

O'REILLY®

Java 8

Leksykon kieszonkowy

POZNAJ NOWOŚCI W JĘZYKU JAVA 8!



HELION

Robert Liguori
Patricia Liguori

Tytuł oryginału: Java 8 Pocket Guide

Tłumaczenie: Robert Górczyński

ISBN: 978-83-246-9628-4

© 2014 Helion S.A.

Authorized Polish translation of the English edition Java 8 Pocket Guide ISBN 9781491900864 © 2014 Gliesian, LLC.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/jav8lk>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

Wprowadzenie	11
<hr/>	
Część I Język	15
Rozdział 1. Konwencje nazw	17
Nazwy klas	17
Nazwy interfejsów	17
Nazwy metod	18
Nazwy egzemplarzy i zmiennych statycznych	18
Nazwy parametrów i zmiennych lokalnych	18
Nazwy parametrów typów ogólnych	18
Nazwy stałych	19
Nazwy typów wyliczeniowych	19
Nazwy pakietów	19
Nazwy adnotacji	20
Nazwy akronimów	20
Rozdział 2. Elementy leksykalne	21
Unicode i ASCII	21
Komentarze	23
Słowa kluczowe	24
Identyfikatory	24

Separatory	25
Operatory	26
Literały	26
Sekwencje sterujące	30
Symbole walut w Unicode	31
Rozdział 3. Typy proste	33
Typy podstawowe	33
Literały dla typów podstawowych	33
Encje zmiennoprzecinkowe	35
Promocja liczbowa dla typów podstawowych	37
Klasy opakowujące	39
Automatyczne pakowanie i rozpakowywanie	39
Rozdział 4. Typy odnośnikowe	43
Porównanie typów odnośnikowych i podstawowych	44
Wartości domyślne	44
Konwersja typów odnośnikowych	46
Konwersja między typami podstawowymi i odnośnikowymi	47
Przekazanie typu odnośnikowego metodzie	47
Porównywanie typów odnośnikowych	48
Kopiowanie typów odnośnikowych	51
Alokacja pamięci i usuwanie nieużytków w przypadku typów odnośnikowych	52
Rozdział 5. Programowanie zorientowane obiektowo	53
Klasy i obiekty	53
Zmiennej długości lista argumentów	59
Klasy i metody abstrakcyjne	60
Statyczne dane składowe, metody, stałe i inicjalizatory	61
Interfejsy	63
Wyliczenia	63
Typy adnotacji	64
Interfejsy funkcjonalne	66

Rozdział 6. Polecenia i bloki	67
Polecenie w postaci wyrażenia	67
Polecenie puste	68
Blok	68
Polecenia warunkowe	68
Polecenia iteracji	70
Transfer kontroli	72
Polecenie synchronized	73
Polecenie assert	73
Polecenia obsługi wyjątków	74
Rozdział 7. Obsługa wyjątków	75
Hierarchia wyjątku	75
Sprawdzone i niesprawdzone wyjątki oraz błędy	76
Najczęstsze sprawdzone i niesprawdzone wyjątki oraz błędy	77
Słowa kluczowe związane z obsługą wyjątków	79
Proces obsługi wyjątków	84
Zdefiniowanie własnej klasy wyjątku	84
Wyświetlanie informacji o wyjątku	85
Rozdział 8. Modyfikatory w Javie	87
Modyfikatory dostępu	88
Inne modyfikatory (nie dotyczące dostępu)	88
<hr/>	
Część II Platforma	91
Rozdział 9. Java SE	93
Najczęściej używane biblioteki API Javy SE	93
Rozdział 10. Podstawy programowania	105
JRE	105
JDK	105
Struktura programu w Javie	106
Narzędzia wiersza poleceń	108
Classpath	114

