

Index

SYMBOLS

- & (ampersand)**, in `extends` clause, 5
- <> (angle braces)**
 - in generic class definitions, 3, 5
 - in generic method definitions, 6
- * (asterisk)**, in regular expressions, 53, 57, 58
- \ (backslash)**, meta-character for, 55
- { } (braces)**, in regular expressions, 56–58
- ^ (caret)**, in regular expressions, 55, 56
- :** (colon)
 - in `for` loop, 7–8
 - in manifest file, 629
- \$ (dollar sign)**, in regular expressions, 55
- ...** (ellipsis), for variable arguments, 9–10
- = (equal sign)**, in property file, 629
- / (forward slash)**
 - directory separator, 42
 - in preference nodes, 63
- () (parentheses)**, in regular expressions, 58
- % (percent sign)**, prefixing filename patterns, 42–43
- .** (period), in regular expressions, 55, 56
- +** (plus sign), in regular expressions, 57, 58
- ?** (question mark), in regular expressions, 57, 58
- [] (square brackets)**, in regular expressions, 56

A

- `absolute()` method, `ResultSet` class, 300
- `absolutePath()` method, `Preference` class, 65
- Abstract Windowing Toolkit (AWT) classes**, 143
- abstraction**, 114
- Accessibility classes**, 143
- actions**, `Swing`, 233–234, 239–242
- `Activatable` class, 450
- activatable remote objects**, 450
- `Adaptee` class, **Adapter pattern**, 121
- `Adapter` class, **Adapter pattern**, 121
- Adapter pattern**, 119–122
 - `addFolder()` method, 440
 - `addHandler()` method, `Logger` class, 31
 - `addListener()` method, `MessageConsumer` class, 548
 - `addLogger()` method, `LogManager` class, 29
 - `addMessage()` method, 439
 - `addNodeChangeListener()` method, `Preference` class, 67
 - `addPreferenceChangeListener()` method, `Preference` class, 67
 - `addPropertyChangeListener()` method, `LogManager` class, 30
 - `addRowSetListener()` method, `RowSet`, 308
 - `AfterLast()` method, `ResultSet` class, 300
- agile methodologies**
 - UP (Unified Process), 83–85, 86–87
 - XP (eXtreme Programming), 81, 85–87
- Agile Modeling: Effective Practices for Extreme Programming and the Unified Process** (Ambler, Scott), 75, 661
- `algorithm()` method, `Key` interface, 592–593
- `AlgorithmParameterGenerator` class, **JCA**, 585, 598
- `AlgorithmParameters` class, **JCA**, 585, 597–598

algorithms, replacing on the fly, 134–138, 192

`aliases()` **method**, `KeyStore` **class**, 596

Ambler, Scott (*Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*), 75, 661

ampersand (&), in extends clause, 5

angle braces (<>)

in generic class definitions, 3, 5

in generic method definitions, 6

Annotation Editor example

`AnnotationEditor` class, 198–200

`ComboListener` class, 213–214

`ComponentListener` class, 202–204

`ExcelAction` class, 207–209

`HighlightPainter` class, 204–205

`OpenAction` class, 209–212

`PopupListener` class, 201–202

`PrintAction` class, 205–207

`XmlAction` class, 207

`Annotation` **interface**, 18

`annotation` **package**, 17

`AnnotationDesc` **interface**, 20–21

`AnnotationDesc.ElementValuePair` **interface**, 21

`AnnotationEditor` **class**, **Annotation Editor example**, 198–200

annotations

definition of, 17–18

examples of, 19–20, 22–26

interfaces in doclet API for, 20–22

types of, 18–19

`annotations()` **method**, 20

`annotationType()` **method**, `AnnotationDesc` **interface**, 21

`AnnotationTypeDoc` **interface**, 21

`AnnotationTypeElementDoc` **interface**, 21

`AnnotationValue` **interface**, 22

Ant application (Apache)

building projects with, 655–658

definition of, 654

development scenarios using, 87–95

installing, 655

learning, using patterns for, 113

running TCPMon with, 496

Apache AXIS

definition of, 533–534

deploying a Web service, 535–537

setting up, 534–535

TCPMon included in, 496

version of, returning, 528–529

writing Web service client, 537–539

Apache Jakarta Project, 499

Apache TCPMon, 496–498, 534, 539

Apache Tomcat server, 321, 534

The Apache XML Project, 499

APIs, learning using patterns, 112–113. See also specific APIs

`appendReplacement()` **method**, `Matcher` **class**, 59

`appendTail()` **method**, `Matcher` **class**, 59

applets

definition of, 636

in JAR files, 629–630

packaging for execution, 638

RMI for, 446

security of, 639

structure of, 636–638

`appletviewer` **command**, 638

application data

definition of, 224–225

persisting in Swing application, 232–235

persisting (saving), 225–227, 230–232

application layer, 479–480

Application scope, WebWork framework, 374

applications. See also database, persisting applications with; serialization; software design and development

deploying

Ant application for, 654–658

applets, 638–639

classpaths, managing, 619–624

EAR (Enterprise Archive), 644–646

EJBs (Enterprise JavaBeans), 643

endorsed directory and, 624

JAR files for, 625–636

Java Web Start, 647–654

Web applications, 639–643

time-based license for

definition of, 235–236

implementing license, 236–238

implementing timeserver, 238–239

applicationScope implicit object, 340

`application.xml` **file**, 644–645

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (Larman, Craig), 83, 111, 661

arguments, variable, 2, 9–10

arrays

- of generic types, 6
- iterators for, 8
- in JNI, 410–416
- of type variables, 6

`asResult()` **method**, `Matcher` **class**, 59

Assertions, JMeter, 108

assignment, boxing and unboxing conversions in, 12–13

associations between classes, creating, 115–116

asterisk (*), in regular expressions, 53, 57, 58

attributes, static, importing, 13–15

authentication

definition of, 612

JAAS (Java Authentication and Authorization Service)

authenticating a subject, 615, 616–617

authorization, 617–618

`AuthPermission` class, 617–618

configurations for authentication, 615–616

credentials, 613, 615

definition of, 612

`Destroyable` interface, 615

executing code with security checks, 613–617

`LoginContext` class, 615

policy file for authorization, 617–618

principals, 614

`PrivilegedAction` interface, 613

`Refreshable` interface, 615

`Subject` class, 612–613

user identification, 612–613

MAC (Message Authentication Code), 583, 602, 611–612

authorization, 617–618

`AuthPermission` **class**, **JAAS, 617–618**

AWT (Abstract Windowing Toolkit) classes, 143

AXIS (Apache)

definition of, 533–534

deploying a Web service, 535–537

setting up, 534–535

TCPMon included in, 496

version of, returning, 528–529

writing Web service client, 537–539

B

backreferences, in regular expressions, 58

backslash (\), meta-character for, 55

bank application example, 478

Beck, Kent (eXtreme Programming Explained), 85, 661

beep character, meta-character for, 55

`BeforeFirst()` **method**, `ResultSet` **class**, 300

`beginTransaction()` **method**, `Session` **class**, 319

bell character, meta-character for, 55

Berners-Lee, Tim (visionary for World Wide Web), 523

BorderLayout manager, 144–151

`BorderLayout()` **method**, `BorderLayout` **class**, 145

boxing and unboxing conversions, 2, 11–13

BoxLayout manager

definition of, 151–152

example of

`CondimentPanel` class, 156–157

Decorator pattern in, 152–153

`DropTargetListener` interface, 154–156

`Food` class, 157–158

`FoodCourt` interface, 153, 156

`FoodGraphic` class, 153–154

`FoodItems` class, 158–161

model of, 152

`BoxLayout()` **method**, `BoxLayout` **class**, 152

braces {}, in regular expressions, 56–58

bug reporting and tracking, 81

`build()` **method**, `CertPathBuilder` **class**, 601

business logic, separating from User Interface logic, 122–130

business tier, J2EE, 87

`Buyer` **class**, **Strategy pattern, 136**

C

CA (certificate authority), 600

CachedRowSetImpl RowSet implementation, 309

Call Level Interface (CLI), X/Open SQL, 282

CallableStatement interface, 292–294, 297

CallbackHandler class, JAAS, 616

`CallNonvirtual[Type]Method()` **function**, 421–423

`CallNonvirtual[Type]MethodA()` **function**, 421–423

`CallNonvirtual[Type]MethodV()` **function**, 421–423

`Call[Type]Method()` **function**, 421–423

`Call[Type]MethodA()` **function**, 421–423

`Call[Type]MethodV()` **function**, 421–423

CardLayout manager

alternative to, 214

definition of, 191

CardLayout manager (continued)

