

Contents

Preface **xxv**

Before You Begin **lix**

I **Intro, Test-Driving a Java Application, and Generative AI** **I**

1.1	Introduction	2
1.2	Hardware	4
1.3	Java	7
1.4	A Brief Review of Object Orientation	8
1.5	Java Application Programming Interface (API) and Open-Source Libraries	11
1.6	Typical Java Program Development Environment	12
1.7	Test-Driving a Java Program with the Java Development Kit (JDK)	15
1.7.1	Compiling and Running the Program with <code>javac</code> and <code>java</code>	15
1.7.2	Running the Program's Source-Code File Directly with the <code>java</code> Command	18
1.8	Internet, World Wide Web, the Cloud and IoT	18
1.8.1	The Internet: A Network of Networks	19
1.8.2	The World Wide Web: Making the Internet User-Friendly	19
1.8.3	The Cloud	20
1.8.4	The Internet of Things (IoT)	20
1.8.5	Edge Computing	21
1.8.6	Mashups	21
1.9	Metaverse	22
1.9.1	Virtual Reality (VR)	22
1.9.2	Augmented Reality (AR)	22
1.9.3	Mixed Reality	23
1.9.4	Blockchain	24
1.9.5	Bitcoin and Cryptocurrency	24
1.9.6	Ethereum	24
1.9.7	Non-Fungible Tokens (NFTs)	25
1.9.8	Web3	25
1.10	Software Development Technologies	25
1.11	Data Analytics and Data Science	27
1.12	How Big Is Big Data?	28

viii Contents

1.12.1	Big-Data Analytics	30
1.12.2	Data Science and Big Data Are Making a Difference: Use Cases	31
1.13	AI—at the Intersection of Computer Science and Data Science	32
1.13.1	Artificial Intelligence (AI)	32
1.13.2	Artificial General Intelligence (AGI)	33
1.13.3	Artificial Intelligence Milestones	33
1.13.4	Machine Learning	34
1.13.5	Deep Learning	35
1.13.6	Reinforcement Learning	35
1.14	Generative AI	35
1.15	Wrap-Up	41

2 Intro to Java Programming 47

2.1	Introduction	48
2.2	Your First Program in Java: Printing a Line of Text	48
2.2.1	Compiling the Program	52
2.2.2	Executing the Program	53
2.3	Modifying Your First Program	54
2.4	Displaying Text with <code>printf</code>	56
2.5	Another Program: Adding Integers	57
2.6	Arithmetic	60
2.7	Decision Making: Equality and Relational Operators	62
2.8	Objects-Natural Case Study: Creating and Using Objects of the Java API's <code>String</code> Class	65
2.9	Wrap-Up	68

3 Control Statements: Part I 69

3.1	Introduction	70
3.2	Control Structures	70
3.2.1	Sequence Structure	71
3.2.2	Selection Statements	72
3.2.3	Iteration Statements	72
3.2.4	Summary of Control Statements	72
3.3	<code>if</code> Single-Selection Statement	73
3.4	<code>if...else</code> Double-Selection Statement	74
3.4.1	Nested <code>if...else</code> Statements	74
3.4.2	Blocks	75
3.4.3	Conditional Operator (<code>?:</code>)	77
3.5	<code>while</code> Iteration Statement	77
3.6	Counter-Controlled Iteration	79
3.6.1	Implementing the Algorithm	79
3.6.2	Integer Division and Truncation	81
3.6.3	Arithmetic Overflow	81
3.6.4	Input Validation	82

3.7	Sentinel-Controlled Iteration	83
3.7.1	Implementing Sentinel-Controlled Iteration	83
3.7.2	Explicitly and Implicitly Converting Between Primitive Types	85
3.7.3	Formatting Floating-Point Numbers	86
3.8	Nested Control Statements	87
3.8.1	Problem Statement	87
3.8.2	Implementing the Program	88
3.9	Compound Assignment Operators	91
3.10	Increment and Decrement Operators	91
3.11	Primitive Types	94
3.12	Objects-Natural Case Study: Super-Sized Integers	94
3.13	Wrap-Up	98

4 Control Statements: Part 2 99

4.1	Introduction	100
4.2	Essentials of Counter-Controlled Iteration	100
4.3	for Iteration Statement	101
4.4	Examples Using the for Statement	104
4.5	Summing the Even Integers from 2 to 20 with the for Statement	105
4.6	Calculating Compound Interest with the for Statement	106
4.7	do...while Iteration Statement	109
4.8	switch Multiple-Selection Statement	111
4.9	break and continue Statements	116
4.10	Logical Operators	118
4.10.1	Conditional AND (&&) Operator	119
4.10.2	Conditional OR () Operator	120
4.10.3	Short-Circuit Evaluation of Complex Conditions	120
4.10.4	Boolean Logical AND (&) and Inclusive OR () Operators	121
4.10.5	Boolean Logical Exclusive OR (^) Operator	121
4.10.6	Logical Complement (!) Operator	121
4.10.7	Example: Generating Logical-Operator Truth Tables	122
4.11	Objects-Natural Case Study: Precise Monetary Calculations with Java API Class <code>BigDecimal</code>	124
4.12	Wrap-Up	128

