The Java® Virtual Machine Specification
Java SE 8 Edition
This page intentionally left blank
To Sophia and Susan, in deepest appreciation.
This page intentionally left blank
Table of Contents

Preface to the Java SE 8 Edition  xv

1  Introduction  1
   1.1  A Bit of History  1
   1.2  The Java Virtual Machine  2
   1.3  Organization of the Specification  3
   1.4  Notation  4
   1.5  Feedback  4

2  The Structure of the Java Virtual Machine  5
   2.1  The class File Format  5
   2.2  Data Types  6
   2.3  Primitive Types and Values  6
      2.3.1  Integral Types and Values  7
      2.3.2  Floating-Point Types, Value Sets, and Values  8
      2.3.3  The returnAddress Type and Values  10
      2.3.4  The boolean Type  10
   2.4  Reference Types and Values  11
   2.5  Run-Time Data Areas  11
      2.5.1  The pc Register  12
      2.5.2  Java Virtual Machine Stacks  12
      2.5.3  Heap  13
      2.5.4  Method Area  13
      2.5.5  Run-Time Constant Pool  14
      2.5.6  Native Method Stacks  14
   2.6  Frames  15
      2.6.1  Local Variables  16
      2.6.2  Operand Stacks  17
      2.6.3  Dynamic Linking  18
      2.6.4  Normal Method Invocation Completion  18
      2.6.5  Abrupt Method Invocation Completion  18
   2.7  Representation of Objects  19
   2.8  Floating-Point Arithmetic  19
      2.8.1  Java Virtual Machine Floating-Point Arithmetic and IEEE 754  19
      2.8.2  Floating-Point Modes  20
      2.8.3  Value Set Conversion  20
   2.9  Special Methods  22
   2.10 Exceptions  23
   2.11 Instruction Set Summary  25
4.10.1.5 Type Checking Abstract and Native Methods 184
4.10.1.6 Type Checking Methods with Code 187
4.10.1.7 Type Checking Load and Store Instructions 194
4.10.1.8 Type Checking for protected Members 196
4.10.1.9 Type Checking Instructions 199
4.10.2 Verification by Type Inference 319
4.10.2.1 The Process of Verification by Type Inference 319
4.10.2.2 The Bytecode Verifier 319
4.10.2.3 Values of Types long and double 323
4.10.2.4 Instance Initialization Methods and Newly Created Objects 323
4.10.2.5 Exceptions and finally 325
4.11 Limitations of the Java Virtual Machine 327

5 Loading, Linking, and Initializing 329
5.1 The Run-Time Constant Pool 329
5.2 Java Virtual Machine Startup 332
5.3 Creation and Loading 332
5.3.1 Loading Using the Bootstrap Class Loader 334
5.3.2 Loading Using a User-defined Class Loader 335
5.3.3 Creating Array Classes 336
5.3.4 Loading Constraints 336
5.3.5 Deriving a Class from a class File Representation 338
5.4 Linking 339
5.4.1 Verification 340
5.4.2 Preparation 340
5.4.3 Resolution 341
5.4.3.1 Class and Interface Resolution 342
5.4.3.2 Field Resolution 343
5.4.3.3 Method Resolution 344
5.4.3.4 Interface Method Resolution 346
5.4.3.5 Method Type and Method Handle Resolution 347
5.4.3.6 Call Site Specifier Resolution 350
5.4.4 Access Control 351
5.4.5 Overriding 352
5.5 Initialization 352
5.6 Binding Native Method Implementations 355
5.7 Java Virtual Machine Exit 355

6 The Java Virtual Machine Instruction Set 357
6.1 Assumptions: The Meaning of "Must" 357
6.2 Reserved Opcodes 358
6.3 Virtual Machine Errors 358
6.4 Format of Instruction Descriptions 359
   mnemonic 360
6.5 Instructions 362
   aaload 363
aastore 364
aconst_null 366
aload 367
aload_<n> 368
anewarray 369
areturn 370
arraylength 371
astore 372
astore_<n> 373
athrow 374
baload 376
bastore 377
bipush 378
caload 379
castore 380
checkcast 381
d2f 383
d2i 384
d2l 385
dadd 386
daload 388
dastore 389
dcmp<op> 390
dconst_<d> 392
div 393
dload 395
dload_<n> 396
dmul 397
dneg 399
drem 400
dreturn 402
dstore 403
dstore_<n> 404
dsub 405
dup 406
dup_x1 407
dup_x2 408
dup2 409
dup2_x1 410
dup2_x2 411
f2d 413
f2i 414
f2l 415
fadd 416
faload 418
fastore 419
fcmp<op> 420
fconst_<f> 422
fdiv 423
fload 425
fload_<n> 426
fmul 427
fneg 429
frem 430
freturn 432
fstore 433
fstore_<n> 434
fsub 435
getfield 436
getstatic 438
goto 440
goto_w 441
i2b 442
i2c 443
i2d 444
i2f 445
i2l 446
i2s 447
iadd 448
iaload 449
iand 450
iastore 451
iconst_<i> 452
idiv 453
if_acmp<cond> 454
if_icmp<cond> 455
if<cond> 457
ifnonnull 459
ifnull 460
iinc 461
iload 462
iload_<n> 463
imul 464
ineg 465
instanceof 466
invokedynamic 468
invokeinterface 473
invokespecial 477
invokestatic 481
invokevirtual 484
ior 489
irem 490
ireturn 491
ishl 492
ishr 493
istore 494
istore_<n> 495
isub 496
iushr 497
ixor 498
jsr 499
jsr_w 500
l2d 501
l2f 502
l2i 503
ladd 504
laload 505
land 506
lastore 507
lcmp 508
lconst_<l> 509
lde 510
lde_w 512
lde2_w 514
ldiv 515
lload 516
lload_<n> 517
lmul 518
lneg 519
lookupswitch 520
lor 522
lrem 523
lreturn 524
lshl 525
lshr 526
istore 527
istore_<n> 528
isub 529
lushr 530
lxor 531
monitorenter 532
monitorexit 534
multianewarray 536
new 538
newarray 540
nop 542
pop 543
pop2 544
putfield 545
putstatic 547
ret 549
return 550
saload 551
lastore 552
7 Opcode Mnemonics by Opcode 559

Index 563

A Limited License Grant 581
Preface to the Java SE 8 Edition

The Java SE 8 Edition of *The Java® Virtual Machine Specification* incorporates all the changes that have been made to the Java Virtual Machine since the Java SE 7 Edition in 2011. In addition, numerous corrections and clarifications have been made to align with popular implementations of the Java Virtual Machine.

