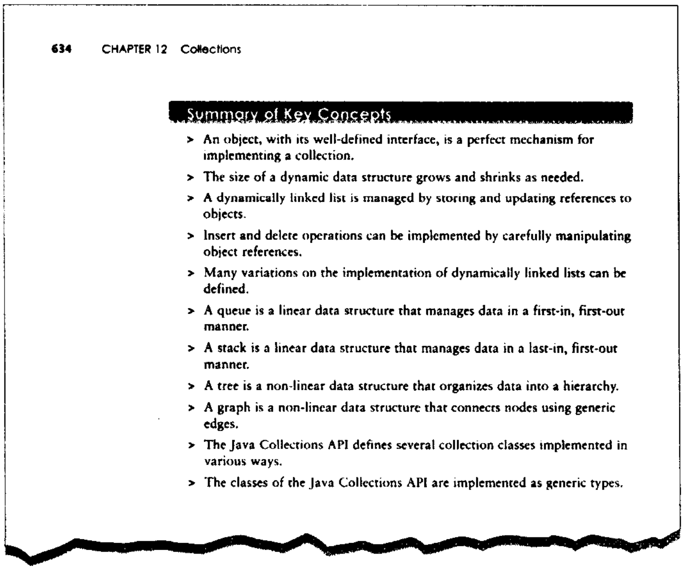
The classes that represent Java GUI components are organized into a class hierarchy.

All of the Java classes that define GUI components are part of a class hierarchy, shown in part in Figure 8.7. Almost all Swing GUI components are derived from the `JComponent` class, which defines how all components work in general. `JComponent` is derived from the `Container` class, which in turn is derived from the `Component` class.

You'll recall that there are two primary GUI APIs used in Java: the Abstract Windowing Toolkit (AWT) and the Swing classes. The AWT is the original set of graphics classes in Java. Swing classes were introduced later, adding components that provided much more functionality than their AWT counterparts. We use Swing components in our examples in this book. In the component class hierarchy, some Swing classes are ultimately derived from AWT classes.

Both `Container` and `Component` are original AWT classes. The `Component` class contains much of the general functionality that applies to all GUI components, such as basic painting and event handling. So although we may prefer to use some of the specific Swing components, they are based on core AWT concepts and respond to the same events as AWT components. Because they are derived from `Container`, many Swing components can serve as containers, though in most circumstances those abilities are curtailed. For example, we've seen that a `JLabel` object can contain an image, but it cannot be used as a generic container to which any component can be added.

Many features that apply to all Swing components are defined in the `JComponent` class and are inherited into its descendants. For example, we have the ability to put a border on any Swing component (as we saw in Chapter 6). This ability is defined once in the `JComponent` class and is inherited by any class that is derived, directly or indirectly, from it.

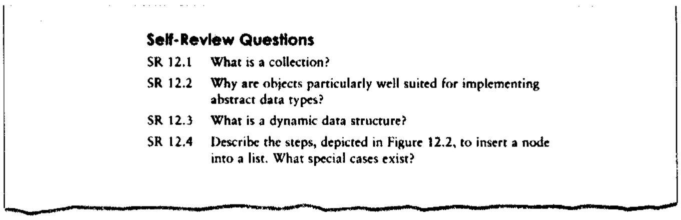

634 CHAPTER 12 Collections

Summary of Key Concepts

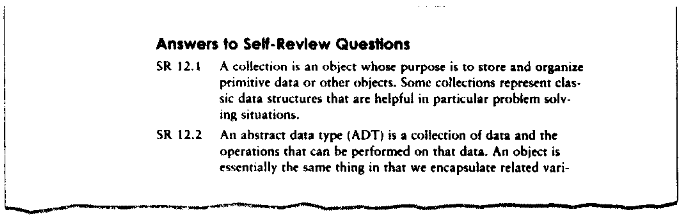
- An object, with its well-defined interface, is a perfect mechanism for implementing a collection.
- The size of a dynamic data structure grows and shrinks as needed.
- A dynamically linked list is managed by storing and updating references to objects.
- Insert and delete operations can be implemented by carefully manipulating object references.
- Many variations on the implementation of dynamically linked lists can be defined.
- A queue is a linear data structure that manages data in a first-in, first-out manner.
- A stack is a linear data structure that manages data in a last-in, first-out manner.
- A tree is a non-linear data structure that organizes data into a hierarchy.
- A graph is a non-linear data structure that connects nodes using generic edges.
- The Java Collections API defines several collection classes implemented in various ways.
- The classes of the Java Collections API are implemented as generic types.

Summary of Key Concepts. The Key Concepts presented throughout a chapter are summarized at the end of the chapter.

Self-Review Questions and Answers. These short-answer questions review the fundamental ideas and terms established in the chapter. They are designed to allow students to assess their own basic grasp of the material. The answers to these questions can be found at the end of the problem sets.

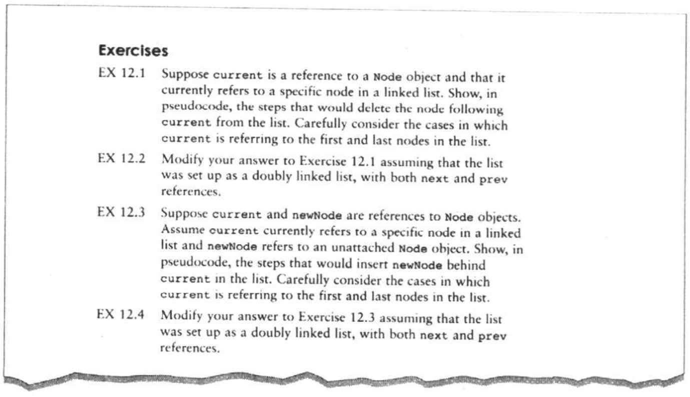

Self-Review Questions

SR 12.1	What is a collection?
SR 12.2	Why are objects particularly well suited for implementing abstract data types?
SR 12.3	What is a dynamic data structure?
SR 12.4	Describe the steps, depicted in Figure 12.2, to insert a node into a list. What special cases exist?


