

# Table of Contents

## Preface to the Java SE 8 Edition xxi

### 1 Introduction 1

- 1.1 Organization of the Specification 2
- 1.2 Example Programs 6
- 1.3 Notation 6
- 1.4 Relationship to Predefined Classes and Interfaces 7
- 1.5 Feedback 7
- 1.6 References 7

### 2 Grammars 9

- 2.1 Context-Free Grammars 9
- 2.2 The Lexical Grammar 9
- 2.3 The Syntactic Grammar 10
- 2.4 Grammar Notation 10

### 3 Lexical Structure 15

- 3.1 Unicode 15
- 3.2 Lexical Translations 16
- 3.3 Unicode Escapes 17
- 3.4 Line Terminators 19
- 3.5 Input Elements and Tokens 19
- 3.6 White Space 20
- 3.7 Comments 21
- 3.8 Identifiers 22
- 3.9 Keywords 24
- 3.10 Literals 24
  - 3.10.1 Integer Literals 25
  - 3.10.2 Floating-Point Literals 31
  - 3.10.3 Boolean Literals 34
  - 3.10.4 Character Literals 34
  - 3.10.5 String Literals 35
  - 3.10.6 Escape Sequences for Character and String Literals 37
  - 3.10.7 The Null Literal 38
- 3.11 Separators 38
- 3.12 Operators 38

## 4 Types, Values, and Variables 41

- 4.1 The Kinds of Types and Values 41
- 4.2 Primitive Types and Values 42
  - 4.2.1 Integral Types and Values 43
  - 4.2.2 Integer Operations 43
  - 4.2.3 Floating-Point Types, Formats, and Values 45
  - 4.2.4 Floating-Point Operations 48
  - 4.2.5 The `boolean` Type and `boolean` Values 51
- 4.3 Reference Types and Values 52
  - 4.3.1 Objects 53
  - 4.3.2 The `Class` Object 55
  - 4.3.3 The `Class String` 56
  - 4.3.4 When Reference Types Are the Same 56
- 4.4 Type Variables 57
- 4.5 Parameterized Types 59
  - 4.5.1 Type Arguments of Parameterized Types 60
  - 4.5.2 Members and Constructors of Parameterized Types 63
- 4.6 Type Erasure 64
- 4.7 Reifiable Types 64
- 4.8 Raw Types 66
- 4.9 Intersection Types 70
- 4.10 Subtyping 71
  - 4.10.1 Subtyping among Primitive Types 71
  - 4.10.2 Subtyping among Class and Interface Types 71
  - 4.10.3 Subtyping among Array Types 73
  - 4.10.4 Least Upper Bound 73
- 4.11 Where Types Are Used 75
- 4.12 Variables 80
  - 4.12.1 Variables of Primitive Type 81
  - 4.12.2 Variables of Reference Type 81
  - 4.12.3 Kinds of Variables 83
  - 4.12.4 `final` Variables 85
  - 4.12.5 Initial Values of Variables 87
  - 4.12.6 Types, Classes, and Interfaces 88

## 5 Conversions and Contexts 91

- 5.1 Kinds of Conversion 94
  - 5.1.1 Identity Conversion 94
  - 5.1.2 Widening Primitive Conversion 94
  - 5.1.3 Narrowing Primitive Conversion 96
  - 5.1.4 Widening and Narrowing Primitive Conversion 99
  - 5.1.5 Widening Reference Conversion 99
  - 5.1.6 Narrowing Reference Conversion 99
  - 5.1.7 Boxing Conversion 100
  - 5.1.8 Unboxing Conversion 102
  - 5.1.9 Unchecked Conversion 103
  - 5.1.10 Capture Conversion 103

5.1.11	String Conversion 105
5.1.12	Forbidden Conversions 106
5.1.13	Value Set Conversion 106
5.2	Assignment Contexts 107
5.3	Invocation Contexts 112
5.4	String Contexts 114
5.5	Casting Contexts 114
5.5.1	Reference Type Casting 117
5.5.2	Checked Casts and Unchecked Casts 121
5.5.3	Checked Casts at Run Time 122
5.6	Numeric Contexts 124
5.6.1	Unary Numeric Promotion 124
5.6.2	Binary Numeric Promotion 125

