Preface to the First Edition

"IT'S A WONDERFUL TIME TO BE ALIVE." At least that's what I've found myself saying over the past couple of decades. When I first started working with computers, they were resources used by a privileged (or in my case, persistent) few. They were physically large, and logically small. They were cast from iron. The challenge was to make these behemoths solve complex problems quickly.

Today, computers are everywhere. They are in the office and at home. They speak to us on telephones; they zap our food in the microwave. They make starting cars in New England a possibility. Everyone's using them. What has aided their introduction into society is their diminished size and cost, and increased capability. The challenge is to make these behemoths solve complex problems quickly.

Thus, while the computer and its applications have changed over time, the challenge remains the same: *How can we get the best performance out of the current technology?* The design and analysis of data structures lay the fundamental groundwork for a scientific understanding of what computers can do efficiently. The motivations for data structure design work accomplished three decades ago in assembly language at the keypunch are just as familiar to us today as we practice our craft in modern languages on computers on our laps. The focus of this material is the identification and development of relatively *abstract* principles for structuring data in ways that make programs efficient in terms of their consumption of resources, *as well as efficient in terms of "programmability."*

In the past, my students have encountered this material in Pascal, Modula-2, and, most recently, C++. None of these languages has been ideal, but each has been met with increasing expectation. This text uses The Java Programming Language[1]—"Java"—to structure data. Java is a new and exciting language that has received considerable public attention. At the time of this writing, for example, Java is one of the few tools that can effectively use the Internet as a computing resource. That particular aspect of Java is not touched on greatly in this text. Still, Internet-driven applications in Java will need supporting data structures. This book attempts to provide a fresh and focused approach to the design and implementation of classic structures in a manner that meshes well with existing Java packages. It is hoped that learning this material in Java will improve the way working programmers craft programs, and the way future designers craft languages.

Pedagogical Implications. This text was developed specifically for use with CS2 in a standard Computer Science curriculum. It is succinct in its approach, and requires, perhaps, a little more effort to read. I hope, though, that this text

[1] Java is a trademark of Sun Microsystems, Incorporated.

becomes not a brief encounter with object-oriented data structure design, but a touchstone for one's programming future.

The material presented in this text follows the syllabus I have used for several years at Williams. As students come to this course with experience using Java, the outline of the text may be followed directly. Where students are new to Java, a couple of weeks early in the semester will be necessary with a good companion text to introduce the student to new concepts, and an introductory Java language text or reference manual is recommended. For students that need a quick introduction to Java we provide a tutorial in Appendix B. While the text was designed as a whole, some may wish to eliminate less important topics and expand upon others. Students may wish to drop (or consider!) the section on induction (Section 5.2.2). The more nontraditional topics—including, for example, iteration and the notions of symmetry and friction—have been included because I believe they arm programmers with important mechanisms for implementing and analyzing problems. In many departments the subtleties of more advanced structures—maps (Chapter 15) and graphs (Chapter 16)—may be considered in an algorithms course. Chapter 6, a discussion of sorting, provides very important motivating examples and also begins an early investigation of algorithms. The chapter may be dropped when better examples are at hand, but students may find the refinements on implementing sorting interesting.

`List`

Associated with this text is a Java package of data structures that is freely available over the Internet for noncommercial purposes. I encourage students, educators, and budding software engineers to download it, tear it down, build it up, and generally enjoy it. In particular, students of this material are encouraged to follow along with the code online as they read. Also included is extensive documentation gleaned from the code by `javadoc`. All documentation—within the book and on the Web—includes pre- and postconditions. The motivation for this style of commenting is provided in Chapter 2. While it's hard to be militant about commenting, this style of documentation provides an obvious, structured approach to minimally documenting one's methods that students can appreciate and users will welcome. These resources, as well as many others, are available from McGraw-Hill at `http://www.mhhe.com/javastructures`.

`nim`

Three icons appear throughout the text, as they do in the margin. The top "compass" icon highlights the statement of a *principle*—a statement that encourages abstract discussion. The middle icon marks the first appearance of a particular class from the `structure` package. Students will find these files at McGraw-Hill, or locally, if they've been downloaded. The bottom icon similarly marks the appearance of example code.

Finally, I'd like to note an unfortunate movement away from studying the implementation of data structures, in favor of studying applications. In the extreme this is a disappointing and, perhaps, dangerous precedent. The design of a data structure is like the solution to a riddle: the process of developing the answer is as important as the answer itself. The text may, however, be used as a reference for using the `structure` package in other applications by selectively avoiding the discussions of implementation.

