

GLOBAL  
EDITION



# Data Structures and Abstractions with Java™

FOURTH EDITION

Frank M. Carrano • Timothy M. Henry

ALWAYS LEARNING

PEARSON

## ONLINE ACCESS

Thank you for purchasing a new copy of *Data Structures and Abstractions with Java<sup>TM</sup>, Fourth Edition, Global Edition*. Your textbook includes twelve months of prepaid access to the book's Premium Content. This prepaid subscription provides you with full access to the following student support areas:

- VideoNotes are step-by-step video tutorials specifically designed to enhance the programming concepts presented in this textbook
- Premium Web Chapters

Use a coin to scratch off the coating and reveal your student access code.  
Do not use a knife or other sharp object as it may damage the code.

To access the *Data Structures and Abstractions with Java<sup>TM</sup>, Fourth Edition, Global Edition*, Premium Content for the first time, you will need to register online using a computer with an Internet connection and a web browser. The process takes just a couple of minutes and only needs to be completed once.

1. Go to **[www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano)**
2. Click on **Companion Website**.
3. Click on the **Register** button.
4. On the registration page, enter your student access code\* found beneath the scratch-off panel. Do not type the dashes. You can use lower- or uppercase.
5. Follow the on-screen instructions. If you need help at any time during the online registration process, simply click the **Need Help?** icon.
6. Once your personal Login Name and Password are confirmed, you can begin using the *Data Structures and Abstractions with Java<sup>TM</sup>* Companion Website!

### To log in after you have registered:

You only need to register for this Companion Website once. After that, you can log in any time at **[www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano)** by providing your Login Name and Password when prompted.

\*Important: The access code can only be used once. This subscription is valid for twelve months upon activation and is not transferable.

# Data Structures and Abstractions with Java, Global Edition

## Table of Contents

Cover

Title Page

Copyright

Contents

Introduction: Organizing Data

Prelude: Designing Classes

- Encapsulation

- Specifying Methods

  - Comments

  - Preconditions and Postconditions

  - Assertions

- Java Interfaces

  - Writing an Interface

  - Implementing an Interface

  - An Interface as a Data Type

  - Extending an Interface

  - Named Constants Within an Interface

- Choosing Classes

  - Identifying Classes

  - CRC Cards

  - The Unified Modeling Language

- Reusing Classes

Chapter 1: Bags

- The Bag

  - A Bags Behaviors

- Specifying a Bag

# Table of Contents

An Interface

Using the ADT Bag

Using an ADT Is Like Using a Vending Machine

The ADT Set

Java Class Library: The Interface set

## Java Interlude 1: Generics

Generic Data Types

Generic Types Within an Interface

Generic Classes

## Chapter 2: Bag Implementations That Use Arrays

Using a Fixed-Size Array to Implement the ADT Bag

An Analogy

A Group of Core Methods

Implementing the Core Methods

Making the Implementation Secure

Testing the Core Methods

Implementing More Methods

Methods That Remove Entries

Using Array Resizing to Implement the ADT Bag

Resizing an Array

A New Implementation of a Bag

The Pros and Cons of Using an Array to Implement the ADT Bag

## Java Interlude 2: Exceptions

The Basics

Handling an Exception

Postpone Handling: The throws Clause

Handle It Now: The try-catch Blocks

Multiple catch Blocks

Throwing an Exception

## Chapter 3: A Bag Implementation That Links Data

Linked Data

Forming a Chain by Adding to Its Beginning

A Linked Implementation of the ADT Bag



# Table of Contents

- The Private Class Node
- An Outline of the Class LinkBag
- Defining Some Core Methods
- Testing the Core Methods
- The Method getFrequencyOf
- The Method contains

## Removing an Item from a Linked Chain

- The Methods remove and clear

## A Class Node That Has Set and Get Methods

The Pros and Cons of Using a Chain to Implement the ADT Bag

## Chapter 4: The Efficiency of Algorithms

### Motivation

### Measuring an Algorithms Efficiency

- Counting Basic Operations
- Best, Worst, and Average Cases

### Big Oh Notation

- The Complexities of Program Constructs

### Picturing Efficiency

### The Efficiency of Implementations of the ADT Bag

- An Array-Based Implementation
- A Linked Implementation
- Comparing the Implementations

## Chapter 5: Stacks

### Specifications of the ADT Stack

### Using a Stack to Process Algebraic Expressions

- A Problem Solved: Checking for Balanced Delimiters in an Infix Algebraic Expression
- A Problem Solved: Transforming an Infix Expression to a Postfix Expression
- A Problem Solved: Evaluating Postfix Expressions
- A Problem Solved: Evaluating Infix Expressions

### The Program Stack

Java Class Library: The Class Stack

## Chapter 6: Stack Implementations

- A Linked Implementation
- An Array-Based Implementation
- A Vector-Based Implementation

# Table of Contents

Java Class Library: The Class Vector

Using a Vector to Implement the ADT Stack

## Chapter 7: Recursion

What Is Recursion?

