

Reges & Stepp

Stuart Reges & Marty Stepp

BUILDING JAVA™ PROGRAMS
A BACK TO BASICS APPROACH
4TH EDITION

BUILDING JAVA™ PROGRAMS

A BACK TO BASICS APPROACH

4TH EDITION

PEARSON

Reges & Stepp

Stuart Reges & Marty Stepp

BUILDING JAVA™ PROGRAMS
A BACK TO BASICS APPROACH
4TH EDITION

BUILDING JAVA™ PROGRAMS

A BACK TO BASICS APPROACH

4TH EDITION

PEARSON



Building Java Programs

A Back to Basics Approach

Fourth Edition

Stuart Reges

University of Washington

Marty Stepp

Stanford University

PEARSON

Boston Columbus Indianapolis New York San Francisco Hoboken
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal
Toronto Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore
Taipei Tokyo

Vice President, Editorial Director: Marcia Horton

Acquisitions Editor: Matt Goldstein

Editorial Assistant: Kristy Alaura

VP of Marketing: Christy Lesko

Director of Field Marketing: Tim Galligan

Product Marketing Manager: Bram Van Kempen

Field Marketing Manager: Demetrius Hall

Marketing Assistant: Jon Bryant

Director of Product Management: Erin Gregg

Team Lead, Program and Project Management: Scott Disanno

Program Manager: Carole Snyder

Project Manager: Lakeside Editorial Services L.L.C.

Senior Specialist, Program Planning and Support: Maura Zaldivar-Garcia

Cover Design: Joyce Wells

R&P Manager: Rachel Youdelman

R&P Project Manager: Timothy Nicholls

Inventory Manager: Meredith Maresca

Cover Art: Matt Walford/Cultura/Getty Images

Full-Service Project Management: Apoorva Goel/Cenveo Publisher Services

Composition: Cenveo® Publisher Services

Printer/Binder: Edwards Brothers Malloy

Cover Printer: Phoenix Color

Text Font: Monotype

The authors and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The authors and publisher make no warranty of any kind, expressed or implied, with regard to these programs or to the documentation contained in this book. The authors and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Copyright © 2017, 2014 and 2011 Pearson Education, Inc. or its affiliates. All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit www.pearsonhighed.com/permissions/.

Acknowledgements of third party content appear on pages [1193–1194](#), which constitute an extension of this copyright page.

PEARSON, and MYPROGRAMMINGLAB are exclusive trademarks in the U.S. and/or other countries owned by Pearson Education, Inc. or its affiliates.

Unless otherwise indicated herein, any third-party trademarks that may appear in this work are the property of their respective owners and any references to third-party trademarks, logos or other trade dress are for demonstrative or descriptive purposes only. Such references are not intended to imply any sponsorship, endorsement, authorization, or promotion of

Pearson's products by the owners of such marks, or any relationship between the owner and Pearson Education, Inc. or its affiliates, authors, licensees or distributors.

Library of Congress Cataloging-in-Publication Data

Names: Reges, Stuart, author. | Stepp, Martin, author.

Title: Building Java programs : a back to basics approach / Stuart Reges, University of Washington; Marty Stepp, Stanford University.

Description: Fourth Edition. | Hoboken, NJ : Pearson, 2016.

Identifiers: LCCN 2015049340 | ISBN 9780134322766 (alk. paper)

Subjects: LCSH: Java (Computer program language)

Classification: LCC QA76.73.J38 R447 2016 | DDC 005.13/3—dc23 LC record available at <http://lcn.loc.gov/2015049340>

10 9 8 7 6 5 4 3 2 1

The logo for Pearson, consisting of the word "PEARSON" in white, uppercase, sans-serif font, centered within a solid black rectangular background.

ISBN 10: 0-13-432276-2

ISBN 13: 978-0-13-432276-6

Preface

The newly revised fourth edition of our *Building Java Programs* textbook is designed for use in a two-course introduction to computer science. We have class-tested it with thousands of undergraduates, most of whom were not computer science majors, in our CS1-CS2 sequence at the University of Washington. These courses are experiencing record enrollments, and other schools that have adopted our textbook report that students are succeeding with our approach.

