

Contents at a Glance

1	Introduction	43
2	Imperative Language Concepts	83
3	Classes and Objects	201
4	Arrays and Their Areas of Use	241
5	Handling Characters and Strings	287
6	Writing Custom Classes	369
7	Object-Oriented Relationship	427
8	Interfaces, Enumerations, Sealed Classes, Records	483
9	There Must Be Exceptions	539
10	Nested Types	599
11	Special Types of Java SE	615
12	Generics<T>	683
13	Lambda Expressions and Functional Programming	731
14	Architecture, Design, and Applied Object Orientation	799
15	Java Platform Module System	813
16	The Class Library	835
17	Introduction to Concurrent Programming	871
18	Introduction to Data Structures and Algorithms	909
19	Files and Data Streams	969
20	Introduction to Database Management with JDBC	1005
21	Bits and Bytes, Mathematics and Money	1011
22	Testing with JUnit	1067
23	The Tools of the JDK	1091

Contents

Preface	31
1 Introduction	43
1.1 Historical Background	43
1.2 On the Popularity of Java: The Key Features	45
1.2.1 Bytecode	46
1.2.2 Executing the Bytecode via a Virtual Machine	46
1.2.3 Platform Independence	46
1.2.4 Java as a Language, Runtime Environment, and Standard Library	47
1.2.5 Object Orientation in Java	47
1.2.6 Java Is Widespread and Well Known	48
1.2.7 Java Is Fast: Optimization and Just-In-Time Compilation	48
1.2.8 Pointers and References	50
1.2.9 Take Out the Trash, Garbage Collector!	51
1.2.10 Exception Handling	52
1.2.11 The Range of Libraries and Tools	52
1.2.12 Comparably Simple Syntax	53
1.2.13 Abandoning Controversial Concepts	53
1.2.14 Java Is Open Source	54
1.2.15 What Java Is Less Suitable for	55
1.3 Java versus Other Languages*	56
1.3.1 Java and C(++)	56
1.3.2 Java and JavaScript	57
1.3.3 A Word about Microsoft, Java, and J++	57
1.3.4 Java and C#/.NET	58
1.4 Further Development and Losses	59
1.4.1 The Development of Java and Its Future Prospects	59
1.4.2 Features, Enhancements, and Specification Requests	60
1.4.3 Applets	61
1.4.4 JavaFX	61
1.5 Java Platforms	62
1.5.1 Java Platform, Standard Edition	62
1.5.2 Java Platform, Micro Edition: Java for the Little Ones	65
1.5.3 Java for the Very, Very Little Ones	65
1.5.4 Java for the Big Ones: Jakarta EE (Formerly Java Platform, Enterprise Edition)	65
1.5.5 Real-Time Java	66

1.6	Java Platform, Standard Edition, Implementations	67
1.6.1	OpenJDK	67
1.6.2	Oracle JDK	68
1.7	Installing the Java Development Kit	69
1.7.1	Installing Oracle JDK on Windows	70
1.8	Compiling and Testing the First Program	71
1.8.1	A Square Numbers Program	72
1.8.2	The Compiler Run	73
1.8.3	The Runtime Environment	74
1.8.4	Common Compiler and Interpreter Issues	74
1.9	Development Environments	75
1.9.1	IntelliJ IDEA	76
1.9.2	Eclipse Integrated Development Environment	80
1.9.3	NetBeans	81
1.10	Further Reading	81

2 Imperative Language Concepts 83

2.1	Elements of the Java Programming Language	83
2.1.1	Tokens	84
2.1.2	Text Encoding by Unicode Characters	85
2.1.3	Identifiers	85
2.1.4	Literals	87
2.1.5	(Reserved) Keywords	87
2.1.6	Summary of the Lexical Analysis	88
2.1.7	Comments	89
2.2	From Classes to Statements	91
2.2.1	What Are Statements?	91
2.2.2	Class Declaration	92
2.2.3	The Journey Begins with <code>main(String[])</code>	93
2.2.4	The First Method Call: <code>println(...)</code>	93
2.2.5	Atomic Statements and Statement Sequences	95
2.2.6	More about <code>print(...)</code> , <code>println(...)</code> , and <code>printf(...)</code> for Screen Output	95
2.2.7	Application Programming Interface Documentation	97
2.2.8	Expressions	98
2.2.9	Expression Statements	98
2.2.10	First Insights into Object Orientation	99
2.2.11	Modifiers	100
2.2.12	Grouping Statements with Blocks	101

2.3	Data Types, Typing, Variables, and Assignments	102
2.3.1	Overview of Primitive Data Types	104
2.3.2	Variable Declarations	106
2.3.3	Automatic Type Detection with var	109
2.3.4	Final Variables and the final Modifier	110
2.3.5	Console Inputs	110
2.3.6	Truth Values	112
2.3.7	Integer Data Types	112
2.3.8	Underscores in Numbers	114
2.3.9	Alphanumeric Characters	115
2.3.10	The float and double Data Types	115
2.3.11	Good Names, Bad Names	117
2.3.12	No Automatic Initialization of Local Variables	118
2.4	Expressions, Operands, and Operators	119
2.4.1	Assignment Operator	119
2.4.2	Arithmetic Operators	121
2.4.3	Unary Minus and Plus	124
2.4.4	Prefix or Postfix Increment and Decrement	124
2.4.5	Assignment with Operation (Compound Assignment Operator)	126
2.4.6	Relational Operators and Equality Operators	128
2.4.7	Logical Operators: NOT, AND, OR, and XOR	129
2.4.8	Short-Circuit Operators	130
2.4.9	The Rank of Operators in Evaluation Order	132
2.4.10	Typecasting (Casting)	135
2.4.11	Overloaded Plus for Strings	140
2.4.12	Operators Missing*	141
2.5	Conditional Statements or Case Distinctions	142
2.5.1	Branching with the if Statement	142
2.5.2	Choosing the Alternative with an if-else Statement	145
2.5.3	The Condition Operator	148
2.5.4	The Switch Statement Provides an Alternative	151
2.5.5	Switch Expressions	157
2.6	Always the Same with Loops	160
2.6.1	The while Loop	161
2.6.2	The do-while Loop	163
2.6.3	The for Loop	164
2.6.4	Loop Conditions and Comparisons with ==*	168
2.6.5	Loop Termination with break and back to Test with continue	171
2.6.6	break and continue with Labels*	174
2.7	Methods of a Class	177
2.7.1	Components of a Method	177

