Welcome to the third edition of *Data Structures and Abstractions with Java*, a book for an introductory course in data structures, typically known as CS-2. Readers of my book *Imagine! Java* can consider this one as a sequel.

I wrote this book with you in mind—whether you are an instructor or a student—based upon my experiences during more than three decades of teaching undergraduate computer science. I wanted my book to be reader friendly so that students could learn more easily and instructors could teach more effectively. To this end, you will find the material covered in small pieces—I call them “segments”—that are easy to digest and facilitate learning. Numerous examples that mimic real-world situations provide a context for the new material and help to make it easier for students to learn and retain abstract concepts. Many simple figures illustrate and clarify complicated ideas. Included are over 60 video tutorials to supplement the instruction and help students when their instructor is unavailable.

I hope that you enjoy reading this book. Like many others before you, you can learn—or teach—data structures in an effective and sustainable way.

Warm regards,

Frank M. Carrano

P. S. I am always available to connect with instructors and students who use my books. Here are a few ways you can reach me:

Find me on Facebook:  www.facebook.com/makingitreal
Follow me on Twitter:  twitter.com/Frank_M_Carrano
Send me an e-mail:    carrano@acm.org
Post on my blog:       frank-m-carrano.com/makingitreal
The topics that we cover in this book deal with the various ways of organizing data so that a given application can access and manipulate data in an efficient way. These topics are fundamental to your future study of computer science, as they provide you with the foundation of knowledge required to create complex and reliable software. Whether you are interested in designing video games or software for robotic-controlled surgery, the study of data structures is vital to your success. Even if you do not study all of the topics in this book now, you are likely to encounter them later. I hope that you will enjoy reading the book, and that it will serve as a useful reference tool for your future courses.

After looking over this preface, you should read the Introduction. There you will quickly see what this book is about and what you need to know about Java before you begin. Appendices A through G review Java basics, classes, inheritance, exceptions, files, and javadoc comments. Note that inside the front and back covers you will find Java’s reserved words, its primitive data types, the precedence of its operators, and a list of Unicode characters.

Please be sure to browse the rest of this preface to see the features that will help you in your studies.
Organization and Structure

This book’s organization, sequencing, and pace of topic coverage make learning and teaching easier by focusing your attention on one concept at a time, by providing flexibility in the order in which you can cover topics, and by clearly distinguishing between the specification and implementation of abstract data types, or ADTs. To accomplish these goals, I have organized the material into 30 chapters, composed of small, numbered segments that deal with one concept at a time. Each chapter focuses on either the specification and use of an ADT or its various implementations. You can choose to cover the specification of an ADT followed by its implementations, or you can treat the specification and use of several ADTs before you consider any implementation issues. The book’s organization makes it easy for you to choose the topic order that you prefer.

Table of Contents at a Glance

The following list of chapter titles shows the overall composition of the book. A further chapter-by-chapter description appears later. Note that highlighted sections are available online.

**Introduction**

1. Bags
2. Bag Implementations That Use Arrays
3. A Bag Implementation That Links Data
4. The Efficiency of Algorithms
5. Stacks
6. Stack Implementations
7. Recursion
8. An Introduction to Sorting
9. Faster Sorting Methods
10. Queues, Deques, and Priority Queues
11. Queue, Deque, and Priority Queue Implementations
12. Lists
13. List Implementations That Use Arrays
14. A List Implementation That Links Data
15. Iterators
16. Sorted Lists
17. Inheritance and Lists
18. Searching
19. Dictionaries
20. Dictionary Implementations
21. Introducing Hashing
22. Hashing as a Dictionary Implementation
23. Trees
24. Tree Implementations
25. A Binary Search Tree Implementation
26. A Heap Implementation
27. Balanced Search Trees
28. Graphs
29. Graph Implementations
30. Mutable, Immutable, and Cloneable Objects

**Appendices**

A. Java Essentials
B. Java Classes
C. Creating Classes from Other Classes
D. Designing Classes
E. Handling Exceptions
F. File Input and Output
G. Documentation and Programming Style

**Glossary**

**Index**
What’s New?

Based on comments from readers and reviewers, I have reorganized some of the material. Many students are familiar with stacks and queues, and so the coverage of these data organizations is much earlier in this edition. Moreover, the reorganization makes the difficult topic of linked data more accessible to students. Adding or removing the first node in a chain of linked nodes is the easiest operation. By introducing the bag, the book uses these simple operations on a linked chain in the bag’s implementation. That data collection is followed by the stack, a more useful organization that has the same simple chain in one of its definitions. Queue implementations provide the opportunity to discuss adding and removing the last node in a chain. Finally, the treatment of lists looks at the more involved operations of adding and removing a node that lies between existing nodes.

