Javalution!

How I learned to stop worrying and play Jenga! with the entire software industry

Development Team
Java Platform Group - Oracle

> “you can't make an omelet without breaking some eggs”

François de Charette (*) 1763 -1796

(*) According to: en.wiktionary.org

Introduction

Georges Saab

Vice President Software Development - Java Platform Group

- Oracle since 2008
- Previously BEA since 2003

Aurelio Garcia-Ribeyro

Director Product Management – Java Platform Group

- Oracle since 2010
- Previously Sun Microsystems since 2008

Charlie Hunt

Senior Developer – JVM Special Projects - Java Platform Group

- Oracle since 20010
- Previously Sun Microsystems since 1999

Introduction

Georges Saab

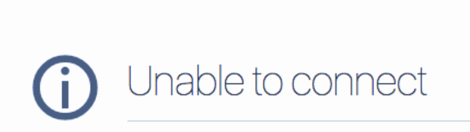
Vice President Software Development - Java Platform Group

- Oracle since 2008
- Previously BEA since 2003

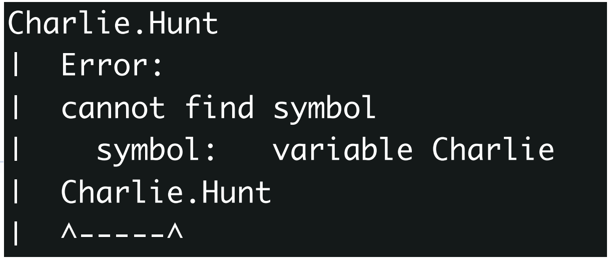
Aurelio Garcia-Ribeyro

Director Product Management – Java Platform Group

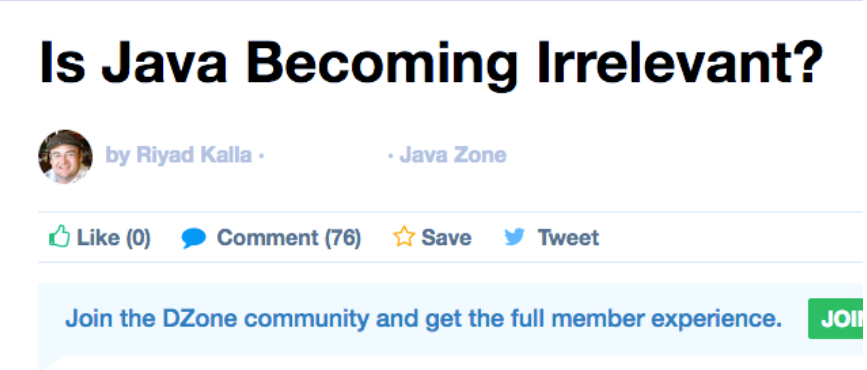
- Oracle since 2010
- Previously Sun Microsystems since 2008



Unable to connect



```
Charlie.Hunt
|  Error:
|  cannot find symbol
|    symbol:   variable Charlie
|  Charlie.Hunt
|  ^-----^
```

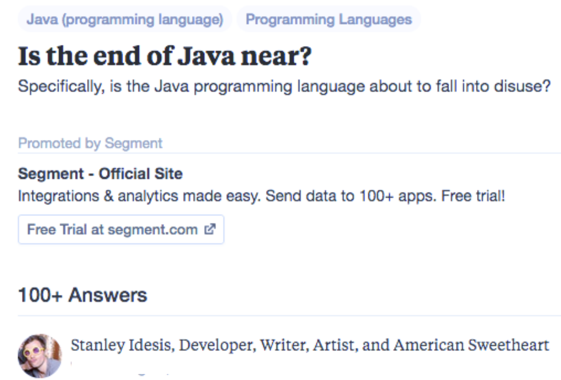


Is Java Becoming Irrelevant?

by Riyad Kalla · · Java Zone

Like (0) Comment (76) Save Tweet

Join the DZone community and get the full member experience. JOI



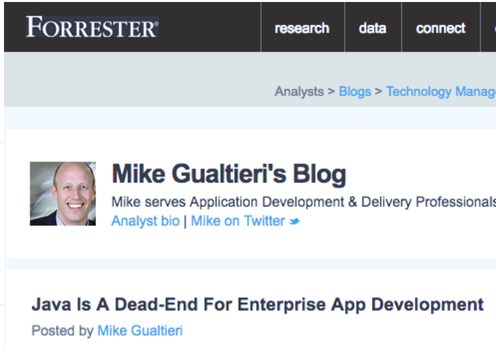
Java (programming language) Programming Languages

Is the end of Java near?

Specifically, is the Java programming language about to fall into disuse?

