

Nowoczesna Java w działaniu

WYRAŻENIA LAMBDA, STRUMIENIE,
PROGRAMOWANIE FUNKCYJNE I REAKTYWNE

RAOUL-GABRIEL URMA, MARIO FUSCO, ALAN MYCROFT

przełożył
KRZYSZTOF KAPUSTKA

APN Promise
Warszawa 2019

Spis treści

<i>Przedmowa</i>	<i>xv</i>
<i>Podziękowania</i>	<i>xvii</i>
<i>O książce</i>	<i>xix</i>
<i>O autorach</i>	<i>xxiv</i>
<i>O ilustracji na okładce</i>	<i>xxv</i>

CZĘŚĆ 1. PODSTAWY1

1. *Java 8, 9,10 i 11: co się dzieje?* 3

1.1	Jak to wszystko wygląda?	4
1.2	Dlaczego Java ciągle się zmienia?	6
	<i>Miejsce języka Java w ekosystemie języków programowania</i>	<i>7</i>
	<i>Przetwarzanie strumieni</i>	<i>9</i>
	<i>Przekazywanie kodu do metod za pomocą parametryzacji zachowania</i>	<i>10</i>
	<i>Równoległość i współdzielone dane modyfikowalne</i>	<i>11</i>
	<i>Java musi ewoluować</i>	<i>12</i>
1.3	Funkcje w Javie	13
	<i>Metody i wyrażenia lambda jako obywatele pierwszej kategorii</i>	<i>13</i>
	<i>Przekazywanie kodu: przykład</i>	<i>16</i>
	<i>Od przekazywania metod do wyrażen lambda</i>	<i>18</i>
1.4	Strumienie	19
	<i>Wielowątkowość jest trudna</i>	<i>21</i>
1.5	Metody domyślne i moduły Java	24
1.6	Inne dobre pomysły z programowania funkcyjnego	25

2. *Przekazywanie kodu z parametryzacją zachowania* 29

2.1	Radzenie sobie ze zmieniającymi się wymaganiami	30
	<i>Pierwsza próba: filtrowanie zielonych jabłek</i>	<i>31</i>
	<i>Druga próba: parametryzacja koloru</i>	<i>31</i>
	<i>Trzecia próba: filtrowanie z użyciem dowolnego atrybutu, jaki przyjdzie nam na myśl</i>	<i>32</i>
2.2	Parametryzacja zachowania	33
	<i>Czwarta próba: filtrowanie na podstawie kryteriów abstrakcyjnych</i>	<i>35</i>

- 2.3 Zwalczanie rozwlekłości 39
 - Klasy anonimowe* 40
 - Piąta próba: korzystanie z anonimowych klas* 40
 - Szósta próba: korzystanie z wyrażeń lambda* 42
 - Siódma próba: wyabstrahowany typ List* 43
- 2.4 Przykłady praktyczne 44
 - Sortowanie z użyciem komparatora* 44
 - Wykonywanie bloku kodu z użyciem interfejsu Runnable* 45
 - Zwracanie rezultatu z użyciem interfejsu Callable* 45
 - Obsługa zdarzeń graficznego interfejsu użytkownika* 46

3. Wyrażenia lambda 49

- 3.1 Wyrażenia lambda w pigułce 50
- 3.2 Gdzie i jak używać wyrażeń lambda 53
 - Interfejs funkcyjny* 54
 - Deskryptor funkcji* 56
- 3.3 Wyrażenia lambda w praktyce: wzorzec execute-around 59
 - Krok 1: Pamiętaj o parametryzacji zachowania* 59
 - Krok 2: Do przekazania zachowań użyj interfejsu funkcyjnego* 60
 - Krok 3: Wykonaj zachowanie!* 60
 - Krok 4: Przekazanie wyrażeń lambda* 61
- 3.4 Korzystanie z interfejsów funkcyjnych 62
 - Interfejs Predicate* 62
 - Interfejs Consumer* 63
 - Interfejs Function* 63
- 3.5 Sprawdzanie typu, wnioskowanie o typach i ograniczenia 68
 - Sprawdzanie typu* 68
 - To samo wyrażenie, różne interfejsy funkcyjne* 70
 - Wnioskowanie o typach* 72
 - Korzystanie ze zmiennych lokalnych* 72
- 3.6 Referencje do metod 74
 - W dużym skrócie* 74
 - Referencje do konstruktorów* 78
- 3.7 Wyrażenia lambda i referencje do metod w praktyce! 80
 - Krok 1: Przekaż kod* 80
 - Krok 2: Użyj anonimowej klasy* 81
 - Krok 3: Użyj wyrażenia lambda* 81
 - Krok 4: Użyj referencji do metody* 82