You will notice that algorithm efficiency—including improved motivation—recursion, and sorting also are covered earlier in this edition than in the previous one. To maintain the focus on data structures, I have moved the first three chapters—Java Classes, Creating Classes from Other Classes, and Designing Classes—to the appendices. The presentation now moves from the introduction immediately to the first data collection, the bag. However, readers who need to study Java classes before embarking on the main topic of this book will find the original coverage intact in the appendices.

Finally, I have added some new features. Extensive examples are presented in the form of “A Problem Solved,” in which a problem is posed and its solution is discussed and implemented. An occasional “Design Decision” explores various design choices of a solution. These two new elements help students to think about important aspects of program design and to consider concepts in a situational context. Another new feature is the availability online of over 60 VideoNotes that provide additional instruction in a more dynamic form than a static textbook. The Notes, Programming Tips, and Questions—with answers—that were featured in the previous edition have been retained. And you will find an introduction to the interface Deque and the class ArrayDeque, as well as additional programming projects.

Here is a summary of what is new:

- Earlier introduction of abstract data types, resizable arrays, and linked data.
- More gradual coverage of linked data.
- Earlier coverage of algorithm efficiency, stacks, recursion, sorting, and queues.
- Better motivation of the need for algorithm efficiency.
- Chapters 1 through 3—Bags, Bag Implementations That Use Arrays, and A Bag Implementation That Links Data—introduce and implement the ADT bag.
- New elements, including A Problem Solved and Design Decision.
- Review of Java classes that appeared in the initial chapters of the second edition is now in the appendices.
- VideoNotes—short instructional tutorials—reinforce key concepts presented in the book.
- Coverage of the standard interface Deque and the class ArrayDeque.
- Additional Programming Projects.
- Answers to Self-Test Questions appear at the end of each chapter instead of an appendix.
Features to Enhance Learning

Each chapter begins with a table of contents, a list of prerequisite chapters or appendices that students should have read, and the learning objectives for the material to be covered. Other pedagogical elements appear throughout the book, as follows:

- **Notes**  Important ideas are presented or summarized in highlighted paragraphs and are meant to be read in line with the surrounding text.

- **Programming Tips**  Suggestions to improve or facilitate programming are featured as soon as they become relevant.

- **Examples**  Numerous examples illuminate new concepts.

- **A Problem Solved**  Large examples are presented in the form of “A Problem Solved,” in which a problem is posed and its solution is discussed, designed, and implemented.

- **Design Decisions**  To give readers insight into the design choices that one could make when formulating a solution, “Design Decision” elements lay out such options, along with the rationale behind the choice made for a particular example. These discussions are often in the context of one of the A Problem Solved examples.

- **Self-Test Questions**  Questions are posed throughout each chapter, integrated within the text, that reinforce the concept just presented. These “self-test” questions help readers to understand the material, since answering them requires pause and reflection. Solutions to these questions are provided at the end of each chapter.

- **VideoNotes**  Online tutorials are a Pearson feature that provides visual and audio support to the presentation given throughout the book. They offer students another way to recap and reinforce key concepts. VideoNotes allow for self-paced instruction with easy navigation, including the ability to select, play, rewind, fast-forward, and stop within each video. Unique VideoNote icons appear throughout this book whenever a video is available for a particular concept or problem. A detailed list of the VideoNotes for this text and their associated locations in the book can be found on page xxiv. VideoNotes are free with the purchase of a new textbook. To purchase access to VideoNotes, please go to pearsonhighered.com/carrano

**Exercises and Programming Projects**  Further practice is available by solving the exercises and programming projects at the end of each chapter. Unfortunately, we cannot give readers the answers to these exercises and programming projects, even if they are not enrolled in a class. Only instructors who adopt the book can receive selected answers from the publisher. For help with these exercises and projects, you will have to contact your instructor.
Accessing Instructor and Student Resource Materials

The following items are available on the publisher’s website at pearsonhighered.com/carrano:

- Java code as it appears in the book
- A link to any misprints that have been discovered since the book was published
- Links to additional online content, which is described next

**Instructor Resources**

The following protected material is available to instructors who adopt this book by logging onto Pearson’s Instructor Resource Center, accessible from pearsonhighered.com/carrano:

- PowerPoint lecture slides
- Instructor solutions manual
- Figures from the book

Additionally, instructors can access the book’s Companion Website for the following online premium content, also accessible from pearsonhighered.com/carrano:

- Instructional VideoNotes
- Chapter 30
- A glossary of terms
- Exercises and projects for Appendices B, C, and D

Please contact your Pearson sales representative for an instructor access code. Contact information is available at pearsonhighered.com/replocator.

**Student Resources**

The following material is available to students by logging onto the Companion Website accessible from pearsonhighered.com/carrano:

- Instructional VideoNotes
- Chapter 30
- A glossary of terms
- Exercises and projects for Appendices B, C, and D

Students must use the access card located in the front of the book to register for and then enter the Companion Website. Students without an access code can purchase access from the Companion Website by following the instructions listed there.

Note that the Java Class Library is available at download.oracle.com/javase/7/docs/api/.
Chapter Overview

Readers of this book should have completed a programming course, preferably in Java. The appendices cover the essentials of Java that we assume readers will know. You can use these appendices as a review or as the basis for making the transition to Java from another programming language. The book itself begins with the Introduction, which sets the stage for the data organizations that we will study.

- **Chapters 1 through 3:** We introduce the bag as an abstract data type (ADT). By dividing the material across several chapters, we clearly separate the specification, use, and implementation of the bag. For example, Chapter 1 specifies the bag and provides several examples of its use. Chapter 2 covers implementations that use arrays and vectors, while Chapter 3 introduces chains of linked nodes and uses one in the definition of a class of bags.

  In a similar fashion, we separate specification from implementation throughout the book when we discuss various other ADTs. You can choose to cover the chapters that specify and use the ADTs and then later cover the chapters that implement them. Or you can cover the chapters as they appear, implementing each ADT right after studying its specification and use. A list of chapter prerequisites appears later in this preface to help you plan your path through the book.

  Chapter 2 does more than simply implement the ADT bag. It shows how to approach the implementation of a class by initially focusing on core methods. When defining a class, it is often useful to implement and test these core methods first and to leave definitions of the other methods for later.

- **Chapter 4:** Here we introduce the complexity of algorithms, a topic that we integrate into future chapters.

- **Chapters 5 and 6:** Chapter 5 discusses stacks, giving examples of their use, and Chapter 6 implements the stack using an array, a vector, and a chain.

- **Chapters 7 through 9:** Next, we present recursion as a problem-solving tool and its relationship to stacks. Recursion, along with algorithm efficiency, is a topic that is revisited throughout the book. For example, Chapters 8 and 9 discuss various sorting techniques and their relative complexities. We consider both iterative and recursive versions of these algorithms.

- **Chapters 10 and 11:** Chapter 10 discusses queues, deques, and priority queues, and Chapter 11 considers their implementations. It is in this latter chapter that we introduce circularly linked and doubly linked chains.

- **Chapters 12, 13, and 14:** The next three chapters introduce the ADT list. We discuss this collection abstractly and then implement it by using an array, a vector, and a chain.

- **Chapter 15:** Next, we discuss iterators in the context of a list. This chapter considers and implements Java’s iterator interfaces `Iterator` and `ListIterator`. The chapter also introduces the interface `Iterable`.

- **Chapters 16 and 17:** Continuing the discussion of a list, Chapter 16 introduces the sorted list, looking at two possible implementations and their efficiencies. Chapter 17 shows how to use the list as a superclass for the sorted list and discusses the general design of a superclass.

- **Chapter 18:** We then examine some strategies for searching an array or a chain in the context of a list or a sorted list. This discussion is a good basis for the sequence of chapters that follows.

- **Chapters 19 through 22:** Chapter 19 covers the specification and use of the ADT dictionary. Chapter 20 presents implementations of the dictionary that are linked or that use arrays. Chapter 21 introduces hashing, and Chapter 22 uses it as a dictionary implementation.

