

contents

part one **Tour of Java**

chapter 1

primitive java

39

- 1.1 the general environment** 40
- 1.2 the first program** 41
 - 1.2.1 comments 41
 - 1.2.2 `main` 42
 - 1.2.3 terminal output 42
- 1.3 primitive types** 42
 - 1.3.1 the primitive types 42
 - 1.3.2 constants 43
 - 1.3.3 declaration and initialization of primitive types 43
 - 1.3.4 terminal input and output 44
- 1.4 basic operators** 44
 - 1.4.1 assignment operators 45
 - 1.4.2 binary arithmetic operators 46
 - 1.4.3 unary operators 46
 - 1.4.4 type conversions 46
- 1.5 conditional statements** 47
 - 1.5.1 relational and equality operators 47
 - 1.5.2 logical operators 48
 - 1.5.3 the `if` statement 49
 - 1.5.4 the `while` statement 50
 - 1.5.5 the `for` statement 50
 - 1.5.6 the `do` statement 51

1.5.7	<code>break</code> and <code>continue</code>	52
1.5.8	the <code>switch</code> statement	53
1.5.9	the conditional operator	53
1.6	methods	54
1.6.1	overloading of method names	55
1.6.2	storage classes	56
	summary	56
	key concepts	56
	common errors	58
	on the internet	59
	exercises	59
	references	61

chapter 2 reference types

63

2.1	what is a reference?	63
2.2	basics of objects and references	66
2.2.1	the dot operator (<code>.</code>)	66
2.2.2	declaration of objects	66
2.2.3	garbage collection	67
2.2.4	the meaning of <code>=</code>	68
2.2.5	parameter passing	69
2.2.6	the meaning of <code>==</code>	69
2.2.7	<i>no operator overloading for objects</i>	70
2.3	strings	71
2.3.1	basics of string manipulation	71
2.3.2	string concatenation	71
2.3.3	comparing strings	72
2.3.4	other <code>String</code> methods	72
2.3.5	converting other types to strings	73
2.4	arrays	73
2.4.1	declaration, assignment, and methods	74
2.4.2	dynamic array expansion	76
2.4.3	<code>ArrayList</code>	78
2.4.4	multidimensional arrays	81
2.4.5	command-line arguments	81
2.4.6	<i>enhanced for loop</i>	82

- 2.5 exception handling** 83
 - 2.5.1 processing exceptions 84
 - 2.5.2 the finally clause 84
 - 2.5.3 common exceptions 85
 - 2.5.4 the throw and throws clauses 87
- 2.6 input and output** 87
 - 2.6.1 basic stream operations 88
 - 2.6.2 the Scanner type 89
 - 2.6.3 sequential files 92
- summary** 95
- key concepts** 96
- common errors** 97
- on the internet** 98
- exercises** 98
- references** 104

chapter 3 objects and classes

105

- 3.1 what is object-oriented programming?** 105
- 3.2 a simple example** 107
- 3.3 javadoc** 109
- 3.4 basic methods** 112
 - 3.4.1 constructors 112
 - 3.4.2 mutators and accessors 112
 - 3.4.3 output and toString 114
 - 3.4.4 equals 114
 - 3.4.5 main 114
- 3.5 example: using java.math.BigInteger** 114
- 3.6 additional constructs** 115
 - 3.6.1 the this reference 117
 - 3.6.2 the this shorthand for constructors 118
 - 3.6.3 the instanceof operator 118
 - 3.6.4 instance members versus static members 119
 - 3.6.5 static fields and methods 119
 - 3.6.6 static initializers 122
- 3.7 example: implementing a BigInteger class** 122