This Edition continues the tradition of specifying the *abstract* Java Virtual Machine, serving as documentation for a concrete implementation only as a blueprint documents a house. An implementation of the Java Virtual Machine must embody this specification, but is constrained by it only where absolutely necessary.

Notable changes to the Java programming language in Java SE 8 have brought corresponding changes to the Java Virtual Machine. To maximize binary compatibility, it has been desirable to specify default methods directly in the Java Virtual Machine, rather than relying on compiler magic that might not be portable across vendors or product releases, and is certainly not applicable to pre-existing class files. In the context of JSR 335, *Lambda Expressions for the Java Programming Language*, Dan Smith at Oracle consulted with implementers to determine how best to integrate default methods into the constant pool and method structures, the method and interface method resolution algorithms, and the bytecode instruction set. JSR 335 also introduced private and static methods in interfaces at the class file level; they too have been carefully integrated with interface method resolution.

A theme of Java SE 8 is co-evolution of the Java SE platform libraries with the Java Virtual Machine. A small but useful example is support for method parameter names at run time: storing such names in the class file structure goes hand in hand with offering a standard API to retrieve them (java.lang.reflect.Parameter). This illustrates an interesting development in the class file structure over the years: the First Edition of this specification defined six attributes, of which three were deemed critical to the Java Virtual Machine, while this Java SE 8 Edition defines 23 attributes, of which five are deemed critical to the Java Virtual Machine; that is to say, attributes now exist primarily to support libraries and tools rather than the Java Virtual Machine itself. To help readers understand the class file structure, this specification more clearly documents the role of each attribute and the constraints placed upon it.
Many colleagues in the Java Platform Group at Oracle have provided valuable support to this specification: Mandy Chung, Joe Darcy, Joel Franck, Staffan Friberg, Yuri Gaevsky, Jon Gibbons, Jeannette Hung, Eric McCorkle, Matherey Nunez, Mark Reinhold, John Rose, Georges Saab, Steve Sides, Bernard Traversat, Michel Trudeau, and Mikael Vidstedt. Particular thanks to Dan Heidinga (IBM), Karen Kinnear, Keith McGuigan, and Harold Seigel for their ironclad commitment to compatibility and security in popular Java Virtual Machine implementations.

Alex Buckley
Santa Clara, California
March, 2014
1.1 A Bit of History

The Java® programming language is a general-purpose, concurrent, object-oriented language. Its syntax is similar to C and C++, but it omits many of the features that make C and C++ complex, confusing, and unsafe. The Java platform was initially developed to address the problems of building software for networked consumer devices. It was designed to support multiple host architectures and to allow secure delivery of software components. To meet these requirements, compiled code had to survive transport across networks, operate on any client, and assure the client that it was safe to run.

The popularization of the World Wide Web made these attributes much more interesting. Web browsers enabled millions of people to surf the Net and access media-rich content in simple ways. At last there was a medium where what you saw and heard was essentially the same regardless of the machine you were using and whether it was connected to a fast network or a slow modem.

Web enthusiasts soon discovered that the content supported by the Web's HTML document format was too limited. HTML extensions, such as forms, only highlighted those limitations, while making it clear that no browser could include all the features users wanted. Extensibility was the answer.

The HotJava browser first showcased the interesting properties of the Java programming language and platform by making it possible to embed programs inside HTML pages. Programs are transparently downloaded into the browser along with the HTML pages in which they appear. Before being accepted by the browser, programs are carefully checked to make sure they are safe. Like HTML pages, compiled programs are network- and host-independent. The programs behave the same way regardless of where they come from or what kind of machine they are being loaded into and run on.
A Web browser incorporating the Java platform is no longer limited to a predetermined set of capabilities. Visitors to Web pages incorporating dynamic content can be assured that their machines cannot be damaged by that content. Programmers can write a program once, and it will run on any machine supplying a Java run-time environment.

1.2 The Java Virtual Machine

The Java Virtual Machine is the cornerstone of the Java platform. It is the component of the technology responsible for its hardware- and operating system-independence, the small size of its compiled code, and its ability to protect users from malicious programs.

The Java Virtual Machine is an abstract computing machine. Like a real computing machine, it has an instruction set and manipulates various memory areas at run time. It is reasonably common to implement a programming language using a virtual machine; the best-known virtual machine may be the P-Code machine of UCSD Pascal.

The first prototype implementation of the Java Virtual Machine, done at Sun Microsystems, Inc., emulated the Java Virtual Machine instruction set in software hosted by a handheld device that resembled a contemporary Personal Digital Assistant (PDA). Oracle's current implementations emulate the Java Virtual Machine on mobile, desktop and server devices, but the Java Virtual Machine does not assume any particular implementation technology, host hardware, or host operating system. It is not inherently interpreted, but can just as well be implemented by compiling its instruction set to that of a silicon CPU. It may also be implemented in microcode or directly in silicon.

The Java Virtual Machine knows nothing of the Java programming language, only of a particular binary format, the class file format. A class file contains Java Virtual Machine instructions (or bytecodes) and a symbol table, as well as other ancillary information.

For the sake of security, the Java Virtual Machine imposes strong syntactic and structural constraints on the code in a class file. However, any language with functionality that can be expressed in terms of a valid class file can be hosted by the Java Virtual Machine. Attracted by a generally available, machine-independent platform, implementors of other languages can turn to the Java Virtual Machine as a delivery vehicle for their languages.
The Java Virtual Machine specified here is compatible with the Java SE 8 platform, and supports the Java programming language specified in *The Java Language Specification, Java SE 8 Edition*.

### 1.3 Organization of the Specification

Chapter 2 gives an overview of the Java Virtual Machine architecture.

Chapter 3 introduces compilation of code written in the Java programming language into the instruction set of the Java Virtual Machine.

Chapter 4 specifies the class file format, the hardware- and operating system-independent binary format used to represent compiled classes and interfaces.

Chapter 5 specifies the start-up of the Java Virtual Machine and the loading, linking, and initialization of classes and interfaces.