2.7.2	Signature Description in the Java Application Programming Interface Documentation	179
2.7.3	Calling a Method	180
2.7.4	Declaring Methods without Parameters	181
2.7.5	Static Methods (Class Methods)	182
2.7.6	Parameters, Arguments, and Value Transfers	183
2.7.7	Ending Methods Prematurely with return	185
2.7.8	Unreachable Source Code for Methods*	185
2.7.9	Methods with Returns	186
2.7.10	Overloading Methods	191
2.7.11	Scope	193
2.7.12	Default Values for Unlisted Arguments*	195
2.7.13	Recursive Methods*	195
2.7.14	Towers of Hanoi*	198
2.8	Further Reading	200

3 Classes and Objects 201

3.1	Object-Oriented Programming	201
3.1.1	Why Object-Oriented Programming at All?	202
3.1.2	When I Think of Java, I Think of Reusability	202
3.2	Members of a Class	203
3.3	Natural Modeling Using Unified Modeling Language*	204
3.4	Creating New Objects	206
3.4.1	Creating an Instance of a Class Using the new Keyword	206
3.4.2	Declaring Reference Variables	207
3.4.3	Let's Get to the Point: Accessing Object Variables and Methods	208
3.4.4	The Connection between new, the Heap, and the Garbage Collector	212
3.4.5	Overview of Point Methods	213
3.4.6	Using Constructors	217
3.5	ZZZZZnake	218
3.6	Tying Packages, Imports, and Compilation Units	220
3.6.1	Java Packages	220
3.6.2	Packages in the Standard Library	221
3.6.3	Full Qualification and Import Declaration	221
3.6.4	Reaching All Types of a Package with Type-Import-on-Demand	223
3.6.5	Hierarchical Structures across Packages and Mirroring in the File System	224
3.6.6	The Package Declaration	224

3.6.7	Unnamed Package (Default Package)	225
3.6.8	Compilation Unit	226
3.6.9	Static Import*	226
3.7	Using References, Diversity, Identity, and Equality	228
3.7.1	null References and the Question of Philosophy	228
3.7.2	Everything to null? Testing References	230
3.7.3	Assignments with References	231
3.7.4	Methods with Reference Types as Parameters	232
3.7.5	Identity of Objects	236
3.7.6	Equivalence and the equals(...) Method	237
3.8	Further Reading	239
4	Arrays and Their Areas of Use	241
4.1	Simple Field Work	241
4.1.1	Basic Components	242
4.1.2	Declaring Array Variables	243
4.1.3	Creating Array Objects with new	244
4.1.4	Arrays with { contents }	245
4.1.5	Reading the Length of an Array via the Object Variable Length	246
4.1.6	Accessing the Elements via the Index	246
4.1.7	Typical Array Errors	248
4.1.8	Passing Arrays to Methods	250
4.1.9	Multiple Return Values*	250
4.1.10	Preinitialized Arrays	251
4.2	The Extended for Loop	252
4.2.1	Using Anonymous Arrays in the Extended for Loop	253
4.2.2	Example: Searching Arrays with Strings	254
4.2.3	Creating Random Player Positions	255
4.3	A Method with a Variable Number of Arguments	256
4.3.1	System.out.printf(...) Accepts Any Number of Arguments	257
4.3.2	Finding the Average of Variable Arguments	257
4.3.3	Vararg Design Tips*	259
4.4	Multidimensional Arrays*	259
4.4.1	Nonrectangular Arrays*	262
4.5	Library Support for Arrays	264
4.5.1	Cloning Can Be Worthwhile: Propagating Arrays	264
4.5.2	Why Can Arrays “Do” So Little?	265
4.5.3	Copying Array Contents	266

4.6	Using the Arrays Class for Comparing, Filling, Searching, and Sorting	267
4.6.1	String Representation of an Array	267
4.6.2	Sorting	268
4.6.3	Parallel Sorting	269
4.6.4	Comparing Arrays of Primitives with Arrays.equals(...) and Arrays.deepEquals(...)*	269
4.6.5	Comparing Object Arrays Using Arrays.equals(...) and Arrays.deepEquals(...)*	271
4.6.6	Searching Differences Using Mismatch (...) *	272
4.6.7	Filling Arrays*	272
4.6.8	Copying Array Sections*	273
4.6.9	Binary Search*	275
4.6.10	Lexicographic Array Comparisons Using compare(...) and compareUnsigned(...)	276
4.6.11	Arrays for Lists with Arrays.asList(...): Convenient for Searching and Comparing*	277
4.6.12	A Long Snake	278
4.7	The Entry Point for the Runtime System: main(...)	281
4.7.1	Correct Declaration of the Start Method	281
4.7.2	Processing Command Line Arguments	282
4.7.3	The Return Type of main(...) and System.exit(int)*	283
4.8	Further Reading	285