Answers to Self-Review Questions

SR 12.1	A collection is an object whose purpose is to store and organize primitive data or other objects. Some collections represent classic data structures that are helpful in particular problem solving situations.
SR 12.2	An abstract data type (ADT) is a collection of data and the operations that can be performed on that data. An object is essentially the same thing in that we encapsulate related vari-

Exercises. These intermediate problems require computations, the analysis or writing of code fragments, and probing questions about the chapter content. While the exercises may deal with code, they generally do not require any online activity.



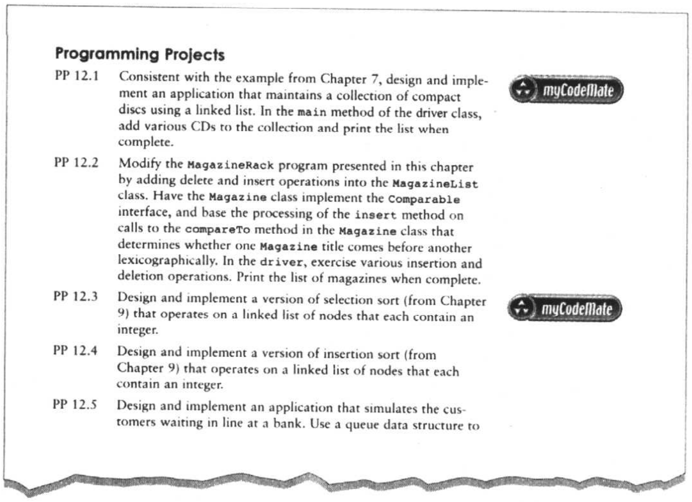
Exercises

EX 12.1 Suppose `current` is a reference to a `Node` object and that it currently refers to a specific node in a linked list. Show, in pseudocode, the steps that would delete the node following `current` from the list. Carefully consider the cases in which `current` is referring to the first and last nodes in the list.

EX 12.2 Modify your answer to Exercise 12.1 assuming that the list was set up as a doubly linked list, with both `next` and `prev` references.

EX 12.3 Suppose `current` and `newNode` are references to `Node` objects. Assume `current` currently refers to a specific node in a linked list and `newNode` refers to an unattached `Node` object. Show, in pseudocode, the steps that would insert `newNode` behind `current` in the list. Carefully consider the cases in which `current` is referring to the first and last nodes in the list.

EX 12.4 Modify your answer to Exercise 12.3 assuming that the list was set up as a doubly linked list, with both `next` and `prev` references.



Programming Projects

PP 12.1 Consistent with the example from Chapter 7, design and implement an application that maintains a collection of compact discs using a linked list. In the `main` method of the driver class, add various CDs to the collection and print the list when complete.

myCodeMate

PP 12.2 Modify the `MagazineRack` program presented in this chapter by adding delete and insert operations into the `MagazineList` class. Have the `Magazine` class implement the `Comparable` interface, and base the processing of the `insert` method on calls to the `compareTo` method in the `Magazine` class that determines whether one `Magazine` title comes before another lexicographically. In the `driver`, exercise various insertion and deletion operations. Print the list of magazines when complete.

PP 12.3 Design and implement a version of selection sort (from Chapter 9) that operates on a linked list of nodes that each contain an integer.

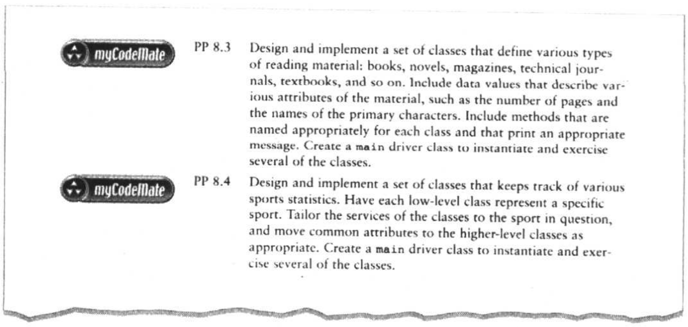
myCodeMate

PP 12.4 Design and implement a version of insertion sort (from Chapter 9) that operates on a linked list of nodes that each contain an integer.

PP 12.5 Design and implement an application that simulates the customers waiting in line at a bank. Use a queue data structure to

Programming Projects. These problems require the design and implementation of Java programs. They vary widely in level of difficulty.

Addison-Wesley's MyCodeMate. Working online, students can view, compile, run, and edit select programming problems and all code listings from the textbook. Look for this MyCodeMate icon to see which Programming Projects are available with your included online subscription to MyCodeMate.



myCodeMate

PP 8.3 Design and implement a set of classes that define various types of reading material: books, novels, magazines, technical journals, textbooks, and so on. Include data values that describe various attributes of the material, such as the number of pages and the names of the primary characters. Include methods that are named appropriately for each class and that print an appropriate message. Create a `main` driver class to instantiate and exercise several of the classes.

myCodeMate

PP 8.4 Design and implement a set of classes that keeps track of various sports statistics. Have each low-level class represent a specific sport. Tailor the services of the classes to the sport in question, and move common attributes to the higher-level classes as appropriate. Create a `main` driver class to instantiate and exercise several of the classes.

Contents