

# Contents



<b>Figures</b>	<b>li</b>
<b>Tables</b>	<b>lv</b>
<b>Examples</b>	<b>lix</b>
<b>Foreword</b>	<b>lxvii</b>
<b>Preface</b>	<b>lxix</b>
Writing This Book	lxix
About This Book	lxx
Using This Book	lxxi
<i>Mock Exam</i>	lxxii
<i>Java SE Platform API Documentation</i>	lxxiii
Book Website	lxxiii
Request for Feedback	lxxiv
About the Authors	lxxiv
<i>Khalid A. Mughal</i>	lxxiv
<i>Vasily A. Strelnikov</i>	lxxiv
Acknowledgments	lxxv
<b>1 Basics of Java Programming</b>	<b>1</b>
1.1 The Java Ecosystem	3
Key Features of Java	4
<i>Multi-paradigm Programming</i>	4
<i>Bytecode Interpreted by the JVM</i>	5
<i>Architecture-Neutral and Portable Bytecode</i>	5
<i>Simplicity</i>	5
<i>Dynamic and Distributed</i>	5
<i>Robust and Secure</i>	6
<i>High Performance and Multithreaded</i>	6
1.2 Classes	7

	Declaring Members: Fields and Methods	8
1.3	Objects	9
	Class Instantiation, Reference Values, and References	9
1.4	Instance Members	10
	Invoking Methods	11
1.5	Static Members	11
1.6	Inheritance	15
1.7	Aggregation	17
	<i>Review Questions</i>	19
1.8	Sample Java Program	21
	Essential Elements of a Java Program	21
	Compiling a Program	22
	Running a Program	23
	Running a Single-File Source-Code Program	23
	The Java Shell Tool (jshe11)	24
1.9	Program Output	25
	Formatted Output	26
	<i>Review Questions</i>	28
<b>2</b>	<b>Basic Elements, Primitive Data Types, and Operators</b>	<b>29</b>
2.1	Basic Language Elements	30
	Lexical Tokens	30
	Identifiers	30
	<i>Examples of Legal Identifiers</i>	30
	<i>Examples of Illegal Identifiers</i>	31
	Keywords	31
	Separators	32
	Literals	32
	Integer Literals	32
	<i>Representing Integers</i>	34
	<i>Calculating Two's Complement</i>	35
	Floating-Point Literals	36
	<i>Examples of double Literals</i>	36
	<i>Examples of float Literals</i>	36
	Underscores in Numerical Literals	36
	<i>Examples of Legal Use of Underscores in Numerical Literals</i>	37
	<i>Examples of Illegal Use of Underscores in Numerical Literals</i>	37
	Boolean Literals	37
	Character Literals	37
	<i>Escape Sequences</i>	38
	String Literals	39
	Whitespace	40
	Comments	40
	<i>Single-Line Comment</i>	40
	<i>Multiple-Line Comment</i>	40

	<i>Documentation Comment</i>	41
2.2	Primitive Data Types	41
	The Integer Types	41
	The char Type	42
	The Floating-Point Types	43
	The boolean Type	43
2.3	Conversions	44
	Widening and Narrowing Primitive Conversions	44
	Widening and Narrowing Reference Conversions	45
	Boxing and Unboxing Conversions	46
	Other Conversions	47
2.4	Type Conversion Contexts	47
	Assignment Context	48
	Method Invocation Context	48
	Casting Context of the Unary Type Cast Operator ( <i>type</i> )	49
	Numeric Promotion Context	50
	<i>Unary Numeric Promotion</i>	50
	<i>Binary Numeric Promotion</i>	50
2.5	Precedence and Associativity Rules for Operators	51
2.6	Evaluation Order of Operands	52
	Left-Hand Operand Evaluation First	53
	Operand Evaluation before Operation Execution	53
	Left-to-Right Evaluation of Argument Lists	54
2.7	The Simple Assignment Operator =	55
	Assigning Primitive Values	55
	Assigning References	55
	Multiple Assignments	56
	Type Conversions in an Assignment Context	57
2.8	Arithmetic Operators: *, /, %, +, -	59
	Arithmetic Operator Precedence and Associativity	59
	Evaluation Order in Arithmetic Expressions	60
	Range of Numeric Values	60
	<i>Integer Arithmetic</i>	60
	<i>Floating-Point Arithmetic</i>	61
	Unary Arithmetic Operators: -, +	62
	Multiplicative Binary Operators: *, /, %	62
	<i>Multiplication Operator: *</i>	62
	<i>Division Operator: /</i>	62
	<i>Remainder Operator: %</i>	63
	Additive Binary Operators: +, -	64
	Numeric Promotions in Arithmetic Expressions	65
	Arithmetic Compound Assignment Operators: *=, /=, %=, +=, -=	67
2.9	The Binary String Concatenation Operator +	68
2.10	Variable Increment and Decrement Operators: ++, --	70
	The Increment Operator ++	70
	The Decrement Operator --	70

	<i>Review Questions</i>	72
2.11	Boolean Expressions	75
2.12	Relational Operators: <, <=, >, >=	75
2.13	Equality	76
	Primitive Data Value Equality: ==, !=	76
	Object Reference Equality: ==, !=	77
	Object Value Equality	78
2.14	Boolean Logical Operators: !, ^, &,	79
	Operand Evaluation for Boolean Logical Operators	80
	Boolean Logical Compound Assignment Operators: &=, ^=,  =	80
2.15	Conditional Operators: &&,	81
	Short-Circuit Evaluation	82
2.16	Integer Bitwise Operators: ~, &,  , ^	84
	<i>Examples of Bitwise Operator Application</i>	85
	Bitwise Compound Assignment Operators: &=, ^=,  =	86
	<i>Examples of Bitwise Compound Assignment</i>	87
2.17	Shift Operators: <<, >>, >>>	87
	The Shift-Left Operator <<	88
	The Shift-Right-with-Sign-Fill Operator >>	89
	The Shift-Right-with-Zero-Fill Operator >>>	90
	Shift Compound Assignment Operators: <<=, >>=, >>>=	91
	<i>Examples of Shift Compound Assignment Operators</i>	91
2.18	The Conditional Operator ?:	92
2.19	Other Operators: new, [], instanceof, ->	93
	<i>Review Questions</i>	94
<b>3</b>	<b>Declarations</b>	<b>97</b>
3.1	Class Declarations	99
3.2	Method Declarations	100
3.3	Statements	101
3.4	Variable Declarations	102
	Declaring and Initializing Variables	102
	Reference Variables	102
	Initial Values for Variables	103
	<i>Default Values for Fields</i>	103
	<i>Initializing Local Variables of Primitive Data Types</i>	104
	<i>Initializing Local Reference Variables</i>	105
	<i>Lifetime of Variables</i>	106
3.5	Instance Methods and the Object Reference this	106
3.6	Method Overloading	108
3.7	Constructors	109
	The Default Constructor	110
	Overloaded Constructors	112
3.8	Static Member Declarations	113
	Static Members in Classes	113