5 Handling Characters and Strings 287

5.1	From ASCII via ISO-8859-1 to Unicode	287
5.1.1	ASCII	287
5.1.2	ISO/IEC 8859-1	288
5.1.3	Unicode	289
5.1.4	Unicode Character Encoding	291
5.1.5	Escape Sequences	292
5.1.6	Notation for Unicode Characters and Unicode Escapes	292
5.1.7	Java Versions Go Hand in Hand with the Unicode Standard*	294
5.2	Data Types for Characters and Strings	295
5.3	The Character Class	296
5.3.1	Is That So?	296
5.3.2	Converting Characters to Uppercase/Lowercase	298
5.3.3	From Character to String	299
5.3.4	From char to int: From Character to Number*	299

5.4	Strings	301
5.5	The String Class and Its Methods	303
5.5.1	String Literals as String Objects for Constant Strings	303
5.5.2	Concatenation with +	303
5.5.3	Multiline Text Blocks with <code>"""</code>	304
5.5.4	String Length and Testing for Empty Strings	309
5.5.5	Accessing a Specific Character with <code>charAt(int)</code>	310
5.5.6	Searching for Contained Characters and Strings	311
5.5.7	The Hangman Game	314
5.5.8	Good That We Have Compared	316
5.5.9	Extracting String Sections	320
5.5.10	Appending Strings, Merging Strings, Case Sensitivity, and Whitespace	325
5.5.11	Searched, Found, and Replaced	328
5.5.12	Creating String Objects with Constructors and <code>fromRepeats*</code>	330
5.6	Mutable Strings with <code>StringBuilder</code> and <code>StringBuffer</code>	333
5.6.1	Creating <code>StringBuilder</code> Objects	334
5.6.2	Converting <code>StringBuilder</code> to Other String Formats	335
5.6.3	Requesting Characters or Strings	335
5.6.4	Appending Data	335
5.6.5	Setting, Deleting, and Reversing Characters and Strings	337
5.6.6	Length and Capacity of a <code>StringBuilder</code> Object*	339
5.6.7	Comparison of <code>StringBuilder</code> Instances and Strings with <code>StringBuilder</code>	340
5.6.8	<code>hashCode()</code> with <code>StringBuilder*</code>	342
5.7	<code>CharSequence</code> as Base Type	342
5.7.1	Basic Operations of the Interface	343
5.7.2	Static <code>compare(...)</code> Method in <code>CharSequence</code>	344
5.7.3	Default Methods in the <code>CharSequence</code> Interface*	345
5.8	Converting Primitives and Strings	345
5.8.1	Converting Different Types to String Representations	345
5.8.2	Converting String Contents to a Primitive Value	347
5.8.3	String Representation in Binary, Hex, and Octal Formats*	349
5.8.4	<code>parse*(...)</code> and <code>print*()</code> Methods in <code>DatatypeConverter*</code>	353
5.9	Concatenating Strings	353
5.9.1	Concatenating Strings with <code>StringJoiner</code>	353
5.10	Decomposing Strings	355
5.10.1	Splitting Strings via <code>split(...)</code>	356
5.10.2	Yes We Can, Yes We Scan: The <code>Scanner</code> Class	356

5.11	Formatting Outputs	360
5.11.1	Formatting and Outputting via format()	361
5.12	Further Reading	367

6 Writing Custom Classes 369

6.1	Declaring Custom Classes with Members	369
6.1.1	Minimum Class	370
6.1.2	Declaring Object Variables	370
6.1.3	Declaring Methods	373
6.1.4	Shadowed Variables	375
6.1.5	The this Reference	377
6.2	Privacy and Visibility	380
6.2.1	For the Public: public	381
6.2.2	Not Public: Passwords Are private	381
6.2.3	Why Not Free Methods and Variables for All?	383
6.2.4	private Is Not Quite Private: It Depends on Who Sees It*	383
6.2.5	Declaring Access Methods for Object Variables	384
6.2.6	Setters and Getters according to the JavaBeans Specification	384
6.2.7	Package-Visibility	386
6.2.8	Visibility Summary	388
6.3	One for All: Static Methods and Class Variables	390
6.3.1	Why Static Members Are Useful	391
6.3.2	Static Members with static	392
6.3.3	Using Static Members via References?*	393
6.3.4	Why Case Sensitivity Is Important*	394
6.3.5	Static Variables for Data Exchange*	395
6.3.6	Static Members and Object Members*	396
6.4	Constants and Enumerations	397
6.4.1	Constants via Static Final Variables	397
6.4.2	Type-Unsafe Enumerations	398
6.4.3	Enumeration Types: Type-Safe Enumerations with enum	400
6.5	Creating and Destroying Objects	405
6.5.1	Writing Constructors	405
6.5.2	Relationship of Method and Constructor	406
6.5.3	The Default Constructor	407
6.5.4	Parameterized and Overloaded Constructors	408
6.5.5	Copy Constructors	410
6.5.6	Calling Another Constructor of the Same Class via this(...)	412

6.5.7	Immutable Objects and Wither Methods	414
6.5.8	We Don't Miss You: The Garbage Collector	416
6.6	Class and Object Initialization*	418
6.6.1	Initializing Object Variables	418
6.6.2	Static Blocks as Class Initializers	420
6.6.3	Initializing Class Variables	421
6.6.4	Compiled Assignments of the Class Variables	421
6.6.5	Instance_INITIALIZER	422
6.6.6	Setting Final Values in the Constructor and Static Blocks	425
6.7	Conclusion	426

