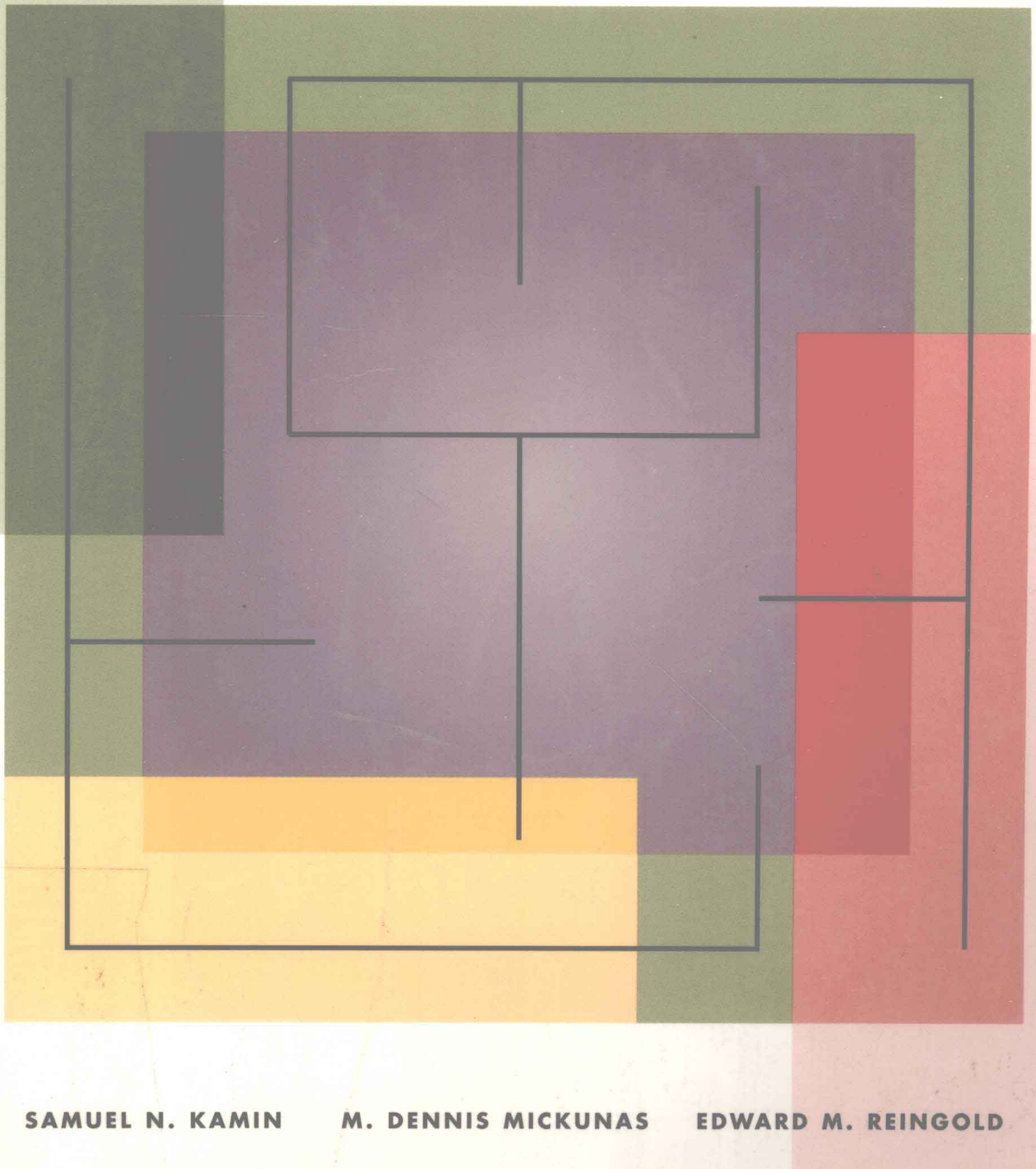
AN INTRODUCTION TO COMPUTER SCIENCE

Using Java™



SAMUEL N. KAMIN M. DENNIS MICKUNAS EDWARD M. REINGOLD

An Introduction to Computer Science Using Java

Samuel N. Kamin
University of Illinois at Urbana–Champaign

M. Dennis Mickunas
University of Illinois at Urbana–Champaign

Edward M. Reingold
University of Illinois at Urbana–Champaign

WCB
McGraw-Hill

Boston Burr Ridge, IL Dubuque, IA Madison, WI New York San Francisco St. Louis
Bangkok Bogotá Caracas Lisbon London Madrid
Mexico City Milan New Delhi Seoul Singapore Sydney Taipei Toronto

WCB/McGraw-Hill

A Division of The ***McGraw-Hill*** *Companies*

AN INTRODUCTION TO COMPUTER SCIENCE USING JAVA®

This book is printed on acid-free paper.

