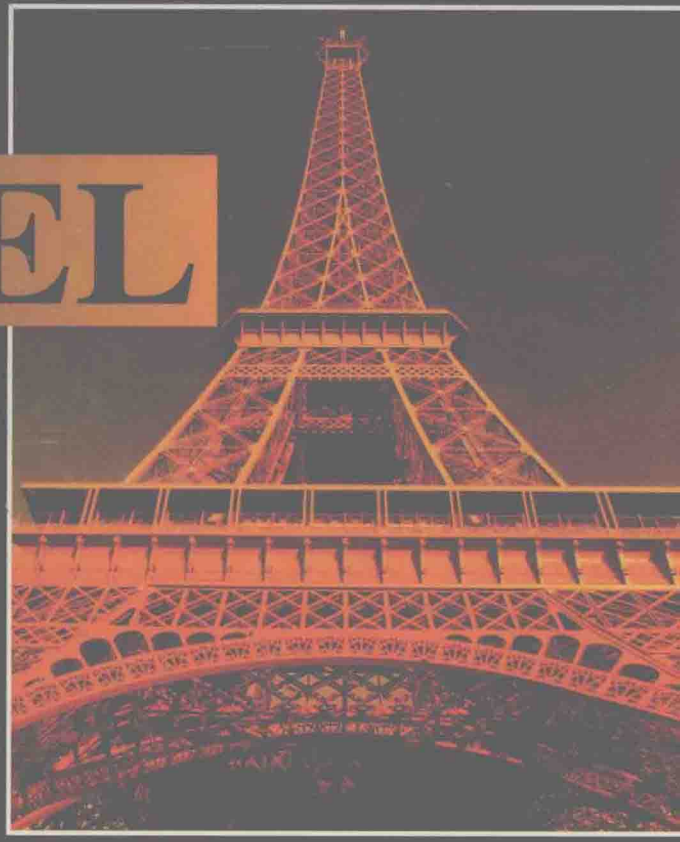
SOFTWARE DEVELOPMENT USING EIFFEL



THERE CAN BE LIFE OTHER THAN C++

- Presents the latest version of Eiffel
- Shows Eiffel in action
- Demonstrates Booch '94 method and notation
- Compares Eiffel with C++

RICHARD WIENER

THE OBJECT-ORIENTED SERIES

SOFTWARE DEVELOPMENT USING EIFFEL:
THERE CAN BE LIFE OTHER THAN C++

RICHARD WIENER

Prentice Hall PTR, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data
Wiener, Richard, 1941–
Software development using Eiffel : there can be life other than C++ / Richard Wiener.
p. cm. — (Prentice Hall object-oriented series)
Includes index.
ISBN 0-13-100686-X (hardcover)
1. Computer software—Development. 2. Eiffel (Computer program language) I. Title. II. Series.
QA76.76.D47W52 1995 94-37783
005.13'3—dc20 CIP

Editorial/Production Management: *Digital Communications Services*
Cover Design: *Miguel Ortiz*
Manufacturing: *Alexis R. Heydt*
Acquisitions Editor: *Paul W. Becker*

© 1995 Prentice Hall PTR
Prentice-Hall, Inc.
A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632

The publisher offers discounts on this book when ordered in bulk quantities. For more information, contact Corporate Sales Department, Prentice Hall PTR, 113 Sylvan Avenue, Englewood Cliffs, NJ 07632. Phone: 800-382-3419; Fax: 201-592-2249; e-mail: dan_rush@prenhall.com

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-100686-X

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

SOFTWARE DEVELOPMENT USING EIFFEL:

THERE CAN BE LIFE OTHER THAN C++

PRENTICE HALL
OBJECT-ORIENTED
SERIES

D. COLEMAN, P. ARNOLD, S. BODOFF,
C. DOLLIN, H. GILCHRIST, F. HAYES
AND P. JEREMAES
Object-Oriented Development:
The Fusion Method