5 Methods 131

5.1	Introduction	132
5.2	Declaring Methods	132
5.3	Notes on Declaring and Using Methods	136
5.4	Case Study: Die Rolling Simulation with Random-Number Generation	137
5.5	Case Study: A Game of Chance; Introducing <code>enums</code>	142
5.6	Scope of Declarations	147
5.7	Method Overloading	150
5.8	Class <code>Math</code> : <code>static</code> Methods and Variables	152

x **Contents**

5.9	Java API Packages	154
5.10	Method-Call Stack and Activation Records	156
5.11	Argument Promotion and Casting	159
5.12	Objects-Natural Case Study: Java Date/Time API	161
5.12.1	Data/Time Manipulations	162
5.12.2	Running the Program for a Different Locale Than Your Own	166
5.13	Wrap-Up	167

6 **Arrays and ArrayLists** **169**

6.1	Introduction	170
6.2	Primitive Types vs. Reference Types	171
6.3	Arrays	171
6.4	Declaring and Creating Arrays	172
6.5	Creating and Initializing an Array	173
6.6	Array Initializers	175
6.7	Calculating Array Element Values	176
6.8	Totaling Array Elements	177
6.9	Intro to Visualization: Using a Bar Chart to Display Array Data Graphically	177
6.10	Using Array Elements as Counters	179
6.11	Analyzing Survey Results	181
6.12	Exception Handling	183
6.12.1	The try Statement	183
6.12.2	The catch Block	184
6.13	Enhanced for Statement: Totaling Array Elements	184
6.14	Passing Arrays to Methods	186
6.15	Pass-By-Value vs. Pass-By-Reference	188
6.16	Multidimensional Arrays	189
6.17	Variable-Length Argument Lists	193
6.18	Command-Line Arguments	194
6.19	Class Arrays	197
6.20	Objects-Natural Case Study: Intro to Collections and Class ArrayList	199
6.21	Wrap-Up	204

7 **Strings, NLP and Regex: Generative AI Foundations** **207**

7.1	Introduction	208
7.2	Fundamentals of Characters and Strings	209
7.3	Class String	210
7.3.1	Creating String Objects	210
7.3.2	String Methods length, charAt and getChars	211
7.3.3	Comparing Strings	212
7.3.4	Locating Characters and Substrings in Strings	218
7.3.5	Extracting Substrings from Strings	220
7.3.6	Concatenating Strings	221

7.3.7	Miscellaneous String Methods	221
7.3.8	String Method <code>valueOf</code>	223
7.4	Class <code>StringBuilder</code>	225
7.4.1	Creating <code>StringBuilder</code> Objects	225
7.4.2	Methods <code>length</code> , <code>capacity</code> , <code>setLength</code> and <code>ensureCapacity</code>	226
7.4.3	Methods <code>charAt</code> , <code>setCharAt</code> , <code>getChars</code> and <code>reverse</code>	227
7.4.4	<code>append</code> Methods	228
7.4.5	Insertion and Deletion Methods	230
7.5	Class <code>Character</code>	232
7.5.1	Character Testing and Case Conversion Methods	232
7.5.2	Character/Digit Conversions	235
7.5.3	Other Character Methods	236
7.6	Tokenizing Strings	237
7.7	Intro to Natural Language Processing (NLP)—at the Root of Generative AI	238
7.8	Objects-Natural Case Study: Intro to Regular Expressions in NLP	240
7.8.1	Matching Complete Strings to Patterns	241
7.8.2	Replacing Substrings	245
7.8.3	Searching for Matches with Classes <code>Pattern</code> and <code>Matcher</code>	246
7.8.4	Simple Data Wrangling Steps Used to Prepare Text for Training NLP and Generative AI Models	248
7.9	Objects-Natural Security Case Study: <code>pMa5tfEKwk59dTvC04Ft1IFQz9mEXnkfYXZwxk4ujGE=</code>	252
7.10	Wrap-Up	260

