Computer Science Series

Consulting Editor:

Professor F.H. Sumner, University of Manchester

- A. Abdellatif, J. Le Bihan, M. Limame, *Oracle - A User's Guide*
Ian O. Angell, *High-resolution Computer Graphics Using C*
Ian O. Angell and Gareth Griffith, *High-resolution Computer Graphics Using FORTRAN 77*
Ian O. Angell and Gareth Griffith, *High-resolution Computer Graphics Using Pascal*
M. Azmoodeh, *Abstract Data Types and Algorithms, second edition*
C. Bamford and P. Curran, *Data Structures, Files and Databases, second edition*
P. Beynon-Davies, *Information Systems Development, second edition*
G.M. Birtwistle, *Discrete Event Modelling on Simula*
Linda E.M. Brackenbury, *Design of VLSI Systems - A Practical Introduction*
Alan Bradley, *Peripherals for Computer Systems*
G.R. Brookes and A.J. Stewart, *Introduction to occam2 on the Transputer*
P.C. Capon and P.J. Jinks, *Compiler Engineering Using Pascal*
Robert Cole, *Computer Communications, second edition*
Eric Davalo and Patrick Naïm, *Neural Networks*
S.M. Deen, *Principles and Practice of Database Systems*
Joyce Duncan, Lesley Rackley and Alexandria Walker, *SSADM in Practice*
D. England *et al.*, *A Sun User's Guide, second edition*
Jean Ettinger, *Programming in C++*
J.S. Florentin, *Microprogrammed Systems Design*
A.B. Fontaine and F. Barrand, *80286 and 90386 Microprocessors*
Michel Gauthier, *Ada - A Professional Course*
M.G. Hartley, M. Healey and P.G. Depledge, *Mini and Microcomputer Systems*
J.A. Hewitt and R.J. Frank, *Software Engineering in Modula-2 - An Object-oriented Approach*
Patrick Jaulent, *The 68000 - Hardware and Software*
M.J. King and J.P. Pardoe, *Program Design Using JSP - A Practical Introduction, second edition*
Bernard Leguy, *Ada - A Programmer's Introduction*
M. Léonard, *Database Design Theory*
David Lightfoot, *Formal Specification Using Z*
A.M. Lister and R.D. Eager, *Fundamentals of Operating Systems, fifth edition*
Elizabeth Lynch, *Understanding SQL*
Tom Manns and Michael Coleman, *Software Quality Assurance, second edition*
G.P. McKeown and V.J. Rayward-Smith, *Mathematical Foundations for Computing*
B.A.E. Meekings, T.P. Kudrycki and M.D. Soren, *A book on C, third edition*
R.J. Mitchell, *C++ Object-oriented Programming*
R.J. Mitchell, *Microcomputer Systems Using the STE Bus*
R.J. Mitchell, *Modula-2 Applied*
Y. Nishinuma and R. Espesser, *UNIX - First Contact*
Ian Pratt, *Artificial Intelligence*
Pham Thu Quang and C. Chartier-Kastler, *MERISE in Practice*
E.J. Redfern, *Introduction to Pascal for Computational Mathematics*

F.D. Rolland, *Programming with VDM*
A.G. Sutcliffe, *Human-Computer Interface Design, second edition*
C.J. Theaker and G.R. Brookes, *Concepts of Operating Systems*
M. Thorin, *Real-time Transaction Processing*
M.R. Tolhurst *et al.*, *Open Systems Interconnection*
A.J. Tyrrell, *Eiffel Object-Oriented Programming*
I.R. Wilson and A.M. Addyman, *A Practical Introduction to Pascal, second edition*

Other titles

Ian O. Angell and Dimitrios Tsoubelis, *Advanced Graphics on VGA and XGA Cards Using Borland C++*
N. Frude, *A Guide to SPSS/PC+, second edition*
Peter Grossman, *Discrete Mathematics for Computing*
Percy Mett, *Introduction to Computing*
Tony Royce, *COBOL - An Introduction*
Tony Royce, *Structured COBOL - An Introduction*

Eiffel Object-Oriented Programming

A.J. Tyrrell

Manchester Metropolitan University

© A.J. Tyrrell, 1995

All rights reserved. No reproduction, copy or transmission of this publication may be made without written permission.

No paragraph of this publication may be reproduced, copied or transmitted save with written permission or in accordance with the provision of the Copyright, Designs and Patents Act 1988, or under the terms of any licence permitting limited copying issued by the Copyright Licensing Agency, 90 Tottenham Court Road, London W1P 9HE.

