

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

In the preface to the first edition of this book, we explained why we thought Java was a good choice for a language with which to teach programming. We argued that Java was a clean and simple object-oriented language; that it would be fun for students to learn because of the ability to program graphics and post programs on their websites; that it was freely available for all major platforms; that its compilers would be friendlier to beginners than the compilers for other languages such as C and C++; and that, by virtue of its similarities to C++, it would make a good jumping off point for learning that very popular, but complex, language.

These arguments have become commonly accepted truisms. The questions about introductory programming have turned to two primary issues: how, and how much, to cover object-oriented programming; and how to cover the fun parts of Java, such as applets, without drowning in details. The goal of an introductory computer science book, after all, is not to teach Java or C++ or any other specific language. It is, as we said in the first edition, “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.”

In considering the revision of the text, we came to two conclusions concerning those questions: that object-oriented programming is an important part of the foundation of computer science; and that there is a way to cover the fun stuff without the details—indeed, in a way that *reinforces* the principles we are teaching, rather than distracting from them. Accordingly, we have substantially rewritten and reorganized the book. The treatment of object-oriented programming has been greatly expanded, and the use of graphics is now pervasive (in the first edition it was optional). Like some other authors, we have written a class library, `CSLib`, that permits the use of graphical elements in programs without the raft of details associated with applets.

What’s New in the Second Edition?

To be specific about the changes from the first edition:

- Object-oriented programming is used from the beginning of the book. The four “classes and methods” chapters—2, 5, 7, and 10—form the backbone of our coverage. `CSLib` allows us to use objects in our very first complete program, and then it allows us to provide the student with considerable practice *using* objects before delving into their design and implementation.
- We use class diagrams from the Unified Modeling Language (UML) to illustrate the relationships between classes. The UML is rapidly gaining widespread acceptance, and it has been adopted as a standard by the Object Management Group (OMG).
- Our examples and exercises involve writing *graphical applications* using `CSLib`. Compared to programming applets using the standard Abstract Windowing Toolkit (AWT) or Swing components, this involves much less “bureaucracy,” while allowing use of essentially the same graphical elements. Once all the requisite language elements have been presented, we show how to transform graphical applications into applets (Chapter 13).

Aside from these changes, much of our approach and style follow the first edition. In particular, since our goal is to introduce computer science, not Java programming, we place a heavy emphasis on the development of algorithms. That object-oriented programming is now considered fundamental does not mean that topics previously considered fundamental have lost any of their importance. Conditional execution, iteration, and recursion are still the basic control structures in computer programming; the design and analysis of efficient algorithms are hardly possible without using arrays; the classic sorting algorithms are still the best and simplest examples of algorithm development. We have not wanted to sacrifice these and other fundamental topics. It is no surprise, then, that the present book is 50% longer than its predecessor!

What This Book Covers

Most of the Java language is covered eventually, with more advanced features naturally receiving a less expansive treatment than essential concepts. All the concepts we would normally consider essential to an introductory programming course—using any language—are covered, of course. These include the basic object-oriented programming paradigm; conditional execution, iteration, and recursion; integer, double, String, character, and Boolean data types and associated operations; one- and two-dimensional arrays; and the best-known sorting algorithms. Specifically, the chapter-by-chapter coverage is as follows:

Chapter 1 introduces the broadest principles of computing, including such notations as *algorithm*, *object*, and *compiler*.

In **Chapter 2**, we begin to write Java programs, using the `CSLib` classes for input and output. These simple programs use no language features other than those needed for object creation and message sending. This chapter serves as an introduction to those concepts, as well as to Java syntax and the mechanics of compilation and execution of Java programs. An important `CSLib` class introduced in this chapter is `DrawingBox`, which includes graphical operations exactly like those used in applets. It is worth noting that using this class is of no greater conceptual difficulty than using text-oriented output would be—arguably less, in fact—and allows us to retain a purely object-oriented view of programming.