S. COOK AND J. DANIELS
Designing Object Systems

B. HENDERSON-SELLERS
A Book of Object-Oriented Knowledge

B. HENDERSON-SELLERS
Book Two of Object-Oriented Knowledge

H. KILOV AND J. ROSS
Information Modelling

P. KRIEF
Prototyping with Objects

K. LANO AND H. HAUGHTON
Object-Oriented Specification Case Studies

J. LINDSKOV KNUDSEN, M. LÖFGREN,
O. LEHRMANN MADSEN AND B. MAGNUSSON
Object-Oriented Environments:
The Mjølner Approach

M. LORENZ
Object-Oriented Software Development:
A Practical Guide

M. LORENZ
Object-Oriented Software Metrics

D. MANDRIOLI AND B. MEYER (eds)
Advances in Object-Oriented
Software Engineering

B. MEYER
An Object-Oriented Environment

B. MEYER
Eiffel: The Language

B. MEYER
Reusable Software

B. MEYER AND J. M. NERSON
Object-Oriented Applications

POMBERGER AND BALSCHEK
An Object-Oriented Approach in
Software Engineering

P. J. ROBINSON
Hierarchical Object-Oriented Design

R. SWITZER
Eiffel: An Introduction

K. WALDEN AND J. M. NERSON
Seamless Object-Oriented Software Architecture

R. WIENER
Software Development Using Eiffel

此为试读,需要完整PDF请访问: www.ertongbook.com

This book is dedicated
to my best friend, my wife
and my love, Hanne.

Preface

THE BOOK

This book is aimed at computer scientists, software development professionals, educators and students, programmers, and in particular C++ programmers, who wish to examine a compelling alternative to the C++ programming language for implementing object-oriented software systems. This alternative is Eiffel. This book is also about object-oriented analysis and design using the Booch '94 method with Eiffel implementation. The reader is assumed to have prior experience with object-oriented programming, most likely using C++ or Smalltalk, and at least basic familiarity with the concepts of object-oriented programming.

Part 1 of the book, Chapters 1 through 7, presents the major features of the Eiffel language from a programmer's perspective. These chapters contain many small but complete examples of Eiffel coding. These are used to illustrate the typical use of various language features. Part 2 of the book, Chapters 8 through 11, shows Eiffel in action. These chapters present case studies that, in addition to Eiffel implementation, show the use of the Booch '94 method of object analysis and design.

The standard reference book for the Eiffel programming language is *Eiffel: The Language,* written by Bertrand Meyer (Prentice Hall, 1992), the creator of Eiffel. This present book is meant to complement this reference, not compete with it. It is recommended that the reader also acquire this standard reference about Eiffel since there is no attempt in this book to duplicate the rigorous and fine-grained details concerning some of the "legal" issues related to the Eiffel language. Such issues are typically of greater concern to compiler writers but not programmers.

ITS CONTENT

Chapter 1, "The Flavor of Eiffel," presents a broad overview of the language, showing its major features. No attempt is made to provide detailed explanations of the many language features that are previewed in this chapter. A small class hierarchy associated with writing instruments is used to sample the flavor of Eiffel.

Chapter 2, "Classes and Objects," introduces and illustrates the various components of an Eiffel class. An Eiffel class serves as an implementation of an abstract data type, as well as a software module. The main issues discussed in this chapter include the creation of objects, features and their visibility, object assignment and equality, copying and cloning objects, polymorphism and dynamic-binding, generic classes and expanded types.

Chapter 3, "Correct Programs," introduces the powerful assertion-handling facilities of Eiffel. Pre and postconditions and class invariants are explained and illustrated. The notions of programming by contract and subcontracting are explored. Finally, Eiffel's exception-handling mechanism is presented.

Chapter 4, "Generic Container Classes," introduces the concept of constrained and unconstrained "genericity" and the construction of reusable software components. Several container classes, including a list, ordered list, stack, queue, search tree and sortable array, are presented to illustrate the basic ideas.

Chapter 5, "Inheritance in Eiffel," discusses the role of inheritance in object-oriented programming. The overall structure of an Eiffel inheritance clause and its many possible subclauses are presented and illustrated. These include redefinition, renaming, export scope and "undefining." Repeated inheritance structures are examined. The construction and role of abstract classes using deferred routines are explored. The issue of "mix-ins" or code inheritance is discussed.

Chapter 6, "Polymorphism and Conformance," focuses on the rules associated with late-binding in Eiffel. An application of polymorphism involving the concept of the "generality of numbers" is presented as a non-trivial application of polymorphism.

Chapter 7, "C++ && Eiffel or is it C++ & Eiffel or is it C++ and Eiffel? : A Clash of Cultures," provides a direct technical comparison of C++ an Eiffel.

Chapter 8, "Object-Oriented Analysis and Design," introduces the Booch '94 method and its notation. A supermarket checkout line simulation is used to illustrate the basic ideas.