Rozdział 11. Zarządzanie pamięcią	115
Rodzaje mechanizmów usuwania nieużytków	115
Narzędzia przeznaczone do zarządzania pamięcią	117
Opcje w wierszu poleceń	118
Zmiana wielkości stosu wirtualnej maszyny Javy	121
Przestrzeń Metaspace	121
Współpraca z mechanizmem GC	121
Rozdział 12. Podstawowe wejście i wyjście	123
Standardowe strumienie in, out i err	123
Hierarchia klas dla podstawowego wejścia i wyjścia	124
Odczyt i zapis pliku	124
Odczyt i zapis gniazda	127
Serializacja	128
Tworzenie archiwum ZIP i rozpakowywanie plików	129
Rozdział 13. Nowe API wejścia-wyjścia (NIO.2)	131
Interfejs Path	131
Klasa Files	132
Funkcje dodatkowe	133
Rozdział 14. Współbieżność	135
Tworzenie wątków	135
Stany wątku	136
Priorytety wątku	136
Najczęściej używane metody dotyczące wątków	137
Synchronizacja	138
Narzędzia współbieżności	139
Rozdział 15. Framework Collections	143
Interfejs Collection	143
Implementacje	144
Metody frameworka Collection	144
Algorytmy klasy Collections	145
Efektywność algorytmu	145
Interfejs funkcjonalny Comparator	146

Rozdział 16. Framework Generics	149
Klasy i interfejsy frameworka Generics	149
Konstruktory wykorzystujące framework Generics	150
Zasada zastępowania	151
Parametry typu, znaki wieloznaczne i granice	151
Zasada get i put	152
Specjalizacja typów generycznych	153
Metody frameworka Generics w niezmodyfikowanych typach	154
Rozdział 17. API skryptowe Javy	155
Języki skryptowe	155
Implementacje silnika skryptów	155
Konfiguracja języków skryptowych i silników	157
Rozdział 18. API daty i godziny	161
Wsteczna zgodność	162
Kalendarze regionalne	162
Kalendarz ISO	162
Rozdział 19. Wyrażenia lambda	169
Podstawy wyrażen lambda	169
Interfejsy funkcjonalne specjalnego przeznaczenia	171
Interfejsy funkcjonalne ogólnego przeznaczenia	172
Zasoby dotyczące wyrażen lambda	174
<hr/>	
Część III Dodatki	175
A API Fluent	177
B Narzędzia firm trzecich	179
C Podstawy UML	189
Skorowidz	199

Współbieżność

Wątki w Javie pozwalają na znacznie efektywniejsze użycie wielu procesorów lub wielu rdzeni w pojedynczym procesorze. W przypadku pojedynczego procesora wątki zapewniają możliwość równoległego wykonywania operacji, na przykład wejścia-wyjścia i przetwarzania.

Programowanie wielowątkowe w Javie jest obsługiwane dzięki funkcjom oferowanym przez klasę `Thread` i interfejs `Runnable`.

Tworzenie wątków

Wątki mogą być tworzone na dwa sposoby: przez rozszerzenie klasy `java.lang.Thread` lub przez implementację interfejsu `java.lang.Runnable`.

Rozszerzenie klasy `Thread`

Rozszerzenie klasy `Thread` i nadpisanie metody `run()` powoduje utworzenie klasy obsługującej wątki. Uruchomienie nowego wątku jest bardzo łatwym zadaniem:

```
class Comet extends Thread {
    public void run() {
        System.out.println("Orbitowanie");
        orbit();
    }
}

Comet halley = new Comet();
halley.run();
```

Pamiętaj, że tylko jedna superklasa może być rozszerzona. Dlatego też klasa rozszerzająca `Thread` nie może rozszerzać już żadnych innych superklas.

Implementacja interfejsu `Runnable`

Implementacja interfejsu funkcjonalnego `Runnable` i zdefiniowanie jego metody `run()` również pozwala na utworzenie klasy obsługującej wątki.

```
class Asteroid implements Runnable {
    public void run() {
        System.out.println("Orbitowanie");
        orbit();
    }
}

Asteroid majaAsteroid = new Asteroid();
Thread majaThread = new Thread(majaAsteroid);
majaThread.run();
```

Pojedynczy możliwy do uruchomienia egzemplarz może być przekazany wielu obiektom wątków. Każdy wątek wykonuje to samo zadanie, jak pokazano w poniższym fragmencie kodu tuż po użyciu wyrażenia lambda:

```
Runnable asteroid = () -> {
    System.out.println("Orbitowanie");
    orbit();
};

Thread asteroidThread1 = new Thread(asteroid);
Thread asteroidThread2 = new Thread(asteroid);
asteroidThread1.run();
asteroidThread2.run();
```

Stany wątku

Wyliczenie `Thread.state` zawiera sześć stanów wątków, które zostały wymienione w tabeli 14.1.

