



2018-11-27

Cologne / JVM-Con 2018

Java 10, 11, 12, and beyond

Michael Vitz

INNOQ

 **@michaelvitz #jvmcon18**



Michael Vitz

Senior Consultant
at INNOQ

- Build, run, and maintain JVM applications
- JavaSPEKTRUM column owner
- ❤️ 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

COMMUNITY

Vote

5 JDK
Java
Version

6 ||

7 ||||

8 ~~||||~~ ~~||||~~ ~~||||~~ ~~||||~~ ||||

9 |||

10 | ||

11 | ||

Usage

Module (JPMs)

Ja || |

Nein ~~||||~~ ~~||||~~ ~~||||~~ ~~||||~~

Support

Ja |

Nein ~~||||~~ ~~||||~~
~~||||~~ ||||

Vendor

Oracle ~~||||~~ ~~||||~~ ||

OpenJDK ~~||||~~ ~~||||~~ ||

IBM |

Azul

2

055



JDK 10



Workshop

[OpenJDK FAQ](#)

[Installing](#)

[Contributing](#)

[Sponsoring](#)

[Developers' Guide](#)

[Mailing lists](#)

[IRC](#) · [Wiki](#)

[Bylaws](#) · [Census](#)

[Legal](#)

JEP Process

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](#).

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

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

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>

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

JEP322:Time-Based Release Versioning

More small enhancements

- @summary
<https://bugs.openjdk.java.net/browse/JDK-8173425>
-  docker
<https://bugs.openjdk.java.net/browse/JDK-8146115>
- List#copyOf, Map#copyOf, Set#copyOf
<https://bugs.openjdk.java.net/browse/JDK-8177290>
- Collectors#toUnmodifiableList/Set/Map
<https://bugs.openjdk.java.net/browse/JDK-8184690>
- Optional::orElseThrow
<https://bugs.openjdk.java.net/browse/JDK-8140281>



JDK 11



Workshop

[OpenJDK FAQ](#)

[Installing](#)

[Contributing](#)

[Sponsoring](#)

[Developers' Guide](#)

[Mailing lists](#)

[IRC](#) · [Wiki](#)

[Bylaws](#) · [Census](#)

[Legal](#)

JEP Process

Source code

[Mercurial](#)

JDK 11

JDK 11 is the open-source reference implementation of version 11 of the Java SE 11 Platform as specified by [JSR 384](#) in the Java Community Process.

JDK 11 reached [General Availability](#) on 25 September 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](#). The release was produced using the JDK Release Process (JEP 3).

Features

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

Security

Serviceability

Sound

Swing

Vulnerability

Web

Projects

(overview)

Features

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)

335: Deprecate the Nashorn JavaScript Engine

336: Deprecate the Pack200 Tools and API

Schedule

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

```
<jaxb with="jdk11">  
  <is provided="false" />  
</jaxb>
```

JEP320: Remove the Java EE and CORBA Modules

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create("http://openjdk.java.net/"))
    .build();
client.sendAsync(request, ofString())
    .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>

