Contents

Community resources vii
Dedication ix
Prefaces xi
Student_preface xiii
- Software everywhere xiii
- Casual and professional software development xiv
- Prior experience — or not xv
- Modern software technology xvi
- Object-oriented software construction xvii
- Formal methods xvii
- Learning by doing xviii
- From the consumer to the producer xviii
- Abstraction xix
- Destination: quality xx

Instructor_preface xxiii
- The challenges of a first course xxiii
- Outside-in: the inverted curriculum xxvii
  - The supporting software xxviii
  - From programming to software engineering xxx
  - Terminology xxx
- Technology choices xxxi
  - Object technology xxxi
  - Eiffel and Design by Contract xxxii
  - Why not Java? xxxiv
- How formal? xxxviii
- Other approaches xl
- Topics covered xli
- Acknowledgments xliii
- Bibliography xlv

Note to instructors: what to cover? xlvii
Contents xlix

PART I: BASICS 1

1 The industry of pure ideas 3
- 1.1 Their machines and ours 3
- 1.2 The overall setup 6
  - The tasks of computers 6
  - General organization 7
  - Information and data 8
  - Computers everywhere 9
  - The stored-program computer 10

1.3 Key concepts learned in this chapter 12
New vocabulary 13
1-E Exercises 13
2 Dealing with objects 15
2.1 A class text 15
2.2 Objects and calls 18
Editing the text 18
Running your first program 20
Dissecting the program 23
2.3 What is an object? 25
Objects you can and cannot kick 25
Features, commands and queries 26
Objects as machines 28
Objects: a definition 29
2.4 Features with arguments 30
2.5 Key concepts learned in this chapter 32
New vocabulary 32
2-E Exercises 32
3 Program structure basics 35
3.1 Instructions and expressions 35
3.2 Syntax and semantics 36
3.3 Programming languages, natural languages 37
3.4 Grammar, constructs and specimens 39
3.5 Nesting and the syntax structure 40
3.6 Abstract syntax trees 41
3.7 Tokens and the lexical structure 43
Token categories 43
Levels of language description 44
Identifiers 44
Breaks and indentation 45
3.8 Key concepts learned in this chapter 46
3-E Exercises 46
4 The interface of a class 47
4.1 Interfaces 47
4.2 Classes 49
4.3 Using a class 51
Defining what makes a good class 51
A mini-requirements document 52
Initial ideas for classes 52
What characterizes a metro line 53
4.4 Queries 55
How long is this line? 55
Experimenting with queries 56
The stations of a line 57
Properties of start and end lines 59
4.5 Commands 59
Building a line 59

4.6 Contracts 61
Preconditions 61
Contracts for debugging 64
Contracts for interface documentation 65
Postconditions 65
Class invariants **67**
Contracts: a definition 68
4.7 Key concepts learned in this chapter 68
4-E Exercises 69
5 Just Enough Logic 71
5.1 Boolean operations 72
Boolean values, variables, operators and expressions 72
Negation 73
Disjunction 74
Conjunction 75
Complex expressions 76
Truth assignment 77
Tautologies 78
Equivalence 79
De Morgan’s laws 81
Simplifying the notation 82
5.2 Implication 84
Definition 84
Relating to inference 85
Getting a practical feeling for implication 86
Reversing an implication 88
5.3 Semistrict boolean operators 89
Semistrict implication 94
5.4 Predicate calculus 94
Generalizing “or” and “and” 95
Precise definition: existentially quantified expression 96
Precise definition: universally quantified expression 97
The case of empty sets 99
5.5 Further reading 100
5.6 Key concepts learned in this chapter 101
New vocabulary 101
5-E Exercises 102
6 Creating objects and executing systems 107
6.1 Overall setup 108
6.2 Entities and objects 109
6.3 Void references 111
The initial state of a reference 111
The trouble with void references 112
Not every declaration should create an object 114
The role of void references 115
Calls in expressions: overcoming your fear of void 116
6.4 Creating simple objects 118
6.5 Creation procedures 122
6.6 Correctness of a creation instruction 126