CardLayout **manager (continued)**

example of

CardLayoutPanel class, 192–194

JButtonStrategy1 class, 194–195

JButtonStrategy2 class, 195

Strategy pattern in, 192

TestStrategy interface, 196–197

CardLayout() **method**, CardLayout **class**, 191

caret (^), in regular expressions, 55, 56

carriage-return, meta-character for, 55

catch clause, type variables in, 7

C/C++ programs, connecting to Java programs with JNI

arrays in, 410–416

data type conversions, 406

e-mail client example using

JNIMailBridge class, 436–439

MAPI routines used in, 439–444

system design for, 434

user interface for, 435

fields, accessing, 416–419

header file for native code, creating, 403–404

invoking native routines, 405

Java code invoking native routines, creating, 402–403

Java exceptions, handling in native code, 423–424

Java objects, using in C/C++, 416–423, 425–429

Java threading, 429–430

methods, invoking in, 419–423

native methods, registering manually, 430–432

native routine library, creating, 404–405

native routines, using in Java code, 401–405

NIO direct buffers, using in, 430

reflection using, 432–434

strings in, 406–410

CDO (Collaborative Data Objects), 434

certificate authority (CA), 600

certificate management, 600–602

Certificate Revocation List (CRL), 584, 600–601

CertificateFactory **class**, JCA, 585, 600–601

certificates, database of, 596–597

certification path, 600

CertPathBuilder **class**, JCA, 585, 601

CertPathValidator **class**, JCA, 585, 601

CertStore **class**, JCA, 585, 601–602

cglib-2.0-rc2.jar **file**, 314

Chain of Responsibility pattern, 162, 164, 166

ChainingInterceptor, **WebWork framework**, 373

character classes, in regular expressions, 56–57

childrenNames() **method**, Preference **class**, 65

cipher, 602

Cipher **class**, JCE

definition of, 603–604

encrypting/decrypting data, 604

wrapping/unwrapping keys, 604–608

CipherInputStream **class**, JCE, 603, 605

CipherOutputStream **class**, JCE, 603, 605

classes. See also objects; specific classes

associations between, creating, 115–116

as data structures, 225

designing, 115

finding in JAR files, 620–624

object graphs for, 225

parameterized, 3–4

serializable, encrypting, 609–611

statically importing data from, 13–15

type-safe (generics), 1, 2–7, 13

classpaths

definition of, 619

determining location of class in, 621–624

guidelines for, 620

limitations of, 619–620

verifying list of classes in, 620

ClassPathVerifier utility, 620

ClassSearch utility

findClass() **method**, 622

findHelper() **method**, 621–622

main() **method**, 623

searchClassPath() **method**, 622

searchJarFile() **method**, 621

using, 623–624

clear() **method**, Preference **class**, 67

clearMessageList() **method**, 439, 440, 444

clearParameters() **method**, PreparedStatement **interface**, 290

CLI (Call Level Interface), X/Open SQL, 282

Client **class**, Adapter pattern, 120

client layer

three-tier model, JDBC API, 284

two-tier model, JDBC API, 283

client tier, J2EE, 87

close() **method**

Handler class, 39

MemoryHandler class, 44

ResultSet class, 302

Session class, 319

SocketHandler class, 41

StreamHandler class, 41

- CLOSE_CURSORS_AT_COMMIT** constant, 299
- CMT (Contact Management Tool) application**
 - adding new contact to, 355–356
 - adding pictures to, 346
 - definition of, 340
 - registering contacts in, 348–350
- code reuse, in JSP 2.0, 335–336**
- CodeGenerator, Hibernate, 315**
- Collaborative Data Objects (CDO), 434**
- collections, 2, 138–142**
- colon (:)**
 - in for loop, 7–8
 - in manifest file, 629
- ComboBoxListener class, Annotation Editor example, 213–214**
- Command class, Command pattern, 130–131**
- Command pattern**
 - Command interface, 130–131
 - CommandManager class, 131
 - definition of, 130
 - example of, 178, 181, 192
 - invoker for, 131–134
- CommandManager class, Command pattern, 131**
- commit() method, Connection class, 311**
- committing transactions, 310–311**
- Common Object Request Broker Architecture (CORBA)**
 - COS (Common Object Service) Naming, 506–507, 509
 - definition of, 505–507
 - example using, 513–522
 - IDL (Interface Definition Language), 507–509
 - IIOB (Internet InterORB Protocol), 503, 506–507, 509
 - ORB (Object Request Broker), 506–507, 509
 - overriding in endorsed directory, 624
 - RMI compatibility with, 510–512
 - when to use, 512
- Common Object Service (COS) Naming**
 - definition of, 506–507, 509
 - overriding in endorsed directory, 624
- commons-collections-2.1.jar file, 314**
- commons-logging-1.0.3.jar file, 314**
- communication, importance in software development, 75**
- communication between components**
 - architecture for, 479–480
 - CORBA (Common Object Request Broker Architecture), 505–512
 - EJBs (Enterprise JavaBeans), 465–475
 - RMI (Remote Method Invocation)
 - architecture of, 446–447
 - communication transport protocol for, 447
 - CORBA compatibility with, 510–512
 - definition of, 445–446, 465, 500–501
 - developing applications with, 448–449
 - distributed objects, 445, 504–505
 - dynamic class loading, 449
 - garbage collection by, 447, 449
 - marshalling and unmarshalling, 501–503
 - network failure and, 447
 - performance of, 447
 - protocols, 503
 - Remote Object Activations, 449–453
 - RMIChat example using, 453–465
 - security and, 447
 - serialization and, 235
 - stubs, generating, 448
 - threading, 448
 - sockets
 - definition of, 480
 - Java Socket API, 481–487
 - protocol, implementing, 487–499
 - types of, 480
 - technologies for, 478
 - Web services
 - definition of, 522–523, 639
 - example using, 523–526, 531–540
 - future of, 540
 - limitations of, 445, 522, 527
 - remote procedure calls with, 526–527
 - SOAP and, 529–530
 - types of, 536
 - when to use, 522–523
 - WSDL and, 528–529
- compile() method, Pattern class, 58**
- Component interface, Composite pattern, 139**
- ComponentListener class, Annotation Editor example, 202–204**
- components. See also communication between components**
 - definition of, 477
 - horizontal, 369
 - JavaBean components, 248–256, 308–309
 - scope of, in WebWork framework, 374
 - Swing components, serialization of, 255
 - vertical, 369
- Composite class, Composite pattern, 140–142**
- Composite pattern, 138–142**
- CONCUR_READ_ONLY** constant, 298
- concurrency of result sets, 298**
- Concurrent Versioning System (CVS), 79**

CONCUR_UPDATABLE constant, 298, 300**Config Elements, JMeter, 108**

`config()` **method**, **Logger class, 33**

configuration data

- deserializing, 232
- modeling, 226–227
- serializing with Java Serialization API, 230–235, 244–245
- serializing with JAXB (Java API for XML Binding)
 - example of, 272–278
 - generating classes from XML schema, 263–268
 - sample XML document for, 257–259
 - XML schema for, 259–263
- serializing with XMLEncoder/Decoder API, 252–254
- verification and validation for, 244–245

configuration information (Java preferences)

- definition of, 63
- examples of, 69–71
- exporting to XML, 68–71
- Preference class, 63–68

configuration management, 78–79**connected RowSet implementations, 308****Connection class**

- JDBC API, 286–287, 311
- JMS, 545, 558

ConnectionFactory class, JMS, 545, 558**connections to database, 286–287, 310****ConsoleCorbaServer command, 521****ConsoleHandler class, 41****Contact Management Tool (CMT) application**

- adding new contact to, 355–356
- adding pictures to, 346
- definition of, 340
- registering contacts in, 348–350
- `contains()` **function**, **JSTL, 341**
- `containsIgnore()` **function**, **JSTL, 341**
- `<context-param>` **element**, **WAR deployment descriptor, 640**

continuous integration, 79**control character, meta-character for, 55****Controller, in MVC, 366–367****Controller class, Model-View-Controller pattern, 125–127****Cookie implicit object, 340****CORBA (Common Object Request Broker Architecture)**

- COS (Common Object Service) Naming, 506–507, 509
- definition of, 505–507
- example using, 513–522
- IDL (Interface Definition Language), 507–509

- IIOB (Internet InterORB Protocol), 503, 506–507, 509
- ORB (Object Request Broker), 506–507, 509
- overriding in endorsed directory, 624
- RMI compatibility with, 510–512
- when to use, 512

CORBA.Object class, 517**CORBA.ORB class, 517****COS (Common Object Service) Naming**

- definition of, 506–507, 509
- overriding in endorsed directory, 624

CosNaming.NamingComponent class, 517**CosNaming.NamingContext class, 517****createConsumer() method, Session class, 548****createCriteria() method, Session class, 319****createPublisher() method, Session class, 547****createSession() method, Connection class, 546****createTextMessage() method, Session class, 547****credentials, 613, 615****CRL (Certificate Revocation List), 584, 600–601****cryptography**

- JCA (Java Cryptography Architecture)
 - algorithm management, 597–598
 - certificate management, 600–602
 - definition of, 583, 584
 - digital key creation, 592–596
 - digital key storage and management, 596–597
 - digital signing and verification, 588–592
 - engine classes in, 584–585
 - message digests, calculating and verifying, 586–588
 - provider packages for, alternatives, 585
 - random number generation, 599–600
 - SUN provider package for, 584
- JCE (Java Cryptography Extension)
 - Cipher class, 603–608
 - converting keys between transparent and opaque, 608–609
 - definition of, 583, 602
 - encrypting and decrypting data, 603–604
 - encrypting serializable classes, 609–611
 - generating secret keys, 608
 - KeyGenerator class, 608
 - message authentication codes, computing, 611–612
 - SealedObject class, 609–611
 - SecretKeyFactory class, 608–609
 - services provided by, 602–603
 - wrapping and unwrapping keys, 604–608

cursor in a result set, 298**cursorMoved event, RowSet, 308****CVS (Concurrent Versioning System), 79**

D**data, application**

- definition of, 224–225
- persisting in Swing application, 232–235
- persisting (saving), 225–227, 230–232

data layer, three-tier model, JDBC API, 284**data model, 224****data structures, classes as, 225****data types**

- for arrays, 410, 411–412
- conversions between
 - boxing and unboxing, 2, 11–13
 - Java and C++ types, 406
 - Java and JDBC types, 290–292
- descriptors for, 417
- translating to C++, 406
- type-safe classes (generics), 1, 2–7, 13
- type-safe enumerations, 2, 15–17

database

- keyword search of, 304–308
- SQL Actions in JSTL, 342–344

database, persisting applications with. See also serialization

- using Hibernate tool
 - architecture of, 312–313
 - configuration, properties for, 317
 - databases supported by, 314
 - forum example using, 320–327
 - persisting objects to database, 317–319
 - requirements for, 314–315
 - XDoclet and, 104
 - XML mappings, 315–317
- using JDBC API
 - connections, managing, 286–287
 - connections, pooling, 310
 - driver types for, 282
 - features of, 281–282
 - meta data for data source, retrieving, 302–308
 - packages in, 283
 - requirements for, 283
 - result sets for SQL queries, 298–302
 - RowSets, using, 308–309
 - SQL batch updates, executing, 294–297
 - SQL statements, executing, 287–294
 - three-tier model for, 284–285
 - transactions, managing, 310–312
 - two-tier model for, 283–284

database of keys and certificates (keystore), 596–597**DatabaseMetaData interface, JDBC API**

- definition of, 302–303
- features of data source, determining, 303–308
- limitations of data source, determining, 303
- methods in, 294

DatagramSocket class, 481**DataSource interface, JDBC API, 286–287****declarations in EL, 333****Decorator pattern, 152–153, 160****defaultValue() method,**

- AnnotationTypeElementDoc interface, 21

DefaultWorkflowInterceptor, WebWork framework, 373**DELETE command, HTTP, 489****delete() method, Session class, 319****deleteEntry() method, KeyStore class, 597****DeleteGlobalRef() function, 428****DeleteLocalRef() function, 416, 425****deleteRow() method, ResultSet class, 301****DeleteWeakGlobalRef() function, 428****deploying Java applications**

- Ant application for, 654–658
 - applets, 638–639
 - classpaths, managing, 619–624
 - EAR (Enterprise Archive), 644–646
 - EJBs (Enterprise JavaBeans), packaging, 643
 - endorsed directory, overriding standards in, 621–624
 - JAR files
 - applets in, 629–630
 - in classpaths, managing, 619–624
 - creating, 94, 625–627
 - definition of, 625
 - digitally signing, 625, 630–634
 - executable, 635–636
 - extracting contents of, 628
 - index file for, 634–635
 - installing as an extension, 620
 - manifest file for, 625, 628–629
 - viewing contents of, 628
 - Java Web Start, 647–654
 - Web applications, 639–643
- Deprecated class, 18, 19**
- <description> element, WAR deployment descriptor, 640**
- descriptors for primitive types, 417**
- deserialization, 228. See also serialization**
- Design Patterns: Elements of Reusable Object-Oriented Software (Gamma, Erich), 111**
- destination, in messaging systems, 544, 564**

Destination **class, JMS, 545, 558**

destroy() **method**

in applets, 637

Destroyable interface, 615

Destroyable **interface, JAAS, 615**

DHPrivateKey **interface, 593**

DHPublicKey **interface, 593**

digest() **method, MessageDigest class, 586**

digital keys. See also encryption

creating, 592–593

definition of, 588

storing and managing, 596–597

Digital Signature Algorithm (DSA), 584, 588–589

Digital Signature Standard (DSS), 589

digital signatures

for data, 588–592

for JAR files, 625

disconnected RowSet implementations, 308

<display-name> **element, WAR deployment descriptor, 640**

<distributable> **element, WAR deployment descriptor, 640**

distributed file system notifications example

description of, 513

IDL generator used in, 514–516

implementing, 516–521

running, 521–522

distributed objects, 445, 504–505. See also RMI

distributed processing

definition of, 543

example application of

Aggregateable interface, 570

deploying, 573–581

description of, 551–552

JMXAgent class, 563–564

JndiHelper class, 556–557

MBean components for, 553

message type for, 552–553

MessageAggregator component, 570–573

MessageListener interface, 555

MessageProcessor component, 553–564

MessageProcessorMBean interface, 555–556

MessageRouter component, 564–566

MessageSplitter component, 566–570

OrderAggregator class, 572–573

OrderProcessor class, 562–563

Processable interface, 562

Routeable interface, 565

Splitable interface, 567

JMS features for, 544–548

JMX features for, 548–551

distributed transactions, 311–312

doAs() **method, 614**

doAsPrivileged() **method, 614**

doclet API, annotations, 19–22

Document Object Model (DOM), 224

Document-based Web services, 536

Documented **class, 18**

doFinal() **method**

Cipher class, 604

Mac class, 611

dollar sign (\$), in regular expressions, 55

DOM (Document Object Model), 224

dom4j libraries, 198, 206

dom4j-1.4.jar **file, 314**

Drag and Drop classes, 143

dragEnter() **method, DropTargetListener interface, 155**

dragExit() **method, DropTargetListener interface, 155**

dragOver() **method, DropTargetListener interface, 155**

DriverManager **class, JDBC API, 286**

drivers for JDBC API, 282

drop() **method, DropTargetListener interface, 155**

dropActionChanged() **method, DropTargetListener interface, 155**

DropTargetListener **interface, 154–155**

DSA (Digital Signature Algorithm), 584, 588–589
.DSA **file extension, 630**

DSAPrivateKey **interface, 593**

DSAPrivateKeySpec **interface, 592**

DSAPublicKey **interface, 593**

DSAPublicKeySpec **interface, 592**

DSS (Digital Signature Standard), 589

Dynamic APIs, overriding in endorsed directory, 624

dynamic class loading in RMI, 449

E

EAR (Enterprise Archive), 644–646

echo server example

description of, 483

running, 487

SocketEcho class, 483–486

The Eclipse Project, 499

EDM (Event Delegation Model), 124

Ehcache-0.6.jar **file, 314**

EIS (Enterprise Information System) tier, J2EE, 87

`ejb-jar.xml` file, 473–475, 643

EJBs (Enterprise JavaBeans). See also JavaBean components

containers, 465, 467–468

definition of, 465

deployment descriptor for, 473–475

in EAR (Enterprise Archive), 644–646

JNDI and, 467

loan calculator example of, 468–475

definition of, 468

EJB deployment descriptor for, 473–475

`LoanBean` class, 469–470

`LoanClient` class, 470–473

`LoanHome` interface, 468–469

`LoanObject` interface, 468

packaging, 643

types of, 466

EL (Expression Language)