3.8 packages	126
3.8.1 the <code>import</code> directive	127
3.8.2 the <code>package</code> statement	129
3.8.3 the <code>CLASSPATH</code> environment variable	130
3.8.4 package visibility rules	131
3.9 a design pattern: composite (pair)	131
summary	132
key concepts	133
common errors	136
on the internet	136
exercises	137
references	143

chapter 4 inheritance 145

4.1 what is inheritance?	146
4.1.1 creating new classes	146
4.1.2 type compatibility	151
4.1.3 dynamic dispatch and polymorphism	152
4.1.4 inheritance hierarchies	153
4.1.5 visibility rules	153
4.1.6 the constructor and <code>super</code>	154
4.1.7 <code>final</code> methods and classes	155
4.1.8 overriding a method	157
4.1.9 type compatibility revisited	157
4.1.10 compatibility of array types	160
4.1.11 covariant return types	160
4.2 designing hierarchies	161
4.2.1 abstract methods and classes	162
4.2.2 designing for the future	166
4.3 multiple inheritance	167
4.4 the interface	170
4.4.1 specifying an interface	170
4.4.2 implementing an interface	171
4.4.3 multiple interfaces	171
4.4.4 interfaces are abstract classes	172

4.5	fundamental inheritance in java	172
4.5.1	the Object class	172
4.5.2	the hierarchy of exceptions	173
4.5.3	i/o: the decorator pattern	174
4.6	implementing generic components using inheritance	178
4.6.1	using Object for genericity	178
4.6.2	wrappers for primitive types	179
4.6.3	autoboxing/unboxing	181
4.6.4	adapters: changing an interface	182
4.6.5	using interface types for genericity	183
4.7	implementing generic components using java 5 generics	186
4.7.1	simple generic classes and interfaces	186
4.7.2	wildcards with bounds	187
4.7.3	generic static methods	188
4.7.4	type bounds	189
4.7.5	type erasure	190
4.7.6	restrictions on generics	190
4.8	the functor (function objects)	193
4.8.1	nested classes	197
4.8.2	local classes	197
4.8.3	anonymous classes	199
4.8.4	nested classes and generics	200
4.9	dynamic dispatch details	200
	summary	204
	key concepts	204
	common errors	207
	on the internet	207
	exercises	209
	references	219

part two **Algorithms and Building Blocks**

chapter 5 **algorithm analysis** **223**

5.1	what is algorithm analysis?	224
5.2	examples of algorithm running times	228

5.3	the maximum contiguous subsequence sum problem	229
5.3.1	the obvious $O(N^3)$ algorithm	230
5.3.2	an improved $O(N^2)$ algorithm	233
5.3.3	a linear algorithm	233
5.4	general big-oh rules	237
5.5	the logarithm	241
5.6	static searching problem	243
5.6.1	sequential search	243
5.6.2	binary search	244
5.6.3	interpolation search	247
5.7	checking an algorithm analysis	248
5.8	limitations of big-oh analysis	249
	summary	250
	key concepts	250
	common errors	251
	on the internet	252
	exercises	252
	references	263

chapter 6 the collections api

265

6.1	introduction	266
6.2	the iterator pattern	267
6.2.1	basic iterator design	268
6.2.2	inheritance-based iterators and factories	270
6.3	collections api: containers and iterators	272
6.3.1	the Collection interface	273
6.3.2	Iterator interface	276
6.4	generic algorithms	278
6.4.1	Comparator function objects	279
6.4.2	the Collections class	279
6.4.3	binary search	282
6.4.4	sorting	282
6.5	the List interface	284
6.5.1	the ListIterator interface	285
6.5.2	LinkedList class	287

- 6.5.3 running time for Lists 289
- 6.5.4 removing from and adding to the middle of a List 292
- 6.6 stacks and queues 294**
 - 6.6.1 stacks 294
 - 6.6.2 stacks and computer languages 295
 - 6.6.3 queues 296
 - 6.6.4 stacks and queues in the collections api 297
- 6.7 sets 297**
 - 6.7.1 the TreeSet class 299
 - 6.7.2 the HashSet class 300
- 6.8 maps 304**
- 6.9 priority queues 310**
- 6.10 views in the collections api 313**
 - 6.10.1 the subList method for Lists 313
 - 6.10.2 the headSet, subSet, and tailSet methods for SortedSets 313
- summary 314**
- key concepts 315**
- common errors 316**
- on the internet 317**
- exercises 317**
- references 326**

chapter 7 recursion

327

- 7.1 what is recursion? 328**
- 7.2 background: proofs by mathematical induction 329**
- 7.3 basic recursion 331**
 - 7.3.1 printing numbers in any base 333
 - 7.3.2 why it works 335
 - 7.3.3 how it works 336
 - 7.3.4 too much recursion can be dangerous 338
 - 7.3.5 preview of trees 339
 - 7.3.6 additional examples 340
- 7.4 numerical applications 345**
 - 7.4.1 modular arithmetic 345
 - 7.4.2 modular exponentiation 346