8 Real-World Modeling with Custom Classes 261

8.1	Introduction	262
8.2	Instance Variables, <i>set</i> Methods and <i>get</i> Methods	263
8.2.1	Account Class with an Instance Variable, and <i>set</i> and <i>get</i> Methods	263
8.2.2	<code>AccountTest</code> Class That Creates and Uses an Account Object	265
8.2.3	Compiling and Executing a Program with Multiple Classes	268
8.2.4	Account UML Class Diagram	268
8.2.5	Notes on Class <code>AccountTest</code>	269
8.2.6	Software Engineering with <code>private</code> Instance Variables and <code>public</code> <i>set</i> and <i>get</i> Methods	270
8.3	Default and Explicit Instance Variable Initialization	271
8.4	Account Class: Initializing Objects with Constructors	271
8.4.1	Declaring an Account Constructor for Custom Object Initialization	272
8.4.2	Class <code>AccountTest</code> : Initializing New Account Objects	272
8.5	Account Class with a Balance	274
8.5.1	Account Class with a <code>BigDecimal</code> balance Instance Variable	274
8.5.2	<code>AccountTest</code> Class	276
8.6	Case Study: Card Shuffling and Dealing Simulation	278
8.7	Case Study: Time Class	283
8.8	Controlling Access to Members	286
8.9	Referring to the Current Object's Members with the <code>this</code> Reference	287

xii Contents

8.10	Case Study: Time Class Overloaded Constructors	289
8.11	Default and No-Argument Constructors	296
8.12	Notes on <i>Set</i> and <i>Get</i> Methods	296
8.13	Composition	297
8.14	enum Types	300
8.15	Garbage Collection	303
8.16	static Class Members	303
8.17	static Import	307
8.18	final Instance Variables	308
8.19	Package Access	309
8.20	record Classes	310
	8.20.1 Pattern Matching for switch Expressions	313
	8.20.2 Decomposing records in switch Expressions	314
8.21	Wrap-Up	315

9 Real-World Modeling with Inheritance, Polymorphism & Interfaces 317

9.1	Introduction	318
9.2	Superclasses and Subclasses	320
9.3	Relationship Between Superclasses and Subclasses	322
	9.3.1 Creating and Using a SalariedEmployee Class	322
	9.3.2 Creating a SalariedEmployee/SalariedCommissionEmployee Inheritance Hierarchy	326
9.4	Class Object	332
9.5	Intro to Polymorphism: Polymorphic Video Game	332
9.6	Demonstrating Polymorphic Behavior	333
9.7	abstract Classes and Methods	336
9.8	Case Study: Payroll System Using Polymorphism	338
	9.8.1 abstract Superclass Employee	339
	9.8.2 Concrete Subclass SalariedEmployee	340
	9.8.3 Concrete Subclass CommissionEmployee	342
	9.8.4 Polymorphic Processing, Operator instanceof and Downcasting	343
9.9	final Methods and Classes	347
9.10	Issues with Constructors Calling Instance Methods	348
9.11	Creating and Using Interfaces	349
	9.11.1 Developing a Payable Hierarchy	350
	9.11.2 Interface Payable	351
	9.11.3 Class Invoice	352
	9.11.4 Modifying Class Employee to Implement Interface Payable	353
	9.11.5 Using Interface Payable to Process Invoices and Employees Polymorphically	355
	9.11.6 Some Common Java API Interfaces	356
9.12	Other Interface Features	357
	9.12.1 default Interface Methods	357
	9.12.2 static Interface Methods	358

9.12.3	Functional Interfaces	358
9.12.4	<code>private</code> Interface Methods	358
9.13	Program to an Interface, Not an Implementation	359
9.13.1	<code>CompensationModel</code> Interface	360
9.13.2	An <code>Employee</code> Has a <code>CompensationModel</code> —Composition and Dependency Injection	360
9.13.3	<code>CompensationModel</code> Implementations	362
9.13.4	Testing the <code>CompensationModel</code> Hierarchy	364
9.13.5	Dependency Injection Design Benefits	366
9.13.6	<code>interfaces</code> vs. <code>abstract</code> Classes	366
9.14	<code>sealed</code> Classes and Interfaces	367
9.15	<code>private</code> Constructors	370
9.16	<code>protected</code> Members	370
9.17	Wrap-Up	372

10 Exception Handling: A Deeper Look 375

10.1	Introduction	376
10.2	Example: Divide by Zero without Exception Handling	378
10.3	Example: Handling <code>ArithmeticExceptions</code> and <code>InputMismatchExceptions</code>	380
10.4	Java Exception Hierarchy	384
10.5	Checked vs. Unchecked Exceptions	386
10.6	<code>finally</code> Block	388
10.7	Stack Unwinding and Obtaining Information from an Exception	390
10.8	Chained Exceptions	394
10.9	Declaring Custom Exceptions	396
10.10	Preconditions and Postconditions	397
10.11	Assertions	398
10.12	<code>try-with-Resources</code> Statement: Automatic Resource Deallocation	399
10.13	Unnamed Variables in <code>catch</code> Handlers	401
10.14	Wrap-Up	402