- 3.8 Przydatne metody do tworzenia wyrażeń lambda 82
 - Komponowanie komparatorów* 83
 - Komponowanie predykatów* 83
 - Komponowanie funkcji* 84
- 3.9 Podobne koncepcje w matematyce 86
 - Całkowanie* 86
 - Łączenie z wyrażeniami lambda w Javie* 8 88

CZĘŚĆ 2. FUNKCYJNE PRZETWARZANIE DANYCH Z UŻYCIEM STRUMIENI.....91

4. Wprowadzenie do strumieni 93

- 4.1 Czym są strumienie? 94
- 4.2 Rozpoczynanie pracy ze strumieniami 98
- 4.3 Strumienie a kolekcje 101
 - Jednokrotne przechodzenie po elementach* 103
 - Iteracja zewnętrzna i iteracja wewnętrzna* 104
- 4.4 Operacje na strumieniach 107
 - Operacje pośrednie* 108
 - Operacje końcowe* 109
 - Praca ze strumieniami* 109
- 4.5 Mapa drogowa 110

5. Praca ze strumieniami 113

- 5.1 Filtrowanie 114
 - Filtrowanie z użyciem predykatu* 114
- 5.1.2 Filtrowanie unikatowych elementów 115
- 5.2 Podział strumienia 116
 - Podział z użyciem predykatu* 116
 - Przycinanie strumienia* 117
 - Pomijanie elementów* 118
- 5.3 Mapowanie 120
 - Aplikowanie funkcji do każdego elementu strumienia* 120
 - Splaszczanie strumieni* 121
- 5.4 Wyszukiwanie i dopasowywanie 125
 - Sprawdzanie, czy predykat pasuje do co najmniej jednego elementu* 125
 - Sprawdzanie, czy predykat pasuje do wszystkich elementów* 125
 - Wyszukiwanie elementu* 126
 - Znajdywanie pierwszego elementu* 127

- 5.5 Redukowanie 128
 - Sumowanie elementów* 128
 - Maksimum i minimum* 130
- 5.6 Strumienie w praktyce 135
 - Dziedzina: handlarze i transakcje* 135
 - Rozwiązania* 136
- 5.7 Strumienie numeryczne 139
 - Specjalizacje strumieni prymitywnych* 140
 - Zakresy numeryczne* 142
 - Strumienie numeryczne w praktyce: trójki pitagorejskie* 142
- 5.8 Tworzenie strumieni 145
 - Strumienie tworzone z wartości* 146
 - Strumień z obiektu nullable* 146
 - Strumienie tworzone z tablic* 146
 - Strumienie tworzone z plików* 147
 - Tworzenie strumieni z funkcji: tworzenie strumieni nieskończonych!* 147
- 5.9 Przegląd 152

6. Zbieranie danych z użyciem strumieni 155

- 6.1 Kolektory w skrócie 157
 - Kolektory jako zaawansowane redukcje* 157
 - Predefiniowane kolektory* 158
- 6.2 Redukowanie i podsumowywanie 159
 - Wyszukiwanie maksymalnej i minimalnej wartości w strumieniu* 160
 - Podsumowywanie* 160
 - Łączenie łańcuchów tekstowych* 162
 - Uogólnione podsumowywanie za pomocą redukcji* 163
- 6.3 Grupowanie 168
 - Manipulowanie pogrupowanymi elementami* 169
 - Grupowanie wielopoziomowe* 171
 - Kolekcjonowanie danych w podgrupach* 172
- 6.4 Partycjonowanie 176
 - Partycjonowanie zaawansowane* 177
 - Partycjonowanie liczb na liczby pierwsze i nie będące pierwszymi* 178
- 6.5 Interfejs Collector 182
 - Sens istnienia metod deklarowanych przez interfejs Collector* 183
 - Łączymy wszystko w jedną całość* 187
- 6.6 Rozwijanie własnego kolektora w celu uzyskania lepszej wydajności 189
 - Dzielenie wyłącznie przez liczby pierwsze* 189
 - Porównywanie wydajności kolektorów* 194

