



# A

## About the Online Content

This book comes complete with Total Tester Online customizable practice exam software with 170 practice exam questions. In addition to preparing readers for the OCP Java SE 8 Programmer II exam, this book includes coverage of nearly all of the upgrade exam objectives for candidates recertifying from OCP Java SE 7 or OCP Java SE 6 credentials. You may obtain content covering the handful of upgrade objectives not in this book from the McGraw-Hill Professional Media Center.

## McGraw-Hill Professional Media Center Download

To access the supplemental upgrade objective content, visit McGraw-Hill Professional's Media Center by clicking the link below and entering this e-book's 13-digit ISBN and your e-mail address. You will then receive an e-mail message with a download link for the additional content.

<http://mhprofessional.com/mediacenter/>

This e-book's ISBN is 9781260117370.

Once you've received the e-mail message from McGraw-Hill Professional's Media Center, click the link included to download a zip file. Extract all of the files from the zip file and save them to your computer. If you do not receive the e-mail, be sure to check your spam folder.

## Total Tester Online System Requirements

We recommend and support the current and previous major versions of Chrome, Firefox, Microsoft Edge, and Safari. These browsers update frequently, and sometimes an update may cause compatibility issues with the Total Tester Online or other content hosted on the Training Hub. If you run into a problem using one of these browsers, please try using another one until the problem is resolved.

## Single User License Terms and Conditions

Online access to the digital content included with this book is governed by the McGraw-Hill Education License Agreement outlined next. By using this digital content, you agree to the terms of that license.

**Access** To register and activate your Total Seminars Training Hub account and access your online practice exam, simply follow these easy steps:

1. Go to [hub.totalsem.com/mheclaim](http://hub.totalsem.com/mheclaim).
2. To register and create a new Training Hub account, enter your e-mail address, name, and password. No further information (such as credit card number) is required to create an account.
3. If you already have a Total Seminars Training Hub account, select Log In and enter your e-mail and password.
4. Enter your Product Key: **m2gw-4nj6-k3zd**
5. Click to accept the user license terms.
6. Click Register and Claim to create your account. You will be taken to the Training Hub and have access to the content for this book.

**Duration of License** Access to your online content through the Total Seminars Training Hub will expire one year from the date the publisher declares the book out of print.

Your purchase of this McGraw-Hill Education product, including its access code, through a retail store is subject to the refund policy of that store.

The Content is a copyrighted work of McGraw-Hill Education and McGraw-Hill Education reserves all rights in and to the Content. The Work is

© 2018 by McGraw-Hill Education, LLC.

**Restrictions on Transfer** The user is receiving only a limited right to use the Content for user's own internal and personal use, dependent on purchase and continued ownership of this book. The user may not reproduce, forward, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish, or sublicense the Content or in any way commingle the Content with other third-party content, without McGraw-Hill Education's consent.

**Limited Warranty** The McGraw-Hill Education Content is provided on an "as is" basis. Neither McGraw-Hill Education nor its licensors make any guarantees or warranties of any kind, either express or implied, including, but not limited to, implied warranties of merchantability or fitness for a particular purpose or use as to any McGraw-Hill Education Content or the information therein or any warranties as to the accuracy, completeness, currentness, or results to be obtained from, accessing or using the McGraw-Hill Education content, or any material referenced in such content or any information entered into licensee's product by users or other persons and/or any material available on or that can be accessed through the licensee's product (including via any hyperlink or otherwise) or as to non-infringement of third-party rights. Any warranties of any kind, whether express or implied, are disclaimed. Any material or data obtained through use of the McGraw-Hill Education content is at your own discretion and risk and user understands that it will be solely responsible for any resulting damage to its computer system or loss of data.

Neither McGraw-Hill Education nor its licensors shall be liable to any subscriber or to any user or anyone else for any inaccuracy, delay, interruption in service, error or omission, regardless of cause, or for any damage resulting therefrom.

In no event will McGraw-Hill Education or its licensors be liable for any indirect, special or consequential damages, including but not limited to, lost time, lost money, lost profits or good will, whether in contract, tort, strict liability or otherwise, and whether or not such damages are foreseen or unforeseen with respect to any use of the McGraw-Hill Education content.

## **Total Tester Online**

Total Tester Online access with this book provides you with a simulation of the Java SE 8 Programmer II (1Z0-809) exam. Exams can be taken in Practice Mode or Exam Mode. Practice Mode provides an assistance window with hints, references to the book, explanations of the correct and incorrect answers, and the option to check your answer as you take the test. Exam Mode provides a simulation of the actual exam. The number of questions, the types of questions, and the time allowed are intended to be an accurate representation of the exam environment. The option to customize your quiz allows you to create custom exams from selected domains or chapters, and you can further customize the number of questions and time allowed.

To take a test, follow the instructions provided in the previous section to register and activate your Total Seminars Training Hub account. When you register, you will be taken to the Total Seminars Training Hub. From the Training Hub Home page, select your exam from the Study drop-down list at the top of the page or from the list of Products You Own on the Home page. You can then select the option to customize your quiz and begin testing yourself in Practice Mode or Exam Mode. All exams provide an overall grade and a grade broken down by domain.

## Technical Support

For questions regarding the Total Tester software or operation of the Training Hub, visit [www.totalsem.com](http://www.totalsem.com) or e-mail [support@totalsem.com](mailto:support@totalsem.com).

For questions regarding book content, e-mail [hep\\_customer-service@mheducation.com](mailto:hep_customer-service@mheducation.com). For customers outside the United States, e-mail [international\\_cs@mheducation.com](mailto:international_cs@mheducation.com).

# INDEX

*Please note that index links point to page beginnings from the print edition. Locations are approximate in e-readers, and you may need to page down one or more times after clicking a link to get to the indexed material.*

## A

- absolute() method, [855–857](#)
- abstract classes, [5](#)
  - constructors, [116](#), [119](#)
  - creating, [8–9](#)
  - implementation, [106](#)
  - inner, [458](#), [460](#)
  - vs. interfaces, [10–11](#)
  - overview, [7–8](#)
- abstract methods
  - inheritance, [75](#)
  - overview, [31–35](#)
  - subclass implementation, [88](#)
- accept() method, [498](#), [504–505](#)
- access and access modifiers, [16–18](#)
  - classes, [3–8](#)
  - constructors, [118](#)
  - encapsulation, [71](#)
  - inner classes, [453](#), [458](#)
  - key points, [55](#), [57](#)
  - lambda expression variables, [494–496](#)
  - levels, [40](#)

- local variables, [29](#)
- overloaded methods, [94](#)
- overridden methods, [88–89](#)
- private, [21–22](#)
- protected and default members, [22–27](#)
- public, [18–20](#)
- static methods and variables, [146](#)

accumulators in streams, [567–570](#)

AclFileAttributeView interface, [283](#)

add() method

- ArrayDeque, [390](#)
- BlockingQueue, [738](#)
- collections, [349](#)
- lists, [351](#), [393](#)
- sets, [378](#), [393](#)

addAll() method, [737](#)

addFirst() method, [390](#)

afterLast() method, [856](#), [858](#)

alive thread state, [650–651](#)

allMatch() method, [576–578](#), [586](#)

American National Standards Institute (ANSI), [843](#)

and() method, [497](#), [511–512](#)

andThen() method

- Consumer, [507–508](#)
- Function, [516](#)
- Predicate, [511](#)

angle brackets (<>) in generic code, [396](#), [398–399](#)

anonymous inner classes

- argument-defined, [466–468](#)
- key points, [477](#)
- overview, [461](#)
- plain-old, flavor one, [461–464](#)
- plain-old, flavor two, [464–465](#)

ANSI (American National Standards Institute), [843](#)

- anyMatch() method, [576–578](#), [586](#)
- append() method, [252](#)
- apply() method, [514–517](#)
- applyAsDouble() method
  - DoubleBinaryOperator, [566–567](#)
  - Function, [517](#)
  - ToDoubleFunction, [561](#)
- applyAsInt() method, [517](#)
- argument-defined anonymous inner classes, [466–468](#)
- arguments
  - assertions, [174–175](#)
  - final, [31](#)
  - overloaded methods, [94](#), [97](#)
  - overridden methods, [90](#)
  - vs. parameters, [35–36](#)
  - super constructors, [120](#)
  - variable argument lists, [35–36](#)
- ArrayBlockingQueue class, [738–739](#), [795](#)
- ArrayDeque class, [347](#), [355](#), [389–392](#)
- ArrayIndexOutOfBoundsException class, [734](#)
- ArrayList class, [376](#)
  - basics, [357–358](#)
  - in collection hierarchy, [347–348](#)
  - collections, [733](#)
  - description, [352](#)
  - element order, [350–351](#)
  - sorting, [363–365](#)
- ArrayListRunnable class, [734](#)
- arrays
  - converting with lists, [375–376](#)
  - declaring, [43–45](#), [410–411](#)
  - key points, [60](#), [431](#)
  - methods, [392](#)
  - polymorphism, [408–410](#)



- returning, [113](#)
- searching, [373–375](#)
- vs. streams, [543](#)
- streams from, [548](#), [552](#)
- type-safe, [394–395](#)

`Arrays.asList()` method, [375](#), [392](#)

`Arrays` class, [347](#), [372–373](#)

ASCII set, [39](#)

`asList()` method, [375](#), [392](#)

`AssertionError` class, [172](#)

- appropriate use, [176](#)
- expression rules, [172](#)

assertions

- appropriate use, [176–179](#)
- disabling, [174–175](#)
- expression rules, [172–173](#)
- key points, [193](#)
- overview, [170–172](#)
- running with, [174](#)
- working with, [174–176](#)

associative accumulators, [568–570](#)

associative pipeline operations, [772](#)

asterisks (\*)

- globs, [294–296](#)
- SQL queries, [817](#)

atomic operations, [673](#)

atomic package, [722–723](#)

atomic variables, [722–725](#), [794](#)

`AtomicInteger` class, [725](#)

attributes

- `BasicFileAttributes`, [283–285](#)
- common, [287](#)
- `DosFileAttributes`, [285–287](#)
- interface types, [282–283](#)

- key points, [316](#), [430](#)
- PosixFileAttributes, [286](#)–[287](#)
- reading and writing, [280](#)–[282](#)
- autoboxing with collections, [358](#)–[362](#)
- AutoCloseable interface, [878](#)
- autocloseable resources
  - key points, [194](#)
  - with try-with-resources statements, [185](#)–[189](#)
- automatic local variables, [41](#)–[43](#)
- Automatic Resource Management feature, [186](#)
- automatic variables. *See* local variables
- available processors, [751](#)
- average() method
  - associativity, [569](#)
  - DoubleStream, [562](#)–[563](#), [565](#)
  - optionals, [572](#)
  - parallel streams, [772](#), [787](#)
  - streams, [571](#)
- averaging streams, [597](#)–[599](#)
- averagingInt() method, [597](#)–[598](#), [603](#)
- await() method
  - barriers, [745](#)
  - conditions, [731](#)

## B

- backed collections, [385](#)–[387](#)
- backslashes (\)
  - globs, [294](#)–[296](#)
  - properties files, [228](#)
- barriers, concurrency, [741](#)–[745](#)
- BaseStream interface, [770](#)
- BasicFileAttributes interface, [282](#)–[285](#)
- BasicFileAttributeView interface, [283](#)

- beforeFirst() method, [856](#), [859](#)
- between() method, [217](#)
- BiConsumer interface
  - arguments, [499](#)
  - methods and description, [523](#)
  - working with, [504](#)–[505](#)
- BiFunction interface
  - arguments, [515](#), [517](#)
  - methods and description, [523](#)
- BIGINT data type, [847](#)
- BINARY LARGE OBJECT (BLOB), [843](#)
- BinaryOperator operators, [566](#), [787](#)
- binarySearch() method
  - arrays, [392](#)
  - collections, [392](#)
  - overview, [373](#)–[375](#)
- BiPredicate interface
  - methods and description, [523](#)
  - working with, [513](#)
- BLOB (BINARY LARGE OBJECT), [843](#)
- blocked thread state
  - considerations, [678](#)–[679](#)
  - deadlocks, [684](#)–[685](#)
  - description, [660](#)
- blocking queues, [737](#)–[741](#), [795](#)
- BlockingQueue collection, [737](#)–[738](#)
  - behavior, [738](#)–[739](#)
  - bounded queues, [739](#)
  - LinkedTransferQueue, [740](#)–[741](#)
  - special-purpose queues, [739](#)–[740](#)
- blocks
  - initialization, [137](#)–[139](#)
  - synchronized, [675](#)–[678](#)
  - synchronizing, [678](#)

