

Contents

Preface: Meeting the challenge of software quality (<i>in progress</i>)	vii
THOUGHT AND EXPRESSION	viii
OLD, NEW AND OUT	viii
THE SIGNAL AND THE NOISE	ix
LANGUAGE LEVEL	x
THE FUTURE OF EIFFEL	x
MISSING ELEMENTS	xi
ACKNOWLEDGMENTS	xi
Preface to the third edition	xii
NON-CLASSICAL NUMBER THEORY	xii
MIGRATION AIDS	xii
About the status of Eiffel	xiii
About the language description	xv
TYPES OF DESCRIPTION, LEVELS OF DISCOURSE	xv
FORMALITY	xvi
ORDER OF PRESENTATION	xvii
PARAGRAPH TYPES AND ROAD SIGNS	xviii
CROSS-REFERENCES AND SHORTCUTS	xix
DESCRIBING A CONSTRUCT	xxi
COMMENTS, WARNINGS AND ADVICE	xxii
GRAPHICAL REPRESENTATIONS	xxii
THE CAKE AND ITS ICING	xxiii
Contents	xxv
PART I: INVITATION TO EIFFEL	1
1 An Eiffel tutorial	3
1.1 OVERVIEW	3
1.2 GENERAL PROPERTIES	4
1.3 THE SOFTWARE PROCESS IN EIFFEL	6
Clusters and the cluster model	7
Seamlessness and reversibility	8
Generalization and reuse	8
Constant availability	9
Compilation technology	9
Quality and functionality	9
1.4 HELLO WORLD	10
1.5 THE STATIC PICTURE: SYSTEM ORGANIZATION	11

	Systems	12
	Classes	12
	Class relations	13
	The global inheritance structure	14
	Clusters	14
	External software	15
1.6	THE DYNAMIC STRUCTURE: EXECUTION MODEL	15
	Objects, fields, values and references	16
	Features	16
	A simple class	18
	Creating and initializing objects	19
	Entities	21
	Calls	22
	Infix and prefix notation	23
	Type declaration	24
	Type categories	24
	Basic operations	26
	Deep operations and persistence	27
	Memory management	28
	Information hiding and the call rule	29
	Execution scenario	30
	Abstraction	31
1.7	GENERICITY	32
1.8	DESIGN BY CONTRACT, ASSERTIONS, EXCEPTIONS	34
	Design by Contract basics	35
	Expressing assertions	35
	Using assertions for built-in reliability	38
	Run-time assertion monitoring	38
	The short form of a class	40
	Exception handling	42
	Other applications of Design by Contract	45
1.9	THE INHERITANCE MECHANISM	45
	Basic inheritance structure	45
	Redefinition	46
	Polymorphism	47
	Dynamic binding	49
	Deferred features and classes	51
	Applications of deferred classes	54
	Structural property classes	56
	Multiple inheritance and feature renaming	57
	Inheritance and contracts	59
	Join and uneffecting	62
	Changing the export status	63
	Flat and flat-short forms	64
	Repeated inheritance and selection	65
	Constrained genericity	68
	Assignment attempt	69
	Covariance and anchored declarations	70
1.10	OTHER IMPORTANT MECHANISMS	73
	Once routines, shared objects, smart initialization and on-demand execution	73
	Constant attributes	75
	Instructions	75

	Lexical conventions	79
1.11	CONCURRENCY AND FURTHER DEVELOPMENTS	79
	SCOOP	79
	Other developments	81
PART II: LANGUAGE DESCRIPTION		83
2	Syntax, validity and semantics	85
2.1	OVERVIEW	85
2.2	SYNTAX: COMPONENTS, SPECIMENS, CONSTRUCTS	85
2.3	TERMINALS, NON-TERMINALS AND TOKENS	87
2.4	THE LEXICAL LEVEL	87
2.5	PRODUCTIONS	88
	Aggregate productions	89
	Choice productions	90
	Repetition productions	90
	Using recursive productions	92
	One production per non-terminal	93
	Non-production syntax rules	93
2.6	REPRESENTING TERMINALS	94
2.7	VALIDITY	96
2.8	INTERPRETING THE CONSTRAINTS	97
2.9	SEMANTICS	98
2.10	CORRECTNESS	99
2.11	TWO-TIER DEFINITION AND UNFOLDED FORMS	99
2.12	THE CONTEXT OF EXECUTING SYSTEMS	101
2.13	TEXTUAL CONVENTIONS	101
3	The architecture of Eiffel software	105
3.1	OVERVIEW	105
3.2	CLASSES	106
3.3	CLASS TEXTS AND CLASS NAMES	107
3.4	CLUSTERS	107
3.5	SYSTEMS	110
4	Classes	115
4.1	OVERVIEW	115
4.2	OBJECTS	115
4.3	FEATURES	116
4.4	USE OF CLASSES	116
4.5	THE CURRENT CLASS	117
4.6	CLASS TEXT STRUCTURE	117
4.7	PARTS OF A CLASS TEXT	119
4.8	ANNOTATING A CLASS	122
4.9	CLASS HEADER	124
	Deferred classes	125
	Expanded classes	126
	Validity of a class header	126
4.10	FORMAL GENERIC PARAMETERS	127
4.11	OBSOLETE MARK	128
5	Features	131
5.1	OVERVIEW	131