## 6 Names 129

6.1	Declarations 130
6.2	Names and Identifiers 137
6.3	Scope of a Declaration 139
6.4	Shadowing and Obscuring 142
6.4.1	Shadowing 144
6.4.2	Obscuring 147
6.5	Determining the Meaning of a Name 148
6.5.1	Syntactic Classification of a Name According to Context 149
6.5.2	Reclassification of Contextually Ambiguous Names 152
6.5.3	Meaning of Package Names 154
6.5.3.1	Simple Package Names 155
6.5.3.2	Qualified Package Names 155
6.5.4	Meaning of <i>PackageOrTypeNames</i> 155
6.5.4.1	Simple <i>PackageOrTypeNames</i> 155
6.5.4.2	Qualified <i>PackageOrTypeNames</i> 155
6.5.5	Meaning of Type Names 155
6.5.5.1	Simple Type Names 156
6.5.5.2	Qualified Type Names 156
6.5.6	Meaning of Expression Names 156
6.5.6.1	Simple Expression Names 156
6.5.6.2	Qualified Expression Names 157
6.5.7	Meaning of Method Names 160
6.5.7.1	Simple Method Names 160
6.6	Access Control 161
6.6.1	Determining Accessibility 162
6.6.2	Details on protected Access 166
6.6.2.1	Access to a protected Member 167
6.6.2.2	Qualified Access to a protected Constructor 167
6.7	Fully Qualified Names and Canonical Names 169

## 7 Packages 173

7.1	Package Members 173
-----	---------------------

7.2	Host Support for Packages	175
7.3	Compilation Units	177
7.4	Package Declarations	178
7.4.1	Named Packages	178
7.4.2	Unnamed Packages	179
7.4.3	Observability of a Package	179
7.5	Import Declarations	180
7.5.1	Single-Type-Import Declarations	180
7.5.2	Type-Import-on-Demand Declarations	183
7.5.3	Single-Static-Import Declarations	184
7.5.4	Static-Import-on-Demand Declarations	184
7.6	Top Level Type Declarations	185

## 8 Classes 189

8.1	Class Declarations	191
8.1.1	Class Modifiers	191
8.1.1.1	abstract Classes	192
8.1.1.2	final Classes	194
8.1.1.3	strictfp Classes	194
8.1.2	Generic Classes and Type Parameters	194
8.1.3	Inner Classes and Enclosing Instances	197
8.1.4	Superclasses and Subclasses	200
8.1.5	Superinterfaces	202
8.1.6	Class Body and Member Declarations	205
8.2	Class Members	206
8.3	Field Declarations	211
8.3.1	Field Modifiers	215
8.3.1.1	static Fields	216
8.3.1.2	final Fields	219
8.3.1.3	transient Fields	219
8.3.1.4	volatile Fields	220
8.3.2	Field Initialization	221
8.3.3	Forward References During Field Initialization	222
8.4	Method Declarations	225
8.4.1	Formal Parameters	226
8.4.2	Method Signature	230
8.4.3	Method Modifiers	231
8.4.3.1	abstract Methods	232
8.4.3.2	static Methods	233
8.4.3.3	final Methods	234
8.4.3.4	native Methods	235
8.4.3.5	strictfp Methods	235
8.4.3.6	synchronized Methods	235
8.4.4	Generic Methods	237
8.4.5	Method Result	237
8.4.6	Method Throws	238
8.4.7	Method Body	240

8.4.8	Inheritance, Overriding, and Hiding 240
8.4.8.1	Overriding (by Instance Methods) 241
8.4.8.2	Hiding (by Class Methods) 245
8.4.8.3	Requirements in Overriding and Hiding 246
8.4.8.4	Inheriting Methods with Override-Equivalent Signatures 250
8.4.9	Overloading 250
8.5	Member Type Declarations 254
8.5.1	Static Member Type Declarations 254
8.6	Instance Initializers 255
8.7	Static Initializers 255
8.8	Constructor Declarations 256
8.8.1	Formal Parameters 257
8.8.2	Constructor Signature 258
8.8.3	Constructor Modifiers 258
8.8.4	Generic Constructors 259
8.8.5	Constructor Throws 259
8.8.6	The Type of a Constructor 259
8.8.7	Constructor Body 259
8.8.7.1	Explicit Constructor Invocations 260
8.8.8	Constructor Overloading 264
8.8.9	Default Constructor 265
8.8.10	Preventing Instantiation of a Class 266
8.9	Enum Types 266
8.9.1	Enum Constants 267
8.9.2	Enum Body Declarations 268
8.9.3	Enum Members 271