Priorytety wątku

Prawidłowy zakres wartości priorytetu wątku to zwykle od 1 do 10, a wartością domyślną jest 5. Priorytety wątku to jeden z aspektów Javy charakteryzujący się najmniejszą przenośnością, ponieważ zakres i wartość domyślna mogą się różnić w poszczególnych wirtualnych maszynach Javy. Wartości priorytetów można pobrać za pomocą `MIN_PRIORITY`, `NORM_PRIORITY` i `MAX_PRIORITY`.

Tabela 14.1. Stany wątku

Stan wątku	Opis
NEW	Wątek, który został utworzony, ale jeszcze nie jest uruchomiony.
RUNNABLE	Wątek, który jest gotowy do uruchomienia.
BLOCKED	„Żywy” wątek, który jest zablokowany i oczekuje na blokadę monitora.
WAITING	„Żywy” wątek, który wywołuje własną metodę <code>wait()</code> lub <code>join()</code> w trakcie oczekiwania na inny wątek.
TIMED_WAITING	„Żywy” wątek, który oczekuje na inny wątek przez określony czas. Wątek uśpiony.
TERMINATED	Wątek, który zakończył działanie.

```
System.out.print(Thread.MAX_PRIORITY);
```

Wątki o niższym priorytecie oferują mniejszą wydajność niż wątki o wyższym priorytecie.

Najczęściej używane metody dotyczące wątków

W tabeli 14.2 wymieniono najczęściej używane metody dotyczące wątków i pochodzące z klasy `Thread`.

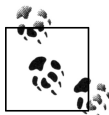
Tabela 14.2. Najczęściej używane metody klasy `Thread` dotyczące wątków

Metoda	Opis
<code>getPriority()</code>	Wartością zwrótną metody jest priorytet wątku.
<code>getState()</code>	Wartością zwrótną metody jest stan wątku.
<code>interrupt()</code>	Przerwanie pracy wątku.
<code>isAlive()</code>	Wartością zwrótną metody jest stan „życia” wątku.
<code>isInterrupted()</code>	Metoda sprawdza, czy nastąpiło przerwanie pracy wątku.
<code>join()</code>	Wątek wywołujący tę metodę musi poczekać na zakończenie działania przez wątek, który jest reprezentowany przez dany obiekt.
<code>setPriority(int)</code>	Ustawienie priorytetu wątku.
<code>start()</code>	Umieszczenie wątku w stanie gotowości do uruchomienia.

W tabeli 14.3 wymieniono najczęściej używane metody dotyczące wątków i pochodzące z klasy `Object`.

Tabela 14.3. Najczęściej używane metody klasy *Object* dotyczące wątków

Metoda	Opis
<code>notify()</code>	Metoda nakazuje obudzenie wątku i jego uruchomienie.
<code>notifyAll()</code>	Metoda nakazuje obudzenie wszystkich wątków oczekujących na inne wątki lub zasoby; następnie algorytm szeregowania wybierze jeden wątek i uruchomi go.
<code>wait()</code>	Wstrzymanie wątku w stanie oczekiwania aż do chwili, gdy inny wątek wywoła metodę <code>notify()</code> lub <code>notifyAll()</code> .



Wywołanie metody `wait()` lub `notify()` spowoduje zgłoszenie wyjątku `InterruptedException`, jeśli wywołanie metody nastąpi względem wątku, który ma ustawioną wartość `true` dla opcji wskazującej na przerwanie jego pracy.

W tabeli 14.4 wymieniono najczęściej używane metody statyczne klasy `Thread` dotyczące wątków, na przykład `Thread.sleep(1000)`.

Tabela 14.4. Metody statyczne dotyczące wątków

Metoda	Opis
<code>activeCount()</code>	Wartością zwrótną metody jest liczba wątków znajdujących się w bieżącej grupie wątków.
<code>currentThread()</code>	Metoda zwraca odniesienie do aktualnie działającego wątku.
<code>interrupted()</code>	Metoda sprawdza, czy nastąpiło przerwanie aktualnie działającego wątku.
<code>sleep(long)</code>	Zablokowanie na podaną w parametrze liczbę milisekund aktualnie działającego wątku.
<code>yield()</code>	Wstrzymanie aktualnego wątku, aby pozwolić innym wątkom na działanie.