7. Równoległe przetwarzanie danych i wydajność 197

- 7.1 Strumienie równoległe 198
 - Zamiana strumienia sekwencyjnego na równoległy* 199
 - Dokonywanie pomiaru wydajności strumienia* 201
 - Poprawne korzystanie ze strumieni równoległych* 207
 - Efektywne korzystanie ze strumieni równoległych* 208
- 7.2 Model rozwidlania i złączania 210
 - Praca z RecursiveTask* 210
 - Najlepsze praktyki korzystania z modelu fork/join* 215
 - Kradzież pracy* 216
- 7.3 Spliterator 218
 - Proces podziału* 218
 - Implementowanie własnego splitera* 220

CZĘŚĆ 3. EFEKTYWNE PROGRAMOWANIE Z UŻYCIEM STRUMIENI I WYRAŻEŃ LAMBDA 229

8. Rozszerzenia API kolekcji 231

- 8.1 Fabryki kolekcji 232
 - Fabryka list* 233
 - Fabryka zbiorów* 234
 - Fabryki map* 235
- 8.2. Praca z listami i zbiorami 236
 - removeIf* 236
 - replaceAll* 237
- 8.3 Praca z kolekcją Map 238
 - forEach* 238
 - Sortowanie* 239
 - getOrDefault* 240
 - Wzorce Compute* 240
 - Wzorce Remove* 242
 - Wzorce Replace* 242
 - Scalanie* 243
- 8.4 Usprawniona klasa ConcurrentHashMap 245
 - Metody reduce i search* 245
 - Zliczanie* 246
 - Widoki zbiorów* 246

9. Refaktoryzacja, testowanie i debugowanie 247

- 9.1 Refaktoryzowanie w celu zwiększenia czytelności i elastyczności 248
 - Podnoszenie czytelności kodu* 248
 - Z klas anonimowych na wyrażenia lambda* 249
 - Z wyrażen lambda na referencje do metod* 250
 - Od imperatywnego przetwarzania kodu do strumieni* 252
 - Zwiększanie elastyczności kodu* 252
- 9.2 Refaktoryzacja zorientowanych obiektowo wzorców projektowych z użyciem wyrażen lambda 255
 - Strategia* 256
 - Metoda szablonowa* 257
 - Obserwator* 259
 - Łańcuch odpowiedzialności* 261
 - Fabryka* 263
- 9.3 Testowanie wyrażen lambda 265
 - Testowanie zachowania widocznego wyrażenia lambda* 265
 - Skupianie się na zachowaniu metody z wykorzystaniem wyrażenia lambda* 266
 - Wyciąganie złożonych wyrażen lambda do oddzielnych metod* 267
 - Testowanie funkcji wyższego rzędu* 267
- 9.4 Debugowanie 268
 - Sprawdzanie wywołań stosu* 268
 - Rejestrowanie informacji* 270

10. Tworzenie języków dziedzinowych z użyciem wyrażen lambda 273

- 10.1 Specyficzny język dla naszej dziedziny 276
 - Zalety i wady języków dziedzinowych* 276
 - Różne warianty języków dziedzinowych na wirtualnej maszynie Javy* 278
- 10.2 Małe języki dziedzinowe w nowoczesnych API Javy 283
 - Strumień jako język dziedzinowy do manipulowania kolekcjami* 284
 - Kolektory jako język dziedzinowy do agregowania danych* 285
- 10.3 Wzorce i techniki do tworzenia języków dziedzinowych w Javie 287
 - Łączenie metod* 290
 - Funkcje zagnieżdżone* 293
 - Sekwencjonowanie funkcji z użyciem wyrażen lambda* 295
 - Łączymy wszystko w jedną całość* 298
 - Korzystanie z referencji do metod w języku dziedzinowym* 300