2 3 4 5 6 7 8 9 0 DOC/DOC 9 0 9 8

ISBN 0-07-034224-5

Vice president and editorial director: Kevin Kane
Publisher: Tom Casson
Executive editor: Elizabeth A. Jones
Developmental editor: Bradley Kosirog
Editorial assistant: Emily J. Gray
Marketing manager: John T. Wannemacher
Senior project manager: Beth Cigler
Production supervisor: Scott Hamilton
Senior designer: Crispin Prebys
Cover design: Z Graphics
Typeface: 10/12 Times Roman
Printer: R. R. Donnelley & Sons Company

Library of Congress Cataloging-in-Publication Data

Kamin, Samuel N.
An introduction to computer science using Java / Samuel N. Kamin, M. Dennis Mickunas, Edward M. Reingold.
p. cm.
Includes index.
ISBN 0-07-034224-5
1. Java (Computer program language) 2. Computer science.
I. Mickunas, M. Dennis. II. Reingold, Edward M., 1954- .
III. Title.
QA76.73.J38K36 1998
005.13'3--dc21 97-36768

http://www.mhhe.com

An Introduction to Computer Science Using Java

McGRAW-HILL SERIES IN COMPUTER SCIENCE

SENIOR CONSULTING EDITOR

C. L. Liu, *University of Illinois at Urbana-Champaign*

CONSULTING EDITOR

Allen B. Tucker, *Bowdoin College*

Fundamentals of Computing and Programming

Computer Organization and Architecture

Computers in Society/Ethics

Systems and Languages

Theoretical Foundations

Software Engineering and Database

Artificial Intelligence

Networks, Parallel and Distributed Computing

Graphics and Visualization

The MIT Electrical and Computer Science Series

Dedicated, with love, to our mothers

Sylvia H. Klersfeld, mother of S.N.K.
Ruth D. Simon ז"ל, mother-in-law of S.N.K.

Norma D. Mickunas, mother of M.D.M.
Frieda G. Foster, mother-in-law of M.D.M.

Leah J. Reingold ז"ל, mother of E.M.R.
Badonna L. Reingold, step-mother of E.M.R.
Charlotte B. Nothmann, mother-in-law of E.M.R.

רב יוסף כי הוה שמע קל כרעא דאמיה אמר איקום מקמי שכינה דאתיא קדושין לא:

When Rabbi Yosef heard his mother's footsteps he said, "Let me stand up, for the Divine Presence is approaching." *Talmud, Kiddushin* 31b

Preface

> Read not to contradict and confute, nor to believe and take for granted, nor to find talk and discourse, but to weigh and consider.
>
> —from *Essays, 50. Of Studies*,
> Francis Bacon

This book is an introduction to the principles and techniques of computer science, using the Java programming language as the medium of instruction.

Java, of course, is the language used in many World Wide Web browsers to create "active" home pages. With it, almost unlimited effects can be realized. The reader completing this book will be capable of creating web pages with very complex behavior.

Let us be clear on one point, however: this book promises no "fabulous applets in just five minutes a day." Programming in Java, as in any programming language, is challenging intellectual work. Our goals are those of any introductory computer science book: to give the reader the tools to develop correct, efficient, well-structured, and stylish programs and to build a foundation for further studies in computer science (CS).

Why, then, use Java? There are excellent pedagogical and practical reasons for using Java as an introductory programming language:

Java is clean. Java is an elegant object-oriented language, with run-time error checking, built-in garbage collection, an exception-handling mechanism—in short, many features ideally suited to introduce programming concepts without the pitfalls of other languages.