	Static Fields in Classes	113
	Static Methods in Classes	115
	<i>Review Questions</i>	116
3.9	Arrays	118
	Declaring Array Variables	118
	Constructing an Array	119
	Initializing an Array	120
	Using an Array	121
	Anonymous Arrays	123
	Multidimensional Arrays	124
3.10	Parameter Passing	127
	Passing Primitive Data Values	129
	Passing Reference Values	131
	Passing Arrays	133
	Array Elements as Actual Parameters	134
	final Parameters	136
3.11	Variable Arity Methods	137
	Calling a Variable Arity Method	138
	Variable Arity and Fixed Arity Method Calls	140
3.12	The main() Method	141
	Program Arguments	142
3.13	Local Variable Type Inference	143
	<i>Review Questions</i>	148
<b>4</b>	<b>Control Flow</b>	<b>153</b>
4.1	Selection Statements	154
	The Simple if Statement	154
	The if-else Statement	155
4.2	The switch Statement	157
	The switch Statement with the Colon (:) Notation	157
	The switch Statement with the Arrow (->) Notation	162
	Using Strings as case Constants	164
	Using Enum Constants as case Constants	165
4.3	The switch Expression	166
	The yield Statement	166
	The switch Expression with the Colon (:) Notation	166
	The switch Expression with the Arrow (->) Notation	168
	Local Variable Scope in the switch Body	171
	Summary of the switch Statement and the switch Expression	171
	<i>Review Questions</i>	172
4.4	Iteration Statements	174
4.5	The while Statement	175
4.6	The do-while Statement	176
4.7	The for(;;) Statement	176
4.8	The for(:) Statement	179

4.9	Transfer Statements	181
4.10	Labeled Statements	182
4.11	The <code>break</code> Statement	182
4.12	The <code>continue</code> Statement	185
4.13	The <code>return</code> Statement	186
	<i>Review Questions</i>	188
<b>5</b>	<b>Object-Oriented Programming</b>	<b>193</b>
5.1	Implementing Inheritance	195
	Relationships: <i>is-a</i> and <i>has-a</i>	198
	The Subtype–Supertype Relationship	199
	Overriding Instance Methods	200
	Covariant return in Overriding Methods	205
	Overriding versus Overloading	206
	Hiding Fields	207
	Hiding Static Methods	208
5.2	The Object Reference <code>super</code>	210
5.3	Chaining Constructors Using <code>this()</code> and <code>super()</code>	213
	The <code>this()</code> Constructor Call	213
	The <code>super()</code> Constructor Call	216
	<i>Review Questions</i>	219
5.4	Abstract Classes and Methods	223
	Abstract Classes	223
	<i>Declaring an abstract Class</i>	223
	<i>Extending an abstract Class</i>	224
	Abstract Methods in Classes	228
	<i>Declaring an abstract Method</i>	228
	<i>Overriding an abstract Method</i>	228
5.5	Final Declarations	230
	Final Classes	230
	Final Methods in Classes	231
	Final Variables	233
	Final Fields in Classes	234
	Final Local Variables in Methods	235
	<code>final</code> Parameters	236
	Definite Assignment Analysis for Final Variables	237
	<i>Review Questions</i>	239
5.6	Interfaces	242
	Defining Interfaces	242
	Abstract Methods in Interfaces	244
	Implementing Interfaces	245
	Extending Interfaces	248
	Interface References	250
	Default Methods in Interfaces	251
	<i>Overriding Default Methods</i>	252

	<i>Conflict Resolution for Default Methods</i>	253
	<i>Interface Evolution</i>	255
	Static Methods in Interfaces	255
	Private Methods in Interfaces	257
	Constants in Interfaces	259
	<i>Review Questions</i>	261
5.7	Arrays and Subtyping	264
	Arrays and Subtype Covariance	264
	Array Store Check	265
5.8	Reference Values and Conversions	266
5.9	Reference Value Assignment Conversions	266
5.10	Method Invocation Conversions Involving References	270
	Overloaded Method Resolution	271
5.11	Reference Casting and the instanceof Operator	274
	The Cast Operator	274
	The instanceof Type Comparison Operator	275
	The instanceof Pattern Match Operator	280
	<i>Scope of Pattern Variables</i>	282
5.12	Polymorphism	284
	<i>Review Questions</i>	288
5.13	Enum Types	293
	Declaring Type-Safe Enums	294
	Using Type-Safe Enums	295
	Declaring Enum Constructors and Members	296
	Implicit Static Methods for Enum Types	298
	Inherited Methods from the java.lang.Enum Class	298
	Extending Enum Types: Constant-Specific Class Bodies	299
	Enum Values in Exhaustive switch Expressions	303
	Declaring Type-Safe Enums, Revisited	304
5.14	Record Classes	305
	Record Class Basics	306
	<i>Restrictions on Record Declarations</i>	309
	Augmenting Basic Record Class Declaration	310
	Record Constructors	310
	<i>Normal Canonical Record Constructor</i>	311
	<i>Compact Canonical Record Constructor</i>	312
	<i>Normal Non-Canonical Record Constructors</i>	313
	Member Declarations	313
	Other Aspects of Record Classes	316
	<i>Implementing Interfaces</i>	316
	<i>Record Serialization</i>	316
	<i>Generic Record Classes</i>	317
	<i>Nested Record Classes</i>	317
	<i>Direct Permitted Subtypes of Sealed Interfaces</i>	317
5.15	Sealed Classes and Interfaces	317
	Sealed Classes	318

Sealed Interfaces	321
<i>Enum and Record Types as Permitted Direct Subtypes</i>	323
Deriving Permitted Direct Subtypes of a Sealed Supertype	323
Using Sealed Classes and Interfaces	324
<i>Review Questions</i>	325
<b>6 Access Control</b>	<b>331</b>
6.1 Design Principle: Encapsulation	332
6.2 Java Source File Structure	333
6.3 Packages	334
Defining Packages	336
Using Packages	337
<i>Importing Reference Types</i>	337
<i>Importing Static Members of Reference Types</i>	339
Compiling Code into Package Directories	343
Running Code from Packages	345
6.4 Searching for Classes on the Class Path	345
<i>Review Questions</i>	349
6.5 Access Modifiers	353
Access Modifiers for Top-Level Type Declarations	353
Access Modifiers for Class Members	356
<i>public Members</i>	358
<i>protected Members</i>	359
<i>Members with Package Access</i>	359
<i>private Members</i>	360
<i>Additional Remarks on Accessibility</i>	360
6.6 Scope Rules	361
Class Scope for Members	361
Block Scope for Local Variables	363
6.7 Implementing Immutability	364
<i>Review Questions</i>	369
<b>7 Exception Handling</b>	<b>371</b>
7.1 Stack-Based Execution and Exception Propagation	373
7.2 Exception Types	377
The java.lang.Exception Class	378
java.lang.ClassNotFoundException	378
java.io.IOException	378
java.io.EOFException	378
java.io.FileNotFoundException	380
java.io.NotSerializableException	380
java.sql.SQLException	380
java.text.ParseException	380
The java.lang.RuntimeException Class	380
java.lang.ArithmeticException	380