Function Tag Library extensions to, 341–342

in JSP scripts, 332–335, 339–340

`element()` method, `AnnotationDesc.ElementValuePair` interface, 21

`elements()` method, `AnnotationTypeDoc` interface, 21

`ElementType` enumeration, 17–18

`ElementValuePair` interface, 21

`elementValues()` method, `AnnotationDesc` interface, 21

ellipsis (...), for variable arguments, 9–10**e-mail client example using JNI**

`JNIMailBridge` class, 436–439

MAPI routines used in, 439–444

system design for, 434

user interface for, 435

`EncodedKeySpec` interface, 592

encryption

JCA (Java Cryptography Architecture)

algorithm management, 597–598

certificate management, 600–602

definition of, 583, 584

digital key creation, 592–596

digital key storage and management, 596–597

digital signing and verification, 588–592

engine classes in, 584–585

message digests, calculating and verifying, 586–588

provider packages for, alternatives, 585

random number generation, 599–600

SUN provider package for, 584

JCE (Java Cryptography Extension)

`Cipher` class, 603–608

converting keys between transparent and opaque, 608–609

definition of, 583, 602

encrypting and decrypting data, 603–604

encrypting serializable classes, 609–611

generating secret keys, 608

`KeyGenerator` class, 608

message authentication codes, computing, 611–612

`SealedObject` class, 609–611

`SecretKeyFactory` class, 608–609

services provided by, 602–603

wrapping and unwrapping keys, 604–608

`end()` method, `Matcher` class, 59, 60

endorsed directory, 624**Endorsed Standard Override Mechanism, 624**

`endsWith()` function, `JSTL`, 341

`EnsureLocalCapacity()` function, 425–426, 427

`entering()` method, `Logger` class, 33

Enterprise Archive (EAR), 644–646**Enterprise Information System (EIS) tier, J2EE, 87****Enterprise Integration Patterns (Hohpe, Gregor), 553****Enterprise JavaBeans (EJBs)**

containers, 465, 467–468

definition of, 465

deployment descriptor for, 473–475

in EAR (Enterprise Archive), 644–646

JNDI and, 467

loan calculator example of, 468–475

definition of, 468

EJB deployment descriptor for, 473–475

`LoanBean` class, 469–470

`LoanClient` class, 470–473

`LoanHome` interface, 468–469

`LoanObject` interface, 468

packaging, 643

types of, 466

entity beans, 466**Entrust, 600**

`Enum` class, 15–17

enumerations, 2, 15–17

`EnumMap` class, 16

`EnumSet` class, 16

`<env-entry>` element, `WAR` deployment descriptor, 641

equal sign (=), in property file, 629

equals() method

`equals()` **method**

Level class, 38

Principal interface, 614

error handling. See exceptions; Java logging

`error()` **method, ErrorManager class, 49**

ErrorManager **class, 49**

`<error-page>` **element, WAR deployment descriptor, 641**

`escapeXml()` **function, JSTL, 341**

estimates for software development, 80–81

Event Delegation Model (EDM), 124

ExcelAction **class, Annotation Editor example, 207–209**

ExceptionCheck() **function, 423**

ExceptionClear() **function, 423–424**

ExceptionDescribe() **function, 423**

ExceptionOccurred() **function, 423**

exceptions

generics and, 7

handling in native code, 423–424

executable JAR file, 635–636

`execute()` **method**

PreparedStatement interface, 289

Statement interface, 288

`executeBatch()` **method, Statement interface, 288**

`executeQuery()` **method**

PreparedStatement interface, 289

Statement interface, 288

`executeUpdate()` **method**

PreparedStatement interface, 289

Statement interface, 288–289

`exiting()` **method, Logger class, 33**

`exportNode()` **method, Preference class, 67**

`exportSubtree()` **method, Preference class, 67**

Expression Language (EL)

Function Tag Library extensions to, 341–342

in JSP scripts, 332–335, 339–340

extends clause

interface restrictions in, 5

super-types defined in, 3

Externalizable **interface, 229–230, 245**

eXtreme Programming Explained (Beck, Kent), 85, 661

eXtreme Programming (XP), 81, 85–87

F

FatalError() **function, 424**

fields, accessing in JNI, 416–419

FileHandler **class, 42–43**

filename patterns, in FileHandler class, 42

FileNotification **interface, 513**

FileSystemWatcher **class, 513–514**

`fill` **variable, GridLayout manager, 178**

`<filter>` **element, WAR deployment descriptor, 640**

Filter **interface, 48**

FilteredRowSetImpl RowSet **implementation, 309**

`<filter-mapping>` **element, WAR deployment descriptor, 641**

`find()` **method**

Matcher class, 60

Session class, 319

FindClass() **function, 433**

`fine()` **method, Logger class, 33**

`finer()` **method, Logger class, 33**

`finest()` **method, Logger class, 33**

First() **method, ResultSet class, 300**

`flags()` **method, Pattern class, 59**

FlowLayout **manager**

definition of, 161

example of

Chain of Responsibility pattern in, 164

FlowLayoutPanel class, 162–164, 166

QuarterHandler class, 165–166

TestHandler class, 165

`FlowLayout()` **method, FlowLayout class, 161**

`flush()` **method**

Handler class, 39

MemoryHandler class, 44

Preference class, 67

StreamHandler class, 41

for loop, new features of, 1, 7–9

`format()` **method, Formatter class, 45**

`formatMessage()` **method, Formatter class, 45**

Formatter **class, 44–48**

form-feed, meta-character for, 55

forum example, Hibernate tool

architecture of, 320

code for, 324–327

database for, 321

file structure for, 321–322

user interface for, 322–323

forward slash (/)

directory separator, 42

in preference nodes, 63

Fowler, Martin (Refactoring), 77, 111, 661

framework for Model 2 Architecture

architecture of, 371–374

definition of, 368–369

IoC (Inversion of Control), 369–371

FromReflectedField() **function**, 433–434

FromReflectedMethod() **function**, 433

function signatures (prototypes), 405

Function Tag Library, JSTL, 341–342

functions, C/C++, using in Java programs with JNI

arrays in, 410–416

data type conversions, 406

e-mail client example using

JNIMailBridge class, 436–439

MAPI routines used in, 439–444

system design for, 434

user interface for, 435

fields, accessing, 416–419

header file for native code, creating, 403–404

invoking native routines, 405

Java code invoking native routines, creating, 402–403

Java exceptions, handling in native code, 423–424

Java objects, using in C/C++, 416–423, 425–429

Java threading, 429–430

methods, invoking in, 419–423

native methods, registering manually, 430–432

native routine library, creating, 404–405

native routines, using in Java code, 401–405

NIO direct buffers, using in, 430

reflection using, 432–434

strings in, 406–410

G

Gaim, 498

Gamma, Erich (*Design Patterns: Elements of Reusable Object-Oriented Software*), 111

garbage collection, RMI applications, 447, 449

generateCertificate() **method**,

CertificateFactory **class**, 600

generateCertificates() **method**,

CertificateFactory **class**, 600

generateCertPath() **method**,

CertificateFactory **class**, 601

generateCRL() **method**, CertificateFactory **class**, 600–601

generateCRLs() **method**, CertificateFactory **class**, 600–601

generateKey() **method**, KeyGenerator **class**, 608

generateKeyPair() **method**, KeyPairGenerator **class**, 594, 596

generateParameters() **method**,

AlgorithmParameterGenerator **class**, 598

generatePrivate() **method**, KeyFactory **class**, 593–594

generatePublic() **method**, KeyFactory **class**, 593–594

generateSecret() **method**, SecretKeyFactory **class**, 608–609

generateSeed() **method**, SecureRandom **class**, 599

generics

arrays of, 6

boxing and, 13

class instances, 5–6

classes, 3–5

definition of, 1, 2

exceptions and, 7

hierarchy restrictions of, 3–4

interfaces, 3–5

methods, 6–7

raw types (type erasure), 4–5

super-types of, 3

GET command, HTTP

definition of, 489–490

implementing, 489, 491–495

get() **method**, Preference **class**, 65

getAlgorithm() **method**, SealedObject **class**, 609, 611

getAnonymousLogger() **method**, Logger **class**, 31

getArrayLength() **function**, 411, 414, 416

getBoolean() **method**, Preference **class**, 66

getByteArray() **method**, Preference **class**, 66

getCertificate() **method**, KeyStore **class**, 597

getCertificateAlias() **method**, KeyStore **class**, 597

getCertificateChain() **method**, KeyStore **class**, 597

getCertificates() **method**, CertStore **class**, 602

getCertPathEncodings() **method**, CertificateFactory **class**, 601

getCertStoreParameters() **method**, CertStore **class**, 602

getCRLs() **method**, CertStore **class**, 602

GetDirectBufferAddress() **function**, 430

GetDirectBufferCapacity() **function**, 430

getDouble() **method**, Preference **class**, 66

getEncoded() **method**

AlgorithmParameters class, 598

Key interface, 593

getEncoding() **method**, Handler **class**, 39

getErrorMessage() **method**, Handler **class**, 39

GetFieldID() **function**, 417

getFilter() method

- getFilter() **method**
 - Handler class, 39
 - Logger class, 32
- getFloat() **method**, Preference **class**, 66
- getFolderContents() **method**, 439, 441
- getFolderList() **method**, 439, 440
- getFormat() **method**, Key **interface**, 593
- getFormatter() **method**, Handler **class**, 39
- getHandlers() **method**, Logger **class**, 32
- getHead() **method**, Formatter **class**, 45
- getInstance() **method**
 - CertStore class, 601–602
 - Cipher class, 603
 - MessageDigest class, 586
- getInt() **method**, Preference **class**, 66
- getKey() **method**, KeyStore **class**, 597
- getKeySpec() **method**
 - KeyFactory class, 594
 - SecretKeyFactory class, 609
- getLevel() **method**
 - Handler class, 39
 - Logger class, 32
 - LogRecord class, 35
- getLocalAddr() **method**, ServletRequest **class**, 332
- getLocalizedName() **method**, Level **class**, 38
- getLocalName() **method**, ServletRequest **class**, 332
- getLocalPort() **method**, ServletRequest **class**, 332
- getLogger() **method**
 - Logger class, 31
 - LogManager class, 29
- getLoggerName() **method**, LogRecord **class**, 35
- getLoggerNames() **method**, LogManager **class**, 29
- getLogManager() **method**, LogManager **class**, 29
- getLong() **method**, Preference **class**, 66
- getMaxColumnsInTable() **method**, DatabaseMetaData **class**, 303
- getMaxRowSize() **method**, DatabaseMetaData **class**, 303
- getMaxStatementLength() **method**, DatabaseMetaData **class**, 303
- getMaxStatements() **method**, DatabaseMetaData **class**, 303
- getMaxUserNameLength() **method**, DatabaseMetaData **class**, 303
- getMessage() **method**, LogRecord **class**, 35
- GetMethodID() **function**, 420–421
- getMillis() **method**, LogRecord **class**, 35
- getName() **method**
 - Level class, 38
 - Logger class, 32
 - Principal interface, 614
- getNumericFunctions() **method**, DatabaseMetaData **class**, 294
- getObject() **method**, SealedObject **class**, 609
- GetObjectArray() **function**, 411
- GetObjectArrayElement() **function**, 416
- GetObjectClass() **function**, 419, 420, 433
- getOutputSize() **method**, Cipher **class**, 605
- getParameterMetaData() **method**, PreparedStatement **interface**, 292
- getParameters() **method**, LogRecord **class**, 35
- getParameterSpec() **method**, AlgorithmParameters **class**, 598
- getParent() **method**, Logger **class**, 32
- GetPrimitiveArrayCritical() **function**, 413
- getPrincipals() **method**, Subject **class**, 613
- getPrivate() **method**, KeyPair **class**, 594
- getPrivateCredentials() **method**, Subject **class**, 613
- getProcedures() **method**, DatabaseMetaData **class**, 294
- getProperty() **method**, LogManager **class**, 29
- getPublic() **method**, KeyPair **class**, 594
- getPublicCredentials() **method**, Subject **class**, 613
- getPushLevel() **method**, MemoryHandler **class**, 44
- getRemotePort() **method**, ServletRequest **class**, 332
- getResourceBundle() **method**
 - Logger class, 32
 - LogRecord class, 36
- getResourceBundleName() **method**
 - Level class, 38
 - Logger class, 32
 - LogRecord class, 36
- getSequenceNumber() **method**, LogRecord **class**, 35
- getSourceClassName() **method**, LogRecord **class**, 35
- getSourceMethodName() **method**, LogRecord **class**, 35