Synchronizacja

Słowo kluczowe `synchronize` jest przeznaczone do nakładania blokad na bloki i metody. Blokada powinna zostać nałożona na blok kodu lub metodę, która próbuje uzyskać dostęp do zasobu współdzielonego o znaczeniu krytycznym. Tego rodzaju blokada monitora jest definiowana w nawiasie klamrowym. Poniżej przedstawiono kilka przykładów synchronizowanych bloków i metod.

Egzemplarz obiektu `t` wraz z synchronizowaną blokadą:

```
synchronized (t) {  
    // Polecenia bloku.  
}
```

Egzemplarz obiektu `this` wraz z synchronizowaną blokadą:

```
synchronized (this) {  
    // Polecenia bloku.  
}
```

Metoda `raise()` wraz z synchronizowaną blokadą:

```
// Pierwszy segment kodu.  
synchronized void raise() {  
    // Polecenia metody.  
}  
  
// Drugi segment kodu.  
void raise() {  
    synchronized (this) {  
        // Polecenia metody.  
    }  
}
```

Metoda statyczna `calibrate()` wraz z synchronizowaną blokadą:

```
class Telescope {  
    synchronized static void calibrate() {  
        // Polecenia metody.  
    }  
}
```



Inna nazwa blokady to *monitor* lub *mutex* (ang. *mutually exclusive lock*).

Narzędzia współbieżności to kolejny sposób na stosowanie współbieżności i zarządzanie nią.

Narzędzia współbieżności

W Javie SE 5.0 wprowadzono klasę narzędziową przeznaczoną do programowania współbieżnego. Wspomniane narzędzia znajdują się w pakiecie `java.util.concurrent` i obejmują wykonawców, współbieżne kolekcje, synchronizatory i narzędzia przeznaczone do odmierzenia czasu.

Wykonawcy

Klasa `ThreadPoolExecutor` oraz jej podklasa `ScheduledThreadPoolExecutor` implementują interfejs `Executor` zapewniający konfigurowaną i elastyczną pulę wątków. Wspomniana pula wątków pozwala komponentom serwera na wykorzystanie zalet wynikających z możliwości wielokrotnego użycia wątków.

Klasa `Executors` dostarcza metody fabryczne (tworzące obiekty) oraz narzędziowe. Spośród nich wymienione poniżej są przeznaczone do tworzenia pul wątków:

`newCachedThreadPool()`

Utworzenie nieograniczonej puli wątków, która automatycznie wielokrotnie używa wątków.

`newFixedThreadPool(int nThreads)`

Utworzenie puli wątków o stałej wielkości, która automatycznie wielokrotnie używa wątków ze współdzielonej, nieograniczonej kolejki.

`newScheduledThreadPool(int corePoolSize)`

Utworzenie puli wątków, dla której zdefiniowano polecenia wykonywane okresowo lub po upływie określonego czasu.

`newSingleThreadExecutor()`

Utworzenie wykonawcy działającego w pojedynczym wątku i operującego na nieograniczonej kolejce.

`newSingleThreadScheduledExecutor()`

Utworzenie wykonawcy działającego w pojedynczym wątku i posiadającego zdefiniowane polecenia wykonywane okresowo lub po upływie określonego czasu.

Poniżej przedstawiono fragment kodu demonstrujący użycie metody `newFixedThreadPool()`:

```
import java.util.concurrent.Executors;
import java.util.concurrent.ExecutorService;

public class ThreadPoolExample {
    public static void main() {
        // Utworzenie zadań
        // (klasa RTask implementuje interfejs Runnable).
        RTask t1 = new RTask("thread1");
        RTask t2 = new RTask("thread2");

        // Utworzenie menedżera wątków.
        ExecutorService threadExecutor = Executors.newFixedThreadPool(2);

        // Przygotowanie wątków do uruchomienia.
        threadExecutor.execute(t1);
        threadExecutor.execute(t2);

        // Zamknięcie wątków.
        threadExecutor.shutdown();
    }
}
```

Współbieżne kolekcje

Nawet jeśli typ kolekcji może być synchronizowany, najlepszym rozwiązaniem jest użycie klas współbieżności zapewniających bezpieczeństwo wątków i oferujących taką samą funkcjonalność, jak zwykłe klasy. Wspomniane klasy kolekcji i ich odpowiedniki współbieżności wymieniono w tabeli 14.5.