	java.lang.ArrayIndexOutOfBoundsException	380
	java.lang.ArrayStoreException	381
	java.lang.ClassCastException	381
	java.lang.IllegalArgumentException	381
	java.lang.IllegalThreadStateException	381
	java.lang.NumberFormatException	381
	java.lang.NullPointerException	381
	java.lang.UnsupportedOperationException	382
	java.time.DateTimeException	382
	java.time.format.DateTimeParseException	382
	java.util.MissingResourceException	382
	The java.lang.Error Class	382
	Checked and Unchecked Exceptions	383
	Defining Customized Exceptions	383
7.3	Exception Handling: try, catch, and finally	384
	The try Block	385
	The catch Clause	385
	The finally Clause	391
7.4	The throw Statement	395
7.5	The throws Clause	396
	Overriding the throws Clause	399
	<i>Review Questions</i>	401
7.6	The Multi-catch Clause	406
	Rethrowing Exceptions	410
	Chaining Exceptions	413
7.7	The try-with-resources Statement	415
	Concise try-with-resources Statement	420
	Implementing the AutoCloseable Interface	421
	Suppressed Exceptions	424
7.8	Advantages of Exception Handling	425
	<i>Review Questions</i>	426
<b>8</b>	<b>Selected API Classes</b>	<b>433</b>
8.1	Overview of the java.lang Package	435
8.2	The Object Class	435
8.3	The Wrapper Classes	439
	Common Wrapper Class Utility Methods	440
	<i>Converting Primitive Values to Wrapper Objects</i>	440
	<i>Converting Strings to Wrapper Objects</i>	441
	<i>Converting Wrapper Objects to Strings</i>	441
	<i>Converting Primitive Values to Strings</i>	441
	<i>Converting Wrapper Objects to Primitive Values</i>	442
	<i>Wrapper Comparison, Equality, and Hash Code</i>	442
	Numeric Wrapper Classes	444
	<i>Converting Numeric Wrapper Objects to Numeric Primitive Types</i>	444

	<i>Converting Strings to Numeric Values</i>	444
	<i>Converting Integer Values to Strings in Different Notations</i>	445
	The Character Class	447
	The Boolean Class	447
	<i>Converting Strings to Boolean Values</i>	447
	<i>Review Questions</i>	448
8.4	The String Class	449
	Internal Representation of Strings	449
	Creating and Initializing Strings	450
	<i>Immutability</i>	450
	<i>String Internment</i>	450
	<i>String Constructors</i>	451
	The CharSequence Interface	454
	Reading Characters from a String	455
	Reading Lines from a String	455
	Comparing Strings	457
	Character Case in a String	458
	Concatenation of Strings	458
	<i>Repeating Strings</i>	459
	Joining of CharSequence Objects	459
	Searching for Characters and Substrings in Strings	461
	Extracting Substrings from Strings	463
	Converting Primitive Values and Objects to Strings	464
	Transforming a String	465
	Indenting Lines in a String	465
	Formatted Strings	467
	Text Blocks	468
	<i>Basic Text Blocks</i>	468
	<i>Using Escape Sequences in Text Blocks</i>	469
	<i>Incidental Whitespace</i>	472
8.5	The StringBuilder Class	474
	Constructing String Builders	474
	<i>Mutability</i>	474
	<i>String Builder Constructors</i>	475
	Reading Characters from String Builders	475
	Searching for Substrings in String Builders	476
	Extracting Substrings from String Builders	476
	Constructing Strings from String Builders	477
	Comparing String Builders	477
	Modifying String Builders	477
	<i>Appending Characters to a String Builder</i>	477
	<i>Inserting Characters in a String Builder</i>	478
	<i>Replacing Characters in a String Builder</i>	478
	<i>Deleting Characters in a String Builder</i>	479
	<i>Reversing Characters in a String Builder</i>	479
	Controlling String Builder Capacity	480

	<i>Review Questions</i>	482
8.6	The Math Class	488
	Miscellaneous Rounding Functions	489
	Exponential Functions	491
	Exact Integer Arithmetic Functions	491
	Pseudorandom Number Generator	493
8.7	The Random Class	493
	Determining the Range	494
	Generating the Same Sequence of Pseudorandom Numbers	495
8.8	Using Big Numbers	495
	BigDecimal Constants	496
	Constructing BigDecimal Numbers	497
	Computing with BigDecimal Numbers	497
	<i>Review Questions</i>	498
<b>9</b>	<b>Nested Type Declarations</b>	<b>501</b>
9.1	Overview of Nested Type Declarations	503
9.2	Static Member Types	507
	Declaring Static Member Types	507
	Using Qualified Name of Nested Types	509
	Instantiating Static Member Classes	510
	Importing Static Member Types	510
	Accessing Members in Enclosing Context	511
9.3	Non-Static Member Classes	514
	Declaring Non-Static Member Classes	514
	Instantiating Non-Static Member Classes	515
	Accessing Members in Enclosing Context	517
	<i>Accessing Hidden Members</i>	518
	<i>Inheritance Hierarchy and Enclosing Context</i>	520
	<i>Review Questions</i>	523
9.4	Local Classes	525
	Declaring Local Classes	525
	Accessing Declarations in Enclosing Context	527
	<i>Accessing Local Declarations in the Enclosing Block</i>	527
	<i>Accessing Members in the Enclosing Class</i>	528
	Instantiating Local Classes	530
9.5	Static Local Types	533
9.6	Anonymous Classes	534
	Declaring Anonymous Classes	534
	Extending an Existing Class	535
	Implementing an Interface	537
	Instantiating Anonymous Classes	538
	<i>Referencing Instances of an Anonymous Class</i>	538
	Accessing Declarations in Enclosing Context	538
	<i>Accessing Local Declarations in the Enclosing Block</i>	539

	<i>Accessing Members in the Enclosing Class</i>	539
	<i>Review Questions</i>	541
<b>10</b>	<b>Object Lifetime</b>	<b>545</b>
10.1	Garbage Collection	547
10.2	Reachable Objects	547
10.3	Facilitating Garbage Collection	550
10.4	Invoking Garbage Collection Programmatically	551
	<i>Review Questions</i>	552
10.5	Initializers	554
10.6	Field Initializer Expressions	554
	Declaration Order of Initializer Expressions	555
	Exception Handling and Initializer Expressions	557
10.7	Static Initializer Blocks	559
	Declaration Order of Static Initializers	559
	Exception Handling in Static Initializer Blocks	563
10.8	Instance Initializer Blocks	565
	Declaration Order of Instance Initializers	565
	Exception Handling and Instance Initializer Blocks	568
10.9	Constructing Initial Object State	569
	<i>Review Questions</i>	572
<b>11</b>	<b>Generics</b>	<b>577</b>
11.1	Introducing Generics	579
11.2	Generic Types and Parameterized Types	581
	Generic Types	581
	<i>Some Restrictions on the Use of Type Parameters in a Generic Type</i>	582
	Parameterized Types	583
	The Diamond Operator (<>)	584
	Parameterized Local Variable Type Inference	585
	Generic Interfaces	586
	Extending Generic Types	587
	Raw Types and Unchecked Warnings	589
11.3	Collections and Generics	592
11.4	Wildcards	593
	The Subtype Covariance Problem with Parameterized Types	593
	Wildcard Types	594
	Subtype Covariance: ? extends Type	595
	Subtype Contravariance: ? super Type	596
	Subtype Bivariance: ?	596
	Subtype Invariance: Type	597
	Some Restrictions on Wildcard Types	598
11.5	Using References of Wildcard Parameterized Types	598
	Generic Reference Assignment	598
	Using Parameterized References to Call Set and Get Methods	600