Introductory computer science courses are often seen as “killer” courses with high failure rates. But as Douglas Adams says in *The Hitchhiker's Guide to the Galaxy*, “Don't panic.” Students can master this material if they can learn it gradually. Our textbook uses a layered approach to introduce new syntax and concepts over multiple chapters.

Our textbook uses an “objects later” approach where programming fundamentals and procedural decomposition are taught before diving into object-oriented programming. We have championed this approach, which we sometimes call “back to basics,” and have seen through years of experience that a broad range of scientists, engineers, and others can learn how to program in a procedural manner. Once we have built a solid foundation of procedural techniques, we turn to object-oriented programming. By the end of the course, students will have learned about both styles of programming.

Here are some of the changes that we have made in the fourth edition:

- New chapter on functional programming with Java 8. As explained below, we have introduced a chapter that uses the new language features available in Java 8 to discuss the core concepts of functional programming.
- New section on images and 2D pixel array manipulation. Image manipulation is becoming increasingly popular, so we have expanded our `DrawingPane1` class to include features that support manipulating

images as two-dimensional arrays of pixel values. This extra coverage will be particularly helpful for students taking an AP/CS A course because of the heavy emphasis on two-dimensional arrays on the AP exam.

- Expanded self-checks and programming exercises. Many chapters have received new self-check problems and programming exercises. There are roughly fifty total problems and exercises per chapter, all of which have been class-tested with real students and have solutions provided for instructors on our web site.

Since the publication of our third edition, Java 8 has been released. This new version supports a style of programming known as functional programming that is gaining in popularity because of its ability to simply express complex algorithms that are more easily executed in parallel on machines with multiple processors. ACM and IEEE have released new guidelines for undergraduate computer science curricula, including a strong recommendation to cover functional programming concepts.

We have added a new [Chapter 19](#) that covers most of the functional concepts from the new curriculum guidelines. The focus is on concepts, not on language features. As a result, it provides an introduction to several new Java 8 constructs but not a comprehensive coverage of all new language features. This provides flexibility to instructors since functional programming features can be covered as an advanced independent topic, incorporated along the way, or skipped entirely. Instructors can choose to start covering functional constructs along with traditional constructs as early as [Chapter 6](#). See the dependency chart at the end of this section.

The following features have been retained from previous editions:

- Focus on problem solving. Many textbooks focus on language details when they introduce new constructs. We focus instead on problem solving. What new problems can be solved with each construct? What pitfalls are novices likely to encounter along the way? What are the most common ways to use a new construct?
- Emphasis on algorithmic thinking. Our procedural approach allows us to

emphasize algorithmic problem solving: breaking a large problem into smaller problems, using pseudocode to refine an algorithm, and grappling with the challenge of expressing a large program algorithmically.

- Layered approach. Programming in Java involves many concepts that are difficult to learn all at once. Teaching Java to a novice is like trying to build a house of cards. Each new card has to be placed carefully. If the process is rushed and you try to place too many cards at once, the entire structure collapses. We teach new concepts gradually, layer by layer, allowing students to expand their understanding at a manageable pace.
- Case studies. We end most chapters with a significant case study that shows students how to develop a complex program in stages and how to test it as it is being developed. This structure allows us to demonstrate each new programming construct in a rich context that can't be achieved with short code examples. Several of the case studies were expanded and improved in the second edition.
- Utility as a CS1+CS2 textbook. In recent editions, we added chapters that extend the coverage of the book to cover all of the topics from our second course in computer science, making the book usable for a two-course sequence. [Chapters 12–19](#) explore recursion, searching and sorting, stacks and queues, collection implementation, linked lists, binary trees, hash tables, heaps, and more. [Chapter 12](#) also received a section on recursive backtracking, a powerful technique for exploring a set of possibilities for solving problems such as 8 Queens and Sudoku.