Java is fun. Java is equipped with a standard library of routines for creating graphics, playing audio files, and so on. Although the *process* of programming in Java is not essentially different from that of programming in any other language, the *results* can be a lot more interesting. One easily can imagine holding applet contests in freshman programming courses!

Java is available. We have done all our programming in Sun's Java Development Kit 1.1, which runs on all major platforms, with nearly flawless portability among them, and it is free! Students can work on their own computers, and instructors don't have to worry about incompatibilities introduced by new lab equipment. (See the "Web Pages" section for Sun's URL.)

Java compilers are user-friendly. Well, at least more user-friendly than many. Above all, because Java is interpreted, run-time errors are announced not with cryptic messages like "segmentation fault"—typical of C and C++ compilers—but instead with messages stating on what source line the error occurred. (And, if anything, other Java processors are likely to be even more friendly than the JDK.)

Java is simplified C++. In many CS curricula, upper-level courses are taught using C++, the most widely used "object-oriented" language. However, in the view of many instructors, C++ is too complex to be an appropriate introductory programming language. Java can provide a gentler introduction to the major concepts and syntax of C++, after which a brief period of migration to C++ will suffice.

The Structure of This Book

Our approach to presenting Java programming concepts represents a compromise among competing requirements. "Object oriented" is a description not only of a set of language features but of a programming philosophy, and it is important to stress that philosophy from the start; on the other hand, a good deal of machinery is needed before one can really *do* anything, object-orientedly or otherwise, and philosophy without examples is empty. Arrays perhaps are the most important "advanced" topic for many instructors, so it must be given due emphasis. Applets are fun and can provide great motivation for students, but they require knowledge of quite a lot of conceptually uninteresting detail; some instructors will prefer to postpone them to late in the semester or to another course. These are a few of the competing interests we have attempted to balance.

Concerning applets, we have chosen what we hope many will find to be an agreeable approach. Although coverage of applets is included in every chapter, it is separated from the coverage of nonapplets ("applications," in Java parlance). Chapter 3 is devoted exclusively to applets (plus a simple introduction to HTML). After that, every chapter except the last covers applets in its final section and only there. These applet sections do double duty, reinforcing the material covered in the chapter and introducing new features of Java's application programming interface (API). The last chapter goes through the development of a single applet, a Reversi-playing program. The basic programming concepts always are presented in the context of (text-oriented) applications, so the instructor can ignore applets entirely; aside from some applet-oriented exercises, which are clearly marked, there is no mention of applets outside the applet sections. (Also, to ease the development of simple applications, we have finessed Java's arcane input mechanism by defining a `Keyboard` class for reading simple integers, decimals, and strings.)

The coverage of programming concepts places a strong emphasis on object-oriented programming. In Java, unlike C++, it is impossible to avoid it, even temporarily. All code is contained in classes and many services can be obtained only by "sending messages." The concepts of object-oriented programming are first introduced in Chapter 1. A general overview of Java—enough to write very simple programs—is the topic of Chapter 2. Chapter 3, as mentioned already, presents some simple HTML and enough of the Java API for students to write simple applets of the "hello, world" variety; no new language features are covered there.

Chapter 4 is a conventional treatment of conditionals. The applet section expands on the treatment of events, providing the ability to respond to events such as button clicks.

Chapter 5 introduces object-oriented programming in earnest, with detailed treatment of methods and classes. We concentrate on instance methods here, as these are more directly tied into the object-oriented approach than class methods and the distinction can be difficult to understand.

Subsequent chapters cover iteration (Chapter 6), one-dimensional arrays (Chapter 7), class variables and methods (Chapter 8), two-dimensional arrays (Chapter 9), and strings and input/output (Chapter 10). The applet sections introduce more event-processing methods, as well as simple animation and database searching.

Chapter 11 covers recursion, including the presentation of quicksort, merge sort, and in the applets section, "fractal" curves (the Sierpiński and Hilbert curves). Chapter 12 presents some of the more advanced features of Java: exceptions, inheritance, interfaces, and abstract classes—all are mentioned in earlier chapters (in the applet sections they cannot be avoided) but this is their first full treatment. In the end, nearly all the Java language is covered, together with the highlights of the API. Chapter 13 shows the development of a large applet, pulling together much of what has been covered in the book.

