

Program szkolenia:

Java - zagadnienia zaawansowane, techniki OO, funkcyjne oraz wzorce

Informacje:

Nazwa:	Java - zagadnienia zaawansowane, techniki OO, funkcyjne oraz wzorce
Kod:	advanced-Java Pro
Kategoria:	Zaawansowana Java
Odbiorcy:	developerzy
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Szkolenie zostało opracowane z myślą o uczestnikach pragnących poszerzyć swoje kompetencje programistyczne w pragmatycznym kierunku.

Materiał został dobrany na podstawie wieloletnich doświadczeń programistów biorących udział w wielu projektach - nie jest to rodzaj bezproduktywnych łamigłówek znanych z niektórych testów certyfikacyjnych.

Zakres szkolenia został rozszerzony o zaawansowane zagadnienia Object Oriented, testowania i aspekty architektury aplikacji.

Zalety szkolenia:

- Realne problemy i pragmatyczne rozwiązania
- Software Craftsmanship
- Zaawansowane wzorce i techniki obiektowe

Szczegółowy program:

1. Nowości w świecie Javy

1.1. Wyrażenia Lambda, Optional/Stream/CompletableFuture, java.time, lepsza inferencja typów, poprawki w HashMap

1.2. G1GC, JPMS, Unified JVM logging, VarHandles, Stackwalker, Collections Factory Methods, Spin-wait hints

1.3. Local-variable type-inference, Parallel Full GC dla G1, Time-Based Release Versioning

1.4. Epsilon, Http Client, LocalVariable syntax for Lambdas, Flight Recorder, Low-overhead heap profiling, ZGC

1.5. Project Amber, Valhalla, Loom

2. Uczenie się myślenia funkcyjnego - krótkie ćwiczenia praktyczne w formule TDD

2.1. lambdy

2.2. monady

2.3. streamy

2.4. Własne Collectors oraz Spliterators (na przykładzie Collectora do ImmutableSet i Sliding Window Spliteratora)

3. Wybór praktycznych wzorców projektowych GoF oraz ich odświeżone wersje w wykorzystaniem receptur funkcyjnych

3.1. Command

3.2. Strategy

3.3. Template Method

3.4. Observer

3.5. Decorator

3.6. Chain of Responsibility

4. SOLID i design na przykładzie problemów z domen biznesowych

4.1. Single Responsibility Principle

4.2. Open-closed Principle

4.3. Liskov Substitution

4.4. Interface Segregation

4.5. Dependency Inversion

4.6. Pułapki Dziedziczenia

4.7. Wstęp do Hexagonal Architecture i niebetonowania domeny frameworkami

5. Praktyczny wstęp do concurrency

5.1. Wątki, definicja, tworzenie

5.2. Przegląd java.util.concurrent

5.3. ExecutorService

5.3.1. Przegląd

5.3.2. Dobre praktyki

5.4. Dlaczego Parallel Streams nie nadają się na produkcję

5.5. Dlaczego lepiej nie korzystać z Executors

5.6. CompletableFuture

5.7. Zadanie: własny scheduler uruchamiający zadania równolegle

5.8. Zadanie: własny prymitywny GC

6. Receptury "effective java"

6.1. Sealed classes w Javie

6.2. Bezpieczny wątkowo i wydajny lazy init

6.3. Jak sobie poradzić z wyjątkami w lambda

7. Użyteczne biblioteki

7.1. Lombok

7.2. Guava/Commons (tylko te części które wciąż mają zastosowanie)

7.3. Vavr

8. Wstęp do programowania reaktywnego z wykorzystaniem Project Reactor