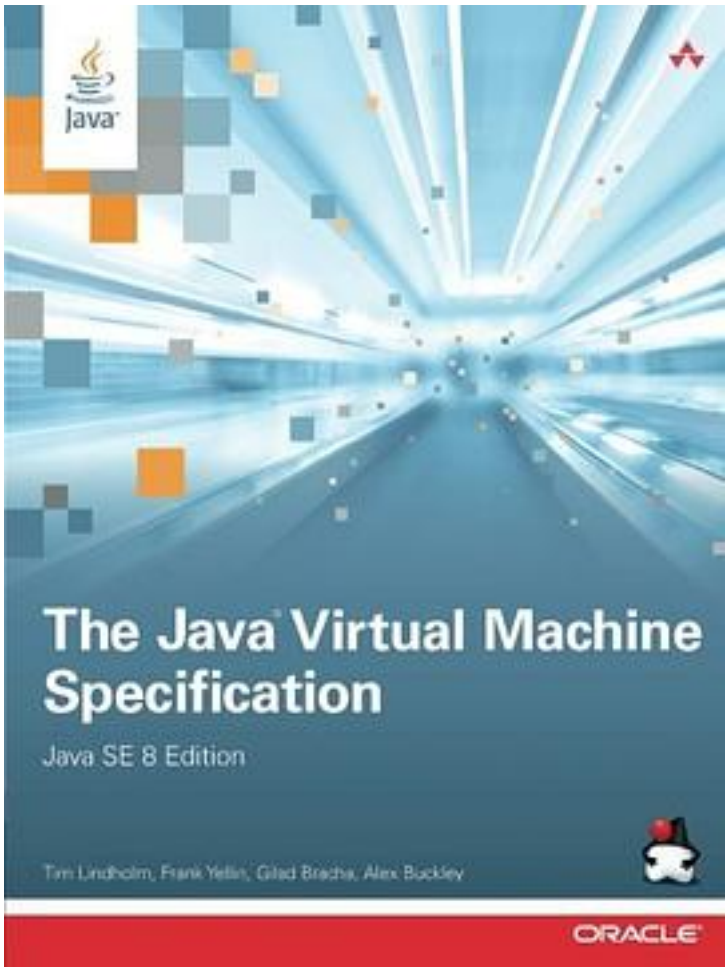
The Java Virtual Machine Specification, Java SE 8 Edition



The Java Virtual Machine Specification, Java SE 8 Edition_下载链接1_

著者:Tim Lindholm

出版者:Addison-Wesley Professional

出版时间:2014-5-17

装帧:Paperback

isbn:9780133905908

Written by the inventors of the technology, The Java® Virtual Machine Specification,

Java SE 8 Edition is the definitive technical reference for the Java Virtual Machine.

The book provides complete, accurate, and detailed coverage of the Java Virtual Machine. It fully describes the new features added in Java SE 8, including the invocation of default methods and the class file extensions for type annotations and method parameters. The book also clarifies the interpretation of class file attributes and the rules of bytecode verification.

作者介绍:

Tim Lindholm is a former Distinguished Engineer at Sun Microsystems. He was a contributor to the Java programming language and the senior architect of the Java Virtual Machine, later working on Java for mobile devices. Prior to Sun, he worked on virtual machines and runtime systems for Prolog at Argonne National Laboratory and Quintus. He holds a B.A. in Mathematics from Carleton College.

Frank Yellin is a former Staff Engineer at Sun Microsystems. He was an original member of the Java project and spent a decade working on runtime systems for interpreted and compiled languages. Prior to Sun, he worked on the compilation of Common Lisp at Lucid. He holds an A.B. in Applied Mathematics from Harvard and an M.S. in Computer Science from Stanford.

Gilad Bracha is the creator of the Newspeak programming language and a former Distinguished Engineer at Sun Microsystems. Prior to Sun, he worked on Strongtalk, the Animorphic Smalltalk System. He holds a Ph.D. in Computer Science from the University of Utah.