GetStaticFieldID() **function, 418**
 GetStaticMethodID() **function, 420–421**
 GetStatic[Type]Field() **function, 418**
 GetStringChars() **function, 408**
 GetStringCritical() **function, 408**
 getStringFunctions() **method,**
 DatabaseMetaData **class, 294**
 GetStringLength() **function, 408**
 GetStringRegion() **function, 408**
 GetStringUTFChars() **function, 408, 410**
 GetStringUTFLength() **function, 408**
 GetStringUTFRegion() **function, 408**
 getSubject() **method, Subject class, 613**
 GetSuperclass() **function, 433**
 getSystemFunctions() **method,**
 DatabaseMetaData **class, 294**
 getTail() **method, Formatter class, 45**
 getThreadID() **method, LogRecord class, 35**
 getThrown() **method, LogRecord class, 35**
 getTimeDateFunctions() **method,**
 DatabaseMetaData **class, 294**
 Get[Type]ArrayElements() **function, 412, 414**
 Get[Type]ArrayRegion() **function, 412–413**
 Get[Type]Field() **function, 417, 419**
 getUseParentHandlers() **method, Logger**
 class, 32
global references to objects, 425, 427–429
greedy operators, in regular expressions, 57
 GridBagLayout **manager**
 definition of, 177–178
 example of
 GridBagLayoutPanel class, 179–181
 GridBagLayoutPanel display, 179
 JComboQuestion class, 181–182
 patterns used in, 178
 running, 182–183
 GridBagLayout() **method, GridBagLayout**
 class, 177
 GridLayout **manager**
 definition of, 167
 example of
 DBPanel class, 173–175
 GridLayoutPanel class, 168–171
 GridLayoutPanel display, 167
 Java2DPanelMouseover class, 171–173
 MyTableModel class, 175–176
 running, 176–177

GridLayout() **method, GridLayout class, 167**
 gridx **variable, GridLayout manager, 177**
 gridy **variable, GridLayout manager, 177**
 group() **method, Matcher class, 60**
 groupCount() **method, Matcher class, 60**
GUI applications
 BorderLayout manager, 144–151
 BoxLayout manager, 151–161
 CardLayout manager, 191–197, 214
 FlowLayout manager, 161–166
 GridBagLayout manager, 177–183
 GridLayout manager, 167–177
 layout managers for, 144
 libraries for, 143
 SpringLayout manager, 183–191

H

handle() **method, CallbackHandler class, 616**
 Handler **class**
 definition of, 38
 methods in, 38–40
 predefined handlers for, 40–44
 Handler **property, 28**
hanging sessions in Model 2 Architecture applications,
 375–377
hash value (message digest), 586. See also JCA
 hashCode() **method**
 Level class, 38
 Principal interface, 614
HEAD command, HTTP, 489
header implicit object, 340
headerValues implicit object, 340
hexadecimal values, meta-characters for, 55
Hibernate Extension package, 315
Hibernate tool
 architecture of, 312–313
 configuration, properties for, 317
 databases supported by, 314
 example using, 379–387, 398
 extending Model 2 frameworks with, 374–377
 forum example using
 architecture of, 320
 code for, 324–327
 database for, 321
 file structure for, 321–322
 user interface for, 322–323

Hibernate tool (continued)

- King's example using, 374
- persisting objects to database, 317–319
- requirements for, 314–315
- XDoclet and, 104
- XML mappings, 315–317

HibernateAction **class**, 377

HibernateFactory **class**, 374–375

HibernateInterceptor **class**, 376

hibernate2.jar **file**, 314

high cohesion, 114

HighlightPainter **class**, **Annotation Editor** example, 204–205

hitEnd() **method**, **Matcher** **class**, 60

Hohpe, Gregor (*Enterprise Integration Patterns*), 553

holdability of result sets, 299

HOLD_CURSORS_OVER_COMMIT **constant**, 299, 302

horizontal components, 369

HTTP protocol

- implementing, 488–498
- Web services using, 522–523, 526–527

HttpAdaptor, 551, 563–564, 581

I

<icon> **element**, **WAR deployment descriptor**, 640

IDL (Interface Definition Language), 507–509

IIOIP (Internet InterORB Protocol), 503, 506–507, 509

IIOIP.NET project, 512

Imager Application, 224

implicit objects in EL expressions, 340

import static **syntax**, 14

IN parameters

- CallableStatement** interface, 293
- PreparedStatement** interface, 289–291, 296–297

index file for JAR files, 634–635

indexOf() **function**, **JSTL**, 341

InetAddress **class**, 481

info() **method**, **Logger** **class**, 33

inheritance, 114

inheritance loop, 117–119

Inherited **class**, 18

init() **method**

- AlgorithmParameterGenerator** **class**, 598
- AlgorithmParameters** **class**, 597–598
- in applets, 637
- Cipher** **class**, 603
- KeyGenerator** **class**, 608
- Mac** **class**, 611
- initialize()** **method**, **KeyPairGenerator** **class**, 594
- initParam** **implicit object**, 340
- initSign()** **method**, **Signature** **class**, 589
- initVerify()** **method**, **Signature** **class**, 589–590
- INOUT parameters**, **CallableStatement** **interface**, 293, 297
- insensitive result sets**, 298
- insets** **variable**, **GridLayout** **manager**, 178
- Installation Wizard** **example**, 215–221
- Interceptors**, in **WebWork** framework, 372–373
- Interface Definition Language (IDL)**, 507–509
- interfaces**. *See also specific interfaces*
 - creating, 117
 - incompatible, communication between, 119–122
 - parameterized, 3–4
- Internet InterORB Protocol (IIOP)**, 503, 506–507, 509
- interprocess communication**, 479–480. *See also communication between components*
- intValue()** **method**, **Level** **class**, 38
- Inversion of Control (IoC)**, 134, 369–371
- invocation protocol for JSP 2.0**, 337–339
- Invoker** **class**, **Command pattern**, 131–134
- IoC (Inversion of Control)**, 134, 369–371
- IOP API**, overriding in endorsed directory, 624
- isAssignableFrom()** **function**, 433
- isCertificateEntry()** **method**, **KeyStore** **class**, 596–597
- isCurrent()** **method**, **Refreshable** **interface**, 615
- isDestroyed()** **method**, **Destroyable** **interface**, 615
- isInstanceOf()** **function**, 433
- isKeyEntry()** **method**, **KeyStore** **class**, 596–597
- isLoggable()** **method**
 - Filter** **interface**, 48
 - Handler** **class**, 39
 - Logger** **class**, 33
 - MemoryHandler** **class**, 44
 - StreamHandler** **class**, 41
- isReadOnly()** **method**, **Subject** **class**, 613
- isSameObject()** **function**, 429
- isUserNode()** **method**, **Preference** **class**, 65
- Iterable** **interface**, 8–9
- Iterator** **interface**, 9
- iterators**, for **loop enhancements** for, 7–9

J

JAAS (Java Authentication and Authorization Service)

- authenticating a subject, 615, 616–617
- authorization, 617–618

AuthPermission class, 617–618
 configurations for authentication, 615–616
 credentials, 613, 615
 definition of, 612
 Destroyable interface, 615
 executing code with security checks, 613–617
 LoginContext class, 615
 policy file for authorization, 617–618
 principals, 614
 PrivilegedAction interface, 613
 Refreshable interface, 615
 Subject class, 612–613
 user identification, 612–613

Jakarta Commons Net package, 499
Jakarta POI libraries, 198, 207
The Jakarta Project (Apache), 499
jar command, 625–628, 635

JAR files
 applets in, 629–630
 in classpaths, managing, 619–624
 creating, 94, 625–627, 658
 definition of, 625
 digitally signing, 625, 630–634
 executable, 635–636
 extracting contents of, 628
 index file for, 634–635
 installing as an extension, 620
 manifest file for, 625, 628–629
 viewing contents of, 628

JAR tool (jar command), 625–628, 635
jarsigner utility, 630, 632–634

Java
 learning, using patterns for, 112–113
 new features, list of, 1–2
 utility libraries, list of, 26

Java 2D classes, 143, 171–173
Java Activation Framework, 534
Java API for XML Binding (JAXB)
 classes used in, 269
 of configuration data
 example of, 272–278
 generating classes from XML schema, 263–268
 sample XML document for, 257–259
 XML schema for, 259–263
 definition of, 223, 256–257
 format for, 257
 future direction of, 279
 generating classes from XML schema, 263–268
 marshalling and unmarshalling XML data, 269–271
 tying into applications, 271–278

when to use, 278–279
 XML schema for, 259–263

Java Archive Tool (jar command), 625–628, 635
Java archives. See JAR files

Java Authentication and Authorization Service (JAAS)
 authenticating a subject, 615, 616–617
 authorization, 617–618
 AuthPermission class, 617–618
 configurations for authentication, 615–616
 credentials, 613, 615
 definition of, 612
 Destroyable interface, 615
 executing code with security checks, 613–617
 LoginContext class, 615
 policy file for authorization, 617–618
 principals, 614
 PrivilegedAction interface, 613
 Refreshable interface, 615
 Subject class, 612–613
 user identification, 612–613

Java character classes, 56–57

Java components, communication between
 EJBs (Enterprise JavaBeans)
 containers, 465, 467–468
 definition of, 465
 deployment descriptor for, 473–475
 in EAR (Enterprise Archive), 644–646
 JNDI and, 467
 loan calculator example of, 468–475
 packaging, 643
 types of, 466

RMI (Remote Method Invocation)
 architecture of, 446–447
 communication transport protocol for, 447
 CORBA compatibility with, 510–512
 definition of, 445–446, 465, 500–501
 developing applications with, 448–449
 distributed objects, 445, 504–505
 dynamic class loading, 449
 garbage collection by, 447, 449
 marshalling and unmarshalling, 501–503
 network failure and, 447
 performance of, 447
 protocols, 503
 Remote Object Activations, 449–453
 RMIChat example using, 453–465
 security and, 447
 serialization and, 235
 stubs, generating, 448
 threading, 448

Java Cryptography Architecture (JCA)

- algorithm management, 597–598
- certificate management, 600–602
- definition of, 583, 584
- digital key creation, 592–596
- digital key storage and management, 596–597
- digital signing and verification, 588–592
- engine classes in, 584–585
- message digests, calculating and verifying, 586–588
- provider packages for, alternatives, 585
- random number generation, 599–600
- SUN provider package for, 584

Java Cryptography Extension (JCE)

- Cipher class, 603–608
- converting keys between transparent and opaque, 608–609
- definition of, 583, 602
- encrypting and decrypting data, 603–604
- encrypting serializable classes, 609–611
- generating secret keys, 608
- KeyGenerator class, 608
- message authentication codes, computing, 611–612
- SealedObject class, 609–611
- SecretKeyFactory class, 608–609
- services provided by, 602–603
- wrapping and unwrapping keys, 604–608

Java Development Kit (JDK). *See also specific APIs*

- new features in 5.0 release, list of, 1–2
- utility libraries in, list of, 26

Java Foundation Classes (JFC), 143–144

Java I/O classes, sockets and, 482

Java logging

- definition of, 26–27
- ErrorManager class (handling errors), 49
- examples of, 49–53
- Filter interface (filtering log records), 48
- Formatter class (formatting log records), 44–48
- Handler class (receive and publish log records), 38–44
- Level class (levels of messages), 37–38
- Logger class (log a message), 27, 30–34
- LogManager class (managing logging system), 28–30
- LogRecord class (encapsulate a log message), 34–37

Java Management Extensions (JMX)

- definition of, 543, 548–549
- MBeans, 548–551

Java Message System (JMS)

- definition of, 543, 544
- messages, sending and receiving, 545–548
- messages, types of, 552–553
- version 1.1 specification, new features in, 558

Java Naming and Directory Interface (JNDI)

- accessing EJB containers using, 467
- accessing RMI registry using, 504
- connecting to JMS with, 556–557
- storing data source name using, 286–287

Java Native Interface (JNI)

- arrays in, 410–416
- data type conversions, 406
- e-mail client example using
 JNIMailBridge class, 436–439
- MAPI routines used in, 439–444
- system design for, 434
- user interface for, 435
- fields, accessing in, 416–419
- header file for native code, creating, 403–404
- invoking native routines, 405
- Java code invoking native routines, creating, 402–403
- Java exceptions, handling in native code, 423–424
- Java objects, using in C/C++, 416–423, 425–429
- Java threading, 429–430
- methods, invoking in, 419–423
- native methods, registering manually, 430–432
- native routine library, creating, 404–405
- native routines, using in Java code, 401–405
- NIO direct buffers, using in, 430
- reflection using, 432–434
- strings in, 406–410

Java Network Launch Protocol (JNLP), 647, 648–650, 654

Java preferences

- definition of, 63
- examples of, 69–71
- exporting to XML, 68–71
- Preference class, 63–68

Java Remote Method Protocol (JRMP), 503

Java Runtime Environment (JRE)

