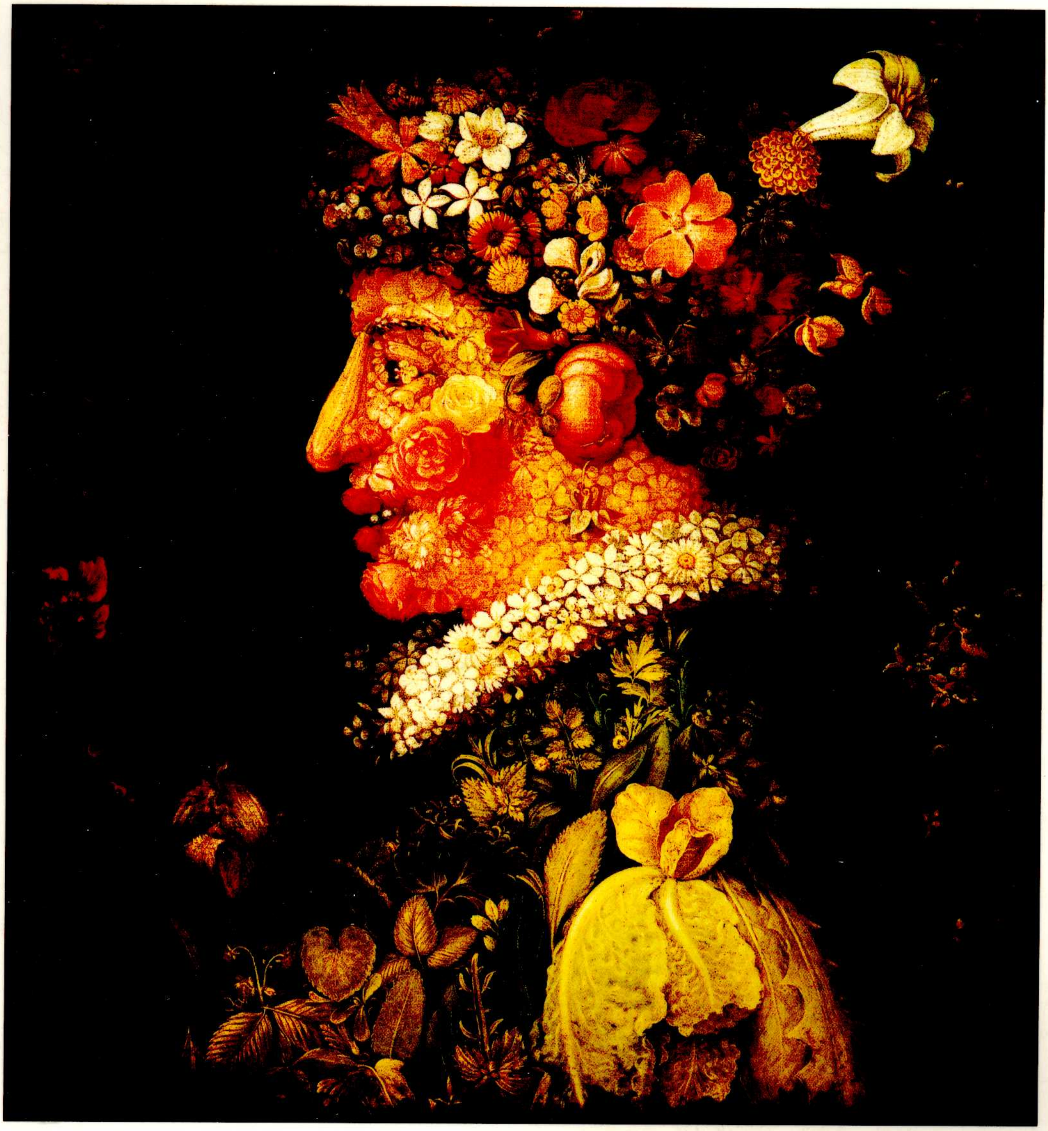
SECOND EDITION

programming.java™

An Introduction to Programming Using Java



DECKER HIRSHFIELD

programming.java

An Introduction to Programming Using Java™

Second Edition

Rick Decker
Stuart Hirshfield

Hamilton College

Brooks/Cole
Thomson Learning™

Australia • Canada • Mexico • Singapore • Spain • United Kingdom • United States

Sponsoring Editor: *Kallie Swanson*
Marketing Team: *Nathan Wilbur, Christina De Veto, Samantha Cabaluna*
Editorial Assistant: *Grace Fujimoto*
Production Coordinator: *Kelsey McGee*
Production Service: *Publishers' Design and Production Services, Inc.*
Manuscript Editor: *Connie Day*
Permissions Editor: *Mary Kay Hancharick*
Cover Design: *Laurie Albrecht*
Typesetting: *Publishers' Design and Production Services, Inc.*
Cover Printing: *Phoenix Color Corporation*
Printing and Binding: *R. R. Donnelley and Sons/ Crawfordsville*

