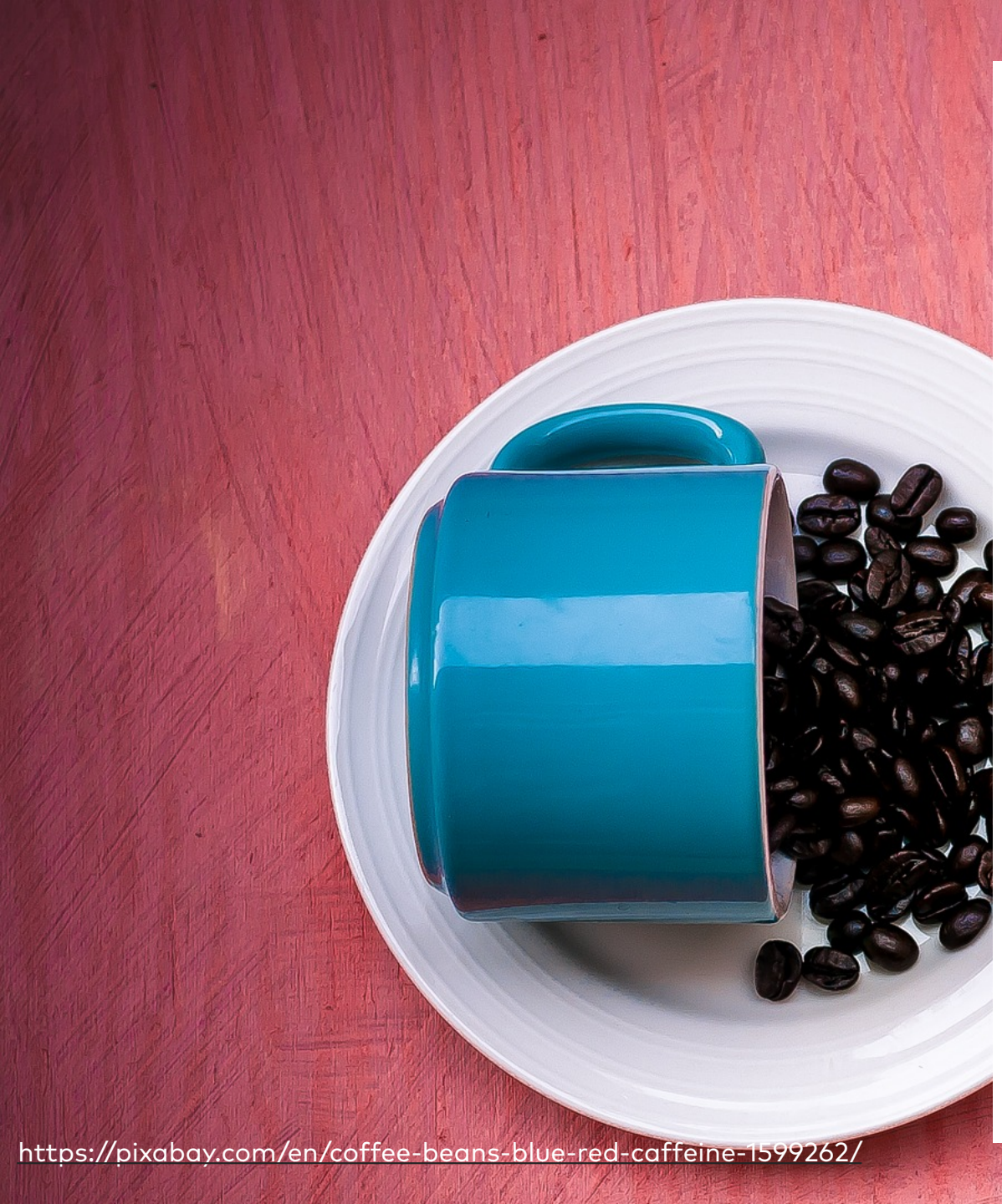


27.06.2018

Nürnberg / DWX2018

Java 10 and beyond

Michael Vitz

@michaelvitz

INNOQ

https://pixabay.com/en/coffee-beans-blue-red-caffeine-1599262/

Michael Vitz

Senior Consultant
bei INNOQ Deutschland GmbH

- Konzeption, Entwicklung, Wartung und Betrieb von JVM Anwendungen
- JavaSPEKTRUM Kolumnist
- ❤️Clojure

www.innoq.com

SERVICES

Strategy & technology consulting
Digital business models
Software architecture & development
Digital platforms & infrastructures
Knowledge transfer, coaching & trainings

