

Index

A

- Abandoning metadata, 124–130
- Abstract classes, 61
- Abstract methods, 61
- AccessControlException class, 78, 157, 177
- AccessibleObject class, 77, 215
- Account object, 263
- Accounts class, 263
- Activatable class, 150
- AddRef method, 293
- AllocHandle, 231
- AllocObject class, 216–217
- AnnotateClass method, 147
- AnnotatedOutputStream class, 148
- Annotating components, XVI
- Apartments, 297
- App library, 200
- AppClassLoader, 45
- Applets, 25–26
- Application class, 22
- Application class loader, 37
- Applications
 - assembling, 11–14
 - assembling with components, 3
 - bypassing or ignoring settings, 14
 - configuration settings, 12
 - data resources, 12
 - dynamic, 12
 - growing and evolving, 4–5
 - instrumenting, 44–45
 - linking correctly, 13–14
 - looking for resources, 13
 - modifying, 12
 - sharing components, 11
 - splitting into components, 11–12

- ArrayList, 142

Arrays

- incrementing elements, 234
 - Java definition of, 233
 - managing, 233–239
 - pinning, 236
 - read-only traversal, 236–237
 - storing, 233, 235

Assembling

- applications, 11–14
 - components, 12
- ATL (Active Template Library), 307
- Attribute class, 102, 181
- attribute_info structure, 99
- Auctioneer interface, 244
- AuctioneerImpl class, 245
- Audit class, 84
- Automating tasks, XVI

B

- Base interface loaded implicitly, 34
- Base class fields and serialization, 109
- Binary boundaries between components, XVI
- Binary classes, 58, 66, 252
 - data structures, 64–65
 - introspecting against, 57
 - metadata, 63–64
- Binary compatibility, 58–62
- Bind modes, 251, 253–255
- Bind times, 244, 246–247, 251, 253–255
 - changing priorities, 249
 - separating specification from, 247–249
- blowUpWorld method, 59
- BMovie class, 58–61

Bootclasspath, 41–43
 Bootstrap class loader, 36, 41–43, 48
 Boss class, 141, 146

C

C++

calling Java from, 208–217
 exception handling, 218–219
 method declarations, 196
 parameters with multiple levels of indirection, 196

