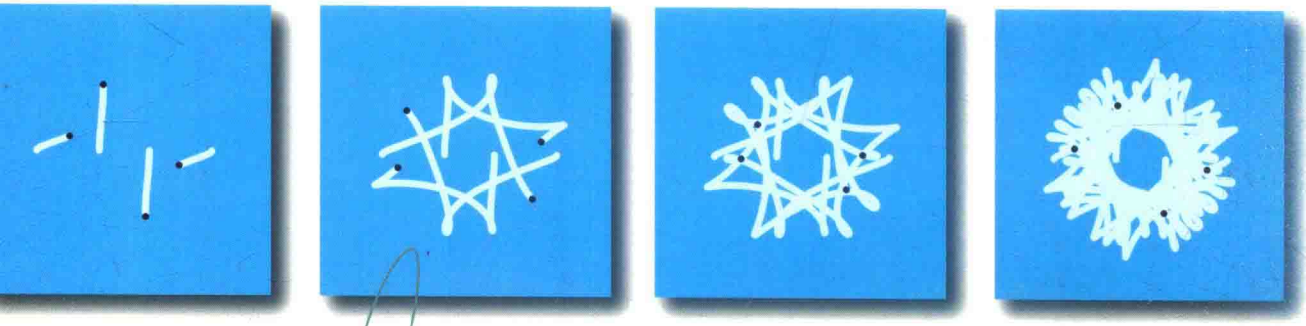
INTRODUCTION TO

Programming in Java

SECOND EDITION



An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

Introduction to Programming in Java

An Interdisciplinary Approach

SECOND EDITION

Robert Sedgewick
Kevin Wayne

Princeton University

Addison-Wesley

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

Library of Congress Control Number: 2017934241

ISBN-13: 978-0-672-33784-0
ISBN-10: 0-672-33784-3

2 17

Introduction to Programming in Java

Second Edition

To Adam, Andrew, Brett, Robbie,
Henry, Iona, Peter, Rose,
and especially Linda

To Jackie, Alex, and Michael

Preface

THE BASIS FOR EDUCATION IN THE last millennium was "reading, writing, and arithmetic"; now it is reading, writing, and *computing*. Learning to program is an essential part of the education of every student in the sciences and engineering. Beyond direct applications, it is the first step in understanding the nature of computer science's undeniable impact on the modern world. This book aims to teach programming to those who need or want to learn it, in a scientific context.

Our primary goal is to *empower* students by supplying the experience and basic tools necessary to use computation effectively. Our approach is to teach students that composing a program is a natural, satisfying, and creative experience. We progressively introduce essential concepts, embrace classic applications from applied mathematics and the sciences to illustrate the concepts, and provide opportunities for students to write programs to solve engaging problems.

We use the Java programming language for all of the programs in this book—we refer to "Java" after "programming in the title to emphasize the idea that the book is about *fundamental concepts in programming*, not Java per se. This book teaches basic skills for computational problem solving that are applicable in many modern computing environments, and is a self-contained treatment intended for people with no previous experience in programming.

This book is an *interdisciplinary* approach to the traditional CS1 curriculum, in that we highlight the role of computing in other disciplines, from materials science to genomics to astrophysics to network systems. This approach emphasizes for students the essential idea that mathematics, science, engineering, and computing are intertwined in the modern world. While it is a CS1 textbook designed for any first-year college student, the book also can be used for self-study or as a supplement in a course that integrates programming with another field.

Coverage The book is organized around four stages of learning to program: basic elements, functions, object-oriented programming, and algorithms (with data structures). We provide the basic information readers need to build confidence in their ability to compose programs at each level before moving to the next level. An essential feature of our approach is the use of example programs that solve intriguing problems, supported with exercises ranging from self-study drills to challenging problems that call for creative solutions.

Basic elements include variables, assignment statements, built-in types of data, flow of control, arrays, and input/output, including graphics and sound.

