

APACHECON Europe, Okt 23<sup>th</sup>, 2019

# Maintaining a Java library in the light of the new Java release train

Benedikt Ritter - [britter@apache.org](mailto:britter@apache.org) -



# Today's speaker

## Benedikt Ritter

- Senior Software Engineer @ Gradle Inc.
- Apache Member
- former Apache Commons PMC
- Podcaster <https://autoweird.fm>



@BenediktRitter



# Agenda

- Java Release Policy
- Java 9 - Dawn of a new Age
- Challenges
- What how?



# JAVA RELEASE POLICY

---

How it was and how it's now

APACHECON  
EUROPE Oct. 22<sup>nd</sup> - 24<sup>th</sup>

2019

# Java Release History

Version	Release Date	End of Free Public Updates
JDK 1.0	January 1996	?
JDK 1.1	February 1997	?
J2SE 1.2	December 1998	?
J2SE 1.3	May 2000	?
J2SE 1.4	February 2002	October 2008
J2SE 5.0	September 2004	November 2009
Java SE 6	December 2006	April 2013
Java SE 7	July 11	April 2015
Java SE 8 (LTS)	March 2014	January 2019

Source: [https://en.wikipedia.org/wiki/Java\\_version\\_history](https://en.wikipedia.org/wiki/Java_version_history)

# Java Release History

Version	Release Date	End of Free Public Updates
Java SE 9	September 2017	March 2018
Java SE 10	March 2018	September 2018
Java SE 11 (LTS)	September 2018	September 2022
Java SE 12	March 2019	September 2019
Java SE 13	September 2019	March 2020
Java SE 14	March 2020	September 2020
Java SE 15	September 2020	March 2021
Java SE 16	March 2021	September 2021
Java SE 17 (LTS)	September 2021	TBA

Source: [https://en.wikipedia.org/wiki/Java\\_version\\_history](https://en.wikipedia.org/wiki/Java_version_history)

# Breaking Changes

- No breaking change between JDK 1.0 and Java SE 8
- New Policy:
  - Feature get marked with deprecated
  - Deprecated feature may be removed in the next major releases
- This means:
  - You are on Java SE 11 because it's LTS and you're not insane
  - `java.lang.String` is marked deprecated in Java 12
  - `java.lang.String` is removed in Java 13
  - You upgrade to the next LTS release which is Java SE 17
  - Your application blows up 💣
- And don't you dare using Java EE
  - Contributed to Eclipse Foundation
  - BUT Eclipse Foundation must not use `javax` namespace

# What happened?!

2017: Oracle   Sun





# JAVA 9 DAWN OF A NEW AGE

---

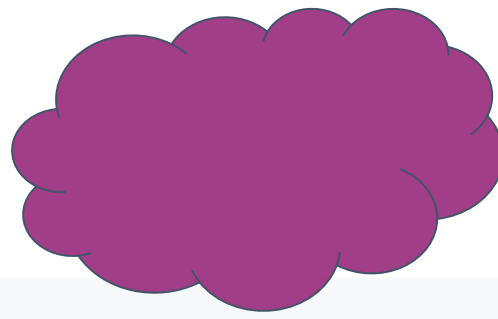
How one release broke an ecosystem

APACHECON  
EUROPE Oct. 22<sup>nd</sup> - 24<sup>th</sup>

2019

# The Idea

## Make Java Cloud ready



- Monolithic runtime is a problem
- Server applications have to ship with e.g. desktop code
- This leads to:
  - High memory requirements for Java apps
  - Considerable startup times compared to e.g. Go
- Idea: split up the runtime
- Therefore: modularize the runtime
- Therefore: Add support for software modules
- Therefore: define a way application developers can define which parts of the runtime they need