- bookseller database overview, [819–822](#)
- boolean type and values
  - bit depth, [39](#)
  - SQL, [843](#), [847](#)
  - wrappers, [361](#)
- BooleanSupplier interface, [499](#)
- bounded queues, [739](#)
- braces ({}). *See* curly braces ({})
- BrokenBarrierException class, [745](#)
- buckets for hashcodes, [340–343](#)
- BufferedReader class, [251](#)
- BufferedWriter class
  - description, [251](#)
  - methods, [259](#)
  - using, [258–259](#)
- bugs. *See* exceptions
- built-in functional interfaces, [497](#), [527](#)
- bytes
  - ranges, [39](#)
  - wrappers, [361](#)

## C

- cached thread pools, [751](#)
- Calendar class, [207–208](#)
- call stacks with threads, [644–646](#), [651](#)
- Callable interface
  - overview, [753–754](#)
  - pipelines, [769](#)
- cancelRowUpdates() method, [862–864](#)
- canExecute() method, [282](#)
- canRead() method, [282](#)
- canWrite() method, [282](#)
- CAS (Compare And Swap) feature, [725](#)

case sensitivity, SQL, [818](#)

casts

- with equals(), [337–338](#)

- key points, [151](#)

- overview, [101–104](#)

catch clause. *See* try and catch feature

categories for functional interfaces, [498](#)

ceiling() method, [384](#)

ceilingKey() method, [384](#)

chaining

- constructors, [117](#)

- I/O classes, [258](#)

changeable data, synchronizing, [680](#)

CHARACTER LARGE OBJECT (CLOB), [843](#)

characters and char type, [847](#)

- bit depth, [39](#)

- globs, [295–296](#)

- wrappers, [361](#)

checked exceptions

- interface implementation, [106](#)

- overloaded methods, [94](#)

- overridden methods, [90](#), [92](#)

ChronoUnit class, [207](#), [216–217](#)

Class class, [677](#)

Class.forName() method, [832–833](#)

ClassCastException class

- downcasts, [102](#)

- with equals(), [337–338](#)

classes

- access, [3–8](#)

- constructors. *See* constructors

- dates and times, [224–226](#)

- declaring, [3](#)

- defining, [2–3](#)

- extending, [4](#), [84](#)
- final, [6–7](#), [45–46](#)
- immutable, [134–137](#)
- inner. *See* inner classes
- interface implementation, [106](#)
- literals, [677](#)
- member, [450](#)
- member declarations, [16](#)
- names, [143](#), [175](#)
- thread-safe, [681–684](#)
- `clearWarnings()` method, [875](#)
- CLOB (CHARACTER LARGE OBJECT), [843](#)
- `close()` method, [256](#), [875–879](#)
- Closeable interface, [188–189](#)
- closing SQL resources, [875–879](#)
- code overview
  - coding to interfaces, [357–358](#)
  - synchronizing, [668–673](#)
- `collect()` method, [589–590](#)
  - description, [601](#)
  - stream reduction, [592–593](#)
  - values from streams, [788–790](#)
- Collection interface, [347–349](#), [770](#)
- collections and Collections Framework, [44](#), [332](#)
  - ArrayDeque class, [389–392](#)
  - ArrayList basics, [357–358](#)
  - autoboxing, [358–362](#)
  - backed, [385–387](#)
  - blocking queues, [737–741](#)
  - Comparable interface, [365–367](#)
  - Comparator interface, [367–368](#)
  - concurrent, [733–741](#)
  - converting arrays and lists, [375–376](#)
  - copy-on-write, [735–736](#)

- diamond syntax, [362–363](#)
- hashcodes for, [340](#)
- implementation classes, [356](#)
- interfaces and classes, [347–351](#)
- key points, [429–430](#)
- legacy, [396–398](#)
- List interface, [351–352](#)
- lists, [376–378](#)
- Map interface, [353–354](#)
- maps, [379–382](#)
- methods, [392](#)
- mixing generic and nongeneric, [399–404](#)
- operations, [346–347](#)
- ordered, [349–351](#)
- overview, [346](#)
- polling, [384](#)
- PriorityQueue class, [387–388](#), [393–394](#)
- Queue interface, [354–355](#)
- searching, [373–375](#)
- searching TreeSets and TreeMaps, [382–383](#)
- serialization, [313](#)
- Set interface, [352–353](#)
- sets, [378–379](#)
- sorted, [351](#)
- sorting, [363–372](#)
- streams from, [546–547](#)
- thread-safe, [736](#)
- unboxing problems, [405](#)
- working with, [733–734](#)

Collections class, [347–349](#), [363](#)

Collections.synchronizedList() method, [682–684](#), [734](#)

Collector interface, [589–592](#)

Collectors class, [589–592](#)

collectors for streams, [600–603](#)

- colons (:)
  - assertions, [172](#)
  - URLs, [269](#)
- column indexes, [844](#)
- combining I/O classes, [258–261](#)
- command-line arguments for assertions, [175–176](#), [178](#)
- comments, properties files, [227–229](#)
- Comparable interface
  - vs. Comparator, [369](#)
  - concurrent collections, [737](#)
  - functional interfaces, [498](#)
  - lambda expressions, [369–372](#), [472–473](#)
  - sort orders, [351](#)
  - working with, [365–367](#)
- Comparator interface
  - vs. Comparable, [369](#)
  - concurrent collections, [737](#)
  - sort orders, [351](#)
  - working with, [367–368](#)
- Compare And Swap (CAS) feature, [725](#)
- compare() method, [368–370](#), [498](#)
- compareAndSet() method, [725](#)
- compareTo() method, [365–368](#), [581–582](#)
- comparing() method, [582–583](#)
- compiler and compiling
  - casts, [102](#)
  - interface implementation, [105–106](#)
  - overloaded methods, [97](#)
  - warnings and fails, [401–404](#)
- compose() method, [516](#)
- compute() method
  - ForkJoinTask, [757](#)
  - RecursiveAction, [761](#)
- computeIfAbsent() method, [515–516](#)



concrete classes

- abstract methods implemented by, [88](#)

- creating, [8–9](#)

- subclasses, [32–33](#)

concrete methods, [84](#)

CONCUR\_READ\_ONLY cursor type, [852–853](#), [867](#)

CONCUR\_UPDATABLE cursor type, [853–854](#), [862](#)

concurrency, [645](#), [722](#)

- atomic variables, [722–725](#)

- collections, [733–741](#)

- CyclicBarrier, [741–745](#)

- Executors. *See* Executors

- Fork/Join Framework. *See* Fork/Join Framework

- key points, [703](#), [794–798](#)

- locks, [726–733](#)

- parallel streams. *See* parallel streams

- ThreadPools. *See* ThreadPools

ConcurrentHashMap class, [736–737](#)

ConcurrentLinkedDeque class, [736](#)

ConcurrentLinkedQueue class, [736](#)

ConcurrentMap interface, [737](#)

ConcurrentSkipListMap class, [737](#)

ConcurrentSkipListSet class, [737](#)

Condition interface, [731–732](#)

conditions for locks, [730–732](#)

connect() method, [829](#)

Connection interface, [823–824](#), [875](#)

connections

- databases, [816–817](#)

- DriverManager class, [825–830](#)

consistency in equals() contract, [339](#)

Console class

- description, [252](#)

- working with, [265–266](#)

- console() method, [265](#)
- constant specific class body, [50–53](#)
- constants
  - enum, [48](#)
  - interface, [12–13](#)
- constructing statements, [836–839](#)
- constructors, [115](#)
  - basics, [116](#)
  - chaining, [117](#)
  - declarations, [36–37](#)
  - default, [118–120](#)
  - enums, [50–51](#)
  - inheritance, [75](#), [123](#)
  - key points, [59](#), [152–153](#)
  - overloaded, [123–128](#)
  - rules, [118–119](#)
  - singleton design pattern, [131](#)
  - super() and this() calls, [126](#)
- Consumer interface, [498](#), [504–508](#), [523](#)
- contains() method
  - collections, [349](#)
  - lists, [393](#)
  - sets, [393](#)
- containsKey() method, [393](#)
- containsValue() method, [393](#)
- contracts in JDBC, [823](#)
- controls, [3](#)
- conversions
  - arrays and lists, [375–376](#)
  - return type, [114](#)
  - strings to URIs, [269](#)
  - types. *See* casts
- Coordinated Universal Time (UTC), [211–212](#)
- copy() method, [272](#)

- copy-on-write collections, [735–736](#)
- copying files, [272–273](#)
- CopyOnWriteArrayList collection, [735–736](#)
- CopyOnWriteArraySet collection, [736](#)
- cost reduction, object-oriented design for, [82](#)
- count() method
  - return values, [571](#)
  - stream elements, [543–544](#)
  - stream reduction, [559](#), [565](#)
  - stream values, [788](#)
- counting
  - instances, [140](#)
  - streams, [599](#)
- counting() method, [599](#), [603](#)
- covariant returns, [112–113](#)
- CPU-intensive vs. I/O-intensive tasks, [748](#)
- createNewFile() method, [254–255](#), [261–262](#), [270](#)
- creationTime() method, [287](#)
- credentials for database, [827](#)
- CRUD operations, [817–818](#)
- curly braces ({} )
  - abstract methods, [32](#)
  - anonymous inner classes, [462–463](#), [465](#)
  - globs, [295–296](#)
  - inner classes, [453](#), [458](#), [462–463](#)
  - lambda expressions, [493](#)
  - methods, [32](#)
- currently running thread state, [658](#)
- currentThread() method, [653](#)
- cursor types for ResultSets, [853](#)
- CyclicBarrier, [741–745](#)

daemon threads, [646](#)

data types in ResultSets, [843](#)

DatabaseMetaData class, [868](#)–873

databases

- connections, [816](#)–817

- credentials, [827](#)

- JDBC. *See* JDBC API

- overview, [814](#)–816

DataSource class, [831](#)

Date class

- description, [207](#)–208

- with SQL, [843](#)

DATE data type, [843](#), [847](#)

dates

- adjustments, [213](#)–214

- classes, [224](#)–226

- durations, [216](#)–217

- examples, [219](#)–220

- format attributes, [281](#)

- formatting, [220](#)–221

- instants, [217](#)–219

- java.time.\* classes, [208](#)–210

- key points, [238](#)–239

- locales, [222](#)–224

- overview, [207](#)

- periods, [214](#)–216

- zoned, [211](#)–213

DateTimeFormatter class, [209](#)–210, [220](#)–222, [225](#)

dead thread state, [651](#), [661](#)

deadlocks

- key points, [704](#)

- threads, [684](#)–685

- tryLock() for, [728](#)–730

Deadly Diamond of Death, [84](#)

## declarations

- arrays, [43–45](#), [410–411](#)
- class members, [16](#)
- classes, [3](#)
- constructors, [36–37](#)
- enum elements, [50–53](#)
- enums, [48–50](#)
- generics, [419–420](#)
- interface constants, [12–13](#)
- interfaces, [9–12](#)
- reference variables, [40](#)
- return types, [112–114](#)
- variables, [37–47](#), [59–60](#)

decoupling tasks from threads, [749–751](#)

## default access

- description, [3](#)
- overview, [4–5](#)
- and protected, [16](#), [22–27](#)

default protection, [16](#)

defaultReadObject() method, [308–309](#)

## defaults

- constructors, [118–120](#)
- interface methods, [14](#)
- locales, [234](#)
- thread priorities, [666](#)

defaultWriteObject() method, [309](#)

## defining

- classes, [2–3](#)
- inner classes, [458](#)
- threads, [647–648](#)

Delayed interface, [740](#)

DelayQueue class, [738–740](#), [795](#)

delete() method, [263–264](#), [272](#)

DELETE operation for SQL, [818](#)

- deleteIfExists() method, [273](#)
- deleteRow() method, [864–865](#)
- deleting files, [272–273](#)
- depth-first searches, [291](#)
- Deque interface, [389](#)
- descending order in collections, [384](#)
- descendingMap() method, [384](#)
- descendingSet() method, [384](#)
- deserialization process, [252](#)
- design patterns
  - description, [129](#)
  - factory, [825–826](#)
  - singleton. *See* singleton design pattern
- diamond syntax, [362–363](#)
- digits in globs, [295–296](#)
- directories
  - creating, [270–271](#)
  - DirectoryStream, [288–289](#)
  - FileVisitor, [289–293](#)
  - iterating through, [288–289](#)
  - key points, [316–317](#)
  - renaming, [299](#)
  - working with, [261–265](#)
- DirectoryStream interface, [288–289](#)
- disabling assertions, [174–175](#)
- Disk Operating System (DOS), [283](#)
- DISTINCT characteristic for streams, [778](#)
- distinct() method, [584](#), [587](#), [777](#), [780](#)
- distributed-across-the-buckets hashcodes, [342](#)
- divide and conquer technique, [756–757](#)
- DOS (Disk Operating System), [283](#)
- DosFileAttributes interface, [283](#), [285–287](#)
- DosFileAttributeView interface, [283](#)
- dots (.)