Functions and modules are the student's first exposure to modular programming. We build upon familiarity with mathematical functions to introduce Java functions, and then consider the implications of programming with functions, including libraries of functions and recursion. We stress the fundamental idea of dividing a program into components that can be independently debugged, maintained, and reused.

Object-oriented programming is our introduction to data abstraction. We emphasize the concepts of a data type and their implementation using Java's class mechanism. We teach students how to *use*, *create*, and *design* data types. Modularity, encapsulation, and other modern programming paradigms are the central concepts of this stage.

Algorithms and data structures combine these modern programming paradigms with classic methods of organizing and processing data that remain effective for modern applications. We provide an introduction to classical algorithms for sorting and searching as well as fundamental data structures and their application, emphasizing the use of the scientific method to understand performance characteristics of implementations.

Applications in science and engineering are a key feature of the text. We motivate each programming concept that we address by examining its impact on specific applications. We draw examples from applied mathematics, the physical and biological sciences, and computer science itself, and include simulation of physical systems, numerical methods, data visualization, sound synthesis, image processing, financial simulation, and information technology. Specific examples include a treatment in the first chapter of Markov chains for web page ranks and case studies that address the percolation problem, n-body simulation, and the small-world phenomenon. These applications are an integral part of the text. They engage students in the material, illustrate the importance of the programming concepts, and provide persuasive evidence of the critical role played by computation in modern science and engineering.

Our primary goal is to teach the specific mechanisms and skills that are needed to develop effective solutions to any programming problem. We work with complete Java programs and encourage readers to use them. We focus on programming by individuals, not programming in the large.

Related texts This book is the second edition of our 2008 text that incorporates hundreds of improvements discovered during another decade of teaching the material, including, for example, a new treatment of hashing algorithms.

The four chapters in this book are identical to the first four chapters of our text *Computer Science: An Interdisciplinary Approach*. That book is a full introductory course on computer science that contains additional chapters on the theory of computing, machine-language programming, and machine architecture. We have published this book separately to meet the needs of people who are interested only in the Java programming content. We also have published a version of this book that is based on Python programming.

The chapters in this volume are suitable preparation for our book *Algorithms, Fourth Edition*, which is a thorough treatment of the most important algorithms in use today.

Use in the curriculum This book is suitable for a first-year college course aimed at teaching novices to program in the context of scientific applications. Taught from this book, any college student will learn to program in a familiar context. Students completing a course based on this book will be well prepared to apply their skills in later courses in their chosen major and to recognize when further education in computer science might be beneficial.

Instructors interested in a full-year course (or a fast-paced one-semester course with broader coverage) should instead consider adopting *Computer Science: An Interdisciplinary Approach*.

Prospective computer science majors, in particular, can benefit from learning to program in the context of scientific applications. A computer scientist needs the same basic background in the scientific method and the same exposure to the role of computation in science as does a biologist, an engineer, or a physicist.

Indeed, our interdisciplinary approach enables colleges and universities to teach prospective computer science majors and prospective majors in other fields in the *same* course. We cover the material prescribed by CS1, but our focus on applications brings life to the concepts and motivates students to learn them. Our interdisciplinary approach exposes students to problems in many different disciplines, helping them to choose a major more wisely.

Whatever the specific mechanism, the use of this book is best positioned early in the curriculum. This positioning allows us to leverage familiar material in high school mathematics and science. Moreover, students who learn to program early in their college curriculum will then be able to use computers more effectively when moving on to courses in their specialty. Like reading and writing, programming is certain to be an essential skill for any scientist or engineer. Students who have grasped the concepts in this book will continually develop that skill through a lifetime, reaping the benefits of exploiting computation to solve or to better understand the problems and projects that arise in their chosen field.

Prerequisites This book is suitable for typical first-year college students. In other words, we do not expect preparation beyond what is typically required for other entry-level science and mathematics courses.

