

# Table of Contents

---

## **Preface to the Java SE 8 Edition xix**

## **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 39
- 3.12 Operators 39

## **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` 56
  - 4.3.3 The Class `String` 56
  - 4.3.4 When Reference Types Are the Same 57
- 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 65
- 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 72
  - 4.10.3 Subtyping among Array Types 73
  - 4.10.4 Least Upper Bound 73
- 4.11 Where Types Are Used 76
- 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 93**

- 5.1 Kinds of Conversion 96
  - 5.1.1 Identity Conversion 96
  - 5.1.2 Widening Primitive Conversion 96
  - 5.1.3 Narrowing Primitive Conversion 98
  - 5.1.4 Widening and Narrowing Primitive Conversion 101
  - 5.1.5 Widening Reference Conversion 101
  - 5.1.6 Narrowing Reference Conversion 101
  - 5.1.7 Boxing Conversion 102
  - 5.1.8 Unboxing Conversion 104
  - 5.1.9 Unchecked Conversion 105
  - 5.1.10 Capture Conversion 105

- 5.1.11 String Conversion 107
- 5.1.12 Forbidden Conversions 108
- 5.1.13 Value Set Conversion 108
- 5.2 Assignment Contexts 109
- 5.3 Invocation Contexts 114
- 5.4 String Contexts 116
- 5.5 Casting Contexts 116
  - 5.5.1 Reference Type Casting 120
  - 5.5.2 Checked Casts and Unchecked Casts 124
  - 5.5.3 Checked Casts at Run Time 125
- 5.6 Numeric Contexts 127
  - 5.6.1 Unary Numeric Promotion 127
  - 5.6.2 Binary Numeric Promotion 128

## 6 Names 131

- 6.1 Declarations 132
- 6.2 Names and Identifiers 139
- 6.3 Scope of a Declaration 141
- 6.4 Shadowing and Obscuring 144
  - 6.4.1 Shadowing 146
  - 6.4.2 Obscuring 149
- 6.5 Determining the Meaning of a Name 150
  - 6.5.1 Syntactic Classification of a Name According to Context 151
  - 6.5.2 Reclassification of Contextually Ambiguous Names 154
  - 6.5.3 Meaning of Package Names 156
    - 6.5.3.1 Simple Package Names 157
    - 6.5.3.2 Qualified Package Names 157
  - 6.5.4 Meaning of *PackageOrTypeNames* 157
    - 6.5.4.1 Simple *PackageOrTypeNames* 157
    - 6.5.4.2 Qualified *PackageOrTypeNames* 157
  - 6.5.5 Meaning of Type Names 157
    - 6.5.5.1 Simple Type Names 158
    - 6.5.5.2 Qualified Type Names 158
  - 6.5.6 Meaning of Expression Names 158
    - 6.5.6.1 Simple Expression Names 158
    - 6.5.6.2 Qualified Expression Names 159
  - 6.5.7 Meaning of Method Names 162
    - 6.5.7.1 Simple Method Names 162
- 6.6 Access Control 163
  - 6.6.1 Determining Accessibility 164
  - 6.6.2 Details on `protected` Access 168
    - 6.6.2.1 Access to a `protected` Member 169
    - 6.6.2.2 Qualified Access to a `protected` Constructor 169
- 6.7 Fully Qualified Names and Canonical Names 171

## 7 Packages 175

- 7.1 Package Members 175

- 7.2 Host Support for Packages 177
- 7.3 Compilation Units 179
- 7.4 Package Declarations 180
  - 7.4.1 Named Packages 180
  - 7.4.2 Unnamed Packages 181
  - 7.4.3 Observability of a Package 181
- 7.5 Import Declarations 182
  - 7.5.1 Single-Type-Import Declarations 182
  - 7.5.2 Type-Import-on-Demand Declarations 185
  - 7.5.3 Single-Static-Import Declarations 186
  - 7.5.4 Static-Import-on-Demand Declarations 186
- 7.6 Top Level Type Declarations 187

## **8 Classes 191**

- 8.1 Class Declarations 193
  - 8.1.1 Class Modifiers 193
    - 8.1.1.1 `abstract` Classes 194
    - 8.1.1.2 `final` Classes 196
    - 8.1.1.3 `strictfp` Classes 196
  - 8.1.2 Generic Classes and Type Parameters 196
  - 8.1.3 Inner Classes and Enclosing Instances 199
  - 8.1.4 Superclasses and Subclasses 202
  - 8.1.5 Superinterfaces 204
  - 8.1.6 Class Body and Member Declarations 207
- 8.2 Class Members 208
- 8.3 Field Declarations 213
  - 8.3.1 Field Modifiers 217
    - 8.3.1.1 `static` Fields 218
    - 8.3.1.2 `final` Fields 221
    - 8.3.1.3 `transient` Fields 221
    - 8.3.1.4 `volatile` Fields 222
  - 8.3.2 Field Initialization 223
  - 8.3.3 Forward References During Field Initialization 224
- 8.4 Method Declarations 227
  - 8.4.1 Formal Parameters 228
  - 8.4.2 Method Signature 232
  - 8.4.3 Method Modifiers 233
    - 8.4.3.1 `abstract` Methods 234
    - 8.4.3.2 `static` Methods 236
    - 8.4.3.3 `final` Methods 236
    - 8.4.3.4 `native` Methods 237
    - 8.4.3.5 `strictfp` Methods 237
    - 8.4.3.6 `synchronized` Methods 238
  - 8.4.4 Generic Methods 239
  - 8.4.5 Method Result 240
  - 8.4.6 Method Throws 240
  - 8.4.7 Method Body 242

