JAVA 14: DIESE JEPS SIND TEIL DES JDKS

JEP 305 – PATTERN MATCHING FOR INSTANCEOF (PREVIEW)

Im Zuge des Projektes Amber wird unter anderem ein Pattern Matching für den Operator instanceof eingeführt. Dies Operator instanceof wird dann Pattern Matching in zwei Variablen speichern. Die Variablen können dann innerhalb eines spezifischen Bereichs verwendet werden. Die „Form" der Objekte kann so präzise definiert werden, wenn/für diese Variablen Methoden aufgerufen werden sollen. Dadurch lassen sich Fehler vermeiden. Bei Nutzung von Pattern Matching für instanceof kommt hier einer starken Reduzierung. Weiter Tipp: Pattern Matching für instanceof ist zunächst noch nur als Preview gem. Java Versionen 14 und 15 Pattern Matching für switch Spracherweiterungen wurde zunächst Expressions kommen.

JEP 343 – PACKAGING TOOL

JEP 345 – NUMA-AWARE MEMORY ALLOCATION FOR G1

Wie Tiefe gedacht, dass man sowohl erstellt dazu noch Nutzer Interface Ebene Ebene Ebene

JAVA 14: DIESE JEPS SIND TEIL DES JDKS

JEP 305 – PATTERN MATCHING FOR INSTANCEOF (PREVIEW)

Im Zuge des Projekts Amber wird unter anderem ein Mechanismus entwickelt, der es ermöglicht, den Operator instanceof auf Muster anwenden zu lassen. Ein Muster besteht aus einem Ausdruck und einer Variable, die nach erfolgreicher Musterprüfung deklariert wird. Dadurch lassen sich zukünftig Fehler beim Casten vermeiden und der Code wird lesbarer und kompakter.

JEP 343 – PACKAGING TOOL

Das jetzt JDK 14 enthaltene Packaging Tool ermöglicht das Verpacken von Java-Anwendungen für die Installation auf den Systemen von Mac OS X, Linux und Windows. Es basiert auf den Projekten JavaFX Packager und Installer Designer und kann sowohl Module als auch Nicht-Module verpacken. Es kann sowohl in der Kommandozeile als auch über Ant aufgerufen werden.

JEP 345 – NUMA-AWARE MEMORY ALLOCATION FOR G1

Nachdem Betriebssysteme und Architekturen als allgemeinen Trend immer mehr Speicher bereitstellen, wird der Speicherzugriff zum Flaschenhals für die Leistung der Anwendungen. Mit NUMA (Non-Uniform Memory Access) wird versucht, dieses Problem zu lösen, indem Speicher, der einem Prozessor zugeordnet ist, schneller als Speicher, der einem anderen Prozessor zugeordnet ist, zugreifen kann. Der JEP soll dabei Gewinne von 10 % bis 20 % bei Benchmarks und 9 % bei einer Datenbank-Last erzielen. Das Ziel ist, den Garbage Collector mit den Mitteln NUMA-fähig zu machen, die das Betriebssystem bereitstellt. Der JEP wird zunächst nur für den Garbage Collector G1 implementiert, der auf Linux x86 und Linux SPARC läuft. Weitere Plattformen sollen folgen, wenn sich die Implementierung als stabil erwiesen hat.

JEP 349 – JFR EVENT STREAMING

Ab Version 9 liefert das Java Flight Recorder Tool (JFR) und Java Mission Control (JMC) JFR 2.0 im JDK mit. Dadurch wird das Profiling und Monitoring von Java-Anwendungen ermöglicht, die in Produktion laufen. Bei einer kontinuierlichen Überwachung von JVMs und Java-Anwendungen gibt es jedoch Einschränkungen, die JFR beheben soll. Zum Beispiel können Ereignisse nur in Dateien auf der Festplatte geschrieben werden, die erst nach Beendigung des Prozesses für die Analyse verfügbar sind. Dieses Verhalten ist nicht optimal, wenn das Monitoring, die Detektion von Problemen und die Analyse von Ereignissen in Echtzeit erfolgen soll. Ein Ziel des JEP ist es, Ereignisse in Echtzeit verfügbar zu machen, um Probleme schnellstmöglich erkennen zu können. Es soll außerdem eine Schnittstelle für Ereignisse, Checkpoints und Metadaten hinzugefügt werden, die es ermöglicht, eigene Ereignisse zu definieren und Ereignisdaten zu konsumieren. Die Ziele des JEP sind, die Performance und Skalierbarkeit von JFR zu verbessern, das Ereignisformat zu dokumentieren und das Streaming von Ereignissen zu ermöglichen, sowohl lokal wie Persistenzlos, als im Netzwerk. Es soll