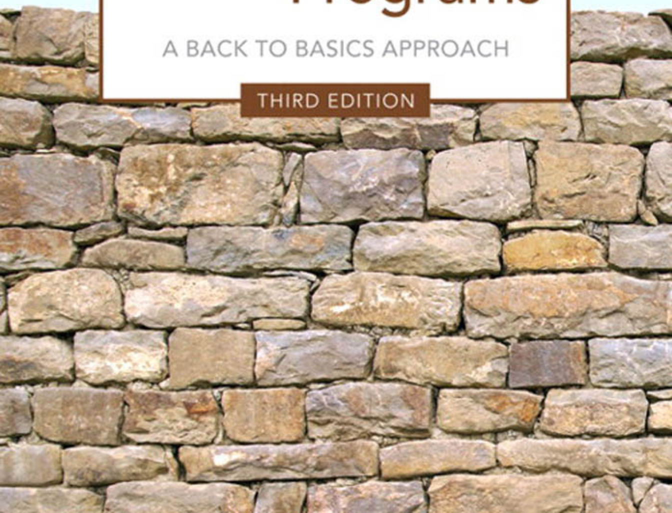
Stuart Reges + Marty Stepp

Building Java Programs

A BACK TO BASICS APPROACH

THIRD EDITION



Third Edition

Building Java Programs

A Back to Basics Approach

Stuart Reges | Marty Stepp

University of Washington

PEARSON

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

Editorial Director, ECS: Marcia Horton
Executive Editor: Matt Goldstein
Editorial Assistant: Jenah Blitz-Stoehr
Director of Marketing: Christy Lesko
Marketing Manager: Yezan Alayan
Senior Marketing Coordinator: Kathryn Ferranti
Director of Production: Erin Gregg
Senior Managing Editor: Scott Disanno
Production Project Manager: Kayla Smith-Tarbox
Manufacturing Buyer: Lisa McDowell
Art Director: Anthony Gemellaro
Cover Designer: Joyce Wells
Manager, Rights and Permissions: Michael Joyce
Text Permission Coordinator: Jackie Bates, GEX
Cover Image: Dimitar Todorov/Alamy
Media Project Manager: Renata Butera
Full-Service Project Management: Mohinder Singh/ Aptara®, Inc.
Composition: Aptara®, Inc.
Printer/Binder: Edwards Brothers
Cover Printer: Edwards Brothers
Text Font: Times

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on the appropriate page within text and on page 1145.

Library of Congress Cataloging-in-Publication Data
Reges, Stuart.
Building Java programs : a back to basics approach / Stuart Reges, Marty Stepp, University of Washington.—Third edition.
pages cm
ISBN-13: 978-0-13-336090-5
ISBN-10: 0-13-336090-3
1. Java (Computer program language) I. Stepp, Martin. II. Title.
QA76.73.J38R447 2014
005.13'3—dc23
2012045404

10 9 8 7 6 5 4 3 2 1

PEARSON

ISBN 13: 978-0-13-336090-5
ISBN 10: 0-13-336090-3

Preface

The newly revised *Building Java Programs* textbook is designed for use in a two-course introduction to computer science. We received such positive feedback on the new chapters that we added in the second edition that we have gone even further to make this book useful for both the first and second course in computer science. We have class-tested it with thousands of undergraduates at the University of Washington, most of whom were not computer science majors.

Introductory computer science courses have a long history at many universities of being "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. The introductory courses at the University of Washington are experiencing record enrollments, and other schools that have adopted our textbook report that students are succeeding with our approach.

Since the publication of our first two editions, there has been a movement toward the "objects later" approach that we have championed (as opposed to the "objects early" approach). We know from 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 third edition:

- **Two new chapters.** We have created new chapters that extend the coverage of the book, using material that we present in our second course in computer science. Chapter 14 explores programming with stacks and queues. Chapter 18 examines the implementation of hash tables and heaps. These expand on Chapters 15–17 added in the second edition that discuss implementation of collection classes using arrays, linked lists, and binary trees.
- **New section on recursive backtracking.** Backtracking is a powerful technique for exploring a set of possibilities for solving a problem. Chapter 12 now has a section on backtracking and examines several problems in detail, including the 8 Queens problem and Sudoku.
- **Expanded self-checks and programming exercises.** We have significantly increased the number and quality of self-check exercises and programming exercises incorporating new problems in each chapter. There are now 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.