- access, [17](#)
- class names, [143](#)
- instance references, [143](#)
- variable argument lists, [36](#)
- double type
  - ranges, [39](#)
  - SQL, [847](#)
- DoubleBinaryOperator operators, [566–567](#)
- DoubleConsumer interface, [501](#), [504](#)
- DoubleFunction interface, [517](#)
- DoublePredicate interface, [513](#)
- DoubleStream interface
  - description, [551–552](#)
  - methods, [570–571](#), [601](#)
  - optionals, [572](#)
- DoubleToIntFunction interface, [517](#)
- DoubleToLongFunction interface, [517](#)
- downcasts, [102–103](#)
- Driver interface, [829](#), [832](#)
- DriverManager class
  - database connections, [816](#)
  - description, [825](#)
  - key points, [882](#)
  - overview, [826–828](#)
  - registering JDBC drivers, [828–830](#)
- drivers, JDBC, [824](#), [828–830](#), [832–833](#)
- durations
  - dates, [216–217](#)
  - threads, [657](#)
- Durations class, [207](#), [216–217](#), [225](#)

## E

eager initialization, [132](#)

- element existence, testing for, [577–578](#)
- element() method, [739](#)
- elevator property, [236](#)
- eligible thread state, [658](#)
- ellipses (...) in variable argument lists, [36](#)
- embarrassingly parallel problems, [764–766](#), [770–771](#)
- empty() method, [576](#)
- empty optionals, [575–576](#)
- enabling assertions, [174–175](#)
- encapsulation
  - benefits, [82](#)
  - key points, [149](#)
  - overview, [70–73](#)
- ENTRY\_CREATE type, [298–299](#)
- ENTRY\_DELETE type, [298–299](#)
- ENTRY\_MODIFY type, [298–299](#)
- entrySet() method, [547](#)
- Enum class, [380](#)
- enums, [48](#)
  - constants, [48](#)
  - declaring, [48–50](#)
  - element declarations, [50–53](#)
  - key points, [60–61](#)
- equal signs (=)
  - reference equality, [334–335](#)
  - wrappers, [360–361](#)
- equality and equality operators
  - hashcodes, [344–346](#)
  - references, [334–335](#)
- equals() method, [332](#)
  - arrays, [392](#)
  - Comparable, [498](#)
  - contract, [339](#)
  - description, [333](#)



- implementing, [336–338](#)
- key points, [428–429](#)
- maps, [353](#), [379–382](#)
- overriding, [334–335](#)
- Set, [352](#)
- wrappers, [360–361](#)
- erasure, type, [403](#)
- escape characters and sequences in globs, [295](#)
- event handlers, [451–452](#)
- ExceptionInInitializerError class, [139](#)
- exceptions
  - interface implementation, [106](#)
  - JDBC, [873–879](#)
  - overridden methods, [90](#), [92](#)
  - rethrowing, [182–184](#)
  - suppressed, [190–192](#)
  - try and catch. *See* try and catch feature
  - try-with-resources feature, [185–189](#), [194](#), [877–878](#)
- exclamation points (!)
  - properties files, [227](#)
  - wrappers, [360–361](#)
- exclusive-OR (XOR) operator, [343](#)
- execute() method
  - description, [839](#)
  - overview, [837–838](#)
- execute permission, [281–282](#), [286](#)
- executeQuery() method
  - description, [839](#)
  - overview, [836](#)
- executeUpdate() method
  - description, [839](#)
  - overview, [836–839](#)
- ExecutionException class, [754](#)
- Executor class, [749](#)

Executors, [745–746](#)

- Callable interface, [753–754](#)

- CPU-intensive vs. I/O-intensive tasks, [748](#)

- decoupling tasks from threads, [749–751](#)

- ExecutorService shutdown, [754–755](#)

- key points, [796](#)

- parallel tasks, [746–747](#)

- thread limits, [747](#)

- thread pools, [751–752](#)

- ThreadLocalRandom, [754](#)

- turns, [748–749](#)

Executors.newCachedThreadPool() method, [752](#)

Executors.newFixedThreadPool() method, [752](#)

ExecutorService, [751–752](#)

- Callable, [753–754](#)

- shutdown, [754–755](#)

ExecutorService.shutdownNow() method, [755](#)

existence, stream elements, [577–578](#)

exists() method, [254–255](#), [270](#), [273](#)

exit() method, [755](#)

explicit values in constructors, [117](#)

expressions

- assertions, [172–173](#)

- globs, [295–296](#)

- regular, [296](#)

extended ASCII set, [39](#)

extending

- classes, [4](#), [84](#)

- inheritance in, [76](#)

- interfaces, [107](#)

- Thread class, [647–648](#)

extends keyword

- illegal uses, [110](#)

- IS-A relationships, [79](#)

## F

factory design patterns, [825–826](#)

fails vs. warnings, [402](#)

FIFO (first-in, first-out) queues, [354–355](#)

File class

- creating files, [252–255](#)

- description, [251](#)

- files and directories, [261](#)

- key points, [315–316](#)

- methods, [259](#)

File.list() method, [264](#)

file:/ protocol, [269](#)

FileInputStream class

- methods, [259](#)

- working with, [257–258](#)

FileNotFoundException class, [181](#)

FileOutputStream class

- methods, [259](#)

- working with, [257–258](#)

FileOwnerAttributeView interface, [283](#)

FileReader class

- description, [251](#)

- methods, [259](#)

- working with, [255–257](#)

files, [261–265](#)

- attributes. *See* attributes

- copying, moving, and deleting, [272–273](#)

- creating, [252–255](#), [270–271](#)

- key points, [315](#)

- navigating, [250–252](#)

- permissions, [281–282](#)

- renaming, [299](#)

- searching for, [264](#)

- streams from, [549–551](#)
- Files class, [267](#), [282](#)
- Files.delete() method, [272](#)
- Files.deleteIfExists() method, [273](#)
- Files.getLastModifiedTime() method, [281](#)
- Files.exists() method, [271](#), [273](#)
- Files.walkFileTree() method, [289–290](#)
- FileSystems.getDefault() method, [293](#)
- FileUtils class, [273](#)
- FileVisitor interface, [289–293](#)
- FileWriter class
  - description, [251](#)
  - methods, [259](#)
  - working with, [255–257](#)
- filter() method for streams
  - with collections, [547](#)
  - description, [544–545](#), [556–557](#)
  - pipelines, [777](#)
  - return values, [571](#)
- final arguments, [31](#)
- final classes, [6–7](#), [45–46](#)
- final constants, [12–13](#)
- final methods
  - nonaccess member modifiers, [30–31](#)
  - overriding, [90](#)
- final modifiers
  - inner classes, [458–460](#)
  - variables, [29](#), [45](#)
- finalize() method, [333](#)
- finally clauses
  - key points, [194](#)
  - with try and catch, [179–183](#)
- findAny() method
  - optionals, [572](#)

- parallel streams, [782–783](#)
- Stream, [576](#), [579–580](#), [586–587](#)
- findFirst() method
  - optionals, [572–573](#), [575](#)
  - Stream, [576](#), [579](#), [586–587](#)
- FindMaxPositionRecursiveTask task, [764](#)
- first-in, first-out (FIFO) queues, [354–355](#)
- first() method, [856](#), [859](#)
- firstDayOfNextYear() method, [214](#)
- fixed thread pools, [751](#)
- flatMap() method, [605–606](#)
- flatMapToDouble() method, [606](#)
- flatMapToInt() method, [606](#)
- flatMapToLong() method, [606](#)
- flexibility from object orientation, [70](#)
- float type and floating-point numbers, [847](#)
  - classes, [5–6](#)
  - ranges, [39](#)
- floor() method, [384](#)
- floorKey() method, [384](#)
- flush() method, [256](#)
- for-each loops, [736](#)
- forEach() method
  - Consumer, [505–506](#)
  - key points, [526](#)
  - parallel streams, [780–782](#)
  - pipelines, [554–555](#)
  - streams, [549–551](#)
- forEachOrdered() method, [770](#), [780–782](#)
- foreign keys, [821–822](#)
- Fork/Join Framework, [755–756](#)
  - divide and conquer technique, [756–757](#)
  - embarrassingly parallel problems, [764–766](#)
  - ForkJoinPool, [757](#)

- ForkJoinTask, [757](#)–758
  - join(), [760](#)–761
  - key points, [796](#)–797
  - RecursiveAction, [761](#)–762
  - RecursiveTask, [762](#)–764
  - work stealing, [759](#)–760
- fork() method, [757](#)
- ForkJoinPool class, [757](#), [775](#)–777, [783](#)–786
- ForkJoinTask class, [757](#)–758
- format() method, [221](#), [252](#), [851](#)
- formatting
  - dates and times, [207](#), [220](#)–221
  - reports, [851](#)–852
- forName() method, [832](#)–833
- fromMillis() method, [287](#)
- Function interface, [499](#), [514](#)–517, [523](#)
- functional interfaces, [496](#)–497
  - binary versions, [529](#)
  - built-in, [497](#)
  - categories, [498](#)
  - Comparable, [369](#)–370, [472](#)–473, [498](#)
  - description, [490](#), [497](#)–498
  - functions, [499](#), [514](#)–517
  - key points, [527](#)–528
  - operators, [517](#)–518
  - overview, [521](#)–522
  - primitive versions, [528](#)–529
  - suppliers, [498](#)–503
  - UnaryOperator, [529](#)
  - writing, [520](#)–521
- Future class, [753](#)–754

`generate()` method, [607](#), [611](#)

generics, [332](#)

- classes, [420–424](#)

- declarations, [419–420](#)

- `equals()`, [334–340](#)

- `hashCode()`, [340–346](#)

- key points, [431–433](#)

- legacy code, [398–399](#)

- methods, [407–419](#), [424–426](#)

- mixing with nongeneric, [399–404](#)

- overview, [394–395](#)

- polymorphism, [405–407](#)

- `toString()`, [333–334](#)

`get()` method

- `HashTable`, [230](#)

- lists, [351](#), [378](#), [393](#)

- maps, [382](#), [393](#)

- optionals, [576](#)

- paths, [268–270](#)

- `Supplier`, [498](#), [500](#)

- unboxing problems, [405](#)

`getAndIncrement()` method, [725](#)

`getAsDouble()` method, [563–564](#), [574](#)

`getAsInt()` method, [501](#), [574](#)

`getAsLong()` method, [574](#)

`getBoolean()` method, [845](#)

`getBundle()` method, [232](#), [234–235](#)

`getColor()` method, [585](#)

`getColumnCount()` method, [848–849](#)

`getColumnDisplaySize()` method, [850](#)

`getColumnName()` method, [849](#)

`getColumns()` method, [869–871](#)

`getConnection()` method, [826–828](#), [832](#)

`getDate()` method, [846](#)

`getDayOfWeek()` method, [211](#)  
`getDefault()` method, [235](#), [293](#)  
`getDelay()` method, [740](#)  
`getDisplayCountry()` method, [223](#)  
`getDisplayLanguage()` method, [223](#)  
`getDouble()` method, [845](#)  
`getDriverName()` method, [869](#), [872](#)  
`getDriverVersion()` method, [869](#), [872](#)  
`getEpoch()` method, [219](#)  
`getErrorCode()` method, [874](#)  
`getFileAttributeView()` method, [287](#)  
`getFileName()` method, [274](#)  
`getFloat()` method, [845](#)  
`getId()` method, [657](#)  
`getInt()` method, [845](#)  
`getLastModifiedTime()` method, [281](#)  
`getLong()` method, [845](#)  
`getMessage()` method, [873](#)  
`getMetaData()` method, [868–869](#)  
`getName()` method, [274](#), [652](#)  
`getNameCount()` method, [274](#)  
`getNextException()` method, [874](#)  
`getNextWarning()` method, [875](#)  
`getObject()` method, [234](#), [846](#), [852](#), [865](#)  
`getParent()` method, [274](#)  
`getPathMatcher()` method, [293](#)  
`getProcedures()` method, [869](#), [871–872](#)  
`getProperty()` method, [230](#)  
`getResultSet()` method, [838–839](#)  
`getRoot()` method, [274](#)  
`getRow()` method, [857](#)  
`getRules()` method, [213](#)  
`getSQLState()` method, [873](#)  
`getState()` method, [651](#)