100+ Answers

Stanley Idesis, Developer, Writer, Artist, and American Sweetheart



FORRESTER research data connect

Analysts > Blogs > Technology Manag

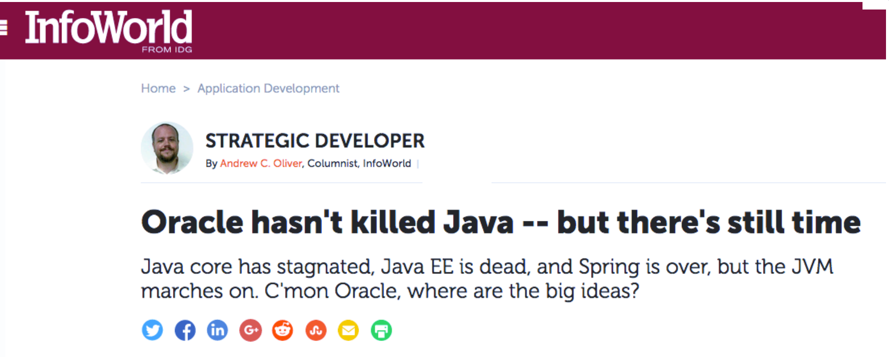
Mike Gualtieri's Blog

Mike serves Application Development & Delivery Professionals

Analyst bio | Mike on Twitter

Java Is A Dead-End For Enterprise App Development

Posted by Mike Gualtieri



InfoWorld FROM IDG

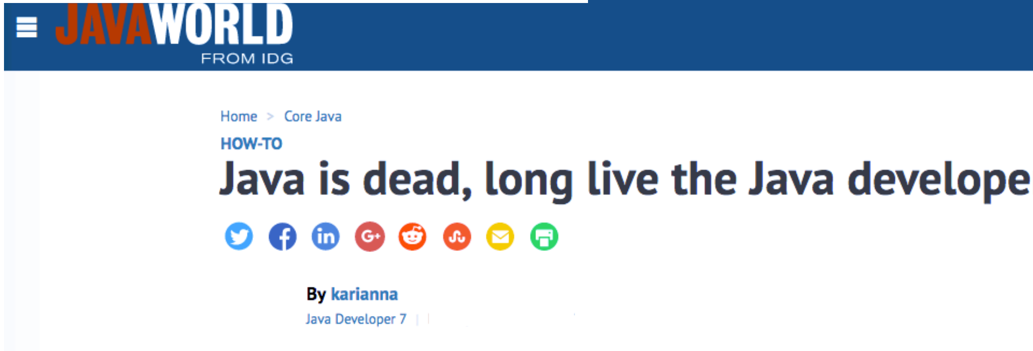
Home > Application Development

STRATEGIC DEVELOPER

By Andrew C. Oliver, Columnist, InfoWorld

Oracle hasn't killed Java -- but there's still time

Java core has stagnated, Java EE is dead, and Spring is over, but the JVM marches on. C'mon Oracle, where are the big ideas?



JAVAWORLD FROM IDG

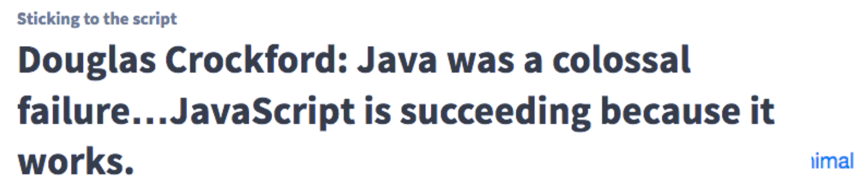
Home > Core Java

HOW-TO

Java is dead, long live the Java develope

By karianna

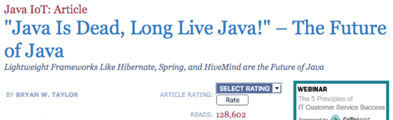
Java Developer 7



Sticking to the script

Douglas Crockford: Java was a colossal failure...JavaScript is succeeding because it works.

iimal



Java IoT: Article

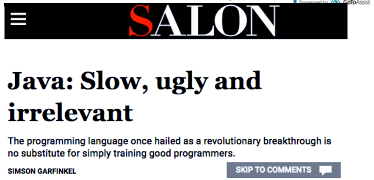
"Java Is Dead, Long Live Java!" – The Future of Java

Lightweight Frameworks Like Hibernate, Spring, and HiveMind are the Future of Java

BY BRYAN W. TAYLOR

ARTICLE RATING: SELECT RATING Rate

READS: 128,602



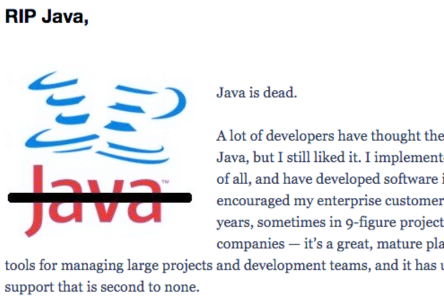
SALON

Java: Slow, ugly and irrelevant

The programming language once hailed as a revolutionary breakthrough is no substitute for simply training good programmers.

SIMSON GARFINKEL

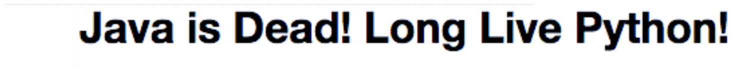
SKIP TO COMMENTS



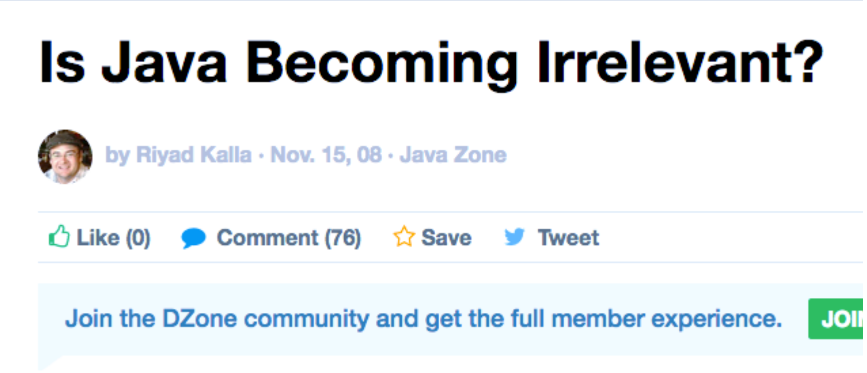
RIP Java,

Java is dead.

