

Program szkolenia:

Efektywne programowanie w Javie oraz zaawansowane techniki OO i wzorce

Informacje:

Nazwa:	Efektywne programowanie w Javie oraz zaawansowane techniki OO i wzorce
Kod:	craft-patterns-Java Pro
Kategoria:	Wzorce projektowe
Grupa docelowa:	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. Elementy języka Java

1.1. Adnotacje

1.1.1. Tworzenie i wykrywanie istnienia własnych adnotacji

1.1.2. Techniki budowania frameworków opartych na własnych adnotacjach

1.2. Typy wyliczeniowe

1.2.1. Typy wyliczeniowe jako pełnoprawne obiekty

1.2.2. Redukcja złożoności kodu

1.2.3. Zamiast instrukcji switch

1.2.4. Rozbudowa do wzorca Visitor zamiast instrukcji switch

2. Typowe potrzaski podczas programowania w Javie

2.1. Wycieki pamięci - sposoby unikania

2.1.1. Mapy

2.1.2. Kolekcje

2.1.3. Singletony

2.2. Dokładność obliczeniowa - dobór typów

2.2.1. Hermetyzacja w Value Objects

2.3. Klonowanie

2.4. Poprawna implementacja equals i hashCode (również w kontekście JPA)

2.4.1. Co to znaczy, że obiekty tożsame

3. Efektywne wykorzystanie klas Biblioteki Standardowej

4. Współbieżność

4.1. Planowe wykonywanie wątków przez Executor

4.2. Kolekcje bezpieczne ze względu na wątki

4.3. Pułapki współbieżności

5. Obsługa wyjątków

5.1. Style i konwencje

5.1.1. Sytuacja niepoprawna to nie wyjątek

5.1.2. Poprawne wykorzystanie wyjątków weryfikowalnych i nieweryfikowalnych

5.2. Typowe błędy podczas obsługi wyjątków

5.2.1. Obsługa i propagacja

5.2.2. Dławienie wyjątków

5.2.3. Problemy z wydajnością

5.2.4. Czy zbierać stacktrace

5.3. Przydatne wyjątki standardowe

6. Efektywne wykorzystanie Object Oriented

6.1. Pułapki dziedziczenia

6.1.1. Zamknięcie kodu na rozbudowę

6.1.2. Zastępowania dziedziczenia kompozycją – praktyczne zalety zmiany podejścia

6.1.2.1. Dziedziczenie nie nadaje się do modelowania ról

6.1.2.2. Liskov Substitution Principle

6.1.2.3. Wzorzec Party

6.1.2.4. Wzorzec Role Object i Extension Object

6.2. Najlepsze praktyki

6.2.1. Klasy abstrakcyjne czy interfejsy

6.3. Code smell

7. Zestaw kilkunastu wzorców implementacyjnych

8. Przegląd efektywnych bibliotek Apache i Google

9. Techniki programistyczne

9.1. Praktyczne elementy GRASP

9.2. Praktyczne wykorzystanie SOLID

9.2.1. Kohezja klas

9.2.1.1. Jak ją osiągnąć

9.2.1.2. Jak wykrywać zapachy kodu

9.2.2. Otwartość na rozbudowę

9.2.2.1. Wzorzec Strategii

9.2.2.2. 3 rodzaje logiki: stabilna, domknięcia, wybór domknięć

9.2.3. Kiedy dziedziczenie nie ma sensu

9.2.4. Zakres interfejsów

9.2.5. Odpowiedni poziom abstrakcji

9.3. Dążenie do kodu o czytelności zbliżonej do prozy

9.3.1. Technika podmiot.orzecznie(dopełnienie, przydawka)

9.3.2. Technika modelowania niezmienników

9.3.3. Technika określania granicy klasy na podstawie analizy Use Case

9.4. Projektowanie otwarte na testy (testability)

10. Testowanie automatyczne

10.1. Rodzaje testów

10.2. Narzędzia

10.3. Techniki testów jednostkowych (fake, stub, mock)

10.4. Biblioteka Mockito