5.2	THE ROLE OF FEATURES	131
5.3	FEATURE CATEGORIES	132
5.4	IMMEDIATE AND INHERITED FEATURES	133
5.5	FEATURES PART: EXAMPLE	134
5.6	GRAPHICAL REPRESENTATION	136
5.7	FEATURES PART: SYNTAX	137
5.8	FORMS OF FEATURE	138
5.9	FEATURE DECLARATIONS: EXAMPLES	139
5.10	FEATURE DECLARATIONS: SYNTAX	140
5.11	FEATURE BODIES	143
5.12	HOW TO RECOGNIZE FEATURES	145
5.13	THE SIGNATURE OF A FEATURE	148
5.14	FEATURE NAME	150
5.15	OPERATOR FEATURES	154
5.16	ASSIGNER PROCEDURES	155
5.17	BRACKET FEATURE	157
5.18	SYNONYMS AND MULTIPLE DECLARATION	158
5.19	VALIDITY OF FEATURE DECLARATIONS	160
5.20	SCOPE OF NAMES	163
5.21	OBSOLETE FEATURES	163
5.22	NO IN-CLASS OVERLOADING	166
6	The inheritance relation	167
6.1	OVERVIEW	167
6.2	AN INHERITANCE PART	167
6.3	FORM OF THE INHERITANCE PART	168
6.4	GRAPHICAL CONVENTION	171
6.5	RELATIONS INDUCED BY INHERITANCE	171
6.6	<i>ANY</i>	172
6.7	<i>NONE</i>	175
6.6	PROHIBITING CYCLES	175
6.7	ADAPTING INHERITED FEATURES	177
6.8	NON-CONFORMING INHERITANCE	178
6.9	RENAMING	180
6.10	FEATURES AND THEIR NAMES	182
6.11	INDEPENDENCE OF INHERITANCE AND EXPANSION	186
7	Clients and exports	187
7.1	OVERVIEW	187
7.2	ENTITIES	187
7.3	CONVENTIONS	188
7.4	SIMPLE CLIENTS	189
7.5	EXPANDED CLIENTS	192
7.6	GENERIC CLIENTS	194
7.7	EXPORT CONTROLS AND INFORMATION HIDING	196
	Restricting exports	197
	Exporting to oneself	198
	Exporting to descendants	199
	Making a feature secret	199
	Adapting the export status of inherited features	200
	Expanding or restricting the export status	202

	The export status of features	202
	Rules on setting the export status	203
7.8	-DOCUMENTING THE CLIENT INTERFACE OF A CLASS	207
	Selecting features	207
	Contract views	208
8	Routines	213
8.1	OVERVIEW	213
8.2	ROUTINE DECLARATION	213
8.3	FORMAL ARGUMENTS	215
8.4	USING A VARIABLE NUMBER OF ARGUMENTS	217
8.5	ROUTINE BODY	218
8.6	LOCAL VARIABLES AND <i>RESULT</i>	221
8.7	EXTERNALS	223
8.8	TYPES OF INSTRUCTIONS	224
9	Correctness and contracts	225
9.1	OVERVIEW	225
9.2	WHY ASSERTIONS?	225
9.3	GRAPHICAL CONVENTION	227
9.4	USES OF ASSERTIONS	228
9.5	FORM OF ASSERTIONS	228
9.6	UNFOLDING ASSERTIONS UNDER INHERITANCE	231
9.7	ASSERTIONS ON INDIVIDUAL FEATURES	231
	Preconditions and postconditions	231
	The contract of a routine	232
	Constraints on routine assertions	233
	“Old” expression	235
	“Only” clause	237
9.8	CLASS INVARIANTS	240
9.9	THE CONSISTENCY OF A CLASS	242
9.10	CHECK INSTRUCTIONS	243
9.11	LOOP INVARIANTS AND VARIANTS	245
9.12	THE CORRECTNESS OF A CLASS	248
9.13	RULES OF RUN-TIME ASSERTION MONITORING	249
	Associated boolean expression	249
	Assertion monitoring	250
	Levels of assertion monitoring	252
	Invariant and qualified calls	253
10	Feature adaptation	255
10.1	OVERVIEW	255
10.2	TERMINOLOGY: REDECLARATION, REDEFINITION, EFFECTING	255
10.3	REDECLARING INHERITED FEATURES: WHY AND HOW	256
10.4	FEATURE ADAPTATION CLAUSES	257
10.5	WHY REDEFINE?	259
10.6	REDEFINITION EXAMPLES	260
10.7	THE REDEFINITION CLAUSE	261
10.8	REDEFINITION IN THE SOFTWARE PROCESS	262
10.9	CHANGING THE SIGNATURE	263
10.10	THE NEED FOR ANCHORED DECLARATIONS	265
10.11	DEFERRED FEATURES	266