Chapter 6 specifies the instruction set of the Java Virtual Machine, presenting the instructions in alphabetical order of opcode mnemonics.

Chapter 7 gives a table of Java Virtual Machine opcode mnemonics indexed by opcode value.

In the Second Edition of *The Java® Virtual Machine Specification*, Chapter 2 gave an overview of the Java programming language that was intended to support the specification of the Java Virtual Machine but was not itself a part of the specification. In *The Java Virtual Machine Specification, Java SE 8 Edition*, the reader is referred to *The Java Language Specification, Java SE 8 Edition* for information about the Java programming language. References of the form: (JLS §x.y) indicate where this is necessary.

1.4 Notation

Throughout this specification we refer to classes and interfaces drawn from the Java SE platform API. Whenever we refer to a class or interface (other than those declared in an example) using a single identifier $N$, the intended reference is to the class or interface named $N$ in the package `java.lang`. We use the fully qualified name for classes or interfaces from packages other than `java.lang`.

Whenever we refer to a class or interface that is declared in the package `java` or any of its subpackages, the intended reference is to that class or interface as loaded by the bootstrap class loader (§5.3.1).

Whenever we refer to a subpackage of a package named `java`, the intended reference is to that subpackage as determined by the bootstrap class loader.

The use of fonts in this specification is as follows:

- A fixed width font is used for Java Virtual Machine data types, exceptions, errors, class file structures, Prolog code, and Java code fragments.
- *Italic* is used for Java Virtual Machine "assembly language", its opcodes and operands, as well as items in the Java Virtual Machine's run-time data areas. It is also used to introduce new terms and simply for emphasis.

Non-normative information, designed to clarify the specification, is given in smaller, indented text.

This is non-normative information. It provides intuition, rationale, advice, examples, etc.

1.5 Feedback

Readers may send feedback about errors, omissions, and ambiguities in this specification to jvms-comments_ww@oracle.com.

Questions concerning the generation and manipulation of class files by javac (the reference compiler for the Java programming language) may be sent to compiler-dev@openjdk.java.net.
A

a bit of history, 1
aaload, 201, 363
stack map frame representation, 180
aastore, 202, 364
abrupt method invocation completion, 18
Exceptions, 23
invokedynamic, 471
synchronization, 67
access control, 351
class and interface resolution, 343
field resolution, 343
interface method resolution, 347
method resolution, 345
type checking for protected members, 196
accessing the run-time constant pool, 46
accessors for Java Virtual Machine artifacts, 169
type checking instructions, 199
verification by type checking, 169
aconst_null, 203, 366
actual and computational types in the Java Virtual Machine, 29
types and the Java Virtual Machine, 29, 29
aload, 367
astore, 372
wide, 557
aload, aload_<n>, 204
aload_<n>, 368
astore_<n>, 373
anewarray, 205, 369

multianewarray, 537
AnnotationDefault attribute, 153
annotations, 67
annotations, 67
areturn, 206, 370
arithmetic, 45
arithmetic instructions, 30
control transfer instructions, 35
array class loading, 336
creation and loading, 333
loading constraints, 337
array type codes, 540
arraylength, 207, 371
getfield, 437
stack map frame representation, 180
arrays, 55
assumptions: the meaning of "must", 357
astore, 372
aload, 367
wide, 557
astore, astore_<n>, 208
astore_<n>, 373
aload_<n>, 368
athrow, 209, 374
abrupt method invocation completion, 18
Exceptions, 23
attributes, 95
ClassFile structure, 74, 74
Code attribute, 105, 106
fields, 92, 92
methods, 95, 95
INDEX

B
baload, 210, 376
  boolean type, 10
  newarray, 541
  stack map frame representation, 180
bastore, 211, 377
  boolean type, 10
  newarray, 541
  stack map frame representation, 180
binary class and interface names, 74
  annotations, 68
  CONSTANT_Class_info structure, 80
  creation and loading, 333
  element_value structure, 133
  field descriptors, 76
  run-time constant pool, 330, 330
binding native method implementations, 355
  invokeinterface, 474
  invokespecial, 479
  invokestatic, 482
  invokevirtual, 485
bipush, 212, 378
boolean type, 10
  primitive types and values, 6
bootstrap loader, 334
  creation and loading, 333
  Java Virtual Machine startup, 332
  loading constraints, 337
  notation, 4
BootstrapMethods attribute, 154
  call site specifier resolution, 350
  CONSTANT_InvokeDynamic_info structure, 90
bytecode behaviors for method handles, 348
  method type and method handle resolution, 348
bytecode verifier, 319

C
call site specifier resolution, 350
  invokedynamic, 468
caload, 213, 379
castore, 214, 380
checkcast, 215, 381
  instanceof, 467
class access and property modifiers, 71
  ClassFile structure, 71, 72, 72
class and interface resolution, 342
  anewarray, 369, 369
  checkcast, 381, 382
  class and interface resolution, 343
  deriving a class from a class file representation, 338, 339
  field resolution, 343
  instanceof, 466, 467
  interface method resolution, 346
  ldc, 510, 511
  ldc_w, 512, 513
  method resolution, 344, 344
  method type and method handle resolution, 347
  multianewarray, 536, 537
  new, 538, 538
class file format, 5, 69
  assumptions: the meaning of "must", 357
  creation and loading, 333
  reserved opcodes, 358
class libraries, 37
  creation and loading, 332
  initialization, 353
class loading, 332
  access control, 351
  class and interface resolution, 342
  creating array classes, 336
  format checking, 158
getfield, 436
invespecial, 477
invokevirtual, 484
putfield, 545
run-time constant pool, 14, 329
verification, 340
verification type system, 174

**ClassFile structure**, 70
annotations, 68
attributes, 95
BootstrapMethods attribute, 154
Deprecation attribute, 129
deriving a class from a class file
representation, 338, 338
EnclosingMethod attribute, 116
format checking, 158
InnerClasses attribute, 114
invespecial, 477
limitations of the Java Virtual Machine, 327,
327, 327, 327
loading using a user-defined class loader, 335
RuntimeInvisibleAnnotations attribute, 135
RuntimeInvisibleTypeAnnotations attribute, 152
RuntimeVisibleAnnotations attribute, 130
RuntimeVisibleTypeAnnotations attribute, 139
Signature attribute, 118
SourceDebugExtension attribute, 124
SourceFile attribute, 123
Synthetic attribute, 118
verification by type checking, 167

