Java 15

Java viert dit jaar zijn zilveren (25ste) verjaardag. Op 23 mei 2020 was het 25 jaar geleden dat Sun Microsystems voor het eerst Java aan de wereld liet zien. Tot de dag van vandaag is het een van de populairste talen, ook al heeft het meer en meer concurrentie van andere talen, zoals Python, Rust en Go. Java heeft niet stilgestaan in al die tijd. De laatste jaren wordt er ook veel aandacht besteed aan het vernieuwen van de taal en het verhelpen van pijnpunten in de taal.

Object georiënteerd Java is voortgekomen uit het "Oak"-project dat in 1991 was begonnen en onder leiding stond van James Gosling. Het kreeg bekendheid vanwege de het idee van "Write Once, Run Anywhere". Dit kwam omdat de Java Virtual Machine (JVM) meerdere hardwareplatforms en besturingssystemen ondersteunde. Java-applets konden worden uitgevoerd vanaf een webpagina en presteerde jarenlang beter dan JavaScript. In Java SE 11 (2018) is dit onderdeel uit de taal gehaald. In 2010 nam Oracle het bedrijf Sun over. Java werd eind 2016 open source en in 2017 is de enterprise versie (Java EE) ook open source geworden onder de Eclipse Foundation vleugels.

Oracle heeft sinds Java SE 10 een 6 maanden releaseschema aangekondigd, in tegenstelling tot de 3 jaars (zo ongeveer) releases daarvoor. De deadline voor nieuwe features in Java is fixed gemaakt. Dit betekent dat het aantal **J**DK **E**nhancement **P**roposals (JEP) dat geïmplementeerd kan worden per release variabel is geworden. Eens in de drie jaar zal een zogenaamde Long-term support (LTS) versie worden gemaakt. Java SE 11 is zo'n LTS versie en zal door Oracle nog tot september 2023 ondersteund worden. Dit betekent dat Java 17 ook weer een LTS wordt. De versies die tussen LTS versies uitkomen krijgen alleen support tot een nieuwe versie uitkomt, dus zo ook Java 15. Java 15 komt in september 2020 uit. Dit artikel beschrijft welke nieuwe features in Java 15 geïmplementeerd zijn. Tijdens het schrijven van dit artikel was de laatste release datum nog niet bereikt en er kunnen dus nog (kleine) scope wijzigingen plaats vinden voordat dit artikel geplaatst wordt.

In Java 15 staan 14 nieuwe features (JEP's) gepland.

Om met wat Java 15 features te kunnen spelen, zonder de early access echt te hoeven installeren, is alles uitgeprobeerd binnen een Docker container met OpenJDK 15 (listing 1).

```
$ docker run -it --rm \
      -v $(pwd)/src/main/java:/src \
      openjdk:15 /bin/bash
$ cd /src
```

Listing 1

Dit artikel zal in drie hoofd-secties verdeeld worden. Het eerste deel zal gaan over nieuwe standaard features. Het tweede deel zal ingaan op preview en incubator features en het laatste deel over deprecated/verwijderde onderdelen. Bij elke feature zal het JEP-nummer vermeld worden.

Nieuwe standaard features

371: Hidden Classes
Hidden classes zijn classes die niet zichtbaar zijn voor de JVM, maar wel dynamisch geladen kunnen worden via hun bytecode. Dit is voornamelijk bedoeld voor frameworks die classes genereren (at runtime) en die indirect gebruiken via reflectie.

373: Reimplement the Legacy DatagramSocket API
Dit is een optimalisatie en modernisatie JEP om de oude implementatie naar een modernere implementatie te brengen voor makkelijker onderhoud.

Ivo Woltring is Software Architect en Codesmith bij Ordina JTech en houdt zich graag bezig met nieuwe ontwikkelingen in de software wereld. Ivo is toevallig ook op 23 mei jarig en ook net 25 geworden 😄

377: ZGC: A Scalable Low-Latency Garbage Collector
In Java 14 is deze Z Garbage Collector voor de grote platformen (Linux/ Windows/macOS) beschikbaar gemaakt, maar was nog experimenteel. In Java 15 zal deze status veranderen naar product feature.

378: Text Blocks
In de laatste drie versies van Java is geëxperimenteerd met de zogenaamde multi-line String. Het begin met JEP 326 in Java 12 en ging door in Java 13. In Java 14 werd nog aardig gesleuteld aan het geheel om het dicht te timmeren met JEP 368. In listing 2 zie je hetzelfde voorbeeld dat in het Java 14 artikel gegeven is, maar het het grote verschil dat het nu gecompileerd kan worden zonder preview activatie (zie listing 2).

Andere code voorbeelden van text blocks zijn te vinden in de Java 13 en 14 artikelen (**http://ivo2u.nl/oz**).