6.7 Memory management and garbage collection 128
6.8 System execution 130
Starting it all 130
The root class, the system and the design process 130
Specifying the root 131
The current object and general relativity 132
The ubiquity of calls: operator aliases 134
Object-oriented programming is relative programming 135
6.9 Appendix: getting rid of void calls 136
6.10 Key concepts learned in this chapter 137
New vocabulary 138
6-E Exercises 138
7 Control structures 139
7.1 Problem-solving structures 139
7.2 The notion of algorithm 141
Example 141
Precision and explicitness: algorithms vs recipes 142
Properties of an algorithm 143
Algorithms vs programs 144
7.3 Control structure basics 146
7.4 Sequence (compound instruction) 147
Examples 147
Compound: syntax 149
Compound: semantics 150
Order overspecification 151
Compound: correctness 152
7.5 Loops 153
Loops as approximations 154
The loop strategy 155
Loop instruction: basic syntax 157
Including the invariant 158
Loop instruction: correctness 159
Loop termination and the halting problem 161
Animating a metro line 166
Understanding and verifying the loop 169
The cursor and where it will go 173
7.6 Conditional instructions 174
Conditional: an example 175
Conditional structure and variations 176
Conditional: syntax 180
Conditional: semantics 181
Conditional: correctness 181
7.7 The lower level: branching instructions 181
Conditional and unconditional branching 182
The **goto** instruction 183
Flowcharts 184
7.8 Goto elimination and structured programming 185
Goto harmful? 185
Avoiding the goto 187
Structured programming 188

The **goto** puts on a mask 189
7.9 Variations on basic control structures 191
Loop initialization 191
Other forms of loop 192
Multi-branch 195
7.10 An introduction to exception handling 200
The role of exceptions 200
A precise framework to discuss failures and exceptions 201
Retrying 202
Exception details 204
The try-catch style of exception handling 204
Two views of exceptions 204
7.11 Appendix: an example of goto removal 205
7.12 Further reading 207
7.13 Key concepts learned in this chapter 207
New vocabulary 208
7-E Exercises 208
8 Routines, functional abstraction and information hiding 211
8.1 Bottom-up and top-down reasoning 211
8.2 Routines as features 213
8.3 Encapsulating a functional abstraction 214
8.4 Anatomy of a routine declaration 215
Interface vs implementation 217
8.5 Information hiding 218
8.6 Procedures vs functions 219
8.7 Functional abstraction 220
8.8 Using routines 222
8.9 An application: proving the undecidability of the halting problem 223
8.10 Further reading 224
8.11 Key concepts learned in this chapter 225
New vocabulary 225
8-E Exercises 225
9 Variables, assignment and references 227
9.1 Assignment 228
Summing travel times 228
Local variables 231
Function results 234
Swapping two values 235
The power of assignment 235
9.2 Attributes 238
Fields, features, queries, functions, attributes 238
Assigning to an attribute 239
Information hiding: modifying fields 240
Information hiding: accessing fields 243
9.3 Kinds of feature 244
The client’s view 244
The supplier’s view 247
Setters and getters 248

9.4 Entities and variables 249
Basic definitions 249
Variable and constant attributes 250
9.5 Reference assignment 252
Building metro stops 252
Building a metro line 254
9.6 Programming with references 256
References as a modeling tool 256
Using references for building linked structures 256
Void references 258
Reversing a linked structure 259
Making lists explicit 262
Where to use reference operations? 263
Dynamic aliasing 265
9.7 Key concepts learned in this chapter 268
New vocabulary 269
Precise feature terminology 269
9-E Exercises 269
PART II: HOW THINGS WORK 271
10 Just enough hardware 273
10.1 Encoding data 273
The binary number system 274
Binary basics 275
Basic representations and addresses 276
Powers of two 277
From cherries to bytes 277
Computing with numbers 279
10.2 More on memory 283
Persistence 283
Transient memory 284
Varieties of persistent memory 284
Registers and the memory hierarchy 287
Virtual memory 288
10.3 Computer instructions 288
10.4 Moore’s “law” and the evolution of computers 290
10.5 Further reading 291
10.6 Key concepts learned in this chapter 292
New vocabulary 293
10-E Exercises 293
11 Describing syntax 295
11.1 The role of BNF 295
Languages and their grammars 296
BNF basics 297
Distinguishing language from metalanguage 299
11.2 Productions 300
Concatenation 300
Choice 301
Repetition 301
Rules on grammars 303

