SZKOLENIE ZAAWANSOWANE

Zaawansowane programowanie w języku Java

J/DET

Czas trwania: 4 dni

Zaawansowane aspekty programowania w języku Java

Cele szkolenia

- Zapoznanie uczestników z zaawansowanymi aspektami programowania w języku Java

Zalety

- Szkolenie przedstawia skomplikowane aspekty języka Java w przystępny i użyteczny sposób
- Praktyka przed teorią - wszystkie szkolenia technologiczne prowadzone są w formie warsztatowej. Konieczna teoria jest wyjaśniana na przykładzie praktycznych zadań
- Konkretne umiejętności - w ramach każdego szkolenia rozwijamy praktyczne umiejętności związane z daną technologią i tematyką
- Nauka z praktykami - wszyscy trenerzy na co dzień pracują w projektach, gwarantuje to dostęp do eksperckiej wiedzy i praktycznego know-how

Dla kogo?

- Programiści chcący poznać język Java na zaawansowanym poziomie

Wymagania

- Umiejętność programowania w języku Java
- Przydatna znajomość podstaw Maven (użyte przy modułach)

Program

1. Enumeracje
    a. Więcej niż wyliczenia
    b. Definiowanie metod w enumeracjach
    c. Atrybuty w enumeracjach
    d. Konstruktory w enumeracjach
    e. API dla enumeracji
2. Tworzenie adnotacji
    a. Podstawy adnotacji
    b. Adnotacje na pakiecie
    c. Introspekcja a adnotacje
    d. Składnia tworzenia adnotacji
    e. Adnotacje wielokrotne
    f. Interfejs AnnotatedElement
    g. Wzmianka o javax.annotation.processing
3. Dynamiczna Java
    a. Dynamiczne tworzenie obiektów
    b. Refleksja
    c. Invokedynamic
    d. Dynamiczne proxy
    e. Wzmianka o module jdk.dynalink
    f. Wzmianka o agentach
4. Typy generyczne
    a. Użycie klas generycznych
    b. Tworzenie typów generycznych
    c. Tworzenie metod generycznych
    d. Typy generyczne a dziedziczenie
    e. Znaczniki (wildcards): ?, extends i super
    f. Znaczniki wielokrotne
    g. Porady użycia znaczników
    h. Konwencje nazewnictwa
    i. Kiedy potrzebny @SupressWarning("unchecked")?
    j. Zacieranie typów (type erasure)
    k. Porady tworzenia generyków
    l. Problem metod generycznych z varargs (@SafeVarargs)
    m. Typy generyczne a tablice
    n. Rekursywne typy generyczne
    o. Ograniczenia generyczności
    p. Ciekawe przypadki (wzorce)
5. Programowanie funkcyjne
    a. Interfejs funkcyjny
    b. Wyrażenia lambda (składnia, użycie, wskazówki)

c. forEach w Iterable i Map
d. Referencje do metod i konstruktorów
e. Wybrane interfejsy funkcyjne
f. Metody prywatne, statyczne i domyślne w interfejsach
g. Złączanie metod (functional composition)
h. Strumienie plików, tablic, kolekcji, adhoc
i. Praca na strumieniach: filtrowanie, reagowanie, przekształcanie
j. Kończenie strumieni z i bez agregacji
k. Strumienie typów prostych
l. StreamSupport
m. Czytelne strumienie
n. Kiedy strumienie, a kiedy nie
o. Dokładniej Optional i Collectors
p. Strumienie współbieżne
q. Gdzie jeszcze API funkcyjne

6. Moduły
a. public zbyt publiczny
b. Co dają moduły
c. Moduły nienazwane i automatyczne
d. Struktura i nazewnictwo modułów
e. Deskryptor modułu
f. Dostęp do zawartości modułu
g. Dostęp dla refleksji
h. Moduły wymagane
i. Udostępnianie i otwieranie pakietów
j. Dostarczanie i używanie usług
k. Dystrybucja usług
l. Moduły a ServiceLoader
m. Kompilacja modułu
n. Pakowanie modułu (modular JAR)
o. Uruchamianie modułu
p. jlink i moduł aplikacji
q. JAR modułu dla wielu wersji Java (multi-release)
r. Opcje Java dla modułów
s. Upgradeable modules
t. Opcje hakowania modułów
u. Wzmianka o narzędziach
v. Wzmianka o warstwach modułów
w. Migracja do modułów
x. jdeps - analiza zależności
y. Problemy migracji do modułów
z. Pliki atrybutów
aa. Testowanie modułów
ab. Wzmianka o jmod

7. Specjalne referencje (OPCJONALNE)
    a. Reference i ReferenceQueue
    b. SoftReference
    c. WeakReference i WeakHashMap
    d. PhantomReference i Cleaner
    e. Cykl życia obiektu i wycieki pamięci (opcjonalnie)