- 8.4.8 Inheritance, Overriding, and Hiding 243
  - 8.4.8.1 Overriding (by Instance Methods) 243
  - 8.4.8.2 Hiding (by Class Methods) 247
  - 8.4.8.3 Requirements in Overriding and Hiding 248
  - 8.4.8.4 Inheriting Methods with Override-Equivalent Signatures 252
- 8.4.9 Overloading 253
- 8.5 Member Type Declarations 256
  - 8.5.1 Static Member Type Declarations 257
- 8.6 Instance Initializers 257
- 8.7 Static Initializers 258
- 8.8 Constructor Declarations 258
  - 8.8.1 Formal Parameters 259
  - 8.8.2 Constructor Signature 260
  - 8.8.3 Constructor Modifiers 260
  - 8.8.4 Generic Constructors 261
  - 8.8.5 Constructor Throws 262
  - 8.8.6 The Type of a Constructor 262
  - 8.8.7 Constructor Body 262
    - 8.8.7.1 Explicit Constructor Invocations 263
  - 8.8.8 Constructor Overloading 267
  - 8.8.9 Default Constructor 267
  - 8.8.10 Preventing Instantiation of a Class 268
- 8.9 Enum Types 269
  - 8.9.1 Enum Constants 270
  - 8.9.2 Enum Body Declarations 271
  - 8.9.3 Enum Members 273

## **9 Interfaces 279**

- 9.1 Interface Declarations 280
  - 9.1.1 Interface Modifiers 280
    - 9.1.1.1 `abstract` Interfaces 281
    - 9.1.1.2 `strictfp` Interfaces 281
  - 9.1.2 Generic Interfaces and Type Parameters 281
  - 9.1.3 Superinterfaces and Subinterfaces 282
  - 9.1.4 Interface Body and Member Declarations 284
- 9.2 Interface Members 284
- 9.3 Field (Constant) Declarations 285
  - 9.3.1 Initialization of Fields in Interfaces 287
- 9.4 Method Declarations 288
  - 9.4.1 Inheritance and Overriding 289
    - 9.4.1.1 Overriding (by Instance Methods) 290
    - 9.4.1.2 Requirements in Overriding 291
    - 9.4.1.3 Inheriting Methods with Override-Equivalent Signatures 291
  - 9.4.2 Overloading 292
  - 9.4.3 Interface Method Body 293

- 9.5 Member Type Declarations 293
- 9.6 Annotation Types 294
  - 9.6.1 Annotation Type Elements 295
  - 9.6.2 Defaults for Annotation Type Elements 299
  - 9.6.3 Repeatable Annotation Types 300
  - 9.6.4 Predefined Annotation Types 304
    - 9.6.4.1 `@Target` 304
    - 9.6.4.2 `@Retention` 305
    - 9.6.4.3 `@Inherited` 306
    - 9.6.4.4 `@Override` 306
    - 9.6.4.5 `@SuppressWarnings` 307
    - 9.6.4.6 `@Deprecated` 308
    - 9.6.4.7 `@SafeVarargs` 309
    - 9.6.4.8 `@Repeatable` 310
    - 9.6.4.9 `@FunctionalInterface` 310
- 9.7 Annotations 310
  - 9.7.1 Normal Annotations 311
  - 9.7.2 Marker Annotations 313
  - 9.7.3 Single-Element Annotations 314
  - 9.7.4 Where Annotations May Appear 315
  - 9.7.5 Multiple Annotations of the Same Type 320
- 9.8 Functional Interfaces 321
- 9.9 Function Types 325

## **10 Arrays 331**

- 10.1 Array Types 332
- 10.2 Array Variables 332
- 10.3 Array Creation 335
- 10.4 Array Access 335
- 10.5 Array Store Exception 336
- 10.6 Array Initializers 337
- 10.7 Array Members 339
- 10.8 `Class` Objects for Arrays 340
- 10.9 An Array of Characters Is Not a `String` 342

## **11 Exceptions 343**

- 11.1 The Kinds and Causes of Exceptions 344
  - 11.1.1 The Kinds of Exceptions 344
  - 11.1.2 The Causes of Exceptions 345
  - 11.1.3 Asynchronous Exceptions 346
- 11.2 Compile-Time Checking of Exceptions 347
  - 11.2.1 Exception Analysis of Expressions 348
  - 11.2.2 Exception Analysis of Statements 349
  - 11.2.3 Exception Checking 350
- 11.3 Run-Time Handling of an Exception 352