	<i>Raw Type References</i>	602
	<i>Unbounded Wildcard References: &lt;?&gt;</i>	602
	<i>Upper Bounded Wildcard References: &lt;? extends Type&gt;</i>	603
	<i>Lower Bounded Wildcard References: &lt;? super Type&gt;</i>	603
	<i>Unbounded Type References: &lt;Type&gt;</i>	604
11.6	Bounded Type Parameters	605
	Multiple Bounds	606
11.7	Generic Methods and Constructors	607
	Declaring Generic Methods	607
	Calling Generic Methods	608
11.8	Implementing a Simplified Generic Stack	612
	<i>Review Questions</i>	614
11.9	Wildcard Capture	619
	Capture Conversion	621
11.10	Flexibility with Wildcard Parameterized Types	621
	Nested Wildcards	621
	Wildcard Parameterized Types as Formal Parameters	623
	Recursive Type Bounds Revisited	625
11.11	Type Erasure	627
	Translation by Type Erasure	627
	Bridge Methods	629
11.12	Implications for Overloading and Overriding	629
	Method Signatures Revisited	630
	Implications for Overloading	630
	Implications for Overriding	631
	<i>The @Override Annotation</i>	632
	<i>Overriding Methods from Non-Generic Supertype</i>	633
	<i>Overriding Methods from Generic Supertype</i>	633
	<i>Genericity and Inherited Methods</i>	634
	<i>Overriding Abstract Methods from Multiple Superinterfaces</i>	635
11.13	Limitations and Restrictions on Generic Types	637
	Reifiable Types	637
	Implications for the instanceof operator	638
	Implications for Casting	639
	Implications for Arrays	641
	Implications for Non-Reifiable Variable Arity Parameter	644
	Implications for Exception Handling	646
	Implications for Nested Classes	647
	Other Implications	649
	<i>Enum Types</i>	649
	<i>Class Literals</i>	650
	<i>Review Questions</i>	650

## 12 Collections, Part I: ArrayList<E>

657

### 12.1 Lists

658

	Overview of the Java Collections Framework	658
12.2	Declaring References and Constructing ArrayLists	660
	Creating Unmodifiable Lists	663
12.3	Modifying an ArrayList<E>	665
	Adding Elements	665
	Replacing Elements	666
	Removing Elements	666
	Modifying Capacity	667
	Primitive Values and ArrayLists	669
12.4	Querying an ArrayList<E>	669
12.5	Iterating Over an ArrayList<E>	671
12.6	Converting an ArrayList<E> to an Array	672
12.7	Creating List Views	673
	Comparing Unmodifiable Lists and List Views	675
12.8	Arrays versus ArrayLists	676
	<i>Review Questions</i>	681
<b>13</b>	<b>Functional-Style Programming</b>	<b>689</b>
13.1	Functional Interfaces	691
	Declaring a Functional Interface	691
	<i>Overriding Abstract Methods from Multiple Interfaces, Revisited</i>	693
	Selected Interfaces in the Java SE Platform API	694
13.2	Lambda Expressions	695
	Lambda Parameters	696
	Lambda Body	697
	Type Checking and Execution of Lambda Expressions	699
	Accessing Members in an Enclosing Class	701
	Accessing Local Variables in the Enclosing Context	703
13.3	Lambda Expressions and Anonymous Classes	704
	Filtering Criteria Defined by Anonymous Classes	706
	Filtering Criteria Defined by Lambda Expressions	707
	Lambda Expressions versus Anonymous Classes	708
	<i>Implementation</i>	708
	<i>Scope</i>	708
	<i>Accessing Inherited Members from the Inheritance Hierarchy</i>	708
	<i>Accessing Local Declarations in the Enclosing Block</i>	708
	<i>Accessing Members in the Enclosing Class</i>	709
	<i>Best Practices</i>	709
	<i>Review Questions</i>	709
13.4	Overview of Built-In Functional Interfaces	711
13.5	Suppliers	715
	Primitive Type Specializations of Supplier<T>	717
13.6	Predicates	719
	Composing Predicates	721
	Primitive Type Specializations of Predicate<T>	722

	Two-Arity Specialization of Predicate<T>: BiPredicate<T, U>	723
13.7	Consumers	725
	Composing Consumers	727
	Primitive Type Specializations of Consumer<T>	727
	Two-Arity Specialization of Consumer<T>: BiConsumer<T, U>	728
	Primitive Type Specializations of BiConsumer<T, U>	728
13.8	Functions	729
	Composing Functions	732
	Primitive Type Specializations of Function<T, R>	733
13.9	Two-Arity Specialization of Function<T, R>: BiFunction<T, U, R>	734
	Composing Two-Arity Functions	735
	Primitive Type Specializations of BiFunction<T, U, R>	736
13.10	Extending Function<T, T>: UnaryOperator<T>	736
	Primitive Type Specializations of UnaryOperator<T>	737
13.11	Extending BiFunction<T, T, T>: BinaryOperator<T>	738
	Primitive Type Specializations of BinaryOperator<T>	739
13.12	Currying Functions	739
13.13	Method and Constructor References	740
	Static Method References	742
	Bounded Instance Method References	744
	Unbounded Instance Method References	745
	Constructor References	747
	Array Constructor References	748
13.14	Contexts for Defining Lambda Expressions	750
	Declaration and Assignment Statements	750
	Method and Constructor Calls	750
	Expressions in return Statements	750
	Ternary Conditional Expressions	751
	Cast Expressions	751
	Nested Lambda Expressions	752
	<i>Review Questions</i>	752
<b>14</b>	<b>Object Comparison</b>	<b>757</b>
14.1	The Objects Class	759
14.2	Implementing the equals() Method	760
	Equivalence Relation of the equals() Method	762
	<i>Reflexivity</i>	762
	<i>Symmetry</i>	763
	<i>Transitivity</i>	763
	<i>Consistency</i>	763
	<i>null Comparison</i>	764
	Checklist for Implementing the equals() Method	765
	<i>Method Overriding Signature</i>	765
	<i>Reflexivity Test</i>	766
	<i>Correct Argument Type</i>	766

	<i>Field Comparisons</i>	766
14.3	Implementing the hashCode() Method	769
	General Contract of the hashCode() Method	772
	Heuristics for Implementing the hashCode() Method	772
14.4	Implementing the java.lang.Comparable<E> Interface	777
14.5	Implementing the java.util.Comparator<E> Interface	785
	<i>Review Questions</i>	790
<b>15</b>	<b>Collections: Part II</b>	<b>797</b>
15.1	The Java Collections Framework	799
	Core Interfaces	799
	Implementations	802
15.2	Collections	806
	Basic Operations	807
	Bulk Operations	807
	Iterating Over a Collection	808
	<i>Using an Explicit Iterator on a Collection</i>	810
	<i>Using the forEachRemaining() Method</i>	812
	<i>Using the for(:) Loop to Iterate Over a Collection</i>	812
	<i>Using the forEach() Method to Iterate Over a Collection</i>	813
	Filtering	813
	<i>Using the remove() Method of an Iterator to Filter a Collection</i>	813
	<i>Using the removeIf() Method to Filter a Collection</i>	814
	Streams	814
	Array Operations	815
15.3	Lists	817
	The List<E> Interface	817
	The ArrayList<E>, LinkedList<E>, and Vector<E> Classes	818
15.4	Sets	820
	The Set<E> Interface	821
	<i>Creating Unmodifiable Sets</i>	821
	The HashSet<E> and LinkedHashSet<E> Classes	823
15.5	Sorted Sets and Navigable Sets	827
	The SortedSet<E> Interface	827
	The NavigableSet<E> Interface	828
	The TreeSet<E> Class	829
15.6	Queues	831
	The Queue<E> Interface	831
	The PriorityQueue<E> and LinkedList<E> Classes	832
15.7	Dequeues	838
	The Deque<E> Interface	838
	The ArrayDeque<E> and LinkedList<E> Classes	839
	<i>Review Questions</i>	843
15.8	Maps	847
	The Map<K,V> Interface	847