FACTS

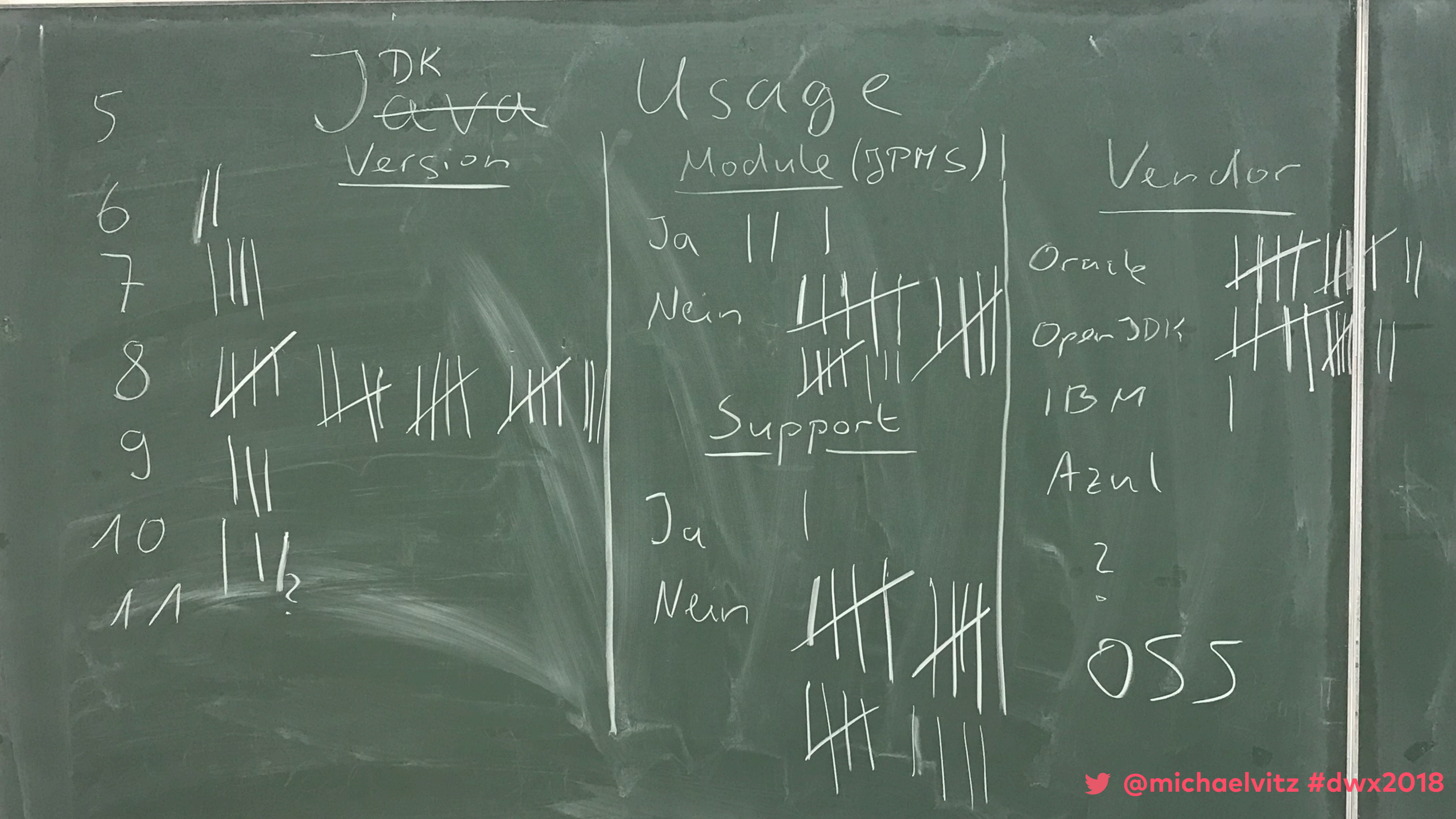
~125 employees
Privately owned
Vendor-independent