## 9 Interfaces 277

9.1	Interface Declarations 278
9.1.1	Interface Modifiers 278
9.1.1.1	abstract Interfaces 279
9.1.1.2	strictfp Interfaces 279
9.1.2	Generic Interfaces and Type Parameters 279
9.1.3	Superinterfaces and Subinterfaces 280
9.1.4	Interface Body and Member Declarations 282
9.2	Interface Members 282
9.3	Field (Constant) Declarations 283
9.3.1	Initialization of Fields in Interfaces 285
9.4	Method Declarations 286
9.4.1	Inheritance and Overriding 287
9.4.1.1	Overriding (by Instance Methods) 288
9.4.1.2	Requirements in Overriding 289
9.4.1.3	Inheriting Methods with Override-Equivalent Signatures 289
9.4.2	Overloading 290
9.4.3	Interface Method Body 291

9.5	Member Type Declarations	291
9.6	Annotation Types	292
9.6.1	Annotation Type Elements	293
9.6.2	Defaults for Annotation Type Elements	297
9.6.3	Repeatable Annotation Types	298
9.6.4	Predefined Annotation Types	302
9.6.4.1	@Target	302
9.6.4.2	@Retention	303
9.6.4.3	@Inherited	304
9.6.4.4	@Override	304
9.6.4.5	@SuppressWarnings	305
9.6.4.6	@Deprecated	306
9.6.4.7	@SafeVarargs	307
9.6.4.8	@Repeatable	308
9.6.4.9	@FunctionalInterface	308
9.7	Annotations	308
9.7.1	Normal Annotations	309
9.7.2	Marker Annotations	311
9.7.3	Single-Element Annotations	312
9.7.4	Where Annotations May Appear	313
9.7.5	Multiple Annotations of the Same Type	318
9.8	Functional Interfaces	319
9.9	Function Types	323

## 10 Arrays 329

10.1	Array Types	330
10.2	Array Variables	330
10.3	Array Creation	332
10.4	Array Access	332
10.5	Array Store Exception	333
10.6	Array Initializers	335
10.7	Array Members	336
10.8	Class Objects for Arrays	338
10.9	An Array of Characters Is Not a String	339

## 11 Exceptions 341

11.1	The Kinds and Causes of Exceptions	342
11.1.1	The Kinds of Exceptions	342
11.1.2	The Causes of Exceptions	343
11.1.3	Asynchronous Exceptions	343
11.2	Compile-Time Checking of Exceptions	344
11.2.1	Exception Analysis of Expressions	346
11.2.2	Exception Analysis of Statements	346
11.2.3	Exception Checking	347
11.3	Run-Time Handling of an Exception	349

## **12 Execution 353**

- 12.1 Java Virtual Machine Startup 353
  - 12.1.1 Load the Class Test 354
  - 12.1.2 Link Test: Verify, Prepare, (Optionally) Resolve 354
  - 12.1.3 Initialize Test: Execute Initializers 355
  - 12.1.4 Invoke Test.main 356
- 12.2 Loading of Classes and Interfaces 356
  - 12.2.1 The Loading Process 357
- 12.3 Linking of Classes and Interfaces 358
  - 12.3.1 Verification of the Binary Representation 358
  - 12.3.2 Preparation of a Class or Interface Type 359
  - 12.3.3 Resolution of Symbolic References 359
- 12.4 Initialization of Classes and Interfaces 360
  - 12.4.1 When Initialization Occurs 361
  - 12.4.2 Detailed Initialization Procedure 363
- 12.5 Creation of New Class Instances 365
- 12.6 Finalization of Class Instances 369
  - 12.6.1 Implementing Finalization 370
  - 12.6.2 Interaction with the Memory Model 372
- 12.7 Unloading of Classes and Interfaces 373
- 12.8 Program Exit 374