Alex Buckley is the Specification Lead for the Java programming language and the Java Virtual Machine at Oracle. He holds a Ph.D. in Computing from Imperial College London.

目录: 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 394 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 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 103The Java® Language Specification
vii
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 118
5.5.2 Checked Casts and Unchecked Casts 122
5.5.3 Checked Casts at Run Time 123
5.6 Numeric Contexts 125
5.6.1 Unary Numeric Promotion 125
5.6.2 Binary Numeric Promotion 126
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 PackageOrTypeNames 155
6.5.4.1 Simple PackageOrTypeNames 155
6.5.4.2 Qualified PackageOrTypeNames 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 173The Java® Language Specification
viii
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 240The Java® Language Specification
ix
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 344
11.2 Compile-Time Checking of Exceptions 345
11.2.1 Exception Analysis of Expressions 346
11.2.2 Exception Analysis of Statements 347
11.2.3 Exception Checking 348
11.3 Run-Time Handling of an Exception 350The Java® Language Specification
xi
12 Execution 355
12.1 Java Virtual Machine Startup 355
12.1.1 Load the Class Test 356
12.1.2 Link Test: Verify, Prepare, (Optionally) Resolve 356
12.1.3 Initialize Test: Execute Initializers 357
12.1.4 Invoke Test.main 358
12.2 Loading of Classes and Interfaces 358
12.2.1 The Loading Process 359
12.3 Linking of Classes and Interfaces 360
12.3.1 Verification of the Binary Representation 360
12.3.2 Preparation of a Class or Interface Type 361
12.3.3 Resolution of Symbolic References 361
12.4 Initialization of Classes and Interfaces 362
12.4.1 When Initialization Occurs 363
12.4.2 Detailed Initialization Procedure 365
12.5 Creation of New Class Instances 367
12.6 Finalization of Class Instances 371
12.6.1 Implementing Finalization 372
12.6.2 Interaction with the Memory Model 374
12.7 Unloading of Classes and Interfaces 375
12.8 Program Exit 376
13 Binary Compatibility 377
13.1 The Form of a Binary 378
13.2 What Binary Compatibility Is and Is Not 384
13.3 Evolution of Packages 385
13.4 Evolution of Classes 385
13.4.1 abstract Classes 385

13.4.2 final Classes 385
13.4.3 public Classes 386
13.4.4 Superclasses and Superinterfaces 386
13.4.5 Class Type Parameters 387
13.4.6 Class Body and Member Declarations 388
13.4.7 Access to Members and Constructors 389
13.4.8 Field Declarations 390
13.4.9 final Fields and static Constant Variables 393
13.4.10 static Fields 395
13.4.11 transient Fields 395
13.4.12 Method and Constructor Declarations 396
13.4.13 Method and Constructor Type Parameters 396
13.4.14 Method and Constructor Formal Parameters 397
13.4.15 Method Result Type 398
13.4.16 abstract Methods 398
13.4.17 final Methods 399
13.4.18 native Methods 399
13.4.19 static Methods 400
13.4.20 synchronized Methods 400he Java® Language Specification
xii
13.4.21 Method and Constructor Throws 400
13.4.22 Method and Constructor Body 400
13.4.23 Method and Constructor Overloading 401
13.4.24 Method Overriding 402
13.4.25 Static Initializers 402
13.4.26 Evolution of Enums 402
13.5 Evolution of Interfaces 402
13.5.1 public Interfaces 402
13.5.2 Superinterfaces 403
13.5.3 Interface Members 403
13.5.4 Interface Type Parameters 403
13.5.5 Field Declarations 404
13.5.6 Interface Method Declarations 404
13.5.7 Evolution of Annotation Types 405
14 Blocks and Statements 407
14.1 Normal and Abrupt Completion of Statements 407
14.2 Blocks 409
14.3 Local Class Declarations 409
14.4 Local Variable Declaration Statements 410
14.4.1 Local Variable Declarators and Types 411
14.4.2 Execution of Local Variable Declarations 412
14.5 Statements 412
14.6 The Empty Statement 414
14.7 Labeled Statements 415
14.8 Expression Statements 416
14.9 The if Statement 417
14.9.1 The if-then Statement 417
14.9.2 The if-then-else Statement 418
14.10 The assert Statement 418
14.11 The switch Statement 421
14.12 The while Statement 425
14.12.1 Abrupt Completion of while Statement 426
14.13 The do Statement 426
14.13.1 Abrupt Completion of do Statement 427