7.4.3 greatest common divisor and multiplicative inverses 348

7.4.4 the rsa cryptosystem 351

7.5 divide-and-conquer algorithms 353

7.5.1 the maximum contiguous subsequence sum problem 354

7.5.2 analysis of a basic divide-and-conquer recurrence 357

7.5.3 a general upper bound for divide-and-conquer running times 361

7.6 dynamic programming 363

7.7 backtracking 367

summary 370

key concepts 372

common errors 373

on the internet 373

exercises 374

references 381

chapter 8 sorting algorithms

383

8.1 why is sorting important? 384

8.2 preliminaries 385

8.3 analysis of the insertion sort and other simple sorts 385

8.4 shellsort 389

8.4.1 performance of shellsort 390

8.5 mergesort 393

8.5.1 linear-time merging of sorted arrays 393

8.5.2 the mergesort algorithm 395

8.6 quicksort 396

8.6.1 the quicksort algorithm 399

8.6.2 analysis of quicksort 401

8.6.3 picking the pivot 404

8.6.4 a partitioning strategy 406

8.6.5 keys equal to the pivot 408

8.6.6 median-of-three partitioning 408

8.6.7 small arrays 409

8.6.8 java quicksort routine 410

8.7 quickselect 412

8.8 a lower bound for sorting 413

summary	415
key concepts	416
common errors	417
on the internet	417
exercises	417
references	422

chapter 9 **randomization** **425**

9.1 why do we need random numbers?	425
9.2 random number generators	426
9.3 nonuniform random numbers	434
9.4 generating a random permutation	436
9.5 randomized algorithms	438
9.6 randomized primality testing	441
summary	444
key concepts	444
common errors	445
on the internet	446
exercises	446
references	449

part three **Applications**

chapter 10 **fun and games** **453**

10.1 word search puzzles	453
10.1.1 theory	454
10.1.2 java implementation	455
10.2 the game of tic-tac-toe	459
10.2.1 alpha-beta pruning	460
10.2.2 transposition tables	463
10.2.3 computer chess	467
summary	470
key concepts	470

common errors	470
on the internet	470
exercises	471
references	472

chapter 11 **stacks and compilers** 473

11.1	balanced-symbol checker	473
11.1.1	basic algorithm	474
11.1.2	implementation	475
11.2	a simple calculator	484
11.2.1	postfix machines	486
11.2.2	infix to postfix conversion	487
11.2.3	implementation	489
11.2.4	expression trees	498
	summary	499
	key concepts	500
	common errors	500
	on the internet	501
	exercises	501
	references	502

chapter 12 **utilities** 503

12.1	file compression	504
12.1.1	prefix codes	505
12.1.2	huffman's algorithm	507
12.1.3	implementation	509
12.2	a cross-reference generator	525
12.2.1	basic ideas	525
12.2.2	java implementation	525
	summary	529
	key concepts	530
	common errors	530
	on the internet	530
	exercises	530
	references	535

chapter 13 simulation**537**

- 13.1 the josephus problem** 537
 - 13.1.1 the simple solution 539
 - 13.1.2 a more efficient algorithm 539
- 13.2 event-driven simulation** 543
 - 13.2.1 basic ideas 543
 - 13.2.2 example: a call bank simulation 544
- summary** 552
- key concepts** 552
- common errors** 553
- on the internet** 553
- exercises** 553