11 Files, I/O Streams, JSON Serialization & CSV Files 405

11.1	Introduction	406
11.2	Files and Streams	407
11.3	Using NIO Classes and Interfaces to Get File and Directory Information	409
11.4	Sequential Text Files	413
11.4.1	Creating a Sequential Text File	413
11.4.2	Reading Data from a Sequential Text File	416
11.4.3	Updating Sequential Files	417
11.5	Case Study: JavaScript Object Notation (JSON) Serialization	418
11.5.1	Serializing a <code>List<Account></code>	420
11.5.2	Deserializing a <code>List<Account></code>	422

xiv Contents

11.6	Case Study: Processing a JSON Response from a Web Service	424
11.7	Case Study: Creating and Reading CSV Files	432
11.7.1	Creating a CSV File	433
11.7.2	Reading a CSV File	435
11.8	Case Study: Reading and Analyzing a CSV File Containing the <i>Titanic</i> Disaster Dataset	437
11.9	Objects-Natural Security Case Study: RSA Public-Key Cryptography	445
11.10	Wrap-Up	452

12 Generic Collections **455**

12.1	Introduction	456
12.2	Collections Overview	457
12.3	Type-Wrapper Classes	458
12.4	Boxing and Unboxing	459
12.5	Lists	459
12.5.1	ArrayList and Iterator	460
12.5.2	LinkedList	463
12.5.3	Views into Collections and Arrays Method asList	466
12.6	Collections Methods	468
12.6.1	Method sort	469
12.6.2	Method shuffle	472
12.6.3	Methods reverse, fill, copy, max and min	475
12.6.4	Method binarySearch	477
12.6.5	Methods addAll, frequency and disjoint	478
12.7	Class PriorityQueue and Interface Queue	480
12.8	Hash Tables	481
12.9	Sets	483
12.10	Maps	486
12.11	Convenience Factory Methods for Creating Immutable Collections	489
12.12	Concurrent Collections	492
12.13	Wrap-Up	492

13 Generic Classes and Methods: A Deeper Look **495**

13.1	Introduction	496
13.2	Motivation for Generic Methods	496
13.3	Generic Methods: Implementation and Compile-Time Translation	498
13.3.1	Generic Method printArray	498
13.3.2	Type Parameter Section of a Generic Method	499
13.3.3	Testing the Generic Method	500
13.3.4	Erasure at Compilation Time	501
13.4	Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type	501
13.5	Overloading Generic Methods	505
13.6	Generic Classes	505
13.6.1	Parameterized Types	505

13.6.2	Implementing a Generic Stack Class	506
13.6.3	Testing the Generic Stack Class	507
13.6.4	Refactoring the Test Program to Manipulate Stack<E> Objects with Generic Methods	511
13.7	Wildcards in Methods That Accept Type Parameters	513
13.7.1	Summing a List<Number>	513
13.7.2	Reimplementing Method sum with a Wildcard Type Argument in Its Parameter	515
13.8	Wrap-Up	517

14 Functional Programming with Lambdas & Streams 519

14.1	Introduction	520
14.2	Streams and Reduction	521
14.2.1	Summing the Integers from 1 through 10 with a for Loop	522
14.2.2	External Iteration with for Is Error Prone	522
14.2.3	Summing with a Stream and Reduction	522
14.2.4	Internal Iteration	524
14.3	Mapping and Lambdas	525
14.3.1	Lambda Expressions	526
14.3.2	Lambda Syntax	526
14.3.3	Intermediate and Terminal Operations	527
14.4	Filtering	528
14.5	How Elements Move Through Stream Pipelines	530
14.6	Method References	531
14.6.1	Creating an IntStream of Random Values	532
14.6.2	Performing a Task on Each Stream Element with forEach and a Method Reference	533
14.6.3	Mapping Integers to String Objects with mapToObj	533
14.6.4	Concatenating Strings with collect	534
14.7	IntStream Operations	534
14.7.1	Creating an IntStream and Displaying Its Values	535
14.7.2	Terminal Operations count, min, max, sum and average	535
14.7.3	Terminal Operation reduce	537
14.7.4	Sorting IntStream Values	539
14.8	Functional Interfaces	540
14.9	Lambdas: A Deeper Look	541
14.10	Stream<Integer> Manipulations	542
14.10.1	Creating a Stream<Integer>, Sorting It and Collecting the Results	543
14.10.2	Filtering a Stream and Storing the Results for Later Use	544
14.10.3	Filtering and Sorting a Stream and Collecting the Results	545
14.10.4	Sorting Previously Collected Results	545
14.11	Stream<String> Manipulations	546
14.11.1	Mapping Strings to Uppercase	546