`(@Nonnull var x) -> process(x)`

JEP323: Local-Variable Syntax for Lambda Parameters

```
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/Predicate

- Optional::isEmpty
<https://bugs.openjdk.java.net/browse/JDK-8184693>
- 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

- Pattern::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>

Distribution License Support

Java SE Development Kit 11 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

Important changes in Oracle JDK 11 License

With JDK 11 Oracle has updated the license terms on which we offer the Oracle JDK.

The new [Oracle Technology Network License Agreement for Oracle Java SE](#) is substantially different from the licenses under which previous versions of the JDK were offered. Please review the new terms carefully before downloading and using this product.

Oracle also offers this software under the [GPL License](#) on jdk.java.net/11

Oracle JDK and OpenJDK builds from Oracle

Starting with Java SE 9, in addition to providing Oracle JDK for free under the [BCL](#), Oracle also started providing builds of [OpenJDK](#) under an [open source license](#) (similar to that of Linux). Oracle is working to make the [Oracle JDK and OpenJDK builds from Oracle interchangeable](#) - targeting developers and organizations that do not want commercial support or enterprise management tools. Beginning with Oracle Java SE 11 (18.9 LTS), the Oracle JDK will continue to be available royalty-free for development, testing, prototyping or demonstrating purposes. As [announced in September 2017](#), with the OracleJDK and builds of Oracle OpenJDK being interchangeable for releases of Java SE 11 and later, the Oracle JDK will primarily be for commercial and support customers and OpenJDK builds from Oracle are for those who do not want commercial support or enterprise management tools.

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			No longer Applicable	

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

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>

GA Releases

[JDK 11](#)

[JDK 10](#)

Early-Access Releases

[JDK 12](#)

[JDK 8](#)

[OpenJFX](#)

[Valhalla](#)

[JMC](#)

Reference

Implementations

[Java SE 11](#)

[Java SE 10](#)

[Java SE 9](#)

[Java SE 8](#)

[Java SE 7](#)

Feedback

[Report a bug](#)

Archive

JDK 11 General-Availability Release

This page provides production-ready open-source builds of the [Java Development Kit, version 11](#), an implementation of the [Java SE 11 Platform](#) under the [GNU General Public License, version 2](#), with the [Classpath Exception](#).

Commercial builds of JDK 11 from Oracle under a [non-open-source license](#), for a wider range of platforms, can be found at the [Oracle Technology Network](#).

Documentation

- [Features](#)
- [Release notes](#)
- [API Javadoc](#)
- [Tool & command reference](#)

Downloads

Linux/x64	tar.gz (sha256)	187611826 bytes
macOS/x64	tar.gz (sha256)	182717049
Windows/x64	zip (sha256)	187396683

Notes



4th Oct 2018: AdoptOpenJDK Java 11 has been released! AdoptOpenJDK Java 1.8.0_181 is coming soon.



Prebuilt OpenJDK Binaries

Java™ is the world's leading programming language and platform. The code for Java is [open source](#) and available at [OpenJDK™](#). AdoptOpenJDK provides prebuilt OpenJDK binaries from a fully open source set of [build scripts](#) and infrastructure. Get [Docker Images on Docker Hub](#). Nightlies can be found in the [Archive](#).

The future of Java and OpenJDK updates without Oracle support



By [Andrew Haley](#) September 24, 2018
aphredhat



+13 rating, 13 votes

OpenJDK

Oracle recently announced that it would no longer supply free (as in beer) binary downloads for JDK releases after a six-month period, and neither would Oracle engineers write patches for OpenJDK bugs after that period. This has caused a great deal of concern among some Java users.

From my point of view, this is little more than business as usual. Several years ago, the OpenJDK 6 updates (jdk6u) project was relinquished by Oracle and I assumed leadership, and then the same happened with OpenJDK 7. Subsequently, Andrew Brygin of Azul took over the leadership of OpenJDK 6. The OpenJDK Vulnerability Group, with members from many organizations, collaborates on critical security issues. With the help of the wider OpenJDK community and my team at Red Hat, we have continued to provide updates for critical bugs and security vulnerabilities at regular intervals. I can see no reason why this process should not work in the same way for OpenJDK 8 and the next long-term support release, OpenJDK 11.

<https://developers.redhat.com/blog/2018/09/24/the-future-of-java-and-openjdk-updates-without-oracle-support/>

[Products](#)[Solutions](#)[Resources](#)[Support](#)[Company](#)[Blog](#)[日本語](#) [中文](#)[Contact Us](#)

Download Zulu[®] tested, certified builds of OpenJDK

Zulu is Free to Download and Use.

Zulu[®] is a certified build of [OpenJDK](#) that is fully compliant with the Java SE standard. Zulu is 100% open source and freely downloadable. Now Java developers, system administrators, and end users can enjoy the full benefits of open source Java with deployment flexibility and control over upgrade timing. [Zulu Enterprise](#) adds the comfort of cost-effective world-class, subscription-based [Java support plans](#).

Need Java in a custom form factor? [Zulu Embedded](#) is ideal for OEMs, embedded and IoT device manufacturers as well as the market

<https://www.azul.com/downloads/zulu/>

Zulu[®] Enterprise Support Pricing

Zulu[®] Enterprise OpenJDK Support is priced on a subscription basis based upon the number of systems (desktops and/or physical or virtual servers) running Java applications. Here's our pricing:

Max # of supported systems	Price/year (Standard Support)	Price/year (Premium Support)
25	\$13,200	Not available
100	\$31,600	\$37,900
1,000	\$94,900	\$113,900
Unlimited	\$284,600	\$341,500

[Blog](#) / [Announcements](#)

Microsoft and Azul Systems bring free Java LTS support to Azure

Veröffentlicht am 25 September, 2018



[Asir Selvasingh](#), Principal Program Manager, Azure Developer Experience

Java developers on Azure and Azure Stack can build and run production Java applications using Azul Systems Zulu Enterprise builds of OpenJDK without incurring additional support costs. If you're currently running Java apps on-premise or with other JDKs, consider moving it to Zulu on Azure for free support and maintenance. Go



Abonnieren

Durchsuchen

Wie sieht die Zukunft aus?
Sehen Sie sich
bevorstehende Änderungen
an Azure-Produkten an.

[Azure-Roadmap](#)

Teilen Sie uns mit, was Sie

Amazon Corretto

No-cost, multiplatform, production-ready distribution of OpenJDK

Amazon Corretto is a no-cost, multiplatform, production-ready distribution of the Open Java Development Kit (OpenJDK). Corretto comes with long-term support that will include performance enhancements and security fixes. Amazon runs Corretto internally on thousands of production services and Corretto is certified as compatible with the Java SE standard. With Corretto, you can develop and run Java applications on popular operating systems, including Amazon Linux 2, Windows, and macOS. Amazon Corretto 8 is in Preview.

Visit our [documentation](#) to learn more.

Amazon Corretto 8 is in Preview

Download the Preview

Installation Guides

[Amazon Linux 2](#)

[Microsoft Windows](#)

[macOS](#)

[Docker](#)



JDK 12

Workshop

[OpenJDK FAQ](#)
[Installing](#)
[Contributing](#)
[Sponsoring](#)
[Developers' Guide](#)

[Mailing lists](#)
[IRC · Wiki](#)

[Bylaws · Census](#)
[Legal](#)

JEP Process

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](#)

JDK 12

This release will be the Reference Implementation of version 12 of the Java SE Platform, as specified by [JSR 386](#) 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

2018/12/13	Rampdown Phase One (fork from main line)
2019/01/17	Rampdown Phase Two
2019/02/07	Release-Candidate Phase
2019/03/19	General Availability

Features

JEPs targeted to JDK 12, so far

- 230: [Microbenchmark Suite](#)
- 325: [Switch Expressions \(Preview\)](#)
- 326: [Raw String Literals \(Preview\)](#)
- 340: [One AArch64 Port, Not Two](#)
- 341: [Default CDS Archives](#)

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

JEP325: Switch Expressions

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

JEP326: Raw String Literals

API Changes

- `{@systemProperty}`
<https://bugs.openjdk.java.net/browse/JDK-5076751>
- `Files#mismatch(Path, Path)`
<https://bugs.openjdk.java.net/browse/JDK-8202302>



Future



Project Amber

Workshop

[OpenJDK FAQ](#)
[Installing](#)
[Contributing](#)
[Sponsoring](#)
[Developers' Guide](#)

[Mailing lists](#)

[IRC](#) · [Wiki](#)

[Bylaws](#) · [Census](#)

[Legal](#)

JEP Process**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](#)

~ ...

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 325](#) Switch Expressions (preview, JDK 12)
- [JEP 326](#) Raw String Literals (preview, JDK 12)
- [JEP draft 8209434](#) Concise Method Bodies

Delivered:

- [JEP 286](#) Local-Variable Type Inference (var) (*JDK 10*)
See also:
 - [Style Guidelines](#)
 - [Frequently Asked Questions](#)
- [JEP 323](#) Local-Variable Syntax for Lambda Parameters (*JDK 11*)

On hold:

- [JEP 301](#) Enhanced Enums. See [explanation](#).

Documents

- [Data Classes for Java](#) (*February 2018*)
- [Style Guidelines for Local Variable Type Inference](#) (*March 2018*)
- [Local Variable Type Inference: Frequently Asked Questions](#) (*October 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

// -> is "single expression form"

