

## **Kod szkolenia: J/ADV**

## **Tytuł szkolenia: Tworzenie wydajnego oprogramowania w języku Java - techniki zaawansowane.**

### **Adresaci szkolenia:**

Szkolenie jest adresowane do programistów języka Java, którzy chcą poznać sekrety tworzenia wydajnego oprogramowania w tym języku.

### **Cel szkolenia:**

Celem szkolenia jest obycie się z praktykami tworzenia wydajnego oprogramowania w Java, poprzez tworzenie benchmark'ów i badanie wydajności, profilowanie aplikacji, używanie struktur danych odpowiednich do trybu pracy, wydajny dostęp do plików i przede wszystkim obalenie błędnych mitów dotyczących pracy Garbage Collector'a. Wielu programistów Java posiadających nawet wieloletnie doświadczenie nie zdaje sobie sprawy z tego jakie grzechy popełnia próbując pomagać wirtualnej maszynie Java, czy Garbage Collector'owi, w rezultacie otrzymując efekt przeciwny do zamierzonego. To szkolenie obala te mity i uczy prawdziwych praktyk tworzenia wydajnego oprogramowania w Javie. A także jak zwiększyć wydajność aplikacji bez dotykania samego kodu, czyli profilowanie JVM i GC.

Uczestnik szkolenia uczy się również używania zaawansowanych elementów języka takich jak refleksja, introspekcja czy dynamiczne proxy.

### **Wymagania:**

Od uczestników szkolenia wymagana jest umiejętność programowania w języku Java (do poznania na kursie J/JP).

### **Parametry szkolenia:**

5\*7 godzin wykładów i warsztatów w proporcji 1/2. W trakcie warsztatów, odnajdywane są wąskie gardła aplikacji, badane aspekty wydajnościowe różnych struktur danych i sposobów dostępu do dysku, a także ćwiczzone zaawansowane aspekty języka Java.

Wielkość grupy: maks. 8-10 osób.

## Polecane szkolenia poprzedzające:

J/JP

## Program szkolenia:

1. Tworzenie wydajnego oprogramowania
  - I. Różne wymiary wydajności
    - i. Wydajność obliczeniowa
    - ii. Wydajność pamięciowa (RAM footprint)
    - iii. Wydajność uruchomienia
    - iv. Wydajność skalowania
  - II. Wydajność a użytkownik - subiektywne odczucie wydajności
  - III. Proces tworzenia wydajnego oprogramowania
    - i. Analiza wydajności
    - ii. Projektowanie a wydajność
    - iii. Architektura a wydajność
    - iv. Kodowanie a wydajność
    - v. Testowanie a wydajność
    - vi. Profilowanie a wydajność
  - IV. Co wpływa na wydajność w Javie
2. Pomiary wydajności
  - I. Problemy ze zwiększaniem wydajności
  - II. Co to jest Benchmarking
    - i. Sposoby dokonywania pomiarów
    - ii. Zalety budowania benchmarków
    - iii. Micro-benchmark
    - iv. Macro-benchmark
    - v. Problemy wyników pomiarowania
  - III. Czym jest profilowanie aplikacji
    - i. Wpływ profilowania na wydajność
    - ii. Problemy z płaskim profilowaniem
    - iii. Zalety narzędzi do profilowania
    - iv. Narzędzia do profilowania (Java 6)
      - A. Problemy do zdiagnozowania
      - B. Opis narzędzi
      - C. Profilowanie w Eclipse
3. Wydajna praca z plikami
  - I. Strumienie
    - i. Podstawy
    - ii. Analiza alternatywnych rozwiązań
    - iii. Porównania wydajności rozwiązań na strumieniach
  - II. Swobodny dostęp do plików
  - III. Lepsza wydajność - bibliotek NIO
    - i. Kanały
      - A. FileChannel
      - B. ByteBuffer
      - C. Przykłady użycia kanałów
      - D. Porównanie wydajność rozwiązań
    - ii. Odzworowywanie plików w pamięci
    - iii. Porównanie wydajności różnych metod dostępu do plików
  - IV. NIO2 - Java7
    - i. Path (operacje na ścieżkach)

