Java 14

Oracle heeft sinds Java SE 10 een 6 maanden releaseschema aangekondigd. De deadline voor nieuwe features in Java is fixed gemaakt. Dit betekent dat het aantal JDK Enhancement Proposals (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 zou betekenen 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 14. Dit artikel beschrijft welke nieuwe features in Java 14 geïmplementeerd zijn.

In Java 14 zijn 16 nieuwe features (JEP's) meegenomen. Om met wat Java 14 features te kunnen spelen, zonder de early access echt te hoeven installeren, is alles uitgeprobeerd binnen een Docker container met OpenJDK 14 (listing 1).

Dit artikel zal in drie hoofdsecties 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

358: Helpful NullPointerExceptions

Deze JEP verbetert de bruikbaarheid van `NullPointerExceptions` door meer informatie te geven. Gegeven de volgende code (listing 2). Krijg je normaal gesproken een error zoals in Listing 3.
Maar als je hem draait met optie `-XX:+ShowCodeDetailsInExceptionMessages` dan krijg je een specifieke foutmelding te zien (Listing 4).

361: Switch Expressions

De Switch had status preview in Java 12 en 13 en wordt nu standaard onderdeel van de taal. De uitbreiding houdt in dat de switch gebruikt kan worden zowel als een statement als een expressie. Code voorbeelden zijn te vinden in de vorige artikelen (**http://ivo2u.nl/oz**).

345: NUMA-Aware Memory Allocation for G1

Non-uniform memory access (NUMA) is een manier om een cluster van microprocessors te configureren in een systeem met meerdere processors, zodat geheugen lokaal kan worden gedeeld, de prestaties kunnen worden verbeterd en de mogelijkheden van het systeem kunnen worden uitgebreid. In Java 14 is NUMA-bewuste geheugentoewijzing geïmplementeerd om de G1-prestaties op grote machines te verbeteren. Als de `+XX:+UseNUMA` parameter wordt meegegeven als de JVM geïnitieerd wordt, zal een gelijke verdeling plaats gaan vinden over het totale aantal NUMA nodes.

349: JFR Event Streaming

De JDK Flight Recorder is verbeterd om continue monitoring van JDK Flight Recorder-gegevens te ondersteunen. Dus in plaats van het

Ivo Woltring is Software Architect en Codesmith bij Ordina JTech en houdt zich graag bezig met nieuwe ontwikkelingen in de softwarewereld.

```
$ docker run -it --rm \
      -v $(pwd)/src/main/java:/src \
      openjdk:14 /bin/bash
$ cd /src
```

Listing 1.

```
public class NPEApp {
    public static void main(String[] args) {
        final String[] s = new String[2];
        s[1].length();
    }
}
```

Listing 2.

opnemen (recording) en dan parsen van een events-bestand, kunnen events nu realtime gestreamd worden zonder een bestand te gebruiken, of ze kunnen vanuit een bestand worden gestreamd. Een code voorbeeld is hier (**http://ivo2u.nl/oz**) te vinden.

352: Non-Volatile Mapped Byte Buffers

Deze JEP voegt nieuwe JDK-specifieke bestandstoewijzingsmodi toe, zodat de FileChannel API kan worden gebruikt om `MappedByteBuffer`-instanties te maken die verwijzen naar non-volatile memory (NVM) (persistent geheugen). Deze functie wordt alleen ondersteund in Linux op dit moment.

364: ZGC on macOS / 365: ZGC on Windows

De Z Garbage Collector (ZGC) is een "scalable low latency" garbage collector.
ZGC was geïntroduceerd in Java 11, maar alleen voor Linux. Nu ook voor macOS en Windows.

PREVIEW EN INCUBATOR FEATURES

Features die in preview zijn kunnen nog compleet wijzigen in volgende versies. Het is niet te adviseren om deze in productie te gebruiken. Preview features moeten speciaal geactiveerd worden. Dit doe je met de `–source 14 ––enable–preview` parameters. De `–Xlint:preview` parameter is niet nodig, maar geeft output over wat er precies allemaal preview in de code is. Incubator modules zijn een middel om niet-definitieve API's en niet definitieve tools in handen te geven van ontwikkelaars, terwijl de API's / Tools in een toekomstige release leiden naar acceptatie of verwijdering.