**Code attribute**, 102
attributes, 95
constraints on Java Virtual Machine Code, 159
defining and naming new attributes, 101
Exceptions, 25
frames, 15
limitations of the Java Virtual Machine, 327,
327
LineNumberTable attribute, 124
local variables, 16
LocalVariableTable attribute, 126
LocalVariableTypeTable attribute, 128
operand stacks, 17
RuntimeInvisibleTypeAnnotations attribute, 152
RuntimeVisibleTypeAnnotations attribute, 139
StackMapTable attribute, 106
throwing and handling Exceptions, 61, 63
verification by type checking, 168
verification of class files, 167

**compiling finally**, 63
astore, 372
astore_<n>, 373
jsr, 499
jsr_w, 500
more control examples, 47
ret, 549

**compiling for the Java Virtual Machine**, 39
instance initialization methods and newly
created objects, 324

**compiling switches**, 57
more control examples, 47

**constant pool**, 78, 329
ClassFile structure, 71
format checking, 158
getfield, 436
getstatic, 438
invokedynamic, 468, 468, 471
invokeinterface, 473
invokespecial, 477
invokestatic, 481
invokevirtual, 484
INDEX

ldc, 510, 510, 510, 510
ldc2_w, 514
ldc_w, 512, 512, 512

method type and method handle resolution, 348
putfield, 545
putstatic, 547
resolution, 342
run-time constant pool, 14, 329

constant pool tags, 79
constant pool, 79

constant value attribute types, 102
ConstantValue attribute, 102

CONSTANT_Class_info structure, 79
arrays, 57
binary class and interface names, 74
BootstrapMethods attribute, 156
ClassFile structure, 72
Code attribute, 105
CONSTANT_Fieldref_info,
CONSTANT_Methodref_info, and
CONSTANT_InterfaceMethodref_info structures, 81
EnclosingMethod attribute, 117
Exceptions attribute, 113
InnerClasses attribute, 114
ldc_w, 512
run-time constant pool, 330
StackMapTable attribute, 109

CONSTANT_Fieldref_info,
CONSTANT_Methodref_info, and
CONSTANT_InterfaceMethodref_info structures, 80
CONSTANT_MethodHandle_info structure, 88, 88, 88
instruction representation, 177
run-time constant pool, 330, 330, 330

CONSTANT_Integer_info and
CONSTANT_Float_info structures, 82
BootstrapMethods attribute, 156
floating-point types, value sets, and values, 9
ldc, 511
ldc_w, 513
run-time constant pool, 332, 332

CONSTANT_InvokeDynamic_info structure, 89
BootstrapMethods attribute, 154
instruction representation, 177
run-time constant pool, 331

CONSTANT_Long_info and
CONSTANT_Double_info structures, 83
BootstrapMethods attribute, 156
ClassFile structure, 71
floating-point types, value sets, and values, 9
ldc2_w, 514
run-time constant pool, 332, 332

CONSTANT_MethodHandle_info structure, 87
BootstrapMethods attribute, 155, 155, 156
run-time constant pool, 330

CONSTANT_MethodType_info structure, 89
BootstrapMethods attribute, 156
run-time constant pool, 331

CONSTANT_NameAndType_info structure, 85
binary class and interface names, 74
CONSTANT_Fieldref_info,
CONSTANT_Methodref_info, and
CONSTANT_InterfaceMethodref_info structures, 81
CONSTANT_InvokeDynamic_info structure, 90
EnclosingMethod attribute, 117
run-time constant pool, 332

CONSTANT_String_info structure, 81
BootstrapMethods attribute, 156
run-time constant pool, 331
**CONSTANT_Utf8_info structure**, 85
AnnotationDefault attribute, 154
attributes, 95
binary class and interface names, 74
BootstrapMethods attribute, 155
Code attribute, 103
**CONSTANT_Class_info structure**, 80
**CONSTANT_MethodType_info structure**, 89
**CONSTANT_NameAndType_info structure**, 85, 85
**CONSTANT_String_info structure**, 82
ConstantValue attribute, 102
Deprecation attribute, 130
descriptors, 75
element_value structure, 133, 134, 134
EnclosingMethod attribute, 117
Exceptions attribute, 113
fields, 91, 92
InnerClasses attribute, 114, 115
limitations of the Java Virtual Machine, 328
LineNumberTable attribute, 125
LocalVariableTable attribute, 126, 127, 127
LocalVariableTypeTable attribute, 128, 129, 129
methods, 94
run-time constant pool, 332
RuntimeInvisibleAnnotations attribute, 135
RuntimeInvisibleParameterAnnotations attribute, 138
RuntimeVisibleAnnotations attribute, 131, 131, 132
RuntimeVisibleParameterAnnotations attribute, 136
Signature attribute, 119, 119
SourceDebugExtension attribute, 124, 124
SourceFile attribute, 123, 123
StackMapTable attribute, 106
Synthetic attribute, 118
**ConstantValue attribute**, 101
initialization, 354
limitations of the Java Virtual Machine, 328
**constraints**, 336
creating array classes, 336
deriving a class from a class file representation, 339
field resolution, 344
interface method resolution, 347
loading using a user-defined class loader, 335
method resolution, 345
preparation, 340
**constraints on Java Virtual Machine Code**, 159
Code attribute, 104
verification, 340, 340
verification of class files, 167
**control transfer instructions**, 34
creating **array classes**, 336
creation and loading, 333
loading constraints, 337
creation and loading, 332
access control, 351
class and interface resolution, 342
creating array classes, 336
format checking, 158
getfield, 436
invokespecial, 477
invokevirtual, 484
putfield, 545
run-time constant pool, 14, 329
verification, 340
verification type system, 174
INDEX

D

d2f, 383
d2f, d2i, d2l, 216
d2i, 384
d2l, 385
dadd, 217, 386
daload, 218, 388
dastore, 219, 389
data types, 6
   baload, 376
   bastore, 377
dcmp<op>, 220, 390
dconst_<d>, 221, 392
ddiv, 222, 393
defineclass, 338
   creation and loading, 334
   loading constraints, 337
   loading using a user-defined class loader, 335
   loading using the bootstrap class loader, 335
defining and naming new attributes, 101
   ClassFile structure, 74
   Code attribute, 106
   fields, 92
   methods, 95