7 Object-Oriented Relationship 427

7.1	Associations between Objects	427
7.1.1	Association Types	427
7.1.2	Unidirectional 1-to-1 Relationship	428
7.1.3	Becoming Friends: Bidirectional 1-to-1 Relationships	429
7.1.4	Unidirectional 1-to-n Relationships	431
7.2	Inheritance	436
7.2.1	Inheritance in Java	437
7.2.2	Modeling Events	438
7.2.3	The Implicit Base Class java.lang.Object	440
7.2.4	Single and Multiple Inheritance*	440
7.2.5	Do Children See Everything? The Protected Visibility	441
7.2.6	Constructors in Inheritance and super(...)	442
7.3	Types in Hierarchies	447
7.3.1	Automatic and Explicit Typecasting	447
7.3.2	The Substitution Principle	450
7.3.3	Testing Types with the instanceof Operator	452
7.3.4	Pattern Matching for instanceof	455
7.4	Overriding Methods	457
7.4.1	Providing Methods in Subclasses with a New Behavior	457
7.4.2	With super to the Parents	461
7.5	Testing Dynamic Bindings	463
7.5.1	Bound to toString()	463
7.5.2	Implementing System.out.println(Object)	465
7.6	Final Classes and Final Methods	466
7.6.1	Final Classes	466
7.6.2	Non-Overridable (final) Methods	466

7.7	Abstract Classes and Abstract Methods	468
7.7.1	Abstract Classes	468
7.7.2	Abstract Methods	470
7.8	Further Information on Overriding and Dynamic Binding	476
7.8.1	No Dynamic Binding for Private, Static, and final Methods	476
7.8.2	Covariant Return Types	476
7.8.3	Array Types and Covariance*	477
7.8.4	Dynamic Binding even with Constructor Calls*	478
7.8.5	No Dynamic Binding for Covered Object Variables*	480
7.9	A Programming Task	482

8 Interfaces, Enumerations, Sealed Classes, Records 483

8.1	Interfaces	483
8.1.1	Interfaces Are New Types	483
8.1.2	Declaring Interfaces	484
8.1.3	Abstract Methods in Interfaces	484
8.1.4	Implementing Interfaces	485
8.1.5	A Polymorphism Example with Interfaces	487
8.1.6	Multiple Inheritance with Interfaces	488
8.1.7	No Risk of Collision with Multiple Inheritance*	491
8.1.8	Extending Interfaces: Subinterfaces	493
8.1.9	Constant Declarations for Interfaces	494
8.1.10	Subsequent Implementation of Interfaces*	494
8.1.11	Static Programmed Methods in Interfaces	495
8.1.12	Extending and Modifying Interfaces	497
8.1.13	Default Methods	498
8.1.14	Declaring and Using Extended Interfaces	500
8.1.15	Public and Private Interface Methods	503
8.1.16	Extended Interfaces, Multiple Inheritance, and Ambiguities*	504
8.1.17	Creating Building Blocks with Default Methods*	508
8.1.18	Marker Interfaces*	512
8.1.19	(Abstract) Classes and Interfaces in Comparison	513
8.2	Enumeration Types	513
8.2.1	Methods on Enum Objects	514
8.2.2	Enumerations with Custom Methods, Constructors, and Initializers*	518
8.3	Sealed Classes and Interfaces	523
8.3.1	Sealed Classes and Interfaces	525
8.3.2	Subclasses Are Final, Sealed, and Non-Sealed	527
8.3.3	Abbreviated Notations	528

8.4	Records	529
8.4.1	Simple Records	529
8.4.2	Records with Methods	531
8.4.3	Customizing Record Constructors	532
8.4.4	Adding Constructors	534
8.4.5	Sealed Interfaces and Records	535
8.4.6	Records: Summary	536

9 There Must Be Exceptions 539

9.1	Fencing In Problem Areas	539
9.1.1	Exceptions in Java with try and catch	540
9.1.2	Checked and Unchecked Exceptions	540
9.1.3	A NumberFormatException (Unchecked Exception)	541
9.1.4	Appending a Date/Timestamp to a Text File (Checked Exception)	543
9.1.5	Repeating Canceled Sections*	545
9.1.6	Empty catch Blocks	546
9.1.7	Catching Multiple Exceptions	547
9.1.8	Combining Identical catch Blocks with Multi-Catch	548
9.2	Redirecting Exceptions and throws at the Head of Methods/ Constructors	549
9.2.1	throws in Constructors and Methods	549
9.3	The Class Hierarchy of Exceptions	550
9.3.1	Members of the Exception Object	550
9.3.2	Base Type Throwable	551
9.3.3	The Exception Hierarchy	552
9.3.4	Catching Super-Exceptions	552
9.3.5	Already Caught?	554
9.3.6	Procedure of an Exceptional Situation	555
9.3.7	No General Catching!	555
9.3.8	Known RuntimeException Classes	557
9.3.9	Interception Is Possible, but Not Mandatory	558
9.4	Final Handling Using finally	558
9.4.1	The Ignorant Version	559
9.4.2	The Well-Intentioned Attempt	560
9.4.3	From Now On, Closing Is Part of the Agenda	560
9.4.4	Summary	562
9.4.5	A try without a catch, but a try-finally	562
9.5	Triggering Custom Exceptions	564
9.5.1	Triggering Exceptions via throw	564
9.5.2	Knowing and Using Existing Runtime Exception Types	566