- endorsed directory in, 624
- installing JAR files in, 585, 620

Java Serialization API

- classes used in, 229–230
- compared to XMLEncoder/Decoder API, 248–249
- of configuration data, 230–235, 244–245
- customizing, 243–245

- definition of, 223, 228
 - extending, 245
 - Externalizable interface, 229–230, 245
 - file format used by, 228
 - ObjectInputStream class, 229
 - ObjectOutputStream class, 229
 - omitting fields from, 243
 - procedure for, 229–230
 - Serializable interface, 229–230
 - serialization with, 229–235
 - time-based license example
 - definition of, 235–236
 - implementing license, 236–238
 - implementing timeserver, 238–239
 - transient keyword, 243
 - tying into applications, 239–242
 - versioning and, 245–247
 - when to use, 247
- Java Server Page (JSP) 2.0**
- benefits of, 351
 - code reuse support, 335–336
 - EL (Expression Language) support, 332–335, 339–340
 - example of, 350–363
 - forum example using
 - architecture of, 320
 - code for, 324–327
 - database for, 321
 - file structure for, 321–322
 - user interface for, 322–323
 - invocation protocol for, 337–339
 - page extensions, 336–337
 - Servlet 2.4 support, 332
- Java Socket API**
- classes in, 481
 - client programming using, 481–482
 - definition of, 481
 - example using, 483–487
 - server programming using, 482–483
- Java Standard Template Library (JSTL) 1.1**
- CMT (Contact Management Tool) application, 340
 - example of, 344–350
 - Function Tag Library, 341–342
 - SQL Actions, 342–344
- Java threading. See threading**
- Java 2 Enterprise Edition (J2EE)**
- architecture of, 87
 - definition of, 113
 - middleware and, 504–505
 - when to use, compared to CORBA, 512
- Java Web Services Development Pack (JWSDP), 263, 534**
- Java Web Start**
- definition of, 647, 654
 - TicTacToe example of
 - definition of, 647–648
 - JNLP file for, 648–650
 - TTTGui class, 653–654
 - TTTLogic class, 650–653
 - TTTMain class, 650
- java.awt library, 143**
- JavaBean components. See also EJBs**
- RowSets support for, 308–309
 - serializing with XMLEncoder/Decoder API, 248–256
- Javadoc API (doclet API), annotations, 19–20**
- javah tool, 403**
- java.sql package, 283**
- Java2WSDL toolset, 533**
- java.util.logging library. See Java logging**
- java.util.prefs library. See Java preferences**
- java.util.regex library. See regular expressions**
- javax.sql package, 283**
- JAXB (Java API for XML Binding)**
- classes used in, 269
 - of configuration data
 - example of, 272–278
 - generating classes from XML schema, 263–268
 - sample XML document for, 257–259
 - XML schema for, 259–263
 - definition of, 223, 256–257
 - format for, 257
 - future direction of, 279
 - generating classes from XML schema, 263–268
 - marshalling and unmarshalling XML data, 269–271
 - tying into applications, 271–278
 - when to use, 278–279
 - XML schema for, 259–263
- JAXBContext class, 269**
- JBoss JMX Agent, 563–564, 575, 580**
- JBoss: Professional Open Source Web site, 499**
- JButton class**
- CardLayout manager using, 192, 194
 - FlowLayout manager using, 162
 - SpringLayout manager using, 184

JCA (Java Cryptography Architecture)

- algorithm management, 597–598
- certificate management, 600–602
- definition of, 583, 584
- digital key creation, 592–596
- digital key storage and management, 596–597
- digital signing and verification, 588–592
- engine classes in, 584–585
- message digests, calculating and verifying, 586–588
- provider packages for, alternatives, 585
- random number generation, 599–600
- SUN provider package for, 584

JCE (Java Cryptography Extension)

- Cipher class, 603–608
- converting keys between transparent and opaque, 608–609
- definition of, 583, 602
- encrypting and decrypting data, 603–604
- encrypting serializable classes, 609–611
- generating secret keys, 608
- KeyGenerator class, 608
- message authentication codes, computing, 611–612
- SealedObject class, 609–611
- SecretKeyFactory class, 608–609
- services provided by, 602–603
- wrapping and unwrapping keys, 604–608

JDBC API

- connections
 - managing, 286–287
 - pooling, 310
- driver types for, 282
- features of, 281–282
- meta data for data source, retrieving, 302–308
- packages in, 283
- requirements for, 283
- result sets for SQL queries, 298–302
- RowSets, using, 308–309
- SQL statements, executing
 - batch updates, 294–297
 - CallableStatement interface for, 292–294
 - PreparedStatement interface for, 289–292
 - Statement interface for, 288–289
- three-tier model for, 284–285
- transactions, managing, 310–312
- two-tier model for, 283–284

JDBC-Net Pure Java Driver, 282

JDBC-ODBC Bridge Driver, 282

JdbcRowSetImpl RowSet implementation, 309

JDialog class

- Annotation Editor example using
 - AnnotationEditor class, 198–200
 - ComboListener class, 213–214
 - ComponentListener class, 202–204
 - ExcelAction class, 207–209
 - HighlightPainter class, 204–205
 - OpenAction class, 209–212
 - PopupListener class, 201–202
 - PrintAction class, 205–207
 - XmlAction class, 207
- definition of, 197

JDK (Java Development Kit). See also *specific APIs*

- new features in 5.0 release, list of, 1–2
- utility libraries in, list of, 26

JFC (Java Foundation Classes), 143–144

JFileChooser class, 209

JFrame class

- Annotation Editor example using
 - AnnotationEditor class, 198–200
 - ComboListener class, 213–214
 - ComponentListener class, 202–204
 - ExcelAction class, 207–209
 - HighlightPainter class, 204–205
 - OpenAction class, 209–212
 - PopupListener class, 201–202
 - PrintAction class, 205–207
 - XmlAction class, 207
- BorderLayout manager and, 144, 145
- definition of, 197

JKS format, 596

JLabel class, 162, 179, 184, 192

JMenuBar class, 204–205

JMeter tool, development scenarios using, 107–109

JMS (Java Message System)

- definition of, 543, 544
- messages, sending and receiving, 545–548
- messages, types of, 552–553
- version 1.1 specification, new features in, 558

JMS server, 573–575

JMSEException class, JMS, 558

JMX Agent, 563–564, 575, 580

JMX (Java Management Extensions)

- definition of, 543, 548–549
- MBeans, 548–551

JNDI (Java Naming and Directory Interface)

- accessing EJB containers using, 467
- accessing RMI registry using, 504

- connecting to JMS with, 556–557
 - storing data source name using, 286–287
 - JNI (Java Native Interface)**
 - arrays in, 410–416
 - data type conversions, 406
 - e-mail client example using
 - JNIMailBridge class, 436–439
 - MAPI routines used in, 439–444
 - system design for, 434
 - user interface for, 435
 - fields, accessing in, 416–419
 - header file for native code, creating, 403–404
 - invoking native routines, 405
 - Java code invoking native routines, creating, 402–403
 - Java exceptions, handling in native code, 423–424
 - Java objects, using in C/C++, 416–423, 425–429
 - Java threading, 429–430
 - methods, invoking in, 419–423
 - native methods, registering manually, 430–432
 - native routine library, creating, 404–405
 - native routines, using in Java code, 401–405
 - NIO direct buffers, using in, 430
 - reflection using, 432–434
 - strings in, 406–410
 - JNIMailBridge **class**, **434, 436–439**
 - JNLP (Java Network Launch Protocol)**, **647, 648–650, 654**
 - join() **function**, **JSTL**, **341**
 - JoinRowSetImpl RowSet implementation**, **309**
 - JOptionPane **class**, **213**
 - JORAM JMS Server**, **573–575**
 - JPanel **class**, **192, 215–216, 218**
 - JRE (Java Runtime Environment)**
 - endorsed directory in, 624
 - installing JAR files in, 585, 620
 - JRMP (Java Remote Method Protocol)**, **503**
 - JSP (Java Server Page) 2.0**
 - benefits of, 351
 - code reuse support, 335–336
 - EL (Expression Language) support, 332–335, 339–340
 - example of, 350–363
 - forum example using
 - architecture of, 320
 - code for, 324–327
 - database for, 321
 - file structure for, 321–322
 - user interface for, 322–323
 - invocation protocol for, 337–339
 - page extensions, 336–337
 - Servlet 2.4 support, 332
 - JSpinner **class**, **184, 190**
 - .jspx **file extension**, **336–337**
 - JSTL (Java Standard Template Library) 1.1**
 - CMT (Contact Management Tool) application, 340
 - example of, 344–350
 - Function Tag Library, 341–342
 - SQL Actions, 342–344
 - JTextArea **class**, **184, 190, 199–200**
 - JTextField **class**
 - CardLayout manager using, 192
 - FlowLayout manager using, 162
 - SpringLayout manager using, 184, 190
 - JTree **class**, **202–203**
 - J2EE (Java 2 Enterprise Edition)**
 - architecture of, 87
 - definition of, 113
 - middleware and, 504–505
 - when to use, compared to CORBA, 512
 - JUnit tool**
 - development scenarios using, 98–101
 - learning, using patterns for, 113
 - JWSDP (Java Web Services Development Pack)**, **263, 534**
- ## K
- key agreement**, **602**
 - Key interface**, **592–593**
 - KeyFactory **class**, **JCA**, **585, 593–594**
 - KeyGenerator **class**, **JCE**, **608**
 - KeyPair **class**, **JCA**, **594**
 - KeyPairGenerator **class**, **JCA**, **585, 594–596**
 - keys, digital. See also encryption**
 - creating, 592–596
 - definition of, 588
 - storing and managing, 596–597
 - keys() **method**, **Preference class**, **65**
 - KeySpec **interface**, **592**
 - keystore**, **596–597, 631–632**
 - KeyStore **class**, **JCA**, **585, 596–597**
 - keytool utility**, **631–632**
 - King, Gavin (Hibernate)**, **374**
 - King's Hibernate example application**, **374**

L

Larman, Craig (*Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*), **83, 111, 661**

Last() **method**, ResultSet **class**, **300**

layout managers

- BorderLayout manager, 144–151
- BoxLayout manager, 151–161
- CardLayout manager, 191–197, 214
- FlowLayout manager, 161–166
- GridBagLayout manager, 177–183
- GridLayout manager, 167–177
- list of, 144
- SpringLayout manager, 183–191

lazy operators, in regular expressions, **57–58**

Leaf **class**, Composite **pattern**, **139–140**

length() **function**, JSTL, **341**

Level **class**, **30, 37–38**

.level **property**, **28**

libraries, utility. See **Java logging; Java preferences; regular expressions**

license, time-based, for applications

- definition of, 235–236
- implementing license, 236–238
- implementing timeserver, 238–239

linefeed, meta-character for, **55**

<listener> **element**, WAR **deployment descriptor**, **641**

Listeners, JMeter, **108**

load() **method**, KeyStore **class**, **596**

loan calculator example of EJBs

- definition of, 468
- EJB deployment descriptor for, 473–475
- LoanBean class, 469–470
- LoanClient class, 470–473
- LoanHome interface, 468–469
- LoanObject interface, 468

local references to objects, **425–427**

log() **method**, Logger **class**, **33**

log4j-1.2.8.jar **file**, **315**

Logger **class**, **27, 30–34**

login() **method**, LoginContext **class**, **615, 617**

logging. See **Java logging**

LoggingInterceptor, **WebWork framework**, **373**

<login-config> **element**, WAR **deployment descriptor**, **641**

LoginContext **class**, JAAS, **615, 616–617**

LoginModule **class**, JAAS, **616**

LogManager **class**, **28–30**

logout() **method**, LoginContext **class**, **617**

logp() **method**, Logger **class**, **34**

logrb() **method**, Logger **class**, **34**

LogRecord **class**, **34–37**

long-term serialization. See **XMLEncoder/Decoder API**

lookingAt() **method**, Matcher **class**, **60**

loose-coupling design principle, **116**

low coupling, **114**

M

MAC (Message Authentication Code), **583, 602, 611–612**

Mac **class**, JCE, **611–612**

manifest file, **625, 628–629**

MapMessage, JMS, **552**

Marshaller **class**, **269**

marshalling and unmarshalling

- in RMI, 501–503
- XML data, 269–271, 448

Matcher **class**, **54, 59–61**

matcher() **method**, Pattern **class**, **59**

matches() **method**

- Matcher class, 60
- Pattern class, 58

MatchResult **interface**, **61**

Maven tool, development scenarios using, **95–98**

MBeans

- definition of, 548
- deploying, 550–551
- JMX Agent and, 580
- naming conventions for, 554
- using, 549–550