- 10.4 Język dziedzinowy Javy 8 w praktyce 303
 - jOOQ* 304
 - Cucumber* 305
 - Spring Integration* 307

CZĘŚĆ 4. JAVA NA CO DZIEŃ.....311

11. Klasa *Optional* jako lepsza alternatywa dla *null* 313

- 11.1 W jaki sposób modelować brak wartości? 314
 - Redukowanie wyjątków `NullPointerException` za pomocą defensywnego sprawdzania* 315
 - Problemy z `null`* 317
 - Jakie są alternatywy dla `null` w innych językach?* 317
- 11.2 Wprowadzenie do klasy *Optional* 318
- 11.3 Wzorce używane do adopcji klasy *Optional* 320
 - Tworzenie obiektów opcjonalnych* 320
 - Wyodrębnianie i przekształcanie wartości z obiektów opcjonalnych za pomocą `map`* 321
 - Łączenie obiektów opcjonalnych za pomocą metody `flatMap`* 322
 - Manipulowanie strumieniem obiektów opcjonalnych* 326
 - Domyślne akcje i odpakowywanie obiektów opcjonalnych* 328
 - Łączenie dwóch obiektów opcjonalnych* 329
 - Odrzucanie pewnych wartości za pomocą metody `filter`* 330
- 11.4 Praktyczne przykłady wykorzystania klasy *Optional* 332
 - Opakowywanie potencjalnych wartości `null` obiektem opcjonalnym* 333
 - Wyjątki kontra obiekty opcjonalne* 333
 - Prymitywne obiekty opcjonalne i dlaczego nie warto ich używać* 334
 - Łączymy wszystko w jedną całość* 334

12. Nowe API daty i godziny 339

- 12.1 *LocalDate*, *LocalTime*, *Instant*, *Duration* i *Period* 340
 - Praca z `LocalDate` i `LocalTime`* 340
 - Łączenie daty i godziny* 342
 - Instant: data i godzina dla maszyn* 343
 - Definiowanie instancji `Duration` i `Period`* 344
- 12.2 Manipulowanie, parsowanie i formatowanie dat 345
 - Praca z obiektami `TemporalAdjuster`* 348
 - Wypisywanie i parsowanie obiektów daty i godziny* 351

- 12.3 Praca z różnymi strefami czasowymi i kalendarzami 353
 - Korzystanie ze stref czasowych* 353
 - Stałe przesunięcie z UTC/Greenwich* 354
 - Korzystanie z alternatywnych systemów kalendarzowych* 355

13. *Metody domyślne* 357

- 13.1 Rozwijanie istniejącego API 360
 - API w wersji 1* 361
 - API w wersji 2* 362
- 13.2 Metody domyślne w skrócie 364
- 13.3 Wzorce użycia dla metod domyślnych 366
 - Metody opcjonalne* 366
 - Wielokrotne dziedziczenie zachowania* 367
- 13.4 Reguły rozwiązywania 371
 - Trzy reguły rozwiązywania, o których trzeba wiedzieć* 372
 - Wygrywa najbardziej szczegółowy interfejs dostarczający metodę domyślną* 372
 - Konflikty i jawne rozwiewanie wątpliwości* 374
 - Problem diamentowy* 375

14. *System modułów Javy* 379

- 14.1 Siła napędowa: myślenie o oprogramowaniu 380
 - Podział odpowiedzialności* 380
 - Ukrywanie informacji* 381
 - Oprogramowanie w Javie* 381
- 14.2 Dlaczego powstał system modułów Javy? 382
 - Ograniczenia modularności* 382
 - Monolityczne JDK* 384
 - Porównanie z OSGi* 384
- 14.3 Moduły Javy – szersza perspektywa 386
- 14.4 Rozwijanie aplikacji z systemem modułów Javy 387
 - Przygotowywanie aplikacji* 387
 - Podstawy systemu modułów Javy* 389
 - Podstawy systemu modułów Javy* 389
- 14.5 Praca z kilkoma modułami 391
 - Klauzula exports* 391
 - Klauzula requires* 392
 - Nazewnictwo* 393
- 14.6 Kompilowanie i pakowanie 393
- 14.7 Moduły automatyczne 397