9.5.3	Testing Parameters and Good Error Messages	568
9.5.4	Declaring New Exception Classes	570
9.5.5	Custom Exceptions as Subclasses of Exception or RuntimeException?	571
9.5.6	Catching and Redirecting Exceptions*	574
9.5.7	Changing the Call Stack of Exceptions*	575
9.5.8	Nested Exceptions*	576
9.6	try with Resources (Automatic Resource Management)	579
9.6.1	try with Resources	580
9.6.2	The AutoCloseable Interface	581
9.6.3	Exceptions to close()	582
9.6.4	Types That Are AutoCloseable and Closeable	583
9.6.5	Using Multiple Resources	583
9.6.6	Suppressed Exceptions*	585
9.7	Special Features of Exception Handling*	588
9.7.1	Return Values for Thrown Exceptions	588
9.7.2	Exceptions and Returns Disappear: The Duo return and finally	589
9.7.3	throws on Overridden Methods	590
9.7.4	Unreachable catch Clauses	592
9.8	Hard Errors: Error*	593
9.9	Assertions*	594
9.9.1	Using Assertions in Custom Programs	595
9.9.2	Enabling Assertions and Runtime Errors	595
9.9.3	Enabling or Disabling Assertions More Detailed	597
9.10	Conclusion	597

10 Nested Types 599

10.1	Nested Classes, Interfaces, and Enumerations	599
10.2	Static Nested Types	601
10.2.1	Modifiers and Visibility	602
10.2.2	Records as Containers	602
10.2.3	Implementing Static Nested Types*	602
10.3	Non-Static Nested Types	603
10.3.1	Creating Instances of Inner Classes	603
10.3.2	The this Reference	604
10.3.3	Class Files Generated by the Compiler*	605
10.4	Local Classes	605
10.4.1	Example with a Custom Declaration	606

10.4.2	Using a Local Class for a Timer	606
10.5	Anonymous Inner Classes	607
10.5.1	Using an Anonymous Inner Class for the Timer	608
10.5.2	Implementing Anonymous Inner Classes*	609
10.5.3	Constructors of Anonymous Inner Classes	609
10.5.4	Accessing Local Variables from Local and Anonymous Classes*	611
10.5.5	Nested Classes Access Private Members	612
10.6	Nests	613
10.7	Conclusion	614

11 Special Types of Java SE 615

11.1	Object Is the Mother of All Classes	616
11.1.1	Class Objects	616
11.1.2	Object Identification with toString()	617
11.1.3	Object Equivalence with equals(...) and Identity	619
11.1.4	Cloning an Object Using clone()*	625
11.1.5	Returning Hash Values via hashCode()*	630
11.1.6	System.identityHashCode(...) and the Problem of Non-Unique Object References*	636
11.1.7	Synchronization*	637
11.2	Weak References and Cleaners	638
11.3	The java.util.Objects Utility Class	639
11.3.1	Built-In Null Tests for equals(...)/hashCode()	639
11.3.2	Objects.toString(...)	640
11.3.3	null Checks with Built-In Exception Handling	640
11.3.4	Tests for null	641
11.3.5	Checking Index-Related Program Arguments for Correctness	642
11.4	Comparing Objects and Establishing Order	643
11.4.1	Naturally Ordered or Not?	643
11.4.2	compare*() Method of the Comparable and Comparator Interfaces	644
11.4.3	Return Values Encode the Order	645
11.4.4	Sorting Candy by Calories Using a Sample Comparator	645
11.4.5	Tips for Comparator and Comparable Implementations	647
11.4.6	Static and Default Methods in Comparator	648
11.5	Wrapper Classes and Autoboxing	651
11.5.1	Creating Wrapper Objects	653
11.5.2	Conversions to a String Representation	654

11.5.3	Parsing from a String Representation	655
11.5.4	The Number Base Class for Numeric Wrapper Objects	655
11.5.5	Performing Comparisons with <code>compare*(...)</code> , <code>compareTo(...)</code> , <code>equals(...)</code> , and Hash Values	657
11.5.6	Static Reduction Methods in Wrapper Classes	660
11.5.7	Constants for the Size of a Primitive Type*	661
11.5.8	Handling Unsigned Numbers*	661
11.5.9	The Integer and Long Classes	663
11.5.10	The Double and Float Classes for Floats	664
11.5.11	The Boolean Class	664
11.5.12	Autoboxing: Boxing and Unboxing	665
11.6	Iterator, Iterable*	670
11.6.1	The Iterator Interface	670
11.6.2	The Supplier of the Iterator	673
11.6.3	The Iterable Interface	674
11.6.4	Extended <code>for</code> and <code>Iterable</code>	674
11.6.5	Internal Iteration	675
11.6.6	Implementing a Custom <code>Iterable*</code>	676
11.7	Annotations in Java Platform, Standard Edition	677
11.7.1	Places for Annotations	677
11.7.2	Annotation Types from <code>java.lang</code>	678
11.7.3	<code>@Deprecated</code>	678
11.7.4	Annotations with Additional Information	679
11.7.5	<code>@SuppressWarnings</code>	679
11.8	Further Reading	682

12 Generics<T> 683

12.1	Introduction to Java Generics	683
12.1.1	Man versus Machine: Type Checking of the Compiler and the Runtime Environment	683
12.1.2	Rockets	684
12.1.3	Declaring Generic Types	686
12.1.4	Using Generics	688
12.1.5	Diamonds Are Forever	690
12.1.6	Generic Interfaces	693
12.1.7	Generic Methods/Constructors and Type Inference	695
12.2	Implementing Generics, Type Erasure, and Raw Types	699
12.2.1	Implementation Options	699
12.2.2	Type Erasure	699

12.2.3	Problems with Type Erasure	700
12.2.4	Raw Types	704
12.3	Restricting Types via Bounds	706
12.3.1	Simple Restrictions with extends	707
12.3.2	Other Supertypes with &	709
12.4	Type Parameters in the throws Clause*	710
12.4.1	Declaring a Class with Type Variable <E extends Exception>	710
12.4.2	Parameterized Type for Type Variable <E extends Exception>	710
12.5	Inheritance and Invariance with Generics	713
12.5.1	Arrays Are Covariant	713
12.5.2	Generics Aren't Covariant, but Invariant	714
12.5.3	Wildcards with ?	715
12.5.4	Bounded Wildcards	717
12.5.5	Bounded Wildcard Types and Bounded Type Variables	720
12.5.6	The PECS Principle	722
12.6	Consequences of Type Erasure: Type Tokens, Arrays*	725
12.6.1	Type Tokens	725
12.6.2	Supertype Tokens	727
12.6.3	Generics and Arrays	728
12.7	Further Reading	729