Deprecated attribute, 129
deriving a class from a class file representation, 338
   creation and loading, 334
   loading constraints, 337
   loading using a user-defined class loader, 335
   loading using the bootstrap class loader, 335
descriptors, 75
   binary class and interface names, 74
   format checking, 158
dload, 395
   wide, 557
dload, dload_<n>, 223
dload_<n>, 396
dmul, 224, 397
dneg, 225, 399
drem, 226, 400
dreturn, 227, 402
dstore, 403
   wide, 557
dstore, dstore_<n>, 228
dstore_<n>, 404
dsub, 229, 405
dup, 230, 406
   operand stacks, 17
dup2, 233, 409
dup2_x1, 234, 410
dup2_x2, 235, 411
dup_x1, 231, 407
dup_x2, 232, 408
dynamic linking, 18

e

E

element_value structure, 132
EnclosingMethod attribute, 116

Exceptions, 23
   abrupt method invocation completion, 18
   athrow, 374
   Code attribute, 104
   normal method invocation completion, 18
   synchronization, 67
   throwing and handling Exceptions, 61, 63
   Virtual Machine errors, 358

Exceptions and finally, 325
   compiling finally, 63
   jsr, 499
   jsr_w, 500
   ret, 549

Exceptions attribute, 113
F
f2d, 413
f2d, f2i, f2l, 237
f2i, 414
f2l, 415
defadd, 238, 416
defaload, 239, 418
defastore, 240, 419
cmp<op>, 241, 420
fconst_, 242
fdiv, 243, 423
feedback, 4
field access and property flags, 91
  fields, 90, 91, 91, 91
field descriptors, 76
  CONSTANT_Class_info structure, 80
  CONSTANT_Fieldref_info,
  CONSTANT_Methodref_info, and
  CONSTANT_InterfaceMethodref_info structures, 81
  CONSTANT_NameAndType_info structure, 85
  element_value structure, 133
  fields, 90, 92
  instruction representation, 178
  LocalVariableTable attribute, 127
  method type and method handle resolution, 348
  putfield, 545
  putstatic, 547
  resolution, 342
fields, 90
  attributes, 95
  ClassFile structure, 73
  ConstantValue attribute, 101
  Deprecated attribute, 129
  RuntimeInvisibleAnnotations attribute, 135
  RuntimeInvisibleTypeAnnotations attribute, 152
  RuntimeVisibleAnnotations attribute, 130
  RuntimeVisibleTypeAnnotations attribute, 139
  Signature attribute, 118
  Synthetic attribute, 118
fload, 425
  types and the Java Virtual Machine, 26
  wide, 557
fload, fload_<n>, 244
fload_<n>, 426
floating-point arithmetic, 19
floating-point modes, 20
d2f, 383
f2d, 413
type conversion instructions, 32
floating-point types, value sets, and values, 8
  CONSTANT_Integer_info and
  CONSTANT_Float_info structures, 82
  CONSTANT_Long_info and
  CONSTANT_Double_info structures, 84
d2f, 383, 383
f2d, 413
floating-point modes, 20
invokedynamic, 469
ldc, 511
ldc2_w, 514
ldc_w, 513
INDEX

more control examples, 48
primitive types and values, 6
floating-point value set parameters, 9
  floating-point types, value sets, and values, 8, 9, 9
fmul, 245, 427
fneg, 246, 429
format checking, 158
  deriving a class from a class file representation, 338
format of examples, 39
format of instruction descriptions, 359
frames, 15
  aaload, 367
  aaload_<n>, 368
  anewarray, 369
  areturn, 370
  astore, 372
  astore_<n>, 373
  athrow, 374
  checkcast, 381
  dload, 395
  dload_<n>, 396
  dreturn, 402
  dstore, 403
  dstore_<n>, 404
  dynamic linking, 18
  fload, 425
  fload_<n>, 426
  format of instruction descriptions, 361
  ireturn, 432
  fstore, 433
  fstore_<n>, 434
  getfield, 436
  getstatic, 438
  iinc, 461
  iload, 462
  iload_<n>, 463
  instanceof, 466
  invokedynamic, 468
  invokeinterface, 473
  invokespecial, 477
  invokestatic, 481
  invokevirtual, 484
  ireturn, 491
  istore, 494
  istore_<n>, 495
  Java Virtual Machine stacks, 12
  ldc, 510
  ldc2_w, 514
  ldc_w, 512
  limitations of the Java Virtual Machine, 327, 327
  iload, 516
  iload_<n>, 517
  load and store instructions, 29
  local variables, 16
  lreturn, 524
  lstore, 527
  lstore_<n>, 528
  multianewarray, 536
  new, 538
  normal method invocation completion, 18
  operand stacks, 17
  pc register, 12
  putfield, 545
  putstatic, 547
  ret, 549
  return, 550
  use of constants, local variables, and control constructs, 41
  wide, 557
frem, 247, 430
freturn, 248, 432
fstore, 433
  wide, 557
fstore, fstore_<n>, 249
fstore_<n>, 434
fsub, 250, 435

G
getfield, 251, 436
getstatic, 252, 438
   initialization, 352
goto, 440
goto, goto_w, 253
goto_w, 441
grammar notation, 75
   signatures, 120

H
heap, 13

I
i2b, 442
i2b, i2c, i2d, i2f, i2l, i2s, 254
i2c, 443
i2d, 444
i2f, 445
i2l, 446
i2s, 447
iadd, 255, 448
   operand stacks, 17
iaload, 256, 449
iand, 257, 450
iastore, 258, 451
iconst_<p>, 452
idiv, 453
if<cond>, 261, 457
if_acmp<cond>, 259, 454
if_iemp<cond>, 260, 455
ifnonnull, 262, 459
ifnull, 263, 460
iinc, 264, 461
   wide, 557
iload, 462
   types and the Java Virtual Machine, 26
   wide, 557
iload, iload_<n>, 265
iload_<n>, 463
imul, 266, 464
ineg, 267, 465
initialization, 352
   ConstantValue attribute, 101
   creation and loading, 334
   getstatic, 438, 439
   invokestatic, 481, 483
   new, 538
   preparation, 340
   putstatic, 547, 548, 548
   special methods, 22
   InnerClasses attribute, 114
   instance initialization methods and newly
   created objects, 323
instanceof, 268, 466
   checkcast, 382
instruction representation, 177
   accessors for Java Virtual Machine artifacts,
   171
   verification by type checking, 169
instruction set summary, 25
instructions, 362
   static constraints, 159
integral types and values, 7
   invokedynamic, 469
   primitive types and values, 6
interface method resolution, 346
   invokeinterface, 473, 475
   invokespecial, 477
INDEX