	Basic Key-Based Operations	848
	Creating Unmodifiable Maps	849
	Advanced Key-Based Operations	852
	Bulk Operations	856
	Collection Views	856
15.9	Map Implementations	857
15.10	Sorted Maps and Navigable Maps	862
	The SortedMap<K, V> Interface	862
	The NavigableMap<K, V> Interface	863
	The TreeMap<K, V> Class	864
	<i>Review Questions</i>	869
15.11	The Collections Class	874
	Unmodifiable Views of Collections	874
	Ordering Elements in Lists	876
	Searching in Collections	878
	Replacing Elements in Collections	880
15.12	The Arrays Class	882
	Sorting Arrays	882
	Searching in Arrays	884
	Comparing Arrays	886
	<i>Array Equality</i>	886
	<i>Array Comparison</i>	887
	<i>Array Mismatch</i>	889
	Miscellaneous Utility Methods in the Arrays Class	890
	<i>Review Questions</i>	892
<b>16</b>	<b>Streams</b>	<b>897</b>
16.1	Introduction to Streams	899
16.2	Running Example: The CD Record Class	900
16.3	Stream Basics	902
	Composing Stream Pipelines	904
	Executing Stream Pipelines	905
	Comparing Collections and Streams	906
	Overview of API for Data Processing Using Streams	907
	<i>The Stream Interfaces</i>	907
	<i>The Collectors Class</i>	908
	<i>The Optional Classes</i>	908
	<i>The Numeric Summary Statistics Classes</i>	908
16.4	Building Streams	909
	Aspects to Consider When Creating Streams	909
	<i>Sequential or Parallel Stream</i>	909
	<i>Ordered or Unordered Stream</i>	909
	<i>Finite or Infinite Stream</i>	910
	<i>Object or Numeric Stream</i>	910
	The Empty Stream	911

	Streams from Specified Values	911
	Using Generator Functions to Build Infinite Streams	913
	<i>Generate</i>	913
	<i>Iterate</i>	913
	Concatenating Streams	914
	Streams from Collections	915
	Streams from Arrays	916
	Building a Numeric Stream with a Range	917
	Numeric Streams Using the Random Class	918
	Streams from a CharSequence	919
	Streams from a String	920
	Streams from a BufferedReader	920
	Streams from Factory Methods in the Files Class	921
	Summary of Stream Building Methods	922
16.5	Intermediate Stream Operations	923
	Aspects of Streams, Revisited	924
	<i>Stream Mapping</i>	924
	<i>Lazy Execution</i>	925
	<i>Short-circuit Evaluation</i>	926
	<i>Stateless and Stateful Operations</i>	926
	<i>Order of Intermediate Operations</i>	926
	<i>Non-interfering and Stateless Behavioral Parameters</i>	928
	Filtering	928
	<i>Filtering Using a Predicate</i>	931
	<i>Taking and Dropping Elements Using Predicates</i>	931
	<i>Selecting Distinct Elements</i>	933
	<i>Skipping Elements in a Stream</i>	934
	<i>Truncating a Stream</i>	935
	Examining Elements in a Stream	939
	Mapping: Transforming Streams	940
	Flattening Streams	942
	Replacing Each Element of a Stream with Multiple Elements	946
	Sorted Streams	949
	Setting a Stream as Unordered	951
	Execution Mode of a Stream	952
	Converting between Stream Types	954
	<i>Mapping between Object Streams</i>	954
	<i>Mapping an Object Stream to a Numeric Stream</i>	954
	<i>Mapping a Numeric Stream to an Object Stream</i>	955
	<i>Mapping between Numeric Streams</i>	956
	Summary of Intermediate Stream Operations	957
16.6	The Optional Class	959
	Declaring and Returning an Optional	960
	Creating an Optional	960
	Querying an Optional	962
	Numeric Optional Classes	964

16.7	Terminal Stream Operations	966
	Consumer Action on Stream Elements	967
	Matching Elements	969
	Finding the First or Any Element	971
	Counting Elements	973
	Finding Min and Max Elements	973
	Implementing Functional Reduction: The <code>reduce()</code> Method	974
	<i>Parallel Functional Reduction</i>	982
	Implementing Mutable Reduction: The <code>collect()</code> Method	984
	<i>Parallel Mutable Reduction</i>	986
	Collecting to an Array	991
	Collecting to a List	992
	Functional Reductions Exclusive to Numeric Streams	993
	<i>Summation</i>	994
	<i>Averaging</i>	994
	<i>Summarizing</i>	995
	Summary of Terminal Stream Operations	997
16.8	Collectors	998
	Collecting to a Collection	1000
	Collecting to a List	1001
	Collecting to a Set	1001
	Collecting to a Map	1001
	<i>Collecting to a ConcurrentMap</i>	1004
	Joining	1004
	Grouping	1005
	<i>Multilevel Grouping</i>	1008
	<i>Grouping to a ConcurrentMap</i>	1010
	Partitioning	1010
	<i>Multilevel Partitioning</i>	1012
	Filtering Adapter for Downstream Collectors	1013
	Mapping Adapter for Downstream Collectors	1014
	Flat Mapping Adapter for Downstream Collectors	1015
	Finishing Adapter for Downstream Collectors	1018
	Downstream Collectors for Functional Reduction	1018
	<i>Counting</i>	1019
	<i>Finding Min/Max</i>	1019
	<i>Summing</i>	1021
	<i>Averaging</i>	1022
	<i>Summarizing</i>	1023
	<i>Reducing</i>	1024
	Summary of Static Factory Methods in the Collectors Class	1026
16.9	Parallel Streams	1030
	Building Parallel Streams	1030
	Parallel Stream Execution	1031
	Factors Affecting Performance	1031
	<i>Benchmarking</i>	1031

<i>Side Effects</i>	1036
<i>Ordering</i>	1036
<i>Autoboxing and Unboxing of Numeric Values</i>	1037
<i>Review Questions</i>	1037

