Data Structures and Algorithms

TP31
B422

John Beidler

DATA STRUCTURES AND ALGORITHMS

An Object-Oriented Approach Using Ada 95

With 29 Illustrations

Springer

John Beidler
Department of Computing Sciences
University of Scranton
Scranton, PA 18510-4664
USA

Series Editors
David Gries
Fred B. Schneider
Department of Computer Science
Cornell University
Upson Hall
Ithaca, NY 14853-7501
USA

Library of Congress Cataloging-in-Publication Data
Beidler, John, 1941–
Data structures and algorithms : an object-oriented approach using Ada 95 / John Beidler.
p. cm. – (Undergraduate texts in computer science)
Includes bibliographical references and index.
ISBN 0-387-94834-1 (hardcover : alk. paper)
1. Object-oriented programming (Computer science) 2. Ada (Computer progam language) 3. Data structures (Computer science) 4. Computer algorithms. I. Title. II. Series.
QA76.64.B43 1996
005.13'3–dc20 96-23982

Printed on acid-free paper.

Production managed by Bill Imbornoni; manufacturing supervised by Joe Quatela.
Camera-ready copy prepared using the author's WordPerfect files.
Printed and bound by R.R. Donnelley and Sons, Harrisonburg, VA.
Printed in the United States of America.

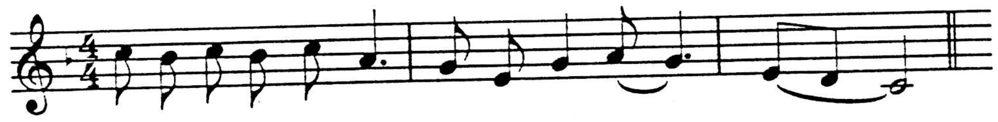
9 8 7 6 5 4 3 2 1

ISBN 0-387-94834-1 Springer-Verlag New York Berlin Heidelberg SPIN 10523571

UNDERGRADUATE TEXTS IN COMPUTER SCIENCE

Editors
David Gries
Fred B. Schneider

Springer
New York
Berlin
Heidelberg
Barcelona
Budapest
Hong Kong
London
Milan
Paris
Santa Clara
Singapore
Tokyo



(Pauling-Bass) Lois Music
As performed by the Shirelles
transcribed by Joan Roccasalvo, CSJ

Preface

Picture this: sitting in a cottage by a peat fire in a small farm village just outside of Limerick City in Ireland, with a pint of Guinness, a copy of Booch's software components in Ada book, and a laptop computer. That picture describes how I spent part of my sabbatical during the 1989–1990 academic year, after spending the summer of 1989 working with an Ada programming group at the Naval Surface Warfare Center in Dahlgren, Virginia. This was preceded by many years of looking at ways of improving the Data Structures and Algorithms course as more and more material filtered out of that course and into the CS 2 course. That year in Limerick provided the opportunity to think through the variety of issues that led to this book.

The Data Structures and Algorithms course is commonly referred to as the ACM CS 7 course. During the past decade, a large amount of material has moved from the CS 7 course to the CS 2 course. I viewed this relationship between the CS 2 and CS 7 courses as an opportunity to enhance the CS 2 course and modernize the CS 7 course. Two elements that played key roles in this process are software reuse and object-oriented programming.

My experiences at Dahlgren, during the summer of 1989, convinced me of Ada's value as an educational tool. I found Ada's features to be great tools for enhancing and presenting software development concepts. Of particular importance, I found Ada's encapsulation features and the ability to present specifications without even a hint of implementational details an extremely important software abstraction feature.

While I was on my sabbatical, my colleagues back at the University of Scranton made the decision to select Ada as the core programming language for the CS 2 and CS 7 courses. We had been using Modula-2 since 1980. Dr. Dennis Martin played a lead role in making the case for the transition to Ada. In fact, the year I was away was a significant year for our department, as we moved to new quarters with new laboratories and a fully networked environment. I'd like to recognize the key role Rich Plishka played in putting it all together.

Between 1990 and 1992 we received two Software Engineering and Ada grants for the development of support resources for the CS 2 and CS 7 courses. This provided us with an opportunity to build upon the resources that were constructed during my sabbatical. In 1992 we started refocusing our materials toward the coming transition to Ada 9X, today we know as Ada 95. I must acknowledge the direct and indirect roles Bob McCloskey played in the development of the data structure suites that I use to support both courses.

This transition to an Ada 95–based resource lead, in the fall of 1994—to experimentation with packages that employed type extension in lieu of generic instantiation as the method of interfacing reusable software components to the needs of clients—eventually led to the construction of a second suite of data structure components that has a very distinct object-oriented flavor.

The Course

The course implied by this book requires substantial software support. I spent my 1989–1990 sabbatical constructing an outline for the course and course support materials, including a component suite, based upon the Booch components, intended to meet educational needs, but with industrial-strength features. I wanted these components to provide good object-oriented support and software reuse experience for students in the CS 2 and CS 7 courses.