Mathematical maturity is important. While we do not dwell on mathematical material, we do refer to the mathematics curriculum that students have taken in high school, including algebra, geometry, and trigonometry. Most students in our target audience automatically meet these requirements. Indeed, we take advantage of familiarity with this curriculum to introduce basic programming concepts.

Scientific curiosity is also an essential ingredient. Science and engineering students bring with them a sense of fascination with the ability of scientific inquiry to help explain what occurs in nature. We leverage this predilection with examples of simple programs that speak volumes about the natural world. We do not assume any specific knowledge beyond that provided by typical high school courses in mathematics, physics, biology, or chemistry.

Programming experience is not necessary, but also is not harmful. Teaching programming is our primary goal, so we assume no prior programming experience. Nevertheless, composing a program to solve a new problem is a challenging intellectual task, so students who have written numerous programs in high school can benefit from taking an introductory programming course based on this book. The book can support teaching students with varying backgrounds because the applications appeal to both novices and experts alike.

Experience using a computer is not necessary, but also is not at all a problem. College students use computers regularly—to communicate with friends and relatives, to listen to music, to process photos, and as part of many other activities. The realization that they can harness the power of their own computer in interesting and important ways is an exciting and lasting lesson.

Goals We cover the CS1 curriculum, but anyone who has taught an introductory programming course knows that expectations of instructors in later courses are typically high: Each instructor expects all students to be familiar with the computing environment and approach that he or she wants to use. For example, a physics professor might expect students to design a program over the weekend to run a simulation; a biology professor might expect students to be able to analyze genomes; or a computer science professor might expect knowledge of the details of a particular programming environment. Is it realistic to meet such diverse expectations? Is it realistic to offer a single introductory CS course for all students, as opposed to a different introductory course for each set of students?

With this book, and decades of experience at Princeton and other institutions that have adopted earlier versions, we answer these questions with a resounding *yes*. The most important reason to do so is that this approach encourages diversity. By keeping interesting applications at the forefront, we can keep advanced students engaged, and by avoiding classifying students at the beginning, we can ensure that every student who successfully masters this material is prepared for further study.

What can *teachers* of upper-level college courses expect of students who have completed a course based on this book?

This is a common introductory treatment of programming, which is analogous to commonly accepted introductory courses in mathematics, physics, biology, economics, or chemistry. *An Introduction to Programming in Java* strives to provide the basic preparation needed by all college students, while sending the clear message that there is much more to understand about computer science than just programming. Instructors teaching students who have studied from this book can expect that they will have the knowledge and experience necessary to enable them to effectively exploit computers in diverse applications.

What can *students* who have completed a course based on this book expect to accomplish in later courses?

Our message is that programming is not difficult to learn and that harnessing the power of the computer is rewarding. Students who master the material in this book are prepared to address computational challenges wherever they might appear later in their careers. They learn that modern programming environments, such as the one provided by Java, help open the door to any computational problem they might encounter later, and they gain the confidence to learn, evaluate, and use other computational tools. Students interested in computer science will be well prepared to pursue that interest; students in other fields will be ready to integrate computation into their studies.

Online lectures A complete set of studio-produced videos that can be used in conjunction with this text is available at

```
http://www.informit.com/title/9780134493831
```

As with traditional live lectures, the purpose is to *inform* and *inspire*, motivating students to study and learn from the text. Our experience is that student engagement with such online material is significantly better than with live lectures because of the ability to play the lectures at a chosen speed and to replay and review the lectures at any time.

Booksite An extensive body of other information that supplements this text may be found on the web at

```
http://introcs.cs.princeton.edu/java
```

For economy, we refer to this site as the *booksite* throughout. It contains material for instructors, students, and casual readers of the book. We briefly describe this material here, though, as all web users know, it is best surveyed by browsing. With a few exceptions to support testing, the material is all publicly available.