305: Pattern Matching for instanceof (Preview)

`instanceof` kan gebruikt worden met een naam erachter. Deze naam functioneert dan als locale variabele als de instanceof expressie waar (true) is. De `if` statement in listing 5 leest als volgt: Als `obj` een instantie van String is cast `obj` dan naar een string en geef het aan `s` (zie listing 5).

359: Records (Preview)

Records bieden een compacte syntaxis voor het declareren van klassen die houders zijn voor onveranderlijke (immutable) gegevens (zie Listing 6).

```
$ javac NPEApp.java && java NPEApp
Exception in thread "main" java.lang.NullPointerException
    at NPEApp.main(NPEApp.java:4)
```

Listing 3.

```
$ javac NPEApp.java && java -XX:+ShowCodeDetailsInExceptionMessages
NPEApp
Exception in thread "main" java.lang.NullPointerException: Cannot in-
voke "String.length()" because "<local1>[1]" is null
    at NPEApp.main(NPEApp.java:4)
```

Listing 4.

```
public class App {
    public static void main(String[] args) {
        //Flip comment op de volgende twee regels om de else tak in te
gaan
        final Object obj = "Het Werkt!!!";
        // final Object obj = 42;
        if(obj instanceof String s) {
            // s is direct te gebruiken als: s = (String) obj;
            System.out.println(s + " <- heeft lengte: " + s.length());
        } else {
            // s is hier niet beschikbaar
            System.out.println("Niet een string");
        }
    }
}
$ javac -source 14 --enable-preview -Xlint:preview App.java
App.java:6: warning: [preview] pattern matching in instanceof is a
preview feature and may be removed in a future release.
        if(obj instanceof String s){
                          ^
1 warning
$ java --enable-preview App
Het Werkt!!! <- heeft lengte: 12
```

Listing 5.

```
record Point(int x, int y) {};
public class JEP359 {
    public static void main(String[] args) {
        final Point p1 = new Point(1, 2);
        final Point p2 = new Point(1, 2);
        if (p1.equals(p2)) {
                System.out.println(p1);
        }
    }
}
$ javac -source 14 --enable-preview -Xlint:preview JEP359.java
JEP359.java:1: warning: [preview] records are a preview feature and
may be removed in a future release.
record Point(int x, int y) {};
^
1 warning
$ java --enable-preview JEP359
Point[x=1, y=2]
```

Listing 6.

368: Text Blocks (Second Preview)

De Text blocks waren al een Preview in Java 13 (JEP 355) en het is nu nog steeds een preview in Java 14, maar nu uitgebreid met twee nieuwe escape sequences: de `\` als line-terminator en de `\s` escape sequence.

De `\s` aan het eind van de zinnen met kleuren (listing 7) garanderen dat de lengte van die zinnen precies 7 zijn en de spaties achter de kleuren blijven bestaan. Andere code voorbeelden zijn te vinden in het Java 13 artikel (**http://ivo2u.nl/oz**).

343: Packaging Tool (Incubator)

De Packaging Tool is een command line tool (`jpackage`) om platform-specifieke packages te maken van Java applicaties. Dit betekent `deb` of `rpm` bestanden op Linux, `pkg` of `dmg` bestanden om macOS en `msi` of `exe` bestanden op Windows. De code uit listing 5 wordt hergebruikt in listing 8.

In de test Docker image was ik niet in staat een `rpm` te bouwen, maar wel een zogenaamde `app-image`.
Dit levert de directory structuur op (Listing 9) met een executable in de bin folder.