Chapter 9, "An Ecological Simulation," presents a case study involving the construction of several dozen classes that illustrates the Booch analysis and design process and the use of the Eiffel language. After completing the initial design, some simple maintenance is performed to illustrate the importance of building a robust software structure.

Chapter 10, "A Game of Strategies and Investment," presents another case study that again involves the construction of several dozen classes. This example focuses on the extensive use of late-binding as a central design principle. After

the initial design is completed, it is critically examined and replaced with an improved design.

Chapter 11, "Simulated Annealing," shows the application of object-oriented analysis, design and programing in a less traditional area — heuristic algorithm design. An abstract class that encapsulates the basic mechanism of simulated annealing is constructed. A concrete class that solves the classic "Travelling Salesperson Problem" is constructed as a subclass of the abstract simulated annealing class to demonstrate the use and power of inheritance in this non-traditional setting.

Style and Format

An area of potential controversy in this book is the deviation from the "standard" Eiffel style recommended in Bertrand Meyer's reference *Eiffel: The Language* (ETL).

The arguments in favor of using a "standard" style are understood and appreciated by this author. It is this author's deep conviction that programming is as much an art as a science. It is an aesthetic and expressive activity that involves the personality as well as the intellect of the programmer. This author believes that every programmer, Eiffel or otherwise, should develop a style that is personal, comfortable and most importantly consistent. Of course, consistent is different from uniform. The programs in this book hopefully illustrate this principle.

The "standard" Eiffel style recommends that multiple word identifiers use an underscore to separate individual words. For example, *average_force* to represent the idea of average force. This book sometimes adheres to this recommendation, but not always. A more Smalltalk influenced style is often evident. This Smalltalk style would use the identifier *averageForce* to represent the idea of average force.

The "standard" Eiffel style involves a myriad of details related to the layout of various language artifacts, formation of identifier names, as well as the use of boldface fonts for all keywords. This recommended style is presented in Appendix A of the ETL. Since there are about 15 pages of details presented there, no attempt will be made here to summarize the "standard" style recommendations. The reader is urged to read this Appendix in the ETL and then decide for himself or herself.

A sample of "standard" Eiffel style is presented in the Appendix of this book so that the reader can either see what they are missing or be further encouraged to develop their own personal style.

Acknowledgments

Several people contributed to this book through discussions of many of the later examples. In particular, thanks go to two computer scientists very close to me: Erik, my son, and Hanne, my wife.

Jim McKim, of the Hartford Graduate Center, did more than review the first draft manuscript. He poured over every line of code in every example, making constructive suggestions, and contributed significantly to improvements in the book. Thank you so much Jim for your extremely constructive and useful effort.

The case studies in Chapters 8, 9 and 10 were class tested in a graduate course at the University of Colorado at Colorado Springs. I wish to thank Wes Munsil, a student in the class, for his constructive feedback throughout the course.

Other useful reviews that led to the improvement of the final manuscript were provided by Dr. Stuart Greenfield of Marist College and Paul Blanco of Object Center in Boston. I wish to thank Paul for his enthusiastic encouragement and help at every stage of this project.

A special thanks goes to Bertrand Meyer, not only for creating the fine language Eiffel, but for his encouragement, friendship and support throughout this project. His classic work, *Object-Oriented Software Construction*, published by Prentice Hall in 1988 has served as an inspiration to me and many other workers in this field. His work associated with Eiffel and his writings in general have made significant contributions to our field.

I wish to thank Interactive Software Engineering (ISE) in Goleta, California, for providing me with the latest versions of their compiler and tools as Eiffel has evolved over the years. In particular, I wish to thank Annie Meyer and Frederik Deramat for their help.

I thank Tower Technology in Austin, Texas, for also providing me with the latest versions of their compiler and tools. Their president, Robert Howard, and marketing director, Madison Cloutier, have provided tremendous help and encouragement throughout this project. Many of the programs and examples in this book were developed and tested using the TowerEiffel system.

I am most grateful to Paul Becker, Publisher at Prentice Hall, for his continuing support.

Finally, on a personal note, I am so grateful for the patience, encouragement, love and support of my wife Hanne. It is to you that this book is dedicated.

PART 1

The Language

此为试读,需要完整PDF请访问: www.ertongbook.com

Contents

此为试读,需要完整PDF请访问: www.ertongbook.com