OFFICES

Monheim
Berlin
Offenbach
Munich
Zurich

CLIENTS

Finance
Telecommunications
Logistics
E-commerce
Fortune 500
SMBs
Startups

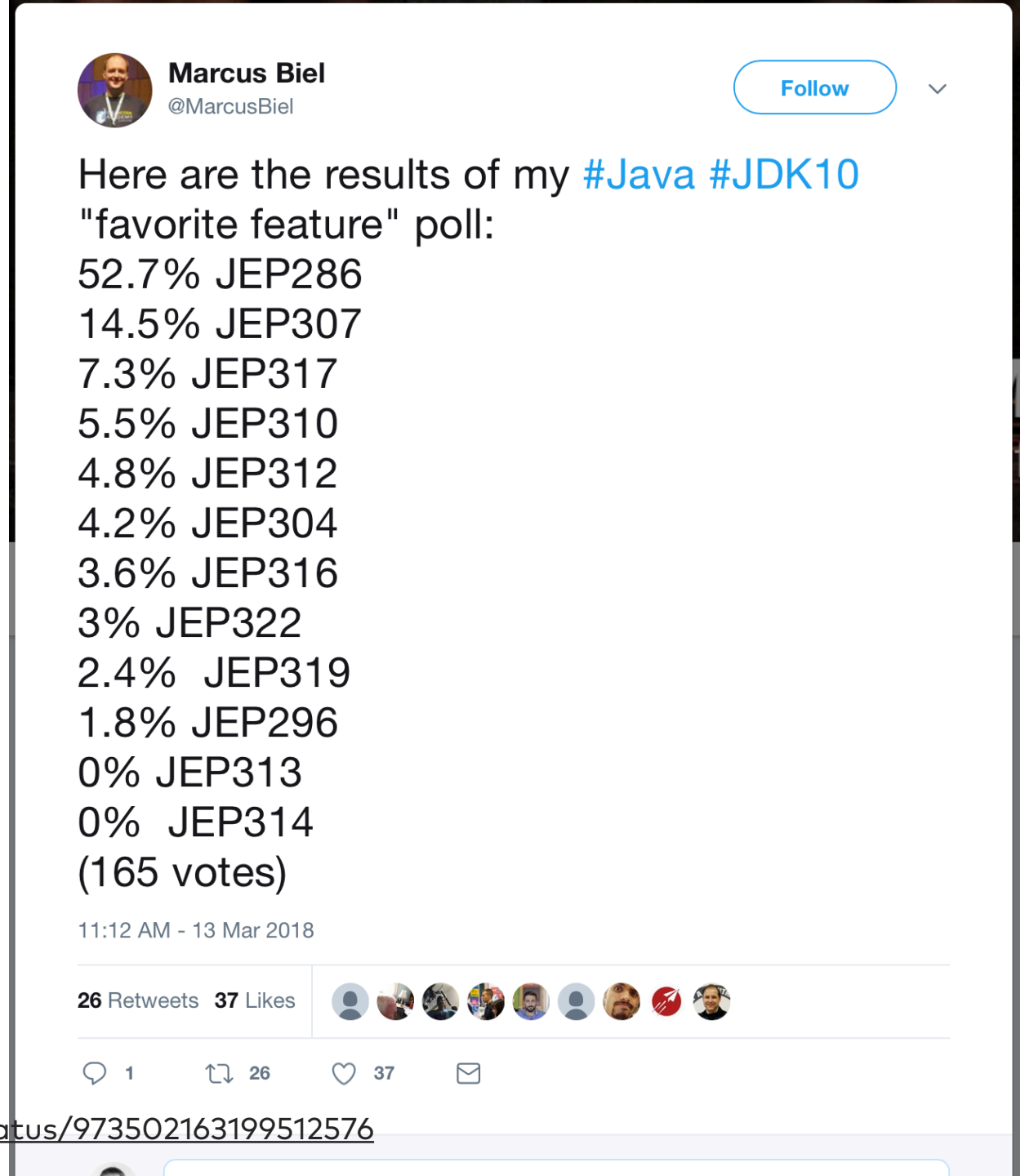

JDK
~~eva~~
Version
5
6 ||
7 |||
8 ||||
9 |||
10 ||
11 ?
Usage
Module (JPMS)
Ja || |
Nein
Support
Ja |
Nein
Vendor
Oracle
OpenJDK
IBM |
Azul
2
055
@michaelvitz #dwx2018

```
public class Quiz1 {

    public static void main(String[] args) {
        final String[] array = {""};
        array[generateArrayIndex()] += "a";
    }

    static int generateArrayIndex() {
        System.out.println("Array Index Evaluated");
        return 0;
    }
}
```

http://marxsoftware.blogspot.com/2018/06/JDK-8204322.html

JDK 10

https://pixabay.com/en/number-ad-yellow-color-asphalt-437928/



https://twitter.com/MarcusBiel/status/973502163199512576

@michaelvitz #dwx2018

- **JEP314: Additional Unicode Language-Tag Extensions**
  http://openjdk.java.net/jeps/314

- **JEP313: Remove the Native-Header Generation Tool**
  http://openjdk.java.net/jeps/313

- **JEP296: Consolidate the JDK Forest into a Single Repository**
  http://openjdk.java.net/jeps/296

- **JEP319: Root Certificates**
  http://openjdk.java.net/jeps/319

```
[1-9][0-9]*((\.0)*\.[1-9][0-9]*)*
```

JEP322:Time-Based Release Versioning

http://openjdk.java.net/jeps/322

- **JEP316: Heap Allocation on Alternative Memory Devices**
  http://openjdk.java.net/jeps/316

- **JEP304: Garbage-Collector Interface**
  http://openjdk.java.net/jeps/304

- **JEP312: Thread-Local Handshakes**
  http://openjdk.java.net/jeps/312

- **JEP310: Application Class-Data Sharing**
  http://openjdk.java.net/jeps/310

- **JEP317: Experimental Java-Based JIT Compiler**
  http://openjdk.java.net/jeps/317

- **JEP307: Parallel Full GC for G1**
  http://openjdk.java.net/jeps/307

var

Cheat Sheet: https://snyk.io/blog/local-type-inference-java-cheat-sheet/
Style Guidelines: http://openjdk.java.net/projects/amber/LVTIstyle.html