Chapter 3 presents the primitive data types `int`, `double`, and `char`, as well as `String`.

Chapter 4 is a fairly conventional treatment of conditionals, including `if` and `switch` statements, Boolean expressions, and the `boolean` data type.

Chapter 5, which is comparatively short, fills in a crucial gap in the previous coverage of classes and objects, by discussing the construction of classes with multiple methods. Up to this point, the student has *written* classes with a single method—in effect, the “main” method of the program—while *using* objects (such as `DrawingBoxes`) that respond to multiple methods.

Chapter 6 is devoted to the treatment of iteration, including `for`, `while` and `do-while` loops. At that point in the book, quite a few interesting programs can be written, using `CSLib`. (One might say that at this point the ability to do interesting graphics is determined mainly by the students’ *mathematical* knowledge.)

Chapter 7 presents for the first time the design and implementation of classes with constructors, instance variables, and multiple methods. By this time, the two earlier “classes and objects” chapters (2 and 5) and the numerous examples will have prepared students for their first construction of “real” classes.

Chapters 8 and 9 are fairly conventional (except, that is, for the use of graphic-oriented examples) introductions to one-dimensional and

two-dimensional arrays, respectively. (We have chosen not to cover higher-dimensional arrays.)

Chapter 10 completes the treatment of classes (short of inheritance) by introducing static (or “class”) variables and methods as well as interfaces (required for the next chapter).

Chapter 11 describes how to write programs that respond to mouse clicks. It is labeled “optional” because it concerns only the Java Application Programming Interface (API) and not general concepts of computer science. However, considering that this part of the API is extremely useful without being terribly complicated, we expect that most instructors will want to cover this chapter.

Chapter 12 concerns inheritance and exceptions. Opinions on the importance of these features at the introductory level vary, but the features are in any case prerequisites for the study of applets and other aspects of the Java API that are covered in subsequent chapters.

Chapter 13 introduces the Java AWT package in earnest, including components, events, and layout managers. We show how to transform the applications we have been writing into applets.

Chapter 14 covers what we consider to be among the most essential topics in computer science: recursion. As shown in the dependency chart in Figure 1, this chapter does not depend upon any advanced language or API features.

Chapter 15 introduces the part of the Java API that concerns file input and output. Although many instructors will want to cover this topic earlier, it is unfortunately rather complicated and makes heavy use of advanced language features such as inheritance and exceptions.

The final chapter, **Chapter 16**, gives our “valedictory,” in the form of a large applet that implements the game of Reversi. This program uses virtually every feature introduced in the book, introduces a few new ones (such as the `VECTOR` class), and implements a pretty good game-playing strategy.

How to Use This Book

A dependency chart of all the chapters in the book is shown in Figure 1. The optional chapters, 11 and 13, are shown as dashed ovals.

Numerous selections of chapters and orders of coverage are possible. One can use the book to teach a pure Java programming course by covering Chapters 1–8, 10–13, and 15; this omits material on, for example, two-dimensional arrays and recursion. On the other hand, a selection better suited to beginning computer science majors might be Chapters 1–10 and 14, which omits some Java arcana in favor of more traditional topics.

In this edition we have added several web chapters covering topics which most instructors will consider too advanced for an introductory class. These are