Tracing a Recursive Method

Recursive Methods That Return a Value

Recursively Processing an Array

Recursively Processing a Linked Chain

The Time Efficiency of Recursive Methods

The Time Efficiency of countDown

The Time Efficiency of Computing  $x^n$

A Simple Solution to a Difficult Problem

A Poor Solution to a Simple Problem

Tail Recursion

Indirect Recursion

Using a Stack Instead of Recursion

## Java Interlude 3: More About Generics

The Interface Comparable

Generic Methods

Bounded Type Parameters

Wildcards

Bounded Wildcards

## Chapter 8: An Introduction to Sorting

Organizing Java Methods That Sort an Array

Selection Sort

Iterative Selection Sort

Recursive Selection Sort

The Efficiency of Selection Sort

Insertion Sort

Iterative Insertion Sort

Recursive Insertion Sort

The Efficiency of Insertion Sort

Insertion Sort of a Chain of Linked Nodes

Shell Sort

# Table of Contents

The Algorithm

The Efficiency of Shell Sort

Comparing the Algorithms

## Chapter 9: Faster Sorting Methods

### Merge Sort

Merging Arrays

Recursive Merge Sort

The Efficiency of Merge Sort

Iterative Merge Sort

Merge Sort in the Java Class Library

### Quick Sort

The Efficiency of Quick Sort

Creating the Partition

Implementing Quick Sort

Quick Sort in the Java Class Library

### Radix Sort

Pseudocode for Radix Sort

The Efficiency of Radix Sort

Comparing the Algorithms

## Java Interlude 4: More About Exceptions

Programmer-Defined Exception Classes

Inheritance and Exceptions

The finally Block

## Chapter 10: Queues, Deques, and Priority Queues

### The ADT Queue

A Problem Solved: Simulating a Waiting Line

A Problem Solved: Computing the Capital Gain in a Sale of Stock

Java Class Library: The Interface Queue

### The ADT Deque

A Problem Solved: Computing the Capital Gain in a Sale of Stock

Java Class Library: The Interface Deque

Java Class Library: The Class ArrayDeque

### The ADT Priority Queue

A Problem Solved: Tracking Your Assignments

Java Class Library: The Class PriorityQueue

# **Table of Contents**

## **Chapter 11: Queue, Deque, and Priority Queue Implementations**

- A Linked Implementation of a Queue

- An Array-Based Implementation of a Queue

  - A Circular Array

  - A Circular Array with One Unused Location

- Circular Linked Implementations of a Queue

  - A Two-Part Circular Linked Chain

- Java Class Library: The Class `AbstractQueue`

- A Doubly Linked Implementation of a Deque

- Possible Implementations of a Priority Queue

## **Chapter 12: Lists**

- Specifications for the ADT List

- Using the ADT List

- Java Class Library: The Interface `List`

- Java Class Library: The Class `ArrayList`

## **Chapter 13: A List Implementation That Uses an Array**

- Using an Array to Implement the ADT List

  - An Analogy

  - The Java Implementation

  - The Efficiency of Using an Array to Implement the ADT List

## **Chapter 14: A List Implementation That Links Data**

- Operations on a Chain of Linked Nodes

  - Adding a Node at Various Positions

  - Removing a Node from Various Positions

  - The Private Method `getNodeAt`

- Beginning the Implementation

  - The Data Fields and Constructor

  - Adding to the End of the List

  - Adding at a Given Position Within the List

  - The Methods `isEmpty` and `toArray`

  - Testing the Core Methods

- Continuing the Implementation

- A Refined Implementation

  - The Tail Reference

- The Efficiency of Using a Chain to Implement the ADT List



# Table of Contents

Java Class Library: The Class LinkedList

## Java Interlude 5: Iterators

What Is an Iterator?

The Interface Iterator

The Interface Iterable

Using the Interface Iterator

Iterable and for-each Loops

The Interface ListIterator

The Interface List Revisited

Using the Interface ListIterator

## Chapter 15: Iterators for the ADT List

Ways to Implement an Iterator

A Separate Class Iterator

An Inner Class Iterator

A Linked Implementation

An Array-Based Implementation

Why Are Iterator Methods in Their Own Class?