```
int length(String s) -> s.length();
```

// = is "method reference form"

```
int length(String s) = String::length;
```

JEP draft: Concise Method Bodies



Project Loom

<https://pixabay.com/en/craftsman-loom-craftsmanship-hands-1839920/>



Workshop

[OpenJDK FAQ](#)

[Installing](#)

[Contributing](#)

[Sponsoring](#)

[Developers' Guide](#)

[Mailing lists](#)

[IRC](#) · [Wiki](#)

[Bylaws](#) · [Census](#)

Loom

The goal of this [Project](#) is to explore and incubate Java VM features and APIs built on top of them for the implementation of lightweight user-mode threads (fibers), delimited continuations (of some form), and related features, such as explicit [tail-call](#).

This Project is sponsored by the [HotSpot Group](#).



Project Valhalla

<https://pixabay.com/en/tunnel-passage-underpass-light-965720/>



Workshop

[OpenJDK FAQ](#)

[Installing](#)

[Contributing](#)

[Sponsoring](#)

[Developers' Guide](#)

[Mailing lists](#)

[IRC](#) · [Wiki](#)

[Bylaws](#) · [Census](#)

[Legal](#)

JEP Process

Valhalla

NOTE: See the [OpenJDK Wiki](#) for details and up-to-date information.

The goal of this [Project](#) is to provide a venue to explore and incubate advanced Java VM and Language feature candidates such as:

- [Value Types](#)
- [Generic Specialization](#)
- And possibly other related topics

This Project is sponsored by the [HotSpot Group](#).

Summary

Provide JVM infrastructure for working with immutable and reference-free objects, in support of efficient by-value computation with non-primitive types.

JEP169: Value Objects

```
class Box<T> {  
}
```

```
new Box<int>( );
```

JEP218: Generics over Primitive Types

Summary

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

JEP3: 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.

JEP12: Preview Language and VM Features

Summary

Enhance the G1 garbage collector to automatically return Java heap memory to the operating system when idle.

Non-Goals

- Sharing of committed but empty pages between Java processes. Memory should be returned (uncommitted) to the operating system.
- The process of giving back memory does not need to be frugal with CPU resources, nor does it need to be instantaneous.
- Use of different methods to return memory other than available uncommitted memory.
- Support for other collectors than G1.

JEP 346: Promptly Return Unused Committed Memory from G1

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

```
private final static Logger LOGGER =  
    Logger.getLogger( "com.foo.Bar" );
```

JEP draft: Lazy Static Final Fields

Thanks! Questions?



<https://www.innoq.com/de/talks/2018/11/jvm-con-2018-java10-11-12/>

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