For more information, contact:
BROOKS/COLE
511 Forest Lodge Road
Pacific Grove, CA 93950 USA
www.brookscole.com

For permission to use material from this work, contact us by
Web: www.thomsonrights.com
fax: 1-800-730-2215
phone: 1-800-730-2214

Printed in United States of America

10 9 8 7 6 5 4 3 2

Library of Congress Cataloging-in-Publication Data

Decker, Rick.
Programming.Java : an introduction to programming using Java / Rick Decker, Stuart Hirshfield.—2nd ed.
p. cm.
ISBN 0-534-37109-4 (pbk.)
1. Java (Computer program language) I. Hirshfield, Stuart. II. Title.
QA76.73.J38D44 1999 99-38991
005.13'3—dc21

THIS BOOK IS PRINTED ON ACID-FREE RECYCLED PAPER

To our students,
understanding as we do the ups,

> *Many are our joys*
> *In youth, but oh! What happiness to live*
> *When every hour brings palpable access*
> *Of knowledge, when all knowledge is delight*
> *And sorrow is not there!*
>
> —William Wordsworth, *The Prelude*

And the downs,

> *My life is one demd horrid grind.*
>
> —Charles Dickens, *Oliver Twist*

PREFACE

This second edition of "programming dot java" (like the first) is intended to be used in a one- or two-semester introduction to programming using the Java™ language. We assume no prior programming experience on the part of our audience. In the tradition of new releases of software, we'll begin with a revision history of this product, describing the features of the original version ("edition," if you prefer), and the changes we have made.

Revision History

`programming java 1.0`

Hands-on experience. In a separate lab manual—available in print and on-line formats—we provide students with detailed, directed, experimental exercises which help them to explore firsthand the principles of OOP and Java in a controlled fashion. These exercises are integrated precisely with the textual material and bring the static text material to life. The text introduces students to new concepts. As they read the text, students encounter a number of sample programs that illustrate the new concepts. Then, in the laboratory, they experiment with "Lablet" programs and extend them using what they have learned from the text. All the sample programs will be provided on a disk that will accompany the lab manual.

Objects from the start. It's almost impossible to avoid classes, objects, and inheritance in a Java program. This means that we could build in an object-oriented approach from the very beginning, rather than just tacking it on later. Specifically, this text introduces students to Java and OOP from the top down. That is, we start by presenting the empowering features of Java and OOP—classes, packages, and inheritance—and defer the algorithmic details until later. After a quick generic introduction to computers and programming, roughly the first third of the text is devoted to using and experimenting with Java's Abstract Windowing Toolkit (the "AWT"). Doing so allows us to provide students with lots of algorithmically simple code to read and to use, all of which emphasizes basic OOP notations and ideas while providing tangible graphical output. Our overriding motivation in this approach is to make sure our students are introduced to object-oriented design early, before any bad habits can set in. By the end of this section of the course, students will be able to use the AWT to describe arbitrarily complex graphical user interfaces (GUIs). More important, they will be completely conversant in the basics of OOP and the use of Java packages.

Graphical orientation. Java contains a rich class library containing complete, platform-independent support for event-driven, graphically-oriented programming.

That is, users can interact with Java programs in a modeless manner by means of a contemporary graphical user interface, rather than by responding to program prompts by typing text into a console window. Just as the language dictates a reliance on object-oriented design, the class library makes it virtually impossible to avoid an approach that emphasizes code reuse. From the instructor's point of view, we gain a natural context for discussing important principles of software design, and our students can build visually interesting programs from the very start. After all, our audience has been raised on programs that have a rich graphical user interface: Java allows them to build the kind of programs they're used to seeing.

programming java 2.0

Java 2 compatibility. When we were writing version 1.0 of this text, the Java language was in the process of a major change, placing us in the uncomfortable position of releasing a book that was aimed at a moving target. Realizing that it would be a while before compiler writers caught up with the language change, we gritted our teeth and wrote it so that it discussed both the old and the (then) new versions of Java. Just to keep us on our toes (or so it seemed), as we were writing the current version, Sun Microsystems introduced yet another version—Java 2. Fortunately, this change entailed only the smallest of modifications, other than a decision to omit—for the moment—any detailed coverage of the new library packages (particularly Swing) that comprise Java 2. This means that the labs, lablets, and language features covered here are compatible with any version of Java with number 1.1 or higher.