## **13 Binary Compatibility 375**

- 13.1 The Form of a Binary 376
- 13.2 What Binary Compatibility Is and Is Not 382
- 13.3 Evolution of Packages 383
- 13.4 Evolution of Classes 383
  - 13.4.1 abstract Classes 383
  - 13.4.2 final Classes 383
  - 13.4.3 public Classes 384
  - 13.4.4 Superclasses and Superinterfaces 384
  - 13.4.5 Class Type Parameters 385
  - 13.4.6 Class Body and Member Declarations 386
  - 13.4.7 Access to Members and Constructors 387
  - 13.4.8 Field Declarations 388
  - 13.4.9 final Fields and static Constant Variables 391
  - 13.4.10 static Fields 393
  - 13.4.11 transient Fields 393
  - 13.4.12 Method and Constructor Declarations 394
  - 13.4.13 Method and Constructor Type Parameters 394
  - 13.4.14 Method and Constructor Formal Parameters 395
  - 13.4.15 Method Result Type 396
  - 13.4.16 abstract Methods 396
  - 13.4.17 final Methods 397
  - 13.4.18 native Methods 397
  - 13.4.19 static Methods 398
  - 13.4.20 synchronized Methods 398

13.4.21	Method and Constructor Throws	398
13.4.22	Method and Constructor Body	398
13.4.23	Method and Constructor Overloading	399
13.4.24	Method Overriding	400
13.4.25	Static Initializers	400
13.4.26	Evolution of Enums	400
13.5	Evolution of Interfaces	400
13.5.1	public Interfaces	400
13.5.2	Superinterfaces	401
13.5.3	Interface Members	401
13.5.4	Interface Type Parameters	401
13.5.5	Field Declarations	402
13.5.6	Interface Method Declarations	402
13.5.7	Evolution of Annotation Types	403

## 14 Blocks and Statements 405

14.1	Normal and Abrupt Completion of Statements	405
14.2	Blocks	407
14.3	Local Class Declarations	407
14.4	Local Variable Declaration Statements	408
14.4.1	Local Variable Declarators and Types	409
14.4.2	Execution of Local Variable Declarations	410
14.5	Statements	410
14.6	The Empty Statement	412
14.7	Labeled Statements	413
14.8	Expression Statements	414
14.9	The if Statement	415
14.9.1	The if-then Statement	415
14.9.2	The if-then-else Statement	416
14.10	The assert Statement	416
14.11	The switch Statement	419
14.12	The while Statement	423
14.12.1	Abrupt Completion of while Statement	424
14.13	The do Statement	424
14.13.1	Abrupt Completion of do Statement	425
14.14	The for Statement	426
14.14.1	The basic for Statement	426
14.14.1.1	Initialization of for Statement	427
14.14.1.2	Iteration of for Statement	427
14.14.1.3	Abrupt Completion of for Statement	428
14.14.2	The enhanced for statement	429
14.15	The break Statement	432
14.16	The continue Statement	434
14.17	The return Statement	436
14.18	The throw Statement	437
14.19	The synchronized Statement	439
14.20	The try statement	440

14.20.1	Execution of try-catch	444
14.20.2	Execution of try-finally and try-catch-finally	445
14.20.3	try-with-resources	447
14.20.3.1	Basic try-with-resources	448
14.20.3.2	Extended try-with-resources	451
14.21	Unreachable Statements	452