MBeanServer

- adaptors for, 551
- definition of, 549
- registering MBeans with, 550–551

MD5 message digest algorithm, **584, 586**

member variables, accessing in JNI, **416–419**

memory, writing log messages to, **43–44**

MemoryHandler **class**, **43–44**

MemoryHandler() **method**, MemoryHandler **class**, **44**

Message Authentication Code (MAC), **583, 602, 611–612**

Message **class**, JMS, **545, 558**

message digests, JCA, **585, 586–588**

Message Oriented Middleware (MOM), 543

`MessageConsumer` class, **JMS, 545, 558**

`MessageDigest` class, **JCA, 585, 586–588**

message-driven beans, 466–467

`MessageListener` class, **JMS, 545, 555, 558**

`MessageProducer` class, **JMS, 558**

`MessagePublisher` class, **JMS, 545**

messages (e-mail), e-mail client example

`JNIMailBridge` class, 436–439

MAPI routines used in, 439–444

system design for, 434

user interface for, 435

messages (JMS)

flow of, in distributed processing, 552

processing, 553–564

routing, 564–566

sending and receiving, 544–548

splitting, 567–570

types of, 552–553

messages (logging)

encapsulating for logging system, 34–37

filtering, 48

formatting, 44–48

levels of, 30, 37–38

logging, 30–34

writing

to external destinations, 38–44

to a file, 42–43

to memory buffer, 43–44

to network, 41–42

to output stream, 40–41

to `System.err`, 41

Message-Style Web services, 536

messaging, overriding in endorsed directory, 624

messaging systems. See JMS

meta data

for data source, retrieving with JDBC API, 302–308

definition of, 2, 17–18

examples of, 19–20, 22–26

interfaces in doclet API for, 20–22

types of, 18–19

meta-characters in regular expressions, 55–57

META-INF directory

digitally signing entries in, 630

EAR descriptor file in, 644–645

EJB deployment descriptor in, 643

JAR index file in, 635

manifest file in, 625

methodologies for software development, 82–87

methods. See also specific methods

boxing and unboxing conversions in, 12–13

exposing to other applications with JMX, 543, 548–551

generic, 6–7

invoking with JNI, 419–423

native, registering manually, 430–432

static, importing, 13–15

variable arguments for, 2, 9–10

Microsoft .NET platform, CORBA and, 512

middle layer, three-tier model, JDBC API, 284

middleware, 504–505

<mime-mapping> element, **WAR deployment descriptor, 641, 642**

Mine, Philip (article about custom persistence delegates), 255

M-Let descriptor file, 579–581

M-Let Service

benefits of, 578–579

definition of, 578

deploying, 579

deployment descriptor for, 579–581

Model, in MVC, 366–367

`Model` class, **Model-View-Controller pattern, 124–125**

Model 1 Architecture

definition of, 329–331

EL features relevant to, 339–340

JSP 2.0 example using, 351–363

JSP 2.0 features relevant to, 331–339

JSTL example using, 344–350

JSTL features relevant to, 340–344

when to use, 330

Model 2 Architecture

benefits of, 367

configuring and deploying applications, 394–397

definition of, 365–367

domain model, defining, 378–384

hanging sessions, preventing, 375–377

Hibernate, extending framework with, 374–377

IoC (Inversion of Control), 369–371

modifying applications, 397–399

use cases, implementing, 384–387

views, developing, 387–394

WebWork framework for, 368–374

when to use, 367–368

`ModelDrivenInterceptor`, **WebWork framework, 373**

modeling, 75

Model-2 pattern (Model-View-Controller pattern)

Model-2 pattern (Model-View-Controller pattern)

- application data using, 224
- definition of, 122
- example of
 - controller, 128–130
 - Model class, 124–125
 - scenarios for, 123–124
 - view component, 125–127

Model-View-Controller (MVC) pattern

- application data using, 224
- definition of, 122
- example of
 - controller, 128–130
 - Model class, 124–125
 - scenarios for, 123–124
 - view component, 125–127

MOM (Message Oriented Middleware), 543

`MonitorEnter()` **function, 429**

`MonitorExit()` **function, 429–430**

`moveToInsertRow()` **method, ResultSet class, 301**

multithreading, using JNI, 429–430

MVC (Model-View-Controller) pattern

- application data using, 224
- definition of, 122
- example of
 - controller, 128–130
 - Model class, 124–125
 - scenarios for, 123–124
 - view component, 125–127
- Model 2 Architecture and, 365–367

N

`name()` **method, Preference class, 65**

`NamingComponent` **class, 517**

`NamingContext` **class, 517**

Native API/Part Java Driver, 282

native code, using in Java programs with JNI

- arrays in, 410–416
- data type conversions, 406
- e-mail client example using
 - `JNIMailBridge` class, 436–439
 - MAPI routines used in, 439–444
 - system design for, 434
 - user interface for, 435
- fields, accessing, 416–419
- header file for native code, creating, 403–404
- invoking native routines, 405

- Java code invoking native routines, creating, 402–403
- Java exceptions, handling in native code, 423–424
- Java objects, using in C/C++, 416–423, 425–429
- Java threading, 429–430
- methods, invoking in, 419–423
- native methods, registering manually, 430–432
- native routine library, creating, 404–405
- native routines, using in Java code, 401–405
- NIO direct buffers, using in, 430
- reflection using, 432–434
- strings in, 406–410

Native-Protocol Pure Java Driver, 282

navigation in Swing applications

- Installation Wizard example of, 215–221
- patterns used for, 214–215

network

- architecture of, 479–480
- failure of, in RMI applications, 447
- writing log messages to, 41–42

`NewDirectByteBuffer()` **function, 430**

`NewGlobalRef()` **function, 427**

newline, meta-character for, 55

`NewLocalRef()` **function, 425**

`NewObjectArray()` **function, 411**

news reader example, 478

`NewString()` **function, 407**

`NewStringUTF()` **function, 407**

`New[Type]Array()` **function, 412**

`NewWeakGlobalRef()` **function, 427**

`Next()` **method, ResultSet class, 300**

`nextBytes()` **method, SecureRandom class, 599**

NIO classes, sockets and, 482

NIO direct buffers, using in JNI, 430

`node()` **method, Preference class, 64**

`nodeExists()` **method, Preference class, 64**

nonscrollable result sets, 298

O

Object Graph Navigation Language (OGNL), 373–374

object graphs

- definition of, 225
- example of, 226–227
- serializing with Java Serialization API
 - classes used in, 229–230
 - compared to XMLEncoder/Decoder API, 248–249
 - of configuration data, 230–235, 244–245

- customizing, 243–245
 - definition of, 223, 228
 - extending, 245
 - file format used by, 228
 - omitting fields from, 243
 - procedure for, 229–230
 - time-based license example, 235–239
 - tying into applications, 239–242
 - versioning and, 245–247
 - when to use, 247
 - serializing with XMLEncoder/Decoder API, 248–256
 - Object Management Group (OMG), 505**
 - Object Request Broker (ORB), 506–507, 509**
 - Object to Relational Mapping (ORM), 312. See also Hibernate tool**
 - `ObjectInputStream` class, 229
 - ObjectMessage, JMS, 552–553**
 - object-oriented concepts, patterns and, 114**
 - `ObjectOutputStream` class, 229
 - objects. See also classes**
 - arrays of, in JNI, 411
 - collections of, treating as one, 138–142
 - distributed, 445, 504–505
 - references for, using in C/C++, 425–429
 - Remote Object Activations, 449–453
 - sealing, 609–611
 - using, in C/C++, 416–423
 - Observer Design pattern, 123**
 - octal values, meta-characters for, 55**
 - ODBC, and JDBC, 282**
 - `odmg-3.0.jar` file, 315
 - OGNL (Object Graph Navigation Language), 373–374**
 - OMG (Object Management Group), 505**
 - opaque representations of keys, 592, 593–594, 597–598, 608–609**
 - Open Systems Interconnection (OSI) architecture, 479**
 - `OpenAction` class, Annotation Editor example, 209–212
 - OpenSymphony Quality Components, 499**
 - operators, in regular expressions, 57–58**
 - ORB (Object Request Broker), 506–507, 509**
 - `orbd` command, 521
 - ORM (Object to Relational Mapping), 312. See also Hibernate tool**
 - OSI (Open Systems Interconnection) architecture, 479**
 - OUT parameters, CallableStatement interface, 293, 297**
 - `Overrides` class, 18, 19
- P**
- packaging Java applications. See deploying Java applications**
 - page extensions in JSP 2.0, 336–337**
 - page-centric approach of Model 1 Architecture, 329**
 - pageContext implicit object, 340**
 - pageScope implicit object, 340**
 - `paint()` method, in applets, 637
 - Param implicit object, 340**
 - parameterized types, 3–4**
 - parameters, arbitrary number of (variable arguments), 2, 9–10**
 - `ParametersInterceptor`, WebWork framework, 373
 - paramValues implicit object, 340**
 - `parent()` method, Preference class, 65
 - parentheses (()), in regular expressions, 58**
 - `parse()` method, Level class, 38
 - Password-Based Encryption (PBE), 602**
 - `Pattern` class, 54, 58–59
 - pattern matching. See regular expressions**
 - `pattern()` method
 - Matcher class, 60
 - Pattern class, 59
 - patterns. See also software design and development**
 - Adapter pattern, 119–122
 - books about, 111
 - building, with design principles, 115–119
 - Command pattern, 130–134
 - Composite pattern, 138–142
 - definition of, 112
 - importance of, 112–114
 - Model-View-Controller (MVC) pattern, 122–130
 - Observer Design pattern, 123
 - Strategy pattern, 134–138
 - PBE (Password-Based Encryption), 602**
 - `PBEKey` interface, 593
 - percent sign (%), prefixing filename patterns, 42–43**
 - performance**
 - batch updates and, 294
 - connection pooling and, 286, 310
 - of enumerations, 15
 - of JDBC three-tier model, 284
 - measuring with JMeter, 107–109
 - ORM tools and, 375
 - PreparedStatement interface and, 289
 - result sets and, 298, 299
 - of RMI applications, 447, 448
 - of software development, 80–81
 - of Web services, 445

period (.), in regular expressions, 55, 56

persistence delegates, 255

persisting applications. See database, persisting applications with; serialization

Person class, Strategy pattern, 137

PKCS#8 format, 596

PKIX LDAP V2 Schema, certificate store using, 584

Plain Old Java Object (POJO), 370

plus sign (+), in regular expressions, 57, 58

POA class, 517

POJO (Plain Old Java Object), 370

policy file, 617–618

polymorphism, 114

POM (Project Object Model), 95

PopLocalFrame() function, 426, 427

PopupListener class, Annotation Editor example, 201–202

PortableInterceptor API, overriding in endorsed directory, 624

PortableServer API, overriding in endorsed directory, 624

PortableServer.POA class, 517

portal example, 478

POSIX character classes, 56–57

POST command, HTTP, 489

Post-Processors, JMeter, 108

Preference class

creating/deleting nodes, 63–65

definition of, 63

events, 67

exporting nodes, 67–68

removing nodes or values of nodes, 67–68

retrieving node information, 65

retrieving nodes, 63–65

retrieving values from nodes, 65–66

setting values on nodes, 66

preferences, Java

definition of, 63

examples of, 69–71

exporting to XML, 68–71

Preference class, 63–68

PreparedStatement interface, JDBC API, 289–292, 296–297

Pre-Processors, JMeter, 108

presentation layer, SQL Actions used in, 342

Previous() method, ResultSet class, 300

primitive types

arrays of, 410, 411–412

boxing conversions for, 11

JNI field descriptors for, 417

translating to C++, 406

unboxing conversions for, 12

Principal interface, JAAS, 614

PrintAction class, Annotation Editor example, 205–207

private key, 588

PrivateKey interface, 593

PrivilegedAction interface, JAAS, 613

programs. See applications; software design and development

Project Object Model (POM), 95

property file

in Ant build files, 655

delimiter in, 629

proprietary protocols, 498

protected variations, 114

protocol. See also specific protocols

definition of, 479

existing, using, 499

implementing with sockets, 487–498

proprietary, 498

RMI support for, 503

protocol layer, 479–480

prototypes (function signatures), 405

pseudo-random number generation, 599–600

public key, 588

PublicKey interface, 593

publish() method

Handler class, 39

MemoryHandler class, 44

SocketHandler class, 41

StreamHandler class, 41

push() method, MemoryHandler class, 44

PushLocalFrame() function, 426, 427

PUT command, HTTP, 489

put() method, Preference class, 66

putBoolean() method, Preference class, 66

putByteArray() method, Preference class, 66

putDouble() method, Preference class, 66

putFloat() method, Preference class, 66

putInt() method, Preference class, 66

putLong() method, Preference class, 66

Q

query tag, JSTL, 343

question mark (?), in regular expressions, 57, 58

queue, JMS, 544

quote() method, Pattern class, 58

quoteReplacement() method, Matcher class, 59

R**random number generation, 599–600****raw types, 4–5****RDBMS (Remote Database Management System),
executing stored procedures on, 292–294**`readConfiguration()` **method**, `LogManager`
class, 29`readObject()` **method**, **Java Serialization API, 243–245**`ReadOnlyIterator` **interface, 9****refactoring (changing code design), 77****Refactoring (Fowler, Martin), 77, 111, 661****reference types, 11, 12****reflection, using JNI, 432–434**`refresh()` **method**, `Refreshable` **interface, 615**`Refreshable` **interface, JAAS, 615**`region()` **method**, `Matcher` **class, 61**`regionEnd()` **method**, `Matcher` **class, 61**`regionStart()` **method**, `Matcher` **class, 61**`Register` **class, 452–453**`RegisterNatives()` **function, 430–431****regular expressions**