Any person who does any unauthorised act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

First published 1995 by
MACMILLAN PRESS LTD
Houndmills, Basingstoke, Hampshire RG21 6XS
and London
Companies and representatives
throughout the world

ISBN 978-0-333-64554-3 ISBN 978-1-349-13875-3 (eBook)
DOI 10.1007/978-1-349-13875-3

A catalogue record for this book is available
from the British Library.

10	9	8	7	6	5	4	3	2	1
04	03	02	01	00	99	98	97	96	95

Eiffel™ is a trademark of the Non-profit International Consortium for Eiffel.

Eiffel/S™ is a trademark of SiG Computer GmbH.

Contents

Preface	ix
1 Introduction to Object Orientation	1
1.1 Process-oriented software development: a text formatter	1
1.2 An object-oriented approach to developing a text formatter	3
1.3 Object-oriented and process-oriented approaches compared	5
1.4 Objects, classes and message passing	8
1.5 Information hiding	10
1.6 Inheritance	11
1.7 The concept of client	13
1.8 An overview of the Eiffel language	14
Exercises	17
2 Writing a Simple Class in Eiffel	19
2.1 The syntax of an Eiffel class	19
2.2 The Eiffel environment	21
2.3 Call instructions	22
2.4 I-O in Eiffel	23
2.5 The assignment instruction	25
2.6 Class attributes	26
2.7 Basic types	27
2.8 Reference types and creation instructions	28
2.9 Strings	30
2.10 Special characters	31
2.11 User-defined names	32
2.12 Reserved words	32
2.13 Special symbols	33
Exercises	33
3 Eiffel Basic Types	35
3.1 Type checking	35
3.2 Operations on numeric types	36
3.3 Boolean expressions	39
3.4 Semi-strict boolean operators	43
3.5 Type CHARACTER	44
Exercises	46

4 Eiffel Control Instructions	48
4.1 Structured programming	48
4.2 Compound instruction	50
4.3 Selection	52
4.4 Iteration	54
4.5 Multi-way selection	58
Exercises	60
5 Eiffel Routines	62
5.1 Routines which do not return a result	63
5.2 Local data	68
5.3 Routines which return a result	69
5.4 Creation routines	73
5.5 Recursive routines	75
5.6 Once routines	77
5.7 External routines	79
Exercises	79
6 Classes, Objects and Instantiation	82
6.1 Classes and Objects	82
6.2 Instantiation	85
6.3 Object destruction	87
6.4 Data abstraction and feature visibility	88
6.5 Issues of assignment and equality	91
6.6 Cloning objects	95
Exercises	95
7 Classes STRING and ARRAY	97
7.1 Class STRING	97
7.2 String comparison	97
7.3 String creation and assignment	98
7.4 String manipulation	100
7.5 Class ARRAY	102
7.6 Array creation	103
7.7 Array insertion and retrieval	104
7.8 Dynamic array operations	107
7.9 Sorting an array	107
7.10 Manifest arrays	110
7.11 Matrices and n -dimensional arrays	111
7.12 Mapping an n -dimensional array on to a vector	118
7.13 Comparing and copying arrays	120

Exercises	120
8 Assertions and Software Correctness	121
8.1 Preconditions and postconditions	121
8.2 Class invariants	126
8.3 Case study: class CLOCK	127
8.4 The check instruction	133
8.5 Loop variant and invariant	134
8.6 Exception handling	137
8.7 The debug instruction	141
8.8 Trace	142
Exercises	143
9 Inheritance	144
9.1 Inheritance for specialisation	144
9.2 Single inheritance in Eiffel	147
9.3 Inheriting for reuse	149
9.4 Polymorphism	153
9.5 Conformance	153
9.6 Reverse assignment attempt	155
9.7 When to inherit	158
Exercises	162
10 Generic Classes	164
10.1 Container classes	164
10.2 A generic class: LINKED_LIST	165
10.3 Non generic solutions to writing container classes	168
10.4 Developing generic classes	170
10.5 Constrained genericity	175
10.6 Genericity in Eiffel: a summary	180
Exercises	180
11 Adapting Inherited Classes	182
11.1 New-exports in Eiffel	183
11.2 Renaming features	187
11.3 Redefining features	188
11.4 Renaming and redefinition compared	190
11.5 Feature redeclaration: stacks, ordered lists and queues	193
11.6 Redeclaring assertions	202
11.7 Covariance	203
Exercises	207