loading constraints, 337
internal form of names, 74
interpretation of field descriptors, 77
field descriptors, 76
verification type system, 174
interpretation of tag values as types, 133
element_value structure, 132, 133
interpretation of target_type values (part 1),
142
RuntimeVisibleTypeAnnotations attribute, 141
interpretation of target_type values (part 2),
143
RuntimeVisibleTypeAnnotations attribute, 141
interpretation of type_path_kind values, 150
type_path structure, 150
introduction, 1
invokedynamic, 269, 468
BootstrapMethods attribute, 154
CONSTANT_InvokeDynamic_info structure, 89
run-time constant pool, 331
invokeinterface, 270, 473
invokespecial, 271, 477
ClassFile structure, 72
special methods, 22
invokestatic, 274, 481
initialization, 352
invokevirtual, 275, 484
invokedynamic, 468, 471
invokespecial, 480
special methods, 23
invoking methods, 51
ior, 276, 489
irem, 277, 490
ireturn, 278, 491
ishl, 492
ishr, 493
istore, 494
    wide, 557
istore, istore_<n>, 280
istore_<n>, 495
isub, 281, 496
iushr, 497
ixor, 282, 498

J
Java Virtual Machine, 2
Java Virtual Machine exit, 355
Java Virtual Machine floating-point arithmetic and IEEE 754, 19
Java Virtual Machine instruction set, 357
Java Virtual Machine stacks, 12
frames, 15
Java Virtual Machine startup, 332
    initialization, 353
jsr, 499
    ret, 549, 549
    return_address type and values, 10
jsr_w, 500
    ret, 549
    return_address type and values, 10

L
l2d, 501
l2d, l2f, l2i, 283
l2f, 502
l2i, 503
ladd, 284, 504
laload, 285, 505
land, 286, 506
lastore, 287, 507
lcmp, 288, 508
lconst_<i>, 289, 509
ldc, 510
call site specifier resolution, 351
ldc_w, 513
ldc, ldc_w, ldc2_w, 290
ldc2_w, 514
ldc_w, 512
ldiv, 291, 515
limitations of the Java Virtual Machine, 327
goto_w, 441
jsr_w, 500
method descriptors, 78
LineNumberTable attribute, 124
linking, 339
verification of class files, 166
lload, 516
wide, 557
lload, lload_<n>, 292
lload_<n>, 517
lmul, 293, 518
lneg, 294, 519
load and store instructions, 29
loading constraints, 336
creating array classes, 336
deriving a class from a class file
representation, 339
field resolution, 344
interface method resolution, 347
loading using a user-defined class loader, 335
method resolution, 345
preparation, 340
loading using a user-defined class loader, 335
creation and loading, 333
loading constraints, 337
loading using the bootstrap class loader, 334
creation and loading, 333
Java Virtual Machine startup, 332
loading constraints, 337
notation, 4
loading, linking, and initializing, 329
local variables, 16
Code attribute, 103
frames, 15
load and store instructions, 29
method descriptors, 78
LocalVariableTable attribute, 126
LocalVariableTypeTable attribute, 128
location of enclosing attribute for target_type
values, 144
RuntimeVisibleTypeAnnotations attribute, 141
lookupswitch, 295, 520
instruction set summary, 25
lor, 296, 522
lrem, 297, 523
lreturn, 298, 524
lshl, 525
lshr, lshl, lshr, lshl, lshr, 299
lshr, 526
lstore, 527
wide, 557
lstore, lload_<n>, 300
lload_<n>, 528
lsub, 301, 529
lushr, 530
lxor, 302, 531
M
method access and property flags, 93
methods, 93, 93, 93, 94, 94
method area, 13
creation and loading, 332
run-time constant pool, 14
method descriptors, 77
INDEX