<b>17</b>	<b>Date and Time</b>	<b>1045</b>
17.1	Date and Time API Overview	1046
17.2	Working with Dates and Times	1049
	Creating Dates and Times	1049
	<i>The LocalDateTime Class</i>	1050
	<i>The LocalDate Class</i>	1051
	<i>The LocalDateTime Class</i>	1051
	Accessing Fields in Dates and Times	1054
	Comparing Dates and Times	1056
	Creating Modified Copies of Dates and Times	1057
	Temporal Arithmetic with Dates and Times	1062
17.3	Using Temporal Units and Temporal Fields	1066
	Temporal Units	1066
	Temporal Fields	1068
17.4	Working with Instants	1071
	Creating Instants	1072
	Accessing Temporal Fields in an Instant	1074
	Comparing Instants	1075
	Creating Modified Copies of Instants	1076
	Temporal Arithmetic with Instants	1077
	Converting Instants	1079
17.5	Working with Periods	1079
	Creating Periods	1080
	Accessing Date Units in a Period	1081
	Comparing Periods for Equality	1082
	Creating Modified Copies of Periods	1083
	Temporal Arithmetic with Periods	1083
17.6	Working with Durations	1087
	Creating Durations	1087
	Accessing Time Units in a Duration	1090
	Comparing Durations	1091
	Creating Modified Copies of Durations	1092
	Temporal Arithmetic with Durations	1092
	Differences between Periods and Durations	1095
17.7	Working with Time Zones and Daylight Savings	1095
	Time Zones and Zone Offsets	1096
	The ZonedDateTime Class	1097
	Creating Zoned Date-Time	1098
	Accessing Fields of Zoned Date-Time	1100
	Creating Modified Copies of Zoned Date-Time	1101

Temporal Arithmetic with Zoned Date-Time	1103
Daylight Savings	1105
17.8 Converting Date and Time Values to Legacy Date	1111
Review Questions	1112
<b>18 Localization</b>	<b>1119</b>
18.1 Using Locales	1120
18.2 Properties Files	1124
Creating a Property List in a Properties File	1124
18.3 Bundling Resources	1126
Resource Bundle Families	1126
Creating Resource Bundles	1127
Property Resource Files	1127
Locating, Loading, and Searching Resource Bundles	1128
Locating Locale-Specific Resources	1132
Constructing the Parent Chain of a Result Resource Bundle	1134
Searching in Resource Bundles	1135
Review Questions	1137
18.4 Core API for Formatting and Parsing of Values	1139
18.5 Formatting and Parsing Number, Currency, and Percentage Values	1141
Static Factory Methods to Create a Formatter	1141
Formatting Number, Currency, and Percentage Values	1141
Accounting Currency Formatting	1143
Parsing Strings to Number, Currency, and Percentage Values	1144
Compact Number Formatting	1145
Compact Number Parsing	1147
Specifying the Number of Digits	1147
Customizing Decimal Number Formatting	1150
18.6 Formatting and Parsing Date and Time	1153
Default Formatters for Date and Time Values	1155
Predefined Formatters for Date and Time Values	1155
Style-Based Formatters for Date and Time Values	1157
Pattern-Based Formatters for Date and Time Values	1160
18.7 Formatting and Parsing Messages	1165
Format Patterns	1166
Formatting Compound Messages	1169
Conditional Formatting	1172
Parsing Values Using Patterns	1177
Additional Support for Formatting in the Java SE Platform APIs	1179
Review Questions	1180
<b>19 Java Module System</b>	<b>1187</b>
19.1 Making the Case for Modules	1189
Stronger Encapsulation	1189
Fine-Grained Visibility of Public Types	1189

	<i>Reusability, Maintainability, and Optimization</i>	1189
	Reliable Configuration	1189
	Better Performance	1190
	Scalable Applications	1190
	Improved Security	1190
	Backward Compatibility	1190
19.2	The Modular JDK	1190
	Reasons for Modularity	1191
	Goals of the Modular JDK	1191
	Overview of the Modular JDK	1191
	Module Graph of the JDK	1193
	The <code>java.base</code> Module	1193
	The <code>java.se</code> Module	1194
19.3	Module Basics	1194
	Module Declaration	1194
	Module Name	1197
	Module Graph	1198
	Readability and Accessibility	1199
	<i>Accessibility Rules for Members</i>	1200
	Implied Module Dependencies	1201
	<i>The requires transitive Directive</i>	1201
	Qualified Export	1202
19.4	Overview of Module Directives	1203
19.5	Creating a Modular Application	1205
	Running Example: Dispensing Canned Advice	1205
	Creating the Application Directory Structure	1206
	Creating an Exploded Module Directory	1207
	<i>Creating the Source Files in the Exploded Modules</i>	1208
19.6	Compiling and Running a Modular Application	1213
	Individual Module Compilation	1213
	Multi-Module Compilation	1214
	Running a Modular Application from Exploded Modules	1215
19.7	Creating JAR Files	1215
	Creating Modular JARs	1216
	Running an Application from a Modular JAR	1218
19.8	Open Modules and the <code>opens</code> Directive	1218
	Qualified Opens Directive	1222
	Open Modules	1223
19.9	Services	1223
	Specifying the Service Interface	1226
	Implementing Service Providers	1226
	Implementing the Service Locator	1227
	Implementing the Service Consumer	1230
	Compiling and Running a Service	1231
19.10	Creating Runtime Images	1231
19.11	Categories of Modules	1233

	The Unnamed Module	1233
	Automatic Modules	1234
	<i>Scheme for Naming Automatic Modules</i>	1234
	Explicit Modules	1235
	Named Modules	1235
19.12	Migrating to Modules	1236
	Bottom-Up Strategy for Code Migration	1236
	Top-Down Strategy for Code Migration	1237
19.13	Exploring Modules	1238
	Listing the Contents of a JAR	1239
	Extracting the Contents of a JAR	1239
	Listing All Observable Modules	1240
	Describing the Module Descriptor of a JAR	1241
	Viewing Dependencies	1242
	<i>Viewing Package-Level Dependencies</i>	1242
	<i>Viewing Module Dependencies</i>	1243
	<i>Viewing Class-Level Dependencies</i>	1244
19.14	Summary of Selected Operations with the JDK Tools	1245
	Selected Options for the javac Tool	1246
	Selected Options for the java Tool	1247
	Selected Options for the jar Tool	1248
	Selected Options for the jdeps Tool	1249
	Selected Options for the jlink Tool	1250
	Final Remarks on Options for the JDK Tools	1251
	<i>Review Questions</i>	1251
<b>20</b>	<b>Java I/O: Part I</b>	<b>1259</b>
20.1	Input and Output	1261
20.2	Byte Streams: Input Streams and Output Streams	1262
	File Streams	1264
	I/O Filter Streams	1266
	Reading and Writing Binary Values	1266
	<i>Writing Binary Values to a File</i>	1267
	<i>Reading Binary Values from a File</i>	1268
20.3	Character Streams: Readers and Writers	1270
	Print Writers	1274
	<i>Writing Text Representation of Primitive Values and Objects</i>	1275
	<i>Writing Formatted Values</i>	1275
	Writing Text Files	1275
	Reading Text Files	1278
	Using Buffered Writers	1279
	Using Buffered Readers	1280
	The Standard Input, Output, and Error Streams	1283
	Comparison of Byte Streams and Character Streams	1284
20.4	The Console Class	1284