Tabela 14.5. Klasy kolekcji i ich odpowiedniki zapewniające bezpieczeństwo wątków

Klasa kolekcji	Odpowiednik zapewniający bezpieczeństwo wątków
HashMap	ConcurrentHashMap
TreeMap	ConcurrentSkipListMap
TreeSet	ConcurrentSkipListSet
podtypy Map	ConcurrentMap
podtypy List	CopyOnWriteArrayList
podtypy Set	CopyOnWriteArraySet
PriorityQueue	PriorityBlockingQueue
Deque	BlockingDeque
Queue	BlockingQueue

Synchronizatory

Synchronizatory to specjalnego przeznaczenia narzędzia synchronizacji. Dostępne synchronizatory zostały wymienione w tabeli 14.6.

Tabela 14.6. Synchronizatory

Synchronizator	Opis
Semaphore	Obsługuje zestaw zezwoleń.
CountDownLatch	Implementuje oczekiwanie aż do zakończenia wykonywania zestawu operacji.
CyclicBarrier	Implementuje oczekiwanie aż do usunięcia najczęściej występujących barier.
Exchanger	Implementuje punkt synchronizacji, w którym wątki mogą wymieniać elementy.

Narzędzia odmierzenia czasu

Wyliczenie `TimeUnit` jest najczęściej stosowane do informowania metod opartych na odmierzeniu czasu, jak powinien być obliczony dany parametr związany z czasem. Przykład przedstawiono w poniższym fragmencie kodu. Dostępne stałe wyliczenia `TimeUnit` zostały wymienione w tabeli 14.7.

```
// tryLock (czas podany jako typ long i jednostka TimeUnit).  
if (lock.tryLock(15L, TimeUnit.DAYS)) {...} // 15 dni.
```

Tabela 14.7. Stałe wyliczenia TimeUnit

Stała	Definicja jednostki	Jednostka (s)	Skrót
NANOSECONDS	1/1000 μ s	.000000001	ns
MICROSECONDS	1/1000 ms	.000001	μ s
MILLISECONDS	1/1000 s	.001	ms
SECONDS	s	1	sec
MINUTES	60 s	60	min
HOURS	60 min	3600	hr
DAYS	24 godziny	86400	d

Skorowidz

λE, 169–174

A

adnotacja, 64, 65

 @FunctionalInterface, 66

agregacja (UML), 195

alokacja pamięci, 52

API

 daty i godziny, 161–167

 Fluent, 161, 177, 178

 NIO.2, 131–133

 skryptowe, 155–159

archiwizacja programu Javy, 111

argument classpath, 114

ASCII, 21, 22

asercje, 73

asocjacja (UML), 195

automatyczne pakowanie, 39, 40

B

biblioteki

 API Javy SE, 93–103

 CORBA, 100

 RMI, 100

 wspomagające programowanie

 w Javie, 182, 183

blok, 68

 catch, 80–83

 finally, 80–83

 try, 80–83

błędy, 77, 79

C

camelCase, 17

D

dane składowe, 54

definiowanie własnej klasy wyjątku,
84

deserializacja obiektu, 129

domknięcia, 169–174

E

elementy leksykalne, 21–32

encje specjalne, 36, 37

encje zmiennoprzecinkowe, 35, 36

F

Fluent, 177, 178

formatowanie obiektów daty
i godziny, 166, 167

Framework Collections, 143–148
framework Generics, 149–154

G

garbage collection, 115–122
generalizacja (UML), 196
granice, 151, 152
GZIP, 129, 130

H

hermetyzacja, 53
hierarchia wyjątku, 75

I

identyfikatory, 24
implementacje silnika skryptów,
155–157
inicjalizator statyczny, 62
interfejs
Collection, 143
Comparator, 146, 147, 148
maszynowy, 164, 165
Runnable, 136
ScriptEngine, 155
Serializable, 128
interfejsy, 63
funkcjonalne, 66
ogólnego przeznaczenia, 172,
173, 174
specjalnego przeznaczenia, 171,
172
interpreter Javy, 109, 110

J

JDBC, 166
JDK, 105, 106
języki skryptowe
zgodne ze specyfikacją JSR 223, 186,
187
JRE, 105