14.14 The for Statement 428
14.14.1 The basic for Statement 428
14.14.1.1 Initialization of for Statement 429
14.14.1.2 Iteration of for Statement 429
14.14.1.3 Abrupt Completion of for Statement 430
14.14.2 The enhanced for statement 431
14.15 The break Statement 434
14.16 The continue Statement 436
14.17 The return Statement 438
14.18 The throw Statement 439
14.19 The synchronized Statement 441
14.20 The try statement 442The Java® Language Specification
xiii
14.20.1 Execution of try-catch 446
14.20.2 Execution of try-finally and try-catch-finally 447
14.20.3 try-with-resources 449
14.20.3.1 Basic try-with-resources 450
14.20.3.2 Extended try-with-resources 453
14.21 Unreachable Statements 454
15 Expressions 461
15.1 Evaluation, Denotation, and Result 461
15.2 Forms of Expressions 462
15.3 Type of an Expression 463
15.4 FP-strict Expressions 464
15.5 Expressions and Run-Time Checks 464
15.6 Normal and Abrupt Completion of Evaluation 466
15.7 Evaluation Order 468
15.7.1 Evaluate Left-Hand Operand First 468
15.7.2 Evaluate Operands before Operation 470
15.7.3 Evaluation Respects Parentheses and Precedence 471
15.7.4 Argument Lists are Evaluated Left-to-Right 472
15.7.5 Evaluation Order for Other Expressions 473
15.8 Primary Expressions 473
15.8.1 Lexical Literals 474
15.8.2 Class Literals 475
15.8.3 this 476
15.8.4 Qualified this 477
15.8.5 Parenthesized Expressions 477
15.9 Class Instance Creation Expressions 478
15.9.1 Determining the Class being Instantiated 479
15.9.2 Determining Enclosing Instances 481
15.9.3 Choosing the Constructor and its Arguments 483
15.9.4 Run-Time Evaluation of Class Instance Creation
Expressions 485
15.9.5 Anonymous Class Declarations 487
15.9.5.1 Anonymous Constructors 487
15.10 Array Creation and Access Expressions 488
15.10.1 Array Creation Expressions 488
15.10.2 Run-Time Evaluation of Array Creation Expressions 489
15.10.3 Array Access Expressions 493
15.10.4 Run-Time Evaluation of Array Access Expressions 493
15.11 Field Access Expressions 496
15.11.1 Field Access Using a Primary 496
15.11.2 Accessing Superclass Members using super 499