11.3 Using BNF 305
Applications of BNF 305
Language generated by a grammar 306
Recursive grammars 307
11.4 Describing abstract syntax 310
11.5 Turning a grammar into a parser 311
11.6 The lexical level and regular automata 311
Lexical constructs in BNF 311
Regular grammars 312
Finite automata 314
Context-free properties 316
11.7 Further reading 318
11.8 Key concepts learned in this chapter 318
New vocabulary 319
11-E Exercises 319
12 Programming languages and tools 321
12.1 programming language styles 322
Classification criteria 322
Functional programming and functional languages 324
Object-oriented languages 327
12.2 Compilation vs interpretation 330
Basic schemes 330
Combining compilation and interpretation 332
Virtual machines, bytecode and jitting 333
12.3 The essentials of a compiler 335
Compiler tasks 336
Fundamental data structures 337
Passes 337
The compiler as verification tool 338
Loading and linking 338
The runtime 339
Debuggers and execution tools 340
12.4 Verification and validation 341
12.5 Text, program and design editors 342
12.6 Configuration management 344
Varieties of configuration management 344
Build tools: from Make to automatic dependency analysis 345
Version control 347
12.7 Total project repositories 351
12.8 Browsing and documentation 352
12.9 Metrics 352
12.10 Integrated development environments 353
12.11 An IDE: EiffelStudio 353
Overall structure 354
Browsing and documentation 355
The melting ice technology 357
12.12 Key concepts introduced in this chapter 359
New vocabulary 360
12-E Exercises 360

PART III: ALGORITHMS AND DATA STRUCTURES 361
13 Fundamental data structures, genericity, and algorithm complexity 363
13.1 Static typing and genericity 363
Static typing 364
Static typing for container classes 364
Generic classes 365
Validity vs correctness 368
Classes vs types 369
Nesting generic derivations 370
13.2 Container operations 371
Queries 371
Commands 372
Standardizing feature names for basic operations 374
Automatic resizing 375
13.3 Estimating algorithm complexity 376
Measuring orders of magnitude 376
Mathematical basis 377
Making the best use of your lottery winnings 378
Abstract complexity in practice 379
Presenting data structures 379
13.4 Arrays 380
Bounds and indexes 381
Creating an array 382
Accessing and modifying array items 383
Bracket notation and assigner commands 384
Resizing an array 386
Using arrays 388
Performance of array operations 388
13.5 Tuples 389
13.6 Lists 391
Cursor queries 392
Cursor movement 395
Iterating over a list 396
Adding and removing items 398
13.7 Linked lists 400
Linked list basics 400
Insertion and removal 401
Reversing a linked list 403
Performance of linked list operations 406
13.8 Other list variants 408
Two-way lists 408
Abstraction and consequences 408
Arrayed lists 409
Multi-array lists 410
13.9 Hash tables 411
13.10 Dispensers 418
13.11 Stacks 420
Stack basics 420
Using stacks 421
Implementing stacks 424

13.12 Queues 428
13.13 Iterating on data structures 431
13.14 Other structures 432
13.15 Further reading 432
13.16 Key concepts learned in this chapter 433
New vocabulary 434
13-E Exercises 434
14 Recursion and trees 435
14.1 Basic examples 436
Recursive definitions 436
Recursively defined grammars 437
Recursively defined data structures 437
Recursively defined algorithms and routines 438
14.2 The tower of Hanoi 441
14.3 Recursion as a problem-solving strategy 446
14.4 Binary trees 447
A recursive routine on a recursive data structure 448
Children and parents 449
Recursive proofs 449
A binary tree of executions 450
More binary tree properties and terminology 451
Binary tree operations 452
Traversals 453
Binary search trees 454
Performance 455
Inserting, searching, deleting 456
14.5 Backtracking and alpha-beta 459
The plight of the shy tourist 459
Getting backtracking right 462
Backtracking and trees 463
Minimax 464
Alpha-beta 468
14.6 From loops to recursion 471
14.7 Making sense of recursion 473
Vicious circle? 473
Boutique cases of recursion 476
Keeping definitions non-creative 478
The bottom-up view of recursive definitions 479
Bottom-up interpretation of a construct definition 482
The towers, bottom-up 483
Grammars as recursively defined functions 484
14.8 Contracts for recursive routines 485
14.9 Implementation of recursive routines 486
A recursive scheme 487
Routines and their execution instances 487
Preserving and restoring the context 488
Using an explicit call stack 489
Recursion elimination essentials 491
Simplifying the iterative version 494
Tail recursion 496