A lot of developers have thought the
Java, but I still liked it. I implement
of all, and have developed software i
encouraged my enterprise customer
years, sometimes in 9-figure project
companies — it's a great, mature pla
tools for managing large projects and development teams, and it has u
support that is second to none.



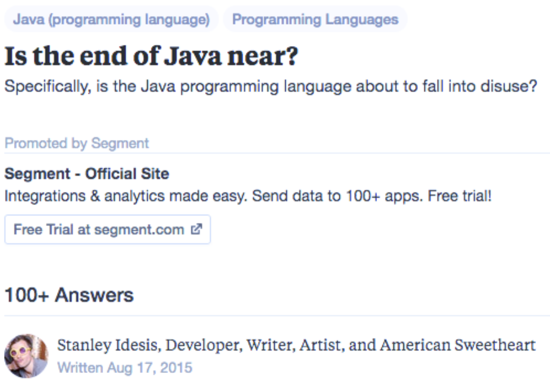
Java is Dead! Long Live Python!



Is Java Becoming Irrelevant?

by Riyad Kalla · Nov. 15, 08 · Java Zone

Like (0)
Comment (76)
Save
Tweet
Join the DZone community and get the full member experience.
JOI

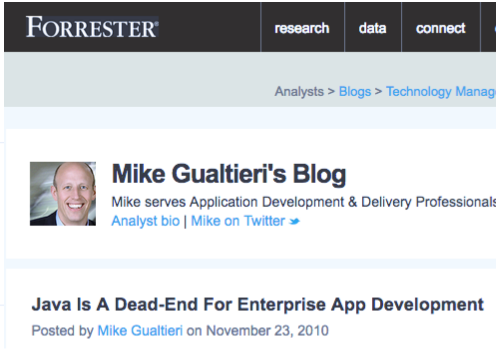


Is the end of Java near?

Specifically, is the Java programming language about to fall into disuse?

100+ Answers

Stanley Idesis, Developer, Writer, Artist, and American Sweetheart
Written Aug 17, 2015

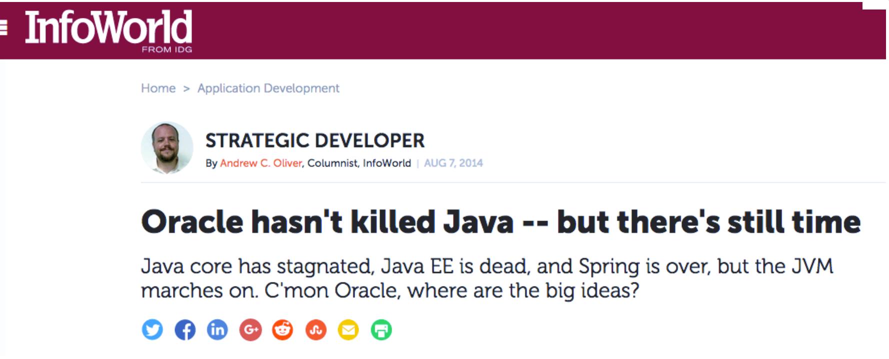


Mike Gualtieri's Blog

Mike serves Application Development & Delivery Professional
Analyst bio | Mike on Twitter

Java Is A Dead-End For Enterprise App Development

Posted by Mike Gualtieri on November 23, 2010

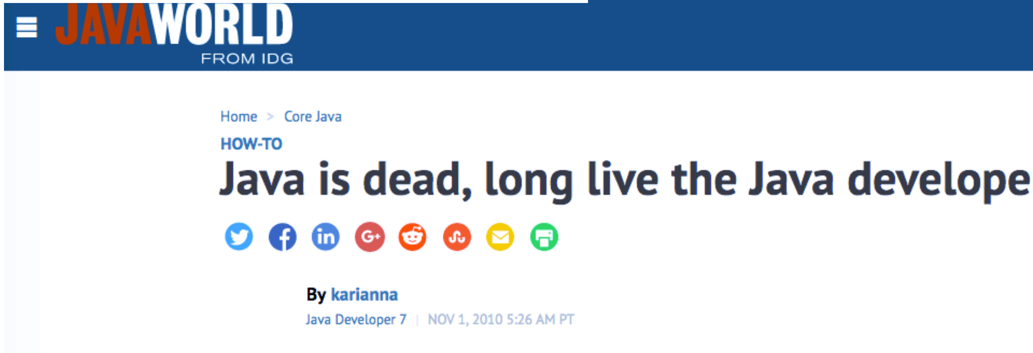


STRATEGIC DEVELOPER

By Andrew C. Oliver, Columnist, InfoWorld | AUG 7, 2014

Oracle hasn't killed Java -- but there's still time

Java core has stagnated, Java EE is dead, and Spring is over, but the JVM marches on. C'mon Oracle, where are the big ideas?