Layers and Dependencies

Many introductory computer science books are language-oriented, but the early chapters of our book are layered. For example, Java has many control structures (including for-loops, while-loops, and if/else-statements), and many books include all of these control structures in a single chapter. While that might make sense to someone who already knows how to program, it can

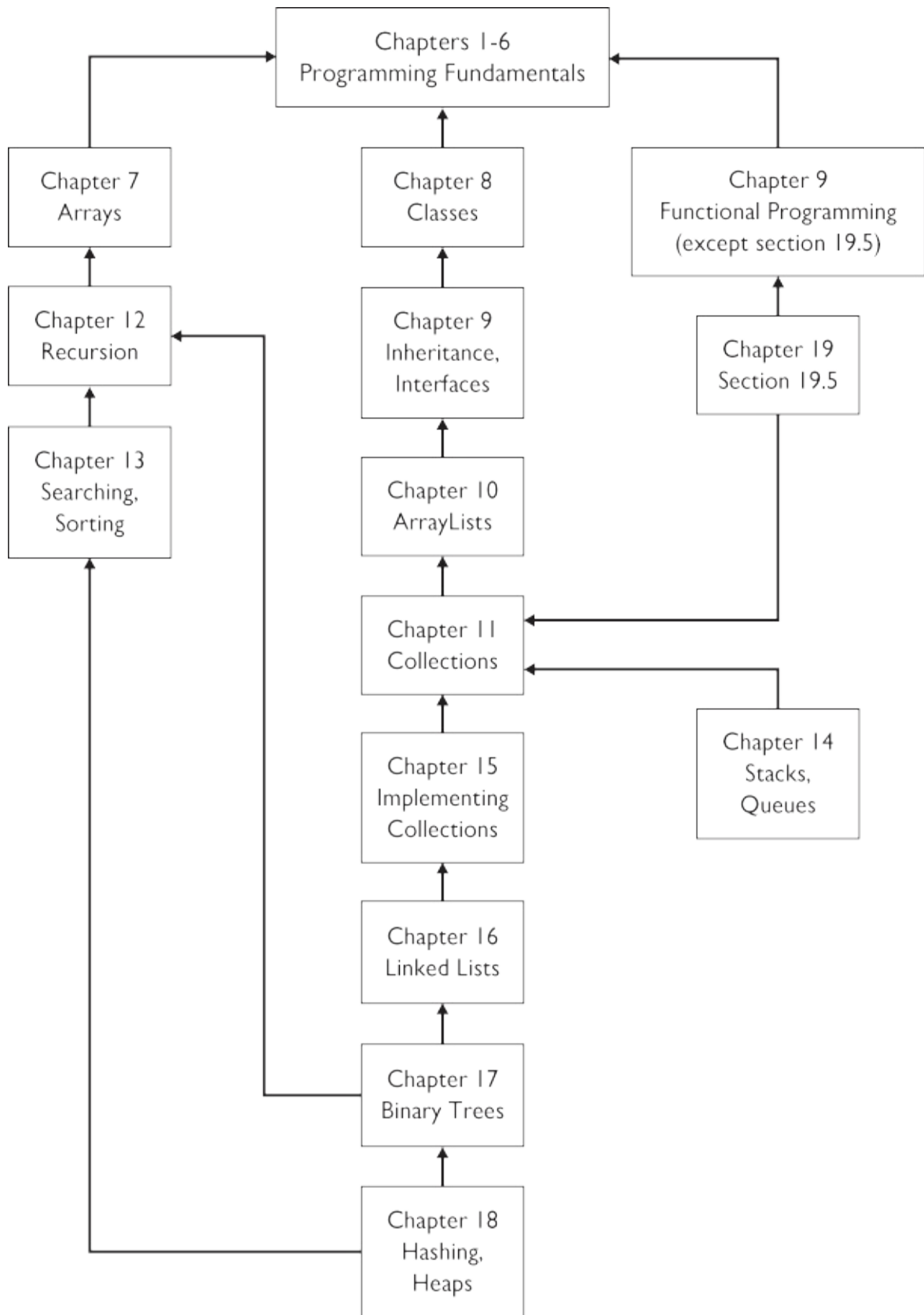
be overwhelming for a novice who is learning how to program. We find that it is much more effective to spread these control structures into different chapters so that students learn one structure at a time rather than trying to learn them all at once.