An Array-Based Implementation of the Interface ListIterator

The Inner Class

## Java Interlude 6: Mutable and Immutable Objects

Mutable Objects

Immutable Objects

Creating a Read-Only Class

Companion Classes

## Chapter 16: Sorted Lists

Specifications for the ADT Sorted List

Using the ADT Sorted List

A Linked Implementation

The Method add

The Efficiency of the Linked Implementation

An Implementation That Uses the ADT List

Efficiency Issues

## Java Interlude 7: Inheritance and Polymorphism

# **Table of Contents**

## Further Aspects of Inheritance

- When to Use Inheritance

- Protected Access

- Abstract Classes and Methods

- Interfaces Versus Abstract Classes

## Polymorphism

## Chapter 17: Inheritance and Lists

- Using Inheritance to Implement a Sorted List

- Designing a Base Class

  - Creating an Abstract Base Class

- An Efficient Implementation of a Sorted List

  - The Method add

## Chapter 18: Searching

- The Problem

- Searching an Unsorted Array

  - An Iterative Sequential Search of an Unsorted Array

  - A Recursive Sequential Search of an Unsorted Array

  - The Efficiency of a Sequential Search of an Array

- Searching a Sorted Array

  - A Sequential Search of a Sorted Array

  - A Binary Search of a Sorted Array

  - Java Class Library: The Method `binarySearch`

  - The Efficiency of a Binary Search of an Array

- Searching an Unsorted Chain

  - An Iterative Sequential Search of an Unsorted Chain

  - A Recursive Sequential Search of an Unsorted Chain

  - The Efficiency of a Sequential Search of a Chain

- Searching a Sorted Chain

  - A Sequential Search of a Sorted Chain

  - A Binary Search of a Sorted Chain

- Choosing a Search Method

## Java Interlude 8: Generics Once Again

- More Than One Generic Type

## Chapter 19: Dictionaries

# Table of Contents

## Specifications for the ADT Dictionary

- A Java Interface

- Iterators

## Using the ADT Dictionary

- A Problem Solved: A Directory of Telephone Numbers

- A Problem Solved: The Frequency of Words

- A Problem Solved: A Concordance of Words

## Java Class Library: The Interface Map

## Chapter 20: Dictionary Implementations

### Array-Based Implementations

- An Unsorted Array-Based Dictionary

- A Sorted Array-Based Dictionary

### Linked Implementations

- An Unsorted Linked Dictionary

- A Sorted Linked Dictionary

## Chapter 21: Introducing Hashing

### What Is Hashing?

### Hash Functions

- Computing Hash Codes

- Compressing a Hash Code into an Index for the Hash Table

### Resolving Collisions

- Open Addressing with Linear Probing

- Open Addressing with Quadratic Probing

- Open Addressing with Double Hashing

- A Potential Problem with Open Addressing

- Separate Chaining

## Chapter 22: Hashing as a Dictionary Implementation

### The Efficiency of Hashing

- The Load Factor

- The Cost of Open Addressing

- The Cost of Separate Chaining

### Rehashing

### Comparing Schemes for Collision Resolution

### A Dictionary Implementation That Uses Hashing

- Entries in the Hash Table

- Data Fields and Constructors

# Table of Contents

The Methods `getValue`, `remove`, and `add`  
Iterators

Java Class Library: The Class `HashMap`

Java Class Library: The Class `HashSet`

## Chapter 23: Trees

### Tree Concepts

Hierarchical Organizations  
Tree Terminology

### Traversals of a Tree

Traversals of a Binary Tree  
Traversals of a General Tree

### Java Interfaces for Trees

Interfaces for All Trees  
An Interface for Binary Trees

### Examples of Binary Trees

Expression Trees  
Decision Trees  
Binary Search Trees  
Heaps

### Examples of General Trees

Parse Trees  
Game Trees

## Chapter 24: Tree Implementations

### The Nodes in a Binary Tree

A Class of Binary Nodes

### An Implementation of the ADT Binary Tree

Creating a Basic Binary Tree  
The Method `privateSetTree`  
Accessor and Mutator Methods  
Computing the Height and Counting Nodes  
Traversals