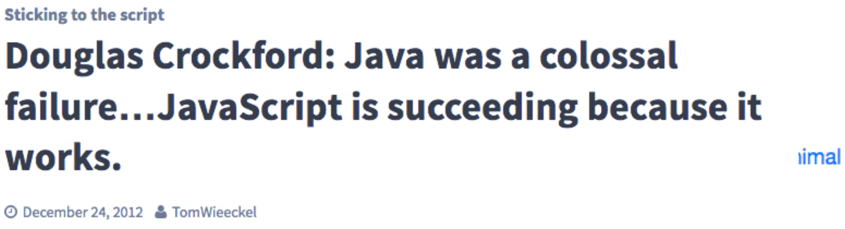


HOW-TO

Java is dead, long live the Java develope

By karianna

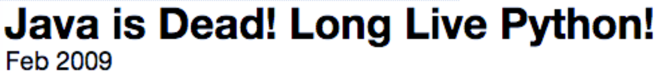
Java Developer 7 | NOV 1, 2010 5:26 AM PT



Sticking to the script

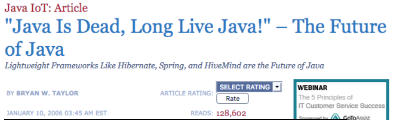
Douglas Crockford: Java was a colossal failure…JavaScript is succeeding because it works.

December 24, 2012 TomWieeckel



Java is Dead! Long Live Python!

Feb 2009



Java IoT: Article

"Java Is Dead, Long Live Java!" – The Future of Java

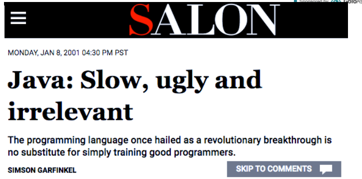
Lightweight Frameworks Like Hibernate, Spring, and HiveMind are the Future of Java

BY BRYAN W. TAYLOR

JANUARY 10, 2006 03:45 AM EST

ARTICLE RATING: SELECT RATING Rate

READS: 128,602

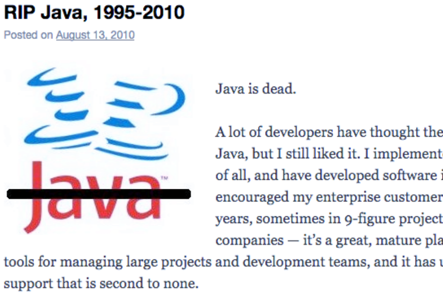


MONDAY, JAN 8, 2001 04:30 PM PST

Java: Slow, ugly and irrelevant

The programming language once hailed as a revolutionary breakthrough is no substitute for simply training good programmers.

SIMSON GARFINKEL



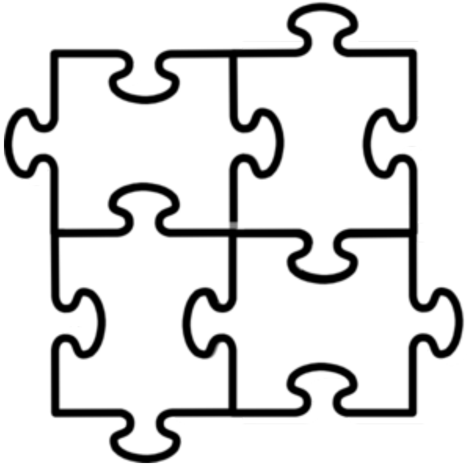
RIP Java, 1995-2010

Posted on August 13, 2010

Java is dead.

A lot of developers have thought the
Java, but I still liked it. I implement
of all, and have developed software i
encouraged my enterprise customer
years, sometimes in 9-figure project
companies — it's a great, mature pla
tools for managing large projects and development teams, and it has u
support that is second to none.

The biggest change in JDK 9: Jigsaw



The good

- Smaller JDK API surface area
- Allows for optimized, smaller runtime
- Decreases on-boarding time for JDK developers
- Avoids class resolving ambiguity
- Allows a clear separation of what is intended to be used by a developer vs internal representation

But…

- Avoids class resolving ambiguity
- Allows a clear separation of what is intended to be used by a developer vs internal representation… and the JDK uses this to hide internal APIs)