13 Lambda Expressions and Functional Programming

731

13.1	Functional Interfaces and Lambda Expressions	731
13.1.1	Classes Implement Interfaces	731
13.1.2	Lambda Expressions Implement Interfaces	733
13.1.3	Functional Interfaces	734
13.1.4	The Type of a Lambda Expression Depends on the Target Type	735
13.1.5	@FunctionalInterface Annotations	740
13.1.6	Syntax for Lambda Expressions	741
13.1.7	The Environment of Lambda Expressions and Variable Accesses	745
13.1.8	Exceptions in Lambda Expressions	751
13.1.9	Classes with an Abstract Method as a Functional Interface?*	755
13.2	Method References	755
13.2.1	Motivation	755
13.2.2	Method References with ::	756
13.2.3	Variations of Method References	756

13.3	Constructor References	759
13.3.1	Writing Constructor References	760
13.3.2	Parameterless and Parameterized Constructors	761
13.3.3	Useful Predefined Interfaces for Constructor References	761
13.4	Functional Programming	762
13.4.1	Code = Data	762
13.4.2	Programming Paradigms: Imperative or Declarative	763
13.4.3	Principles of Functional Programming	764
13.4.4	Imperative Programming and Functional Programming	767
13.4.5	Comparator as an Example of Higher-Order Functions	769
13.4.6	Viewing Lambda Expressions as Mappings or Functions	769
13.5	Functional Interfaces from the <code>java.util.function</code> Package	770
13.5.1	Blocks with Code and the Functional Interface Consumer	771
13.5.2	Supplier	773
13.5.3	Predicates and <code>java.util.function.Predicate</code>	773
13.5.4	Functions via the Functional Interface <code>java.util.function.Function</code>	775
13.5.5	I Take Two	779
13.5.6	Functional Interfaces with Primitives	782
13.6	Optional Is Not a Non-Starter	784
13.6.1	Using null	785
13.6.2	The Optional Type	787
13.6.3	Starting Functional Interfaces with Optional	789
13.6.4	Primitive-Optional with Special Optional* Classes	792
13.7	What Is So Functional Now?	795
13.7.1	Recyclability	795
13.7.2	Stateless, Immutable	795
13.8	Further Reading	797

14 Architecture, Design, and Applied Object Orientation 799

14.1	SOLID Modeling	799
14.1.1	Three Rules	800
14.1.2	SOLID	800
14.1.3	Don't Be STUPID	802
14.2	Architecture, Design, and Implementation	803
14.3	Design Patterns	803
14.3.1	Motivation for Design Patterns	804
14.3.2	Singleton	805

14.3.3	Factory Methods	806
14.3.4	Implementing the Observer Pattern with Listeners	807
14.4	Further Reading	811

15 Java Platform Module System 813

15.1	Class Loader and Module/Classpath	813
15.1.1	Loading Classes per Request	813
15.1.2	Watching the Class Loader at Work	814
15.1.3	JMOD Files and JAR Files	815
15.1.4	Where the Classes Come from: Search Locations and Special Class Loaders	816
15.1.5	Setting the Search Path	817
15.2	Importing Modules	819
15.2.1	Who Sees Whom?	819
15.2.2	Platform Modules and a JMOD Example	821
15.2.3	Using Internal Platform Features: --add-exports	821
15.2.4	Integrating New Modules	824
15.3	Developing Custom Modules	825
15.3.1	Module com.tutego.candytester	825
15.3.2	Module Declaration with module-info.java and Exports	826
15.3.3	Module com.tutego.main	826
15.3.4	Module info File with requires	827
15.3.5	Writing Module Inserters: Java Virtual Machine Switches -p and -m	828
15.3.6	Experiments with the Module Info File	829
15.3.7	Automatic Modules	829
15.3.8	Unnamed Modules	830
15.3.9	Readability and Accessibility	831
15.3.10	Module Migration	832
15.4	Further Reading	833

16 The Class Library 835

16.1	The Java Class Philosophy	835
16.1.1	Modules, Packages, and Types	836
16.1.2	Overview of the Packages of the Standard Library	838

16.2	Simple Time Measurement and Profiling*	842
16.2.1	Profilers	843
16.3	The Class Class	843
16.3.1	Obtaining a Class Object	843
16.3.2	A Class Is a Type	846
16.4	The Utility Classes System and Members	846
16.4.1	Memory of the Java Virtual Machine	847
16.4.2	Number of CPUs or Cores	848
16.4.3	System Properties of the Java Environment	848
16.4.4	Setting Custom Properties from the Console*	850
16.4.5	Newline Characters and line.separator	851
16.4.6	Environment Variables of the Operating System	852
16.5	The Languages of Different Countries	853
16.5.1	Regional Languages via Locale Objects	853
16.6	Overview of Important Date Classes	857
16.6.1	Unix Time: January 1, 1970	858
16.6.2	System.currentTimeMillis()	858
16.6.3	Simple Time Conversions via TimeUnit	859
16.7	Date-Time API	860
16.7.1	Initial Overview	860
16.7.2	Human Time and Machine Time	861
16.7.3	The LocalDate Date Class	863
16.8	Logging with Java	864
16.8.1	Logging Application Programming Interfaces	865
16.8.2	Logging with java.util.logging	865
16.9	Maven: Resolving Build Management and Dependencies	867
16.9.1	Dependency to Be Accepted	868
16.9.2	Local and the Remote Repository	868
16.9.3	Lifecycles, Stages, and Maven Plugins	869
16.10	Further Reading	869