Call stack logging, 46

Callee class, 189

Caller class, 189

Caller class loader, 21

CallVoidMethod method, 213, 219

CallVoidMethodA method, 221

Capability, 16

CauseConfusion methods, 199

CComBSTR class, 307

Chars API, 239

checkPermission method, 299

Child loader and classes, 49

Circular reference problems during resolution, 141–142

Class class, 21, 67–68, 70, 123

Class loader architecture, 5, 11

assembling applications, 14

capability, 16

configurability, 16

explicit class loading, 19

extensibility, 15–16

goals, 14–17

handling name and version conflicts, 16–17

security, 17

transparency, 15

Class loader delegation model, 5

Class loaders, 11, 14, 55, 281

applet, 25

caller's, 21

choosing, 174–175

configuration options, 36

consistency rule, 23–24

context, 5

crossing boundaries, 72

custom, 5, 153–190

delegation, 24–25, 27

effective use of, 14

exotic designs, 15–16

explicit use of, 29

getting past security to, 175–177

hierarchy, 52

holding cache of classes, 35–36

hot deployment, 29–34

installing all classes under, 52

instantiating, 53, 177

interaction with, 36

inversion, 49–55

JNI (Java Native Interface), 202–204

keeping in memory, 36

limited sharing of classes, 25

loading arbitrary resources, 22–23

meeting common needs, 16

memory, 35

multiple versions of native code, 203

overriding findLibrary method, 202

parent, 24

role, 157–159

rules, 23–29

security hole, 177

since SDK 1.2, 160–162

source-aware, 23

system, 25

transforming, 162–167

visibility rule, 25–26

Class loading, 55

capable, 16

debugging, 43–49

dynamic, 28–29, 251

easily configurable, 16

errors, 28

explicit, 18–23

implicit, 19–23, 24, 29

name conflicts, 16–17

reading attributes during, 183–188

supporting flexible binding modes, 251

version incompatibilities, 16–17

Class object, 18, 35

ClassCastException class, 144

Class-controlled object replacement, 137–139

Classes

already-loaded, 23–24

child loader, 49

compatibility, 66

compatible and incompatible changes, 117–118

- configuring loading, 36–43
- core, 25
- customizing descriptors, 124
- deploying multiple versions simultaneously, 17
- diagnosing loading problems, 44
- discovering fields, 68
- dynamic proxies, 57
- dynamically discovering characteristics, 66
- dynamically loading, 19
- extensible loading, 15–16
- extensions path, 39–41
- externalizable, 125–127
- finding code, 147–150
- garbage collecting, 36–37
- in class loader hierarchy, 50
- incremental changes, 30
- invisible, 60
- loaded at different times, 27
- loaded by bootstrap class loader, 41
- loaded from classpath, 41
- loaded from different locations, 27
- loading, 37–39
- loading resources for, 5
- loading wrong version, 28
- locating, 5, 38–39
- matching streams with, 113–118
- memory, 35
- naming rule, 16–17
- nonexistent, 60
- overwriting metadata, 122–123
- package names, 16
- parent loader, 49
- permissions, 155–157
- postponing linking, 21
- preventing creation of multiple instances, 28
- public methods, 69
- referencing nonexistent or invalid entry, 60
- referencing other classes, 49
- referring to another class, 19
- relationship between old and new versions, 34
- sealed JAR files, 178
- security, 25, 41
- separating serialized form from in-memory representation, 138
- serializable, 106–109
- serving old version, 17
- tracing loading, 45
- type information, 67
- unintended references, 35
- unloading, 35–36
- verified, 41
- version-reconciliation, 5
- versions, 14
- visibility, 23
- ClassEx file, 102
- ClassFormatError, 166
- Class-level metadata, 94
- Class-level object replacement, 137–140
- Class-level resolution, 140
- ClassLoader class, 17–18, 20–21, 95, 147, 153, 158–159, 175–176
 - checking another path, 202
 - convenience constructor, 25
 - modifications, 160–161
- ClassLoader object, 35
- ClassNotFoundError class, 43
- ClassNotFoundException class, 18, 19, 113
- ClassNotter class, 165–167
- Class-oriented design, 3
- Classpath class loader, 36, 37–39
- classpath command-line switch, 37–38
- CLASSPATH environment variable, 37, 38
- Client applications, 2
- Client class, 61–62
- Client code, 30
- Clients, 33
 - installing dynamic proxies, 90–91
 - reference of base class or interface type, 34
 - reference of implementation type, 34
 - state of original object available to server factory, 34
- Close method, 133, 232
- close method, 302
- Code
 - bind times, 244, 246–247
 - choosing specification language, 249
 - commonalities, 243–244
 - controlled sharing, 28
 - domain analysis, 243
 - ease of generating, 251
 - generation, 5–6
 - generation boosting performance, 252
 - generation language, 270–271
 - high in class loader hierarchy, 50
 - integrity endangered, 24
 - in-the-large generation, 252–253

- Code (con't.)
 - in-the-small generation, 253
 - levels of commitment to generation, 252–253
 - loading, 36
 - reasons for generating, 243–250
 - reuse requiring multiple use, 249
 - RMI (Remote Method Invocation) generation, 255–257
 - separating specification from bind time, 247–249
 - sharing, 5
 - trusted, 175
- Code attribute, 101
- CodeBase attribute, 155
- Collections
 - generic, 267
 - strongly-typed, 267–271
- COM (Component Object Model), 285–286, 290–291
 - apartments, 297
 - context model, 297
 - error handling, 295–296
 - GUID (Globally Unique ID), 291
 - local loader, 291–292
 - object lifecycle, 293–294
 - platform impedance, 290
 - security, 298–299
 - stubs, 297
 - thread affinity, 296–298
 - type discovery, 294–295
 - type information, 292–293
- com.develop package, 269
- com.develop.benchmark package, 92
- com.develop.hello package, 95
- com.develop.jawin.tools package, 311
- Com.develop.version.audit property, 188
- COMException class, 295–296, 310
- Commonalities, 243
- Compilers and custom virtual machine attributes, 101
- Compile-time binding, 247
- Component approach, 3, 6–9
- Component-oriented design, 3
- Components, XV, 3
 - annotating, XVI
 - binary boundaries between, XVI
 - bloated, 5–6
 - class loaders, 281
 - concurrency protection mechanisms, 297
 - control over fitting together, 12–13
 - deployment, 12–13
 - EJB (Enterprise JavaBeans), 264–265
 - extending, 6
 - loading configuration information, 5
 - locating, 12
 - mobile, 105
 - multiple versions, 12
 - non-Java, XVI
 - platforms, 12
 - rogue, 12
 - sharing, 11
 - splitting applications into, 11–12
 - thread-affine, 296–297
 - type information, 281
 - unable to find, 12
- COMSharedStubBuilder class, 311
- COMSharedStubDriver class, 311, 314
- Configurability, 16
- Connect method, 170
- Consistency rule, 23
 - breaking, 24
 - forbidding unloading and reloading from single loader, 31
 - implicit class loading, 29
- Constant pool, 63–64
- ConstantValue attribute, 101
- Constructor class, 72–73, 77
- Constructors
 - serialization, 110–111
 - throwing exception, 82
- Contact interface, 1–2
- Contact management domain, 6–9
- Contact management systems, 1
- ContactFinder interface, 1–2
- Contacts, 2
- Containers
 - code generation, 266–267
 - EJB (Enterprise JavaBeans), 264–265
- Container-transaction element, 263
- Context class loader, 5, 49–55
- Context loader, 53
- Context model, 297
- contextLocalize method, 298
- CORBA, 197
- Core API
 - classes, 26, 46
 - instrumenting, 46–49

- loading packages, 41
- protecting integrity, 26
- Core classes, 25
 - recompiling subset, 42
 - replacing, 42, 49
- Core packages, loading, 36
- cp command-line switch, 37–38
- createClassLoader method, 158
- createPoint method, 31
- Critical APIs, 237–238, 239
- Critical array API, 239
- Critical region, 238
- Critical string API, 239
- Cross-platform communication, 6
- CrunchArray method, 292
- current directory (.), 19
- custom attributes, 5
- Custom class loaders, 5, 153–190
 - adding configuration options, 16
 - pre-Java 2, 159
- Custom metadata, 98–102
 - debugging support, 188–189
 - reading, 177–189
 - reading attributes during class loading, 183–188
 - serializable classes as attributes, 179–182
 - version attributes classes, 178–179
- Custom serialization code, 271–276
- Custom type formats, 300
- Customizing class descriptors, 124