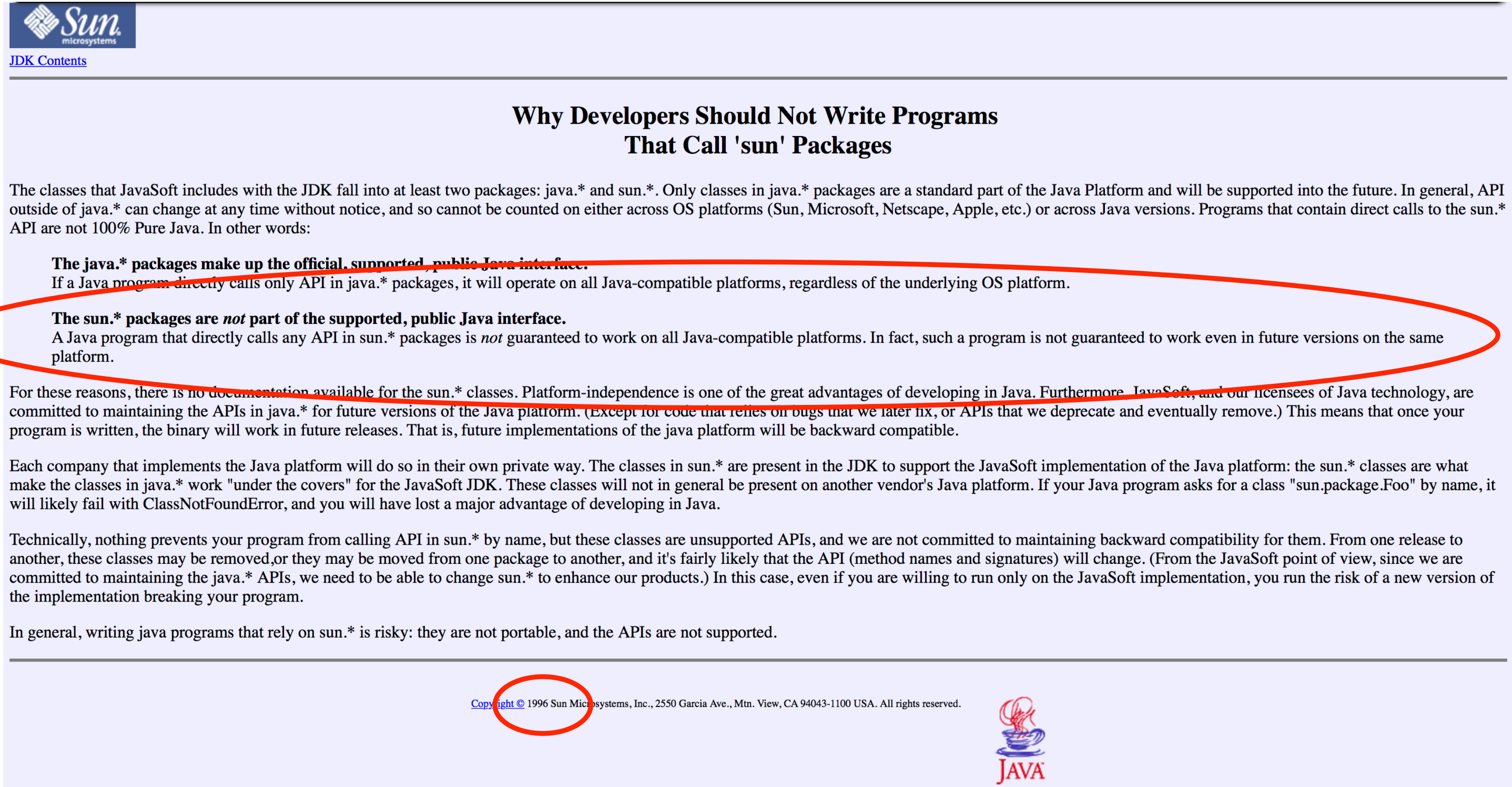
Background: Categories of APIs in the JDK

Supported, intended for external use

- JCP standard, java.*, javax.*
- JDK-specific API, some com.sun.*, some jdk.*

Unsupported, JDK-internal, not intended for external use

- sun.* mostly



JDK Contents

Why Developers Should Not Write Programs That Call 'sun' Packages

The classes that JavaSoft includes with the JDK fall into at least two packages: java.* and sun.*. Only classes in java.* packages are a standard part of the Java Platform and will be supported into the future. In general, API outside of java.* can change at any time without notice, and so cannot be counted on either across OS platforms (Sun, Microsoft, Netscape, Apple, etc.) or across Java versions. Programs that contain direct calls to the sun.* API are not 100% Pure Java. In other words:

> **The java.* packages make up the official, supported, public Java interface.**
> If a Java program directly calls only API in java.* packages, it will operate on all Java-compatible platforms, regardless of the underlying OS platform.
>
> **The sun.* packages are *not* part of the supported, public Java interface.**
> A Java program that directly calls any API in sun.* packages is *not* guaranteed to work on all Java-compatible platforms. In fact, such a program is not guaranteed to work even in future versions on the same platform.

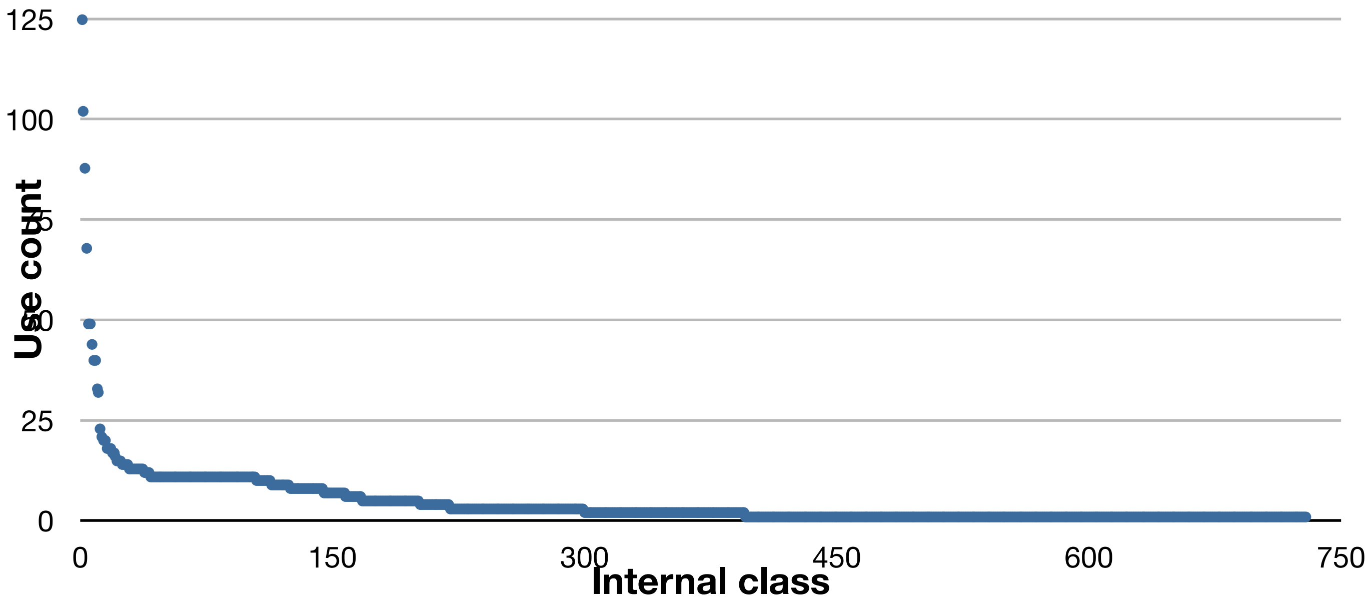
For these reasons, there is no documentation available for the sun.* classes. Platform-independence is one of the great advantages of developing in Java. Furthermore, JavaSoft, and our licensees of Java technology, are committed to maintaining the APIs in java.* for future versions of the Java platform. (Except for code that relies on bugs that we later fix, or APIs that we deprecate and eventually remove.) This means that once your program is written, the binary will work in future releases. That is, future implementations of the java platform will be backward compatible.