- getString() method, [846](#)
- getSuppressed() method, [878](#)
- getTableName() method, [850](#)
- getters encapsulation, [71](#)
- getTime() method, [846](#)
- getTimestamp() method, [846](#)
- getUpdateCount() method, [838–839](#)
- getWarnings() method, [875](#)
- globs, [289](#), [294–296](#)
- graphs, object, [303–306](#)
- Greenwich Mean Time (GMT), [211–212](#)
- grouping streams, [593–597](#)
- groupingBy() method, [593–597](#), [602](#)
- groups in PosixFileAttributes, [286–287](#)
- guarantees with threads, [655–656](#), [665](#)

## H

- handle and declare pattern, [182](#)
- hard-coding credentials, [827](#)
- HAS-A relationships
  - key points, [149](#)
  - overview, [78–82](#)
- hashCode() method
  - contract, [344–346](#)
  - generics, [332–333](#)
  - HashSet, [353](#)
  - implementing, [342–344](#)
  - key points, [428–429](#)
  - maps, [379–382](#)
  - overriding, [340–343](#)
- hashcodes overview, [340–343](#)
- HashMap collection, [347](#)
  - description, [354](#)

- hashcodes, [340](#)
- HashSet collection, [347](#)
  - description, [353](#)
  - hashcodes, [340](#)
  - ordering, [378](#)
- Hashtable collection, [230](#), [347](#)
  - description, [354](#)
  - keys, [336](#)
  - ordering, [350](#)
- hasNext() method, [377](#)
- headMap() method, [386](#)
- heads of queues, [390](#)
- headSet() method, [386](#)
- hiding implementation details, [71](#)
- hierarchy in tree structures, [275](#)
- higher-level classes, [301](#)
- higher() method, [384](#)
- higherKey() method, [384](#)

## I

- identifiers
  - Map, [353](#)
  - threads, [657](#)
- identity arguments for streams, [567](#)
- identity() method, [516](#)–[517](#)
- IllegalMonitorStateException class, [693](#), [729](#)
- IllegalThreadStateException class, [657](#)
- immutable classes
  - key points, [153](#)
  - overview, [134](#)–[137](#)
- immutable objects, thread safe, [736](#)
- implementation details, hiding, [71](#)
- implementers of interfaces, [464](#)–[465](#)

implementing interfaces

- key points, [151](#)

- overview, [105–111](#)

implements keyword

- illegal uses, [110](#)

- IS-A relationships, [79](#)

indexes

- ArrayLists, [350](#)

- columns, [844](#)

- List, [351](#)

- searches, [373](#)

indexOf() method, [351](#), [378](#), [393](#)

IndexOutOfBoundsException class, [682](#)

information about ResultSets, [848–850](#), [868–873](#)

inheritance

- access modifiers, [17–18](#), [24–25](#)

- constructors, [123](#)

- event handlers, [452](#)

- evolution, [75–77](#)

- HAS-A relationships, [80–82](#)

- IS-A relationships, [78–79](#)

- key points, [149](#)

- multiple, [84](#), [110–111](#)

- overview, [74–75](#)

- serialization, [309–312](#)

inherited methods, overriding, [91](#)

initialization blocks, [137–139](#)

- inheritance, [75](#)

- key points, [154](#)

initialization of variables, [42](#)

injection attacks, [838](#)

inner classes, [3](#), [450–451](#)

- anonymous. *See* anonymous inner classes

- defining, [458](#)

- instantiating, [454](#)–455
- key points, [476](#)
- lambda expressions, [469](#)–473, [478](#)
- method-local, [458](#)–460
- modifiers, [458](#)
- objects, [455](#)–456
- overview, [451](#)–453
- referencing instances, [456](#)–458
- regular, [453](#)–454
- static, [468](#)–469
- input/output (I/O), [249](#)
  - combining classes, [258](#)–261
  - Console class, [265](#)–266
  - directories. *See* directories
  - files. *See* files
  - key points, [315](#)–317
  - path creation, [268](#)–270
  - Runnable and Callable, [754](#)
  - serialization, [301](#)–313
  - WatchService, [297](#)–300
- input-output intensive vs. CPU-intensive tasks, [748](#)
- InputStream.read() method, [748](#)
- INSERT operation, [817](#)–818
- inserting rows, [866](#)–868
- insertion points in searches, [373](#)
- insertRow() method, [863](#), [868](#)
- instance methods
  - inheritance, [75](#)
  - overriding, [90](#)
  - polymorphic, [86](#)
  - references, [520](#)
- instance variables
  - constructors, [117](#)
  - inheritance, [75](#)

- overview, [40–41](#)
- instances
  - counting, [140](#)
  - initialization blocks, [138](#)
  - references to, [143](#), [456–458](#)
- instantiation, [152–153](#)
  - eager and lazy, [132](#)
  - inner classes, [454–455](#)
  - static nested classes, [468–469](#)
  - threads, [644–646](#), [649–651](#)
- Instant class, [207](#), [225](#)
- instants in dates, [217–219](#)
- IntConsumer interface, [499](#), [504](#), [523](#)
- integers and int data type
  - ranges, [39](#)
  - ResultSets, [847](#)
  - wrappers, [361](#)
- interfaces, [71](#)
  - vs. abstract classes, [10–11](#)
  - attributes, [282–283](#)
  - constants, [12–13](#)
  - constructors, [119](#)
  - declaring, [9–12](#)
  - extending, [107](#)
  - functional. *See* functional interfaces
  - implementing, [105–111](#), [151](#)
  - JDBC, [823–824](#)
  - key points, [56](#)
  - methods, [14–15](#)
- intermediate operations
  - pipelines, [553](#)
  - streams, [544](#), [552–553](#)
- interrupt() method, [755](#)
- InterruptedException class

- barriers, [745](#)
- Callable, [754](#)
- Conditions, [730](#)–[732](#)
- sleep(), [662](#)
- WatchService, [299](#)
- IntFunction interface
  - arguments, [517](#)
  - methods and description, [523](#)
- IntPredicate interface
  - arguments, [513](#)
  - methods and description, [523](#)
- IntStream interface, [551](#)
  - methods, [552](#), [571](#), [601](#)
  - optionals, [572](#)
- IntSupplier interface, [523](#)
- IntToDoubleFunction interface, [517](#)
- IntToLongFunction interface, [517](#)
- invokeAll() method, [762](#)
- invoking
  - overloaded methods, [96](#)–[98](#)
  - Polymorphic methods, [85](#)
- I/O. *See* input/output (I/O)
- IOException class
  - directories, [262](#)
  - files, [181](#)
- IS-A relationships
  - key points, [149](#)
  - overview, [78](#)–[79](#)
  - polymorphism, [83](#)
  - return types, [115](#)
  - Serializable, [309](#)
- isAfterLast() method, [859](#)–[860](#)
- isAlive() method, [651](#)
- isBeforeFirst() method, [859](#)

- isDaylightSavings() method, [213](#)
- isDirectory() method, [284](#)
- isEqual() method, [497](#), [512–513](#)
- isFirst() method, [860](#)
- isHidden() method, [287](#)
- isInterrupted() method, [755](#)
- isLast() method, [860](#)
- isLeapYear() method, [220](#)
- ISO Latin-1 characters, [39](#)
- isParallel() method, [769–770](#)
- isPresent() method, [573–574](#), [576](#)
- isReadOnly() method, [287](#)
- Iterable class, [274](#)
- iterate() method, [607–608](#), [611](#)
- iteration
  - collections, [350](#), [735–736](#)
  - directories, [288–289](#)
  - paths, [274–275](#)
- iterator() method
  - collections, [349](#), [735–736](#)
  - lists, [393](#)
  - sets, [393](#)
- Iterators
  - collections, [735–736](#)
  - lists, [376–377](#)

## J

- java.io.Console class
  - description, [252](#)
  - working with, [265–266](#)
- java.io.IOException class
  - directories, [262](#)
  - files, [181](#)

- java.io.ObjectInputStream class
  - description, [252](#)
  - working with, [258](#), [301–303](#)
- java.io.ObjectOutputStream class
  - description, [252](#)
  - working with, [258](#), [301–303](#)
- java.io package
  - classes, [258–259](#)
  - files and directories, [261–265](#)
- java.lang.Class class, [677](#)
- java.lang.ClassCastException class
  - downcasts, [102](#)
  - with equals(), [337–338](#)
- java.lang.Enum class, [380](#)
- java.lang.Object class
  - collections, [396](#), [405](#)
  - description, [332](#)
  - equals() method. *See* equals() method
  - inheritance from, [74](#)
  - threads, [659](#), [699](#)
- java.lang.Runnable interface
  - executing, [746](#)
  - I/O activities, [754](#)
  - implementing, [648](#)
  - threads, [647](#), [699](#)
- java.lang.Runtime class, [751](#)
- java.lang.StringBuilder class, [681](#)
- java.lang.System class, [226](#)
- java.lang.Thread class
  - description, [644](#)
  - extending, [647–648](#)
  - methods, [646](#), [658](#)
  - thread methods, [699](#)
- Java Naming and Directory Interface (JNDI) lookup, [831](#)



- java.nio.file package, [267–268](#), [288](#)
- java.nio.file.attribute package, [267](#), [280](#)
- java.nio.file.Path interface
  - key points, [315–316](#)
  - methods, [274–275](#)
  - working with, [267–268](#)
- Java resource bundles, [233–234](#)
- java.sql.Connection interface, [823–824](#), [875](#)
- java.sql.Date class
  - description, [207–208](#)
  - with SQL, [843](#)
- java.sql.Driver interface, [829](#), [832](#)
- java.sql.DriverManager class
  - database connections, [816](#)
  - description, [825](#)
  - key points, [882–883](#)
  - overview, [826–828](#)
  - registering JDBC drivers, [828–830](#)
- java.sql package, [823](#)
- java.sql.ResultSet interface, [823–824](#)
- java.time.\* classes, [208–210](#)
- java.time.Durations class, [207](#), [216–217](#), [225](#)
- java.time.format.DateTimeFormatter class, [209–210](#), [220–222](#), [225](#)
- java.time.Instants class, [207](#), [225](#)
- java.time.LocalDate class, [207](#), [225–226](#)
- java.time.LocalDateTime class, [207–210](#), [225–226](#)
- java.time.LocalTime class, [207](#), [210](#), [225–226](#)
- java.time.OffsetDateTime class, [207](#), [214](#), [225](#)
- java.time.Periods class, [207](#), [225](#)
- java.time.temporal.ChronoUnit class, [207](#), [216–217](#)
- java.time.temporal.TemporalAdjusters class, [207](#), [214](#)
- java.time.ZonedDateTime class, [207](#), [212–214](#), [218](#), [226](#)
- java.util.ArrayList class. *See* ArrayList class
- java.util.Calendar class, [207–208](#)

- java.util.Collection interface, [347–349](#)
- java.util.Collections class, [347–349](#), [363](#)
- java.util.concurrent.atomic package, [722–723](#)
- java.util.concurrent.Callable interface
  - overview, [753–754](#)
  - pipelines, [769](#)
- java.util.concurrent collections, key points, [794–796](#)
- java.util.concurrent.Delayed interface, [740](#)
- java.util.concurrent.Executor class, [749](#)
- java.util.concurrent.ForkJoinPool class, [757](#), [775–777](#), [783–786](#)
- java.util.concurrent.ForkJoinTask class, [757–758](#)
- java.util.concurrent.Future class, [753–754](#)
- java.util.concurrent.locks.Condition interface, [731–732](#)
- java.util.concurrent.locks.Lock interface, [727](#)
- java.util.concurrent.locks package, [722–723](#), [726–727](#)
- java.util.concurrent.locks.ReentrantLock class, [727–730](#)
- java.util.concurrent package, [722](#). *See also* concurrency
- java.util.concurrent.ThreadLocalRandom class, [754](#)
- java.util.concurrent.ThreadPoolExecutor, [752](#)
- java.util.Date class
  - limitations, [207–208](#)
  - with SQL, [843](#)
- java.util.function.Consumer interface, [498](#), [504–508](#)
- java.util.function.Function interface, [499](#), [514–517](#)
- java.util.function package, [497](#), [521–523](#)
- java.util.function.Predicate interface, [369](#), [497–498](#), [508–514](#)
- java.util.function.Supplier interface, [498–503](#)
- java.util.Hashtable class, [230](#)
- java.util.List interface, [347](#)
  - implementations, [351–352](#)
  - methods, [393](#)
  - threads, [733](#)
- java.util.Locale class
  - resource bundles, [231](#)