17 Introduction to Concurrent Programming 871

17.1	Concurrency and Parallelism	871
17.1.1	Multitasking, Processes, and Threads	872
17.1.2	Threads and Processes	873
17.1.3	How Concurrent Programs Can Increase Speed	874
17.1.4	How Java Can Provide for Concurrency	875

17.2	Generating Existing Threads and New Threads	875
17.2.1	Main Thread	876
17.2.2	Who Am I?	876
17.2.3	Implementing the Runnable Interface	876
17.2.4	Starting Thread with Runnable	878
17.2.5	Parameterizing Runnable	879
17.2.6	Extending the Thread Class*	880
17.3	Thread Members and States	882
17.3.1	The Name of a Thread	882
17.3.2	The States of a Thread*	882
17.3.3	Sleepers Wanted	883
17.3.4	When Threads Are Finished	885
17.3.5	Terminating a Thread Politely Using Interrupts	885
17.3.6	Unhandled Exceptions, Thread End, and UncaughtExceptionHandler	887
17.3.7	The stop() from the Outside and the Rescue with ThreadDeath*	889
17.3.8	Stopping and Resuming the Work*	891
17.3.9	Priority*	892
17.4	Enter the Executor	893
17.4.1	The Executor Interface	893
17.4.2	Happy as a Group: The Thread Pools	895
17.4.3	Threads with return via Callable	897
17.4.4	Memories of the Future: The Future Return	899
17.4.5	Processing Multiple Callable Objects	902
17.4.6	CompletionService and ExecutorCompletionService	903
17.4.7	ScheduledExecutorService: Repetitive Tasks and Time Controls	904
17.4.8	Asynchronous Programming with CompletableFuture (CompletionStage)	905
17.5	Further Reading	907

18 Introduction to Data Structures and Algorithms 909

18.1	Lists	909
18.1.1	First List Example	910
18.1.2	Selection Criterion ArrayList or LinkedList	911
18.1.3	The List Interface	911
18.1.4	ArrayList	918
18.1.5	LinkedList	919
18.1.6	The Array Adapter Arrays.asList(...)	921
18.1.7	ListIterator*	923

18.1.8	Understanding toArray(...) of Collection: Recognizing Traps	924
18.1.9	Managing Primitive Elements in Data Structures	927
18.2	Sets	928
18.2.1	A First Example of a Set	928
18.2.2	Methods of the Set Interface	930
18.2.3	HashSet	932
18.2.4	TreeSet: The Sorted Set	933
18.2.5	The Interfaces NavigableSet and SortedSet	935
18.2.6	LinkedHashSet	937
18.3	Associative Memory	938
18.3.1	The HashMap and TreeMap Classes and Static Map Methods	938
18.3.2	Inserting and Querying the Associative Memory	942
18.4	The Stream API	944
18.4.1	Declarative Programming	944
18.4.2	Internal versus External Iteration	945
18.4.3	What Is a Stream?	946
18.5	Creating a Stream	947
18.5.1	Stream.of(...)	948
18.5.2	Stream.generate(...)	949
18.5.3	Stream.iterate(...)	949
18.5.4	Parallel or Sequential Streams	950
18.6	Terminal Operations	951
18.6.1	Number of Elements	951
18.6.2	And Now All: forEach(...)	951
18.6.3	Getting Individual Elements from the Stream	952
18.6.4	Existence Tests with Predicates	953
18.6.5	Reducing a Stream to Its Smallest or Largest Element	953
18.6.6	Reducing a Stream with Its Own Functions	954
18.6.7	Writing Results to a Container, Part 1: collect(...)	956
18.6.8	Writing Results to a Container, Part 2: Collector and Collectors	957
18.6.9	Writing Results to a Container, Part 3: Groupings	959
18.6.10	Transferring Stream Elements to an Array or an Iterator	961
18.7	Intermediary Operations	962
18.7.1	Element Previews	963
18.7.2	Filtering Elements	963
18.7.3	Stateful Intermediary Operations	963
18.7.4	Prefix Operations	965
18.7.5	Images	966
18.8	Further Reading	968

19 Files and Data Streams 969

19.1 Old and New Worlds in java.io and java.nio	969
19.1.1 java.io Package with the File Class	969
19.1.2 NIO.2 and the java.nio Package	970
19.2 File Systems and Paths	970
19.2.1 FileSystem and Path	971
19.2.2 The Files Utility Class	977
19.3 Random Access Files	980
19.3.1 Opening a RandomAccessFile for Reading and Writing	980
19.3.2 Reading from RandomAccessFile	981
19.3.3 Writing with RandomAccessFile	983
19.3.4 The Length of the RandomAccessFile	983
19.3.5 Back and Forth within the File	984
19.4 Base Classes for Input/Output	985
19.4.1 The Four Abstract Base Classes	985
19.4.2 The Abstract Base Class OutputStream	986
19.4.3 The Abstract Base Class InputStream	988
19.4.4 The Abstract Base Class Writer	990
19.4.5 The Appendable Interface*	991
19.4.6 The Abstract Base Class Reader	992
19.4.7 The Interfaces Closeable, AutoCloseable, and Flushable	995
19.5 Reading from Files and Writing to Files	996
19.5.1 Obtaining Byte-Oriented Data Streams via Files	997
19.5.2 Obtaining Character-Oriented Data Streams via Files	997
19.5.3 The Function of OpenOption in the Files.new*(...) Methods	999
19.5.4 Loading Resources from the Module Path and from JAR Files	1001
19.6 Further Reading	1003