xvi **Contents**

14.11.2 Filtering Strings Then Sorting Them in Case-Insensitive Ascending Order	547
14.11.3 Filtering Strings Then Sorting Them in Case-Insensitive Descending Order	547
14.12 Stream<Employee> Manipulations	548
14.12.1 Creating and Displaying a List<Employee>	549
14.12.2 Filtering Employees with Salaries in a Specified Range	550
14.12.3 Sorting Employees By Multiple Fields	552
14.12.4 Mapping Employees to Unique-Last-Name Strings	554
14.12.5 Grouping Employees By Department	555
14.12.6 Counting the Number of Employees in Each Department	556
14.12.7 Summing and Averaging Employee Salaries	556
14.13 Creating a Stream<String> from a File	558
14.14 Streams of Random Values	561
14.15 Infinite Streams	562
14.16 Additional Notes on Interfaces	564
14.17 Wrap-Up	564

15 JavaFX Graphical User Interfaces: Part I **567**

15.1 Introduction	568
15.2 JavaFX Scene Builder	569
15.3 JavaFX Application Window Structure	570
15.4 Welcome Application: Displaying Text and an Image	571
15.4.1 Opening Scene Builder and Creating the File Welcome.fxml	571
15.4.2 Adding an Image to the Folder Containing Welcome.fxml	573
15.4.3 Creating a VBox Layout Container	573
15.4.4 Configuring the VBox Layout Container	573
15.4.5 Adding and Configuring a Label	574
15.4.6 Adding and Configuring an ImageView	574
15.4.7 Previewing the Welcome GUI	576
15.5 Tip Calculator Application: Intro to Event Handling	577
15.5.1 Test-Driving the Tip Calculator	577
15.5.2 Technologies Overview	579
15.5.3 Building the Application's GUI	582
15.5.4 TipCalculator Class	588
15.5.5 TipCalculatorController Class; Implementing an Event Handler with a Lambda	590
15.6 Features Covered in the Other JavaFX Chapters	594
15.7 Wrap-Up	594

16 JavaFX GUI: Part 2 **597**

16.1 Introduction	598
16.2 Laying Out Nodes in a Scene Graph	598
16.3 Painter Application: RadioButtons, Mouse Events and Shapes	600
16.3.1 Technologies Overview	601

16.3.2	Creating the <code>Painter.fxml</code> File	602
16.3.3	Building the GUI	603
16.3.4	<code>Painter</code> Subclass of <code>Application</code>	605
16.3.5	<code>PainterController</code> Class	606
16.4	Color Chooser Application: Property Bindings and Property Listeners	610
16.4.1	Technologies Overview	611
16.4.2	Building the GUI	612
16.4.3	<code>ColorChooser</code> Subclass of <code>Application</code>	613
16.4.4	<code>ColorChooserController</code> Class	614
16.5	Cover Viewer Application: Data-Driven GUIs with JavaFX Collections	616
16.5.1	Technologies Overview	617
16.5.2	Building the GUI	617
16.5.3	<code>CoverViewer</code> Subclass of <code>Application</code>	619
16.5.4	<code>CoverViewerController</code> Class	619
16.6	Cover Viewer Application: Customizing <code>ListView</code> Cells	621
16.6.1	Technologies Overview	622
16.6.2	Copying the <code>CoverViewer</code> Application	622
16.6.3	<code>ImageTableCell</code> Custom <code>ListView</code> Cell Layout	622
16.6.4	<code>CoverViewerController</code> Class	624
16.7	<code>FileChooser</code> and <code>DirectoryChooser</code> Dialogs	625
16.8	Other JavaFX Capabilities and JavaFX Accessibility	631
16.9	JavaFX Updates	633
16.10	JavaFX Resources and Libraries	634
16.11	Wrap-Up	636