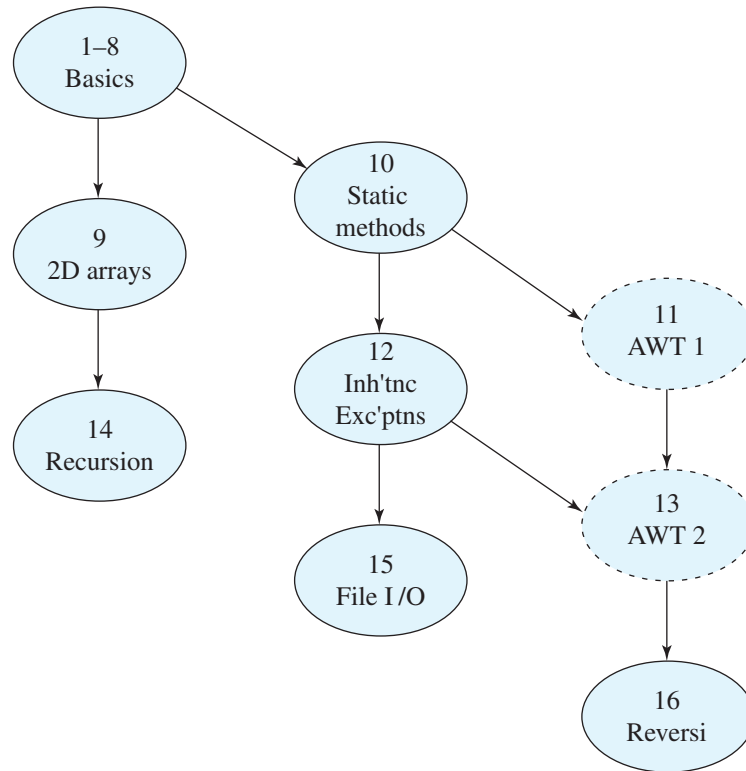


FIGURE 1 Chapter Dependency Diagram.

the Swing classes; threads and synchronization; and collection classes. These chapters are available on our website.

All of the important Java classes are covered to some extent in this book. Of course, the `java.awt` and `java.awt.event` classes are covered only in the AWT chapters (11 and 13). In particular, following are the classes that are covered in each of the indicated packages.

Classes in the `java.lang` package

Double	Integer	Math
Object	String	StringBuffer
System		

Classes in the `java.util` package

Calendar	Enumeration	Vector
----------	-------------	--------

Classes in the `java.awt` package

<code>BorderLayout</code>	<code>Button</code>	<code>Canvas</code>
<code>Checkbox</code>	<code>CheckboxGroup</code>	<code>Color</code>
<code>Component</code>	<code>Container</code>	<code>Dimension</code>
<code>FlowLayout</code>	<code>Font</code>	<code>FontMetrics</code>
<code>Graphics</code>	<code>GridLayout</code>	<code>Label</code>
<code>Panel</code>	<code>Point</code>	<code>Polygon</code>
<code>Scrollbar</code>	<code>TextArea</code>	<code>TextComponent</code>
<code>TextField</code>		

Classes in the `java.awt.event` package

<code>ActionListener</code>	<code>AdjustmentEvent</code>	<code>AdjustmentListener</code>
<code>EventObject</code>	<code>ItemEvent</code>	<code>ItemListener</code>
<code>MouseEvent</code>	<code>MouseListener</code>	<code>MouseMotionListener</code>
<code>TextListener</code>	<code>WindowListener</code>	

Classes in the `java.applet` package

<code>Applet</code>	<code>URL</code>
---------------------	------------------

Classes in the `java.io` package

<code>BufferedReader</code>	<code>BufferedWriter</code>	<code>FileReader</code>
<code>FileWriter</code>	<code>PrintWriter</code>	<code>Reader</code>
<code>Writer</code>		

Pedagogical Features

We have retained all of the pedagogical features from the first edition that were judged effective, we expanded some of them, and we introduced new features that were requested by instructors. The main features are

- In the first edition, we included in most chapters a “debugging section,” in which a complete program development was illustrated, warts and all. We have continued that policy in this book, as we have found that students find it useful and encouraging to see how even experienced programmers can struggle with errors, both silly and subtle.
- We also had occasional “bug alert” boxes, pointing out common pitfalls of a language feature or programming idiom being introduced; again, we have retained this feature.
- In this revision, however, we have changed somewhat our approach to exercises. Instead of placing exercises after each section, we have placed exercises only at the ends of chapters.
- However, we have added numerous “quick exercises” throughout the text. These are easy problems intended to reinforce the concepts being introduced; in many cases, their main purpose is to encourage readers to try out on the computer what they’ve learned from the text. In principle, readers should do—or at least completely understand—every quick exercise.