## 15 Expressions 459

15.1	Evaluation, Denotation, and Result	459
15.2	Forms of Expressions	460
15.3	Type of an Expression	461
15.4	FP-strict Expressions	462
15.5	Expressions and Run-Time Checks	462
15.6	Normal and Abrupt Completion of Evaluation	464
15.7	Evaluation Order	466
15.7.1	Evaluate Left-Hand Operand First	466
15.7.2	Evaluate Operands before Operation	468
15.7.3	Evaluation Respects Parentheses and Precedence	469
15.7.4	Argument Lists are Evaluated Left-to-Right	470
15.7.5	Evaluation Order for Other Expressions	471
15.8	Primary Expressions	471
15.8.1	Lexical Literals	472
15.8.2	Class Literals	473
15.8.3	this	474
15.8.4	Qualified this	475
15.8.5	Parenthesized Expressions	475
15.9	Class Instance Creation Expressions	476
15.9.1	Determining the Class being Instantiated	478
15.9.2	Determining Enclosing Instances	480
15.9.3	Choosing the Constructor and its Arguments	481
15.9.4	Run-Time Evaluation of Class Instance Creation Expressions	484
15.9.5	Anonymous Class Declarations	485
15.9.5.1	Anonymous Constructors	485
15.10	Array Creation and Access Expressions	487
15.10.1	Array Creation Expressions	487
15.10.2	Run-Time Evaluation of Array Creation Expressions	488
15.10.3	Array Access Expressions	491
15.10.4	Run-Time Evaluation of Array Access Expressions	492
15.11	Field Access Expressions	494
15.11.1	Field Access Using a Primary	494
15.11.2	Accessing Superclass Members using super	497
15.12	Method Invocation Expressions	499
15.12.1	Compile-Time Step 1: Determine Class or Interface to Search	500
15.12.2	Compile-Time Step 2: Determine Method Signature	502
15.12.2.1	Identify Potentially Applicable Methods	509

15.12.2.2	Phase 1: Identify Matching Arity Methods Applicable by Strict Invocation	511
15.12.2.3	Phase 2: Identify Matching Arity Methods Applicable by Loose Invocation	512
15.12.2.4	Phase 3: Identify Methods Applicable by Variable Arity Invocation	513
15.12.2.5	Choosing the Most Specific Method	514
15.12.2.6	Method Invocation Type	516
15.12.3	Compile-Time Step 3: Is the Chosen Method Appropriate?	517
15.12.4	Run-Time Evaluation of Method Invocation	520
15.12.4.1	Compute Target Reference (If Necessary)	520
15.12.4.2	Evaluate Arguments	522
15.12.4.3	Check Accessibility of Type and Method	523
15.12.4.4	Locate Method to Invoke	524
15.12.4.5	Create Frame, Synchronize, Transfer Control	528
15.13	Method Reference Expressions	529
15.13.1	Compile-Time Declaration of a Method Reference	532
15.13.2	Type of a Method Reference	537
15.13.3	Run-Time Evaluation of Method References	539
15.14	Postfix Expressions	542
15.14.1	Expression Names	543
15.14.2	Postfix Increment Operator <code>++</code>	543
15.14.3	Postfix Decrement Operator <code>--</code>	544
15.15	Unary Operators	544
15.15.1	Prefix Increment Operator <code>++</code>	546
15.15.2	Prefix Decrement Operator <code>--</code>	546
15.15.3	Unary Plus Operator <code>+</code>	547
15.15.4	Unary Minus Operator <code>-</code>	547
15.15.5	Bitwise Complement Operator <code>~</code>	548
15.15.6	Logical Complement Operator <code>!</code>	548
15.16	Cast Expressions	549
15.17	Multiplicative Operators	550
15.17.1	Multiplication Operator <code>*</code>	551
15.17.2	Division Operator <code>/</code>	552
15.17.3	Remainder Operator <code>%</code>	554
15.18	Additive Operators	556
15.18.1	String Concatenation Operator <code>+</code>	557
15.18.2	Additive Operators ( <code>+</code> and <code>-</code> ) for Numeric Types	559
15.19	Shift Operators	561
15.20	Relational Operators	562
15.20.1	Numerical Comparison Operators <code>&lt;</code> , <code>&lt;=</code> , <code>&gt;</code> , and <code>&gt;=</code>	563
15.20.2	Type Comparison Operator <code>instanceof</code>	564
15.21	Equality Operators	565
15.21.1	Numerical Equality Operators <code>==</code> and <code>!=</code>	566
15.21.2	Boolean Equality Operators <code>==</code> and <code>!=</code>	567
15.21.3	Reference Equality Operators <code>==</code> and <code>!=</code>	567
15.22	Bitwise and Logical Operators	568
15.22.1	Integer Bitwise Operators <code>&amp;</code> , <code>^</code> , and <code> </code>	568