Taking advantage of invertible functions 497
14.10 Key concepts learned in this chapter 500
New vocabulary 500
14-E Exercises 500
15 Devising and engineering an algorithm: Topological Sort 505
15.1 The problem 505
Example applications 506
Points in a plane 507
15.2 The basis for topological sort 509
Binary relations 509
Acyclic relations 510
Order relations 511
Order relations vs acyclic relations 512
Total orders 514
Acyclic relations have a topological sort 516
15.3 Practical considerations 517
Performance requirements 517
Class framework 518
Input and output 518
Overall form of the algorithm 519
Cycles in the constraints 520
Overall class organization 523
15.4 Basic algorithm 526
The loop 526
A “natural” choice of data structures 527
Performance analysis of the natural solution 528
Duplicating the information 529
Spicing up the class invariant 530
Numbering the elements 531
Basic operations 532
The candidates 533
The loop, final form 536
Initializations and their time performance 538
Putting everything together 541
15.5 Lessons 542
Interpretation vs compilation 542
Time-space tradeoffs 544
Algorithms vs systems and components 544
15.6 Key concepts learned in this chapter 545
New vocabulary 545
15.7 Appendix: terminology note on order relations 546
15-E Exercises 546
PART IV: OBJECT-ORIENTED TECHNIQUES 549
16 Inheritance 551
16.1 Taxis are vehicles 552
Inheriting features 552
Inheritance terms 554
Features from a higher authority 555
The flat view 556

16.2 Polymorphism 557
Definitions 558
Polymorphism is not conversion 559
Polymorphic data structures 560
16.3 Dynamic binding 562
16.4 Typing and inheritance 563
16.5 Deferred classes and features 565
16.6 Redefinition 570
16.7 Beyond information hiding 573
Beware of choices bearing many cases 574
16.8 A peek at the implementation 575
16.9 What happens to contracts? 580
Invariant accumulation 581
Precondition weakening and postcondition strengthening 582
Contracts in deferred classes 585
Contracts tame inheritance 586
16.10 Overall inheritance structure 586
16.11 Multiple inheritance 588
Using multiple inheritance 588
Renaming features 590
From multiple to repeated inheritance 592
16.12 Genericity plus inheritance 594
Polymorphic data structures 594
Constrained genericity 596
16.13 Uncovering the actual type 599
The object test 602
Assignment attempt 604
Using dynamic casts wisely 605
16.14 Reversing the structure: visitors and agents 606
The dirty little secret 606
The Visitor pattern 608
Improving on Visitor 613
16.15 Further reading 613
16.16 Key concepts learned in this chapter 614
New vocabulary 615
16-E Exercises 616
17 Operations as objects: agents and lambda calculus 619
17.1 Beyond the duality 619
17.2 Why objectify operations? 621
Four applications of agents 621
A world without agents 623
17.3 Agents for iteration 627
Basic iterating schemes 627
Iterating for predicate calculus 628
Agent types 629
A home for fundamental iterators 631
Writing an iterator 631
17.4 Agents for numerical programming 634

17.5 Open operands 636
Open arguments 636
Open targets 638
17.6 Lambda calculus 640
Operations on functions 640
Lambda expressions 641
Currying 643
Generalized currying 645
Currying in practice 645
The calculus 646
Lambda calculus and agents 651
17.7 Inline agents 652
17.8 Other language constructs 654
Agent-like mechanisms 655
Routines as arguments 656
Function pointers 656
Many Little Wrappers and nested classes 657
17.9 Further reading 658
17.10 Key concepts learned in this chapter 658
New vocabulary 659
17-E Exercises 660
18 Event-driven design 663
18.1 Event-driven GUI programming 664
Good old input 664
Modern interfaces 664
18.2 Terminology 666
Events, publishers and subscribers 666
Arguments and event types 668
Keeping the distinction clear 671
Contexts 673
18.3 Publish-subscribe requirements 674
Publishers and subscribers 674
The model and the view 675
Model-View-Controller 677
18.4 The observer pattern 678
About design patterns 678
Observer basics 679
The publisher side 679
The subscriber side 681
Publishing an event 684
Assessing the Observer pattern 684
18.5 Using agents: the event library 686
Basic API 686
Using event types 687
Event type implementation 689
18.6 Subscriber discipline 690
18.7 Software architecture lessons 691
Choosing the right abstractions 691
MVC revisited 692
The model as publisher 693