New Lablets, new labs. We've streamlined many of the sample Lablets and completely replaced some, in the interest of not clobbering our readers with too much detail. The new model of event handling introduced in Java 1.1 necessitated changes in most of the Lablets, and, most important, we squashed a number of bugs that cropped up in the earlier programs. Of course, the changes in the Lablets resulted in a number of changes in the associated labs, as well.

New features. This version of the text includes review self-test questions at the end of nearly every section, with answers provided near the end of each chapter. We've also added end-of chapter reference sections that describe in compact form all the classes introduced in the chapter. In the back of the book, we've included a glossary of new terms and a complete rewrite of the atrocious index that came with the first version.

Coverage

You won't find the traditional CS 1 chapters, "Loops" and "Selection Statements," here. Our approach, as we've explained, is to cover object-oriented design and programming from the beginning, and to adopt a "just in time" approach to the algorithmic language features. **Chapter 1** provides a background context, discussing the evolution of programming, the Internet and the Worldwide Web, and providing an overview of Java.

The next three chapters introduce the AWT fundamentals. **Chapter 2** discusses the Applet class and graphical programming. **Chapter 3** introduces the basic graphical user interface components, such as buttons, labels, textfields, and checkboxes. The Lablet for this chapter uses all the components in the front end of an online ordering program. **Chapter 4** continues the investigation of the AWT classes by discussing containers, layouts, windows, frames, dialogs, and menus.

Chapter 5 begins our transition from using the AWT classes for visual design to writing programs that actively interact with the user. Here we introduce the Java language features that students will need, discussing primitive types, identifiers, scope, access, expressions, and statements. **Chapter 6** introduces event-driven programming and, now that they're necessary, the Java selection statements. **Chapter 7** recapitulates everything that we've introduced and then explores the design process in detail, moving from a vaguely-worded description of a problem to a fully functional simulation of an automatic teller machine. **Chapter 8** discusses arrays and the String class, introducing loops. We begin to discuss algorithmic programming by producing a visual sorting demonstration.

The remaining four chapters cover the other classes needed to produce all but the more arcane Java programs. **Chapter 9** introduces exceptions. **Chapter 10** discusses file input and output, culminating with the design and exploration of a word processor. **Chapter 11** introduces threads. **Chapter 12** concludes the book, describing how to make an applet interact with the Web, using sounds, images, and animation.

programming.java contains more material than most instructors will wish to introduce in a single semester. Many people we've talked to indicate that they cover most of Chapters 1 through 8 and then, depending on their interests and the interests of their students pick a topic or two from the remaining four chapters. We've heard that some schools have what might be called a "CS 1.5" course—covering Chapters 8 through 12 along with a healthy dose of software engineering—either as a requirement or an elective between CS 1 and the traditional Data Structures course that usually follows CS 1.

Thanks

programming.java, second edition wouldn't exist were it not for the efforts of a number of diligent and helpful people, many of whom we're proud to count among our friends. Our heartfelt thanks go out to the people at Brooks/Cole, especially Kallie Swanson, our editor; Grace Fujimoto, Brooks/Cole's editorial assistant for computer science titles; and Kelsey McGee, the production coordinator; Nathan Wilbur, senior marketing manager for engineering and computer science; and Mark Bergeron, of Publishers' Design and Production Services, Inc. Thanks also go to Connie Day, for reading the manuscript and magically transforming our sow's ear into a silk purse of accepted standards of punctuation, grammar, and orthography, and to Michael Fry of Lebanon Valley College, Kent Jones of Whitworth College, Robert Kline of West Chester University, J. Richard Rinewalt of Texas Christian University, Celia Schahczenski of Montana Tech of the University of Montana, and Susanne Steiger of Massachusetts Bay Community College for their detailed, thorough, and enormously helpful reviews of this edition. Special thanks, finally, go to our families for putting up with frequently absent and always bemused husbands and fathers. As Brian Dennehy said about his family in his Tony acceptance speech, "they paid for this with that most precious of coins, time." Writing and producing a book is a process that ranks right up there on the misery scale with cholera, except that it takes longer. It can never be called a pleasant experience, but the help, support, and warmth of all involved at least made it bearable.

Rick Decker
Stuart Hirshfield

CONTENTS