Each company that implements the Java platform will do so in their own private way. The classes in sun.* are present in the JDK to support the JavaSoft implementation of the Java platform: the sun.* classes are what make the classes in java.* work "under the covers" for the JavaSoft JDK. These classes will not in general be present on another vendor's Java platform. If your Java program asks for a class "sun.package.Foo" by name, it will likely fail with ClassNotFoundError, and you will have lost a major advantage of developing in Java.

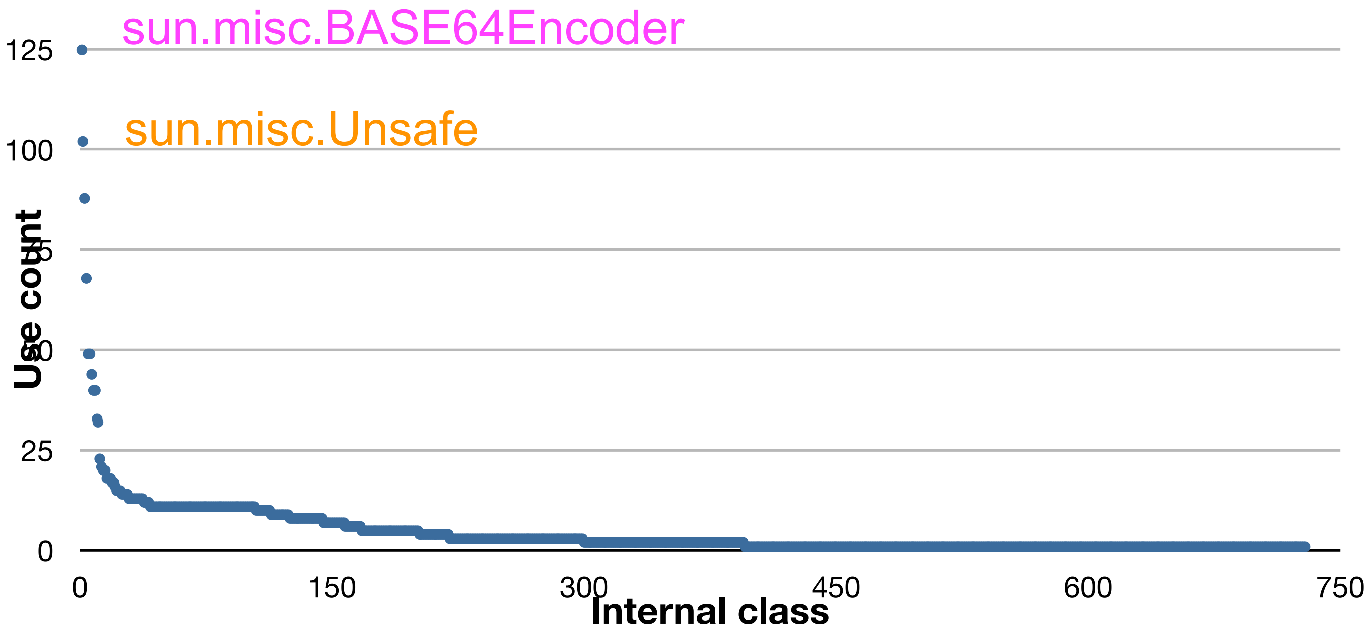
Technically, nothing prevents your program from calling API in sun.* by name, but these classes are unsupported APIs, and we are not committed to maintaining backward compatibility for them. From one release to another, these classes may be removed,or they may be moved from one package to another, and it's fairly likely that the API (method names and signatures) will change. (From the JavaSoft point of view, since we are committed to maintaining the java.* APIs, we need to be able to change sun.* to enhance our products.) In this case, even if you are willing to run only on the JavaSoft implementation, you run the risk of a new version of the implementation breaking your program.

In general, writing java programs that rely on sun.* is risky: they are not portable, and the APIs are not supported.