15.12 Method Invocation Expressions 500
15.12.1 Compile-Time Step 1: Determine Class or Interface to Search 502
15.12.2 Compile-Time Step 2: Determine Method Signature 504
15.12.2.1 Identify Potentially Applicable Methods 510he Java® Language Specification xiv
15.12.2.2 Phase 1: Identify Matching Arity Methods Applicable by Strict Invocation 513
15.12.2.3 Phase 2: Identify Matching Arity Methods Applicable by Loose Invocation 514
15.12.2.4 Phase 3: Identify Methods Applicable by Variable Arity Invocation 514
15.12.2.5 Choosing the Most Specific Method 515
15.12.2.6 Method Invocation Type 518
15.12.3 Compile-Time Step 3: Is the Chosen Method Appropriate? 518
15.12.4 Run-Time Evaluation of Method Invocation 521
15.12.4.1 Compute Target Reference (If Necessary) 522
15.12.4.2 Evaluate Arguments 523
15.12.4.3 Check Accessibility of Type and Method 524
15.12.4.4 Locate Method to Invoke 525
15.12.4.5 Create Frame, Synchronize, Transfer Control 529
15.13 Method Reference Expressions 531
15.13.1 Compile-Time Declaration of a Method Reference 534
15.13.2 Type of a Method Reference 539
15.13.3 Run-time Evaluation of Method References 541
15.14 Postfix Expressions 544
15.14.1 Expression Names 545
15.14.2 Postfix Increment Operator ++ 545
15.14.3 Postfix Decrement Operator -- 545
15.15 Unary Operators 546
15.15.1 Prefix Increment Operator ++ 548
15.15.2 Prefix Decrement Operator -- 548
15.15.3 Unary Plus Operator + 549
15.15.4 Unary Minus Operator - 549
15.15.5 Bitwise Complement Operator ~ 550
15.15.6 Logical Complement Operator ! 550
15.16 Cast Expressions 550
15.17 Multiplicative Operators 552
15.17.1 Multiplication Operator * 553
15.17.2 Division Operator / 554
15.17.3 Remainder Operator % 555
15.18 Additive Operators 558
15.18.1 String Concatenation Operator + 558
15.18.2 Additive Operators (+ and -) for Numeric Types 561
15.19 Shift Operators 563
15.20 Relational Operators 564
15.20.1 Numerical Comparison Operators <, <=, >, and >= 564
15.20.2 Type Comparison Operator instanceof 566
15.21 Equality Operators 567
15.21.1 Numerical Equality Operators == and != 567
15.21.2 Boolean Equality Operators == and != 568
15.21.3 Reference Equality Operators == and != 569
15.22 Bitwise and Logical Operators 569
15.22.1 Integer Bitwise Operators &, ^, and | 570The Java® Language Specification

15.22.2 Boolean Logical Operators &, ^, and | 571
15.23 Conditional-And Operator && 571
15.24 Conditional-Or Operator || 572
15.25 Conditional Operator ? : 573
15.25.1 Boolean Conditional Expressions 580
15.25.2 Numeric Conditional Expressions 580
15.25.3 Reference Conditional Expressions 581
15.26 Assignment Operators 582
15.26.1 Simple Assignment Operator = 583
15.26.2 Compound Assignment Operators 589
15.27 Lambda Expressions 595
15.27.1 Lambda Parameters 597
15.27.2 Lambda Body 600
15.27.3 Type of a Lambda Expression 603
15.27.4 Run-time Evaluation of Lambda Expressions 605
15.28 Constant Expressions 606
16 Definite Assignment 609
16.1 Definite Assignment and Expressions 615
16.1.1 Boolean Constant Expressions 615
16.1.2 Conditional-And Operator && 615
16.1.3 Conditional-Or Operator || 616
16.1.4 Logical Complement Operator ! 616
16.1.5 Conditional Operator ? : 616
16.1.6 Conditional Operator ? : 617
16.1.7 Other Expressions of Type boolean 617
16.1.8 Assignment Expressions 617
16.1.9 Operators ++ and -- 618
16.1.10 Other Expressions 618
16.2 Definite Assignment and Statements 619
16.2.1 Empty Statements 619
16.2.2 Blocks 619
16.2.3 Local Class Declaration Statements 621
16.2.4 Local Variable Declaration Statements 621
16.2.5 Labeled Statements 621
16.2.6 Expression Statements 622
16.2.7 if Statements 622
16.2.8 assert Statements 622
16.2.9 switch Statements 623
16.2.10 while Statements 623
16.2.11 do Statements 624
16.2.12 for Statements 624
16.2.12.1 Initialization Part of for Statement 625
16.2.12.2 Incrementation Part of for Statement 625
16.2.13 break, continue, return, and throw Statements 626
16.2.14 synchronized Statements 626
16.2.15 try Statements 626he Java® Language Specification