## 12 Execution 357

- 12.1 Java Virtual Machine Startup 357
  - 12.1.1 Load the Class `Test` 358
  - 12.1.2 Link `Test`: Verify, Prepare, (Optionally) Resolve 358
  - 12.1.3 Initialize `Test`: Execute Initializers 359
  - 12.1.4 Invoke `Test.main` 360
- 12.2 Loading of Classes and Interfaces 360
  - 12.2.1 The Loading Process 361
- 12.3 Linking of Classes and Interfaces 362
  - 12.3.1 Verification of the Binary Representation 362
  - 12.3.2 Preparation of a Class or Interface Type 363
  - 12.3.3 Resolution of Symbolic References 363
- 12.4 Initialization of Classes and Interfaces 364
  - 12.4.1 When Initialization Occurs 365
  - 12.4.2 Detailed Initialization Procedure 367
- 12.5 Creation of New Class Instances 370
- 12.6 Finalization of Class Instances 373
  - 12.6.1 Implementing Finalization 375
  - 12.6.2 Interaction with the Memory Model 376
- 12.7 Unloading of Classes and Interfaces 378
- 12.8 Program Exit 379

## 13 Binary Compatibility 381

- 13.1 The Form of a Binary 382
- 13.2 What Binary Compatibility Is and Is Not 388
- 13.3 Evolution of Packages 389
- 13.4 Evolution of Classes 389
  - 13.4.1 `abstract` Classes 389
  - 13.4.2 `final` Classes 389
  - 13.4.3 `public` Classes 390
  - 13.4.4 Superclasses and Superinterfaces 390
  - 13.4.5 Class Type Parameters 391
  - 13.4.6 Class Body and Member Declarations 392
  - 13.4.7 Access to Members and Constructors 393
  - 13.4.8 Field Declarations 394
  - 13.4.9 `final` Fields and `static` Constant Variables 397
  - 13.4.10 `static` Fields 399
  - 13.4.11 `transient` Fields 399
  - 13.4.12 Method and Constructor Declarations 400
  - 13.4.13 Method and Constructor Type Parameters 400
  - 13.4.14 Method and Constructor Formal Parameters 401
  - 13.4.15 Method Result Type 402
  - 13.4.16 `abstract` Methods 402
  - 13.4.17 `final` Methods 403
  - 13.4.18 `native` Methods 403
  - 13.4.19 `static` Methods 404
  - 13.4.20 `synchronized` Methods 404

- 13.4.21 Method and Constructor Throws 404
- 13.4.22 Method and Constructor Body 404
- 13.4.23 Method and Constructor Overloading 405
- 13.4.24 Method Overriding 406
- 13.4.25 Static Initializers 406
- 13.4.26 Evolution of Enums 406
- 13.5 Evolution of Interfaces 406
  - 13.5.1 `public` Interfaces 406
  - 13.5.2 Superinterfaces 407
  - 13.5.3 Interface Members 407
  - 13.5.4 Interface Type Parameters 407
  - 13.5.5 Field Declarations 408
  - 13.5.6 Interface Method Declarations 408
  - 13.5.7 Evolution of Annotation Types 409

## **14 Blocks and Statements 411**

- 14.1 Normal and Abrupt Completion of Statements 411
- 14.2 Blocks 413
- 14.3 Local Class Declarations 413
- 14.4 Local Variable Declaration Statements 414
  - 14.4.1 Local Variable Declarators and Types 415
  - 14.4.2 Execution of Local Variable Declarations 416
- 14.5 Statements 416
- 14.6 The Empty Statement 418
- 14.7 Labeled Statements 419
- 14.8 Expression Statements 420
- 14.9 The `if` Statement 421
  - 14.9.1 The `if-then` Statement 422
  - 14.9.2 The `if-then-else` Statement 422
- 14.10 The `assert` Statement 422
- 14.11 The `switch` Statement 425
- 14.12 The `while` Statement 429
  - 14.12.1 Abrupt Completion of `while` Statement 430
- 14.13 The `do` Statement 431
  - 14.13.1 Abrupt Completion of `do` Statement 431
- 14.14 The `for` Statement 433
  - 14.14.1 The basic `for` Statement 433
    - 14.14.1.1 Initialization of `for` Statement 434
    - 14.14.1.2 Iteration of `for` Statement 434
    - 14.14.1.3 Abrupt Completion of `for` Statement 435
  - 14.14.2 The enhanced `for` statement 436
- 14.15 The `break` Statement 438
- 14.16 The `continue` Statement 440
- 14.17 The `return` Statement 442
- 14.18 The `throw` Statement 444
- 14.19 The `synchronized` Statement 446
- 14.20 The `try` statement 447