20 Introduction to Database Management with JDBC 1005

20.1 Relational Databases and Java Access	1005
20.1.1 The Relational Model	1005
20.1.2 Java Application Programming Interfaces for Accessing Relational Databases	1006
20.1.3 The JDBC API and Implementations: The JDBC Driver	1006
20.1.4 H2 Is the Tool in Java	1007

20.2	A Sample Query	1008
20.2.1	Steps to Query the Database	1008
20.2.2	Accessing the Relational Database with Java	1008
20.3	Further Reading	1009

21 Bits and Bytes, Mathematics and Money 1011

21.1	Bits and Bytes	1011
21.1.1	The Bit Operators: Complement, AND, OR, and XOR	1012
21.1.2	Representation of Integers in Java: Two's Complement	1013
21.1.3	The Binary, Octal, and Hexadecimal Place Value Systems	1014
21.1.4	Effect of Typecasting on Bit Patterns	1016
21.1.5	Working without Signs	1018
21.1.6	The Shift Operators	1021
21.1.7	Setting, Clearing, Reversing, and Testing a Bit	1023
21.1.8	Bit Methods of the Integer and Long Classes	1024
21.2	Floating Point Arithmetic in Java	1025
21.2.1	Special Values for Infinity, Zero, and Not a Number	1026
21.2.2	Standard Notation and Scientific Notation for Floats*	1029
21.2.3	Mantissas and Exponents*	1029
21.3	The Members of the Math Class	1031
21.3.1	Object Variables of the Math Class	1031
21.3.2	Absolute Values and Signs	1032
21.3.3	Maximums/Minimums	1032
21.3.4	Rounding Values	1033
21.3.5	Remainder of an Integer Division*	1035
21.3.6	Division with Rounding toward Negative Infinity and Alternative Remainders*	1036
21.3.7	Multiply-Accumulate	1038
21.3.8	Square Root and Exponential Methods	1038
21.3.9	The Logarithm*	1039
21.3.10	Angle Methods*	1040
21.3.11	Random Numbers	1041
21.4	Accuracy and the Value Range of Type and Overflow Control*	1042
21.4.1	The Largest and Smallest Values	1042
21.4.2	Overflow and Everything Entirely Exact	1042
21.4.3	What in the World Does the ulp Method Do?	1045
21.5	Random Numbers: Random, ThreadLocalRandom, and SecureRandom	1046
21.5.1	The Random Class	1047

21.5.2	ThreadLocalRandom	1050
21.5.3	The SecureRandom Class*	1050
21.6	Large Numbers*	1051
21.6.1	The BigInteger Class	1051
21.6.2	Example: Quite Long Factorials with BigInteger	1058
21.6.3	Large Floats with BigDecimal	1059
21.6.4	Conveniently Setting the Calculation Accuracy via MathContext	1062
21.6.5	Calculating even Faster with Mutable Implementations	1063
21.7	Money and Currency	1064
21.7.1	Representing Amounts of Money	1064
21.7.2	ISO 4217	1064
21.7.3	Representing Currencies in Java	1065
21.8	Further Reading	1066

22 Testing with JUnit 1067

22.1	Software Tests	1067
22.1.1	Procedure for Writing Test Cases	1068
22.2	The JUnit Testing Framework	1068
22.2.1	JUnit Versions	1069
22.2.2	Integrating JUnit	1069
22.2.3	Test-Driven Development and the Test-First Approach	1069
22.2.4	Test, Implement, Test, Implement, Test, Rejoice	1070
22.2.5	Running JUnit Tests	1072
22.2.6	assert*(...) Methods of the Assertions Class	1073
22.2.7	Testing Exceptions	1076
22.2.8	Setting Limits for Execution Times	1077
22.2.9	Labels with @DisplayName	1078
22.2.10	Nested Tests	1078
22.2.11	Ignoring Tests	1078
22.2.12	Canceling Tests with Methods of the Assumptions Class	1079
22.2.13	Parameterized Tests	1079
22.3	Java Assertion Libraries and AssertJ	1081
22.3.1	AssertJ	1081
22.4	Structure of Large Test Cases	1083
22.4.1	Fixtures	1083
22.4.2	Collections of Test Classes and Class Organization	1084
22.5	Good Design Enables Effective Testing	1085

22.6	Dummy, Fake, Stub, and Mock	1087
22.7	JUnit Extensions and Testing Add-Ons	1089
22.7.1	Web Tests	1089
22.7.2	Testing the Database Interface	1089
22.8	Further Reading	1089

23 The Tools of the JDK 1091

23.1	Overview	1091
23.1.1	Structure and Common Switches	1092
23.2	Translating Java Sources	1092
23.2.1	The Java Compiler of the Java Development Kit	1092
23.2.2	Native Compilers	1092
23.3	The Java Runtime Environment	1093
23.3.1	Switches of the Java Virtual Machine	1093
23.3.2	The Difference between java.exe and javaw.exe	1095
23.4	Documentation Comments with Javadoc	1096
23.4.1	Setting a Documentation Comment	1096
23.4.2	Creating Documentation with the javadoc Tool	1098
23.4.3	HTML Tags in Documentation Comments*	1099
23.4.4	Generated Files	1099
23.4.5	Documentation Comments at a Glance*	1100
23.4.6	Javadoc and Doclets*	1101
23.4.7	Deprecated Types and Members	1101
23.4.8	Javadoc Verification with DocLint	1104
23.5	The JAR Archive Format	1105
23.5.1	Using the jar Utility	1105
23.5.2	The Manifest	1105
23.5.3	Launching Applications in Java Archives: Executable JAR Files	1106
23.6	Further Reading	1107

The Author	1109
------------------	------

Index	1111
-------------	------