10.12	DEFERRED CLASSES FOR DESCRIBING ABSTRACTIONS	267
10.13	DEFERRED CLASSES FOR SYSTEM DESIGN AND ANALYSIS	268
10.14	EFFECTING A DEFERRED FEATURE	270
10.15	PARTIALLY DEFERRED CLASSES AND PROGRAMMED ITERATION	271
10.16	REDECLARATION AND TYPING	274
10.17	REDECLARATION AND ASSERTIONS	277
10.18	RULES ON INHERITED ASSERTIONS	281
10.19	UNDEFINING A FEATURE	283
10.20	REDEFINITION AND EFFECTING	285
10.21	THE JOIN MECHANISM	286
10.22	MERGING EFFECTIVE FEATURES	288
10.23	NAME CLASHES	290
10.24	ADDING TO INHERITED BEHAVIOR: PRECURSOR	293
	The need for a precursor mechanism	293
	Precursor basics and examples	294
	Choosing between multiple precursors	295
	Precursor specification	296
10.25	REDEFINITION AND UNDEFINITION RULES	300
10.26	DEFERRED AND EFFECTIVE FEATURES AND CLASSES	303
10.27	ORIGIN AND SEED	305
10.28	REDECLARATION RULES	306
10.29	RULES ON JOINING FEATURES	309
11	Types	315
11.1	OVERVIEW	315
11.2	THE ROLE OF TYPES	315
11.3	WHERE TO USE TYPES	317
11.4	HOW TO DECLARE A TYPE	319
11.5	INSTANCES AND VALUES	321
11.6	INSTANCES OF A CLASS	323
11.7	BASE CLASS, BASE TYPE AND TYPE SEMANTICS	324
11.8	CLASS TYPES WITHOUT GENERICITY	326
11.9	EXPANDED TYPES	327
	Role of expanded types	327
	Defining expanded types	329
	Basic types	330
11.10	ANCHORED TYPES	331
	Anchored examples	332
	Anchoring to <i>Current</i>	332
	Anchoring to an expanded or generic	333
	Avoiding anchor cycles	335
		336
	Validity and semantics of anchored types	336
11.11	GUARANTEEING ATTACHMENT	339
11.12	STAND-ALONE TYPES	340
12	Genericity	341
12.1	OVERVIEW	341
12.2	GENERIC CLASSES	341
12.3	GENERIC CLASSES AND GENERIC DERIVATIONS	343

12.4	SELF-INITIALIZING FORMALS	344
12.5	CONSTRAINED AND UNCONSTRAINED GENERICITY	345
12.6	CONSTRAINED GENERICITY	346
12.7	RULES ON CONSTRAINED GENERICITY	348
12.8	CONSTRAINTS AND CREATION	352
12.9	RECURSIVE GENERIC CONSTRAINTS	354
12.10	SEMANTICS OF GENERIC TYPES	355
12.11	CURRENT TYPE, FEATURES OF A TYPE	357
12.12	APPLYING GENERICITY TO TYPES	358
12.13	THE CASE OF MULTIPLE CONSTRAINTS	359
13	Tuples	363
13.1	OVERVIEW	363
13.2	TUPLES IN A NUTSHELL	363
13.3	USING TUPLE TYPES AND TUPLES	364
13.4	ANONYMOUS CLASSES	367
13.5	CONFORMANCE ===== TO BE REWRITTEN	370
13.6	MULTIPLE RESULTS AND VARIABLE NUMBERS OF ARGUMENTS	370
	Emulating multiple results	371
	Emulating a variable number of arguments	372
13.7	TUPLES AS ARRAYS ===== TO BE REWRITTEN	372
14	Conformance	375
14.1	OVERVIEW	375
14.2	CONVERTIBILITY AND COMPATIBILITY	376
14.3	APPLICATIONS OF CONFORMANCE	377
14.4	EXPRESSION AND SIGNATURE CONFORMANCE	378
14.5	DIRECT AND INDIRECT CONFORMANCE	379
14.6	CONFORMANCE TO A NON-GENERIC REFERENCE TYPE	381
14.7	GENERICALLY DERIVED REFERENCE TYPES	382
14.8	FORMAL GENERIC PARAMETER CONFORMANCE	385
14.9	EXPANDED TYPE CONFORMANCE	386
14.10	TUPLE TYPE CONFORMANCE	388
14.11	ANCHORED TYPE CONFORMANCE	390
15	Convertibility	391
15.1	OVERVIEW	391
15.2	WHY IMPLICIT CONVERSION?	391
15.3	CONVERSION BASICS AND EXAMPLES	392
15.4	CONVERSION QUERIES	395
15.5	USING CONVERSIONS PROPERLY	398
15.6	CONVERSION PRINCIPLES	400
15.7	CONVERSION SYNTAX AND VALIDITY	402
15.8	SEMANTICS OF CONVERSION	407
15.9	CONVERTING AN EXPRESSION EXPLICITLY	408
15.10	EXPRESSION CONVERTIBILITY: THE ROLE OF PRECONDITIONS	412
15.11	MULTIPLE CONVERSION TYPES	417
15.12	MIXED-TYPE EXPRESSIONS: TARGET CONVERSION	419
	Using target conversion	420
	Validity of target conversion	421
	Target conversion: a discussion	422