- working with, [222–223](#), [225](#)
- java.util.logging.Logger class, [501–503](#)
- java.util.NavigableMap interface, [347](#), [382](#)
- java.util.NavigableSet interface, [347](#), [382](#)
- java.util.NoSuchElementException class, [391](#)
- java.util package, [385](#), [431](#)
- java.util.Properties class, [226–230](#)
- java.util.ResourceBundle class, [227](#), [231–233](#)
- javax.sql.DataSource class, [831](#)
- JDBC API, [814](#)
  - database connections, [816–817](#)
  - database overview, [814–816](#)
  - driver implementation versions, [832–833](#)
  - DriverManager class, [825–830](#)
  - drivers, [824](#), [828–830](#)
  - exceptions and warnings, [873–879](#)
  - interfaces, [823–824](#)
  - key points, [882](#)
  - query submissions, [833–839](#)
  - result sets. *See* ResultSets
  - SQL queries, [817–818](#)
  - statements, [835–839](#)
  - test database, [819–822](#)
  - URL, [830–832](#)
- JIT (Just In Time) compiler, [724](#)
- JNDI (Java Naming and Directory Interface) lookup, [831](#)
- join() method
  - Fork/Join Framework, [757](#), [760–761](#)
  - key points, [703](#)
  - locks, [679](#)
  - threads, [658](#), [666–668](#), [699](#)
- join tables, [821–822](#)
- joining() method, [599–600](#), [603](#)
- joining streams, [599–600](#)

Just In Time (JIT) compiler, [724](#)

## K

`key.pollEvents()` method, [299](#)

`key.reset()` method, [299](#)

keys

- databases, [815](#), [820–822](#)

- hashtables, [336](#)

- Map, [353](#)

- properties files, [227](#)

`keySet()` method, [393](#)

## L

Labels\_en.properties file, [231–232](#)

Labels\_fr.properties file, [231–232](#)

lambda expressions

- Comparable interface, [369–372](#), [472–473](#)

- functional interfaces. *See* functional interfaces

- inner classes, [469–473](#), [478](#)

- key points, [526](#)

- method references, [518–520](#)

- passing to methods, [494](#)

- side effects, [506–507](#)

- syntax, [490–493](#)

- variable access from, [494–496](#)

LARGE OBJECT types, [843](#)

large tasks, divide and conquer technique for, [756–757](#)

`last()` method, [856](#), [859](#)

`lastDayOfMonth()` method, [214](#)

`lastModified()` method, [282](#)

`lastModifiedTime()` method, [287](#)

lazy initialization, [132](#)

- lazy streams, [555–556](#)
- leap years, [220](#)
- legacy code in generics, [398–399](#)
- LIKE operator, [841](#)
- limit() method, [607–608](#), [611](#), [777](#), [780](#)
- lines() method, [549](#), [592–593](#)
- LinkedBlockingDeque class, [738–739](#), [795](#)
- LinkedBlockingQueue class, [738–739](#), [795](#)
- LinkedHashMap class, [347](#), [354](#)
- LinkedHashSet class, [347](#), [350](#), [353](#)
- LinkedList class, [347](#), [352](#), [376](#)
- LinkedTransferQueue class, [738](#), [740–741](#), [795](#)
- list.iterator() method, [735–736](#)
- list() method, [230](#), [264](#)
- ListIterator interface, [736](#)
- ListResourceBundle class, [233–234](#)
- lists and List interface, [347](#)
  - ArrayLists. *See* ArrayList class
  - collecting items in, [593](#)
  - converting with arrays, [375–376](#)
  - description, [349](#)
  - implementations, [351–352](#)
  - key points, [431](#)
  - methods, [393](#)
  - threads, [733](#)
  - working with, [376–378](#)
- literals
  - class, [677](#)
  - globs, [295](#)
- livelocks
  - key points, [704](#)
  - threads, [685–686](#)
- load() method, [230](#)
- local variables

- access modifiers, [29](#)
- inner classes, [459](#)
- key points, [58](#)
- on stack, [41](#)
- working with, [41–43](#)

LocalDate class, [207](#), [225–226](#)

LocalDateTime class, [207–210](#), [225–226](#)

Locale class

- resource bundles, [231](#)
- working with, [222–223](#), [225](#)

locales

- dates and times, [222–224](#)
- default, [234](#)
- key points, [239](#)
- resource bundles. *See* resource bundles

localhosts in URLs, [831](#)

LocalTime class, [207–210](#), [225–226](#)

Lock interface, [727](#)

locks, [726–727](#)

- conditions, [730–732](#)
- obtaining, [678–679](#)
- ReentrantReadWriteLock, [732–733](#)
- synchronization, [674–677](#)

locks package, [722–723](#), [726–727](#)

log() method, [501–503](#)

Logger class, [501–503](#)

long type, [39](#)

LongConsumer interface, [501](#), [504](#)

LongFunction interface, [517](#)

LongPredicate interface, [513](#)

LongStream interface, [551–552](#)

- methods, [571](#), [601](#)
- optionals, [572](#)

LongToDoubleFunction interface, [517](#)

- LongToIntFunction interface, [517](#)
- LONGVARCHAR data type, [847](#)
- loop constructs, wait() in, [697–699](#)
- lower-level classes, [301–302](#)
- lower() method, [384](#)
- lowercase characters in natural ordering, [394](#)
- lowerKey() method, [384](#)

## M

- main() method
  - overloaded, [98](#)
  - threads, [644–646](#)
- maintainability, object orientation for, [70](#)
- map-filter-reduce operations
  - overview, [556–559](#)
  - streams, [570–572](#), [619–620](#)
  - working with, [560–564](#)
- map() method, [556–557](#), [570](#), [777](#)
- mapping() method, [595–596](#), [602](#)
- maps and Map interface, [347](#)
  - description, [349](#)
  - methods, [393](#)
  - overview, [353–354](#)
  - working with, [379–382](#)
- mapToDouble() method, [561–562](#), [570](#)
- mapToInt() method, [570](#)
- mapToLong() method, [570](#)
- mapToObj() method, [570](#)
- matching methods, [577–580](#)
- max() method
  - optionals, [572](#)
  - streams, [565](#), [571](#)
- maxBy() method, [600](#), [603](#)

member modifiers, nonaccess, [29–36](#)

members

access. *See* access and access modifiers

declaring, [16](#)

key points, [58–59](#)

META-INF/services/java.sql.Driver file, [832](#)

metadata for ResultSets, [848–850](#), [868–873](#)

method-local inner classes, [458–459](#)

key points, [476–477](#)

working with, [459–460](#)

methods

abstract, [31–35](#)

access modifiers. *See* access and access modifiers

anonymous inner classes, [462](#)

assertions, [176–178](#)

enums, [50–51](#)

final, [30–31](#), [45](#)

generics, [407–419](#), [424–426](#)

instance, [86](#), [90](#)

interface implementation, [105–106](#)

interfaces, [14–15](#)

lambda expression references, [518–520](#)

overloaded, [94–100](#)

overridden, [87–93](#)

parameters, [278](#)

passing lambda expressions to, [494](#)

static, [47](#), [139–142](#)

synchronized, [35](#)

variable argument lists, [35–36](#)

min() method

optionals, [572](#)

streams, [565](#), [571](#)

minBy() method, [600](#), [603](#)

minusSeconds() method, [214](#)



- minusWeeks() method, [214](#)
- minusX() method, [221](#)
- MissingResourceException class, [235](#)
- mixing generic and nongeneric collections, [399](#)–404
- mkdir() method, [262](#)
- modifiers. *See* access and access modifiers
- monitors for locking, [675](#), [726](#)
- move() method, [272](#)
- moveToCurrentRow() method, [863](#), [868](#)
- moveToInsertRow() method, [863](#), [867](#)
- moving files, [272](#)–273
- moving in ResultSets, [841](#)–842, [852](#)–861
- multi-catch clauses, [179](#)–183, [194](#)
- multidimensional arrays, [44](#)
- multiple inheritance, [84](#), [110](#)–111
- multiple threads, starting and running, [654](#)–657
- multithreading, [644](#)
- mutable data, synchronizing, [680](#)
- mutators in encapsulation, [71](#)

## N

### names

- classes and interfaces, [175](#)
- constructors, [37](#), [116](#), [118](#)
- dot operator, [143](#)
- files and directories, [264](#), [299](#)
- threads, [652](#)

native threads, [645](#)

natural ordering, [394](#)

NavigableMap interface, [347](#), [382](#)

NavigableSet interface, [347](#), [382](#)

### navigation

ResultSets, [841–842](#), [852–861](#)

TreeSets and TreeMaps, [382–384](#)

negate() method, [497](#), [511–512](#)

negative numbers, representing, [38](#)

nested classes, [3](#)

inner. *See* inner classes

static, [450](#), [468–469](#)

new keyword for inner classes, [454](#)

### new thread state

description, [659](#)

starting threads, [651](#)

newCachedThreadPool() method, [752](#)

newFixedThreadPool() method, [752](#)

newLine() method, [251](#)

### next() method

Iterator, [377](#)

ResultSets, [841–842](#), [852](#), [856](#)

String, [607](#)

nextInt() method, [754](#)

### NIO.2

- attributes, [280–287](#)
- DirectoryStream, [288–289](#)
- files and directories, [270–280](#)
- files and paths, [267–268](#)
- FileVisitor, [289–293](#)
- key points, [315–316](#)
- PathMatcher, [293–297](#)
- permissions, [281–282](#)
- WatchService, [297–300](#)
- no-arg constructors, [118–119](#)
- non-static fields, synchronizing, [680–681](#)
- non-synchronized method, [675](#)
- nonaccess member modifiers, [5–6](#), [29–30](#)
  - abstract methods, [31–35](#)
  - final arguments, [31](#)
  - final methods, [30–31](#)
  - key points, [55](#)
  - methods with variable argument lists, [35–36](#)
  - synchronized methods, [35](#)
- noneMatch() method, [576–578](#), [586–587](#)
- nongeneric code, updating to generic, [398–399](#)
- nongeneric collections, mixing with generic, [399–404](#)
- normalize() method, [276–277](#)
- normalizing paths, [275–277](#)
- NoSuchElementException class, [391](#), [563](#)
- NoSuchFileException class, [272](#)
- notExists() method, [271](#), [273](#)
- notify() method
  - Class, [699](#)
  - description, [333](#)
  - key points, [704](#)
  - locks, [679](#)
  - Object, [753](#)
  - threads, [659](#), [690–692](#), [695](#), [697–699](#)

- notifyAll() method
  - Class, [699](#)
  - description, [333](#)
  - Object, [753](#)
  - threads, [659](#), [690](#), [695–699](#)
- now() method, [208](#), [218](#)
- null values
  - returning, [113](#)
  - wrappers, [362](#)
- NullPointerException class, [362](#)
- number signs (#) in properties files, [227](#)
- NumberFormat class, [222](#)
- numbers
  - primitives. *See* primitives
  - random, [754](#)

## O

- ObjDoubleConsumer interface, [504](#)
- Object class
  - collections, [396](#), [405](#)
  - description, [332](#)
  - inheritance from, [74](#)
  - threads, [659](#), [699](#)
- object graphs, [303–306](#)
- Object.notify() method, [753](#)
- Object.notifyAll() method, [753](#)
- object orientation (OO), [69–70](#)
  - benefits, [82](#)
  - casting, [101–104](#)
  - constructors. *See* constructors
  - encapsulation, [70–73](#)
  - inheritance, [74–82](#)
  - initialization blocks, [137–139](#)

- interface implementation, [105–111](#)
- overloaded methods, [94–100](#)
- overridden methods, [87–93](#)
- polymorphism, [83–87](#)
- return types, [112–114](#)
- singleton design pattern, [128–133](#)
- statics, [139–146](#)
- Object.wait() method, [753](#)
- ObjectInputStream class
  - description, [252](#)
  - working with, [258](#), [301–303](#)
- ObjectOutputStream class
  - description, [252](#)
  - working with, [258](#), [301–303](#)
- objects and object references
  - arrays, [44](#)
  - inner classes, [455–456](#)
  - overloaded methods, [96–97](#)
  - updating columns with, [865–866](#)
- ObjIntConsumer interface, [504](#)
- ObjLongConsumer interface, [504](#)
- of() method
  - dates, [209–210](#), [215](#), [221](#)
  - optionals, [576](#)
  - Stream, [548](#)
- orElse() method, [576](#)
- offer() method
  - ArrayQueue, [390](#)
  - BlockingQueue, [738](#)
  - LinkedList, [352](#)
  - PriorityQueue, [387–388](#), [393](#)
- offerFirst() method, [390](#)
- OffsetDateTime class, [207](#), [214](#), [225](#)
- ofMinutes() method, [217](#)