12 Abstract Classes	209
12.1 Using an Eiffel deferred class: COMPARABLE	209
12.2 Undefined an effective feature	212
12.3 Simulating abstract classes	213
12.4 Case study: deferred class PRINTABLE	213
12.5 Case study: class CHESS_PIECE	215
Exercises	221
13 Multiple and Repeated Inheritance	222
13.1 Multiple inheritance	222
13.2 Repeated inheritance	229
13.3 Indirect repeated inheritance	230
13.4 Selecting a feature	233
13.5 Direct repeated inheritance	235
Exercises	238
14 Case Studies	240
14.1 Solving the eight queens problem	240
14.2 Implementing a back propagating neural network	253
Exercises	273
Appendix 1. Reserved words	275
Appendix 2. Special symbols	276
Appendix 3. Special characters	277
Appendix 4. Using Eiffel/S	278
Bibliography	281
Index	282

Preface

This year Eiffel celebrates its tenth anniversary, and it is also some seven years since the publication of Bertrand Meyer's now classic study, *Object-Oriented Software Construction*. In the last two years it has begun to achieve wider recognition both for teaching and for the production of quality software. There are now at least four vendors supplying a range of Eiffel compilers, and a number of student texts have recently appeared. In the author's view there is currently no better language for learning about object-oriented programming.

This text aims to provide an accessible introduction to programming with Eiffel. It is intended principally for those who have at least some basic knowledge of programming in an imperative language such as Pascal or C. It is also suitable for those who have some knowledge of object-orientation, and wish to find out about the Eiffel language. The emphasis of the text is very much on practical examples, which the reader is encouraged to try out.

Chapter 1 provides a general introduction to object-oriented principles. It also includes an overview of the Eiffel language.

Chapters 2, 3, 4 and 5 take a more bottom up approach, and cover the essentials of Eiffel as a programming language; in these chapters, ideas of object-orientation have a secondary role; advanced readers should be able to use these chapters largely for reference purposes.

Chapter 6 returns to object orientation, and deals with instantiation of objects from classes, and issues of object comparison and assignment. Chapter 7 provides an introduction to library support for strings and arrays, which in Eiffel are not part of the language. Chapters 8 to 13 cover advanced features, and the material in these chapters is more challenging: chapter 8 deals with assertions and design by contract; chapters 9, 11 and 13 cover inheritance; chapter 10 deals with generic classes; chapter 12 deals with abstract classes.

The final chapter consists of two case studies, which are intended to show how bigger problems may be solved using object-oriented design. The first, which builds on classes developed in earlier chapters, provides a solution to the eight queens problem; the second designs and implements a back propagating neural network.

The text takes an informal approach to design. No particular methodology or notation is pushed. The author broadly shares Bertrand Meyer's scepticism about language independent design methodologies: ultimately somebody has to write some code which runs on a real machine. If a language is sufficiently high-level to allow us to use it to design and to reason about design, as is Eiffel, then the design and implementation process may be seamless: design in Eiffel and let the compiler write the C code!

The text does emphasise that design for reuse is difficult, and that object-oriented designers should not expect to get the best solution first time. An object-oriented approach encourages us to model an application; model-building is itself a learning process which helps us to understand the application better, and hence to build a better model. The author never tires of telling his students about the experiences of the creators of Smalltalk, who, having completed the project, were told to throw it away and do it again, a luxury which, unfortunately, is experienced by few software developers. More experienced readers can probably think of many current software products which could have benefitted from such a practice. The examples in the text are, therefore, not presented as final solutions. If they stimulate thought, and provoke readers to produce better designs, this will be regarded as a sign of success. It is to be hoped that many readers will come to share the view of the author, and of others in the growing Eiffel community, that programming with Eiffel is challenging, but also very satisfying.

Some of the longer examples and other supporting material may be accessed via the internet: <http://www.doc.mmu.ac.uk/STAFF/J.Tyrrell>

My thanks to numerous students taught using early drafts of this text, whose comments were usually honest, and always useful. I must also thank the following: Michael O'Docherty of the PEVE unit at the Victoria University of Manchester, an indefatigable Eiffel enthusiast, for giving both encouragement and criticism; Roger Browne of Everything Eiffel, for his readiness to share his encyclopaedic knowledge of the Eiffel language; my colleague, Pamela Quick, for comments on some of the introductory material; my colleague Zuhair Bandar, without whom the neural networks case study would never have been written; Stuart Kent at Brighton University, for many constructive comments on the draft of the text; Malcolm Stewart of Macmillan Press, for his encouragement, patience and good humour. Finally, my special thanks to Inger Marie, for her unfailing support throughout the period when this was being written.

A.J. Tyrrell