How to Use This Book

Most instructors will consider the nonapplet material in Chapters 1–9 to be fundamental. This material is presented in nearly traditional fashion but for the predominance of object-oriented concepts, as described earlier. Chapters 10–12 cover more advanced material, which many instructors may feel they can do without; and the large example developed in Chapter 13 is decidedly optional. Chapters 10–13 are entirely independent of one another and can be covered or not, in any order.

The main decisions for the instructor are how to incorporate applets into his or her course and how much of the advanced material to cover. The book allows for a variety of approaches:

- An *integrated applications and applets course* will follow the book as written, going as far as time and the instructor's interests allow. The emphasis on applets can be increased by assigning the applet-oriented exercises in the nonapplet sections. Many instructors will want to conclude the course with an applet contest, perhaps with students working in groups.
- An *applications-only course* will omit Chapters 3 and 12 and all of the applet sections. As we said earlier, one can cover this material and remain nearly unaware of the existence of applets. This approach has the advantage that it permits the treatment of more programming concepts in the available time, without the distraction of having to learn the details of Java's API.
- A likely compromise is an *applications followed by applets* approach. The instructor might cover the nonapplet material in, say, Chapters 1–8 (skipping Chapter 3), then go back and pick up the applets material before continuing with Chapter 9 and beyond. An advantage of this approach is that students will have some programming experience before starting to program applets, which will make it easier for them to absorb the API details and, perhaps, save time in the long run.

Concerning the order of the chapters, the coverage is progressive, so that Chapters 1, 2, 4, ..., 9 must be covered in order. Chapter 3 is necessary only if applets are to be covered. Chapter 10, "Strings, Characters, and File I/O," does not depend on any material presented after Chapter 6 (although we would recommend that Chapter 7 precede it), so some instructors will want to present this material earlier. Chapters 11–13 depend on the material in Chapters 1–9, but are completely independent of one another, so any or all of them can be covered after Chapter 9.

Pedagogical Features

We have included in each chapter a number of features intended to enhance the student's understanding. Of course, numerous exercises from a wide variety of fields are included. Each chapter includes a summary reiterating the major concepts in that chapter and listing the new keywords and API methods introduced there. Frequent "Bug Alert" boxes warn the reader of common mistakes and misconceptions. Many early chapters contain a "debugging section," in which we follow the development of a program from initial specification, through syntax errors and logic errors, to a complete solution. These sections allow us to present debugging techniques, illustrate mistakes often made in using the features introduced in that chapter, and perhaps most important, show the readers that they are not alone in making mistakes—even stupid ones—when programming.

Some of the exercises (about a third of them) are labeled special in some way. There are three types of labels. A hand (☞) indicates an exercise we think every student should do; solutions to these exercises are provided at the back of the book. (Solutions to the remaining exercises can be found in the *Solutions Manual*, available to teachers from the publisher without cost.) A star

(★) indicates that an exercise is unusually difficult or extensive. A picture of a mouse (🖱) indicates that the exercise involves applets; we have been sure to include an adequate number of nonapplet exercises in the application sections, so that an instructor is not obliged to cover applets.

The outside margins contain keywords and phrases from the nearby text. The outside margins are visible on both left- and right-hand pages as one flips through the text, making it easy to locate a particular discussion.

The inside margins are used occasionally to display a "curvy road" sign. Such a sign warns the reader that the nearby material is more subtle or difficult than other material and deserves special attention. More than one curvy sign warns that more care should be taken in reading the material. Naturally, we have tried our best to smooth and widen all the roads, but some curvy ones are unavoidable.

Which Version of Java?

The Java language is quite stable and has been for some time. However, the API, especially the part relating to applets, changed quite significantly early in 1997. In particular, the treatment of "events," which are used in all but the very simplest applets, is quite different in JDK 1.1 from what it was in JDK 1.0. We have used JDK 1.1 exclusively and have made no attempt to accommodate both versions; anyone wishing to use this book to learn to program applets must obtain JDK 1.1. As of this writing (Fall 1997), the latter is available (see `http://java.sun.com`) on all major platforms except MacOS, and should be available for Macs soon.