17 JavaFX Graphics and Multimedia 639

17.1	Introduction	640
17.2	Controlling Fonts with Cascading Style Sheets (CSS)	641
17.2.1	CSS That Styles the GUI	641
17.2.2	FXML That Defines the GUI—Introduction to XML Markup	645
17.2.3	Referencing the CSS File from FXML	647
17.2.4	Specifying the <code>VBox</code> 's Style Class	648
17.2.5	Programmatically Loading CSS	648
17.3	Displaying Two-Dimensional Shapes	649
17.3.1	Defining Two-Dimensional Shapes with FXML	649
17.3.2	CSS That Styles the Two-Dimensional Shapes	651
17.4	<code>PolyLines</code> , <code>Polygons</code> and <code>Paths</code>	655
17.4.1	GUI and CSS	655
17.4.2	<code>PolyShapesController</code> Class	656
17.5	Transforms	660
17.6	Playing Video with <code>Media</code> , <code>MediaPlayer</code> and <code>MediaView</code>	662
17.6.1	<code>VideoPlayer</code> GUI	663
17.6.2	<code>VideoPlayerController</code> Class	665
17.7	Transition Animations	669
17.7.1	<code>TransitionAnimations.fxml</code>	669
17.7.2	<code>TransitionAnimationsController</code> Class	671

xviii Contents

17.8	Timeline Animations	675
17.9	Frame-by-Frame Animation with <code>AnimationTimer</code>	677
17.10	CSS Transitions	680
17.11	Drawing on a Canvas	685
17.12	Three-Dimensional Shapes	690
17.13	FXGL: A Brief Intro to Game Programming with JavaFX	694
17.14	Wrap-Up	695

18 Concurrency: Platform Threads to Virtual Threads 697

18.1	Introduction	698
18.2	<code>sort/parallelSort</code> Timings with the Date/Time API	700
18.3	Sequential vs. Parallel Streams	703
18.4	Creating and Executing Platform Threads with the Executor Framework	707
18.5	Project Loom Overview	711
18.6	Creating and Executing Virtual Threads with the Executor Framework	712
18.6.1	Launching Virtual Threads with a Thread-Per-Task Executor	713
18.6.2	Other Ways to Launch Virtual Threads	715
18.7	Profiling Platform vs. Virtual Threads	717
18.7.1	<code>Future</code> and <code>Callable</code> Interfaces	717
18.7.2	Threading Approaches We'll Use	718
18.7.3	Launching and Timing Large Numbers of Concurrent Tasks	718
18.7.4	Testing the Threading Approaches	720
18.7.5	Performance Analysis	721
18.8	Structured Concurrency and Scoped Values	723
18.9	Thread Synchronization Overview	731
18.10	Producer/Consumer Relationship with <code>ArrayBlockingQueue</code>	733
18.10.1	Synchronization and State Dependence	733
18.10.2	Class <code>ArrayBlockingQueue</code>	734
18.10.3	Overview of the Producer/Consumer Example	734
18.10.4	Interface <code>Buffer</code>	735
18.10.5	Class <code>BlockingBuffer</code>	735
18.10.6	Class <code>BlockingBufferTest</code>	736
18.10.7	Bounded Buffers	739
18.10.8	Other Concurrent Collections	740
18.11	Multithreading in JavaFX	741
18.11.1	Performing Computations in a Worker Thread: Fibonacci Numbers	742
18.11.2	Processing Intermediate Results: Sieve of Eratosthenes	747
18.12	Wrap-Up	753

19 Building API-Based Java Generative AI Applications 757

19.1	Introduction	758
------	--------------	-----

19.2	OpenAI APIs	760
19.2.1	Some of OpenAI's Models	760
19.2.2	OpenAI API Capabilities	760
19.2.3	Get an OpenAI Developer Account	761
19.2.4	OpenAI Developer API Key	762
19.3	Setting Up a Java Environment	763
19.3.1	Simple-OpenAI Open Source Library: Connecting to OpenAI from Java	763
19.3.2	IntelliJ Maven-Based Java Project	763
19.4	Text Generation Via Chat Completions	764
19.4.1	Text Summarization	764
19.4.2	Sentiment Analysis	768
19.4.3	Accessible Image Descriptions	771
19.4.4	Language Detection and Translation	774
19.4.5	Code Generation	776
19.4.6	Named Entity Recognition (NER) and Structured Outputs	780
19.5	Speech Synthesis and Speech Recognition	783
19.5.1	English Speech-to-Text for Audio Transcription	783
19.5.2	Text-to-Speech	784
19.6	Image Generation	786
19.7	Video	789
19.7.1	Generating Video Closed Captions	789
19.7.2	Sora	792
19.7.3	Other Video GenAIs	794
19.8	Moderation	794
19.9	Class <code>OpenAIUtilities</code>	797
19.9.1	Builder Design Pattern	797
19.9.2	<code>package</code> and <code>import</code> Statements	798
19.9.3	Nested <code>record</code> Class <code>Message</code>	799
19.9.4	<code>SimpleOpenAI</code> Client Object	799
19.9.5	<code>chat</code> Method	799
19.9.6	Helper Method <code>makeChatMessages</code>	801
19.9.7	<code>describeImage</code> Method	802
19.9.8	<code>translate</code> Method	803
19.9.9	<code>chatWithStructuredOutput</code> Method	804
19.9.10	<code>speechToText</code> Method	806
19.9.11	<code>textToSpeech</code> Method	807
19.9.12	<code>image</code> Method	808
19.9.13	<code>speechToVTT</code> Method	809
19.9.14	<code>checkPrompt</code> Method	811
19.10	Wrap-Up	812
20	Accessing Databases with JDBC and SQLite	829
20.1	Introduction	830
20.2	Relational Databases	831