16 Repeated inheritance	425
16.1 OVERVIEW	425
16.2 CASES OF REPEATED INHERITANCE	426
16.3 THE TWO QUESTIONS OF REPEATED INHERITANCE	427
16.4 SHARING AND REPLICATION	428
16.5 THE CASE OF REDECLARED FEATURES	434
16.6 THE CASE OF ATTRIBUTES	440
16.7 THE CASE OF CONFLICTING GENERIC DERIVATIONS	442
16.8 KEEPING THE ORIGINAL VERSION OF A REDEFINED FEATURE	443
16.9 USING REPLICATION: COUNTERS AND ITERATION	445
16.10 THE SEMANTICS OF REPLICATION	449
16.11 RETAINING VICTORS FROM ALTERNATIVE BRANCHES	452
16.12 THE NEED FOR SELECT	455
16.13 THE REPEATED INHERITANCE CONSISTENCY CONSTRAINT	455
16.14 THE INHERITED FEATURES OF A CLASS	461
17 Control structures	469
17.1 OVERVIEW	469
17.2 COMPOUND	469
17.3 CONDITIONAL	472
17.4 MULTI-BRANCH CHOICE	474
17.5 OBJECT TEST	483
17.6 USING SELECTION INSTRUCTIONS PROPERLY	483
17.7 LOOP	486
Loop structure and properties	486
Loop semantics	488
Ensuring non-void references in a loop	489
17.8 THE DEBUG INSTRUCTION	489
18 Attributes	491
18.1 OVERVIEW	491
18.2 GRAPHICAL REPRESENTATION	491
18.3 VARIABLE ATTRIBUTES	492
18.4 ATTRIBUTES IN FULL FORM	492
18.5 CONSTANT ATTRIBUTES	493
18.6 CONSTANT ATTRIBUTES WITH MANIFEST VALUES	494
19 Objects, values and entities	497
19.1 OVERVIEW	497
19.2 OBJECTS AND THEIR TYPES	498
19.3 VALUES AND INSTANCES	498
19.4 BASIC TYPES	500
19.5 REFERENCE AND COPY SEMANTICS	500
19.6 COMPOSITE OBJECTS AND THEIR FIELDS	500
19.7 REFERENCE ATOMICITY	502
19.8 EXPRESSIONS AND ENTITIES	504
19.9 SEMANTICS: EVALUATING AND INITIALIZING ENTITIES	506
20 Creating objects	515
20.1 OVERVIEW	515
20.2 FORMS OF CREATION: AN OVERVIEW	516

20.3	BASIC FORM OF CREATION INSTRUCTIONS	517
20.4	OMITTING THE CREATION PROCEDURE	519
20.5	CREATORS AND INHERITANCE	526
20.6	USING AN EXPLICIT TYPE	527
	Specifying the creation type	527
	Choosing between types	528
	Creation and deferred classes	529
	Single choice and factory objects	529
20.7	RESTRICTING CREATION AVAILABILITY	531
20.8	THE CASE OF EXPANDED TYPES	534
20.9	CREATING INSTANCES OF FORMAL GENERICS	535
20.10	PRECONDITIONS OF CREATION PROCEDURES	538
20.11	CREATION SYNTAX AND VALIDITY	539
20.12	CREATION SEMANTICS	548
20.13	REMOTE CREATION	549
20.14	CREATION EXPRESSIONS AND ANONYMOUS OBJECTS	550
21	Comparing and duplicating objects	557
21.1	OVERVIEW	557
21.2	COPYING AN OBJECT	557
21.3	EQUALITY EXPRESSIONS	557
	Effect of a copy operation	563
	Specification of default copy	564
	Tuning copy semantics	565
21.4	CLONING AN OBJECT	567
	Using cloning	567
	Twin	568
	Specification of default cloning	568
	Cloning, types and factories	570
21.5	DEEP COPYING AND CLONING	571
21.6	OBJECT EQUALITY	572
21.7	DEEP EQUALITY	574
22	Attaching values to entities	579
22.1	OVERVIEW	579
22.2	ROLE OF REATTACHMENT OPERATIONS	580
22.3	FORMS OF UNCONDITIONAL REATTACHMENT	580
22.4	SYNTAX AND VALIDITY OF ASSIGNMENT	581
22.5	THE STATUS OF FORMAL ROUTINE ARGUMENTS	582
22.6	CONVERSIONS	583
22.7	SEMANTICS OF REATTACHMENT	585
22.8	AN EXAMPLE	593
22.9	ABOUT REATTACHMENT	595
22.10	EFFECT ON GENERIC PROGRAMMING	596
22.11	POLYMORPHISM	598
22.12	ASSIGNER CALL	599
22.13	SEMI-STRICT OPERATORS	602
	The notion of strictness	603
	The need for semi-strict operators	603
	More on strictness	606
22.14	CONDITIONAL REATTACHMENT	607
	Limitations of unconditional reattachment	607

