Ada for Software Engineers (Second Edition with Ada 2005)

Mordechai Ben-Ari

Ada for Software Engineers (Second Edition with Ada 2005)

Springer

Mordechai Ben-Ari, BSc, MSc, PhD
Weizmann Institute of Science
Rehovot 76100
Israel

ISBN: 978-1-84882-313-6 e-ISBN: 978-1-84882-314-3

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2009921268

First published 1998

Printed on acid-free paper

Springer Science+Business Media
springer.com

Preface

Albert Einstein once said everything should be made as simple as possible, but not simpler. Einstein could have been talking about programming languages, as the landscape is strewn with "simple" languages that, several versions later, have 500-page reference manuals!

The truth is that we expect a lot of our programming languages. We demand support for encapsulation and abstraction, type checking and exception handling, polymorphism and more. Ada, unlike other languages which grew by the gradual addition of features, was designed as a coherent programming language for complex software systems. *Ada for Software Engineers (ASE)* is written to equip you with the knowledge necessary to use the Ada programming language to develop software systems.

Although Ada never achieved the popularity of Java and the C-something languages, it remains the programming language of choice for reliable software. Every time you step on an airplane or a fast train, you are quite literally trusting your life to software written in Ada. Given the low level of reliability of much of the software we use daily, I can only regret that the use of Ada is not more widespread, and I hope that this book will encourage more software engineers to give Ada a chance.

Intended audience

The book is intended for software engineers making the transition to Ada, and for upper-level undergraduate and graduate students, including those who have had the good fortune to study Ada as their first programming language. No specific knowledge of Ada is assumed; the prerequisites are a basic knowledge of computer science and computer systems, and at least two years of programming experience. As the title implies, if you are a software engineer or training to become one, this book is for you.

Structure of the book

The complexity of modern programming languages leaves textbook writers with a painful dilemma: either gloss over the gory details, or write books that are heavy enough to be classified as lethal weapons. Is there another way? A concise description of Ada is freely available: the *Ada Reference Manual (ARM)* [7], which is the document defining the language standard. The *ARM* has a reputation for being ponderous reading meant for "language lawyers." Nevertheless, I believe that—with a bit of guidance—software engineers can learn to read most of the *ARM*. *ASE* is based upon two premises: (a) it is best to learn the individual language constructs within the context of actual programs, and (b) if the learning is based upon the terminology and concepts used in the language standard, you will be able to use the *ARM* as your primary reference manual.

The Ada language will be taught using a few relatively large case studies, rather than a large number of small examples each crafted to demonstrate a particular construct or rule. Experienced programmers know that the key to mastering a programming language is not to memorize the syntax and semantics of individual constructs, but to learn how to integrate the constructs into language-specific paradigms. We will need to gloss over details when explaining a case study; rest assured that everything will eventually be explained. Certain sections are marked with an asterisk and can be omitted during your initial study of Ada. This material is not necessarily more difficult, but you can't learn everything at once, and these are topics that can be left for your second reading of the book.

The chapters in the book can be divided into five parts and three appendices. The first two parts contain the core concepts of Ada and should probably be read first and in sequence (skipping the starred sections, of course). The others chapters can be read as necessary.

1. After an introductory chapter, Chapters 2–5 quickly cover elementary language constructs such as statements, subprograms, arrays, records and pointers that should be familiar from your previous programming experience.
2. The second part is based upon the progressive development of a case study demonstrating the Ada constructs for programming-in-the-large (including object-oriented programming): packages and private types (Chapter 6), type extension, inheritance, class-wide types and dynamic polymorphism (Chapter 7–8), and generics (Chapter 9).
3. Chapters 10–14 cover other topics and are more or less independent of the previous ones: exceptions, the type system in depth, access types (pointers), numerics and input–output.
4. The fourth part contains material on program structure (Chapter 15), the container library (Chapter 16), and interfaces (Chapter 17).
5. Chapters 18–22 include topics broadly covered by the term *systems programming*: multitasking, hardware interfaces, and real-time and distributed systems.

A special feature of the book is the comprehensive *Glossary* (Appendix A), which explains the *ARM* terminology with examples. In addition, discussions in the text always contain references to the relevant paragraphs in the *ARM*. The *Index of ARM Sections* will be invaluable when you begin to use the *ARM* as a reference manual.

At the end of each chapter are *Projects* and *Quizzes*. Projects are non-trivial programming exercises, many of which ask you to extend the case studies in the text. Quizzes are *not* routine exercises as the term is usually used; they are intended help you understand the finer points of the Ada language, and can be skipped until you have significant experience programming with Ada. Each quiz is a small Ada program; you are to examine it and decide what will happen when the program is compiled, and—if the program contains executable statements and compiles successfully—the result of the execution. Appendix B *Hints* refers you to the relevant clauses of the *ARM* from which you should be able to find the answers to the quizzes *before* looking at Appendix C *Answers*. (To save space, **`with`** and **`use`** clauses for `Ada.Text_IO`, `Ada.Integer_IO` and `Ada.Exceptions` are omitted.)

Supplementary material and links

All the case studies, quizzes and programs in the book were compiled and executed. The full source code is available in the software archive at:

- `http://www.springer.com/978-1-84882-313-6`.

The programs were developed using the GNAT compiler for Ada 2005. It is available in commercial, academic and free versions; see:

- `https://libre.adacore.com`.

There are two comprehensive websites on Ada:

- ACM Special Interest Group on Ada (SIGAda), `http://www.sigada.org`,
- Ada Resource Association (ARA), `http://www.adaic.com`.

From these websites you can download documents such as the *ARM*, and find links to compiler vendors, support software, relevant conferences, and other resources.

Ada 2005

This book was original written for the Ada 95 version of the language. That edition is out of print and I have used the occasion of the publication of a new version of Ada, Ada 2005, to expand and improve the book. Constructs of Ada 2005 are freely used, but sections that use them are annotated to indicate which constructs are new. The software archive for the previous edition of the book will remain freely available.

Acknowledgments

First edition: I would like to thank Michael Feldman, Kevlin Henney, Richard Riehle, Reuben Sumner, Tucker Taft and Peter Wegner for reviewing the manuscript, and Simon Plumtree of John Wiley for his support and assistance.

Second edition: I am grateful to Edmond Schonberg for his constant help as I learned Ada 2005. I am indebted to Robert Duff for his comprehensive review that enabled me to correct many bugs in the text. The staff at AdaCore has been extremely helpful answering my queries on the GNAT compiler. I would like to thank Beverley Ford and the staff at Springer for their friendly and efficient handling of the publishing issues that I raised.

Mordechai (Moti) Ben-Ari
Rehovot, 2008

Contents