15.22.2	Boolean Logical Operators &, ^, and	569
15.23	Conditional-And Operator &&	570
15.24	Conditional-Or Operator	570
15.25	Conditional Operator ? :	571
	15.25.1 Boolean Conditional Expressions	579
	15.25.2 Numeric Conditional Expressions	579
	15.25.3 Reference Conditional Expressions	580
15.26	Assignment Operators	581
	15.26.1 Simple Assignment Operator =	582
	15.26.2 Compound Assignment Operators	588
15.27	Lambda Expressions	594
	15.27.1 Lambda Parameters	596
	15.27.2 Lambda Body	599
	15.27.3 Type of a Lambda Expression	602
	15.27.4 Run-Time Evaluation of Lambda Expressions	604
15.28	Constant Expressions	605

## 16 Definite Assignment 607

16.1	Definite Assignment and Expressions	613
	16.1.1 Boolean Constant Expressions	613
	16.1.2 Conditional-And Operator &&	613
	16.1.3 Conditional-Or Operator	614
	16.1.4 Logical Complement Operator !	614
	16.1.5 Conditional Operator ? :	614
	16.1.6 Conditional Operator ? :	615
	16.1.7 Other Expressions of Type boolean	615
	16.1.8 Assignment Expressions	615
	16.1.9 Operators ++ and --	616
	16.1.10 Other Expressions	616
16.2	Definite Assignment and Statements	617
	16.2.1 Empty Statements	617
	16.2.2 Blocks	617
	16.2.3 Local Class Declaration Statements	619
	16.2.4 Local Variable Declaration Statements	619
	16.2.5 Labeled Statements	619
	16.2.6 Expression Statements	620
	16.2.7 if Statements	620
	16.2.8 assert Statements	620
	16.2.9 switch Statements	621
	16.2.10 while Statements	621
	16.2.11 do Statements	622
	16.2.12 for Statements	622
	16.2.12.1 Initialization Part of for Statement	623
	16.2.12.2 Incrementation Part of for Statement	623
	16.2.13 break, continue, return, and throw Statements	624
	16.2.14 synchronized Statements	624
	16.2.15 try Statements	624

16.3	Definite Assignment and Parameters	626
16.4	Definite Assignment and Array Initializers	626
16.5	Definite Assignment and Enum Constants	626
16.6	Definite Assignment and Anonymous Classes	627
16.7	Definite Assignment and Member Types	627
16.8	Definite Assignment and Static Initializers	628
16.9	Definite Assignment, Constructors, and Instance Initializers	628

## 17 Threads and Locks 631

17.1	Synchronization	632
17.2	Wait Sets and Notification	632
17.2.1	Wait	633
17.2.2	Notification	634
17.2.3	Interruptions	635
17.2.4	Interactions of Waits, Notification, and Interruption	635
17.3	Sleep and Yield	636
17.4	Memory Model	637
17.4.1	Shared Variables	640
17.4.2	Actions	640
17.4.3	Programs and Program Order	641
17.4.4	Synchronization Order	642
17.4.5	Happens-before Order	643
17.4.6	Executions	646
17.4.7	Well-Formed Executions	647
17.4.8	Executions and Causality Requirements	647
17.4.9	Observable Behavior and Nonterminating Executions	650
17.5	final Field Semantics	652
17.5.1	Semantics of final Fields	654
17.5.2	Reading final Fields During Construction	654
17.5.3	Subsequent Modification of final Fields	655
17.5.4	Write-Protected Fields	656
17.6	Word Tearing	657
17.7	Non-Atomic Treatment of double and long	658

## 18 Type Inference 659

18.1	Concepts and Notation	660
18.1.1	Inference Variables	660
18.1.2	Constraint Formulas	661
18.1.3	Bounds	661
18.2	Reduction	663
18.2.1	Expression Compatibility Constraints	663
18.2.2	Type Compatibility Constraints	667
18.2.3	Subtyping Constraints	668
18.2.4	Type Equality Constraints	670
18.2.5	Checked Exception Constraints	671
18.3	Incorporation	673
18.3.1	Complementary Pairs of Bounds	674

	18.3.2	Bounds Involving Capture Conversion	674
18.4	Resolution	675	
18.5	Uses of Inference	677	
	18.5.1	Invocation Applicability Inference	678
	18.5.2	Invocation Type Inference	679
	18.5.3	Functional Interface Parameterization Inference	685
	18.5.4	More Specific Method Inference	686

## **19 Syntax** 689

### **Index** 715

## **A Limited License Grant** 755