backreferences in, 58

character classes in, 56–57

definition of, 53

examples of, 53–54, 61–63

groups of matching characters in, 58

`Matcher` class (compare pattern to string), 59–61

meta-characters for, 55–57

operators for, 57–58

`Pattern` class (compiling and storing regular
expressions), 58–59`relative()` **method**, `ResultSet` **class, 300**`ReleasePrimitiveArrayCritical()` **function, 413**`ReleaseStringChars()` **function, 408**`ReleaseStringCritical()` **function, 408**`ReleaseStringUTFChars()` **function, 408, 410**`Release[Type]ArrayElements()` **function,
412, 414****reluctant operators, in regular expressions, 57–58****Remote Database Management System (RDBMS),
executing stored procedures on, 292–294**`Remote` **interface, 502****Remote Method Invocation (RMI)**

architecture of, 446–447

communication transport protocol for, 447

CORBA compatibility with, 510–512

definition of, 445–446, 465, 500–501

developing applications with, 448–449

distributed objects, 445, 504–505

dynamic class loading, 449

garbage collection by, 447, 449

marshalling and unmarshalling, 501–503

network failure and, 447

performance of, 447

protocols, 503

`Remote Object Activations, 449–453`

security and, 447

serialization and, 235

stubs, generating, 448

threading, 448

Remote Object Activations

definition of, 449–450

example of, 450–453

**Remote Procedure Call (RPC). See also RMI; Web
services**

history of, 446

platform independent, Web services as, 526–527

RMI (Remote Method Invocation) and, 500–501

`RemoteFileSystemWatcher` **interface, 513**`remove()` **method**, `Preference` **class, 67**`removeHandler()` **method**, `Logger` **class, 31**`removeNode()` **method**, `Preference` **class, 64**`removeNodeChangeListener()` **method,
Preference **class, 67****`removePreferenceChangeListener()` **method,
Preference **class, 67****`removePropertyChangeListener()` **method,
LogManager **class, 30****`replace()` **function, JSTL, 341**`replaceAll()` **method**, `Matcher` **class, 61**`replaceFirst()` **method**, `Matcher` **class, 61****Request scope, WebWork framework, 374****requestScope** implicit object, 340`requireEnd()` **method**, `Matcher` **class, 60****requirement changes in software development, 75–76**`reset()` **method**`LogManager` class, 29`Matcher` class, 60<resource-ref> **element, WAR deployment
descriptor, 641**`ResultSet` **class**

concurrency of, 298

definition of, 298

encapsulated by `JdbcRowSetImpl`, 309

holdability of, 299

serialization and, 229–230

types of, 298

using, 299–302

Retention class

Retention **class**, **18, 19**

RetentionPolicy **enumeration**, **18**

reuse

code, 335–336

framework, 367

reverse containment relationship, **118**

reverse engineering for proprietary protocols, **498**

RMI (Remote Method Invocation)

architecture of, 446–447

communication transport protocol for, 447

CORBA compatibility with, 510–512

definition of, 445–446, 465, 500–501

developing applications with, 448–449

distributed objects, 445, 504–505

dynamic class loading, 449

garbage collection by, 447, 449

marshalling and unmarshalling, 501–503

network failure and, 447

performance of, 447

protocols, 503

Remote Object Activations, 449–453

security and, 447

serialization and, 235

stubs, generating, 448

threading, 448

RMI registry, **446, 453, 465, 503–504**

rmic tool, **448, 464**

RMIChat example using RMI

ChatApplet class, 460–464

ChatUser class, 459–460

compiling, 464–465

RMIChat interface, 454–455

RMIChatImpl class, 455–459

running, 465

user interface for, 453–454

rmid activation daemon, **450, 453**

RMI-IIOP, **510–512**

Role **interface**, **Strategy pattern**, **135**

rollback() **method**, **Connection class**, **311**

rolling back transactions, **310–311**

rowChanged event, **RowSet**, **308**

rowSetChanged event, **RowSet**, **308**

RowSets, **308–309**

RPC (Remote Procedure Call). *See also* **RMI; Web services**

history of, 446

platform independent, Web services as, 526–527

RMI (Remote Method Invocation) and, 500–501

RPC-based Web services, **536**

.RSA **file extension**, **630**

RSAMultiplePrimePrivateCrtKeySpec **interface**, **592**

RSAMultiPrimePrivateCrtKeySpec **interface**, **593**

RSAPrivateCrtKeySpec **interface**, **593**

RSAPrivateCrtKeySpec **interface**, **592**

RSAPrivateKey **interface**, **593**

RSAPrivateKeySpec **interface**, **592**

RSAPublicKey **interface**, **593**

RSAPublicKeySpec **interface**, **592**

run() **method**, **PrivilegedAction interface**, **613**

S

saveOrUpdate() **method**, **Session class**, **319**

savepoints for transactions, **311**

scalability

of connection pooling, 310

of EJBs, 445, 465

of JDBC three-tier model, 284

of JDBC two-tier model, 283

of JMS processing architecture, 552

of Model 2 Architecture, 367

of RMI, 445

of Web services, 522, 527

scopes of components, **WebWork framework**, **374**

scripting elements in EL, **333**

scripting tools

Ant application, 87–95, 113, 496, 654–658

Maven tool, 95–98

XDoclet tool, 101–107, 113

scriptlets in EL, **333**

scrollable result sets, **298**

SealedObject **class**, **JCE**, **609–611**

SecretKey **interface**, **593**

SecretKeyFactory **class**, **JCE**, **608–609**

SecureRandom **class**, **JCA**, **585, 599–600**

security

of applets, 639

JAAS (Java Authentication and Authorization Service)

authenticating a subject, 615, 616–617

authorization, 617–618

AuthPermission class, 617–618

configurations for authentication, 615–616

credentials, 613, 615

definition of, 612

Destroyable interface, 615

- executing code with security checks, 613–617
- LoginContext class, 615
- policy file for authorization, 617–618
- principals, 614
- PrivilegedAction interface, 613
- Refreshable interface, 615
- Subject class, 612–613
- user identification, 612–613
- JAR files, digitally signing, 625, 630–634
- of Java Web Start application, 650
- JCA (Java Cryptography Architecture)
 - algorithm management, 597–598
 - certificate management, 600–602
 - definition of, 583, 584
 - digital key creation, 592–596
 - digital key storage and management, 596–597
 - digital signing and verification, 588–592
 - engine classes in, 584–585
 - message digests, calculating and verifying, 586–588
 - provider packages for, alternatives, 585
 - random number generation, 599–600
 - SUN provider package for, 584
- JCE (Java Cryptography Extension)
 - Cipher class, 603–608
 - converting keys between transparent and opaque, 608–609
 - definition of, 583, 602
 - encrypting and decrypting data, 603–604
 - encrypting serializable classes, 609–611
 - generating secret keys, 608
 - KeyGenerator class, 608
 - message authentication codes, computing, 611–612
 - SealedObject class, 609–611
 - SecretKeyFactory class, 608–609
 - services provided by, 602–603
 - wrapping and unwrapping keys, 604–608
- MAC (Message Authentication Code), 583
- of Model 2 Architecture, 367
- of RMI applications, 447
- <security-constraint> **element, WAR deployment descriptor, 641, 642**
- <security-role> **element, WAR deployment descriptor, 641**
- seed value for random number generation, 599**
- Seller **class, Strategy pattern, 136**
- Semantic Web, Web services and, 522**
- sendMail() **method, 439**
- sensitive result sets, 298**
- serializable classes, encrypting, 609–611**
- Serializable **interface, 229–230, 502**
- serialization. See also database, persisting applications with**
 - APIs for, list of, 223
 - of application data, 224–227
 - of configuration data, with JAXB
 - generating classes from XML schema, 263–268
 - sample XML document for, 257–259
 - XML schema for, 259–263
 - definition of, 228
 - with Java Serialization API
 - classes used in, 229–230
 - compared to XMLEncoder/Decoder API, 248–249
 - of configuration data, 230–235, 244–245
 - customizing, 243–245
 - definition of, 223, 228
 - extending, 245
 - file format used by, 228
 - omitting fields from, 243
 - procedure for, 229–230
 - time-based license example, 235–239
 - tying into applications, 239–242
 - versioning and, 245–247
 - when to use, 247
 - with JAXB (Java API for XML Binding)
 - classes used in, 269
 - of configuration data, 272–278
 - definition of, 223, 256–257
 - format for, 257
 - future direction of, 279
 - generating classes from XML schema, 263–268
 - marshalling and unmarshalling XML data, 269–271
 - tying into applications, 271–278
 - when to use, 278–279
 - XML schema for, 259–263
 - with XMLEncoder/Decoder API
 - classes used in, 250–251
 - compared to Java Serialization API, 248–249
 - of configuration data, 252–254
 - customizing, 254–255
 - definition of, 223
 - format for, 249–250
 - persistence delegates and, 255
 - procedure for, 251
 - when to use, 255–256

serialver tool, 246–247

serialVersionUID field, 246–247

server layer, two-tier model, JDBC API, 283

ServerSocket class, 481, 482–483

Service Provider Interface (SPI) for engine classes, 584–585

<servlet> element, WAR deployment descriptor, 641, 642

Servlet 2.4 support in JSP 2.0, 332

<servlet-mapping> element, WAR deployment descriptor, 641

ServletRequest class, 332

session beans, 466

Session class

 Hibernate, 319

 JMS, 545, 558

Session scope, WebWork framework, 374

<session-config> element, WAR deployment descriptor, 641

SessionFactory object, Hibernate, 317

sessionScope implicit object, 340

setAutoCommit() method, Connection class, 311

setBoolean() method, PreparedStatement interface, 290

setCertificateEntry() method, KeyStore class, 597

setDataSource tag, JSTL, 343

setDate() method, PreparedStatement interface, 290

setDouble() method, PreparedStatement interface, 290

setEncoding() method

 Handler class, 39

 StreamHandler class, 41

setErrorManager() method, Handler class, 39

setFilter() method, Handler class, 39

setFloat() method, PreparedStatement interface, 290

setFormatter() method, Handler class, 39

setInt() method, PreparedStatement interface, 290

setKeyEntry() method, KeyStore class, 597

setLevel() method

 Handler class, 39

 Logger class, 31

 LogRecord class, 36

setLoggerName() method, LogRecord class, 36

setLong() method, PreparedStatement interface, 290

setMessage() method, LogRecord class, 36

setMillis() method, LogRecord class, 37

setObject() method, PreparedStatement interface, 291–292

SetObjectArrayElement() function, 411

setOutputStream() method, StreamHandler class, 41

setParameters() method, LogRecord class, 37

setParent() method, Logger class, 31

setPushLevel() method, MemoryHandler class, 44

setReadOnly() method, Subject class, 613

setResourceBundle() method, LogRecord class, 36

setResourceBundleName() method, LogRecord class, 36

setSavePoint() method, Connection class, 311

setSeed() method, SecureRandom class, 599

setSequenceNumber() method, LogRecord class, 37

setSourceClassName() method, LogRecord class, 36

setSourceMethodName() method, LogRecord class, 36

SetStatic[Type]Field() function, 418

setString() method, PreparedStatement interface, 290

setThreadID() method, LogRecord class, 36

setThrown() method, LogRecord class, 36

Set[Type]ArrayRegion() function, 413

Set[Type]Field() function, 418, 419

setUseParentHandlers() method, Logger class, 31

severe() method, Logger class, 33

.SF file extension, 630

SHA-1 message digest algorithm, 584, 586, 588

SHA1PRNG pseudo-random number generator, 584, 596

sign() method, Signature class, 589

Signature class, JCA, 585, 589–592

signature-related files, 630

signatures, digital

 for data, 588–592

 for JAR files, 625

Simple Object Access Protocol (SOAP), 445, 529–530, 639

SimpleFormatter class, 45

Singleton pattern, 215

slash (/)