The following table shows how the layered approach works in the first six chapters:

Chapter	Control Flow	Data	Programming Techniques	Input/Output
1	methods	String literals	procedural decomposition	println, print
2	definite loops (for)	variables, expressions, int, double	local variables, class constants, pseudocode	
3	return values	using objects	parameters	console input, 2D graphics (optional)
4	conditional (if/else)	char	pre/post conditions, throwing exceptions	printf
5	indefinite loops (while)	boolean	assertions, robust programs	
6		Scanner	token/line-based file processing	file I/O

[Chapters 1–6](#) are designed to be worked through in order, with greater flexibility of study then beginning in [Chapter 7](#). [Chapter 6](#) may be skipped, although the case study in [Chapter 7](#) involves reading from a file, a topic that is covered in [Chapter 6](#).

The following is a dependency chart for the book:



Supplements

<http://www.buildingjavaprograms.com/>

Answers to all self-check problems appear on our web site and are accessible to anyone. Our web site has the following additional resources for students:

- Online-only supplemental chapters, such as a chapter on creating Graphical User Interfaces
- Source code and data files for all case studies and other complete program examples
- The **DrawingPane1 class** used in the optional graphics Supplement 3G

Our web site has the following additional resources for teachers:

- PowerPoint slides suitable for lectures
- Solutions to exercises and programming projects, along with homework specification documents for many projects
- Sample exams and solution keys
- Additional lab exercises and programming exercises with solution keys
- Closed lab creation tools to produce lab handouts with the instructor's choice of problems integrated with the textbook

To access protected instructor resources, contact us at authors@buildingjavaprograms.com. The same materials are also available at <http://www.pearsonhighered.com/cs-resources>. To receive a password for this site or to ask other questions related to resources, contact your Pearson sales representative.

MyProgrammingLab

MyProgrammingLab is an online practice and assessment tool that helps students fully grasp the logic, semantics, and syntax of programming. Through practice exercises and immediate, personalized feedback, MyProgrammingLab improves the programming competence of beginning students who often struggle with basic concepts and paradigms of popular high-level programming languages. A self-study and homework tool, the MyProgrammingLab course consists of hundreds of small practice exercises organized around the structure of this textbook. For students, the system automatically detects errors in the logic and syntax of code submissions and offers targeted hints that enable students to figure out what went wrong, and why. For instructors, a comprehensive grade book tracks correct and incorrect answers and stores the code inputted by students for review.

For a full demonstration, to see feedback from instructors and students, or to adopt MyProgrammingLab for your course, visit the following web site:
<http://www.myprogramminglab.com/>

VideoNotes



We have recorded a series of instructional videos to accompany the textbook. They are available at the following web site: www.pearsonhighered.com/cs-resources

Roughly 3–4 videos are posted for each chapter. An icon in the margin of the page indicates when a VideoNote is available for a given topic. In each video, we spend 5–15 minutes walking through a particular concept or problem, talking about the challenges and methods necessary to solve it. These videos make a good supplement to the instruction given in lecture classes and in the textbook. Your new copy of the textbook has an access code that will allow you to view the videos.

Acknowledgments

First, we would like to thank the many colleagues, students, and teaching assistants who have used and commented on early drafts of this text. We could not have written this book without their input. Special thanks go to H el ene Martin, who pored over early versions of our first edition chapters to find errors and to identify rough patches that needed work. We would also like to thank instructor Benson Limketkai for spending many hours performing a technical proofread of the second edition.

Second, we would like to thank the talented pool of reviewers who guided us in the process of creating this textbook:

- Greg Anderson, Weber State University
- Delroy A. Brinkerhoff, Weber State University
- Ed Brunjes, Miramar Community College
- Tom Capaul, Eastern Washington University
- Tom Cortina, Carnegie Mellon University
- Charles Dierbach, Towson University
- H.E. Dunsmore, Purdue University
- Michael Eckmann, Skidmore College
- Mary Anne Egan, Siena College
- Leonard J. Garrett, Temple University
- Ahmad Ghafarian, North Georgia College & State University
- Raj Gill, Anne Arundel Community College