- 14.8 Deklaracja modułu i klauzule 398
 - requires* 398
 - exports* 398
 - requires transitive* 399
 - exports to* 399
 - open i opens* 400
 - uses i provides* 400
- 14.9 Większy przykład i gdzie dowiedzieć się więcej 400

CZĘŚĆ 5. ROZSZERZONA WSPÓLBIEŻNOŚĆ JAVY..... 403

15. *Konceptje stojące za interfejsem `CompletableFuture` i programowaniem reaktywnym* 405

- 15.1 Rozwój wsparcia Javy dla wyrażania współbieżności 408
 - Wątki i abstrakcje wyższego poziomu* 409
 - Wykonawcy i pule wątków* 411
 - Inne abstrakcje wątków – niezagnieżdżone z wywołaniami metod* 413
 - Czego chcemy od wątków?* 415
- 15.2 Synchroniczne i asynchroniczne API 415
 - API z wykorzystaniem obiektów `Future`:* 418
 - API w stylu reaktywnym:* 418
 - Uśpienie (i inne operacje blokowania) uznawane za szkodliwe* 420
 - W jaki sposób wyjątki działają z asynchronicznymi API?* 422
- 15.3 Model skrzynek i kanałów 423
- 15.4 `CompletableFuture` i kombinatory dla współbieżności 425
- 15.5 Protokół publikacji i subskrypcji oraz programowanie reaktywne 428
 - Przykład sumowania dwóch zdarzeń `Flow`* 429
 - Wąskie gardło* 434
 - Prosta forma prawdziwego wąskiego gardła* 435
- 15.6 Reaktywne systemy kontra reaktywne programowanie 436
- 15.7 Mapa drogowa 436

16. *`CompletableFuture`: kompozycyjne programowanie asynchroniczne* 439

- 16.1 Proste użycie obiektów `Future` 440
 - Obiekty `Future` i ich ograniczenia* 441
 - Korzystanie z obiektów `CompletableFuture` w celu budowy aplikacji asynchronicznej* 442

- 16.2 Implementowanie asynchronicznego API 443
 - Konwertowanie metody synchronicznej na metodę asynchroniczną* 444
 - Obsługa błędów* 446
- 16.3 Pisanie kodu, który się nie blokuje 449
 - Zrównoleglanie żądań z użyciem strumieni równoległych* 450
 - Tworzenie asynchronicznych żądań przy użyciu obiektów `CompletableFuture`* 450
 - Poszukiwanie rozwiązania, które skaluje się lepiej* 453
 - Korzystanie z niestandardowego wykonawcy* 454
- 16.4 Tworzenie przepływu asynchronicznych zadań 456
 - Implementowanie usługi zniżek* 457
 - Korzystanie z usługi `Discount`* 458
 - Komponowanie synchronicznych i asynchronicznych operacji* 459
 - Łączenie dwóch obiektów `CompletableFuture` – zależnego i niezależnego* 462
 - Refleksje na temat obiektów `Future` i `CompletableFuture`* 464
 - Efektywne korzystanie z wartości przekroczenia dozwolonego czasu* 465
- 16.5 Reagowanie na ukończenie obiektu `CompletableFuture` 466
 - Refaktoryzacja aplikacji do wyszukiwania najlepszej ceny* 467
 - Łączymy wszystko w jeden przykład praktyczny* 469
- 16.6 Mapa drogowa 470