# TADA



# module-info.java

Source: <https://knowyourmeme.com/memes/the-name-is-tada>

# Slow Java 9 adoption

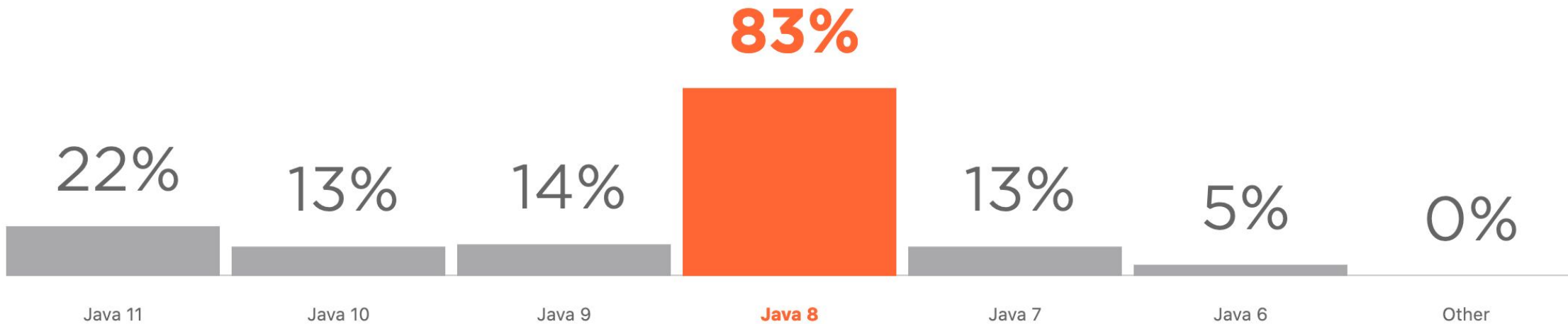
- Previous releases had nice new APIs that made life easier
  - Java 5: Generics
  - Java 8: Streaming API, java.util.time
- Feature could be adopted gradually
- With the module system it's all or nothing

*“The module system is mainly interesting for people writing books about the module system.”*

Marc Philipp, JUnit 5 Project Lead

# Java 9 adoption in numbers

Which versions of Java do you regularly use?



Although Java 10 and 11 have become more popular, Java 8 is still the most used version.

Source: <https://www.jetbrains.com/lp/devecosystem-2019/java/>



# CHALLENGES

---

Java library war stories

APACHECON  
EUROPE Oct. 22<sup>nd</sup> - 24<sup>th</sup>

2019

# Voices from the community


From: Paul Benedict <p...>


To: ...

Subject: ...

Date: ...

Location: ...

 **Jar** @jc

 **Sergei Egorov** @bsideup

[View Source](#) [Permalink](#) [Reply](#)

I'm moving to @CircleK reasons



The cause of the script --

Funny that @gradle 5 does not work on jdk13 because they need to parse bytecode for the "compiler awareness", which I guess is only used when you have api/implementation-style dependencies (not every project have moved to them yet)

isc to clear

One more time, what was wrong with "compile"? 🤔

[Tweet übersetzen](#)

 **Zheka Kozlov**  @ZhekaKozlov · 12. Okt.

Why the fuck do build systems use ASM?? Why do they need it?  
mail.openjdk.java.net/pipermail/jdk-...

vnload

Paul

# Release Train

- Staying compatible with different Java Versions
  - Used to be easy because downwards compatibility
  - Not really, see ASM
- It's hard to impossible to test against old JDKs
  - Logging requires 1.2 or higher
  - Nobody today has a working JDK 1.2
- As a core library you can't move fast and break things





# Breaking Changes

- To reiterate: This has never happened before in Java!
- You might run into the situation where:
  - Your library works up until Java 13
  - But not from Java 14 upwards
- If only internals are affected, you may be able to fix this
- If your API is affected you are basically forced to break your API as well

**Introducing a breaking change into a library like Commons Lang is a major undertaking!**

# Module System

- It's hard (impossible?) to cross compile
  - We want to run on Java 8, module-info.class is invalid on Java 8
  - We want to define module boundaries on Java 9
- Fix: use `Automatic-Module-Name` Manifest header
- One method that closed the Java 9 door for Commons Lang:

```
public void addChangeListener(final PropertyChangeListener listener) {  
    changeSupport.addPropertyChangeListener(listener);  
}
```



# WHAT NOW?

---

How to deal with this mess?

APACHECON  
EUROPE Oct. 22<sup>nd</sup> - 24<sup>th</sup>

2019

# Options

## For the library author

- Stay on Java 8
  - At the moment best for the user base
  - Eventually your library will not run anymore (breaking changes in the JDK)
  - Not keeping pace will lead to irrelevance
- Go with the flow
  - A lot of organizations are not ready for this
  - User base still seems to be on Java 8
  - Might require breaking changes
  - Getting ready for module system can be hard

a.k.a. Apache Commons Style

a.k.a. JUnit 5 Style

# I honestly don't know...





**APACHECON**

EUROPE

**APACHECON**  
EUROPE Oct. 22<sup>nd</sup> - 24<sup>th</sup>

**2019**

# LOOKING FORWARD TO YOUR COMMENTS AND THOUGHTS

Title of Presentations

---



**Location**

401 West 20nd 21212 NYC



**Phone**

+1 212 - 758 - 1234



**Email**

hello@y.com



**Website**

www.apace.org

**APACHECON**  
EUROPE Oct. 22<sup>nd</sup> - 24<sup>th</sup>

# 2019