K

kalendarz ISO, 162–164
klasa, 53, 54
abstrakcyjna, 60
BufferedInputStream, 126
BufferedOutputStream, 126
BufferedReader, 124, 127
Collections, 145, 146
DataInputStream, 125, 127
DataOutputStream, 126, 128
DateTimeFormatter, 166
Executors, 140
Files, 132
FileWriter, 126
InputStream, 124
ObjectInputStream, 129
ObjectOutputStream, 128
opakowująca, 39
OutputStream, 124
Path, 132
PrintWriter, 126, 127
PushbackInputStream, 126
Reader, 124
Thread, 135
ThreadPoolExecutor, 139
wejścia-wyjścia, 123–130
Writer, 124
ZipInputStream, 129
ZipOutputStream, 129
klonowanie, 51, 52
kolekcje, 143
komentarze, 23
kompilator Javy, 108, 109
kompozycja (UML), 195
konfiguracja
języka skryptowego, 157
silnika skryptowego, 158
konstrukcja
try-catch, 81
try-catch-finally, 82, 83
try-finally, 82
konstruktory, 56, 57

konwencje nazw, 17

konwersja

między typami podstawowymi
i odnośnikowymi, 47

typów odnośnikowych, 46, 47

kopiowanie typów odnośnikowych,
51, 52

L

literały, 26, 28, 29

dla typów podstawowych, 33–35

M

mechanizm

CMS, 117

G1, 117

Parallel, 116

Parallel Compacting, 116

Serial, 116

metaadnotacja Retention, 65

metoda, 54–56

abstrakcyjna, 61

Duration.between(), 165

equals(), 49

finalize(), 122

getAnnotation(), 65

getMessage(), 85

getPriority(), 137

getState(), 137

interrupt(), 137

isAlive(), 137

isInterrupted(), 137

join(), 137

klasy Object dotycząca wątków,
138

klasy Thread dotycząca wątków,
137

lines(), 127

main(), 107

notify(), 138

notifyAll(), 138

printf(), 60

printStackTrace(), 85

setPriority(int), 137

start(), 137

statyczna dotycząca wątków, 138

statyczna, 62

toString(), 85

typu vararg, 59

wait(), 138

write(), 128

writeInt(), 126

modyfikator

abstract, 89

default, 89

dostępu, 87, 88

final, 89

Javy, 87–90

native, 89

package-private, 88

private, 88

protected, 88

public, 88

static, 89

strictfp, 89

synchronized, 89

transient, 89

volatile, 89

moment, 166

N

nadpisywanie metod, 56

narzędzia

firm trzech wspomagające

programowanie w Javie, 179–187

javadoc, 112, 113

odmierzania czasu, 141

przeznaczone

do zarządzania pamięcią, 117,
118

wiersza poleceń, 108–114

współbieżności, 139–141

NIO, 123
NIO.2, 131–133
notacja duże O, 147

O

obiekt, 53, 54
obsługa wyjątków, 75–86
odczyt gniazda, 127, 128
odczyt plików, 124–126
okres, 165
OOP, *Patrz* programowanie
zorientowane obiektowo
operatory, 26
równości, 48
warunkowe, 38

P

parametry typu, 151, 152
platforma
 aplikacji sieciowych
 wspomagające programowanie
 w Javie, 184, 185
 Java SE, 93–103
pliki JAR, 111, 112
podklasa, 57
pola, 54
polecenie, 67–74
 assert, 73
 break, 72
 continue, 72
 do-while, 71
 for, 70
 if else if, 69
 if, 68
 if-else, 69
 iteracji, 70, 71
 obsługi wyjątków, 74
 puste, 68
 return, 73
 switch, 69
 w postaci wyrażenia, 67

warunkowe, 68, 69
while, 71
porównywanie
 ciągów tekstowych, 49
 typów odnośnikowych, 48, 49
 wyliczeń, 50
powiązanie tymczasowe (UML), 195
Przestrzeń Metaspaces, 121
priorytety wątku, 136, 137
programowanie
 wielowątkowe, 135
 zorientowane obiektowo, 53–66
promocja liczbowa, 37
przeciążanie metod, 55