- ii. Files (operacje na plikach i katalogach)
  - A. Kopiowanie/Przenoszenie
  - B. System plików
  - C. Nowe funkcje odczytu/zapisu
  - D. StandardOpenOptions
  - E. Nowa konstrukcja try i Closeable (try-with-resources)
- V. Serializacja
  - i. Podstawy serializacji
  - ii. Problemy z serializacją
  - iii. Analiza wyników serializacji
  - iv. Sposoby optymalizacji serializacji
- 4. Algorytm wydajności
  - I. Sztuka doboru algorytmu
    - i. Porównanie algorytmów
    - ii. Elegancja a brute-force
    - iii. Dziedzina problemu a algorytm
  - II. Problemy z rekurencją
  - III. Nie tylko algorytm się liczy
- 5. Kolekcje i tablice
  - I. API kolekcji
    - i. Podstawowe interfejsy
      - A. Iterator
      - B. Iterable
      - C. Collection
        - 1. Set
        - 2. List
        - 3. Queue
        - 4. Deque
      - D. Map
    - ii. Najważniejsze funkcje
  - II. Struktury danych - podstawy wydajnych operacji
    - i. Tablica haszująca
      - A. Kontrakt między equals a hashCode
      - B. Kontrakt equals
      - C. Problemy z danymi w tablicy haszującej
    - ii. Przeszukiwanie binarne
      - A. Drzewa czerwono-czarne
        - 1. Opis zagadnienia
        - 2. Zasady działania
        - 3. Przykłady działania
  - III. Zbiory
    - i. Wydajność standardowych implementacji
      - A. HashSet
      - B. TreeSet
      - C. LinkedHashSet
      - D. EnumSet
    - ii. Nowe interfejsy zbiorów (Java 6)
      - A. NavigableSet
    - iii. Optymalizacja pracy na zbiorach
    - iv. Porównanie wydajności różnych implementacji przy różnych zadaniach
  - IV. Listy
    - i. Wydajność standardowych implementacji
      - A. ArrayList
      - B. LinkedList

- ii. Optymalizacja pracy z listami
- iii. Porównanie wydajności list
- iv. Opis struktur ArrayList i LinkedList - wizualizacja złożoności operacji
- v. Interfejs RandomAccess a wydajność przeglądania listy
- V. Kolejki
  - i. Wydajność standardowych implementacji
    - A. PriorityQueue
    - B. LinkedList
  - ii. Nowe API - kolejki dwustronne (Java 6)
    - A. Interfejs Deque
    - B. Wydajność implementacji kolejki dwustronnej
      - 1. LinkedList
      - 2. ArrayDeque
- VI. Mapy
  - i. Wydajność standardowych implementacji
    - A. HashMap
    - B. TreeMap
    - C. EnumMap
    - D. IdentityHashMap
    - E. WeakHashMap
    - F. LinkedHashMap
  - ii. Nowe API map (Java6)
    - A. NavigableMap
  - iii. Optymalizacja pracy na mapach
  - iv. Porównanie wydajności map
  - v. Jak stworzyć cache
- VII. Stare kontenery (Java 1.0 i 1.1)
  - i. Implementacje starych kontenerów
    - A. Hashtable
    - B. Vector
    - C. Stack
    - D. BitSet
  - ii. Enumerator - Enumeration
  - iii. Porównanie wydajności z nowymi implementacjami
- VIII. Widok kolekcji
  - i. Zalety używania
  - ii. Niebezpieczeństwa
  - iii. Optymalizacja
- IX. Collections- klasa pomocnicza
  - i. Podstawowe funkcje
  - ii. Rola w optymalizacji
  - iii. Wykorzystanie w algorytmach
- X. Wydajność a kolekcje odporne na wielowątkowość
  - i. Wydajność kolejek wielowątkowych
    - A. ConcurrentLinkedQueue
  - ii. Wydajność kolejek blokujących
    - A. BlockingQueue
    - B. ArrayBlockingQueue
    - C. DelayQueue
    - D. LinkedBlockingQueue
    - E. PriorityBlockingQueue
    - F. SynchronousQueue
  - iii. Wydajność dwustronnych kolejek blokujących
    - A. BlockingDeque