22.15	MEMORY MANAGEMENT	608
22.16	SEMANTICS OF EQUALITY	610
23	Feature call	613
23.1	OVERVIEW	613
23.2	PARTS OF A CALL	614
23.3	USES OF CALLS	615
23.4	UNIFORM ACCESS	616
23.5	OPERATOR AND BRACKET FORMS	616
23.6	COMPLEX TARGETS	617
23.7	CALL SYNTAX	618
23.8	COMPONENTS OF A CALL	620
23.9	NON-OBJECT CALLS	621
23.10	CLASS VALIDITY	623
	Export validity	624
	Argument validity	626
	Target validity and Void-Safe Eiffel	627
	Combining the rules	628
23.11	INTRODUCTION TO CALL SEMANTICS	628
23.12	DYNAMIC BINDING	630
23.13	THE IMPORTANCE OF BEING DYNAMIC	631
23.14	ONCE ROUTINES	633
	Once basics	633
	Once uses	633
	Predefined once keys	634
	Further once tuning	635
	Once routine semantics	636
23.15	ATTRIBUTES AND EXTERNALS	638
23.16	THE MACHINERY OF EXECUTING CALLS	639
	Scheme for a routine call	639
	Current object and routine	640
	Naming the current object	641
23.17	PRECISE CALL SEMANTICS	643
	Rule for non-once routines	643
	General call semantics	643
23.18	CALLS AS EXPRESSIONS	647
24	Eradicating void calls	649
24.1	OVERVIEW	649
24.2	OVERALL SCHEME	650
24.3	THE OBJECT TEST	650
24.4	VOID TESTS	654
24.5	CERTIFIED ATTACHMENT PATTERNS	655
24.6	ATTACHED EXPRESSIONS	656
25	Typing-related properties	657
25.1	OVERVIEW	657
25.2	SYNTAX VARIANTS	658
25.3	BASIC CONCEPTS	659
25.4		660
25.5	SYSTEM-LEVEL VALIDITY	661
25.6	VIOLATING SYSTEM VALIDITY	663

25.7	NOTES ON THE TYPE POLICY	665
25.8	WHY DISTINGUISH?	669
25.9	A LOOK AT THE DYNAMIC CLASS SET	670
25.10	THE CALL VALIDITY RULE	673
25.11	CREATION VALIDITY (SYSTEM-LEVEL)	678
26	Exception handling	681
26.1	OVERVIEW	681
26.2	WHAT IS AN EXCEPTION?	682
26.3	EXCEPTION HANDLING POLICY	683
26.4	RESCUE CLAUSES AND ORGANIZED PANIC	684
26.5	THE DEFAULT RESCUE	686
26.6	RETRY INSTRUCTIONS AND RESUMPTION	687
26.7	SYSTEM FAILURE AND THE EXCEPTION HISTORY TABLE	690
26.8	SYNTAX AND VALIDITY OF THE EXCEPTION CONSTRUCTS	693
26.9	EXCEPTION CORRECTNESS	694
26.10	SEMANTICS OF EXCEPTION HANDLING	694
26.11	EXCEPTION CORRECTNESS	699
26.12	FINE-TUNING THE MECHANISM	701
26.13	OVERVIEW	704
26.14	PLATFORM-DEPENDENT SIGNAL CODES	704
26.15	CLASS EXCEPTIONS	706
27	Agents, iteration and introspection	711
27.1	OVERVIEW	711
27.2	A QUICK PREVIEW	711
27.3	FROM CALLS TO AGENTS	715
	Feature calls and their operands	715
	Delaying calls	716
	Agents and their operands	717
27.4	AGENT TYPES	718
27.5	CALL AGENTS	720
	All-closed agents	721
	Keeping operands open	722
	The brace convention	724
	Omitting the argument list	724
	A summary of the possibilities	725
27.6	USING AGENTS	725
	GUI programming: establishing a direct connection to the Business Model	726
	Integrating a function	728
	Iteration examples	730
27.7	TWO ADVANCED EXAMPLES	735
	Error processing without the mess	735
	Once per object	736
27.8	USING INLINE AGENTS	738
27.9	ACCESSING FEATURE PROPERTIES	741
27.10	THE BASE CLASS AND TYPE	742
27.11	AGENT SYNTAX	743
	Syntax of call agents	744
	Syntax of inline agents	745
27.12	AGENT VALIDITY	746

Validity of call agents	746
Validity of inline agents	747
27.13 AGENT SEMANTICS	748
Call-agent equivalent of an inline agent	748
Open and closed operands	750
Type and value of an agent expression	751
28 Expressions	753
28.1 OVERVIEW	753
28.2 GENERAL FORM OF EXPRESSIONS	753
28.3 SUBEXPRESSIONS	756
28.4 PARENTHESIZED EXPRESSIONS	757
28.5 OPERATOR EXPRESSIONS	757
Operator expression basics	757
Operator expression syntax	758
Precedence and Parenthesized Form	758
Accounting for target conversion	762
Operator expression validity and semantics	763
28.6 SEMISTRICKT BOOLEAN OPERATORS	765
28.7 BRACKET EXPRESSIONS	769
28.8 THE EQUIVALENT DOT FORM	771
28.9 BOOLEAN EXPRESSIONS	772
28.10 ENTITIES	772
28.11 THE TYPE OF AN EXPRESSION	773
28.12 EXPRESSIONS AND THE SEMICOLON	775
29 Constants	777
29.1 OVERVIEW	777
29.2 GENERAL FORM OF CONSTANTS	777
29.3 FORCING A TYPE ON A CONSTANT	779
29.4 THE TYPE OF A CONSTANT	780
29.5 INTEGER CONSTANTS	782
29.6 REAL CONSTANTS	782
29.7 CHARACTER CONSTANTS	783
29.8 MANIFEST STRINGS	784
Basic manifest strings	786
Verbatim strings	788
Choosing between basic and verbatim manifest strings	794
“Once” string expressions	795
Run-time model for manifest strings	797
29.9 MANIFEST TUPLES	799
29.10 SEMANTICS OF CONSTANT ATTRIBUTES	803
30 Basic types	805
30.1 OVERVIEW	805
30.2 EXPANSION STATUS	805
30.3 BASIC CLASSES AND THEIR INHERITANCE STRUCTURE	807
30.4 BOOLEANS	809
30.5 CHARACTERS	809
30.6 INTEGERS	810
30.7 REALS	810
30.8 ADDRESSES	811