Errata

In the introduction to his *Guide to the Perplexed*, the great 12th-century philosopher, physician, and rabbinic commentator Moses Maimonides outlines seven categories of contradiction or error to be found in books:

- The author quotes various sources that disagree.
- The author has changed his mind on a point but neglects to remove all the rejected material.
- Something is not to be taken literally but has inner content.
- An apparent (but not real) contradiction stems from the necessity to explain one thing before another.
- A simplification is made for purposes of explanation but later the point is explained in full.
- A contradiction escapes the author.
- The author is intentionally concealing something.

Maimonides avers that all the errors in his *Guide to the Perplexed* are of the fifth and seventh types. Would that the present authors could make such a claim!

There undoubtedly are errors of substance, style, spelling, and grammar in this book, try mightily as we did to prevent and eliminate them. All the programs and segments of programs were compiled and thoroughly tested before inclusion in the text.

If you should happen to notice an error, please bring it to our attention. We can be reached at the e-mail addresses

```
kamin@cs.uiuc.edu
mickunas@cs.uiuc.edu
reingold@cs.uiuc.edu
```

Of course, we can be contacted by snail-mail at

Department of Computer Science
University of Illinois at Urbana—Champaign
1304 West Springfield Avenue
Urbana, IL 61801-2987

Web Pages

We have established a home page for this book on the World Wide Web:

```
http://www.mhhe.com/engcs/compsci/kamin
```

This home page gives easy access to the programs included in the book. All of the major pieces of Java code in our presentation are available there; erroneous or bad-example code is not available nor are some of the one- or two-line examples. You also can view the applets that are presented in the book, as well as some of those that are assigned as exercises (for which we don't provide the source on-line, naturally).

The examples are written using Sun Microsystem's Java Development Kit, although naturally only a very small part of the presentation depends on this. It is important, though, that you have the correct version, which is JDK 1.1; the earlier version, 1.0, was widely disseminated in web browsers and still exists. In fact, the latest version of Netscape Navigator incorporates version 1.0, so that almost none of our applet examples can be run in that browser; however, by the time you read this, the newer Netscape Communicator, using JDK version 1.1, will be available.

In any case, your best bet is to obtain an implementation of JDK 1.1 and use the `appletviewer` program to develop applets. Using `appletviewer` has the added advantage that you can write output to the standard output stream (normally, the command window from which the `appletviewer` was invoked), an invaluable aid in debugging. JDK 1.1 can be obtained from the web page

```
http://java.sun.com
```

Click on the "download" button. That page also contains links to other Java implementations, including both alternative sources of implementations for the major platforms and implementations for platforms not supported by Sun.

Acknowledgments

Our Sponsoring Editor during this project was Betsy Jones. Betsy, together with our Developmental Editor, Brad Kosirog, diligently shepherded us through our last year of effort. They were ably assisted by Emily Gray. Beth Cigler (Senior Project Manager) efficiently handled copyediting, composition, and proofreading.

We prepared the original manuscript using LaTeX, drawing many of the figures with the powerful `pstricks` macros written by Timothy Van Zandt of Princeton University.

We were fortunate to get feedback from a number of highly qualified and perceptive outside reviewers. Although we may not always have agreed with—or even enjoyed seeing—their comments, we always found them thoughtful and thought provoking. Many thanks to

Ann Ford, University of Michigan
Ephraim Glinert, Rensselaer Polytechnic Institute
Michael T. Goodrich, Johns Hopkins University
William Hankley, Kansas State University
Lily Hou, Carnegie Mellon University
Dale Johnson, Gadsen State Community College
Michael Johnson, Carnegie Mellon University
Brian Malloy, Clemson University
David Poplawski, Michigan Technological University
Brent Seales, University of Kentucky
Stephen Slade, Yale University
Don Smith, Rutgers University
Lou Steinberg, Rutgers University
David Teague, Western Carolina University
Dawn Wilkins, University of Mississippi

Finally, our wives and daughters bore the brunt of this effort as much as we did ourselves. Our thanks and love, now and always, to them.

S.N.K.
M.D.M.
E.M.R.

Contents

1 WHAT IS PROGRAMMING?

2 BASIC ELEMENTS OF JAVA

3 APPLETS

4 DECISION MAKING

5 CLASSES AND METHODS I

6 ITERATION

7 ONE-DIMENSIONAL ARRAYS

8 CLASSES AND METHODS II

9 NESTED LOOPS AND TWO-DIMENSIONAL ARRAYS