- B. LinkedBlockingDeque
    - iv. Wydajność list wielowątkowych
      - A. CopyOnWriteArrayList
    - v. Wydajność map wielowątkowych
      - A. ConcurrentMap
      - B. ConcurrentNavigableMap
      - C. ConcurrentHashMap
      - D. ConcurrentSkipListMap
  - XI. Tablice
    - i. Zalety stosowania tablic
    - ii. Wady stosowania tablic
    - iii. Porównanie tablic z ArrayList
    - iv. Klasa pomocnicza Arrays
- 6. Caliper - microbenchmark Framework
  - I. Czym jest Caliper
  - II. SimpleBenchmark
  - III. Przykłady menchmarków
  - IV. Uruchomienie Caliper
  - V. Benchmarki parametryzowane (@Param)
  - VI. Parametry Caliper
  - VII. Prezentacja wyników przez WWW
  - VIII. Pułapki
- 7. Kontrola ładowania klas
  - I. Ładowanie klas- proces
  - II. Klasa Class
  - III. ClassLoader- kiedy tworzyć?
  - IV. Po co kontrolować ładowanie klas?
  - V. Zachłanne ładowanie klas
    - i. Jak działa zachłanne ładowanie klas
    - ii. Przykład zachłannego ładowania klas
    - iii. Kontrola zachłannego ładowania klas
  - VI. Redukowanie ilości klas
    - i. Klasy wewnętrzne
    - ii. Łączenie listenerów
    - iii. Refleksja
    - iv. Dynamiczne Proxy
  - VII. Kiedy warto kontrolować ładowanie klas?
- 8. Zarządzanie pamięcią
  - I. Java a zarządzanie pamięcią
    - i. Jak działa odśmiecanie pamięci (Garbage Collector)
    - ii. Co gwarantuje Garbage Collector?
    - iii. Cykl życia obiektu
      - A. Fazy cyklu życia obiektu
      - B. Wyspy obiektów
      - C. Metoda finalize
      - D. Problemy i pułapki
  - II. Typy referencji a Garbage Collector i proces odśmiecania
    - i. SoftReference
    - ii. WeakReference
    - iii. PhantomReference
  - III. Wycieki pamięci w Javie
    - i. Powody
    - ii. Złe praktyki
    - iii. Unikanie

- IV. Garbage Collector - złe praktyki
  - V. Ograniczanie zajętości pamięci
  - VI. Opcje strojenia Garbage Collector'a
    - i. Parametry konfiguracyjne Garbage Collector'a
      - A. Słaba teoria generacji - podstawa wydajnej pracy na obiektach
      - B. Typy odśmiecania
        - 1. Minor Collections
        - 2. Major Collections
      - C. Generacje obiektów
        - 1. Organizacja pamięci w Javie
        - 2. Młoda generacja
          - 1. Eden
          - 2. Survivor Space
        - 3. Stara generacja
        - 4. Pamięć permanentna
      - D. Parametry wydajnościowe Garbage Collector
      - E. Wymiarowanie pamięci (generacji)
    - ii. Optymalizacja algorytmu odśmiecania dla konkretnej aplikacji na konkretnej platformie sprzętowej
    - iii. Rodzaje GC
      - A. Rodzaje
        - 1. Serial Collector (Serial + Serial Old)
        - 2. Parallel Collector (Parallel Scavenge + Parallel Old)
        - 3. Concurrent Collector
        - 4. Parallel Copying Collector
        - 5. G1 – Garbage First
      - B. Zasady działania
      - C. Preferencje
      - D. Skalowalność
      - E. Strojenie
      - F. Problemy i rozwiązania
    - iv. Opcje podglądu pracy GC
    - v. Zalecenia przy wyborze GC
9. Maszyna wirtualna Javy
  - I. Podstawowe tryby pracy a wydajność
  - II. Parametry wydajnościowe JVM
  - III. Porównanie wydajności różnych wersji JVM (1.4, 5.x, 6.x, 7.x)
    - i. Na poziomie różnych benchmarków
      - A. JBB2000
      - B. JBB2005
      - C. VolanoMark 2.5
      - D. I/O Benchmark Comparison
      - E. CMS Server Benchmark
    - ii. Na poziomie różnych parametrów wydajnościowych
      - A. Działanie po stronie serwera
      - B. Startup
      - C. RAM footprint
      - D. Kopiowanie tablic
      - E. Wydajność GC
      - F. Operacje wejścia-wyjścia
10. Problemy z optymalizacją
  - I. Optymalizacje kompilatora a micro-benchmark
  - II. Micro-benchmark a GC
  - III. Uruchamianie wielu aplikacji

- IV. Przyzwyczajenia programistów
- V. Optymalizacja za kompilator
- VI. Przedwczesna optymalizacja
- VII. Antywzorce związane z wydajnością
  - i. Zachłanność
  - ii. Pośpiech
  - iii. Lenistwo
  - iv. Ignorancja