31	Interfacing with C, C++ and other environments	813
31.1	OVERVIEW: THE COMPONENT COMBINATOR	813
31.2	WHAT EIFFEL CAN DO WITH THE REST OF THE WORLD	814
31.3	WHEN TO USE EXTERNAL SOFTWARE	815
31.4	REGISTERED LANGUAGES AND THE ROLE OF C	817
31.5	BASICS OF EXTERNAL ROUTINES	818
31.6	EXECUTING AN EXTERNAL CALL	821
31.7	ARGUMENT AND RESULT TRANSMISSION	821
31.8	PASSING THE ADDRESS OF AN EIFFEL FEATURE	823
31.9	SPECIAL INTERFACE SUBLANGUAGES	827
31.10	GENERAL SUBLANGUAGE MECHANISMS	827
	Specifying an external routine signature	828
	Specifying external files	830
31.11	THE C INTERFACE SUBLANGUAGE	832
	Syntax specification	833
	Specifying C code inline	834
	Controlling the Eiffel-C type correspondence	836
31.12	THE C++ INTERFACE SUBLANGUAGE	837
	The syntax specification	838
	Conditions on C++ features	838
	Processing C++ features	839
	Extra argument	840
31.13	WRAPPING C++ CLASSES: LEGACY++	841
	The role of Legacy++	841
	Calling Legacy++	841
	Result of applying Legacy++	841
	Legacy++ limitations	842
	Legacy++ example	842
31.14	USING DYNAMIC LINKED LIBRARIES (DLLS)	845
	The static DLL sublanguage	846
31.15	DESC: CALLING A DLL ROUTINE DETERMINED AT RUN TIME	848
	DESC overview	849
	Creating a library object	849
	Creating a routine object	850
	Type codes	852
	Calling a routine	853
	Accessing the result of a function	853
	Consistency requirements and protection against errors	853
	Sharing and freeing	854
31.16	THE CECIL LIBRARY	855
	Cecil overview	855
	Cecil role and status	855
	Compiling for Cecil	856
	Avoiding abusive optimization	856
	Basic Cecil conventions	858
	Initializing the Eiffel 4 run-time	859
	Manipulating values of basic Eiffel types	860
	Manipulating Eiffel class types	861
	Accessing an Eiffel object	861
	Creating an Eiffel object	863
	Calling routines	864
	Requesting a non-existing routine	865

Accessing field objects	866
ISE Eiffel specifics	866
32 Lexical components	869
32.1 OVERVIEW	869
32.2 CHARACTER SETS	869
32.3 CHARACTER CATEGORIES	870
32.4 GENERAL FORMAT	871
32.5 BREAKS	871
32.6 COMMENTS	871
32.7 TEXT LAYOUT	875
32.8 LETTER CASE	876
32.9 TOKEN CATEGORIES	877
32.10 RESERVED WORDS	878
32.11 SPECIAL SYMBOLS	879
32.12 IDENTIFIERS	881
32.13 OPERATORS	882
32.14 CHARACTERS	884
32.15 STRINGS	888
32.16 INTEGERS	889
32.17 REAL NUMBERS	892
33 Concurrency (not done)	895
33.1 OVERVIEW	895
34 Style guidelines (not done)	897
34.1 OVERVIEW	897
34.2 LETTER CASE	897
34.3 CHOICE OF NAMES	898
34.4 GRAMMATICAL CATEGORIES FOR FEATURE NAMES	900
34.5 GROUPING FEATURES	901
34.6 HEADER COMMENTS	902
34.7 OTHER COMMENTS	904
34.8 EIFFEL NAMES IN COMMENTS	904
34.9 LAYOUT	905
34.10 OPTIONAL SEMICOLONS	909
34.11 LEXICAL CONVENTIONS	910
34.12 FONTS	910
34.13 GUIDELINES FOR ANNOTATING CLASSES	911
PART III: KERNEL LIBRARY CLASSES	915
35 Universal features and class <i>ANY</i>(in progress)	917
35.1 OVERVIEW	917
35.2 INPUT AND OUTPUT FEATURES	918
35.3 DUPLICATION AND COMPARISON ROUTINES	918
35.4 OBJECT PROPERTIES	919
35.5 PLATFORM-DEPENDENT FEATURES	919
35.6 OTHER UNIVERSAL FEATURES	920
36 Arrays and strings (not done)	921
36.1 OVERVIEW	921
36.2 REPRESENTATION	921