- Michael Hostetler, Park University
- David Hovemeyer, York College of Pennsylvania
- Chenglie Hu, Carroll College
- Philip Isenhour, Virginia Polytechnic Institute
- Andree Jacobson, University of New Mexico
- David C. Kamper, Sr., Northeastern Illinois University
- Simon G.M. Koo, University of San Diego
- Evan Korth, New York University
- Joan Krone, Denison University
- John H.E.F. Lasseter, Fairfield University
- Eric Matson, Wright State University
- Kathryn S. McKinley, University of Texas, Austin
- Jerry Mead, Bucknell University
- George Medelinskas, Northern Essex Community College
- John Neitzke, Truman State University
- Dale E. Parson, Kutztown University
- Richard E. Pattis, Carnegie Mellon University
- Frederick Pratter, Eastern Oregon University
- Roger Priebe, University of Texas, Austin
- Dehu Qi, Lamar University

- John Rager, Amherst College
- Amala V.S. Rajan, Middlesex University
- Craig Reinhart, California Lutheran University
- Mike Scott, University of Texas, Austin
- Alexa Sharp, Oberlin College
- Tom Stokke, University of North Dakota
- Leigh Ann Sudol, Fox Lane High School
- Ronald F. Taylor, Wright State University
- Andy Ray Terrel, University of Chicago
- Scott Thede, DePauw University
- Megan Thomas, California State University, Stanislaus
- Dwight Tuinstra, SUNY Potsdam
- Jeannie Turner, Sayre School
- Tammy VanDeGrift, University of Portland
- Thomas John VanDrunen, Wheaton College
- Neal R. Wagner, University of Texas, San Antonio
- Jiangping Wang, Webster University
- Yang Wang, Missouri State University
- Stephen Weiss, University of North Carolina at Chapel Hill
- Laurie Werner, Miami University

- Dianna Xu, Bryn Mawr College
- Carol Zander, University of Washington, Bothell

Finally, we would like to thank the great staff at Pearson who helped produce the book. Michelle Brown, Jeff Holcomb, Maurene Goo, Patty Mahtani, Nancy Kotary, and Kathleen Kenny did great work preparing the first edition. Our copy editors and the staff of Aptara Corp, including Heather Sisan, Brian Baker, Brendan Short, and Rachel Head, caught many errors and improved the quality of the writing. Marilyn Lloyd and Chelsea Bell served well as project manager and editorial assistant respectively on prior editions. For their help with the third edition we would like to thank Kayla Smith-Tarbox, Production Project Manager, and Jenah Blitz-Stoehr, Computer Science Editorial Assistant. Mohinder Singh and the staff at Aptara, Inc., were also very helpful in the final production of the third edition. For their great work on production of the fourth edition, we thank Louise Capulli and the staff of Lakeside Editorial Services, along with Carole Snyder at Pearson. Special thanks go to our lead editor at Pearson, Matt Goldstein, who has believed in the concept of our book from day one. We couldn't have finished this job without all of their hard work and support.

Stuart Reges

Marty Stepp

Break through

To Improving results

MyProgrammingLab™

Through the power of practice and immediate personalized feedback, MyProgrammingLab helps improve your students' performance.

Programming Practice

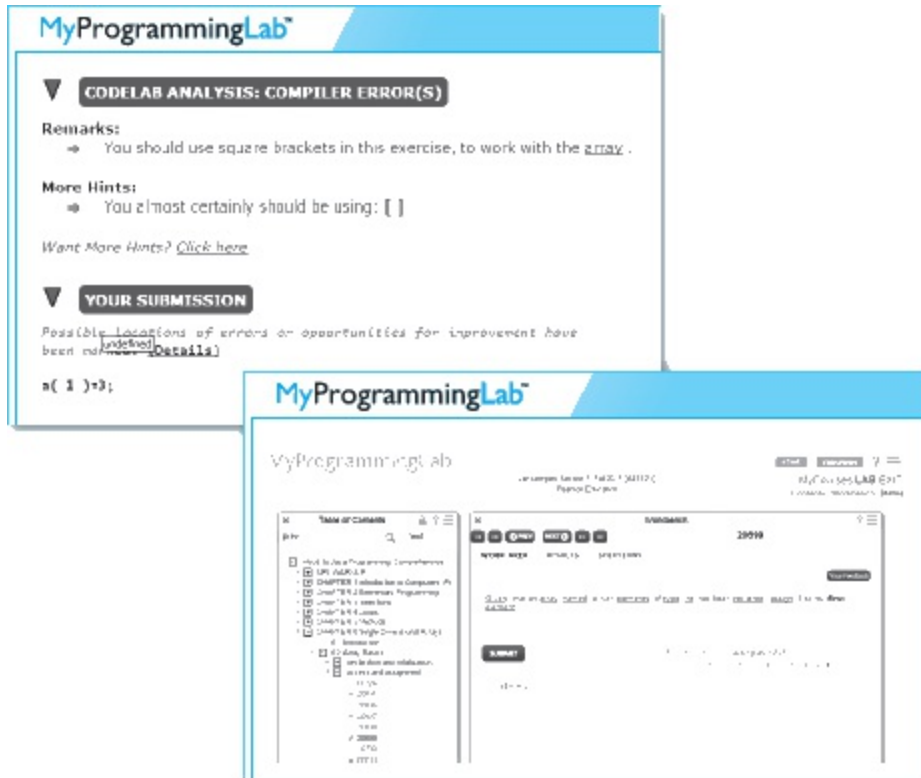
With MyProgrammingLab, your students will gain first-hand programming experience in an interactive online environment.