It is assumed that the reader of this book is familiar with the topics normally covered in a strong CS 2 course. We expect the CS 2 course to be a broad-based introduction to the discipline with a strong emphasis on analysis, design, abstraction, and the basics of formal specifications, and with a software engineering flavor. In the CS 2 course the students gain a fundamental understanding of the essential concepts of the basic logical data structures, like stacks, queues, lists, and trees. This book moves forward from that foundation with an in-depth presentation of representation, encapsulation, and measurement issues.

There are four recurring themes in this book: abstraction, implementation, encapsulation, and measurement. One significant difference you may find between this book and others is that this book addresses a great variety of encapsulation issues and separates representational issues from encapsulation issues.

There is more than enough material in this book to support a CS 7 course. My preference is to cover the first six chapters in the first half of the semester. My experience has been that the first two chapters must be addressed carefully. Students never seem to fully grasp the implications of static representations, the fundamental reasons behind the need for both `private` and `limited private` types, and the subtleties of tagged and controlled types. While covering Chapters 3 through 6, I emphasize encapsulation and the variety of implementation and measurement issues.

During the second half of the semester, I cover about 80 percent of the material in the remaining chapters, leaving the uncovered material as the basis for individual and team projects. For example, in Chapter 7 I may cover the AVL tree restructuring in depth, then give a cursory presentation of B-trees, leaving the construction of B-tree algorithms as an assignment. Another year I will switch and do B-trees in depth and give various AVL-based assignments.

Support

This book describes, and is supported by, two approaches to encapsulation, the traditional encapsulation of reusable Ada resources in generic packages and a polymorphic approach that makes use of Ada 95's object-oriented features. Both suites may be obtained across the World Wide Web from

```
http://academic.uofs.edu/faculty/beidler/
```

by following the Ada link. The packages may also be obtained through an anonymous ftp from

```
ftp.cs.uofs.edu
```

in the `pub/Ada` directory. For those without network access, the data structure component suites may also be obtained by contacting the author at (717) 941-7774.

Gratia Tibi Ago . . .

So many people contributed in so many ways to this manuscript. Of particular note are the many students who suffered through many variations of course notes in Cmps 240 from 1991 to 1996. They were the biggest contributors. They pushed, and questioned, and made being an educator a real treat.

I'd like to thank my departmental colleagues. I could not forget our departmental support staff of Mary Alice Lynott and Bill Gunshannon. Without Mary Alice's assistance at critical times, and Bill's work in keeping our "ancient" UNIX systems alive, I wonder if this book would ever have been completed. If it was not for our department's unofficial open-door policy, I could not have tried out many ideas on Rich, Bob, Paul, Bi, Dennis, and Dick. I would also like to thank Chip for allowing me to try to improve his racquetball game.

My sabbatical in Ireland played an important role in getting this book started. If Mary Engel, the associate dean of the College of Arts and Sciences, had not met with Barra O'Cinneadie of the University of Limerick, my sabbatical would not have come to be. The year at Limerick was made both a pleasure and an academic success by such good folks as Wally Ryder, Tony Cahill, Norah Powers, and Mike Coughlin. Nor can I forget the invaluable contribution of Fionbarr McLaughlin and Liam O'Brien in teaching me to play squash.

In the last few years, numerous discussions with three good "Ada" friends helped this project along. They are John McCormack, Mike Feldman, and Nick DeLillo. Nick and I, along with our wives, share a full appreciation of the other CIA, The Culinary Institute of America. Now if only he could learn to make a good cappuccino.

Of course, what makes this venture so worthwhile is the support and love of my family. They may be added to the dedication, if you figured it out, by replacing the word "one" with the word "ones" when you sing the song. Now that this book is completed, you may find it difficult getting in touch with me. I'm busy playing with my grandchildren.

Jack Beidler
Summer 1996

Contents

1

Preliminaries

This book presents data structure techniques in the context of object-oriented software development with the eventual implementation of algorithms in Ada 95. Object-oriented software development is a contemporary approach to the design of reliable and robust software. The complexity of the implementation of software systems is a combination of the complexity of the representations of information and the complexity of the algorithms that manipulate the representations. **Data structures** is the study of methods of representing objects, the design of algorithms to manipulate the representations, the proper **encapsulation** of objects in a reusable form, and the **evaluation** of the cost of the implementation, including the measurement of the complexity of the time and space requirements.

Programming languages play an important role in representing the solutions to problems in an efficient, reliable, and maintainable manner. Many modern programming languages support the layered representation of information and algorithms, frequently referred to as abstraction. **Abstraction** is the separation, or layering, of software to distinguish between **what** a data structure represents, or **what** an algorithm accomplishes, from the implementational details of **how** things are actually carried out. Abstraction is very important because frequently there are several competing methods for representing a structure. Usually, there are tradeoffs among competing representations and their algorithms. These