Preface to the Second Edition

Since the first edition of *Java Structures* support for writing programs in Java[2] has grown considerably. At that time the Java Development Toolkit consisted of 504 classes in 23 packages[3] In Java 1.2 (also called Java 2) Sun rolled out 1520 classes in 59 packages. This book is ready for Java 1.4, where the number of classes and packages continues to grow.

Most computer scientists are convinced of the utility of Java for programming in a well structured and platform independent manner. While there are still significant arguments about important aspects of the language (for example, support for generic types), the academic community is embracing Java, for example, as the subject of the Computer Science Advanced Placement Examination.

It might seem somewhat perplexing to think that many aspects of the original Java environment have been retracted (or *deprecated*) or reconsidered. The developers at Sun have one purpose in mind: to make Java the indispensable language of the current generation. As a result, documenting their progress on the development of data structures gives us valuable insight into the process of designing useful data structures for general purpose programming. Those students and faculty considering a move to this second edition of *Java Structures* will see first-hand some of the decisions that have been made in the intervening years. During that time, for example, the `Collection`-based classes were introduced, and are generally considered an improvement. Another force—one similar to calcification—has left a trail of backwards compatible features that are sometimes difficult to understand. For example, the `Iterator` class was introduced, but the `Enumeration` class was not deprecated. One subject of the first edition—the notion of `Comparable` classes—has been introduced into a number of important classes including `String` and `Integer`. This is a step forward and a reconsideration of what we have learned about that material has lead to important improvements in the text.

Since the main purpose of the text is to demonstrate the design and behavior of traditional data structures, we have not generally tracked the progress of Java where it blurs the view. For example, Java 2 introduces a `List` interface (we applaud) but the `Vector` class has been extended to include methods that are, essentially, motivated by linked lists (we wonder). As this text points out frequently, the purpose of an interface is often to provide *reduced* functionality. If the data structure does not *naturally* provide the functionality required by the application, it is probably not an effective tool for solving the problem: search elsewhere for an effective structure.

---

[2] The Java Programming Language is a trademark of Sun Microsystems, Incorporated.

[3] David Flanagan, et al., *Java in a Nutshell*, O'Reilly & Associates.

As of this writing, more than $100,000$ individuals have searched for and downloaded the `structure` package. To facilitate using the comprehensive set of classes with the Java 2 environment, we have provided a number of features that support the use of the `structure` package in more concrete applications. Please see Appendix C.

Also new to this edition are more than 200 new problems, several dozen exercises, and over a dozen labs we regularly use at Williams.

Acknowledgments. Several students, instructors, and classes have helped to shape this edition of *Java Structures*. Parth Doshi and Alex Glenday—diligent Williams students—pointed out a large number of typos and stretches of logic. Kim Bruce, Andrea Danyluk, Jay Sachs, and Jim Teresco have taught this course at Williams over the past few years, and have provided useful feedback. I tip my hat to Bill Lenhart, a good friend and advisor, who has helped improve this text in subtle ways. To Sean Sandys I am indebted for showing me new ways to teach new minds.

The various reviewers have made, collectively, hundreds of pages of comments that have been incorporated (as much as possible) into this edition: Eleanor Hare and David Jacobs (Clemson University), Ram Athavale (North Carolina State University), Yannick Daoudi (McGill University), Walter Daugherty (Texas A&M University), Subodh Kumar (Johns Hopkins University), Toshimi Minoura (Oregon State University), Carolyn Schauble (Colorado State University), Val Tannen (University of Pennsylvania), Frank Tompa (University of Waterloo), Richard Wiener (University of Colorado at Colorado Springs), Cynthia Brown Zickos (University of Mississippi), and my good friend Robbie Moll (University of Massachusetts). Deborah Trytten (University of Oklahoma) has reviewed both editions! Still, until expert authoring systems are engineered, authors will remain human. Any mistakes left behind or introduced are purely those of the author.

The editors and staff at McGraw-Hill–Kelly Lowery, Melinda Dougharty, John Wannemacher, and Joyce Berendes–have attempted the impossible: to keep me within a deadline. David Hash, Phil Meek, and Jodi Banowetz are responsible for the look and feel of things. I am especially indebted to Lucy Mullins, Judy Gantenbein, and Patti Evers whose red pens have often shown me a better way.