16.3 Definite Assignment and Parameters 628
16.4 Definite Assignment and Array Initializers 628
16.5 Definite Assignment and Enum Constants 628
16.6 Definite Assignment and Anonymous Classes 629
16.7 Definite Assignment and Member Types 629
16.8 Definite Assignment and Static Initializers 630

16.9 Definite Assignment, Constructors, and Instance Initializers 630
17 Threads and Locks 633
17.1 Synchronization 634
17.2 Wait Sets and Notification 634
17.2.1 Wait 635
17.2.2 Notification 636
17.2.3 Interruptions 637
17.2.4 Interactions of Waits, Notification, and Interruption 637
17.3 Sleep and Yield 638
17.4 Memory Model 639
17.4.1 Shared Variables 642
17.4.2 Actions 642
17.4.3 Programs and Program Order 643
17.4.4 Synchronization Order 644
17.4.5 Happens-before Order 645
17.4.6 Executions 648
17.4.7 Well-Formed Executions 649
17.4.8 Executions and Causality Requirements 649
17.4.9 Observable Behavior and Nonterminating Executions 652
17.5 final Field Semantics 654
17.5.1 Semantics of final Fields 656
17.5.2 Reading final Fields During Construction 656
17.5.3 Subsequent Modification of final Fields 657
17.5.4 Write-protected Fields 658
17.6 Word Tearing 659
17.7 Non-atomic Treatment of double and long 660
18 Type Inference 661
18.1 Concepts and Notation 662
18.1.1 Inference Variables 662
18.1.2 Constraint Formulas 663
18.1.3 Bounds 663
18.2 Reduction 665
18.2.1 Expression Compatibility Constraints 665
18.2.2 Type Compatibility Constraints 670
18.2.3 Subtyping Constraints 670
18.2.4 Type Equality Constraints 672
18.2.5 Checked Exception Constraints 673
18.3 Incorporation 675
18.3.1 Complementary Pairs of Bounds 676he Java® Language Specification
xvii
18.3.2 Bounds Involving Capture Conversion 676
18.4 Resolution 677
18.5 Uses of Inference 679
18.5.1 Invocation Applicability Inference 680
18.5.2 Invocation Type Inference 681
18.5.3 Functional Interface Parameterization Inference 687
18.5.4 More Specific Method Inference 688
19 Syntax 691
Index 717
A Limited License Grant 757
・・・・・・・(收起)

The Java Virtual Machine Specification, Java SE 8 Edition_下载链接1_

标签

Java

JVM

计算机

评论

疫情期间太无聊又开始翻看了。

读的Java SE 11版本的

啰嗦的占大多数，不过还是解决了一些问题

The Java Virtual Machine Specification, Java SE 8 Edition_下载链接1_

书评

1.
边敲边实践，本人用的sublime编辑器再加上javap插件，屏幕开两栏，左边java代码，右边bytecode，对照着看挺好 2.在线文档，可结合着看，地址：
http://www.weblearn.hs-bremen.de/risse/RST/docs/JavaVM/vmspec.pdf 3.
以前看过《自制编程语言》其中有门语言就类似java，作...

规范性的东西，不同的JVM厂商有不同的JVM实现。很多东西，JVM规范并没有强制要求，具体还是要看JVM实现。这本书写的还是不错，但是不容忽视的一点是，看着看着，你一定会睡着。你真的会睡着的。 这本书的封面不错，看起来比较有感觉。

---
sun的vmspec是免费的在线的，看起来却很轻松，非常适合想了解vm底层的java程序员，看完之后对bytecode应该能看懂了

The Java® Virtual Machine Specification Java SE 8 Edition Tim Lindholm Frank Yellin Gilad Bracha Alex Buckley 2015-02-13 Online version: http://docs.oracle.com/javase/specs/jvms/se8/html/

The Java Virtual Machine Specification, Java SE 8 Edition_下载链接1_