Mc Graw Hill Education 大学计算机教育国外著名教材系列（影印版）

JAVA™ STRUCTURES

Data Structures in Java™ for the Principled Programmer Second Edition

数据结构 Java语言描述（第2版）

Duane A. Bailey 著

清华大学出版社

JAVA™ STRUCTURES

Data Structures in Java™ for the Principled Programmer™

数据结构 Java语言描述

大学计算机教育国外著名教材系列（影印版）

Java™ Structures

Data Structures in Java™ for the Principled Programmer

Second Edition

数据结构 Java 语言描述

（第 2 版）

Duane A. Bailey 著

Duane A. Bailey
Java™ Structures: Data Structures in Java™ for the Principled Programmer, Second Edition
EISBN: 0-07-239909-0

Copyright © 2003 by The McGraw-Hill Companies, Inc.

Original language published by The McGraw-Hill Companies, Inc. All Rights reserved. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Authorized English language edition jointly published by McGraw-Hill Education (Asia) Co. and Tsinghua University Press. This edition is authorized for sale only to the educational and training institutions, and within the territory of the People's Republic of China (excluding Hong Kong, Macao SAR and Taiwan). Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书英文影印版由清华大学出版社和美国麦格劳-希尔教育出版(亚洲)公司合作出版。此版本仅限在中华人民共和国境内(不包括中国香港、澳门特别行政区及中国台湾地区)针对教育及培训机构之销售。未经许可之出口，视为违反著作权法，将受法律之制裁。

未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

北京市版权局著作权合同登记号　图字：01-2003-3527

本书封面贴有 McGraw-Hill 公司防伪标签，无标签者不得销售。

图书在版编目(CIP)数据

数据结构 Java 语言描述 = Java™ Structures: Data Structures in Java™ for the Principled Programmer（第 2 版）/ [美] 贝利（Bailey, D. A.）著. —影印本. —北京：清华大学出版社，2003.10
（大学计算机教育国外著名教材系列）
ISBN 7-302-07415-1

Ⅰ. 数… Ⅱ. 贝… Ⅲ. ① 数据结构－高等学校－教材－英文 ② Java 语言－程序设计－高等学校－教材－英文 Ⅳ. ①TP311.12 ②TP312

中国版本图书馆 CIP 数据核字（2003）第 092836 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）6277 0175　　客户服务：（010）6277 6969
责任编辑：周维焜
印 刷 者：北京牛山世兴印刷厂
装 订 者：三河市金元装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×230　印张：33.25
版　　次：2003 年 11 月第 1 版　2003 年 11 月第 1 次印刷
书　　号：ISBN 7-302-07415-1/TP・5477
印　　数：1~5000
定　　价：46.00 元

出 版 说 明

进入21世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的争夺。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从1996年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书（影印版）”等一系列引进图书，受到了国内读者的欢迎和支持。跨入21世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列（影印版）”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列（影印版）”做得更好，更适合高校师生的需要。

清华大学出版社
2002.10

此为试读，需要完整PDF请访问：www.ertongbook.com

Preface to the First Edition

"IT'S A WONDERFUL TIME TO BE ALIVE." At least that's what I've found myself saying over the past couple of decades. When I first started working with computers, they were resources used by a privileged (or in my case, persistent) few. They were physically large, and logically small. They were cast from iron. The challenge was to make these behemoths solve complex problems quickly.

Today, computers are everywhere. They are in the office and at home. They speak to us on telephones; they zap our food in the microwave. They make starting cars in New England a possibility. Everyone's using them. What has aided their introduction into society is their diminished size and cost, and increased capability. The challenge is to make these behemoths solve complex problems quickly.