D

- Data
 - alternate representations, XVI
 - structures and binary classes, 64–65
- Data(N) classes, 149
- DataInput interface, 124
- DataOutput interface, 124, 130
- DCOM, 197
- Debugging
 - custom metadata support, 188–190
 - replacing core classes, 42
- Debugging class loading
 - instrumenting Core API, 46–49
 - tracing class loading, 45
 - verbose:class flag, 45
- Default field data, 119
- DefaultReadObject method, 119, 120
- DefaultWriteObject method, 124–125, 128, 129
- Defective PointImpl listing, 30
- defineClass method, 158–162, 165, 187, 216
- definePackage method, 95
- delegate method, 88–89
- Delegation rule, 24–25
- Delegation, 25
 - of class loaders, 24–25, 27
 - generic, 84–85
 - instead of implementation inheritance, 83
 - as namespaces, 27
 - object-oriented design, 5
- delete method, 156
- DeleteGlobalRef method, 225–226
- DeleteLocalRef method, 226–227, 229
- DemoLogging class, 85–86
- Deployer, 13–14
- Deployment
 - discouraging analysis of, 4
 - simplify, 23
- Deployment descriptor, 263–265
- Deployment-time binding, 247
- deposit method, 262, 263
- Deserialize-on-demand, 149–150
- Deserializing objects, 113
- Design-time binding, 247
- Developer's machine, 4
- Development environments, XVI
- DiplomaticContact interface, 3
- Directories and extensions path, 40
- Djava.security.manager flag, 156
- DLLs (Dynamic Link Libraries)
 - error reporting, 302
 - loader architecture, 299–300
 - object lifecycle, 301–302
 - security, 303
 - thread affinity, 302
 - type discovery, 302
 - type information, 300–301
- doGet method, 257–258
- Domain analysis, 5, 243, 250
- Dynamic applications, 12
- Dynamic bind mode, 254
- Dynamic bindings, 251
- Dynamic class loading, 28–29, 251
- Dynamic proxies, XVI, 57, 251, 254, 256–257, 565
 - advantages, 91
 - generating on-the-fly, 91

- Dynamic proxies (con't.)
 - generic delegation, 84–85
 - implementing forwarding handler, 86–87
 - implementing `InvocationHandler` interface, 85–86
 - installed by client or server, 90–91
 - `InvocationHandler` interface as generic service, 87–89
 - method call forwarding, 87
 - reflective invocation, 83

E

- Early binding, 252
- Earth singleton, 138
- `Echo1` class, 65
- Eiffel programming language, 16
- EJB (Enterprise JavaBeans), XVII, 254
 - alternate implementations, 265–267
 - bean object, 261
 - containers, 264–265
 - deployment descriptor, 261, 263–265
 - entity beans, 260
 - generic classes, 265
 - home interface, 260, 261
 - remote interface, 260, 261
 - serialization, 105
 - session beans, 260
 - skeletons, 264
 - stubs, 264
- Element array API, 235–236, 239
- Enable methods, 134
- Encapsulation, XV
- `EnsureLocalCapacity` API, 230
- `EnsureLocalCapacity` method, 228–229
- Enterprise applications, 282
- Entity beans, 260
- Environment variables, 201–202
- `equals` method, 86
- Error 82, 205
- Error handling
 - COM (Component Object Model), 295–296
 - JNI (Java Native Interface), 217–223
- Errors and class loading, 28
- Exception class, 295
- `ExceptionCheck` method, 219–221

- `ExceptionClear` method, 220, 223
- `ExceptionOccurred` method, 219–221
- Exceptions
 - declared and binary compatibility, 61–62
 - handling with `InvocationHandler` interface, 89
 - reflective invocation, 81–83
- Exceptions attribute, 101
- Explicit class loading, 18–23
- Explicit loading APIs, 21
- Explicitly
 - loading from URL, 18
 - managing serializable fields, 119–124
- `ExtClassLoader`, 45
- Extensibility, 15–16
- Extensions, loading, 36
- Extensions class loader, 36, 48
- Extensions loader, 39–40, 45
- Extensions path, 39–41
- Externalizable class, 125–127
- Externalizer class, 271, 273

F

- Field class, 72–73, 77
- Fields, 60
- File URL, 18
- `FilePermission` permission, 155, 157
- final class, 61
- final fields, 79–81
- `finalize` method, 232, 293, 294, 301, 302
- finally block, 232
- `FindBestMatch` method, 229–230
- `FindClass` method, 102, 160, 165, 183, 216, 219, 226
- `FindContact` interface, 1
- Finding class code, 147–150
- `findLibrary` method, 202
- `FindResources` method, 183
- Flexible binding modes, 251
- `Foo` class, 43
- `ForgetToLoadLibrary` class, 205
- `FormatMessage` function, 295, 302
- `forName` method, 21, 216
- Forwarding handler, implementing, 86–87
- `FuncPtr` function, 310
- Function pointers, 72