370: Foreign-Memory Access API (Incubator)

Deze JEP biedt een API waarmee Java-programma's veilig en efficiënt toegang kunnen krijgen tot geheugen buiten de Java-heap. Dit kan handig zijn om:

- De kosten en de onvoorspelbaarheid van de Garbage collections te vermijden (helemaal als grote caches worden bijgehouden).
- Geheugen te delen tussen meerdere processen.
- Geheugen-content te serializeren en deserializeren door bestanden naar geheugen te laden (via bijvoorbeeld `mmap`)

Een voorbeeld is te zien in listing 10. Voor deze JEP moet een module handmatig toegevoegd moet worden met: `--add-modules jdk.incubator.foreign`

Deprecated en verwijderde Features

Deze sectie beschrijft JEP's die gemarkeerd worden als deprecated of verwijderd zijn. Deprecated betekent dat ze gevlagd worden voor toekomstige verwijdering.

```
public class JEP368 {
    public static void main(String[] args) {
        final String s = """
                Deze regel heeft geen enter hier \
                en een enkele spatie tussen de quotes: "\s"
                groen \s
                rood  \s
                blauw \s
                """;
        System.out.println(s);
    }
}
$ javac -source 14 --enable-preview JEP368.java && java --enable-pre-
view JEP368
Note: JEP368.java uses preview language features.
Note: Recompile with -Xlint:preview for details.
Deze regel heeft geen enter hier en een enkele spatie tussen de
quotes: " "
groen
rood
blauw
```

Listing 7.

```
# Compile de voorbeeld code
$ javac -source 14 --enable-preview App.java
Note: App.java uses preview language features.
Note: Recompile with -Xlint:preview for details.
# Maak een manifest file zoals deze...
$ cat App.mf
Manifest-Version: 1.0
Main-Class: App
# Maak de jar file...
$ jar cmf App.mf app.jar App.class App.java
# Package de applicatie naar een zogenaamde app-image...
$ jpackage --name app --input . --main-jar app.jar \
  --type app-image --java-options "--enable-preview"
WARNING: Using incubator modules: jdk.incubator.jpackage
# Kijken of het werkt...
$ ./app/bin/app
Het Werkt!!! <- heeft lengte: 12
```

Listing 8.

```
.
└── app
    ├── bin
    └── lib
        ├── app
        └── runtime
            ├── conf
            │   ├── management
            │   ├── sdp
            │   └── security
            │       └── policy
            ├── legal
            │   ├── java.base
            │   ├── [nog veel meer items hier ...]
            └── lib
                ├── jfr
                ├── security
                └── server
```

Listing 9.

```
import jdk.incubator.foreign.MemoryAddress;
import jdk.incubator.foreign.MemorySegment;
public class JEP370 {
    public static void main(String[] args) {
        try (MemorySegment segment = MemorySegment.allocateNative(1024)) {
                MemoryAddress base = segment.baseAddress();
                System.out.println(base);
        }
    }
}
$ javac -source 14 --enable-preview --add-modules jdk.incubator.foreign JEP370.java && java --enable-preview
--add-modules jdk.incubator.foreign JEP370
warning: using incubating module(s): jdk.incubator.foreign
1 warning
WARNING: Using incubator modules: jdk.incubator.foreign
MemoryAddress{ region: MemorySegment{ id=0x683ceb2a limit: 1024 } offset=0x0 }
```

Listing 10.

362: Deprecate the Solaris and SPARC Ports

Solaris/SPARC, Solaris/x64, and Linux/SPARC ports zijn deprecated verklaard en zullen in een toekomstige release verwijderd worden.

363: Remove the Concurrent Mark Sweep (CMS) Garbage Collector

CMS Garbage Collection was deprecated verklaard in Java 9 en is nu in Java 14 weggehaald.

366: Deprecate the ParallelScavenge + SerialOld GC Combination

Deze zijn deprecated verklaard omdat ze weinig gebruikt werden en zullen in een toekomstige release verwijderd worden.

367: Remove the Pack200 Tools and API

De Pack200 Tools en API zijn in Java 11 deprecated verklaard en in Java 14 verwijderd.

Conclusie

Een hoop JEP's zijn de revue gepasseerd en een paar leuke features zijn erbij gekomen. Voor de developers is de switch nu standaard geworden en we kunnen nu betere NPE tracing doen. De multi-line string is aan het evolueren en een hoop oud zeer is weggehaald. Al met al een mooie release.

REFERENTIES:

http://openjdk.java.net/projects/jdk/14/
http://ivo2u.nl/oz
https://www.oracle.com/technetwork/java/java-se-support-roadmap.html