	<i>Review Questions</i>	1287
20.5	Object Serialization	1289
	The <code>ObjectOutputStream</code> Class	1290
	The <code>ObjectInputStream</code> Class	1291
	<i>Efficient Record Serialization</i>	1292
	Selective Serialization	1294
	<i>Both the Compound Object and Its Constituents Are Serializable</i>	1294
	<i>Transient Fields Are Not Serializable</i>	1296
	<i>Serializable Compound Object with Non-Serializable Constituents</i>	1296
	Customizing Object Serialization	1297
	Serialization and Inheritance	1299
	<i>Superclass Is Serializable</i>	1300
	<i>Superclass Is Not Serializable</i>	1300
	Serialization and Versioning	1302
	<i>Review Questions</i>	1306
<b>21</b>	<b>Java I/O: Part II</b>	<b>1315</b>
21.1	Characteristics of a Hierarchical File System	1317
	Hierarchical File Systems	1317
	Absolute and Relative Paths	1318
	<i>Current and Parent Directory Designators</i>	1319
	Symbolic Links	1319
21.2	Creating Path Objects	1319
	Creating Path Objects with the <code>Path.of()</code> Method	1321
	Creating Path Objects with the <code>Paths.get()</code> Method	1322
	Creating Path Objects Using the Default File System	1322
	Interoperability with the <code>java.io.File</code> Legacy Class	1322
	Interoperability with the <code>java.net.URI</code> Class	1323
21.3	Working with Path Objects	1324
	Querying Path Objects	1324
	Converting Path Objects	1327
	<i>Converting a Path to an Absolute Path</i>	1328
	<i>Normalizing a Path</i>	1329
	<i>Resolving Two Paths</i>	1330
	<i>Constructing the Relative Path between Two Paths</i>	1331
	<i>Link Option</i>	1332
	<i>Converting a Path to a Real Path</i>	1332
	Comparing Path Objects	1333
21.4	Operations on Directory Entries	1334
	Characteristics of Methods in the <code>Files</code> Class	1334
	<i>Handling System Resources</i>	1334
	<i>Handling Exceptions</i>	1335
	<i>Handling Symbolic Links</i>	1335
	<i>Specifying Variable Arity Parameters</i>	1335
	<i>Executing Atomic Operations</i>	1335



	Determining the Existence of a Directory Entry	1336
	Uniqueness of a Directory Entry	1337
	Deleting Directory Entries	1338
	Copy Options	1339
	Copying Directory Entries	1339
	<i>Copy and Replace Directory Entries</i>	1340
	<i>Copy Files Using I/O Streams</i>	1342
	Moving and Renaming Directory Entries	1343
	<i>Moving Directory Entries</i>	1344
	<i>Renaming Directory Entries</i>	1344
	<i>Atomic Move</i>	1344
21.5	Reading and Writing Files Using Paths	1345
	Open Options	1345
	Reading and Writing Character Data	1346
	<i>Reading and Writing Character Data Using Buffered I/O Streams</i>	1346
	<i>Reading and Writing Character Data Using Path Objects</i>	1348
	Reading and Writing Bytes	1350
	<i>Reading and Writing Bytes Using I/O Streams</i>	1350
	<i>Reading and Writing Bytes Using Path Objects</i>	1351
21.6	Managing File Attributes	1352
	Accessing Individual File Attributes	1352
	<i>Determining the File Size</i>	1354
	<i>Determining the Kind of Directory Entry</i>	1355
	<i>Determining File Accessibility</i>	1355
	<i>Timestamp for Last Modification Time</i>	1355
	<i>Accessing the Owner</i>	1356
	<i>Handling File Permissions</i>	1357
	<i>Accessing File Attributes through View and Attribute Names</i>	1359
	Bulk Operations to Retrieve File Attributes	1360
	<i>The BasicFileAttributes Interface</i>	1362
	<i>The PosixFileAttributes Interface</i>	1364
	<i>The DosFileAttributes Interface</i>	1365
	<i>File Attribute Views</i>	1365
	<i>The BasicFileAttributeView Interface</i>	1366
	<i>The PosixFileAttributeView Interface</i>	1368
	<i>The DosFileAttributeView Interface</i>	1370
21.7	Creating Directory Entries	1371
	Creating Regular and Temporary Files	1371
	<i>Creating Regular Files</i>	1372
	<i>Creating Temporary Files</i>	1373
	<i>Creating Symbolic Links</i>	1374
	Creating Regular and Temporary Directories	1375
	<i>Creating Regular Directories</i>	1375
	<i>Creating Temporary Directories</i>	1377
21.8	Stream Operations on Directory Entries	1378
	Reading Text Lines Using a Functional Stream	1378

Directory Hierarchy Traversal	1379
<i>Traversal Strategies</i>	1380
Listing the Immediate Entries in a Directory	1381
File Visit Option	1381
Walking the Directory Hierarchy	1382
<i>Limiting Traversal by Depth Specification</i>	1384
<i>Handling Symbolic Links</i>	1384
<i>Copying an Entire Directory</i>	1384
Searching for Directory Entries	1386
<i>Review Questions</i>	1388
 <b>22 Concurrency: Part I</b>	 <b>1399</b>
22.1 Threads and Concurrency	1401
Multitasking	1401
Understanding Concurrency and Parallelism	1402
22.2 Runtime Organization for Thread Execution	1403
22.3 Creating Threads	1404
Implementing the Runnable Interface	1405
Extending the Thread Class	1409
Types of Threads	1412
<i>Review Questions</i>	1413
22.4 Thread Lifecycle	1414
Objects, Monitors, and Locks	1414
Thread States	1415
Thread Priorities	1419
Thread Scheduler	1420
Starting a Thread	1421
Running and Yielding	1421
Executing Synchronized Code	1422
<i>Acquiring the Object Lock</i>	1423
<i>Synchronized Methods</i>	1423
<i>Reentrant Synchronization</i>	1425
<i>Synchronization on Class Lock</i>	1425
<i>Synchronized Statements</i>	1426
Interrupt Handling	1428
Sleeping and Waking Up	1430
Thread Coordination	1431
<i>Waiting</i>	1432
<i>Notified</i>	1433
<i>Timed-out</i>	1434
<i>Interrupted</i>	1434
<i>Spurious Wakeup</i>	1434
<i>Using Wait and Notify</i>	1435
Joining	1437
Blocking for I/O	1440

	Thread Termination	1440
	<i>Controlling a Thread</i>	1441
	Deprecated Thread Methods	1443
22.5	Thread Issues	1443
	Liveness and Fairness	1443
	Deadlock	1443
	Livelock	1445
	Starvation	1447
	Memory Consistency Errors	1449
	<i>Happens-Before Relationship</i>	1449
	<i>Review Questions</i>	1450
<b>23</b>	<b>Concurrency: Part II</b>	<b>1455</b>
23.1	Utility Classes TimeUnit and ThreadLocalRandom	1457
	The TimeUnit Enum Type	1457
	The ThreadLocalRandom Class	1458
23.2	The Executor Framework	1459
	The Executor Interface	1460
	The Executors Class	1461
	The ExecutorService Interface	1463
	Using an Executor Service	1465
	<i>Controlling Shutdown and Termination of an Executor Service</i>	1469
	Defining Tasks That Return a Result	1471
	Representing Task Results	1472
	Submitting Tasks and Handling Task Results	1473
	<i>Submitting Tasks Individually</i>	1474
	<i>Invoking Tasks Collectively</i>	1476
	The ScheduledExecutorService Interface	1476
	<i>Scheduling One-Shot Tasks</i>	1476
	<i>Scheduling Periodic Tasks</i>	1478
	Task Cancellation	1481
23.3	The Fork/Join Framework	1484
23.4	Writing Thread-Safe Code	1487
	Immutability	1489
	Volatile Fields	1490
	Atomic Variables	1493
	Intrinsic Locking Revisited	1496
	Programmatic Locking	1497
	<i>Reentrant Lock</i>	1497
	<i>Reentrant Read-Write Lock</i>	1502
23.5	Special-Purpose Synchronizers	1507
	Cyclic Barrier	1507
	Count-Down Latch	1510
23.6	Synchronized Collections and Maps	1513
	Serial Access	1515