17. Programowanie reaktywne 471

- 17.1 Manifest reaktywny 472
 - Reaktywność na poziomie aplikacji* 473
 - Reaktywność na poziomie systemu* 475
- 17.2 Reaktywne strumienie oraz API Flow 476
 - Wprowadzenie do klasy `Flow`* 477
 - Nasza pierwsza reaktywna aplikacja* 480
 - Przekształcanie danych z użyciem interfejsu `Processor`* 485
 - Dlaczego Java nie dostarcza implementacji API Flow?* 487
- 17.3 Korzystanie ze biblioteki reaktywnej RxJava 488
 - Tworzenie i korzystanie z obiektów `Observable`* 489
 - Przekształcanie i łączenie obiektów `Observable`* 494

CZĘŚĆ 6. PROGRAMOWANIE FUNKCYJNE I DALSZY ROZWÓJ JĘZYKA JAVA 499

18. *Myślenie funkcyjne* 501

- 18.1 Implementowanie i utrzymywanie systemów 502
 - Współdzielone dane modyfikowalne* 502
 - Programowanie deklaratywne* 504
 - Dlaczego programowanie funkcyjne?* 505
- 18.2 Czym jest programowanie funkcyjne? 505
 - Java w stylu funkcyjnym* 506
 - Transparentność referencyjna* 508
 - Programowanie obiektowo zorientowane a programowanie funkcyjne* 509
 - Styl funkcyjny w praktyce* 510
- 18.3 Rekurencja kontra iteracja 512

19. *Techniki programowania funkcyjnego* 517

- 19.2 Wszechobecne funkcje 518
 - Funkcje wyższego rzędu* 518
 - Rozwijanie funkcji* 520
- 19.2 Trwałe struktury danych 522
 - Destrukcyjne aktualizacje kontra programowanie funkcyjne* 522
 - Kolejny przykład z drzewami* 524
 - Korzystanie z podejścia funkcyjnego* 526
- 19.3 Leniwa ewaluacja z użyciem strumieni 528
 - Samodefiniujący się strumień* 528
 - Nasze własne leniwe listy* 531
- 19.4 Dopasowanie do wzorców 536
 - Wzorzec projektowy odwiedzającego* 536
 - Dopasowywanie do wzorców na ratunek* 537
- 19.5 Różności 541
 - Buforowanie lub memoizacja* 541
 - Co oznacza „zwraca ten sam obiekt”?* 542
 - Kombinatory* 543

20. *Mieszanie programowania funkcyjnego i zorientowanego obiektowo: Java i Scala* 545

- 20.1 Wprowadzenie do języka Scala 546
 - Przykład „Hello beer”* 546
 - Podstawowe struktury danych: List, Set, Map, Tuple, Stream i Option* 548

- 20.2 Funkcje 554
 - Funkcje pierwszej kategorii w Scali* 554
 - Funkcje anonimowe i domknięcia* 555
 - Rozwijanie funkcji* 557
- 20.3 Klasy i cechy 559
 - Mniej rozwlekłości z klasami w Scali* 559
 - Cechy w Scali kontra interfejsy w Javie* 560

21. Konkluzje oraz co dalej z Javą 563

- 21.1 Podsumowanie funkcji Javy 8 564
 - Parametryzacja zachowania (wyrażenia lambda i referencje do metod)* 564
 - Strumienie* 565
 - CompletableFuture* 566
 - Optional* 566
 - API Flow* 567
 - Metody domyślne* 567
- 21.2 Podsumowanie systemu modułów Javy 9 567
- 21.3 Wnioskowanie o typach lokalnych zmiennych w Javie 10 569
- 21.4 Co przyniesie Javie przyszłość? 570
 - Wariancja po stronie deklaracji* 570
 - Dopasowywanie do wzorców* 571
 - Bogatsza forma generyków* 572
 - Większe wsparcie dla niezmienności* 574
 - Typy wartości* 575
- 21.5 Przyspieszanie rozwoju języka Java 578
- 21.6 Słowo końcowe 580

Dodatek A. Różne aktualizacje języka 581

Dodatek B. Różne aktualizacje bibliotek 587

Dodatek C. Równoległe wykonywanie wielu operacji na strumieniu 597

Dodatek D. Wyrażenia lambda i kod pośredni maszyny wirtualnej Javy 607

Indeks 613