Betsy Jones, publisher and advocate, has seen it all and yet kept the faith: thanks.

Be aware, though: long after these pages are found to be useless folly, my best work will be recognized in my children, Kate, Megan, and Ryan. None of these projects, of course, would be possible without the support of my best friend, my north star, and my partner, Mary.

Enjoy!

Duane A. Bailey
Williamstown, May 2002

Preface to the $\sqrt{7}$ Edition

In your hand is a special edition of *Java Structures* designed for use with two semesters of Williams' course on data structures, Computer Science 136. This version is only marginally different than the preceding edition, but is positioned to make use of Java 5 (the trademarked name for version 1.5 of the JDK). Because Java 5 may not be available (yet) on the platform you use, most of the code available in this book will run on older JDK's. The one feature that would not be available is Java's new `Scanner` class from the `java.util` package; an alternative is my `ReadStream` class, which is lightly documented in Section B.3.1 on page 494. It is a feature of the `structure` package soon to be removed.

In making this book available in this paperbound format, my hope is that you find it a more inviting place to write notes: additions, subtractions, and updates that you're likely to have discussed in class. Sometimes you'll identify improvements, and I hope you'll pass those along to me. In any case, you can download the software (as hundreds of thousands have done in the past) and modify it as you desire.

On occasion, I will release new sections you can incorporate into your text, including a discussion of how the `structure` package can make use of *generic* types.

I have spent a considerable amount of time designing the `structure` package. The first structures were available 8 years ago when Java was still in its infancy. Many of the structures have since been incorporated (directly or indirectly) into Sun's own `JDK`. (Yes, we've sold a few books in California.) Still, I feel the merit of my approach is a slimness that, in the end, you will not find surprising.

Meanwhile, for those of you keeping track, the following table (adapted from the 121 cubic inch, 3 pound 6 ounce, Fifth edition of David Flanagan's essential *Java in a Nutshell*) demonstrates the growth of Java's support:

JDK	**Packages**	**Classes**	**Features**
1.0	8	212	First public version
1.1	23	504	Inner classes
1.2 (Java 2)	59	1520	Collection classes
1.3	76	1842	A "maintenance" release.
1.4	135	2991	Improvments, including `assert`
1.5 (Java 5)	166	3562	Generics, autoboxing, and "varargs."

Seeing this reminds me of the comment made by Niklaus Wirth, designer of Pascal and the first two releases of Modula. After the design team briefed him on the slew of new features to be incorporated into Modula 3, he parried: "But, what features have you removed?" A timeless question.

Acknowledgments. This book was primarily written for students of Williams College. The process of publishing and reviewing a text tends to move the focus off campus and toward the bottom line. The Route 7 edition[4]—somewhere between editions 2 and 3—is an initial attempt to bring that focus back to those students who made it all possible.

For nearly a decade, students at many institutions have played an important role in shaping these resources. In this edition, I'm especially indebted to Katie Creel '10 (Williams) and Brian Bargh '07 (Messiah): thanks!

Many colleagues, including Steve Freund '95 (Stanford, now at Williams), Jim Teresco '92 (Union, now at Mount Holyoke), and especially Gene Chase '65 (M.I.T., now at Messiah) continue to nudge this text in a better direction. Brent Heeringa '99 (Morris, now at Williams) showers all around him with youthful enthusiasm.

And a most special thanks to Bill Mueller for the shot heard around the world—the game-winning run that showed all things were possible. Called by Joe Castiglione '68 (Colgate, now at Fenway):

> "Three-and-one to Mueller. One out, nineth inning. 10-9 Yankees, runner at first. Here's the pitch...swing and a High Drive Deep to Right...Back Goes Sheffield to the Bullpen...AND IT IS GONE!...AND THE RED SOX HAVE WON IT!...ON A WALKOFF TWO RUN HOMER BY BILL MUELLER OFF MARIANO RIVERA! CAN YOU BELIEVE IT?!"

Have I been a Red Sox fan all my life? Not yet.

Finally, nothing would be possible without my running mate, my Sox buddy, and my best friend, Mary.

Cheers!

Duane A. Bailey '82 (Amherst, now at Williams)
Williamstown, September 2007

[4] Route 7 is a scenic byway through the Berkshires and Green Mountains that eddies a bit as it passes through Williamstown and Middlebury.