 directory separator, 42

 in preference nodes, 63

SOAP (Simple Object Access Protocol), 445, 529–530, 639

Socket **class**, 481–482

SocketHandler **class**, 41–42

sockets

definition of, 480

history of, 446

Java Socket API, 481–487

protocol, implementing, 487–499

types of, 480

software design and development. See also Model 1

Architecture; Model 2 Architecture; performance; scalability

Ant, development scenarios using, 87–95

bug reporting and tracking, 81

code reuse, in JSP 2.0, 335–336

communication, importance of, 75

configuration management, 78–79

continuous integration, 79

criteria for, 114

discipline required for, 76

estimates for, 80–81

guidelines for, 75–81

JMeter, development scenarios using, 107–109

JUnit, development scenarios using, 98–101

Maven, development scenarios using, 95–98

methodologies for, 82–87

modeling, 75

new technologies, learning, 78

patterns and, 113–114

principles for, building patterns with, 115–119

process for, building, 78

quality of, measuring, 74

refactoring (changing code design), 77

requirement changes, handling, 75–76

short development iterations, 79–80

traceability of code to requirements, 76–77

unit testing, 79

writing code, importance of, 77

XDoclet, development scenarios using, 101–107

source code control, 78–79

SourceForge Web site, XDoclet tool, 101

SPI (Service Provider Interface) for engine classes, 584–585

spiral methodology, 82

split() **function**, JSTL, 341

split() **method**, Pattern **class**, 59

SpringLayout **manager**

definition of, 183

example of

JAddEventButton class, 189

JButtonSave class, 189–190

running, 190–191

SpringLayoutPanel class, 184–189

SpringLayoutPanel display, 184

SpringLayout() **method**, SpringLayout **class**, 183

SQL Actions, JSTL

definition of, 342–344

example of, 344–350

SQL statements

executing with JDBC API

batch updates, 294–297

CallableStatement interface, 292–294, 297

PreparedStatement interface, 289–292, 296–297

ResultSet class, 298–302

Statement interface, 288–289, 295–296

executing with JSTL, 342–344

square brackets ([]), in regular expressions, 56

standard transactions, 311

start() **method**

in applets, 637

Matcher class, 60

MessageConsumer class, 548

startsWith() **function**, JSTL, 341

State **pattern**, 214–215

stateful-session beans, 466

stateless-session beans, 466, 468

Statement **interface**, JDBC API, 288–289

static data, importing, 2, 13–15

StaticParametersInterceptor, **WebWork framework**, 373

stop() **method**, in applets, 637

store() **method**, KeyStore **class**, 596

stored procedures, executing on RDBMS, 292–294

Strategy **pattern**

CardLayout manager using, 192

definition of, 134–135

example of, 135–138

StreamHandler **class**, 40–41

strings

EL expressions

Function Tag Library extensions to, 341–342

in JSP scripts, 332–335, 339–340

in JNI, 406–410

pattern matching with, 53–63

stubs, 448, 501

Subject **class, JAAS, 612–613**

substring() **function, JSTL, 341**

substringAfter() **function, JSTL, 341**

substringBefore() **function, JSTL, 341**

subtyping, 135

SUN provider package, 584

super-types, 3

supportResultSetConcurrency() **method,**

DatabaseMetaData **class, 298**

supportsBatchUpdates() **method,**

DatabaseMetaData **class, 303**

supportsGroupBy() **method,**

DatabaseMetaData **class, 303**

supportsSavepoints() **method,**

DatabaseMetaData **class, 303**

supportsStoredProcedures() **method,**

DatabaseMetaData **class, 294, 303**

supportsTransactions() **method,**

DatabaseMetaData **class, 303**

Swing applications

actions, 233–234, 239–242

BorderLayout manager, 144–151

BoxLayout manager, 151–161

CardLayout manager, 191–197, 214

e-mail client example using, 434–435

example of, 122–130

FlowLayout manager, 161–166

GridBagLayout manager, 177–183

GridLayout manager, 167–177

layout managers for, 144

navigation flows, managing, 214–221

serialization and deserialization in, 232–235

SpringLayout manager, 183–191

Swing classes, 143

Swing components, serialization of, 255

sync() **method, Preference class, 68**

System.err, **writing log messages to, 41**

systemNodeForPackage() **method, Preference class, 64**

systemRoot() **method, Preference class, 64**

T

tab, meta-character for, 55

.tag **file extension, 335–336**

<taglib> **element, WAR deployment descriptor, 641**

.tagx **file extension, 335–336**

Target **class, 18, 19, 120**

TCP (Transmission Control Protocol)

monitoring, 496–498

sockets, 480

TCPMon (Apache), 496–498, 534, 539

TestActivationImpl **class, 450–451**

TestClient **class, 451**

testing tools

JMeter tool, 107–109

JUnit tool, 98–101, 113

TestRemoteInterface **interface, 450**

TextMessage, JMS, 552, 558

ThreadGroups, 107

threading

sockets and, 483

using JNI, 429–430

using RMI, 448

three-tier model for JDBC API, 284–285

Throw() **function, 424**

throwing() **method, Logger class, 34**

ThrowNew() **function, 424**

throws **clause, type variables in, 7**

TicTacToe example of Java Web Start

definition of, 647–648

JNLP file for, 648–650

TTTGui class, 653–654

TTTLogic class, 650–653

TTTMain class, 650

tiers of J2EE architecture, 87

time-based license for applications

definition of, 235–236

implementing license, 236–238

implementing timeserver, 238–239

TimerInterceptor, **WebWork framework, 373**

toLowerCase() **function, JSTL, 341**

toMatchResult() **method, MatchResult interface, 61**

topic, JMS, 544

ToReflectedField() **function, 434**

ToReflectedMethod() **function, 433**

toString() **method**

AnnotationValue interface, 22

Level class, 38

Preference class, 68

Principal interface, 614

toUpperCase() **function, JSTL, 341**

traceability of software to requirements, 76–77

transactions

holding result sets open after, 299
managing, 310–312

transient **keyword**, **Java Serialization API**, 243

translateKey() **method**, **SecretKeyFactory**
class, 609

Transmission Control Protocol (TCP)

monitoring, 496–498
sockets, 480

transparent representations of keys, 592, 593–594,
608–609

transport layer, 479–480

trim() **function**, **JSTL**, 341

troubleshooting. *See* exceptions; **Java logging**; **performance**

trusted certificate entry, 596

two-tier model for JDBC API, 283–284

type erasure, 4–5

type variables

arrays of, 6
definition of, 3, 5
exceptions and, 7

TYPE_FORWARD_ONLY constant, 298

type-safe classes (generics)

arrays of, 6
boxing and, 13
defining, 5
definition of, 1, 2
exceptions and, 7
instantiating, 5–6
methods, 6–7
parameterized types, 3–4
raw types (type erasure), 4–5

type-safe enumerations, 2, 15–17

TYPE_SCROLL_INSENSITIVE constant, 298

TYPE_SCROLL_SENSITIVE constant, 298

U

UDP (User Datagram Protocol) sockets, 480

UDT (User Defined Types), 292

UML modeling, 75

unboxing (and boxing) conversions, 2, 11–13

Unicode characters, 407–408

Unified Process (UP), 83–85, 86–87

unit testing, 79

Unmarshaller **class**, 269

unmarshalling

in **RMI**, 501–503
XML data, 269–271, 448

UnregisterNatives() **function**, 431

UP (Unified Process), 83–85, 86–87

update() **method**

Cipher **class**, 604
Mac **class**, 611
Session **class**, 319
Signature **class**, 589

update tag, **JSTL**, 343

use cases, 84, 87, 384–387

usePattern() **method**, **Matcher** **class**, 60

User Datagram Protocol (UDP) sockets, 480

User Defined Types (UDT), 292

user identification, in **JAAS**, 612–613. *See also* authentication

user interface development. *See* **GUI applications**

User Interface Logic, separating from business logic.

See **MVC (Model-View-Controller) pattern**

user requests, handling. *See* **Command pattern**

userNodeForPackage() **method**, **Preference**
class, 64

userRoot() **method**, **Preference** **class**, 65

UTF-8 character encoding, 406–408

utility libraries. *See* **Java logging**; **Java preferences**;
regular expressions

V

validate() **method**, **CertPathValidator** **class**, 601

Validator **class**, 269

value() **method**

AnnotationDesc.ElementValuePair interface, 21

AnnotationValue interface, 22

ValueStack, **WebWork** framework, 373

variable arguments, 2, 9–10

velocity, 81

verification of data with digital signature, 588–592

verify() **method**, **Signature** **class**, 590

Verisign, 600

versioning, **Java Serialization API** and, 245–247

vertical components, 369

View, in **MVC**, 366–367

View **class**, **Model-View-Controller** pattern, 125–127

Visitor pattern, 178, 182

Visual Studio, creating project for JNI, 404–405

W

WAR (Web ARchive File)

in AXIS, for Web services, 533
creating, 104, 658
definition of, 639–640
deploying, 640–643
in EAR (Enterprise Archive), 644–646
warning() **method**, Logger **class**, 33
Waterfall methodology, 82–83, 86–87
weak global references to objects, 425, 427–429
weather example

JavaBean for, 531–532
WeatherGetter class, 532–533
without Web services, 523–526
Web services for, 533–540

Web application archive. See WAR

**Web applications. See also Model 1 Architecture;
Model 2 Architecture**

applets
definition of, 636
in JAR files, 629–630
packaging for execution, 638
RMI for, 446
security of, 639
structure of, 636–638
deploying, 639–643
Java Web Start, 647–654
visualizations
developing with JSP 350–363
developing with JSTL, 344–350

Web ARchive File (WAR)

in AXIS, for Web services, 533
creating, 104, 658
definition of, 639–640
deploying, 640–643
in EAR (Enterprise Archive), 644–646

Web services

client for, writing with AXIS, 537–539
definition of, 522–523, 639
deploying with AXIS, 535–537
example using, 523–526, 531–540
future of, 540
limitations of, 445, 522, 527
remote procedure calls with, 526–527
SOAP and, 529–530
types of, 536
when to use, 522–523, 540
WSDL and, 528–529

**Web Services Description Language (WSDL),
528–529, 537**

Web sites

The Apache XML Project, 499
The Eclipse Project, 499
Gaim, 498
IIOPNET project, 512
The Jakarta Project, 499
JBoss: Professional Open Source, 499
JDBC drivers, 282
JMeter tool, 107–108
JNDI (Java Naming and Directory Interface), 287
King's Hibernate example application, 374
OMG (Object Management Group), 506
OpenSymphony Quality Components, 499
SourceForge, XDoclet tool, 101
WSDL (Web Services Description Language), 528

Web Start

definition of, 647, 654
TicTacToe example of
definition of, 647–648
JNLP file for, 648–650
TTTGui class, 653–654
TTTLogic class, 650–653
TTTMain class, 650

Web tier, J2EE, 87

WebRowSetImpl RowSet implementation, 309

WebWork framework

architecture of, 371–374
definition of, 368–369
extending with Hibernate, 374–377
Interceptors in, 372–373
IoC (Inversion of Control), 369–371
OGNL (Object Graph Navigation Language) for,
373–374
scopes of components, 374
ValueStack in, 373
weightx **variable**, GridLayout **manager**, 178
weighty **variable**, GridLayout **manager**, 178
<welcome-file-list> **element**, WAR **deployment
descriptor**, 641, 642

World Wide Web, evolution of

description of, 523
example illustrating, 524–526
Web services and, 523, 526
wrap() **method**, Cipher **class**, 604–608
writeObject() **method**, Java **Serialization API**,
243–245

wrox.pattern library. **See** patterns

WSDL (Web Services Description Language), 528–529, 537

WSDL2Java toolset, 533, 537–538

X

X.509 Certificate path builder, 584

X.509 format, 596

XDoclet tool

development scenarios using, 101–107

learning, using patterns for, 113

xjc compiler, **263–264**

XML

exporting preferences to, 68–71

Hibernate mappings, 315–317

in WAR deployment descriptor, 640–643

XML schema-based serialization. See JAXB

XmlAction class, **Annotation Editor example, 207**

XMLDecoder class, **250–251**

XMLEncoder class, **250–251**

XMLEncoder/Decoder API

classes used in, 250–251

compared to Java Serialization API, 248–249

of configuration data, 252–254

customizing, 254–255

definition of, 223

format for, 249–250

persistence delegates and, 255

procedure for, 251

when to use, 255–256

XMLFormatter class, **45–47**

X/Open SQL Call Level Interface (CLI), 282

XP (eXtreme Programming), 81, 85–87

XSD (XML Schema Definition), serialization based on.

See JAXB

XWork command framework

definition of, 371

examples using, 384–387, 395–396