- ofNullable() method, [575–576](#)
- operating system thread limits, [747](#)
- operations for streams, [556–559](#)
- operators for functional interfaces, [517–518](#)
- Optional class, [573](#)
- OptionalDouble class, [562–564](#), [573](#)
- OptionalInt class, [573](#)
- OptionalLong class, [573](#)
- optionals, [562–564](#)
  - key points, [620](#)
  - working with, [572–576](#)
- or() method, [497](#), [511–512](#)
- order
  - ArrayList elements, [350–351](#)
  - collections, [384](#)
  - instance initialization blocks, [138](#)
  - natural, [394](#)
  - threads, [658](#)
- ORDERED characteristic for streams, [778](#)
- ordered collections, [349–351](#)
- OutputStream.write() method, [748](#)
- overloaded constructors, [123–128](#)
- overloaded methods, [94–95](#)
  - invoking, [96–98](#)
  - key points, [150–151](#)
  - legal, [95](#), [99](#)
  - main(), [98](#)
  - vs. overridden, [95](#), [100](#)
  - polymorphism, [98](#)
  - return types, [112](#)
- overridden methods, [87–91](#)
  - illegal, [92–93](#)
  - invoking superclass, [91–92](#)
  - vs. overloaded, [95](#), [100](#)

- @override annotation, [93–94](#)
  - polymorphism, [98](#)
  - return types, [112–113](#)
  - static, [146](#)
- @override annotation, [93–94](#), [151](#)
- overriding
  - anonymous inner class methods, [462](#)
  - equals(), [334–335](#)
  - hashCode(), [340–343](#)
  - key points, [150–151](#), [428–429](#)
  - private methods, [21–22](#)
  - run(), [647](#)

## P

- package-centric languages, [3](#)
- package-level access, [4–5](#)
- packages
  - access, [3](#)
  - assertions, [175](#)
- parallel Fork/Join Framework. *See* Fork/Join Framework
- parallel() method, [612–613](#), [767](#), [769–770](#), [779](#)
- parallel streams, [612–614](#), [766–767](#)
  - associative pipeline operations, [772](#)
  - embarrassingly parallel problems, [770–771](#)
  - findAny(), [782–783](#)
  - forEach() and forEachOrdered(), [780–782](#)
  - key points, [623](#), [796–798](#)
  - pipelines, [767–770](#)
  - RecursiveTask, [783–786](#)
  - reduce(), [786–790](#)
  - side effects, [773–774](#)
  - stateful operations, [773–778](#)
  - stateless operations, [773](#)

- unordered, [778–780](#)
- parallel tasks, identifying, [746–747](#)
- parallelStream() method, [769–770](#)
- parameterized types, [396–397](#)
- parameters
  - vs. arguments, [35–36](#)
  - lambda expressions, [491–492](#)
  - methods, [278](#)
  - multi-catch and catch, [181–182](#)
- parentheses ()
  - arguments, [31](#), [35](#)
  - lambda expressions, [493](#)
  - try-with-resources, [877–878](#)
- parse() method, [209–210](#), [221](#)
- parsing searches. *See* searches
- partitioning streams, [593–597](#)
- partitioningBy() method, [596](#), [602](#)
- passing lambda expressions to methods, [494](#)
- passwords, hard-coding, [827](#)
- PathMatcher, [293–297](#)
- paths and Path interface
  - creating, [268–270](#)
  - iterating through, [274–275](#)
  - key points, [315–316](#)
  - methods, [274–275](#)
  - normalizing, [275–277](#)
  - relativizing, [278–280](#)
  - resolving, [277–278](#)
  - retrieving, [273–275](#)
  - working with, [266–268](#)
- Paths class, [267](#), [315–316](#)
- Paths.get() method, [268–270](#)
- patterns, design
  - factory, [825–826](#)



- singleton. *See* singleton design pattern
- peek() method
  - ArrayDeque, [390](#)
  - BlockingQueue, [739](#)
  - LinkedList, [352](#)
  - parallel streams, [788](#)
  - pipelines, [767](#)
  - PriorityQueue, [387–388](#), [393](#)
  - streams, [559](#), [580](#)
- percent signs (%) for LIKE operator, [841](#)
- Periods class, [207](#), [225](#)
- periods in dates, [214–216](#)
- permissions
  - files, [281–282](#)
  - PosixFileAttributes, [286–287](#)
- pipe (|) characters for multi-catch clauses, [180](#)
- pipelines and pipeline operations
  - associative, [772](#)
  - key points, [618](#)
  - parallel streams, [767–770](#)
  - stateful operations, [773–778](#)
  - streams, [553–554](#)
- plus() method, [217](#)
- plusMinutes() method, [214](#), [217](#)
- plusX() method, [221](#)
- plusYears() method, [214](#)
- poll() method
  - ArrayDeque, [390](#)
  - BlockingQueue, [739](#)
  - LinkedList, [352](#)
  - PriorityQueue, [387–388](#), [393](#)
  - WatchService, [299](#)
- pollEvents() method, [299](#)
- pollFirst() method, [384](#)

- pollFirstEntry() method, [384](#)
- polling, [384](#)
- pollLast() method, [384](#), [391](#), [393–394](#)
- polymorphism
  - abstract classes, [8](#)
  - anonymous inner classes, [463](#)
  - arrays, [408–410](#)
  - generics, [405–407](#)
  - inheritance, [77](#)
  - key points, [149–150](#)
  - overloaded and overridden methods, [98](#)
  - overview, [83–87](#)
- pools, thread. *See* ThreadPools
- pop() method, [390–391](#), [393](#)
- Portable Operating System Interface (POSIX), [283](#)
- PosixFileAttributes interface, [283](#)
- PosixFileAttributeView interface, [283](#), [286–287](#)
- postVisitDirectory() method, [291](#)
- predicates and Predicate interface
  - description, [497–498](#)
  - functional interfaces, [498](#), [508–514](#)
  - lambda expressions, [369](#)
  - methods and description, [523](#)
  - working with, [508–514](#)
- preemptive multitasking, [749](#)
- preemptive priority-based scheduling, [664–668](#)
- preventing thread execution, [661](#)
- previous() method, [856](#), [858](#)
- preVisitDirectory() method, [291](#)
- primary database keys, [815](#), [820–822](#)
- primitives
  - arrays, [44](#)
  - declarations, [37–39](#)
  - final, [45](#)

- returning, [114](#)
- streams, [551](#)
- wrapper classes, [358–362](#)
- printf() method, [252](#)
- printing
  - file permissions, [282](#)
  - reports, [850–852](#)
- PrintInputStream class, [252](#)
- println() method, [258](#), [519–520](#)
- PrintWriter class, [252](#), [258–259](#)
- priority of threads, [658](#), [664–668](#)
- PriorityBlockingQueue class, [738](#), [795](#)
- PriorityQueue class, [347](#), [355](#), [387–388](#), [393–394](#)
- private modifiers, [16](#)
  - inner classes, [458](#)
  - overriding, [21–22](#)
  - overview, [21–22](#)
- processors, available, [751](#)
- Properties class, [226–230](#)
- properties files, [226–230](#), [239](#)
- protected modifiers, [16](#)
  - inner classes, [458](#)
  - working with, [22–27](#)
- public access, [5](#)
- public modifiers, [16](#)
  - constants, [12–13](#)
  - encapsulation, [71](#)
  - inner classes, [458](#)
  - overview, [18–20](#)
- push() method, [390](#)
- put() method, [230](#)
  - BlockingQueue, [738](#)
  - maps, [393](#)
- putIfAbsent() method, [737](#)

## Q

### queries

- key points, [882–883](#)
- result sets. *See* [ResultSets](#)
- SQL, [817–818](#)
- submitting, [833–839](#)

### query strings, [838](#)

### question marks (?)

- generics, [426](#)
- globs, [295–296](#)

### Queue interface, [347](#), [354–355](#)

### queues

- blocking, [737–741](#)
- bounded, [739](#)
- description, [349](#)
- LinkedTransferQueue, [740–741](#)
- special-purpose, [739–740](#)
- threads, [658](#)

### quotes (' , ") in SQL, [818](#)

## R

### race conditions, [672–673](#), [686–690](#)

### random numbers, [754](#)

### RandomInitRecursiveAction task, [764](#)

### range() method, [607](#), [611](#)

### ranges

- numbers, [39](#)
- streams, [609](#), [611](#)

### RDBMSs (Relational Database Management Systems), [815](#)

### read() method, [251](#), [256](#), [748](#)

### read-only objects, [736](#)

### read permissions, [281–282](#), [286](#)

- readAttributes() method, [286–287](#)
- reading
  - attributes, [280–282](#)
  - ResultSets, [842–847](#)
- readLine() method, [251](#), [260](#), [263](#)
- readObject() method, [302](#), [306–309](#)
- readPassword() method, [265](#)
- REAL data type, [847](#)
- RecursiveAction class, [761–762](#)
- RecursiveTask task, [762–764](#), [783–786](#)
- Red-Black tree structure, [353](#)
- redefined static methods, [146](#)
- reduce operations and reduce() method
  - optionals, [572](#)
  - parallel streams, [786–790](#)
  - streams, [556–557](#), [571](#), [777](#)
  - working with, [565–568](#)
- ReentrantLock class, [727–730](#)
- ReentrantReadWriteLock object, [732–733](#)
- references, [83](#)
  - casting, [101–104](#), [151](#)
  - declaring, [40](#)
  - equality, [334–335](#)
  - inner classes, [456–458](#)
  - instances, [143](#)
  - lambda expression methods, [518–520](#)
  - overloaded methods, [96–97](#)
  - returning, [113](#)
- reflexivity with equals(), [339](#)
- registering JDBC drivers, [828–830](#)
- regular expressions (regex) vs. globs, [296](#)
- regular inner classes, [453–454](#)
- Relational Database Management Systems (RDBMSs), [815](#)
- relative() method, [856–858](#)

- relative paths, [269](#)
- relativize() method, [279](#)–[280](#)
- relativizing paths, [278](#)–[280](#)
- releasing locks, [675](#)
- remove() method
  - BlockingQueue, [738](#)
  - collections, [349](#), [737](#)
  - description, [393](#)–[394](#)
  - lists, [393](#)
  - maps, [393](#)
  - sets, [393](#)
- removeFirst() method, [682](#)
- removeIf() method, [511](#)
- removeLast() method, [391](#), [394](#)
- renameTo() method, [264](#)
- renaming files and directories, [299](#)
- replace() method, [737](#)
- replaceAll() method, [515](#)–[516](#)
- reports, printing, [850](#)–[852](#)
- reset() method, [299](#)
  - barriers, [745](#)
  - WatchService, [299](#)
- resolve() method, [278](#)
- resolving paths, [277](#)–[278](#)
- resource bundles, [227](#), [231](#)–[233](#)
  - default locales, [234](#)
  - Java, [233](#)–[234](#)
  - key points, [239](#)
  - selecting, [235](#)–[237](#)
- ResourceBundle class, [231](#)–[233](#)
- resources
  - autocloseable, [185](#)–[189](#)
  - closing, [875](#)–[879](#)
- results, database, [817](#)–[818](#). *See also* ResultSets

ResultSet interface, [823–824](#)

ResultSetMetaData class, [848](#)

ResultSets

- cursor types, [853](#)

- information about, [848–850](#), [868–873](#)

- key points, [882–883](#)

- moving in, [841–842](#), [852–861](#)

- overview, [840](#)

- reading from, [842–847](#)

- reports, [850–852](#)

- updating, [861–862](#)

resume() method, [660](#)

rethrowing exceptions, [182–184](#)

retrieving path information, [273–275](#)

return type

- constructors, [37](#), [116](#), [118](#)

- declarations, [112–114](#)

- key points, [151–152](#)

- lambda expressions, [492](#)

- overloaded methods, [94](#), [112](#)

- overridden methods, [90](#), [112–113](#)

- returning values, [113–114](#)

reuse, inheritance for, [76](#)

reuseless code, [450](#)

reverse() method, [392](#)

reversed() method, [582–583](#)

reverseOrder() method, [392](#)

ride.in property, [236](#)

rowDeleted() method, [865](#)

rows

- database, [814–815](#)

- inserting, [866–868](#)

rowUpdated() method, [863](#)

rs.getObject() method, [852](#)

- rules
  - constructors, [118](#)–[119](#)
  - expression, [172](#)–[173](#)
- run() method
  - overriding, [647](#)
  - Runnable, [699](#)
  - threads, [646](#), [651](#)–[652](#), [656](#)
- Runnable interface
  - executing, [746](#)
  - I/O activities, [754](#)
  - implementing, [648](#)
  - threads, [647](#), [699](#)
- runnable thread state, [659](#)
- running thread state
  - description, [659](#)–[660](#)
  - threads, [659](#)
- running with assertions, [174](#)
- runtime
  - disabling assertions at, [174](#)
  - enabling assertions at, [174](#)
- Runtime class, [751](#)
- runtime exceptions for overridden methods, [90](#)