Thus, while the computer and its applications have changed over time, the challenge remains the same: *How can we get the best performance out of the current technology?* The design and analysis of data structures lay the fundamental groundwork for a scientific understanding of what computers can do efficiently. The motivations for data structure design work accomplished three decades ago in assembly language at the keypunch are just as familiar to us today as we practice our craft in modern languages on computers on our laps. The focus of this material is the identification and development of relatively *abstract* principles for structuring data in ways that make programs efficient in terms of their consumption of resources, *as well as efficient in terms of "programmability."*

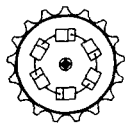
In the past, my students have encountered this material in Pascal, Modula-2, and, most recently, C++. None of these languages has been ideal, but each has been met with increasing expectation. This text uses The Java Programming Language[1]—"Java"—to structure data. Java is a new and exciting language that has received considerable public attention. At the time of this writing, for example, Java is one of the few tools that can effectively use the Internet as a computing resource. That particular aspect of Java is not touched on greatly in this text. Still, Internet-driven applications in Java will need supporting data structures. This book attempts to provide a fresh and focused approach to the design and implementation of classic structures in a manner that meshes well with existing Java packages. It is hoped that learning this material in Java will improve the way working programmers craft programs, and the way future designers craft languages.

Pedagogical Implications. This text was developed specifically for use with CS2 in a standard Computer Science curriculum. It is succinct in its approach, and requires, perhaps, a little more effort to read. I hope, though, that this text

[1] Java is a trademark of Sun Microsystems, Incorporated.

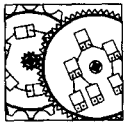
becomes not a brief encounter with object-oriented data structure design, but a touchstone for one's programming future.

The material presented in this text follows the syllabus I have used for several years at Williams. As students come to this course with experience using Java, the outline of the text may be followed directly. Where students are new to Java, a couple of weeks early in the semester will be necessary with a good companion text to introduce the student to new concepts, and an introductory Java language text or reference manual is recommended. For students that need a quick introduction to Java we provide a tutorial in Appendix B. While the text was designed as a whole, some may wish to eliminate less important topics and expand upon others. Students may wish to drop (or consider!) the section on induction (Section 4.2.2). The more nontraditional topics—including, for example, iteration and the notions of symmetry and friction—have been included because I believe they arm programmers with important mechanisms for implementing and analyzing problems. In many departments the subtleties of more advanced structures—maps (Chapter 14) and graphs (Chapter 15)—may be considered in an algorithms course. Chapter 5, a discussion of sorting, provides very important motivating examples and also begins an early investigation of algorithms. The chapter may be dropped when better examples are at hand, but students may find the refinements on implementing sorting interesting.



List

Associated with this text is a Java package of data structures that is freely available over the Internet for noncommercial purposes. I encourage students, educators, and budding software engineers to download it, tear it down, build it up, and generally enjoy it. In particular, students of this material are encouraged to follow along with the code online as they read. Also included is extensive documentation gleaned from the code by `javadoc`. All documentation—within the book and on the Web—includes pre- and postconditions. The motivation for this style of commenting is provided in Chapter 2. While it's hard to be militant about commenting, this style of documentation provides an obvious, structured approach to minimally documenting one's methods that students can appreciate and users will welcome. These resources, as well as many others, are available from McGraw-Hill at `http://www.mhhe.com/javastructures`.



nim

Three icons appear throughout the text, as they do in the margin. The top "compass" icon highlights the statement of a *principle*—a statement that encourages abstract discussion. The middle icon marks the first appearance of a particular class from the `structure` package. Students will find these files at McGraw-Hill, or locally, if they've been downloaded. The bottom icon similarly marks the appearance of example code.

Finally, I'd like to note an unfortunate movement away from studying the implementation of data structures, in favor of studying applications. In the extreme this is a disappointing and, perhaps, dangerous precedent. The design of a data structure is like the solution to a riddle: the process of developing the answer is as important as the answer itself. The text may, however, be used as a reference for using the `structure` package in other applications by selectively avoiding the discussions of implementation.

Preface to the Second Edition

Since the first edition of *Java Structures* support for writing programs in Java[2] has grown considerably. At that time the Java Development Toolkit consisted of 504 classes in 23 packages[3] In Java 1.2 (also called Java 2) Sun rolled out 1520 classes in 59 packages. This book is ready for Java 1.4, where the number of classes and packages continues to grow.

Most computer scientists are convinced of the utility of Java for programming in a well structured and platform independent manner. While there are still significant arguments about important aspects of the language (for example, support for generic types), the academic community is embracing Java, for example, as the subject of the Computer Science Advanced Placement Examination.