The following features have been retained from the first edition:

- **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.

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.

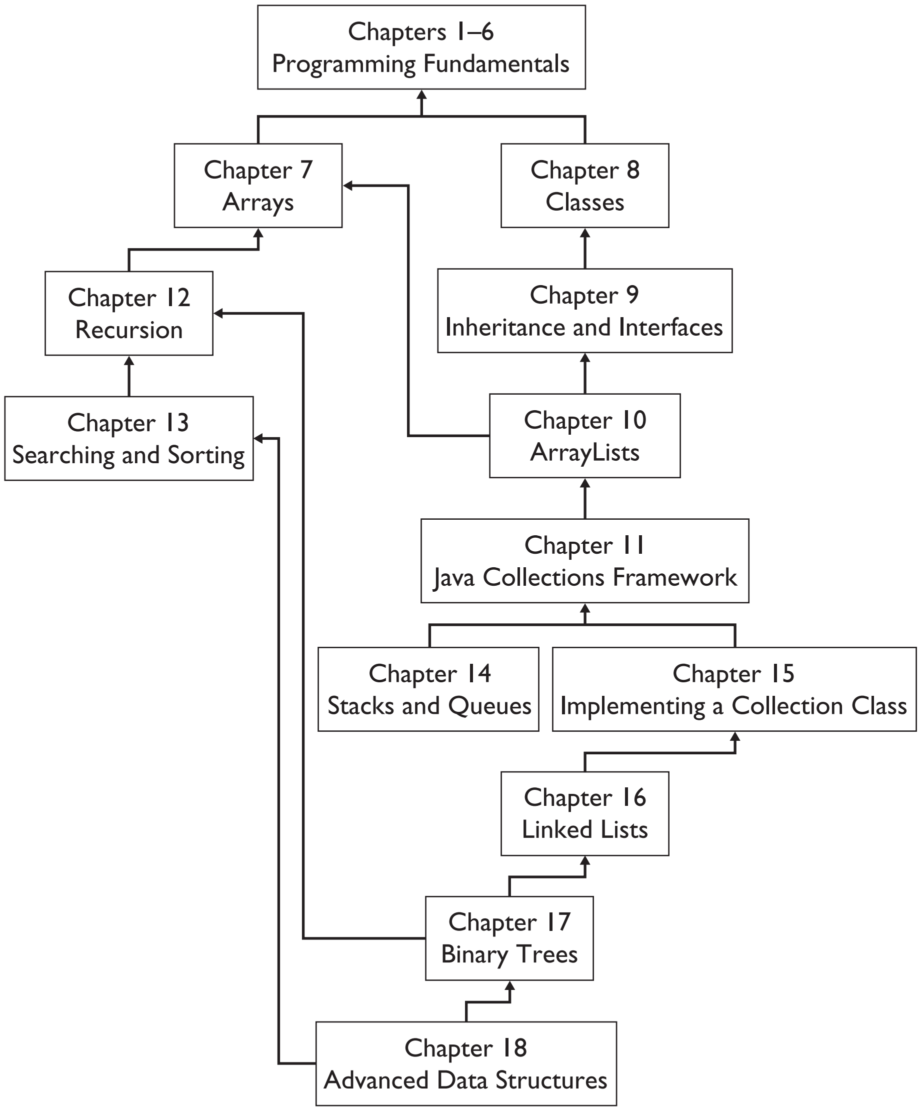
The following table shows how the layered approach works in the first six chapters:

The Layers

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 graphics (optional)
4	conditional `(if/else)`	`char`	pre/post conditions throwing exceptions	`printf`
5	indefinite loops `(while)`	`boolean`	assertions robust programs	
6		`Scanner`	token-based processing line-based processing	file input/output

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

Answers to all self-check problems appear on our web site at http://www.building javaprograms.com/ and are accessible to anyone. Our web site also has the following additional resources available 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 **`DrawingPanel` class** used in the optional graphics Supplement 3G

Instructors can access the following resources from our web site at http://www.buildingjavaprograms.com/:

- **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/regesstepp/. 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 gradebook 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 www.myprogramminglab.com.

VideoNotes

VideoNote

We have recorded a series of instructional videos to accompany the textbook. They are available at http://www.pearsonhighered.com/regesstepp. 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