areturn, 370
CONSTANT_Fieldref_info, 370
CONSTANT_Methodref_info, and CONSTANT_InterfaceMethodref_info structures, 81
CONSTANT_InvokeDynamic_info structure, 90
CONSTANT_MethodType_info structure, 89
CONSTANT_NameAndType_info structure, 85
element_value structure, 134
instruction representation, 178
invokeinterface, 473
invokespecial, 477
invokestatic, 481
invokevirtual, 484
invoking methods, 51
limitations of the Java Virtual Machine, 327
method type and method handle resolution, 347, 348
MethodParameters attribute, 157
methods, 92, 94
run-time constant pool, 331
RuntimeInvisibleParameterAnnotations attribute, 138
RuntimeVisibleParameterAnnotations attribute, 137
special methods, 22
structural constraints, 164, 164
method descriptors for method handles, 349
method type and method handle resolution, 349
method invocation and return instructions, 35
normal method invocation completion, 18
synchronization, 36
method resolution, 344
interface method resolution, 346
invokeinterface, 474
invokespecial, 477, 478, 479
invokestatic, 481, 482
invokevirtual, 484, 485, 487
invoking methods, 53
loading constraints, 337
method type and method handle resolution, 347
call site specifier resolution, 350, 351, 351
CONSTANT_MethodHandle_info structure, 88
initialization, 353
invokevirtual, 486, 486, 487
ldc, 510, 511
ldc_w, 512, 513
special methods, 23
MethodParameters attribute, 156
methods, 92
AnnotationDefault attribute, 153
attributes, 95
ClassFile structure, 73
Code attribute, 102
Deprecated attribute, 129
Exceptions attribute, 113
floating-point modes, 20
method type and method handle resolution, 350
MethodParameters attribute, 156
RuntimeInvisibleAnnotations attribute, 135
RuntimeInvisibleParameterAnnotations attribute, 137
RuntimeVisibleAnnotations attribute, 136
RuntimeVisibleParameterAnnotations attribute, 138
RuntimeVisibleTypeAnnotations attribute, 152
special methods, 22
structural constraints, 164, 164
method descriptors for method handles, 349
method type and method handle resolution, 349
method invocation and return instructions, 35
normal method invocation completion, 18
synchronization, 36
method resolution, 344
interface method resolution, 346
invokeinterface, 474
invokespecial, 477, 478, 479
invokestatic, 481, 482
invokevirtual, 484, 485, 487
invoking methods, 53
loading constraints, 337
method type and method handle resolution, 347
call site specifier resolution, 350, 351, 351
CONSTANT_MethodHandle_info structure, 88
initialization, 353
invokevirtual, 486, 486, 487
ldc, 510, 511
ldc_w, 512, 513
special methods, 23
MethodParameters attribute, 156
methods, 92
AnnotationDefault attribute, 153
attributes, 95
ClassFile structure, 73
Code attribute, 102
Deprecated attribute, 129
Exceptions attribute, 113
floating-point modes, 20
method type and method handle resolution, 350
MethodParameters attribute, 156
RuntimeInvisibleAnnotations attribute, 135
RuntimeInvisibleParameterAnnotations attribute, 137
RuntimeVisibleAnnotations attribute, 136
RuntimeVisibleParameterAnnotations attribute, 138
RuntimeVisibleTypeAnnotations attribute, 152
special methods, 22
structural constraints, 164, 164
method descriptors for method handles, 349
method type and method handle resolution, 349
method invocation and return instructions, 35
normal method invocation completion, 18
synchronization, 36
method resolution, 344
interface method resolution, 346
invokeinterface, 474
invokespecial, 477, 478, 479
invokestatic, 481, 482
invokevirtual, 484, 485, 487
invoking methods, 53
loading constraints, 337
method type and method handle resolution, 347
call site specifier resolution, 350, 351, 351
CONSTANT_MethodHandle_info structure, 88
initialization, 353
invokevirtual, 486, 486, 487
ldc, 510, 511
ldc_w, 512, 513
special methods, 23
MethodParameters attribute, 156
methods, 92
AnnotationDefault attribute, 153
attributes, 95
ClassFile structure, 73
Code attribute, 102
Deprecated attribute, 129
Exceptions attribute, 113
floating-point modes, 20
method type and method handle resolution, 350
MethodParameters attribute, 156
RuntimeInvisibleAnnotations attribute, 135
RuntimeInvisibleParameterAnnotations attribute, 137
RuntimeVisibleAnnotations attribute, 136
RuntimeVisibleParameterAnnotations attribute, 138
RuntimeVisibleTypeAnnotations attribute, 152
special methods, 22
structural constraints, 164, 164
method descriptors for method handles, 349
method type and method handle resolution, 349
method invocation and return instructions, 35
normal method invocation completion, 18
synchronization, 36
method resolution, 344
interface method resolution, 346
invokeinterface, 474
invokespecial, 477, 478, 479
invokestatic, 481, 482
invokevirtual, 484, 485, 487
invoking methods, 53
loading constraints, 337
method type and method handle resolution, 347
call site specifier resolution, 350, 351, 351
CONSTANT_MethodHandle_info structure, 88
initialization, 353
invokevirtual, 486, 486, 487
ldc, 510, 511
ldc_w, 512, 513
special methods, 23
MethodParameters attribute, 156
methods, 92
AnnotationDefault attribute, 153
attributes, 95
ClassFile structure, 73
Code attribute, 102
Deprecated attribute, 129
Exceptions attribute, 113
floating-point modes, 20
method type and method handle resolution, 350
MethodParameters attribute, 156
RuntimeInvisibleAnnotations attribute, 135
RuntimeInvisibleParameterAnnotations attribute, 137
RuntimeVisibleAnnotations attribute, 136
RuntimeVisibleParameterAnnotations attribute, 138
RuntimeVisibleTypeAnnotations attribute, 152
special methods, 22
structural constraints, 164, 164
method descriptors for method handles, 349
method type and method handle resolution, 349
method invocation and return instructions, 35
normal method invocation completion, 18
synchronization, 36
method resolution, 344
interface method resolution, 346
invokeinterface, 474
special methods, 22
synchronization, 36
Synthetic attribute, 118

mnemonic, 360
monitorenter, 303, 532
invokeinterface, 474
invokespecial, 478
invokestatic, 481
invokevirtual, 485
monitorexit, 534

monitorexit, 304, 534
areturn, 370
athrow, 374
dreturn, 402
freturn, 432
invokeinterface, 475
invokespecial, 479
invokestatic, 482
invokevirtual, 486
ireturn, 491
lreturn, 524
monitorenter, 532
return, 550

more control examples, 47
dcmp<op>, 390
fcmp<op>, 420

multianewarray, 305, 536

N

native method stacks, 14
native methods, 355
invokeinterface, 474
invokespecial, 479
invokestatic, 482
invokevirtual, 485

nested class access and property flags, 116
InnerClasses attribute, 116, 116

new, 306, 538
initialization, 352
StackMapTable attribute, 109

newarray, 307, 540
baload, 376
bastore, 377
boolean type, 10
multianewarray, 537

dop, 308, 542

normal method invocation completion, 18
   synchronization, 67

notation, 4

O

Object creation and manipulation, 34
   load and store instructions, 29

opcode mnemonics by opcode, 559

operand stack, 409, 410, 411, 408, 544

operand stack management instructions, 34

operand stacks, 17
   Code attribute, 103
   format of instruction descriptions, 361
   frames, 15
   load and store instructions, 29
   structural constraints, 163

operations on the operand stack, 59

organization of the specification, 3

overriding, 352
   invokevirtual, 484
   preparation, 340
   type checking abstract and native methods, 184
   verification of class files, 167

P

pc register, 12
INDEX

pop, 543
pop, pop2, 309
pop2, 544
predefined class file attributes (by class file version), 99
  attributes, 96
predefined class file attributes (by location), 100
  attributes, 96
  ClassFile structure, 74
  Code attribute, 105
  fields, 92
  methods, 95
predefined class file attributes (by section), 98
  attributes, 95
preparation, 340
  loading constraints, 337, 337
primitive types and values, 6
  multianewarray, 536
  new, 538
  newarray, 540
  preparation, 340
process of verification by type inference, 319
public design, private implementation, 37
  reserved opcodes, 358
putfield, 310, 545
putstatic, 311, 547
  initialization, 352