379: Shenandoah: A Low-Pause-Time Garbage Collector
Sinds Java 12 is deze GC een experimentele feature. In Java 15 wordt het een product feature. Shenandoah is een algoritme dat vooral geschikt is voor toepassingen waarvoor reactiesnelheid en voorspelbare korte pauzes belangrijk zijn. De korte pauze tijd wordt verkregen door meer garbage collection werk gelijktijdig te doen met het draaiende Java programma.
Om deze GC te activeren moet je deze opties meegeven: `–XX:+UseShenandoahGC`. De `–XX:+UnlockExperimentalVMOptions` is dus niet meer nodig.

```
public class JEP378 {
    public static void main(String[] args) {
        final String s = """
                Deze regel heeft geen enter hier \
                en een enkele spatie tussen de quotes: " "
                groen \s
                rood
                blauw \s
                """;
        System.out.println(s);
    }
}

$ javac JEP378.java
$ java JEP378
Deze regel heeft geen enter hier en een enkele spatie tussen de
quotes: " "
groen
rood
blauw
```

Listing 2

339: Edwards-Curve Digital Signature Algorithm (EdDSA)
De JEP behandelt de implementatie van de cryptografische handtekeningen (signature scheme) gebaseerd op het EdDSA-algoritme (RFC 8032). Een voordeel van het EdDSA-algoritme is de betere performance en beveiliging vergeleken met andere vergelijkbare algoritmes. Dit algoritme wordt bijvoorbeeld al ondersteund in libraries als OpenSSL.

Preview en Incubator Features

360: Sealed Classes (Preview)
Deze JEP biedt Java een manier om implementaties en overerving van classes, interfa-

ces en records te beperken tot wat expliciet toegestaan is. Het voorbeeld hieronder zal bestaat uit drie class files (Fruit, Orange en Apple) waarbij expliciet is aangegeven dat Orange van Fruit mag erven, maar Apple is daar niet in opgenomen, wat zal resulteren in een compile error (zie listing 3).

375: Pattern Matching for instanceof (Second Preview)
Dit is de tweede preview en de opvolger van JEP305 die in het Java 14 artikel (vorige editie) behandeld is. De type test kan gevolgd worden door een binding variabele die daarna meteen gebruikt kan worden. In het voorbeeld van listing 4 is dat de `bs` variabele. De `bs` variabele wordt in de if geïntroduceerd en in hetzelfde statement al gebruikt (`bs.length()`).

383: Foreign-Memory Access API (Second Incubator)
Deze API is al in Java 14 aangeboden als incubator module en is in dat artikel uitgebreid behandeld geweest. Deze JEP bevat enkele wijzigingen op de API. Vandaar ook de tweede incubator iteratie.

384: Records (Second Preview)
In Java 14 is de eerste preview van Records aangeboden. Records bieden een compacte syntaxis voor het declareren van klassen die houders zijn voor onveranderlijke (immutable) gegevens. Dit is de tweede review iteratie waarin feedback uit de eerste preview is verwerkt.

Deprecated en verwijderde Features

372: Remove the Nashorn JavaScript Engine
De Nashorn JavaScript script engine is volledig verwijderd, inclusief de jjs tool. Deze was sinds Java 11 deprecated verklaard. De hoofdreden was dat de ontwikkeling van ECMAScript zo snel gaat dat het moeilijk was om de ontwikkelingen bij te blijven.

374: Disable and Deprecate Biased Locking
Moderne computers zorgen ervoor dat Biased Locking geen toegevoegde waarde meer heeft. Biased Locking was een optimalisatie techniek gebruikt in de HotSpot Virtual Machine om de overhead te verlagen van zogenaamde "uncontended locking". Deze release zal het default uit zetten en het is deprecated verklaard. Het kan nog aangezet worden met `XX:+UseBiasedLocking`, maar zal in een toekomstige release verwijderd worden.

```
public sealed class Fruit permits Orange {}

public non-sealed class Orange extends Fruit {}

public class Apple extends Fruit {}

$ javac --enable-preview -source 15 Fruit.java Orange.java Apple.java
Apple.java:1: error: class is not allowed to extend sealed class: Fruit
public class Apple extends Fruit {
       ^
Note: Some input files use preview language features.
Note: Recompile with -Xlint:preview for details.
1 error
```

Listing 3

```
$ jshell --enable-preview
|  Welcome to JShell -- Version 15-ea
|  For an introduction type: /help intro

jshell> var s = "ivonet.nl"
s ==> "ivonet.nl"

jshell> if (s instanceof String bs && bs.length() > 5) System.out.
println("Mooie string!")
Mooie string!
```

Listing 4

381: Remove the Solaris and SPARC Ports
In Java 14 zijn de Solaris/SPARC, Solaris/x64, and Linux/SPARC ports deprecated verklaard en in Java 15 zijn ze verwijderd.

385: Deprecate RMI Activation for Removal
RMI Activation is een verouderd gedeelte van RMI (Remote Method Invocation). Het is al optioneel sinds Java 8 en wordt in Java 15 dus deprecated verklaard.

Conclusie

Het is lang verwacht, maar de Multi-line Strings zijn een feit! Naast deze heerlijke feature zijn er vooral veel doorontwikkelingen op preview features van Java 14. Ook wordt er grondig gesneden in verouderde onderdelen van de code base waardoor het geheel veel toekomst vaster wordt. ■

REFERENTIES

http://openjdk.java.net/projects/jdk/15/
https://tools.ietf.org/html/rfc8032
https://www.oracle.com/java/technologies/java-tuning.html#section4.2.5
http://ivo2u.nl/oz (voorbeeld code en vorige artikelen)
https://www.oracle.com/technetwork/java/java-se-support-roadmap.html