- 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.

Obtaining a Java Compiler

All our programmers are based on version 1.3 of the Java 2 system, available from Sun Microsystems (website: `java.sun.com`). The naming of the different versions has been a source of confusion. For the most part, our programs will run on any version as recent as 1.1 (the change from 1.0 to 1.1 having been the most significant as far as the essential elements of the Java language go).

Use of Integrated Development Environment (IDE) greatly simplifies the edit-compile-execute cycle. There are a number of commercial IDEs for Java, one of which—Code Warrior—is supplied with this book.

Feedback to the Authors

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:

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

If you should happen to notice an error, please bring it to our attention. We also welcome any ideas or feedback from our readers. We can be reached at the e-mail addresses

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

or by regular mail at:

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

Web Page

The web page for this book is

www.mhhe.com/engcs/compsci/kamin

There you will find

- Javadoc documentation for CSLib.
- Source code for the CSLib.
- Source code for all the examples given in the text.
- Answers to quick exercises.
- Runnable versions of solutions to the end-of-chapter programming exercises. (Source code is, of course, not included!)
- The “bonus” chapters covering the Swing classes; threads and synchronization; and collection classes.
- A list of errata.
- Links to the Java documentation and download pages at Sun Microsystems.
- Instructions on using the Code Warrior IDE.

To the Instructor

A separate password-protected instructor’s web page is also available. Please contact your McGraw-Hill Higher Education representative to obtain access to it. There you will find

- Solutions to all the end-of-chapter exercises.
- Chapter by chapter Powerpoint presentations.
- An alternate version of CSLib to assist with grading student assignments.

Acknowledgments

First Edition

Our Sponsoring Editor for the first edition 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 `ps`tricks 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 (Gadsden 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), and Dawn Wilkins (University of Mississippi).

We received feedback from many readers. Those who were the first to point out errors were Maryam Abbassian, Mark Blanchard, Lawrie Brown, Miranda Callahan, Chuck Ehlschlaeger, Yuval Feinstein, Eric Han, Bharat Khardia, William Knight, Jonathan Kozolchyk, Naomi Lindenstrauss, Jiten Patel, Neil Rhoads, Sam Rhoads, Nan Schaller, Arie Schlesinger, John A. Trono, William Voss, Ali Yazici, and Eliyahu Yona.

Second Edition

Our Sponsoring Editor for the second edition was again Betsy Jones, and Emily Lupash was our Developmental Editor. They were both very helpful with all the details involved, and also incredibly patient as deadlines approached (and sometimes passed). Susan Bruschi (Senior Project Manager) provided quick, efficient turnaround on copyediting, composition, and proofreading.

As with the first edition, we received a great deal of very perceptive feedback from outside reviewers. Their suggestions were always well-reasoned, and have resulted in a vastly improved second edition. Many thanks to:

Byron Weber Becker (University of Waterloo), Kenneth D. Blaha (Pacific Lutheran University), Jonathan E. Cook (New Mexico State University), James H. Cross (Auburn University), Walter C. Daugherty (Texas A & M University), Roger Ferguson (Grand Valley State University), Philip Gilbert (California State University, Northridge), Scott Grissom (Grand Valley State University), Joanne F. Houlahan (Johns Hopkins University), Eliot Jacobson (University of California, Santa Barbara), Stan Kwasny (Washington University), Alan Saleski

(Loyola University, Chicago), Dale Shaffer (Lander University), David Surma (Valparaiso University), Phil Ventura (State University of New York, Buffalo), Jennifer L. Welch (Texas A & M University), and Lan Yang (California Polytechnic University, Pomona).

As before, our families and friends endured cancelled plans, inattention, late nights, and crisis deadlines. We are truly indebted to them.

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