	Compound Mutually Exclusive Operations	1517
23.7	Concurrent Collections and Maps	1519
	Concurrent Collections	1522
	<i>The ConcurrentSkipListSet&lt;E&gt; Class</i>	1524
	<i>The ConcurrentLinkedQueue&lt;E&gt; Class</i>	1525
	<i>The ConcurrentLinkedDeque&lt;E&gt; Class</i>	1525
	Concurrent Maps	1528
	<i>The ConcurrentMap&lt;K, V&gt; Interface</i>	1530
	<i>The ConcurrentNavigableMap&lt;K, V&gt; Interface</i>	1531
	Blocking Queues	1533
	<i>The BlockingQueue&lt;E&gt; Interface</i>	1536
	<i>The TransferQueue&lt;E&gt; Interface</i>	1537
	<i>The BlockingDeque&lt;E&gt; Interface</i>	1537
	Copy-on-Write Collections	1540
	<i>The CopyOnWriteArrayList&lt;E&gt; Class</i>	1541
	<i>The CopyOnWriteArraySet&lt;E&gt; Class</i>	1542
	Review Questions	1543

## 24 Database Connectivity

**1549**

24.1	Introduction to Relational Databases	1550
	Relational Tables	1550
	Basic SQL Statements	1552
	<i>The CREATE Statement</i>	1552
	<i>The INSERT Statement</i>	1553
	<i>The SELECT Statement</i>	1553
	<i>The UPDATE Statement</i>	1554
	<i>The DELETE Statement</i>	1554
24.2	Introduction to JDBC	1555
	Closing JDBC Resources	1556
24.3	Establishing a Database Connection	1557
	JDBC URL	1558
	Getting a Database Connection	1558
	<i>Connecting to the musicDB Database</i>	1558
	Legacy JDBC Driver Management	1559
24.4	Creating and Executing SQL Statements	1560
	Basic Statement	1561
	<i>SQL Injection</i>	1563
	Prepared Statement	1564
	<i>Substitutional Parameterization</i>	1564
	Callable Statement	1568
	<i>Syntax of Calling Stored Procedures and Functions</i>	1568
	<i>IN Parameters</i>	1568
	<i>OUT Parameters</i>	1569
	<i>INOUT Parameters</i>	1570
24.5	Processing Query Results	1572

	Traversing the Result Set	1572
	<i>Result Set Navigation Methods</i>	1573
	Optimizing the Fetch Size	1574
	Extracting Column Values from the Current Row	1575
	<i>Result Set Methods to Extract Column Values of the Current Row</i>	1575
24.6	Customizing Result Sets	1577
	Result Set Type	1578
	Result Set Concurrency	1579
	Result Set Holdability	1579
24.7	Discovering Database and ResultSet Metadata	1581
24.8	Implementing Transaction Control	1583
	<i>Review Questions</i>	1586
<b>25</b>	<b>Annotations</b>	<b>1593</b>
25.1	Basics of Annotations	1595
25.2	Declaring Annotation Types	1596
	Declaring Annotation Type Elements	1598
	Element Modifiers for Annotation Type Elements	1599
	Element Type of Annotation Type Elements	1599
	Defaults for Annotation Type Elements	1600
25.3	Applying Annotations	1601
	Applying Normal Annotations	1601
	Applying Marker Annotations	1603
	Applying Single-Element Annotations	1603
25.4	Meta-Annotations	1605
	The @Retention Meta-Annotation	1605
	The @Target Meta-Annotation	1607
	The @Inherited Meta-Annotation	1612
	The @Documented Meta-Annotation	1612
	The @Repeatable Meta-Annotation	1613
25.5	Selected Standard Annotations	1615
	The @Override Annotation	1616
	The @FunctionalInterface Annotation	1617
	The @Deprecated Annotation	1618
	The @SuppressWarnings Annotation	1620
	The @SafeVarargs Annotation	1623
	Other Annotations	1625
25.6	Processing Annotations	1625
	<i>Review Questions</i>	1631
<b>26</b>	<b>Secure Coding</b>	<b>1637</b>
26.1	Application Security Overview	1638
26.2	Security Threat Categories	1640
	Denial-of-Service (DoS) Attacks	1640
	Sensitive Information Leakage	1641

	<i>Value Obfuscation</i>	1641
	<i>Data Encryption</i>	1642
	Code Injection and Input Validation	1643
	<i>SQL Injection</i>	1643
	Code Corruption	1644
	<i>Encapsulation</i>	1644
	<i>Data Integrity</i>	1644
	<i>Robustness</i>	1645
	<i>Extensibility</i>	1645
	<i>Immutability</i>	1645
26.3	Java Security Policies	1646
	Executing Privileged Code	1647
26.4	Additional Security Guidelines	1648
	Accessing the File System	1648
	Deserialization	1649
	Class Design	1649
	<i>Review Questions</i>	1649
<b>A</b>	<b>Taking the Java SE 17 and Java SE 11 Developer Exams</b>	<b>1653</b>
A.1	Preparing for the Exam	1653
A.2	Registering for the Online Proctored Exam	1655
	Obtaining an Oracle Exam Attempt	1655
	Scheduling for the Online Proctored Exam	1655
	Candidate Agreement	1655
	After Taking the Exam	1655
A.3	How the Online Proctored Exam Is Conducted	1656
	Requirements for the Online Proctored Exam	1656
	The Exam Program	1656
	Utilizing the Allotted Time	1657
	The Exam Result	1657
A.4	The Questions	1658
	Assumptions About the Exam Questions	1658
	Types of Questions Asked	1658
	Types of Answers Expected	1659
	Topics Covered by the Questions	1659
<b>B</b>	<b>Exam Topics: Java SE 17 Developer</b>	<b>1661</b>
<b>C</b>	<b>Exam Topics: Java SE 11 Developer</b>	<b>1667</b>
<b>D</b>	<b>Annotated Answers to Review Questions</b>	<b>1673</b>
<b>E</b>	<b>Mock Exam: Java SE 17 Developer</b>	<b>1749</b>

<b>F</b>	<b>Annotated Answers to Mock Exam</b>	<b>1779</b>
<b>G</b>	<b>Java Logging API Overview</b>	<b>1789</b>
G.1	Purpose of the Logging API	1789
G.2	Configuring Logging	1790
G.3	Writing Log Messages	1791
G.4	Applying Guarded Logging	1793
G.5	Summary	1793
	<b>Index</b>	<b>1795</b>