G

- Garbage collector, 35
 - classes, 36–37
 - encouraging, 132–133
 - interacting with, 224–231
 - local references, 224
- gc method, 204
- Generating stubs, 311–316
- GeneratorBase class, 271
- Generators, 247–248, 249
- Generic collections, 267
- Generic delegation and dynamic proxies, 84–85
- Generic interceptors, 83
- Generic services, XVI, 87–89
- Generic skeletons, 256
- Generic stubs, 308
- get method, 72
- GetAnswer method, 194, 197
- getBadDateValue method, 90
- getClass method, 35
- getClassLoader method, 19, 35, 41
- GetContentType method, 170
- GetDate method, 170
- getDeclared method, 72
- getDeclaredField method, 70–71, 210
- getDeclaredFields method, 69
- getDeclaredXXX accessor method, 68–70
- GetErrorInfo function, 295
- getExpiration method, 170
- GetField class, 121, 275
- getField method, 70–71, 123–124, 210
- getFieldID method, 210
- getFields method, 69
- getGoodDateValue method, 90
- getInputStream method, 170
- GetIntArrayElements method, 236, 237
- GetIntArrayRegion method, 235
- getInterfaces method, 67
- GetLastError function, 302
- getMethod method, 73, 75
- GetMethodID method, 213
- getModifiers method, 69–70
- getOutputStream method, 170
- getParent method, 24, 44
- GetPrimitiveArrayCritical method, 238
- GetProcAddress function, 299, 310
- getProtectionDomain method, 162
- GetRegion method, 240
- getResource method, 22, 165
- getResourceAsStream method, 22
- getResources method, 22
- getSuperclass method, 67
- getSystemClassLoader method, 25
- getTargetException method, 82
- GetTokenInfo Win32 API, 300–301, 302
- getURLBase method, 165
- getVersionInfo method, 187
- getXXX accessor method, 68–70
- Global references, 224–225, 226
- GP (generative programming), 243
 - binary classes, 252
 - bind modes, 253–255
 - bind times, 253–255
 - class loading supporting flexible binding modes, 251
 - code generation boosting performance, 252
 - code generation language, 270–271
 - custom serialization code, 271–276
 - domain analysis, 250
 - ease of generating Java source code, 251
 - EJB (Enterprise JavaBeans) code generation, 260–267
 - generating stubs, 311–316
 - inputs and outputs, 254–255
 - JSP (Java Server Pages) code generation, 257–260
 - levels of commitment to code generation, 252–253
 - marshalling architecture, 303–311
 - reasons for generating code, 243–250
 - strongly-typed collections, 267–271
 - target language, 270–271
 - type information as free specification document, 250–251
- GUID (Globally Unique ID), 291

H

- Handler class, 169, 174
- hashCode method, 86
- HashMap class, 24
- Header files, 300
- Hello.jsp servlet, 257, 258–259
- Helper components, 253

Hot deployment, 29–34
 HRESULT type, 295
 Humanoid class, 106–107, 113

I

IDL (Interface Definitions Language), 292–293
 IDL files, 300
 IIDs, 294
 IllegalAccessException class, 78, 81
 IllegalArgumentException class, 81, 226
 Implementation class, 34
 Implicit class loading, 19–23, 24, 29
 Implicit class references, 20
 InadequateNationalInfrastructure class, 61
 Incompatible changes, 62
 IncompatibleClassChangeError class, 60
 InfoSet, 282
 Inheritance, XV
 delegation instead of, 83
 extensible design with, 2
 modeling variabilities, 245–246
 multiple, 83
 object-oriented design, 5
 InnerClass attribute, 101
 Installed optional packages
 completely trusted by virtual machine, 40–41
 loading, 39–41
 security, 40–41
 InstantiationException class, 81
 Instruction strings, 309–311
 Instrumenting
 applications, 44–45
 core API, 46–49
 int return type, 197
 Integer class, 75
 Integer.class class, 75
 Integer.TYPE class, 75
 Interceptors, 83
 Interface stubs, generating, 314–316
 Interfaces, 72, 294
 Internal name collisions, 17
 Intrinsic functions, 307
 Introspecting against binary classes, 57
 Introspection, 57
 InvalidClassException class, 116, 118, 123

Invariants, 16
 Inversion
 avoiding problems, 50
 context class loader, 49–55
 InvocationHandler interface, 87, 254
 as generic service, 87–89
 handling exceptions, 89
 implementing, 85–86
 InvocationHandler.invoke method, 87
 InvocationTargetException class, 82, 89
 invoke method, 86, 87, 265
 isCompatibleWith API, 96
 ItemImpl class, 246
 itf1 variable, 295
 itf5 variable, 295
 itfs array, 85
 ITypeInfo interface, 292
 ITypeLib interface, 292
 IUnknown interface, 293

J

J2EE implementation, 53
 -jar command-line switch, 38
 JAR files
 adding version information, 95–96
 extensions loader, 40
 sealing packages, 97
 Java
 accessing code libraries not written in, 192
 accessing platform-specific resources, 192
 calling from C++, 208–217
 code and dotted notation, 63–64
 converting to XML, XVI
 direct access to system memory, 191–192
 hand-tuned, peak performance code, 192
 IDE wizards, 72
 loaders, XV–XVI
 memory protections, 193
 multiple inheritance, 196
 passing parameters by value, 196
 reasons for generating code, 250–253
 serialization, 5
 tasks ill-suited for, 191–192
 type information, XVI
 WORA (write-once, run-anywhere), 191