- **Chapters 23 through 27:** Chapter 23 discusses trees and their possible uses. Included among the several examples of trees is an introduction to the binary search tree and the heap. Chapter 24 considers implementations of the binary tree and the general tree, and Chapter 25 focuses on the implementation of the binary search tree. Chapter 26 shows how to use an array to implement the heap. Chapter 27
introduces balanced search trees. Included in this chapter are the AVL, 2-3, 2-4, and red-black trees, as well as B-trees.

- **Chapters 28 and 29:** Next, we discuss graphs and look at several applications and two implementations.
- **Chapter 30:** The final chapter expands on the notion of mutable objects and immutable objects, and introduces cloning. If a client can maintain a reference to the data within an ADT, it can change that data without using the class’s public methods, if the data is mutable. We consider steps that you can take to prevent the client from doing so.
- **Appendices A through G:** The appendices provide supplemental coverage of Java. As we mentioned earlier, Appendix A reviews Java up to but not including classes. However, this appendix also covers the Scanner class, enumerations, boxing and unboxing, and the for-each loop. Appendix B discusses Java classes, Appendix C expands this topic by looking at composition and inheritance, and Appendix D focuses on class design. Appendix E covers exception handling, and Appendix F discusses files. Appendix G considers programming style and comments. It introduces javadoc comments and defines the tags that we use in this book.

Acknowledgments

My sincere appreciation and thanks go to the following reviewers for carefully reading the previous edition and making candid comments and suggestions that greatly improved the work.

Steven Andrianoff—St. Bonaventure University
Brent Baas—LeTourneau University
Timothy Henry—University of Rhode Island
Ken Martin—University of North Florida
Bill Siever—Northwest Missouri State University
Lydia Sinapova—Simpson College
Lubomir Stanchev—Indiana University
Judy Walters—North Central College
Xiaohui Yuan—University of North Texas

Special thanks go to my support team during the lengthy process of revising this book. My editor, Tracy Dunkelberger, gave me her constant enthusiasm, encouragement, wisdom, and guidance. Melinda Haggerty and Allison Michael coordinated the review process, and Stephanie Sellinger oversaw the development of the book and its supplements. My long-time copy editor, Rebecca Pepper, ensured that my presentation is clear, correct, and grammatical. Jeff Holcomb, Bob Engelhardt, and Louise Capulli directed the production of the book. This team was there for me whenever I needed them at a moment's notice. Thank you so much!

My gratitude for the previously mentioned people does not diminish my appreciation for the help provided by many others. Tim Henry, my colleague at URI, created over 60 VideoNotes that provide extra instruction for every chapter of the book. Steve Armstrong produced the lecture slides for this edition and previous editions of the book. Thank you again to the reviewers of the first two editions of the book:

- **Reviewers for the second edition:**
  Harold Anderson—Marist College
  Razvan Andonie—Central Washington University
  Tom Blough—Rensselaer Polytechnic Institute
Chris Brooks—University of San Francisco
Adrienne Decker—University at Buffalo, SUNY
Henry Etlinger—Rochester Institute of Technology
Derek Harter—Texas A&M University
Timothy Henry—University of Rhode Island
Robert Holloway—University of Wisconsin, Madison
Charles Hoot—Oklahoma City University
Teresa Leyk—Texas A&M University
Robert McGlinn—Southern Illinois University, Carbondale
Edward Medvid—Marymount University
Charles Metzler—City College of San Francisco
Daniel Zeng—University of Arizona

Reviewers for the first edition:
David Boyd—Valdosta State University
Dennis Brylow—Purdue University
Michael Croswell—Industry trainer/consultant
Matthew Dickerson—Middlebury College
Robert Holloway—University of Wisconsin, Madison
John Motil—California State University, Northridge
Bina Ramamurthy—University at Buffalo, SUNY
David Surma—Valparaiso University

I continue to appreciate the many others who helped during previous editions. They include Alan Apt, James Blanding, Lianne Dunn, Mike Giacobbe, Toni Holm, Charles Hoot, Brian Jepson, Rose Kernan, Christianna Lee, Patrick Lindner, John Lovell, Vince O’Brien, Patty Roy, Walt Savitch, Ben Schomp, Heather Scott, Carole Snyder, Chirag Thakkar, Camille Trentacoste, Nate Walker, and Xiaohong Zhu.

Finally, I thank my family and friends—Doug, Ted, Vandee, Nancy, Sue, Tom, Maybeth, Marge, and Lorraine—for giving me a life away from computers.

Thank you, everyone, for your expertise and good cheer.

Frank M. Carrano