Immediate, Personalized Feedback

MyProgrammingLab automatically detects errors in the logic and syntax of their code submission and offers targeted hints that enables students to figure out what went wrong and why.

Graduated Complexity

MyProgrammingLab breaks down programming concepts into short, understandable sequences of exercises. Within each sequence the level and sophistication of the exercises increase gradually but steadily.



Dynamic Roster

Students' submissions are stored in a roster that indicates whether the submission is correct, how many attempts were made, and the actual code submissions from each attempt.

Pearson eText

The Pearson eText gives students access to their textbook anytime, anywhere

Step-By-Step Videonote Tutorials

These step-by-step video tutorials enhance the programming concepts presented in select Pearson textbooks.

For more information and titles available with MyProgrammingLab, please visit www.myprogramminglab.com.

Copyright © 2016 Pearson Education, Inc. or its affiliate(s). All rights reserved. HELO88173 · 11/15

LOCATION OF VIDEO NOTES IN THE TEXT



www.pearsonhighered.com/cs-resources VideoNote

Chapter 1	Pages 31 , 40
Chapter 2	Pages 65 , 74 , 89 , 97 , 110
Chapter 3	Pages 141 , 156 , 161 , 167
Chapter 3G	Pages 197 , 215
Chapter 4	Pages 243 , 251 , 278
Chapter 5	Pages 324 , 327 , 329 , 333 , 356
Chapter 6	Pages 396 , 409 , 423
Chapter 7	Pages 458 , 465 , 484 , 505
Chapter 8	Pages 535 , 547 , 555 , 568
Chapter 9	Pages 597 , 610 , 626
Chapter 10	Pages 672 , 677 , 686
Chapter 11	Pages 716 , 729 , 737
Chapter 12	Pages 764 , 772 , 809
Chapter 13	Pages 834 , 837 , 843
Chapter 14	Pages 889 , 896
Chapter 15	Pages 930 , 936 , 940
Chapter 16	Pages 972 , 979 , 992
Chapter 17	Pages 1037 , 1038 , 1048
Chapter 18	Pages 1073 , 1092

Brief Contents

1. [Chapter 1 Introduction to Java Programming 1](#)
2. [Chapter 2 Primitive Data and Definite Loops 63](#)
3. [Chapter 3 Introduction to Parameters and Objects 137](#)
4. [Supplement 3G Graphics \(Optional\) 196](#)
5. [Chapter 4 Conditional Execution 238](#)
6. [Chapter 5 Program Logic and Indefinite Loops 315](#)
7. [Chapter 6 File Processing 387](#)
8. [Chapter 7 Arrays 443](#)
9. [Chapter 8 Classes 530](#)
10. [Chapter 9 Inheritance and Interfaces 587](#)
11. [Chapter 10 ArrayLists 662](#)
12. [Chapter 11 Java Collections Framework 715](#)
13. [Chapter 12 Recursion 754](#)
14. [Chapter 13 Searching and Sorting 832](#)
15. [Chapter 14 Stacks and Queues 884](#)
16. [Chapter 15 Implementing a Collection Class 922](#)
17. [Chapter 16 Linked Lists 965](#)
18. [Chapter 17 Binary Trees 1017](#)