It might seem somewhat perplexing to think that many aspects of the original Java environment have been retracted (or *deprecated*) or reconsidered. The developers at Sun have one purpose in mind: to make Java the indispensable language of the current generation. As a result, documenting their progress on the development of data structures gives us valuable insight into the process of designing useful data structures for general purpose programming. Those students and faculty considering a move to this second edition of *Java Structures* will see first-hand some of the decisions that have been made in the intervening years. During that time, for example, the `Collection`-based classes were introduced, and are generally considered an improvement. Another force—one similar to calcification—has left a trail of backwards compatible features that are sometimes difficult to understand. For example, the `Iterator` class was introduced, but the `Enumeration` class was not deprecated. One subject of the first edition—the notion of `Comparable` classes—has been introduced into a number of important classes including `String` and `Integer`. This is a step forward and a reconsideration of what we have learned about that material has lead to important improvements in the text.

Since the main purpose of the text is to demonstrate the design and behavior of traditional data structures, we have not generally tracked the progress of Java where it blurs the view. For example, Java 2 introduces a `List` interface (we applaud) but the `Vector` class has been extended to include methods that are, essentially, motivated by linked lists (we wonder). As this text points out frequently, the purpose of an interface is often to provide *reduced* functionality. If the data structure does not *naturally* provide the functionality required by the application, it is probably not an effective tool for solving the problem: search elsewhere for an effective structure.

---

[2] The Java Programming Language is a trademark of Sun Microsystems, Incorporated.

[3] David Flanagan, et al., *Java in a Nutshell*, O'Reilly & Associates.

As of this writing, more than 100,000 individuals have searched for and downloaded the `structure` package. To facilitate using the comprehensive set of classes with the Java 2 environment, we have provided a number of features that support the use of the `structure` package in more concrete applications. Please see Appendix C.

Also new to this edition are more than 200 new problems, several dozen exercises, and over a dozen labs we regularly use at Williams.

Acknowledgments. Several students, instructors, and classes have helped to shape this edition of *Java Structures*. Parth Doshi and Alex Glenday—diligent Williams students—pointed out a large number of typos and stretches of logic. Kim Bruce, Andrea Danyluk, Jay Sachs, and Jim Teresco have taught this course at Williams over the past few years, and have provided useful feedback. I tip my hat to Bill Lenhart, a good friend and advisor, who has helped improve this text in subtle ways. To Sean Sandys I am indebted for showing me new ways to teach new minds.

The various reviewers have made, collectively, hundreds of pages of comments that have been incorporated (as much as possible) into this edition: Eleanor Hare and David Jacobs (Clemson University), Ram Athavale (North Carolina State University), Yannick Daoudi (McGill University), Walter Daugherty (Texas A&M University), Subodh Kumar (Johns Hopkins University), Toshimi Minoura (Oregon State University), Carolyn Schauble (Colorado State University), Val Tannen (University of Pennsylvania), Frank Tompa (University of Waterloo), Richard Wiener (University of Colorado at Colorado Springs), Cynthia Brown Zickos (University of Mississippi), and my good friend Robbie Moll (University of Massachusetts). Deborah Trytten (University of Oklahoma) has reviewed both editions! Still, until expert authoring systems are engineered, authors will remain human. Any mistakes left behind or introduced are purely those of the author.

The editors and staff at McGraw-Hill–Kelly Lowery, Melinda Dougharty, John Wannemacher, and Joyce Berendes–have attempted the impossible: to keep me within a deadline. David Hash, Phil Meek, and Jodi Banowetz are responsible for the look and feel of things. I am especially indebted to Lucy Mullins, Judy Gantenbein, and Patti Evers whose red pens have often shown me a better way.

Betsy Jones, publisher and advocate, has seen it all and yet kept the faith: thanks.

Be aware, though: long after these pages are found to be useless folly, my best work will be recognized in my children, Kate, Megan, and Ryan. None of these projects, of course, would be possible without the support of my best friend, my north star, and my partner, Mary.

Enjoy!