R

realizacja (UML), 196
rozpakowywanie, 40

S

SAM, 66
sekwencje sterujące, 30
separatory, 25
serializacja obiektu, 128
silniki skrypcowe, 155–159
słowo kluczowe, 24
 abstract, 60
 extends, 57
 final, 62
 static, 62
 super, 58
 synchronize, 138
 synchronized, 73
 this, 58
 throw, 80
stałe statyczne, 62
stany wątku, 137
statyczne dane składowe, 61
stos, 121
 zmiana wielkości stosu
 wirtualnej maszyny Javy, 121, 122

struktura programu w Javie, 106–108
strumienie danych wejściowych
 i wyjściowych, 124, 125
superklasa, 57
synchronizacja, 138, 139
synchronizator, 141
System.err, 124
System.in, 123
System.out, 124

Ś

środowiska IDE, 183, 184

T

tablice, 45, 46
TimeUnit, 141, 142
tokeny, 21–32
tworzenie archiwum ZIP
 i rozpakowywanie plików, 129, 130
typy
 generyczne, specjalizacja, 153
 odnośnikowe, 43–52
 podstawowe, 33–41
 a typy odnośnikowe, 44
 proste, 33–41

U

UML, 189–198
 diagram klas, 189–191
 diagram obiektu, 191
 diagram pakietu, 192, 193
 diagram sekwencji, 196, 197
 relacje klas, 194–196
Unicode, 21
 symbole walut, 31, 32
usuwanie nieużytków, 115–122
 opcje w wierszu polecenia, 118–120

W

wartość domyślna, 44
wątki, 135–142
weryfikacja silnika skryptowego, 158
współbieżne kolekcje, 141
współbieżność, 135–142
wyjątek, 75
wyjątki
 niesprawdzane, 76–79
 sprawdzane, 76–78
wyczerpanie, 63
wyrażenia lambda, 169–174

X

XSD, 166

Z

zapis
 gniazda, 127, 128
 pliku, 124–126
zarządzanie pamięcią, 115–122
zasada zastępowania, 151
ZIP, 129
zmienne egzemplarza, 44, 54
zmienne lokalne, 44
znaki
 ASCII, 22, 23
 wieloznaczne, 151, 152
Zunifikowany język modelowania,
 Patrz UML

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA



Helion SA

Java 8. Leksykon kieszonkowy

Java jest dziś językiem, który programiści wybierają najczęściej, gdy mają przed sobą skomplikowany projekt, wymagający najwyższej wydajności, jakości, bezpieczeństwa oraz integracji z innymi systemami. Rozwijany od blisko dwudziestolecia, język ten doczekał się wersji oznaczonej numerem 8. Ta edycja została wzbogacona o wiele nowości, m.in. o długo oczekiwane wyrażenia lambda. Jeżeli szukasz poręcznej książki, do której możesz sięgnąć w przypadku wątpliwości, to trafieś na doskonałą pozycję!

Tę publikację, należącą do popularnej serii Leksykon kieszonkowy, możesz mieć zawsze przy sobie. W środku znajdziesz konwencje nazw oraz podstawowe elementy języka. W kolejnych rozdziałach zdobędziesz wiedzę na temat typów prostych oraz programowania zorientowanego obiektowo. Ponadto zawarto tu przystępne omówienie wyrażen lambda, współbieżności oraz zasad dostępu do plików i sieci. Java posiada rozbudowany mechanizm obsługi sytuacji wyjątkowych, który również został omówiony w tym podręczniku. Książka ta jest obowiązkową pozycją na półce każdego programisty języka Javy – jeśli chce on mieć zawsze pod ręką wiarygodne źródło informacji na temat tego języka.

Dzięki tej książce:

- poznasz elementy i składnię języka Java
- zrozumiesz zasadę działania wyrażen lambda
- wykorzystasz nowe metody dostępu do plików
- zaznajomisz się z nowościami w Javie 8
- będziesz mieć zawsze pod ręką solidne źródło wiedzy

Twój przewodnik po języku Java!

helion.pl
księgarnia
internetowa

Nr katalogowy: 25011



Księgarnia internetowa:
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900



0 601 339900



Helion

Sprawdź najnowsze promocje:

🔗 <http://helion.pl/promocje>

Książki najchętniej czytane:

🔗 <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

🔗 <http://helion.pl/novosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po **WIECEJ**



KOD KORZYŚCI

ISBN 978-83-246-9628-4



Cena 32,90 zł