JEP286: Local-Variable Type Inference

http://openjdk.java.net/jeps/286

OpenJDK

Workshop
OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide

Mailing lists
IRC · Wiki

Bylaws · Census
Legal

JEP Process

search

Source code
Mercurial
Bundles (6)

Groups
(overview)
2D Graphics
Adoption
AWT
Build
Compatibility & Specification Review
Compiler
Conformance
Core Libraries
Governing Board
HotSpot
Internationalization
JMX
Members
Networking
NetBeans Projects
Porters
Quality

JDK 10

JDK 10 is the open-source reference implementation of the Java SE 10 Platform as defined by JSR 383 in the Java Community Process.

JDK 10 reached General Availability on 20 March 2018. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal.

Features

286: Local-Variable Type Inference
296: Consolidate the JDK Forest into a Single Repository
304: Garbage-Collector Interface
307: Parallel Full GC for G1
310: Application Class-Data Sharing
312: Thread-Local Handshakes
313: Remove the Native-Header Generation Tool (javah)
314: Additional Unicode Language-Tag Extensions
316: Heap Allocation on Alternative Memory Devices
317: Experimental Java-Based JIT Compiler
319: Root Certificates
322: Time-Based Release Versioning

Schedule

2017/12/14 Rampdown Phase One

http://openjdk.java.net/projects/jdk/10/

Weitere “Kleinigkeiten”

- @summary
- 
- java.util.Collection#copyOf
- java.util.stream.Collectors#toUnmodifiableXxx
- java.util.Optional#orElseThrow

JDK 11

https://pixabay.com/en/pool-balls-billiard-eleven-eight-1003955/

OpenJDK

Workshop
OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide

Mailing lists
IRC · Wiki

Bylaws · Census
Legal

JEP Process

search

Source code
Mercurial
Bundles (6)

Groups
(overview)
2D Graphics
Adoption
AWT

JDK 11

This release will be the Reference Implementation of a future version of the Java SE Platform, as specified by JSR 384 in the Java Community Process.