36.3	RESIZING	923
36.4	BASIC ARRAY HANDLING	924
36.5	COPYING AND COMPARING ARRAYS	925
36.6	MANIFEST ARRAYS	927
36.7	STRINGS	927
37	Persistence (<i>not done</i>)	931
37.1	OVERVIEW	931
37.2	CLASSES FOR PERSISTENCE	931
37.3	OBJECTS AND THEIR DEPENDENTS	932
37.4	RETRIEVAL, TYPING, AND THE ASSIGNMENT ATTEMPT	933
37.5	STORING AND RETRIEVING AN ENTIRE STRUCTURE	935
37.6	CLASS STORABLE	937
37.7	ENVIRONMENTS	940
37.8	OPENING AND CLOSING ENVIRONMENTS	941
37.9	RECORDING AND ACCESSING OBJECTS IN AN ENVIRONMENT	941
37.10	THE OBJECTS OF AN ENVIRONMENT	942
37.11	REQUESTING INFORMATION ABOUT ENVIRONMENTS	943
37.12	STORING ENVIRONMENTS	944
37.13	RETRIEVING AN ENVIRONMENT	945
37.14	AN ENVIRONMENT EXAMPLE	946
37.15	CLASS <i>ENVIRONMENT</i>	950
38	Input and output (<i>not done</i>)	955
38.1	OVERVIEW	955
38.2	PURPOSE OF THE CLASS	955
38.3	INPUT TECHNIQUES	956
38.4	CLASS STANDARD_FILES	958
A	ELKS: The Eiffel Library Kernel Standard	961
A.1	OVERVIEW	961
A.2	CONTENTS OF THIS STANDARD	961
A.3	COMPATIBILITY CONDITIONS	962
A.4	REQUIRED CLASSES	963
A.5	REQUIRED ANCESTRY LINKS	964
A.6	SHORT FORMS OF REQUIRED CLASSES	964
A.6.1	CLASS <i>ANY</i>	965
A.6.2	CLASS <i>TYPE</i>	966
A.6.3	CLASS <i>PART_COMPARABLE</i>	967
A.6.4	CLASS <i>COMPARABLE</i>	968
A.6.5	CLASS <i>HASHABLE</i>	969
A.6.6	CLASS <i>NUMERIC</i>	970
A.6.7	CLASS <i>INTERVAL</i>	971
A.6.8	CLASS <i>BOOLEAN</i>	972
A.6.9	CLASS <i>CHARACTER</i>	973
A.6.10	CLASS <i>INTEGER_GENERAL</i>	974
A.6.11	CLASS <i>INTEGER</i>	977
A.6.12	CLASS <i>INTEGER_8</i>	978
A.6.13	CLASS <i>INTEGER_16</i>	979
A.6.14	CLASS <i>INTEGER_64</i>	980
A.6.15	CLASS <i>REAL_GENERAL</i>	981
A.6.16	CLASS <i>REAL</i>	983

A.6.17 CLASS <i>TYPED_POINTER</i>	984
A.6.18 CLASS <i>POINTER</i>	985
A.6.19 CLASS <i>ARRAY</i>	986
A.6.20 CLASS <i>ANONYMOUS</i>	987
A.6.21 CLASS <i>STRING</i>	988
A.6.22 CLASS <i>STD_FILES</i>	991
A.6.23 CLASS <i>FILE</i>	992
A.6.24 CLASS <i>STORABLE</i>	995
A.6.25 CLASS <i>MEMORY</i>	996
A.6.26 CLASS <i>EXCEPTIONS</i>	997
A.6.27 CLASS <i>ARGUMENTS</i>	998
A.6.28 CLASS <i>PLATFORM</i>	999
A.6.29 CLASS <i>ONCE_MANAGER</i>	1000
A.6.30 CLASS <i>ROUTINE</i>	1001
A.6.31 CLASS <i>PROCEDURE</i>	1002
A.6.32 CLASS <i>FUNCTION</i>	1003
A.6.33 CLASS <i>PREDICATE</i>	1004

PART IV: THE LACE CONTROL LANGUAGE _____ **1005**

B Specifying systems in Lace (*in progress*) **1007**

B.1 OVERVIEW	1007
B.2 A SIMPLE EXAMPLE	1008
B.3 ON THE ROLE OF LACE	1009
B.4 A COMPLETE EXAMPLE	1010
B.5 BASIC CONVENTIONS	1012
B.6 BASICS OF CLUSTER CLAUSES	1014
B.7 STORING PROPERTIES WITH A CLUSTER	1016
B.8 EXCLUDING AND INCLUDING SOURCE FILES	1016
B.9 SPECIFYING OPTIONS	1018
B.10 SPECIFYING EXTERNAL ELEMENTS	1022
B.11 ONCE CONTROL	1023
B.12 GENERATION	1023
B.13 VISIBLE FEATURES	1024
B.14 COMPLETE LACE GRAMMAR	1027
B.15 LACE VALIDITY RULES	1027