- Java 2 SDK, XIX, 118
- Java launcher, 230
- Java Reflection API, 57, 66–72, 101–102
- Java Virtual Machine Specification, 103
- JavaBeans, XVII, 66–67, 105
- javah utility, 199, 206–207
- java.io.DataOutputStream stream, 114
- Java.io.Externalizable class, 125
- java.io.File.pathSeparatorChar constant, 38
- java.io.NotSerializableException class, 108
- java.io.ObjectOutputStream wrapper stream, 108
- java.io.Serializable interface, 106
- java.lang package, 75
- java.lang.Class class, 67, 102
- java.lang.ClassLoader class, 207–208, 216
- java.lang.IllegalArgumentException class, 226
- java.lang.Package class, 94
- java.lang.reflect package, 67
- java.lang.reflect.Constructor class, 216
- java.lang.reflect.Field class, 68
- java.lang.reflect.Method class, 71
- java.lang.reflect.Modifiers class, 70
- java.lang.String class, 64
- java.library.path system property, 201–202
- java.net.URL class, 168
- java.net.URLConnection class, 169
- java.net.URLStreamHandler class, 169
- javap utility, 65
- java.policy file, 41
- java.protocol.handler.pkgs property, 173
- java.rmi.MarshalledObject class, 150
- java.rmi.RemoteException class, 289
- java.security.AccessControlContext class, 158
- java.security.CodeSource class, 155
- java.security.Permission class, 155
- java.security.Policy class, 155
- java.security.ProtectionDomain class, 158
- java.security.SecureClassLoader class, 161–162
- java.util package, 267
- java.util.Date class, 87–88
- java.util.Stack class, 269
- JavaVM pointer, 218
- Jawin, 285
 - COM threading rules, 298
 - instruction strings, 310–311
 - translucent stubs, 286–289
 - Win32 stubs, 301
- JCFE (Java Class File Editor), 102, 181
- JconstructorID type, 216
- JCP (Java Community Process), 282
- JINI, XVII, 105
- jint return type, 197
- JNDI (Java Naming and Directory Interface), 262
- JNI (Java Native Interface), 6, 79, 191, 285–286
 - APIs for field access, 210
 - APIs for method access, 213
 - array region APIs, 234
 - calling C-style API, 212
 - calling Java from C++, 208–217
 - class loader and construction functions, 216
 - class loaders, 202–204
 - cross-platform standard, 193
 - error handling, 217–223
 - finding and loading native code, 194–208
 - handling C++ exceptions, 218–219
 - handling Java exceptions from native code, 219–222
 - improving performance by precalculating jfield-IDs, 211–212
 - jobject references, 224
 - layer of indirection, 195
 - loading native libraries, 199–202
 - minimalist mapping from Java types to C++ types, 197
 - minimizing round trips, 211–213
 - name mappings, 195
 - name-mangling scheme, 198
 - overloaded names, 198–199
 - passing arrays, 233
 - performance comparisons, 214
 - reflection, 215
 - reflective invocation and, 214–217
 - resource management, 223–240
 - stubs, 287
 - throwing Java exceptions from native code, 222–223
 - troubleshooting native loading, 207–208
 - type mappings, 195–197
 - virtual machine, 192
- JNI_ABORT flag, 236–237
- JNI_COMMIT flag, 236–237
- JNICALL macro, 197
- JNIEnv type, 197, 208, 218, 222–223
- JNIEnvUtil class, 221, 222

JNIEXPORT macro, 197
 JNI_OnLoad method, 212, 220–221, 222, 226
 JNI_OnUnload method, 226
 JNI_VERSION_1_2 method, 212
 jobject references, 224
 jobject type, 197, 208, 210
 JSP (Java Server Pages), XVII, 251, 253, 254
 code generation, 257–260
 generator language for Java, 270–271
 strongly-typed stack classes, 268–269
 _jspService method, 258

L

Late binding, 252
 Launcher and command line processing, 39
 LaunchVehicle interface, 98–99
 LD_LIBRARY_PATH environment variable, 201
 LICENSE file, 49
 LineNumberTable attribute, 101
 Linking
 application correctly, 13–14
 methods at runtime, 59
 native libraries, 199–202
 with full paths, 200
 ListBaseTypes class, 67
 ListByLastName console client, 2
 ListMostFields class, 69–70
 LittleEndianOutputStream, 310
 load method, 200–202
 loadClass method, 21, 47, 159–161
 LoaderDemo class, 19, 27–28
 Loading native libraries, 199–202
 LoadLibrary Win32 function, 310
 loadLibrary method, 200–202, 205–206, 299
 LoadMe class, 19, 20, 27–28
 LoadMeAlso class, 20
 LoadMeBase class, 20
 LoadMeThree class, 20
 LoadMeToo class, 20
 LoadNonExistentLibrary class, 205
 LoadTheWrongLibrary class, 206
 Local references, 224, 226, 227
 LocalVariableTable attribute, 101
 Locating components, 12
 logConstructor method, 46
 Logging call stacks, 46
 LoggingHandler class, 85–86

M

-m switch, 95
 MadScientist class, 58–61
 Main class, 49
 Mars singleton, 138
 MarshalledObject class, 150
 Marshalling architecture, 6, 286–287
 generic stubs, 308
 instruction strings, 309–311
 shared stubs, 303–307
 Message object, 149
 MessageBoxW function, 300, 309–311
 Metadata, 57, 66
 abandoning, 124–130
 accessing from packages, 96
 binary classes, 63–64
 binary compatibility, 58–62
 class-level, 94
 correct units for numeric arguments, 98
 custom, 98–102
 externalizable classes, 125–127
 overwriting, 122–123
 reading custom, 177–189
 serialization, 105–106
 setting for packages, 95–96
 skipping, 127
 supplementing, 197
 tables of allowable state transitions, 98
 version information, 94
 Method class, 71, 72–73, 77, 256, 257
 Methods
 call forwarding, 87
 converting to network messages, XVI
 invalid arguments, 81
 invisible, 60
 linking at runtime, 59
 nonexistent, 60
 overloaded, 198
 reflection, 71–72
 small-grained, 211
 throwing exception, 82
 virtual and non-virtual invocation, 215
 midl compiler, 292
 Mobile components, 105
 Modifying applications, 12
 Money class, 263
 Moving corrected PointImpl, 33