Status

The development repositories are open for bug fixes, small enhancements, and JEPs as proposed and tracked via the JEP Process.

Schedule

2018/06/28	Rampdown Phase One (fork from main line)
2018/07/19	All Tests Run
2018/07/26	Rampdown Phase Two
2018/08/16	Initial Release Candidate
2018/08/30	Final Release Candidate
2018/09/25	General Availability

http://openjdk.java.net/projects/jdk/11/

Conformance
Core Libraries
Governing Board
HotSpot
Internationalization
JMX
Members
Networking
NetBeans Projects
Porters
Quality
Security
Serviceability
Sound
Swing
Vulnerability
Web

Projects
(overview)
Amber
Annotations Pipeline 2.0
Audio Engine
Build Infrastructure
Caciocavallo
Closures
Code Tools
Coin
Common VM Interface
Compiler Grammar
Detroit
Device I/O
Duke
Font Scaler
Framebuffer Toolkit
Graal
Graphics Rasterizer
HarfBuzz Integration
IcedTea

Features

JEPs proposed to target JDK 11

	review ends
335: Deprecate the Nashorn JavaScript Engine	2018/06/26

JEPs targeted to JDK 11, so far

181: Nest-Based Access Control
309: Dynamic Class-File Constants
315: Improve Aarch64 Intrinsics
318: Epsilon: A No-Op Garbage Collector
320: Remove the Java EE and CORBA Modules
321: HTTP Client (Standard)
323: Local-Variable Syntax for Lambda Parameters
324: Key Agreement with Curve25519 and Curve448
327: Unicode 10
328: Flight Recorder
329: ChaCha20 and Poly1305 Cryptographic Algorithms
330: Launch Single-File Source-Code Programs
331: Low-Overhead Heap Profiling
332: Transport Layer Security (TLS) 1.3
333: ZGC: A Scalable Low-Latency Garbage Collector (Experimental)
336: Deprecate the Pack200 Tools and API

Last update: 2018/6/25 21:15 UTC

http://openjdk.java.net/projects/jdk/11/

Summary

Upgrade existing platform APIs to support version 10.0 of the Unicode Standard.

Goals

Support the latest version of Unicode, mainly in the following classes:

- `Character` and `String` in the `java.lang` package,
- `NumericShaper` in the `java.awt.font` package, and
- `Bidi`, `BreakIterator`, and `Normalizer` in the `java.text` package.

Non-Goals

Four related Unicode specifications will not be implemented by this JEP:

- UTS #10, Unicode Collation Algorithm
- UTS #39, Unicode Security Mechanisms
- UTS #46, Unicode IDNA Compatibility Processing
- UTS #51, Unicode Emoji

JEP327: Unicode 10

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
        .uri(URI.create("http://openjdk.java.net/"))
        .build();
client.sendAsync(request, asString())
        .thenApply(HttpResponse::body)
        .thenAccept(System.out::println)
        .join();
```

http://openjdk.java.net/groups/net/httpclient/intro.html

JEP321: HTTP Client

http://openjdk.java.net/jeps/321