xx Contents

20.3	Setting Up the SQLite RDBMS	832
20.4	A books Database	832
20.4.1	SELECT Queries	836
20.4.2	WHERE Clause	837
20.4.3	ORDER BY Clause	838
20.4.4	Merging Data from Multiple Tables with INNER JOIN	839
20.4.5	INSERT INTO Statement	839
20.4.6	UPDATE Statement	840
20.4.7	DELETE FROM Statement	841
20.5	Connecting to and Querying a Database with JDBC	842
20.5.1	Automatic Driver Discovery	843
20.5.2	Connecting to the Database	843
20.5.3	Creating a Statement for Executing Queries	844
20.5.4	Executing a Query	844
20.5.5	Processing a Query's ResultSet	844
20.6	Querying the books Database	846
20.6.1	DisplayQueryResults App's GUI	846
20.6.2	DisplayQueryResultsController Class	848
20.7	PreparedStatement	852
20.7.1	AddressBook App That Uses PreparedStatement	853
20.7.2	record Class Person	854
20.7.3	Class PersonQueries	854
20.7.4	AddressBook GUI	858
20.7.5	Class AddressBookController	860
20.8	Stored Procedures	864
20.9	Transaction Processing	864
20.10	Wrap-Up	865

21 Java Platform Module System 867

21.1	Introduction	868
21.2	Module Declarations	873
21.2.1	requires	873
21.2.2	requires transitive—Implied Readability	873
21.2.3	exports and exports...to	874
21.2.4	uses	874
21.2.5	provides...with	874
21.2.6	open, opens and opens...to	874
21.2.7	Restricted Keywords	876
21.3	Modularized Welcome App	876
21.3.1	Welcome App's Structure	876
21.3.2	Class Welcome	879
21.3.3	module-info.java	880
21.3.4	Module-Dependency Graph	880
21.3.5	Compiling a Module	880
21.3.6	Running an App from a Module's Exploded Folders	882

21.3.7	Packaging a Module into a Modular JAR File	883
21.3.8	Running the <code>Welcome</code> App from a Modular JAR File	884
21.3.9	Aside: Classpath vs. Module Path	884
21.4	Creating and Using a Custom Module	885
21.4.1	Exporting a Package for Use in Other Modules	885
21.4.2	Using a Class from a Package in Another Module	886
21.4.3	Compiling and Running the Example	888
21.4.4	Packaging the App into Modular JAR Files	889
21.4.5	Strong Encapsulation and Accessibility	890
21.5	Module-Dependency Graphs: A Deeper Look	890
21.5.1	<code>java.sql</code>	891
21.5.2	<code>java.se</code>	891
21.5.3	Error: Module Graph with a Cycle	893
21.6	Migrating Code to Modules	894
21.6.1	Unnamed Module	895
21.6.2	Automatic Modules	895
21.6.3	<code>jdeps</code> Tool for Java Dependency Analysis	896
21.7	Resources in Modules	898
21.7.1	Requiring Multiple Modules	900
21.7.2	Opening a Module for Reflection	900
21.7.3	Compiling the Module	900
21.7.4	Running a Modularized App	900
21.7.5	Module-Dependency Graph	901
21.8	Creating Custom Runtimes with <code>jlink</code>	902
21.8.1	Custom Runtime Containing Only <code>java.base</code>	902
21.8.2	Creating a Custom Runtime for the <code>Welcome</code> App	903
21.8.3	Executing the <code>Welcome</code> App Using a Custom Runtime	904
21.8.4	Using the Module Resolver on a Custom Runtime	904
21.9	Services and <code>ServiceLoader</code>	905
21.9.1	Service-Provider Interface	907
21.9.2	Loading and Consuming Service Providers	908
21.9.3	<code>uses</code> Module Directive and Service Consumers	911
21.9.4	Running the App with No Service Providers	911
21.9.5	Implementing a Service Provider	912
21.9.6	<code>provides...with</code> Module Directive and Declaring a Service Provider	912
21.9.7	Running the App with One Service Provider	913
21.9.8	Implementing a Second Service Provider	914
21.9.9	Running the App with Two Service Providers	915
21.10	Wrap-Up	915