MTA (multi-threaded apartment), 297
Multiple inheritance, 83, 196

N

Name conflicts, 16–17
Name mappings, 195
Namespaces, delegations as, 27–28
Naming rule, 16–17
Native code
 bypassing language protection modifiers, 193
 calling Java constructor, 216
 dangers of, 193–194
 failures, 217–218
 finding and loading, 194–208
 global references, 224–225
 handling Java exceptions, 219–222
 loading, 204
 reflective access, 209
 throwing Java exceptions, 222–223
 type-safety, 193
native keyword, 194
Native libraries
 common errors loading, 205–207
 loading, 199–202
 loading old version, 204
 troubleshooting loading, 207
 visible across all class loaders, 203
Native methods
 instantiating Person class, 79
 loading, 200
 populating fields, 81
 shared stubs implementation, 305–306
Native resources, managing, 231–232
Native singleton implementation, 29
nativeMove method, 209–212
NativePoint class, 208–209
neverLoaded method, 206
NewGlobalRef method, 224–226
newInstance method, 176, 177, 216
NewObject method, 216
newProxyInstance method, 85
Non-Java components, XVI
NoOpResourceTransformer class, 165
NoSuchFieldException class, 71
NoSuchMethodError class, 60
not handler, 174
not protocol and protocol handlers, 169–170

Not referencing type dynamically replaced, 33
not stream handler, 168
NotConnection class, 174
NotInputStream class, 170–171, 174
NotSerializableException class, 131, 135
NotURLReader class, 172–173

O

obj reference, 20
Object approach, 3
Object class, 86
Object graphs, 130
 controlling ordering, 145–147
 custom serialization hooks, 145
 encouraging garbage collector, 132–133
 preserving identity, 131–132
 pruning, 131
 serializing more data than wanted, 131
Object replacement
 class-controlled, 137–139
 ordering rules, 139–144
 stream-controlled, 134
ObjectArgs array, 308
ObjectInputStream class, 112, 115, 118, 124, 134, 145
 defaultReadObject method, 112
 GetField method, 119, 120–121
ObjectInputValidation class, 145, 146–147
Object-oriented approach, questions raised by, 3
Object-oriented design, 3, 5
Object-oriented languages modeling problem domain, 3
ObjectOutputStream class, 108, 133, 134, 271
 GetField method, 119
 PutField method, 119
Objects, XV, 3
 deserializing, 113
 dynamically instantiating, 66
 explicitly destroying, 35
 preserving identity, 131–132
 serializing state of, XVI
ObjectStreamClass class, 123, 271
ObjectStreamField class, 123
OpenConnection method, 169
Optimization, 20
OSDeallocResource method, 231
out variable, 258

OutOfMemoryError, 132, 228
 Overloaded methods, 198
 Overloaded names, 198–199
 Overriding
 class metadata, 122–123
 default SUID, 115–116

P

Package class, 96
 Packages
 accessing metadata, 96
 names, 16
 reflection, 5, 94–98, 178–180
 sealing, 97
 setting metadata, 95–96
 sharing, 39
 versioning mechanism weaknesses, 97–98
 page import directive, 258
 Parent class loader, 24
 Parent loader, 49
 ParseURL method, 174
 PATH environment variable, 201
 Per-interface stubs, 288
 Permission class, 299
 Permissions, 155–157
 Person class, 79, 106–107, 113–116
 age field, 118
 all-fields constructor, 79–80
 externalizable version, 126–127
 fullName field, 119
 PhotonTorpedo class, 131
 Pinning, 236
 Planet class, 138–139
 Platforms and components, 12
 Platform-specific native mapping of Java int type, 195
 Point interface listing, 30
 PointClient class, 30–31, 33
 PointClient listing, 32
 PointImpl class, 30–31, 32
 PointServer class, 31
 Policy class, 155
 Policy file, 155–156
 Polymorphism, XV
 PopLocalFrame method, 229
 Pre-Java 2 custom class loaders, 159

Primitive types
 language differences, 195
 reading and writing, 124–125
 reflection, 75
 reflective invocation, 75
 signatures, 75
 wrapping, 74–76
 Primordial class loader, 41
 Principle of least surprise, 15
 private keyword, 60–61, 76
 PrivilegedAction, 176
 Processes and SecurityManager class, 77
 ProgID, 291
 protected keyword, 76
 ProtectionDomain, 166, 175
 Protocol handlers, 168
 choosing, 174–175
 implementing, 169–171
 installing custom, 171–174
 not protocol, 169–170
 Protocolhandler class, 154
 Proxies, dynamic. *See* dynamic proxies
 public keyword, 76
 PushLocalFrame method, 229, 230
 put method, 121
 PutField class, 121, 123–125, 128, 129, 275

Q

QueryInterface method, 294

R

readExternal method, 126, 271
 ReadFields method, 119, 120
 ReadInstance class, 108–109, 110, 113, 136, 137
 readInt method, 125
 ReadObject method, 119, 120, 121, 125, 128, 146
 readObject method, 111–112, 276
 Read-only traversal, 236–237
 ReadResolve method, 144
 Reference frames, 229–230
 Reference type versus referenced class, 20
 Referenced classes
 in class loader hierarchy, 50
 loaded implicitly listing, 19
 versus reference type, 20