chapter 14 graphs and paths**557**

- 14.1 definitions** 558
 - 14.1.1 representation 560
- 14.2 unweighted shortest-path problem** 569
 - 14.2.1 theory 569
 - 14.2.2 java implementation 575
- 14.3 positive-weighted, shortest-path problem** 575
 - 14.3.1 theory: dijkstra's algorithm 576
 - 14.3.2 java implementation 580
- 14.4 negative-weighted, shortest-path problem** 582
 - 14.4.1 theory 582
 - 14.4.2 java implementation 583
- 14.5 path problems in acyclic graphs** 585
 - 14.5.1 topological sorting 585
 - 14.5.2 theory of the acyclic shortest-path algorithm 587
 - 14.5.3 java implementation 587
 - 14.5.4 an application: critical-path analysis 590
- summary** 592
- key concepts** 593
- common errors** 594
- on the internet** 595

exercises 595

references 599

part four **Implementations**

chapter 15 **inner classes and implementation of ArrayList** **603**

15.1 **iterators and nested classes** 604

15.2 **iterators and inner classes** 606

15.3 **the AbstractCollection class** 610

15.4 **StringBuilder** 614

15.5 **implementation of ArrayList with an iterator** 615

summary 620

key concepts 621

common errors 621

on the internet 621

exercises 621

chapter 16 **stacks and queues** **625**

16.1 **dynamic array implementations** 625

16.1.1 **stacks** 626

16.1.2 **queues** 630

16.2 **linked list implementations** 635

16.2.1 **stacks** 636

16.2.2 **queues** 639

16.3 **comparison of the two methods** 643

16.4 **the java.util.Stack class** 643

16.5 **double-ended queues** 645

summary 645

key concepts 645

common errors 645

on the internet 646

exercises 646

chapter 17 linked lists**649**

- 17.1 basic ideas** 649
 - 17.1.1 header nodes 651
 - 17.1.2 iterator classes 652
- 17.2 java implementation** 654
- 17.3 doubly linked lists and circularly linked lists** 660
- 17.4 sorted linked lists** 663
- 17.5 implementing the collections api LinkedList class** 665
- summary** 676
- key concepts** 676
- common errors** 677
- on the internet** 677
- exercises** 677

chapter 18 trees**681**

- 18.1 general trees** 681
 - 18.1.1 definitions 682
 - 18.1.2 implementation 683
 - 18.1.3 an application: file systems 684
- 18.2 binary trees** 688
- 18.3 recursion and trees** 695
- 18.4 tree traversal: iterator classes** 697
 - 18.4.1 postorder traversal 701
 - 18.4.2 inorder traversal 705
 - 18.4.3 preorder traversal 705
 - 18.4.4 level-order traversals 708
- summary** 709
- key concepts** 710
- common errors** 711
- on the internet** 712
- exercises** 712

chapter 19 binary search trees**715**

- 19.1 basic ideas** 715
 - 19.1.1 the operations 716
 - 19.1.2 java implementation 718
- 19.2 order statistics** 725
 - 19.2.1 java implementation 726
- 19.3 analysis of binary search tree operations** 730
- 19.4 avl trees** 734
 - 19.4.1 properties 735
 - 19.4.2 single rotation 737
 - 19.4.3 double rotation 740
 - 19.4.4 summary of avl insertion 742
- 19.5 red-black trees** 743
 - 19.5.1 bottom-up insertion 744
 - 19.5.2 top-down red-black trees 746
 - 19.5.3 java implementation 747
 - 19.5.4 top-down deletion 754
- 19.6 aa-trees** 756
 - 19.6.1 insertion 758
 - 19.6.2 deletion 760
 - 19.6.3 java implementation 761
- 19.7 implementing the collections api TreeSet and
 TreeMap classes** 766
- 19.8 b-trees** 784
 - summary** 790
 - key concepts** 791
 - common errors** 792
 - on the internet** 792
 - exercises** 793
 - references** 797

chapter 20 hash tables**799**

- 20.1 basic ideas** 800
- 20.2 hash function** 801
 - 20.2.1 headCode in java.lang.String 803