22 Recursion and Big O 919

22.1	Introduction	920
22.2	Recursion Concepts	920
22.3	Recursion Example: Factorials	921

xxii **Contents**

22.4	Recursion Example: Fibonacci Series	924
22.5	Recursion vs. Iteration	927
22.6	Towers of Hanoi	929
22.7	Fractals	931
22.7.1	Koch Curve Fractal	931
22.7.2	(Optional) Case Study: Lo Feather Fractal	932
22.7.3	(Optional) Fractal App GUI	934
22.7.4	(Optional) FractalController Class	936
22.8	Recursive Backtracking	941
22.9	Big <i>O</i> Notation	942
22.9.1	$O(1)$ Algorithms	942
22.9.2	$O(n)$ Algorithms	942
22.9.3	$O(n^2)$ Algorithms	942
22.10	Common Big <i>O</i> Notations	943
22.11	Wrap-Up	943

A **Introduction to JShell for Interactive Java** **945**

A.1	Introduction	946
A.2	Introduction to JShell	948
A.2.1	Starting a JShell Session	948
A.2.2	Executing Statements	949
A.2.3	Declaring Variables Explicitly	950
A.2.4	Listing and Executing Prior Snippets	952
A.2.5	Evaluating Expressions and Declaring Variables Implicitly	953
A.2.6	Using Implicitly Declared Variables	954
A.2.7	Viewing a Variable's Value	954
A.2.8	Resetting a JShell Session	955
A.2.9	Writing Multiline Statements	955
A.2.10	Editing Code Snippets	956
A.2.11	Exiting JShell	959
A.3	Command-Line Input in JShell	959
A.4	Declaring and Using Classes	960
A.4.1	Creating a Class in JShell	960
A.4.2	Explicitly Declaring Reference-Type Variables	961
A.4.3	Creating Objects	961
A.4.4	Manipulating Objects	962
A.4.5	Creating a Meaningful Variable Name for an Expression	962
A.4.6	Saving and Opening Code-Snippet Files	963
A.5	Discovery with JShell Auto-Completion	964
A.5.1	Auto-Completing Identifiers	964
A.5.2	Auto-Completing JShell Commands	965
A.6	Exploring a Class's Members and Viewing Documentation	966
A.6.1	Listing Class Math's static Members	966
A.6.2	Viewing a Method's Parameters	967
A.6.3	Viewing a Method's Documentation	967

A.6.4	Viewing a <code>public</code> Field's Documentation	968
A.6.5	Viewing a Class's Documentation	968
A.6.6	Viewing Method Overloads	969
A.6.7	Exploring Members of a Specific Object	969
A.7	Declaring Methods	971
A.7.1	Forward Referencing an Undeclared Method— Declaring Method <code>displayCubes</code>	971
A.7.2	Declaring a Previously Undeclared Method	972
A.7.3	Testing <code>cube</code> and Replacing Its Declaration	972
A.7.4	Testing Updated Method <code>cube</code> and Method <code>displayCubes</code>	973
A.8	Exceptions	973
A.9	Importing Classes and Adding Packages to the CLASSPATH	974
A.10	Using an External Editor	977
A.11	Summary of JShell Commands	977
A.11.1	Getting Help in JShell	978
A.11.2	<code>/edit</code> Command: Additional Features	979
A.11.3	<code>/reload</code> Command	979
A.11.4	<code>/drop</code> Command	980
A.11.5	Feedback Modes	980
A.11.6	Other JShell Features Configurable with <code>/set</code>	982
A.12	Keyboard Shortcuts for Snippet Editing	983
A.13	How JShell Reinterprets Java for Interactive Use	983
A.14	IDE JShell Support	984
A.15	Wrap-Up	984

B **Formatted Output** **1001**

B.1	Introduction	1002
B.2	Output with <code>printf</code>	1002
B.3	Integer Formatting	1003
B.4	Floating-Point Number Formatting	1004
B.5	String and Character Formatting	1005
B.6	Other Conversion Characters	1006
B.7	Field Widths and Precisions	1007
B.8	Formatting Flags	1009
B.9	Argument Indices for Explicit Positioning in Format Strings	1012
B.10	Escape Sequences	1013
B.11	Formatting Strings in Memory	1014
B.12	Wrap-Up	1015

C **Number Systems** **1017**

C.1	Introduction	1018
C.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	1021
C.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	1022

xxiv [Contents](#)

C.4	Converting from Binary, Octal or Hexadecimal to Decimal	1022
C.5	Converting from Decimal to Binary, Octal or Hexadecimal	1024
C.6	Negative Binary Numbers: Two's Complement Notation	1025

Index **1027**