The booksite contains a condensed version of the text narrative for reference while online, hundreds of exercises and programming problems (some with solutions), hundreds of easily downloadable Java programs, real-world data sets, and our I/O libraries for processing text, graphics, and sound. It is the web presence associated with the book and is a living document that is accessed millions of times per year. It is an essential resource for everyone who owns this book and is critical to our goal of making computer science an integral component of the education of all college students.

One of the most important implications of the booksite is that it empowers teachers and students to use their own computers to teach and learn the material. Anyone with a computer and a browser can begin learning to program by following a few instructions on the booksite. The process is no more difficult than downloading a media player or a song.

For *teachers*, the booksite contains resources for teaching that (together with the book and the studio-produced videos) are sufficiently flexible to support many of the models for teaching that are emerging as teachers embrace technology in the 21st century. For example, at Princeton, our teaching style was for many years based on offering two lectures per week to a large audience, supplemented by two class sessions per week where students meet in small groups with instructors or teaching

assistants. More recently, we have moved to a model where students watch lectures online and we hold *class meetings* once a week in addition to the two class sessions. Other teachers may work completely online. Still others may use a "flipped" model involving enrichment of the lectures after students watch them.

For *students*, the booksite contains quick access to much of the material in the book, including source code, plus extra material to encourage self-learning. Solutions are provided for many of the book's exercises, including complete program code and test data. There is a wealth of information associated with programming assignments, including suggested approaches, checklists, FAQs, and test data.

For *casual readers*, the booksite is a resource for accessing all manner of extra information associated with the book's content. All of the booksite content provides web links and other routes to pursue more information about the topic under consideration. There is far more information accessible than any individual could fully digest, but our goal is to provide enough to whet any reader's appetite for more information about the book's content.

Acknowledgments This project has been under development since 1992, so far too many people have contributed to its success for us to acknowledge them all here. Special thanks are due to Anne Rogers, for helping to start the ball rolling; to Dave Hanson, Andrew Appel, and Chris van Wyk, for their patience in explaining data abstraction; and to Lisa Worthington and Donna Gabai, for being the first to truly relish the challenge of teaching this material to first-year students. We also gratefully acknowledge the efforts of `/dev/126` ; the faculty, graduate students, and teaching staff who have dedicated themselves to teaching this material over the past 25 years here at Princeton University; and the thousands of undergraduates who have dedicated themselves to learning it.

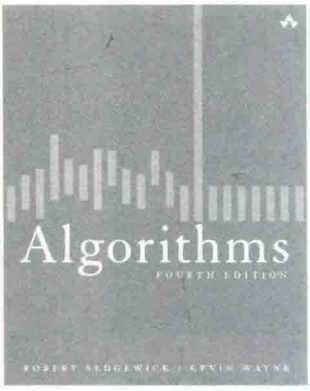
Robert Sedgewick
Kevin Wayne

February 2017

Chapter One

Also from Addison-Wesley

informit.com/sedgewick



992 pages • 972 exercises
152 programs • 350 figures
ISBN-13: 978-0-321-57351-3

The definitive guide to algorithms

- Essential information about algorithms and data structures
- A classic text, thoroughly updated
- Real-world examples throughout
- An indispensable body of knowledge for solving large problems by computer

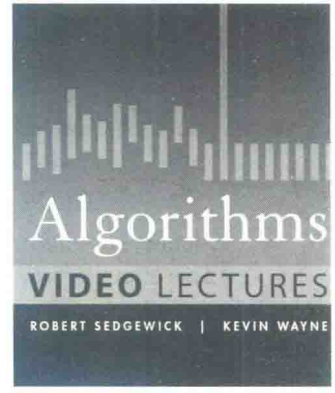
Algorithms

Fourth Edition

Robert Sedgewick & Kevin Wayne
Princeton University

Also available: Companion video lectures

- Studio-produced
- Fully coordinated with textbook content
- Ideal for flipped classrooms and online learning



24 lectures • 24+ hours
ISBN-13: 978-0-13-438443-6

Pearson

Contents

Programs

Elements of Programming

Functions and Modules