### An Implementation of an Expression Tree

### General Trees

A Node for a General Tree

### Using a Binary Tree to Represent a General Tree

# Table of Contents

## Java Interlude 9: Cloning

### Cloneable Objects

- Cloning an Array

- Cloning a Chain

- A Sorted List of Clones

- Cloning a Binary Node

## Chapter 25: A Binary Search Tree Implementation

### Getting Started

- An Interface for the Binary Search Tree

- Duplicate Entries

- Beginning the Class Definition

### Searching and Retrieving

### Traversing

### Adding an Entry

- A Recursive Implementation

- An Iterative Implementation

### Removing an Entry

- Removing an Entry Whose Node Is a Leaf

- Removing an Entry Whose Node Has One Child

- Removing an Entry Whose Node Has Two Children

- Removing an Entry in the Root

- A Recursive Implementation

- An Iterative Implementation

### The Efficiency of Operations

- The Importance of Balance

- The Order in Which Nodes Are Added

### An Implementation of the ADT Dictionary

## Chapter 26: A Heap Implementation

### Reprise: The ADT Heap

- Using an Array to Represent a Heap

- Adding an Entry

- Removing the Root

- Creating a Heap

- Heap Sort

# Table of Contents

## Chapter 27: Balanced Search Trees

### AVL Trees

- Single Rotations

- Double Rotations

- Implementation Details

### 2-3 Trees

- Searching a 2-3 Tree

- Adding Entries to a 2-3 Tree

- Splitting Nodes During Addition

### 2-4 Trees

- Adding Entries to a 2-4 Tree

- Comparing AVL, 2-3, and 2-4 Trees

### Red-Black Trees

- Properties of a Red-Black Tree

- Adding Entries to a Red-Black Tree

- Java Class Library: The Class TreeMap

### B-Trees

## Chapter 28: Graphs

### Some Examples and Terminology

- Road Maps

- Airline Routes

- Mazes

- Course Prerequisites

- Trees

### Traversals

- Breadth-First Traversal

- Depth-First Traversal

### Topological Order

### Paths

- Finding a Path

- The Shortest Path in an Unweighted Graph

- The Shortest Path in a Weighted Graph

### Java Interfaces for the ADT Graph

## Chapter 29: Graph Implementations

### An Overview of Two Implementations



# Table of Contents

The Adjacency Matrix

The Adjacency List

## Vertices and Edges

Specifying the Class Vertex

The Inner Class Edge

Implementing the Class Vertex

## An Implementation of the ADT Graph

Basic Operations

Graph Algorithms

## Appendix A: Documentation and Programming Style

### Naming Variables and Classes

### Indenting

### Comments

Single-Line Comments

Comment Blocks

When to Write Comments

Java Documentation Comments

## Appendix B: Java Basics (online)

### Introduction

Applications and Applets

Objects and Classes

A First Java Application Program

### Elements of Java

Identifiers

Reserved Words

Variables

Primitive Types

Constants

Assignment Statements

Assignment Compatibilities

Type Casting

Arithmetic Operators and Expressions

Parentheses and Precedence Rules

# **Table of Contents**

Increment and Decrement Operators

Special Assignment Operators

Named Constants

The Class Math

Simple Input and Output Using the Keyboard and Screen

Screen Output

Keyboard Input Using the Class Scanner

The if-else Statement

Boolean Expressions

Nested Statements

Multiway if-else Statements

The Conditional Operator (Optional)

The switch Statement

Enumerations

Scope

Loops

The while Statement

The for Statement

The do-while Statement

Additional Loop Information

The Class String

Characters Within Strings

Concatenation of Strings

String Methods

The Class StringBuilder

Using Scanner to Extract Pieces of a String

Arrays

Array Parameters and Returned Values

Initializing Arrays

Array Index Out of Bounds

Use of = and == with Arrays

Arrays and the For-Each Loop

# Table of Contents

Multidimensional Arrays

Wrapper Classes

## Appendix C: Java Classes (online)

Objects and Classes

Using the Methods in a Java Class

References and Aliases

Defining a Java Class

Method Definitions

Arguments and Parameters

Passing Arguments

A Definition of the Class Name

Constructors

The Method toString

Methods That Call Other Methods

Methods That Return an Instance of Their Class

Static Fields and Methods

Overloading Methods

Enumeration as a Class

Packages

The Java Class Library

## Appendix D: Creating Classes from Other Classes

Composition

Adapters

Inheritance

Invoking Constructors from Within Constructors

Private Fields and Methods of the Superclass

Overriding and Overloading Methods

Multiple Inheritance

Type Compatibility and Superclasses

The Class Object

## Appendix E: File Input and Output (online)

# **Table of Contents**

## **Preliminaries**

Why Files?

Streams

The Kinds of Files

File Names

## **Text Files**

Creating a Text File

Reading a Text File

Changing Existing Data in a Text File

Defining a Method to Open a Stream

## **Binary Files**

Creating a Binary File of Primitive Data

Reading a Binary File of Primitive Data

Strings in a Binary File

Object Serialization

## **Glossary (online)**

## **Index**