```
javac Foo.java \
  && java Foo \
  && rm Foo.class
```

JEP330: Launch Single-File Source-Code Programs

java.io

- java.io.FileReader(java.lang.String, java.nio.charset.Charset)
- java.io.FileWriter(java.lang.String, java.nio.charset.Charset)

- java.io.InputStream#nullInputStream
- java.io.OutputStream#nullOutputStream

- java.io.Reader#nullReader
- java.io.Writer#nullWriter

java.lang.String

- `String::repeat(int)`
  https://bugs.openjdk.java.net/browse/JDK-8197594
- `String::lines`
  https://bugs.openjdk.java.net/browse/JDK-8200380
- `String::strip`, `String::stripLeading`, `String::stripTrailing`
  https://bugs.openjdk.java.net/browse/JDK-8200377
- `String::isBlank`
  https://bugs.openjdk.java.net/browse/JDK-8200436

java.util.Optional/Stream/Predicate

- `Optional::isEmpty`
  https://bugs.openjdk.java.net/browse/JDK-8184693
- `Stream::toList`
  https://bugs.openjdk.java.net/browse/JDK-8180352
- `Predicate#not`
  https://bugs.openjdk.java.net/browse/JDK-8050818

java.nio.file.Files/Path

- `Files#isSameContent`
  https://bugs.openjdk.java.net/browse/JDK-8202302
- `Files#readString, Files#writeString`
  https://bugs.openjdk.java.net/browse/JDK-8202055
- `Path#of(URI), Path#of(String, String…)`
  https://bugs.openjdk.java.net/browse/JDK-8199485

java.lang.Character/CharSequence

- `Character#toString(int)`
  https://bugs.openjdk.java.net/browse/JDK-8198837

- `CharSequence#compare(CharSequence, CharSequence)`
  https://bugs.openjdk.java.net/browse/JDK-8195867

java.util.regex.Pattern

- `Files::asMatchPredicate`
  https://bugs.openjdk.java.net/browse/JDK-8201308

java.lang.Thread

- `Thread::stop(Throwable), Thread::destroy`
  https://bugs.openjdk.java.net/browse/JDK-8204243

Zukunft

https://pixabay.com/en/binoculars-looking-man-discovery-1209011/

Summary

Define the process by which contributors in the OpenJDK Community produce time-based, rapid-cadence JDK feature releases.

JEP 3: JDK Release Process

Summary

A preview language or VM feature is a new feature of the Java SE Platform that is fully specified, fully implemented, and yet impermanent. It is available in a JDK feature release to provoke developer feedback based on real world use; this may lead to it becoming permanent in a future Java SE Platform.

JEP 12: Preview Language and VM Features

Summary

Introduce a modernized general-purpose hash function into the JVM, usable by arbitrary classes and arrays.

Enable new classes to use it as a back-end for the standard `Object.hashCode` API.

Supply APIs for legacy types (including arrays and collections) to begin to make use of this hash code.

JEP draft: Better hash codes

Summary

Make the G1 garbage collector optionally automatically give back memory to the operating system when idle.

JEP draft: Timely Reducing Unused Committed Memory

http://openjdk.java.net/jeps/8204089

Support

Java SE Public Updates				
Release	**GA Date**	**End of Public Updates Notification**	**Commercial User End of Public Updates**	**Personal User End of Public Updates**
7	July 2011	March 2014	April 2015	
8	March 2014	September 2017	January 2019****	December 2020****
9 (non-LTS)	September 2017	September 2017	March 2018	
10 (18.3^) (non-LTS)	March 2018	March 2018	September 2018	
11 and later (non-LTS)		No longer Applicable		

http://www.oracle.com/technetwork/java/javase/eol-135779.html

Oracle Java SE Support Roadmap*†				
Release	**GA Date**	**Premier Support Until** Notification**	**Extended Support Until****	**Sustaining Support****
6	December 2006	December 2015	December 2018	Indefinite
7	July 2011	July 2019	July 2022	Indefinite
8	March 2014	March 2022	March 2025	Indefinite
9 (non-LTS)	September 2017	March 2018	Not Available	Indefinite
10 (18.3^) (non-LTS)	March 2018	September 2018	Not Available	Indefinite
11 (18.9^ LTS)	September 2018***	September 2023	September 2026	Indefinite
12 (19.3^ non-LTS)	March 2019***	September 2019	Not Available	Indefinite

http://www.oracle.com/technetwork/java/javase/eol-135779.html

Java SE Subscription Pricing		
Volume	**Subscription Metric**	**Monthly Subscription Price**
1-99	Processor	$25.00
100-249	Processor	$23.75
250-499	Processor	$22.50
500-999	Processor	$20.00
1,000-2,999	Processor	$17.50
3,000-9,999	Processor	$15.00
10,000-19,999	Processor	$12.50
20,000+	Contact for details	

http://www.oracle.com/technetwork/java/javaseproducts/overview/javasesubscriptionfaq-4891443.html

Support of Deployment Technology and JavaFX

The web deployment technology, consisting of the Java Plugin and Web Start technologies, has a shorter support lifecycle. For major releases through Java SE 7, and for the Java SE 8 Plugin, Oracle provides five (5) years of Premier Support for these technologies. Java SE 8 will continue to support Web Start through Premiere and Extended Support timeframes. Java deployment technology will not be supported beyond Java SE 8. See the Oracle Lifetime Support Policy for details.

Deployment Technology for Java SE 6 and Java SE 7 has been removed as of October 2017. Although the deployment stack may be included in Java SE 9 or later releases, Java SE 8 is the recommended and only supported version of the deployment stack. The Java SE 8 deployment stack may be used to run Java SE 6, or Java SE 7 applications on Windows platforms.

JavaFX will be removed from Oracle Java SE products starting with Java SE 11 (18.9 LTS). Support for JavaFX on Java SE 8 will continue through the entire Premiere Support term, March 2022.

http://www.oracle.com/technetwork/java/javase/eol-135779.html

- Oracle will no longer offer a stand-alone JRE for desktops. Starting with JDK 11 Oracle will only produce a JDK and a Server JRE.
- The Alpine Linux build runs on Linux distributions that use the musl C library.

http://jdk.java.net/11/

Project Amber

https://pixabay.com/en/heart-amber-pendant-necklace-1202129/

OpenJDK

Workshop
OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide

Mailing lists
IRC · Wiki

Bylaws · Census
Legal

JEP Process

search

Source code
Mercurial
Bundles (6)

Groups
(overview)
2D Graphics
Adoption
AWT
Build
Compatibility & Specification Review
Compiler
Conformance
Core Libraries
Governing Board
HotSpot
Internationalization
JMX
Members
Networking

Project Amber

The goal of Project Amber is to explore and incubate smaller, productivity-oriented Java language features that have been accepted as candidate JEPs under the OpenJDK JEP process. This Project is sponsored by the Compiler Group.

Status of JEPs

Currently in progress:

- JEP 302 Lambda Leftovers
- JEP 305 Pattern Matching
- JEP 323 Local-Variable Syntax for Lambda Parameters
- JEP 325 Switch Expressions
- JEP 326 Raw String Literals

Delivered in JDK 10:

- JEP 286 Local-Variable Type Inference (`var`). See the style guidelines for using `var`.

On hold:

- JEP 301 Enhanced Enums. See explanation.

Documents

- Data Classes for Java *(February 2018)*
- Style Guidelines for Local Variable Type Inference *(March 2018)*

