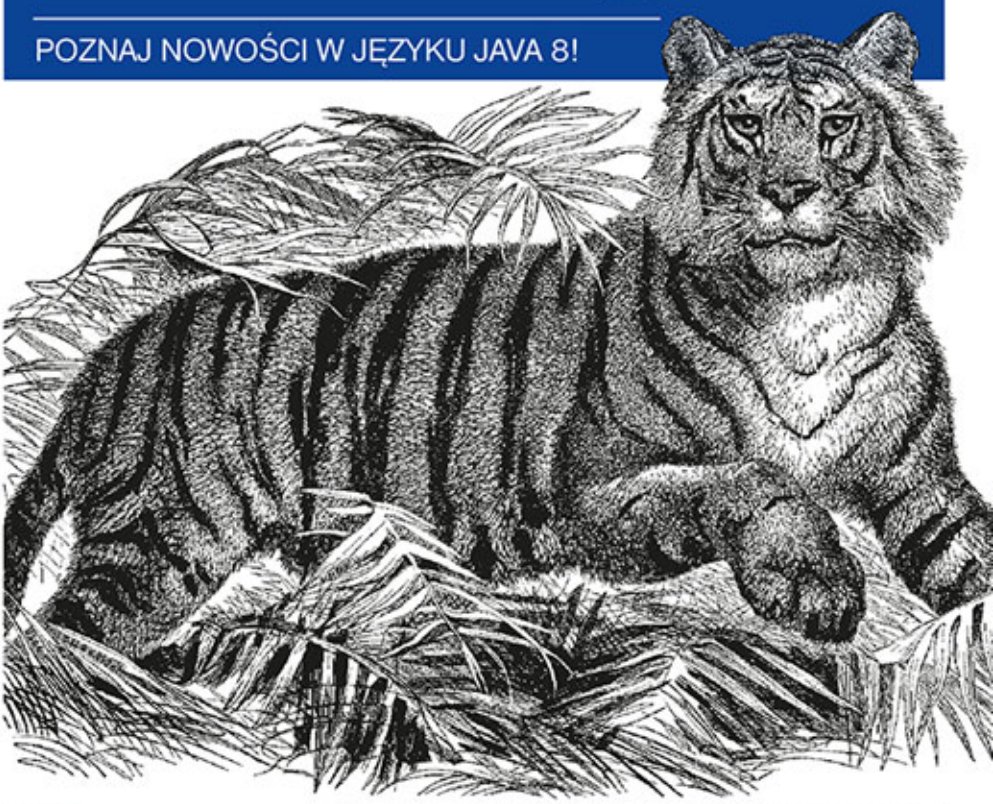
O'REILLY®

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

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.

Spis treści

ROZDZIAŁ 14.

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 `wait()` lub `join()` 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
`getPriority()`	Wartością zwrotną metody jest priorytet wątku.
`getState()`	Wartością zwrotną metody jest stan wątku.
`interrupt()`	Przerwanie pracy wątku.
`isAlive()`	Wartością zwrotną metody jest stan „życia" wątku.
`isInterrupted()`	Metoda sprawdza, czy nastąpiło przerwanie pracy wątku.
`join()`	Wątek wywołujący tę metodę musi poczekać na zakończenie działania przez wątek, który jest reprezentowany przez dany obiekt.
`setPriority(int)`	Ustawienie priorytetu wątku.
`start()`	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
`notify()`	Metoda nakazuje obudzenie wątku i jego uruchomienie.
`notifyAll()`	Metoda nakazuje obudzenie wszystkich wątków oczekujących na inne wątki lub zasoby; następnie algorytm szeregowania wybierze jeden wątek i uruchomi go.
`wait()`	Wstrzymanie wątku w stanie oczekiwania aż do chwili, gdy inny wątek wywoła metodę `notify()` lub `notifyAll()`.

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
`activeCount()`	Wartością zwrotną metody jest liczba wątków znajdujących się w bieżącej grupie wątków.
`currentThread()`	Metoda zwraca odniesienie do aktualnie działającego wątku.
`interrupted()`	Metoda sprawdza, czy nastąpiło przerwanie aktualnie działającego wątku.
`sleep(`*`long`*`)`	Zablokowanie na podaną w parametrze liczbę milisekund aktualnie działającego wątku.
`yield()`	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 odmierzania 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 `newFixed➥ThreadPool()`:

```
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.
`CyclicBarrer`	Implementuje oczekiwanie aż do usunięcia najczęściej występujących barier.
`Exchanger`	Implementuje punkt synchronizacji, w którym wątki mogą wymieniać elementy.

Narzędzia odmierzania czasu

Wyliczenie `TimeUnit` jest najczęściej stosowane do informowania metod opartych na odmierzaniu 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 µs	.000000001	ns
MICROSECONDS	1/1000 ms	.000001	µs
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

G

H

I

J

K

L

M

N

O

P

R

S

Ś

T

U

W

X

Z

O'REILLY®

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 trafiłeś 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 wyrażeń 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 wyrażeń 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/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-246-9628-4

Cena 32,90 zł

Informatyka w najlepszym wydaniu

9 788324 696284