Uses of JDK-internal APIs



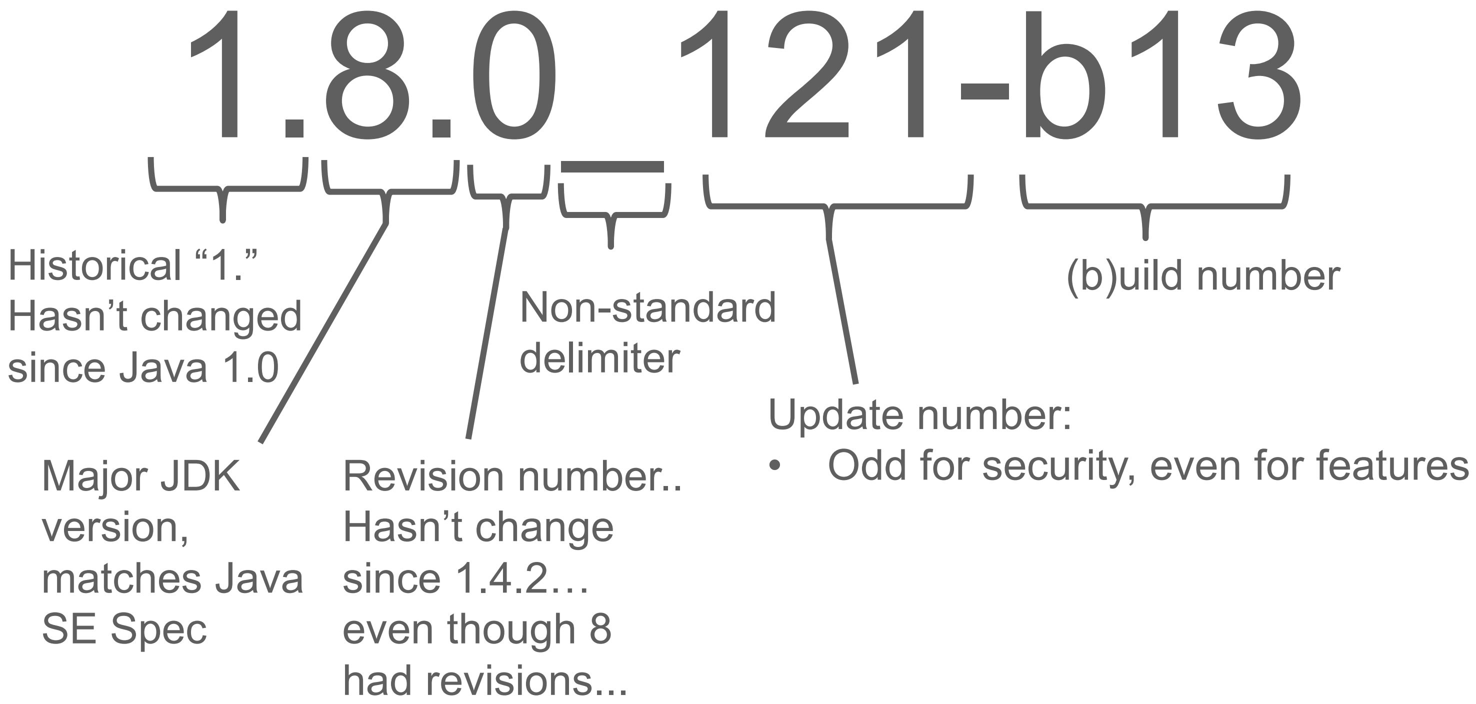
Uses of JDK-internal APIs



New Version String: Current Format

1.8.0_121-b13

New Version String: Current Format



1.8.0_121-b13

Historical “1.”
Hasn’t changed
since Java 1.0

Major JDK
version,
matches Java
SE Spec

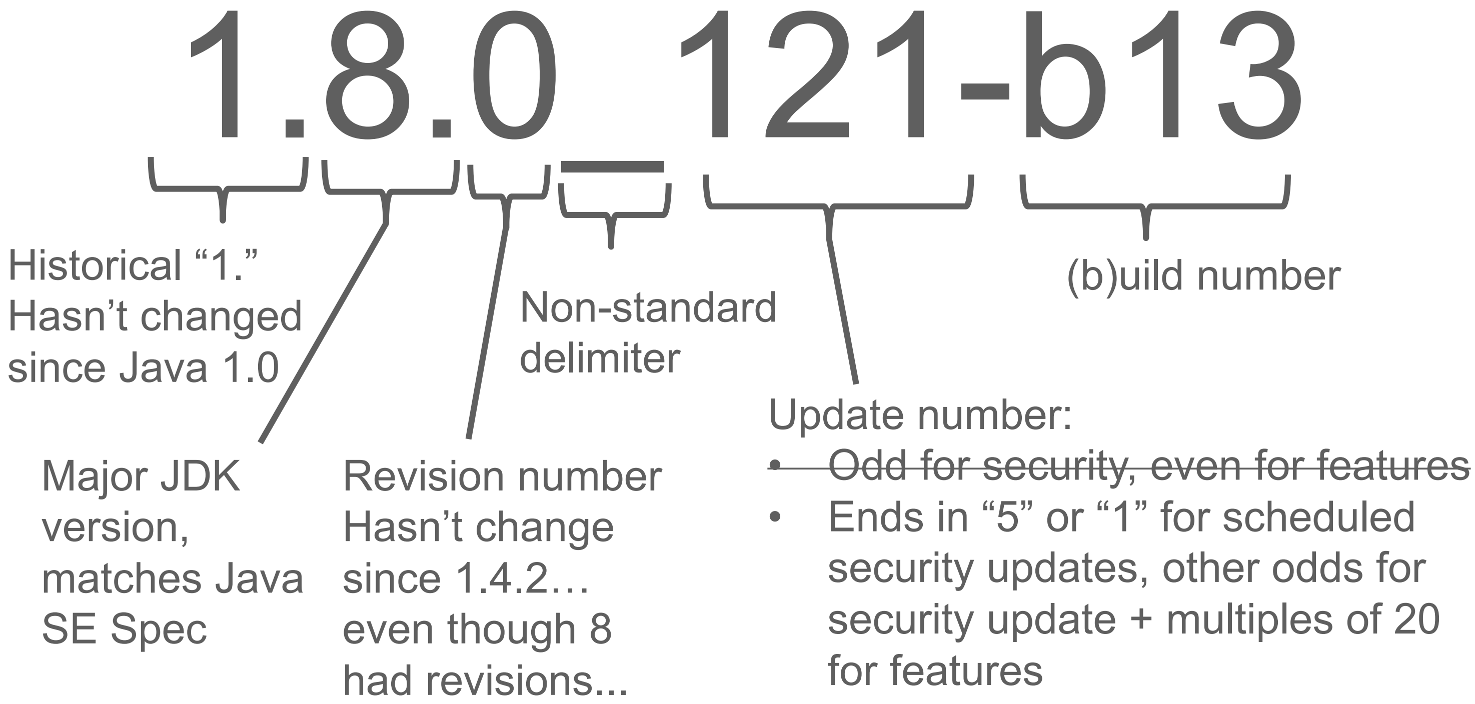
Revision number..
Hasn’t change
since 1.4.2…
even though 8
had revisions...