```
BiFunction<Integer, String, String> biss =
    (i, _) -> String.valueOf(i);


Map<String, Integer> msi = ...
...
String key = computeSomeKey();
msi.computeIfAbsent(key, key -> key.length())
```

Optional: Better disambiguation for functional expression

JEP302: Lambda Leftovers

```
String formatted;
switch (obj) {
    case Integer i: formatted = String.format("int %d", i); break;
    case Byte b:    formatted = String.format("byte %d", b); break;
    case Long l:    formatted = String.format("long %d", l); break;
    case Double d:  formatted = String.format("double %f", l); break;
    case String s:  formatted = String.format("String %s", s); break;
    default:        formatted = obj.toString();
}
```

JEP305: Pattern Matching

```
int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> 6;
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    case WEDNESDAY -> 9;
    case null -> 0;
};
```

JEP325: Switch Expressions

http://openjdk.java.net/jeps/325

```
String query = ``
                 SELECT `EMP_ID`, `LAST_NAME` FROM `EMPLOYEE_TB`
                 WHERE `CITY` = 'INDIANAPOLIS'
                 ORDER BY `EMP_ID`, `LAST_NAME`;
               ``;
```

JEP326: Raw String Literals

```
public class Quiz2 {

    public static void main(String[] args) {
        Double doubleValue = false
            ? 1.0
            : new HashMap<String, Double>().get("1");
        System.out.println(“Value: " + doubleValue);
    }
}
```

http://marxsoftware.blogspot.com/2018/06/jdk8-jdk10-ternary-boxing-unboxing.html

Danke! Fragen?

INNOQ
www.innoq.com

Michael Vitz
michael.vitz@innoq.com

+49 151 19116015
michaelvitz

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin
Germany
+49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
+49 2173 3366-0

Kreuzstr. 16
80331 München
Germany
+49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
+41 41 743 0116