- Reflection, 5, 6, 57
 - discovering fields of classes, 68
 - dynamically discovering class characteristics, 66
 - dynamically instantiating objects, 66
 - instantiating Person class, 79
 - invocation, 92
 - JavaBeans, 66–67
 - JNI (Java Native Interface), 215
 - limiting, 76
 - methods, 71–72
 - packages, 94–98
 - performance, 92–94
 - primitive types, 75
 - reflective launcher, 73–74
 - serialization, 66–67
 - serializing instances to stream, 66
 - Reflective invocation
 - bypassing language access rules, 76–81
 - dynamic proxies, 83
 - exceptions, 81–83
 - JNI and, 214–217
 - primitive types, 75
 - Reflective launcher, 73–74
 - Reflective modification of final fields, 79–81
 - RegCloseKey function, 301, 302
 - Region API, 234, 239
 - registerValidation method, 145, 146
 - Registry and ProgID, 291
 - RegOpenKey function, 301
 - Release method, 293, 294
 - ReleaseIntArrayElements method, 236
 - ReleasePrimitiveArrayCritical method, 238
 - reloadImpl method, 31
 - Remote marker interface, 255
 - RemoteException class, 289
 - ReplaceObject method, 135
 - Replacer class, 136, 137
 - Replacing classes at runtime, 24
 - Reporter class, 77–78
 - RequiredVersions class, 187
 - reset method, 132–133
 - resolveClass method, 147, 159
 - resolveObject method, 135, 139, 145
 - Resolver class, 136
 - ResolvingInputStream class, 148
 - Resource management
 - arrays, 233–239
 - interacting with garbage collector, 224–231
 - native resources, 231–232
 - strings, 239–240
 - Resources
 - default rules for locating, 14
 - enumeration of URLs, 22
 - first matching URL, 22
 - loading, 22–23
 - loading for classes, 5
 - looking for, 13
 - multiple locations for, 13
 - multiple versions coexisting, 11
 - thread-affine, 296
 - ResourceTransformer class, 162
 - RMI (Remote Method Invocation), XVII
 - annotation, 148–150
 - code generation, 255–257
 - java.rmi.MarshalledObject class, 150
 - serialization, 105, 148
 - skeletons, 255
 - stubs, 255
 - translucent stubs, 289
 - rmic (RMI stub compiler), 253–254, 255–256
 - Rocket class, 61–62
 - RocketShip class, 15
 - Rogue components, 12
 - rtd.jar file, 42
 - rt.jar file, 41, 46
 - run method, 73
 - runFinalization method, 204
 - RunFromURL class, 73–74
 - Runtime
 - assembling application, 13
 - binding, 247
 - replacing classes at, 24
 - type errors, 70–71
 - type-safe, 267
 - RuntimeException class, 82, 89
- ## S
- SafeBoss class, 146
 - SAPI.jar file, 51
 - with explicit class loading, 52–53
 - using thread context class loader, 54
 - Save method, 316
 - Schemas, 282
 - Sealing packages, 97

- SecureClassLoader class, 153–154, 161–162
- Security, XVII, 155–157
 - class loader architecture, 17
 - classes, 25
 - COM (Component Object Model), 298–299
 - DLLs (Dynamic Link Libraries), 303
 - extensions exempt from, 40
 - getting to class loader from, 175–177
 - installed optional packages, 40–41
 - loading custom policy, 42–43
 - permissions, 155–157
 - policy file, 155–156
 - role of class loaders, 157–159
 - serialization, 107–108
- Security manager and blocked permissions, 156–157
- SecurityManager class, 77
- SelfDestruct class, 156–157
- Serializable, 271
 - classes as attributes, 179–182
 - fields, 119–124
 - object, 179
- Serializable class, 125
- SerializableAttribute class, 179–181
- Serialization, 66–67, 105
 - base class fields, 109
 - basics, 106–111
 - compatible and incompatible changes, 117–118
 - constructors, 110–111
 - custom class descriptors, 124
 - custom code, 271–276
 - deserialize-on-demand, 149–150
 - disabling, 109
 - EJB (Enterprise JavaBeans), 105
 - explicitly managing serializable fields, 119–124
 - finding class code, 147–150
 - fingerprint, 114–115
 - JavaBeans, 105
 - JINI, 105
 - local classes, 107
 - matching streams with classes, 113–118
 - metadata, 105–106
 - NonSerializable instance, 135–136
 - object graphs, 130–133
 - object replacement, 134–147
 - readObject method, 111–112
 - RMI (Remote Method Invocation), 105, 148
 - security, 107–108
 - skipping some fields, 109
 - static fields, 109
 - SUID (serialVersionUID), 114–115
 - transparently handling object graphs, 145
 - writeObject method, 111–112
 - writing custom data, 124–125
- SerializeMe class, 273
- SerialPersistentFields array, 123, 125
- serialver tool, 114, 116
- serialVersionUID field, 115–116, 271
- Server code, 30
- Servers, installing dynamic proxies, 90–91
- Services, generic, XVI
- Servlets, 282
- Session beans, 260
- Session with PointClient listing, 32–33
- setAccessible method, 76–81
- SetErrorInfo method, 295
- setExpiration method, 170
- SetIntArrayRegion method, 235
- setURLStreamHandlerfactory method, 171–172
- SHA-1 (Secure Hash Algorithm), 114
- Shared stubs, 303–307
 - generating, 311–314
 - native methods implementation, 305–306
- Sharing
 - code, 5
 - components, 11
 - packages, 39
- Signatures and primitive types, 75
- SignedBy attribute, 155
- SimplePerson class, 135
- Simplicity application, 15
- Singleton, 138
- Skeletons, 255, 256, 264
- Skipping metadata, 127
- Small-grained methods, 211
- SOAP, 254–255
- Source code, ease of generating, 251
- Source-aware class loaders, 23
- SourceFile attribute, 101
- specVersion value, 96
- Splitting applications into components, 11–12
- STA (single-threaded apartment), 297
- Stack classes, 269–270
- Standalone applications, 11
- Static bind mode, 251, 254
- static fields and serialization, 109