## S

- scheduled thread pools, [752](#)
- ScheduledThreadPoolExecutor, [752](#)
- scheduler, thread, [658](#)
- schema, database, [819](#)–[822](#)
- scope of lambda expressions, [495](#), [506](#)
- searches
  - arrays and collections, [373](#)–[375](#)
  - files, [264](#)
  - key points, [431](#), [621](#)–[622](#)



- with streams, [576–580](#), [586–588](#)
- TreeSets and TreeMaps, [382–383](#)
- SELECT operation
  - description, [817–818](#)
  - SELECT \* FROM, [833](#)
  - SELECT with WHERE, [817](#)
- semicolons (;)
  - abstract methods, [8](#), [31](#)
  - anonymous inner classes, [462–463](#)
  - enums, [49](#)
- separation of concerns principle, [745](#)
- sequential() method, [770](#)
- Serializable interface, [302](#), [309](#)
- serialization, [301](#)
  - collections, [313](#)
  - inheritance, [309–312](#)
  - key points, [317](#)
  - object graphs, [303–306](#)
  - ObjectOutputStream and ObjectInputStream, [301–303](#)
  - static variables, [313](#)
  - transient variables, [345–346](#)
  - writeObject() and readObject(), [306–309](#)
- serialization process, [252](#)
- setAttribute() method, [286](#)
- setLastModified() method, [282](#)
- setName() method, [657](#)
- setPosixFilePermissions() method, [287](#)
- setPriority() method
  - key points, [666](#), [702](#)
  - threads, [658](#)
- setProperty() method, [230](#)
- sets and Set interface, [347](#)
  - characters in globs, [295](#)
  - description, [349](#)

- methods, [393](#)
- overview, [352–353](#), [378–379](#)
- setters encapsulation, [71](#)
- setTimes() method, [284](#), [287](#)
- shadowed variables, [42](#)
- short-circuiting operations, [576](#)
- short type
  - ranges, [39](#)
  - wrappers, [361](#)
- shutdown of ExecutorService, [754–755](#)
- shutdownNow() method, [755](#)
- side effects
  - assertions, [178–179](#)
  - lambda expressions, [506–507](#)
  - parallel streams, [773–774](#)
  - streams, [588](#), [593](#)
- signalAll() method, [732](#)
- signatures of methods, [100](#), [106](#)
- signed numbers, [38](#)
- SimpleFileVisitor class, [289–290](#)
- single thread pools, [752](#)
- singleton design pattern, [128](#)
  - benefits, [133](#)
  - description, [129](#)
  - key points, [153](#)
  - problem, [129–130](#)
  - solution, [130–133](#)
- size
  - arrays, [45](#)
  - numbers, [39](#)
- size() method
  - collections, [349](#), [737](#)
  - lists, [378](#), [393](#)
  - maps, [393](#)

- sets, [393](#)
- skip() method, [611](#), [777](#), [780](#)
- SKIP\_SIBLINGS result type, [292](#)
- SKIP\_SUBTREE result type, [292](#)
- slashes (/) for globs, [294](#)–[295](#)
- sleep() method
  - key points, [702](#)
  - locks, [679](#)
  - overview, [661](#)–[664](#)
  - threads, [646](#), [658](#), [668](#), [699](#)
  - working with, [664](#)
- sleeping thread state, [664](#)
  - description, [660](#)
  - overview, [661](#)–[664](#)
- sort() method
  - arrays, [372](#)–[373](#), [392](#)
  - collections, [363](#)–[365](#), [370](#)–[372](#), [392](#)
- SORTED characteristic for streams, [778](#)
- sorted collections, [349](#)–[351](#)
- sorted() method, [580](#)–[582](#), [587](#), [777](#), [780](#)
- SortedMap interface, [347](#)
- SortedSet interface, [347](#)
- sorting
  - arrays, [372](#)–[373](#), [392](#)
  - collections, [363](#)–[372](#), [392](#)
  - Comparable interface, [365](#)–[367](#)
  - Comparator interface, [367](#)–[368](#)
  - key points, [431](#), [621](#)–[622](#)
  - streams, [580](#)–[588](#)
- source operations for pipelines, [553](#)
- sources of streams, [588](#)–[589](#)
- spaces in natural ordering, [394](#)
- special-purpose queues, [739](#)–[740](#)
- special relationships in inner classes, [452](#), [468](#)

- split() method, [604](#)
- spliterator() method, [779](#)
- spontaneous wakeup, [698](#)
- spreadsheets. *See* databases
- SQL (Structured Query Language), [815](#)–[816](#)
  - closing resources, [875](#)–[879](#)
  - injection attacks, [838](#)
  - queries, [817](#)–[818](#)
  - types, [843](#)
- SQLException class
  - driver classes, [830](#)
  - ResultSets, [859](#), [867](#)–[868](#)
- SQLWarning class, [874](#)–[875](#)
- square brackets ([]) for array elements, [44](#)
- stack
  - local variables, [41](#)
  - threads, [747](#)
- StandardWatchEventsKinds class, [298](#)–[299](#)
- start() method
  - alternative, [746](#)
  - threads, [646](#), [650](#), [654](#)–[656](#), [699](#)
- starting threads, [644](#)–[646](#), [651](#)–[659](#)
- starvation of threads, [686](#), [704](#)
- stateful operations in parallel streams, [773](#)–[778](#)
- stateless operations in parallel streams, [773](#)
- Statement interface, [823](#)–[824](#), [835](#)
- statements
  - constructing and using, [836](#)–[839](#)
  - databases, [816](#)
  - description, [835](#)
- states
  - key points, [701](#)–[703](#)
  - threads, [659](#)–[661](#)
- static constants, [12](#)–[13](#)

- static fields, synchronizing, [680–681](#)
- static initialization blocks, [138](#)
- static nested classes, [450](#)
  - key points, [477–478](#)
  - overview, [468–469](#)
- static variables and methods, [47](#)
  - constructors, [119](#)
  - inheritance, [75](#)
  - inner classes, [458](#), [460–461](#)
  - interfaces, [14–15](#)
  - key points, [60](#), [154](#)
  - locks, [679](#)
  - overriding, [91](#), [146](#)
  - overview, [139–142](#)
  - references, [520](#)
  - serialization, [313](#)
  - singleton design pattern, [131](#)
  - synchronizing, [677](#)
- stop() method, [660](#)
- store() method, [230](#)
- stored procedures
  - information about, [871–872](#)
  - invoking, [837](#)
- stream classes, [252](#)
- Stream interface, [770](#)
  - methods, [570–571](#), [601](#)
  - optionals, [572](#)
- stream() method, [543](#), [547–548](#), [604](#)
- streams
  - accumulators, [567–570](#)
  - from arrays, [548](#)
  - benefits, [552–553](#)
  - from collections, [546–547](#)
  - collectors, [600–603](#)

- counting, [599](#)
- creation methods, [551](#)–552
- element existence, [577](#)–578
- from files, [549](#)–551
- generating, [606](#)–611
- grouping and partitioning, [593](#)–597
- joining, [599](#)–600
- key points, [617](#)–619
- lazy, [555](#)–556
- map-filter-reduce operations, [560](#)–564, [570](#)–572
- maxBy(), [600](#)
- minBy(), [600](#)
- of(), [548](#)
- operations, [556](#)–559
- optionals, [572](#)–576
- overview, [542](#)–546
- parallel. *See* parallel streams
- pipelines, [553](#)–554
- primitives, [551](#)
- reduce operation, [565](#)–568
- searching with, [576](#)–580, [586](#)–588
- sorting, [580](#)–588
- sources, [588](#)–589
- of streams, [603](#)–606, [623](#)
- summing and averaging, [597](#)–599
- values from, [589](#)–593

strictfp modifiers

- classes, [5](#)–6
- inner classes, [458](#)

StringBuffer class, [681](#)

StringBuilder class, [681](#)

strings

- converting to URIs, [269](#)
- immutable, [134](#)

- Structured Query Language (SQL), [815–816](#)
  - closing resources, [875–879](#)
  - injection attacks, [838](#)
  - queries, [817–818](#)
  - types, [843](#)
- subclasses
  - anonymous inner classes, [464–465](#)
  - concrete, [32–33](#)
- subdirectories, [289–293](#)
- subdividing tasks, [756–757](#)
- subMap() method, [385–386](#)
- submitting queries, [833–839](#)
- subpath() method, [274](#)
- subSet() method, [386](#)
- subtypes for reference variables, [83](#)
- sum() method, [565](#), [571](#)
- summingInt() method, [597](#), [603](#)
- super() calls for constructors, [126](#)
- super constructor arguments, [120](#)
- superclasses
  - constructors, [118](#)
  - overridden methods, [91–92](#)
  - Serializable, [309](#)
- suppliers
  - functional interfaces, [498–503](#)
  - methods and description, [523](#)
- supportsANSI92EntryLevelSQL() method, [869](#), [873](#)
- suppressed exceptions, [190–192](#)
- suspend() method, [660](#)
- symmetry of equals(), [339](#)
- synchronization
  - blocks, [675–678](#)
  - code, [668–673](#)
  - key points, [703](#)

- locks, [674–677](#)
- methods, [35](#), [132](#), [674–675](#)
- need for, [679–681](#)
- static methods, [677](#)
- synchronizedList() method, [682–684](#), [734](#)
- SynchronousQueue class, [738–740](#), [795](#)
- System class, [226](#)
- System.exit() method, [755](#)
- system resources for threads, [747](#)

## T

- tables. *See* databases
- tailMap() method, [386](#)
- tails of queues, [391](#)
- tailSet() method, [386](#)
- take() method
  - BlockingQueue, [739](#)
  - LinkedTransferQueue, [740](#)
  - WatchService, [299](#)
- tasks
  - CPU-intensive vs. I/O-intensive, [748](#)
  - decoupling from threads, [749–751](#)
  - subdividing, [756–757](#)
- TemporalAdjusters class, [207](#), [214](#)
- terminal operations
  - pipelines, [553](#)
  - streams, [553](#)
- TERMINATE result type, [292](#)
- test database overview, [819–822](#)
- test() method, [497–498](#), [508–511](#)
- thenComparing() method, [582–583](#)
- this() calls for constructors, [118](#), [126](#)
- this keyword, [43](#), [456–457](#)



## Thread class

- description, [644](#)
- extending, [647](#)–[648](#)
- methods, [646](#), [658](#), [699](#)
- thread methods, [699](#)

`Thread.currentThread().isInterrupted()` method, [755](#)

`Thread.interrupt()` method, [755](#)

thread-safe classes, [681](#)–[684](#)

thread-safe code, [132](#)

thread-safe collections, [736](#)

`ThreadLocalRandom` class, [754](#)

`ThreadPoolExecutor`, [752](#)

ThreadPools, [745](#)–[746](#)

- cached, [751](#)
- fixed, [751](#)
- scheduled, [752](#)
- single, [752](#)

## threads

- blocked code, [678](#)–[679](#)
- concurrency. *See* concurrency
- creating and putting to sleep, [664](#)
- deadlocks, [684](#)–[685](#)
- decoupling from tasks, [749](#)–[751](#)
- defining, [647](#)–[648](#)
- exercise, [678](#)
- instantiating, [644](#)–[646](#), [649](#)–[651](#)
- interaction, [690](#)–[695](#)
- key points, [701](#)–[704](#)
- limits, [747](#)
- livelocks, [685](#)–[686](#)
- locks, [678](#)–[679](#)
- making, [646](#)–[647](#)
- multiple, [654](#)–[657](#)
- names, [652](#)

- notifyAll(), [695–699](#)
- overview, [644–646](#)
- preventing execution, [661](#)
- priorities, [664–668](#)
- race conditions, [686–690](#)
- scheduler, [658](#)
- sleeping state, [661–664](#)
- starting, [644–646](#), [651–659](#)
- starvation, [686](#)
- states and transitions, [659–661](#)
- synchronization and locks, [674–677](#)
- synchronization need, [679–681](#)
- synchronization of code, [668–673](#)
- thread-safe classes, [681–684](#)
- threads of execution, [644](#), [648](#), [650](#)
- three-dimensional arrays, [44–45](#)
- TIME data type, [847](#)
- time-slicing scheduler, [664](#)
- times
  - adjustments, [213–214](#)
  - classes, [224–226](#)
  - examples, [219–220](#)
  - formatting, [220–221](#)
  - java.time.\* classes, [208–210](#)
  - key points, [238–239](#)
  - locales, [222–224](#)
  - zoned, [211–213](#)
- TIMESTAMP data type, [847](#)
- toArray() method
  - lists, [375–376](#), [393](#)
  - sets, [378](#), [393](#)
- toCollection() method, [591](#), [596](#), [602](#)
- toConcurrentMap() method, [790](#)
- ToDoubleBiFunction interface, [517](#)