Duane A. Bailey
Williamstown, May 2002

Contents

Preface to First Edition ix

Preface to the Second Edition xi

0 Introduction 1
0.1 Read Me . . . 1
0.2 He Can't Say That, Can He? . . . 2

1 The Object-Oriented Method 5
1.1 Data Abstraction and Encapsulation . . . 6
1.2 The Object Model . . . 7
1.3 Object-Oriented Terminology . . . 8
1.4 A Special-Purpose Class: A Bank Account . . . 11
1.5 A General-Purpose Class: An Association . . . 14
1.6 Sketching an Example: A Word List . . . 18
1.7 Sketching an Example: A Rectangle Class . . . 20
1.8 Interfaces . . . 22
1.9 Who Is the User? . . . 24
1.10 Conclusions . . . 25
1.11 Laboratory: The Day of the Week Calculator . . . 29

2 Comments, Conditions, and Assertions 33
2.1 Pre- and Postconditions . . . 34
2.2 Assertions . . . 34
2.3 Craftsmanship . . . 36
2.4 Conclusions . . . 37
2.5 Laboratory: Using Javadoc Commenting . . . 39

3 Vectors 43
3.1 The Interface . . . 45
3.2 Example: The Word List Revisited . . . 47
3.3 Example: Word Frequency . . . 48
3.4 The Implementation . . . 50
3.5 Extensibility: A Feature . . . 53
3.6 Example: L-Systems . . . 55
3.7 Example: Vector-Based Sets . . . 57
3.8 Example: The Matrix Class . . . 60
3.9 Conclusions . . . 63
3.10 Laboratory: The Silver Dollar Game . . . 67

4 Design Fundamentals **69**
4.1 Asymptotic Analysis Tools 69
4.1.1 Time and Space Complexity 70
4.1.2 Examples 73
4.1.3 The Trading of Time and Space 79
4.1.4 Back-of-the-Envelope Estimations 80
4.2 Self-Reference 82
4.2.1 Recursion 82
4.2.2 Mathematical Induction 89
4.3 Properties of Design 96
4.3.1 Symmetry 96
4.3.2 Friction 98
4.4 Conclusions 98
4.5 Laboratory: How Fast Is Java? 103

5 Sorting **107**
5.1 Approaching the Problem 107
5.2 Selection Sort 110
5.3 Insertion Sort 113
5.4 Mergesort 115
5.5 Quicksort 119
5.6 Radix Sort 122
5.7 Sorting Objects 126
5.8 Ordering Objects Using Comparators 128
5.9 Vector-Based Sorting 131
5.10 Conclusions 132
5.11 Laboratory: Sorting with Comparators 135

6 A Design Method **137**
6.1 The Interface-Based Approach 137
6.1.1 Design of the Interface 138
6.1.2 Development of an Abstract Implementation 139
6.1.3 Implementation 140
6.2 Example: Development of Generators 140
6.3 Example: Playing Cards 143
6.4 Conclusions 148

7 Iterators **149**
7.1 Java's Enumeration Interface 149
7.2 The Iterator Interface 151
7.3 Example: Vector Iterators 153
7.4 Example: Rethinking Generators 155
7.5 Example: Filtering Iterators 157
7.6 Conclusions 160
7.7 Laboratory: The Two-Towers Problem 163

此为试读,需要完整PDF请访问: www.ertongbook.com

8 Lists **167**
8.1 Example: A Unique Program 170
8.2 Example: Free Lists 171
8.3 Partial Implementation: Abstract Lists 174
8.4 Implementation: Singly Linked Lists 176
8.5 Implementation: Doubly Linked Lists 189
8.6 Implementation: Circularly Linked Lists 194
8.7 Implementation: Vectors 197
8.8 List Iterators 197
8.9 Conclusions 199
8.10 Laboratory: Lists with Dummy Nodes 203

9 Linear Structures **207**
9.1 Stacks 209
9.1.1 Example: Simulating Recursion 210
9.1.2 Vector-Based Stacks 213
9.1.3 List-Based Stacks 215
9.1.4 Comparisons 216
9.2 Queues 217
9.2.1 Example: Solving a Coin Puzzle 219
9.2.2 List-Based Queues 222
9.2.3 Vector-Based Queues 223
9.2.4 Array-Based Queues 226
9.3 Example: Solving Mazes 230
9.4 Conclusions 232
9.5 Laboratory: A Stack-Based Language 235
9.6 Laboratory: The Web Crawler 239