- static final primitive type, 80
- Static proxies, 6
- Statically typed, 20
- Stream handlers, 174–175
- Stream-level object replacement, 134, 139–140
- StreamPerson class, 136
- Streams
 - matching with classes, 113–118
 - writing correct format, 121–122
- String class, 49, 123, 240
- Strings
 - managing, 239–240
 - Unicode, 240
 - UTF acronym, 240
- StringStack class, 269
- Strongly-typed collections, 267–271
- Strongly-typed stack classes, 268–269
- Stub classes, 300
- Stubs, 255, 256, 264
 - APIs sharing, 304
 - automating generation of, 257
 - COM (Component Object Model), 297
 - finalize method, 293–294
 - generating, 311–316
 - IDL (Interface Definitions Language), 292
 - performance, 288
 - Release method, 293–294
- StuffedAnimal class, 50–51
- StuffedAnimalFactory class, 50–51
- SUID (serialVersionUID), 114–116
- sun.boot.library.path, 202
- sun.net.www.protocol package, 173
- superclass reference, 63
- Superhero class, 77–78
- Superinterfaces in class loader hierarchy, 50
- suppressAccessChecks permission, 78–79
- Synthetic attribute, 101
- System class loader, 25, 36, 37, 45, 48
 - extensions path loader, 39–41
 - implicitly delegating, 28
- Systems, evaluating performance, 92

T

- Target language, 270–271
- Tasks, automating, XVI
- TeddyBear class, 51–52
- Teller class, 263

- TestTransformingLoader class, 166
- TestTrappingHandler class, 90
- Thread affinity, 296–298, 302
- Thread context class loader, 53–54
- Thread-affine components, 296–297
- Thread-affine resources, 296
- Threads
 - thread-specific context loader, 53
 - user-interface subsystems, 297
- threaten method, 58–59, 60
- ThreeRoundFiveDollar class, 244–245
- Throw method, 223
- ThrowNew method, 223
- TimeNativePoint class, 211
- TimePoint class, 211
- Timer class, 92
- TLB (Type Library), 292, 300
- TNA (thread-neutral apartment), 298
- TokenInformationClass class, 301
- toString method, 86
- Tracing class loading, 45
- transfer method, 261, 262
- TransformClassBytes method, 162, 166
- Transforming class loaders, 162–167
- TransformingClassLoader class, 162–167
- transient keyword, 109, 131, 137
- Translucent stubs, 286–289
- Transparency, 15
- TrappingHandler class, 88, 90
- Troubleshooting native loading, 207
- Trusted code, 175
- Twain, Mark, 191
- Type errors, 70–71
- Type information, XVI, 57, 67, 281
 - as free specification document, 250–251
 - COM (Component Object Model), 292–293
 - custom type formats, 300
 - DLLs (Dynamic Link Libraries), 300–301
 - header files, 300
 - IDL (Interface Definitions Language), 292–293
 - IDL files, 300
 - TLBs (type libraries), 300
 - type library, 292
 - uses for, XVI
- Type library (TLB), 292, 300
- Type mappings, 195–197
- Type-safety, 193

U

UltimateQuestion class, 197, 199–201
 UndeclaredThrowableException class, 82, 89
 Unicode, 240
 Unintended references, 35
 Unloading classes, 35–36
 UnreflectiveSerialize generator, 275, 276
 UnsatisfiedLinkError class, 205–206
 UpdateVersionInfo method, 187
 URL class, 168, 173
 URLClassLoader class, 18–19, 23, 46, 55–56, 79, 95, 97, 113, 148, 153–154, 162, 165–166, 174, 176
 adding logging, 46–47
 output from logging version, 48
 replacing, 49
 URLConnection class, 170
 URLs, 18
 URLStreamHandler class, 169, 171, 174
 URLStreamHandlerFactory interface, 172
 useResource method, 232
 User-interface subsystems and threads, 297
 UTF acronym, 240

V

validateObject method, 145–146
 Variabilities, 243
 choice or specification for, 244
 inheritance, 245–246
 object-oriented approaches to modeling, 244–246
 VARIANT data type, 293
 Vector class, 269–270
 verbose: class flag, 56
 Verbose:jni flag, 207
 Version
 attributes classes, 178–179
 incompatibilities, 16–17
 mismatches, 59
 Version information, 94
 adding to jar files, 95–96
 weaknesses, 97–98
 Version strings, 96
 VersionInfo class, 181–182
 VersioningLoader class, 183–189

VersionMatcher interface, 187–188
 versionMatches method, 187
 Virtual and non-virtual invocation, 215
 Virtual machine, 192
 class name format, 63–64
 custom attributes, 101–102
 damaged, 24
 garbage collection, 35
 hook for customization, 99
 integrity endangered, 24
 local references, 227
 native code failures, 217–218
 preloading classes, 20
 standard attributes, 99–101
 Visibility rule, 25–26, 29

W

Win32 API, 285–286
 DLLs (Dynamic Link Libraries), 299–303
 platform impedance, 290
 shared stubs, 304–305
 WireWorker class, 142
 withdraw method, 262, 263
 WORA (write-once, run-anywhere), 191
 Worker class, 141–144
 Wrapping primitive types, 74–76
 writeClassDescriptor method, 124
 writeExternal method, 271
 writeFields method, 121
 WriteInstance class, 108–110
 WriteInt method, 125
 writeObject method, 111–112, 121, 122–123, 125, 128–130, 276
 WriteSimplePerson class, 136–137
 Writing raw data only, 128–130
 WSDL (Web Services Description Language), 282

X

X-Code, 315
 XML, XVI, 282
 XPath, 282

Y

YouMustReset class, 133