- ToDoubleFunction interface, [517](#), [561](#)
- toInstant() method, [213](#), [218](#)
- ToIntBiFunction interface, [517](#)
- ToIntFunction interface, [517](#), [597](#)
- toList() method, [590](#), [596](#), [602](#)
- ToLongBiFunction interface, [517](#)
- ToLongFunction interface, [517](#)
- toMap() method
  - Collectors, [602](#)
  - parallel streams, [790](#)
- top-level classes, [55](#), [450](#)
- toSet() method
  - Collectors, [602](#)
  - parallel streams, [790](#)
- toString() method
  - arrays, [392](#)
  - null tests, [852](#)
  - objects, [866](#)
  - overview, [333](#)–[334](#)
  - Path, [274](#)
- transfer() method, [740](#)
- transient variables
  - overview, [46](#)–[47](#)
  - serialization, [345](#)–[346](#)
- transitions with threads, [659](#)–[661](#)
- transitivity of equals(), [339](#)
- tree structures, [275](#), [354](#)
- TreeMap class, [347](#)
  - methods, [386](#)
  - navigating, [382](#)–[384](#)
  - overview, [354](#)
- TreeSet class, [347](#)
  - creating, [378](#)
  - methods, [386](#)

- navigating, [382–384](#)
- overview, [354](#)
- try and catch feature
  - file I/O, [254](#)
  - finally, [179–183](#)
  - key points, [194](#)
  - multi-catch clauses, [179–183](#)
- try-with-resources feature
  - autocloseable resources with, [185–189](#), [194](#)
  - working with, [877–878](#)
- tryLock() method, [728–730](#)
- two-dimensional arrays, [44–45](#)
- TYPE\_FORWARD\_ONLY cursor type, [852–853](#), [874](#)
- type-safe arrays, [394–395](#)
- TYPE\_SCROLL\_INSENSITIVE cursor type, [853–854](#), [863](#)
- TYPE\_SCROLL\_SENSITIVE cursor type, [853](#), [874](#)
- types
  - casting. *See* casts
  - erasure, [403](#)
  - parameters, [396–397](#)
  - return. *See* return type

## U

- UML (Unified Modeling Language), [82](#), [132](#)
- UnaryOperator interface, [517–518](#), [523](#)
- unboxing problems, [405](#)
- unchecked exceptions, [90](#)
- Unicode characters, [39](#)
- Unified Modeling Language (UML), [82](#), [132](#)
- unique Map keys, [353](#)
- unordered collections, [349](#)
- unordered() method, [770](#), [779](#), [788](#)
- unordered parallel streams, [778–780](#)

- unpredictability of threads, [655–656](#), [665](#)
- unsorted collections, [349–350](#)
- UnsupportedOperationException class, [736](#)
- upcasting, [103](#)
- UPDATE operation, [818](#)
- updateFloat() method, [862](#)
- updateObject() method, [865–866](#)
- updateRow() method, [862–863](#)
- updating
  - nongeneric code to generic, [398–399](#)
  - ResultSets, [861–862](#)
  - SQL, [818](#)
- upper case
  - natural ordering, [394](#)
  - SQL commands, [818](#)
- URIs, converting strings to, [269](#)
- URLs, JDBC, [830–832](#)
- usernames, hard-coding, [827](#)
- UTC (Coordinated Universal Time), [211–212](#)

## V

- valueOf() method, [866](#)
- values
  - key points, [622–623](#)
  - from streams, [589–593](#)
- values() method, [51](#)
- var-args
  - key points, [59](#)
  - methods, [35–36](#)
- VARCHAR data type, [847](#)
- variable argument lists, [35–36](#)
- variables
  - access. *See* access and access modifiers

- atomic, [722–725](#)
- declarations, [37–39](#), [59–60](#)
- enums, [50–51](#)
- final, [45](#)
- inner classes, [459](#)
- instance, [40–41](#)
- lambda expressions, [494–496](#)
- local. *See* local variables
- primitives, [37–39](#)
- static, [47](#), [139–142](#)
- transient, [46–47](#)
- Vector class, [347](#), [352](#)
- versions of JDBC drivers, [832–833](#)
- vertical bars (|) for multi-catch clauses, [180](#)
- visibility, access, [4](#), [29](#)
- visitFile() method, [290–291](#)
- visitFileFailed() method, [291](#)
- void return type, [114](#)
- volatile keyword for threads, [689](#)

## W

- wait() method
  - Class, [699](#)
  - description, [333](#)
  - key points, [704](#)
  - locks, [679](#)
  - loops, [697–699](#)
  - Object, [753](#)
  - threads, [659](#), [690–695](#)
- waiting thread state, [660](#)
- walkFileTree() method, [289–290](#)
- warnings
  - vs. fails, [402](#)

- JDBC, [873–879](#)
- WatchKeys, [299](#)
- WatchService, [297–300](#)
- weakly consistent iterators, [737](#)
- wildcards
  - generics, [426](#)
  - globs, [294–296](#)
  - LIKE operator, [841](#)
- with() method, [221](#)
- work stealing, [759–760](#)
- wrapped classes
  - I/O, [258](#)
  - overview, [251–252](#)
  - primitives, [358–362](#)
- write() method, [251](#), [256](#), [258](#), [748](#)
- write permissions, [281–282](#), [286](#)
- writeObject() method, [302](#), [306–309](#)
- Writer class, [260](#)
- writing attributes, [280–282](#)

## X

- XML, [843](#)
- XOR (exclusive-OR) operator, [343](#)
- Xss1024k option, [747](#)

## Y

- yield() method
  - key points, [702](#)
  - locks, [679](#)
  - overview, [664–668](#)
  - threads, [646](#), [658](#), [699](#)

## Z

zoned dates and times, [207](#), [211–213](#)

ZonedDateTime class, [207](#), [212–214](#), [218](#), [226](#)

ZonedRules class, [212–214](#)





# Beta Test Oracle Software

Get a first look at our newest products—and help perfect them. You must meet the following criteria:

- ✓ Licensed Oracle customer or Oracle PartnerNetwork member
- ✓ Oracle software expert
- ✓ Early adopter of Oracle products

**Please apply at: [pdpm.oracle.com/BPO/userprofile](http://pdpm.oracle.com/BPO/userprofile)**

**ORACLE®**

If your interests match upcoming activities, we'll contact you. Profiles are kept on file for 12 months.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates.



# Join the Largest Tech Community in the World



Download the latest software, tools,  
and developer templates



Get exclusive access to hands-on  
trainings and workshops



Grow your professional network through  
the Oracle ACE Program



Publish your technical articles – and  
get paid to share your expertise

**Join the Oracle Technology Network**  
**Membership is free. Visit [community.oracle.com](http://community.oracle.com)**

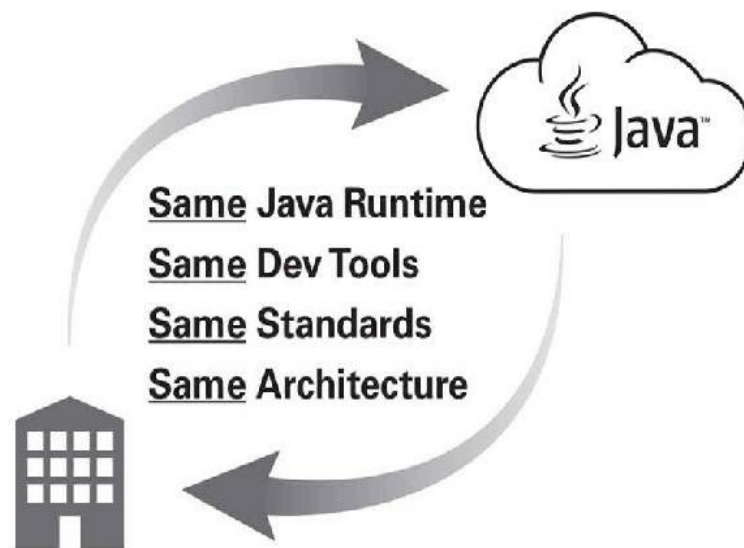
@OracleOTN [facebook.com/OracleTechnologyNetwork](https://facebook.com/OracleTechnologyNetwork)

ORACLE®



# Push a Button

## Move Your Java Apps to the Oracle Cloud



**... or Back to Your Data Center**

**ORACLE®**

[cloud.oracle.com/java](https://cloud.oracle.com/java)

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates.



# **Oracle Learning Library**

## **Created by Oracle Experts**

## **FREE for Oracle Users**

- ✓ Vast array of learning aids
- ✓ Intuitive & powerful search
- ✓ Share content, events & saved searches
- ✓ Personalize your learning dashboard
- ✓ Find & register for training events

**ORACLE®**

[oracle.com/oll](https://oracle.com/oll)

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates.







## Reach More than 640,000 Oracle Customers with Oracle Publishing Group

Connect with the Audience that Matters Most to Your Business

**ORACLE**  
MAGAZINE

### Oracle Magazine

The Largest IT Publication in the World

Circulation: 325,000

Audience: IT Managers, DBAs, Programmers, and Developers

**PROFIT** ORACLE

### Profit

Business Insight for Enterprise-Class Business Leaders to Help Them Build a Better Business Using Oracle Technology

Circulation: 90,000

Audience: Top Executives and Line of Business Managers

**Java**  
magazine

### Java Magazine

The Essential Source on Java Technology, the Java Programming Language, and Java-Based Applications

Circulation: 225,000 and Growing Steadily

Audience: Corporate and Independent Java Developers, Programmers, and Architects



For more information  
or to sign up for a FREE  
subscription: Scan the  
QR code to visit Oracle  
Publishing online.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

**ORACLE®**

## Single User License Terms and Conditions

Online access to the digital content included with this book is governed by the McGraw-Hill Education License Agreement outlined next. By using this digital content you agree to the terms of that license.

**Access** To register and activate your Total Seminars Training Hub account, simply follow these easy steps.

1. Go to [hub.totalsem.com/mheclaim](http://hub.totalsem.com/mheclaim).
2. To Register and create a new Training Hub account, enter your email address, name, and password. No further information (such as credit card number) is required to create an account.
3. If you already have a Total Seminars Training Hub account, select “Log in” and enter your email and password.
4. Enter your Product Key: **m2gw-4nj6-k3zd**
5. Click to accept the user license terms.
6. Click “Register and Claim” to create your account. You will be taken to the Training Hub and have access to the content for this book.

**Duration of License** Access to your online content through the Total Seminars Training Hub will expire one year from the date the publisher declares the book out of print.

Your purchase of this McGraw-Hill Education product, including its access code, through a retail store is subject to the refund policy of that store.

The Content is a copyrighted work of McGraw-Hill Education and McGraw-Hill Education reserves all rights in and to the Content. The Work is © 2018 by McGraw-Hill Education, LLC.

**Restrictions on Transfer** The user is receiving only a limited right to use the Content for user’s own internal and personal use, dependent on purchase and continued ownership of this book. The user may not reproduce, forward, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish, or sublicense the Content or in any way commingle the Content

with other third-party content, without McGraw-Hill Education's consent.

**Limited Warranty** The McGraw-Hill Education Content is provided on an "as is" basis. Neither McGraw-Hill Education nor its licensors make any guarantees or warranties of any kind, either express or implied, including, but not limited to, implied warranties of merchantability or fitness for a particular purpose or use as to any McGraw-Hill Education Content or the information therein or any warranties as to the accuracy, completeness, currentness, or results to be obtained from, accessing or using the McGraw-Hill Education content, or any material referenced in such content or any information entered into licensee's product by users or other persons and/or any material available on or that can be accessed through the licensee's product (including via any hyperlink or otherwise) or as to non-infringement of third-party rights. Any warranties of any kind, whether express or implied, are disclaimed. Any material or data obtained through use of the McGraw-Hill Education content is at your own discretion and risk and user understands that it will be solely responsible for any resulting damage to its computer system or loss of data.

Neither McGraw-Hill Education nor its licensors shall be liable to any subscriber or to any user or anyone else for any inaccuracy, delay, interruption in service, error or omission, regardless of cause, or for any damage resulting therefrom.

In no event will McGraw-Hill Education or its licensors be liable for any indirect, special or consequential damages, including but not limited to, lost time, lost money, lost profits or good will, whether in contract, tort, strict liability or otherwise, and whether or not such damages are foreseen or unforeseen with respect to any use of the McGraw-Hill Education content.