R

receiving arguments, 50
  invoking methods, 52
reference types and values, 11
  anewarray, 369
  control transfer instructions, 35
  field resolution, 344
  interface method resolution, 347, 347
  method resolution, 345, 345
  multianewarray, 536
  new, 538
  newarray, 540
  preparation, 340, 340, 340, 341, 341
representation of objects, 19
reserved opcodes, 358
  static constraints, 159
resolution, 341
  creation and loading, 334
  loading constraints, 337
ret, 549
  jsr, 499
  jsr_w, 500
  returnaddress type and values, 10
    wide, 557
return, 312, 550
  ret, 549
returnaddress type and values, 10
  primitive types and values, 6
run-time constant pool, 14, 329
  dynamic linking, 18
  frames, 15
  getfield, 436
  getstatic, 438
  invokedynamic, 468, 468, 471
  invokeinterface, 473
  invokespecial, 477
  invokestatic, 481
  invokevirtual, 484
  ldc, 510, 510, 510, 510
  ldc2_w, 514
  ldc_w, 512, 512, 512
  method type and method handle resolution, 348
  putfield, 545
  putstatic, 547
  resolution, 342
run-time constant pool, 329
run-time data areas, 11
RuntimeInvisibleAnnotations attribute, 135
RuntimeInvisibleParameterAnnotations attribute, 137
RuntimeInvisibleTypeAnnotations attribute, 152
RuntimeVisibleAnnotations attribute, 130
  annotations, 67
  element_value structure, 134
  RuntimeInvisibleAnnotations attribute, 136
  RuntimeInvisibleParameterAnnotations attribute, 139
  RuntimeVisibleParameterAnnotations attribute, 137
  RuntimeVisibleTypeAnnotations attribute, 141
RuntimeVisibleParameterAnnotations attribute, 136
RuntimeVisibleTypeAnnotations attribute, 139
  RuntimeInvisibleTypeAnnotations attribute, 153

S
saload, 313, 551
sastore, 314, 552
Signature attribute, 118
signatures, 119
  LocalVariableTypeTable attribute, 129
  Signature attribute, 118
sipush, 315, 553
SourceDebugExtension attribute, 124
SourceFile attribute, 123
special methods, 22
  ClassFile structure, 74
  Code attribute, 102
CONSTANT_Fieldref_info,
CONSTANT_Methodref_info, and
CONSTANT_InterfaceMethodref_info structures, 81
CONSTANT_MethodHandle_info structure, 88
CONSTANT_NameAndType_info structure, 85
ConstantValue attribute, 101
constraints on Java Virtual Machine Code, 159
initialization, 352
instance initialization methods and newly created objects, 324
invokedynamic, 469
invokeinterface, 473
invokespecial, 477, 480
invokestatic, 481
invokevirtual, 484, 484, 486
method area, 13
method invocation and return instructions, 35
method resolution, 344
methods, 92, 94, 94
new, 539
putfield, 545
putstatic, 547
static constraints, 161
structural constraints, 163
Synthetic attribute, 118
unqualified names, 75
working with class instances, 53
stack map frame representation, 178
accessors for Java Virtual Machine artifacts, 171
type checking load and store instructions, 194, 194
verification by type checking, 169
StackMapTable attribute, 106
INDEX

stack map frame representation, 179
verification by type checking, 168
startup, 332
initialization, 353
static constraints, 159
limitations of the Java Virtual Machine, 328
structural constraints, 163
limitations of the Java Virtual Machine, 328
structure of the Java Virtual Machine, 5
swap, 316, 554
operand stacks, 17
synchronization, 36, 66
areturn, 370, 370
athrow, 374, 375
dreturn, 402, 402
freturn, 432, 432
ireturn, 491, 491
lreturn, 524, 524
monitorenter, 532
monitorexit, 534, 535, 534
return, 550, 550
synchronization, 36, 66
Synthetic attribute, 118
methods, 94

T

tableswitch, 317, 555
instruction set summary, 25
target_info union, 144
RuntimeVisibleTypeAnnotations attribute, 141
throwing and handling Exceptions, 60
Exceptions, 25
more control examples, 47
throwing Exceptions, 36
type checking abstract and native methods, 184
verification by type checking, 169
type checking for protected members, 196
access control, 352
getfield, 251
invokevirtual, 275
verification by type checking, 169
type checking instructions, 199
stack map frame representation, 182
verification by type checking, 169
type checking load and store instructions, 194
stack map frame representation, 180
verification by type checking, 169
type checking methods with Code, 187
stack map frame representation, 179
StackMapTable attribute, 107
verification by type checking, 169
type conversion instructions, 32
type support in the Java Virtual Machine
instruction set, 28
types and the Java Virtual Machine, 26, 26
use of constants, local variables, and control constructs, 43
type_path structure, 148
RuntimeVisibleTypeAnnotations attribute, 141
types and the Java Virtual Machine, 26
arithmetic instructions, 30
control transfer instructions, 35
data types, 6
dup, 406
dup_x1, 407
load and store instructions, 30
operand stack, 409, 409, 410, 410, 411, 411, 411, 411, 408, 408, 544, 544
pop, 543
stack map frame representation, 182
swap, 554
type conversion instructions, 32
unqualified names, 75
    binary class and interface names, 74
    CONSTANT_NameAndType_info structure, 85
    fields, 91
    LocalVariableTable attribute, 127
    LocalVariableTypeTable attribute, 129
    methods, 94
    signatures, 120
use of constants, local variables, and control constructs, 40
    accessing the run-time constant pool, 46
    more control examples, 47
user-defined class loaders, 335
    creation and loading, 333
    loading constraints, 337

values of types long and double, 323
verification, 340
    creation and loading, 334
verification by type checking, 167
    StackMapTable attribute, 106, 106
verification by type inference, 319
    verification by type checking, 167
verification of class files, 166
    assumptions: the meaning of "must", 357
    operand stacks, 17
    verification, 340
verification type system, 173
    accessors for Java Virtual Machine artifacts, 171
    verification by type checking, 169
Virtual Machine errors, 358
    Exceptions, 24
INDEX

W

wide, 318, 557
  aload, 367
  astore, 372
  dload, 395
  dstore, 403
  fload, 425
  fstore, 433
  iinc, 461
  iload, 462
  istore, 494
  lload, 516
  lstore, 527
  ret, 549

working with class instances, 53
  accessing the run-time constant pool, 46
  invoking methods, 52