10 Ordered Structures **241**
10.1 Comparable Objects Revisited 241
10.1.1 Example: Comparable Ratios 242
10.1.2 Example: Comparable Associations 244
10.2 Keeping Structures Ordered 246
10.2.1 The OrderedStructure Interface 247
10.2.2 The Ordered Vector and Binary Search 247
10.2.3 Example: Sorting Revisited 252
10.2.4 A Comparator-based Approach 253
10.2.5 The Ordered List 255
10.2.6 Example: The Modified Parking Lot 258
10.3 Conclusions 260
10.4 Laboratory: Computing the "Best Of" 263

11 Binary Trees **265**
11.1 Terminology 265
11.2 Example: Pedigree Charts 268
11.3 Example: Expression Trees 269
11.4 Implementation 270
11.4.1 The BinaryTree Implementation 270
11.5 Example: An Expert System 275
11.6 Traversals of Binary Trees 278
11.6.1 Preorder Traversal 279
11.6.2 In-order Traversal 282
11.6.3 Postorder Traversal 283
11.6.4 Level-order Traversal 284
11.6.5 Recursion in Iterators 286
11.7 Property-Based Methods 287
11.8 Example: Huffman Compression 291
11.9 Example Implementation: Ahnentafel 295
11.10Conclusions 296
11.11Laboratory: Playing Gardner's Hex-a-Pawn 301

12 Priority Queues **303**
12.1 The Interface 303
12.2 Example: Improving the Huffman Code 305
12.3 A Vector-Based Implementation 306
12.4 A Heap Implementation 307
12.4.1 Vector-Based Heaps 308
12.4.2 Example: Heapsort 314
12.4.3 Skew Heaps 317
12.5 Example: Circuit Simulation 321
12.6 Conclusions 325
12.7 Laboratory: Simulating Business 329

13 Search Trees **331**
13.1 Binary Search Trees 331
13.2 Example: Tree Sort 332
13.3 Example: Associative Structures 333
13.4 Implementation 335
13.5 Splay Trees 341
13.6 Splay Tree Implementation 343
13.7 An Alternative:Red-Black Trees 348
13.8 Conclusions 351
13.9 Laboratory: Improving the BinarySearchTree 353

14 Maps **355**
14.1 Example Revisited: The Symbol Table . . . 355
14.2 The Interface . . . 356
14.3 Simple Implementation: MapList . . . 358
14.4 Constant Time Maps: Hash Tables . . . 360
14.4.1 Open Addressing . . . 361
14.4.2 External Chaining . . . 369
14.4.3 Generation of Hash Codes . . . 371
14.4.4 Hash Codes for Collection Classes . . . 377
14.4.5 Performance Analysis . . . 378
14.5 Ordered Maps and Tables . . . 378
14.6 Example: Document Indexing . . . 381
14.7 Conclusions . . . 384
14.8 Laboratory: The Soundex Name Lookup System . . . 387

15 Graphs **389**
15.1 Terminology . . . 389
15.2 The Graph Interface . . . 390
15.3 Implementations . . . 394
15.3.1 Abstract Classes Reemphasized . . . 394
15.3.2 Adjacency Matrices . . . 396
15.3.3 Adjacency Lists . . . 402
15.4 Examples: Common Graph Algorithms . . . 408
15.4.1 Reachability . . . 408
15.4.2 Topological Sorting . . . 410
15.4.3 Transitive Closure . . . 413
15.4.4 All Pairs Minimum Distance . . . 414
15.4.5 Greedy Algorithms . . . 415
15.5 Conclusions . . . 420
15.6 Laboratory: Converting Between Units . . . 425

A Answers **427**
A.1 Solutions to Self Check Problems . . . 427
A.2 Solutions to Odd-Numbered Problems . . . 437

B Beginning with Java **473**
B.1 A First Program . . . 473
B.2 Declarations . . . 475
B.2.1 Primitive Types . . . 475
B.2.2 Reference Types . . . 477
B.3 Important Classes . . . 478
B.3.1 The ReadStream Class . . . 478
B.3.2 The PrintStream Class . . . 479
B.3.3 Strings . . . 479
B.4 Control Constructs . . . 480
B.4.1 Conditional Statements . . . 480

B.4.2 Loops 481
B.5 Methods 483
B.6 Inheritance and Subtyping 484
B.6.1 Inheritance 484
B.6.2 Subtyping 485
B.6.3 Interfaces and Abstract Classes 486
B.7 Use of the Assert Command 487
B.8 Use of the Keyword `Protected` 488