19. [Chapter 18 Advanced Data Structures 1071](#)
20. [Chapter 19 Functional Programming with Java 1107](#)
 1. [Appendix A Java Summary 1149](#)
 2. [Appendix B The Java API Specification and Javadoc Comments 1164](#)
 3. [Appendix C Additional Java Syntax 1170](#)

Contents

1. [Chapter 1 Introduction to Java Programming 1](#)
 1. [1.1 Basic Computing Concepts 2](#)
 1. [Why Programming? 2](#)
 2. [Hardware and Software 3](#)
 3. [The Digital Realm 4](#)
 4. [The Process of Programming 6](#)
 5. [Why Java? 7](#)
 6. [The Java Programming Environment 8](#)
 2. [1.2 And Now—Java 10](#)
 1. [String Literals \(Strings\) 14](#)
 2. [System.out.println 15](#)
 3. [Escape Sequences 15](#)
 4. [print versus println 17](#)
 5. [Identifiers and Keywords 18](#)
 6. [A Complex Example: DrawFigures1 20](#)
 7. [Comments and Readability 21](#)
 3. [1.3 Program Errors 24](#)
 1. [Syntax Errors 24](#)

2. [Logic Errors \(Bugs\) 28](#)
4. [1.4 Procedural Decomposition 28](#)
 1. [Static Methods 31](#)
 2. [Flow of Control 34](#)
 3. [Methods That Call Other Methods 36](#)
 4. [An Example Runtime Error 39](#)
5. [1.5 Case Study: DrawFigures 40](#)
 1. [Structured Version 41](#)
 2. [Final Version without Redundancy 43](#)
 3. [Analysis of Flow of Execution 44](#)
2. [Chapter 2 Primitive Data and Definite Loops 63](#)
 1. [2.1 Basic Data Concepts 64](#)
 1. [Primitive Types 64](#)
 2. [Expressions 65](#)
 3. [Literals 67](#)
 4. [Arithmetic Operators 68](#)
 5. [Precedence 70](#)
 6. [Mixing Types and Casting 73](#)
 2. [2.2 Variables 74](#)
 1. [Assignment/Declaration Variations 79](#)

2. [String Concatenation 82](#)
3. [Increment/Decrement Operators 84](#)
4. [Variables and Mixing Types 87](#)
3. [2.3 The for Loop 89](#)
 1. [Tracing for Loops 91](#)
 2. [for Loop Patterns 95](#)
 3. [Nested for Loops 97](#)
4. [2.4 Managing Complexity 99](#)
 1. [Scope 99](#)
 2. [Pseudocode 105](#)
 3. [Class Constants 108](#)
5. [2.5 Case Study: Hourglass Figure 110](#)
 1. [Problem Decomposition and Pseudocode 111](#)
 2. [Initial Structured Version 113](#)
 3. [Adding a Class Constant 114](#)
 4. [Further Variations 117](#)
3. [Chapter 3 Introduction to Parameters and Objects 137](#)
 1. [3.1 Parameters 138](#)
 1. [The Mechanics of Parameters 141](#)
 2. [Limitations of Parameters 145](#)

3. [Multiple Parameters 148](#)
4. [Parameters versus Constants 151](#)
5. [Overloading of Methods 151](#)
2. [3.2 Methods That Return Values 152](#)
 1. [The Math Class 153](#)
 2. [Defining Methods That Return Values 156](#)
3. [3.3 Using Objects 160](#)
 1. [String Objects 161](#)
 2. [Interactive Programs and Scanner Objects 167](#)
 3. [Sample Interactive Program 170](#)
4. [3.4 Case Study: Projectile Trajectory 173](#)
 1. [Unstructured Solution 177](#)
 2. [Structured Solution 179](#)
4. [Supplement 3G Graphics \(Optional\) 196](#)
 1. [3G.1 Introduction to Graphics 197](#)
 1. [DrawingPanel 197](#)
 2. [Drawing Lines and Shapes 198](#)
 3. [Colors 203](#)
 4. [Drawing with Loops 206](#)
 5. [Text and Fonts 210](#)