20.3	linear probing	805
20.3.1	naive analysis of linear probing	806
20.3.2	what really happens: primary clustering	807
20.3.3	analysis of the find operation	808
20.4	quadratic probing	810
20.4.1	java implementation	814
20.4.2	analysis of quadratic probing	823
20.5	separate chaining hashing	823
20.6	hash tables versus binary search trees	824
20.7	hashing applications	826
	summary	826
	key concepts	827
	common errors	828
	on the internet	828
	exercises	828
	references	830

chapter 21 a priority queue: the binary heap 833

21.1	basic ideas	834
21.1.1	structure property	834
21.1.2	heap-order property	836
21.1.3	allowed operations	837
21.2	implementation of the basic operations	840
21.2.1	insertion	840
21.2.2	the deleteMin operation	842
21.3	the buildHeap operation: linear-time heap construction	844
21.4	advanced operations: decreaseKey and merge	849
21.5	internal sorting: heapsort	849
21.6	external sorting	852
21.6.1	why we need new algorithms	852
21.6.2	model for external sorting	853
21.6.3	the simple algorithm	853
21.6.4	multiway merge	855
21.6.5	polyphase merge	856
21.6.6	replacement selection	858

summary	859
key concepts	860
common errors	860
on the internet	861
exercises	861
references	865

part five **Advanced Data Structures**

chapter 22 splay trees 869

22.1 self-adjustment and amortized analysis	870
22.1.1 amortized time bounds	871
22.1.2 a simple self-adjusting strategy (that does not work)	871
22.2 the basic bottom-up splay tree	873
22.3 basic splay tree operations	876
22.4 analysis of bottom-up splaying	877
22.4.1 proof of the splaying bound	880
22.5 top-down splay trees	883
22.6 implementation of top-down splay trees	886
22.7 comparison of the splay tree with other search trees	891
summary	892
key concepts	892
common errors	893
on the internet	893
exercises	893
references	894

chapter 23 merging priority queues 897

23.1 the skew heap	897
23.1.1 merging is fundamental	898
23.1.2 simplistic merging of heap-ordered trees	898

- 23.1.3 the skew heap: a simple modification 899
- 23.1.4 analysis of the skew heap 900

23.2 the pairing heap 902

- 23.2.1 pairing heap operations 903
- 23.2.2 implementation of the pairing heap 904
- 23.2.3 application: dijkstra's shortest weighted path algorithm 910

summary 914

key concepts 914

common errors 914

on the internet 915

exercises 915

references 916

chapter 24 the disjoint set class

919

24.1 equivalence relations 920

24.2 dynamic equivalence and applications 920

- 24.2.1 application: generating mazes 921
- 24.2.2 application: minimum spanning trees 924
- 24.2.3 application: the nearest common ancestor problem 927

24.3 the quick-find algorithm 930

24.4 the quick-union algorithm 931

- 24.4.1 smart union algorithms 933
- 24.4.2 path compression 935

24.5 java implementation 936

24.6 worst case for union-by-rank and path compression 939

- 24.6.1 analysis of the union/find algorithm 940

summary 947

key concepts 947

common errors 948

on the internet 948

exercises 949

references 951

	appendix A operators	953
--	--------------------------------	------------

	appendix B graphical user interfaces	955
--	--	------------

- B.1 the abstract window toolkit and swing** 956
- B.2 basic objects in swing** 957
 - B.2.1 *Component* 958
 - B.2.2 *Container* 959
 - B.2.3 *top-level containers* 959
 - B.2.4 *JPanel* 960
 - B.2.5 *important i/o components* 962
- B.3 basic principles** 966
 - B.3.1 *layout managers* 967
 - B.3.2 *graphics* 971
 - B.3.3 *events* 973
 - B.3.4 *event handling: adapters and anonymous inner classes* 975
 - B.3.5 *summary: putting the pieces together* 977
 - B.3.6 *is this everything i need to know about swing?* 978
- summary** 979
- key concepts** 979
- common errors** 981
- on the internet** 982
- exercises** 982
- references** 983

	appendix C bitwise operators	985
--	--	------------

	index	989
--	--------------	------------