PART V: COMPLEMENTS _____ **1029**

C On language design and evolution **1031**

C.1 SIMPLICITY, COMPLEXITY	1031
C.2 CONSISTENCY	1032
C.3 UNIQUENESS	1034
C.4 TOLERANCE AND DISCIPLINE	1036
C.5 METHODOLOGY	1037
C.6 MEA CULPA, MEA MAXIMA CULPA	1037
C.7 THE LANGUAGE AND THE LIBRARIES	1037
C.8 ON SYNTAX	1038
C.9 THE INVENTOR AND THE ASSEMBLER	1040
C.10 FROM THE INITIAL DESIGN TO THE ASYMPTOTE	1040
C.11 EXTENSIONS	1041
C.12 CHANGES	1042

C.13	THE POLITICS OF LANGUAGE EVOLUTION	1043
D	Credits (<i>in progress</i>)	1045
D.1	OVERVIEW	1045
D.2	IEFFEL SOFTWARE	1045
D.3	ECMA TC39-TG4	1046
D.4	IEFFEL COMMUNITY	1046
D.5	CREDIT FOR SPECIFIC INVENTIONS	1048
E	A brief history of Eiffel	1051
F	Language changes from the previous edition	1053
F.1	OVERVIEW	1053
F.2	REMOVED MECHANISMS	1054
F.3	BACKWARD COMPATIBILITY	1055
F.4	NEW CONSTRUCTS	1056
38.5	SEMANTIC EXTENSIONS AND CHANGES	1057
F.5	KERNEL LIBRARY CHANGES	1058
F.6	LEXICAL AND SYNTACTIC CHANGES	1060
F.7	CHANGES IN VALIDITY CONSTRAINTS AND CONFORMANCE RULES	1061
G	Changes from early versions	1067
G.1	OVERVIEW	1067
G.2	SCOPE OF THE CHANGES	1067
G.3	OLDER POST-OOSC-1 EXTENSIONS	1068
G.4	SEMICOLONS	1069
G.5	FEATURE ADAPTATION	1069
G.6	SPECIFYING EXPORT STATUS	1069
G.7	ADAPTING PRECONDITIONS AND POSTCONDITIONS	1070
G.8	REMOVING AMBIGUITIES IN REPEATED INHERITANCE	1071
G.9	RENAMING, REDEFINING, UNDEFINING AND JOINING	1071
G.10	SYNONYMS	1072
G.11	FROZEN FEATURES	1072
G.12	ANCHORING TO A FORMAL ARGUMENT	1072
G.13	CREATION SYNTAX	1073
G.14	UNIFORM SEMANTICS FOR DOT NOTATION	1073
G.15	MANIFEST ARRAYS	1074
G.16	DEFAULT RESCUE	1074
G.17	EXPANDED CLASSES	1074
G.18	SEMANTICS OF EXPANDED TYPES	1074
G.19	FREE INFIX AND PREFIX OPERATORS	1075
G.20	OBSOLETE CLAUSE	1075
G.21	RESERVED WORDS	1075
G.22	OTHER LEXICAL CHANGES	1076
H	Eiffel bibliography (<i>not done</i>)	1077
H.1	OVERVIEW	1077
H.2	BOOKS	1077
H.3	ISE MANUALS	1077
H.4	INFORMATION SOURCES	1082

PART VI: REFERENCE	1083
I.1 INTRODUCTION	1085
I.2 LANGUAGE SPECIFICATION	1085
J Syntax in alphabetical order	1129
J.1 OVERVIEW	1129
J.2 SYNTAX	1129
K Reserved words, special symbols, operator precedence	1139
K.1 OVERVIEW	1139
K.2 RESERVED WORDS	1139
K.3 SPECIAL SYMBOLS	1140
K.4 OPERATORS AND THEIR PRECEDENCE	1141
K.5 KEYWORDS AND SYMBOLS OF SPECIAL INTERFACE SUBLANGUAGES	1141
L Syntax diagrams (<i>not done</i>)	1143
PART VII: THE LANGUAGE STANDARD	1147
1.1 Overview	1
1.2 “The Standard”	1
1.3 Aspects covered	1
1.4 Aspects not covered	1
2.1 Definition	1
2.2 Compatibility and non-default options	2
2.3 Departure from the Standard	2
3.1 Earlier Eiffel language specifications	2
3.2 Eiffel Kernel Library	2
3.3 Floating point number representation	2
3.4 Character set: Unicode	3
3.5 Character set: ASCII	3
3.6 Phonetic alphabet	3
5.1 Standard elements	3
5.2 Normative elements	3
5.3 Rules on definitions	3
5.4 Use of defined terms	4
5.5 Unfolded forms	4
5.6 Language description	4
5.7 Validity: “if and only if” rules	4
6.1 Name of the language	5
6.2 Pronunciation	5
7.1 Design principles	5
7.2 Object-oriented design	6
7.3 Classes	6
7.4 Types	10
7.5 Assertions	11
7.6 Exceptions	13
7.7 Genericity	14
7.8 Inheritance	15
7.9 Polymorphism and dynamic binding	17
7.10 Combining genericity and inheritance	18

7.11	Deferred classes	20
7.12	Tuples and agents	21
7.13	Type- and void-safety	22
7.14	Putting a system together	23
PART VIII: BACK MATTER		161
Index		163