6. [Images 213](#)
2. [3G.2 Procedural Decomposition with Graphics 215](#)
 1. [A Larger Example: DrawDiamonds 216](#)
3. [3G.3 Case Study: Pyramids 219](#)
 1. [Unstructured Partial Solution 220](#)
 2. [Generalizing the Drawing of Pyramids 222](#)
 3. [Complete Structured Solution 223](#)
5. [Chapter 4 Conditional Execution 238](#)
 1. [4.1 if/else Statements 239](#)
 1. [Relational Operators 241](#)
 2. [Nested if/else Statements 243](#)
 3. [Object Equality 250](#)
 4. [Factoring if/else Statements 251](#)
 5. [Testing Multiple Conditions 253](#)
 2. [4.2 Cumulative Algorithms 254](#)
 1. [Cumulative Sum 254](#)
 2. [Min/Max Loops 256](#)
 3. [Cumulative Sum with if 260](#)
 4. [Roundoff Errors 262](#)
 3. [4.3 Text Processing 265](#)

1. [The char Type 265](#)
2. [char versus int 266](#)
3. [Cumulative Text Algorithms 267](#)
4. [system.out.printf 269](#)
4. [4.4 Methods with Conditional Execution 274](#)
 1. [Preconditions and Postconditions 274](#)
 2. [Throwing Exceptions 274](#)
 3. [Revisiting Return Values 278](#)
 4. [Reasoning about Paths 283](#)
5. [4.5 Case Study: Body Mass Index 285](#)
 1. [One-Person Unstructured Solution 286](#)
 2. [Two-Person Unstructured Solution 289](#)
 3. [Two-Person Structured Solution 291](#)
 4. [Procedural Design Heuristics 295](#)
6. [Chapter 5 Program Logic and Indefinite Loops 315](#)
 1. [5.1 The while Loop 316](#)
 1. [A Loop to Find the Smallest Divisor 317](#)
 2. [Random Numbers 320](#)
 3. [Simulations 324](#)
 4. [do/while Loop 325](#)

2. [5.2 Fencepost Algorithms 327](#)
 1. [Sentinel Loops 329](#)
 2. [Fencepost with if 330](#)
3. [5.3 The boolean Type 333](#)
 1. [Logical Operators 335](#)
 2. [Short-Circuited Evaluation 338](#)
 3. [boolean Variables and Flags 342](#)
 4. [Boolean Zen 344](#)
 5. [Negating Boolean Expressions 347](#)
4. [5.4 User Errors 348](#)
 1. [Scanner Lookahead 349](#)
 2. [Handling User Errors 351](#)
5. [5.5 Assertions and Program Logic 353](#)
 1. [Reasoning about Assertions 355](#)
 2. [A Detailed Assertions Example 356](#)
6. [5.6 Case Study: NumberGuess 361](#)
 1. [Initial Version without Hinting 361](#)
 2. [Randomized Version with Hinting 363](#)
 3. [Final Robust Version 367](#)
7. [Chapter 6 File Processing 387](#)

1. [6.1 File-Reading Basics 388](#)
 1. [Data, Data Everywhere 388](#)
 2. [Files and File Objects 388](#)
 3. [Reading a File with a Scanner 391](#)
2. [6.2 Details of Token-Based Processing 396](#)
 1. [Structure of Files and Consuming Input 398](#)
 2. [Scanner Parameters 403](#)
 3. [Paths and Directories 404](#)
 4. [A More Complex Input File 407](#)
3. [6.3 Line-Based Processing 409](#)
 1. [String Scanners and Line/Token Combinations 410](#)
4. [6.4 Advanced File Processing 415](#)
 1. [Output Files with PrintStream 415](#)
 2. [Guaranteeing That Files Can Be Read 420](#)
5. [6.5 Case Study: Zip Code Lookup 423](#)
8. [Chapter 7 Arrays 443](#)
 1. [7.1 Array Basics 444](#)
 1. [Constructing and Traversing an Array 444](#)
 2. [Accessing an Array 448](#)
 3. [A Complete Array Program 451](#)