C Collections **491**
C.1 Collection Class Features 491
C.2 Parallel Features 491
C.3 Conversion 492

D Documentation **493**
D.1 Structure Package Hierarchy 493
D.2 Principles 495

E Environments **497**
E.1 Downloading Software 497
E.2 Creating Libraries 497
E.3 Creating Project Stationery 498

F Further Reading **501**

G Glossary **503**

Index **507**

Chapter 0

Introduction

Concepts:
- ▷ Approaches to this material
- ▷ Principles

This is an important notice.
Please have it translated.
—The Phone Company

YOUR MOTHER probably provided you with constructive toys, like blocks or Tinker Toys[1] or Legos. These toys are educational: they teach us to think spatially and to build increasingly complex structures. You develop modules that can be stuck together and rules that guide the building process.

If you are reading this book, you probably enjoyed playing with constructive toys. You consider writing programs an artistic process. You have grown from playing with blocks to writing programs. The same guidelines for building structures apply to writing programs, save one thing: there is, seemingly, no limit to the complexity of the programs you can write. *I lie.*

Well, almost. When writing large programs, the *data structures* that maintain the data in your program govern the space and time consumed by your running program. In addition, large programs take time to write. Using different structures can actually have an impact on how long it takes to *write* your program. Choosing the wrong structures can cause your program to run poorly or be difficult or impossible to implement effectively.

Thus, part of the program-writing process is choosing between different structures. Ideally you arrive at solutions by analyzing and comparing their various merits. This book focuses on the creation and analysis of traditional data structures in a modern programming environment, The Java Programming Language, or Java for short.

0.1 Read Me

As might be expected, each chapter is dedicated to a specific topic. Many of the topics are concerned with specific data structures. The structures we will investigate are abstracted from working implementations in Java that are available to you if you have access to the Internet.[2] Other topics concern the

[1] All trademarks are recognized.
[2] For more information, see `http://www.cs.williams.edu/JavaStructures`.

"tools of the trade." Some are mathematical and others are philosophical, but all consider the process of programming well.

The topics we cover are not all-inclusive. Some useful structures have been left out. Instead, we will opt to learn the *principles of programming data structures*, so that, down the road, you can design newer and better structures yourself.

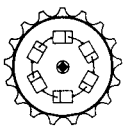
Perhaps the most important aspect of this book is the set of problems at the end of each section. *All are important for you to consider.* For some problems I have attempted to place a reasonable hint or answer in the back of the book. Why should you do problems? Practice makes perfect. I could show you how to ride a unicycle, but if you never practiced, you would never learn. If you study and understand these problems, you will find your design and analytical skills are improved. As for your mother, she'll be proud of you.

Unicycles: the ultimate riding structure.

Sometimes we will introduce problems in the middle of the running text—these problems do not have answers (sometimes they are repeated as formal problems in the back of the chapter, where they *do* have answers)—they should be thought about carefully as you are reading along. You may find it useful to have a pencil and paper handy to help you "think" about these problems on the fly.

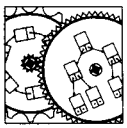
Exercise 0.1 *Call*[3] *your Mom and tell her you're completing your first exercise. If you don't have a phone handy, drop her a postcard. Ask her to verify that she's proud of you.*

This text is brief and to the point. Most of us are interested in experimenting. We will save as much time as possible for solving problems, perusing code, and practicing writing programs. As you read through each of the chapters, you might find it useful to read through the source code online. As we first consider the text of files online, the file name will appear in the margin, as you see here.



Structure

The top icon refers to files in the `structure` package, while the bottom icon refers to files supporting examples.



Example

One more point—this book, like most projects, is an ongoing effort, and the latest thoughts are unlikely to have made it to the printed page. If you are in doubt, turn to the website for the latest comments. You will also find online documentation for each of the structures, generated from the code using `javadoc`. It is best to read the online version of the documentation for the most up-to-date details, as well as the documentation of several structures not formally presented within this text.

0.2 He Can't Say That, Can He?

Sure! Throughout this book are little political comments. These remarks may seem trivial at first blush. Skip them! If, however, you are interested in ways

[3] Don't e-mail her. Call her. Computers aren't everything, and they're a poor medium for a mother's pride.

此为试读，需要完整PDF请访问：www.ertongbook.com