Invest then enjoy 694
Assessing software architectures 694
18.8 Further reading 695
18.9 Key concepts learned in this chapter 696
New vocabulary 697
18-E Exercises 697
PART V: TOWARDS SOFTWARE ENGINEERING 699
19 Introduction to software engineering 701
19.1 Basic definitions 702
19.2 The DIAMO view of software engineering 704
19.3 Components of quality 705
Process and product 705
Immediate product quality 707
Long-term product quality 708
Process quality 710
Tradeoffs 712
19.4 Major software development activities 712
19.5 Lifecycle models and agile development 714
The waterfall 714
The spiral model 715
The cluster model 716
Agile development 717
19.6 Requirements analysis 718
Products of the requirements phase 719
The IEEE standard 719
Scope of requirements 720
Obtaining requirements 720
The glossary 722
Machine properties and domain engineering 723
Fifteen properties of good requirements 724
19.7 Verification and validation 727
Varieties of quality assurance 728
Testing 728
Static techniques 732
19.8 Capability maturity models 735
CMMI scope 735
CMMI disciplines 736
Goals, practices and process areas 737
Two models 737
Assessment levels 738
19.9 Further reading 740
19.10 Key concepts learned in this chapter 742
New vocabulary 743
Acronym collection 743
19-E Exercises 743
PART VI: APPENDICES 745
A An introduction to Java (from material by Marco Piccioni) 747
A.1 Language background and style 747

A.2 Overall program structure 748
The Java Virtual Machine 748
Packages 748
Program execution 749
A.3 Basic object-oriented model 750
The Java type system 750
Classes and members 751
Information hiding 752
Static members 753
Abstract classes and interfaces 753
Overloading 754
Run-time model, object creation and initialization 755
Arrays 757
Exception handling 758
A.4 Inheritance and genericity 760
Inheritance 760
Redefinition 760
Polymorphism, dynamic binding and casts 761
Genericity 762
A.5 Further program structuring mechanisms 763
Conditional and branching instructions 763
Loops 765
A.6 Absent elements 766
Design by Contract 766
Multiple inheritance 766
Agents 766
A.7 Specific language features 767
Nested and anonymous classes 767
Type conversions 771
Enumerated types 771
Varargs 772
Annotations 772
A.8 Lexical and syntactic aspects 773
Keywords 774
Operators 774
A.9 Bibliography 774
B An introduction to C# (from material by Benjamin Morandi) 775
B.1 Language background and style 776
.NET, the CLI and language interoperability 776
The favorite son 777
B.2 Overall program structure 777
Classes and structs 777
Program execution 778
B.3 Basic object-oriented model 778
Static members and classes 778
Export status 779
Fields 779
Basic types 780
References and values 780
Constants 781
Methods 781

Overloading 782
Properties 782
Constructors 783
Destructors 784
Operators 785
Arrays and indexers 786
Genericity 788
Basic statements 788
Control structures 789
Exception handling 790
Delegates and events 791
B.4 Inheritance 794
Inheriting from a class 794
You may only specify one parent class, here K. 794
Abstract members and classes 794
Interfaces 795
Accessibility and inheritance 796
Overriding and dynamic binding 796
Inheritance and creation 798
Run-Time Type Identification 798
B.5 Further program structuring mechanisms 799
Namespaces 799
Extension methods 800
Attributes 801
B.6 Absent elements 802
B.7 Specific language features 803
Unsafe code 803
Enumeration types 803
Linq 804
B.8 Lexical aspects 804
B.9 Bibliography 804
C An introduction to C++ (from material by Nadia Polikarpova) 805
C.1 Language background and style 805
C.2 Overall program organization 806
C.3 Basic object-oriented model 808
Built-in types 808
Derived types 808
Combining derived type mechanisms 812
User-defined types 812
Classes 813
Information hiding 816
Scoping 817
Operators 818
Overloading 818
Static declarations 818
Object lifetime 819
Initialization 821
Exception handling 822
Templates 823
C.4 Inheritance 825
Overriding 825

Export status and inheritance 825
Precursor access 826
Static and dynamic binding 826
Pure virtual functions 827
Multiple inheritance 827
Inheritance and object creation 828
C.5 Further program structuring mechanisms 829
C.6 Absent elements 829
Contracts 829
Agents 830
Constrained genericity 830
Overall inheritance structure 831
C.7 Specific language features 831
Argument defaults 831
Nested classes 831
C.8 Libraries 831
C.9 Syntactic and lexical aspects 832
Instructions as expressions 832
Control structures 833
Assignment and assignment-like instructions 835
Expressions and operators 836
Identifiers 837
Literals 837
Keywords 838
C.10 Further reading 838
D From C++ to C 839
D.1 Absent elements 839
D.2 Language background and style 840
D.3 Further reading 842
E Using the EiffelStudio environment 843
E.1 Eiffelstudio basics 843
E.2 Setting up a project 844
E.3 Bringing up classes and views 845
E.4 Specifying a root class and creation procedure 845
E.5 Contract monitoring 846
E.6 Controlling execution and inspecting objects 846
E.7 Panic mode (not!) 846
E.8 To know more 846
Picture credits 847
Index 849