Non-standard
delimiter

Update number:
- Odd for security, even for features

(b)uild number

New Version String: Current Format



1.8.0_121-b13

Historical “1.”
Hasn’t changed
since Java 1.0

Major JDK
version,
matches Java
SE Spec

Revision number
Hasn’t change
since 1.4.2…
even though 8
had revisions...

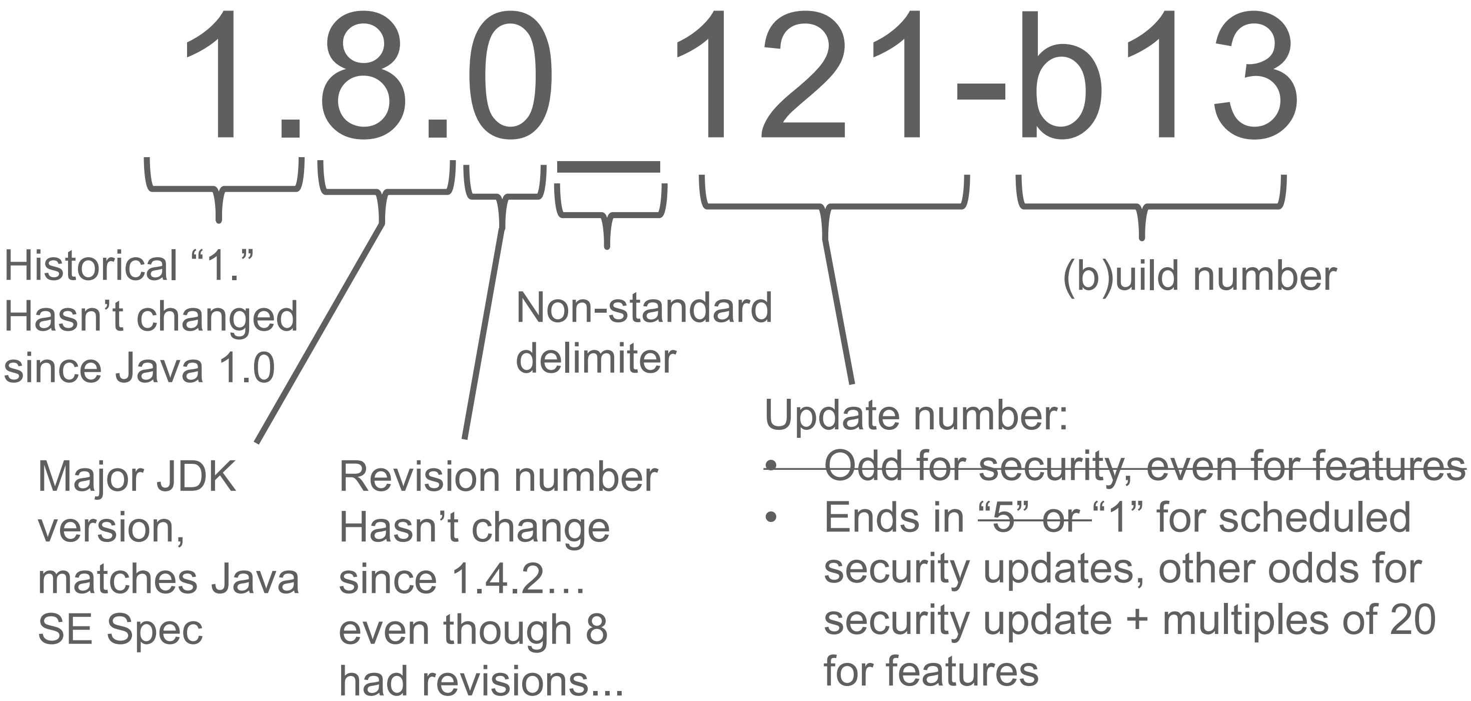
Non-standard
delimiter

Update number:

- ~~Odd for security, even for features~~
- Ends in “5” or “1” for scheduled security updates, other odds for security update + multiples of 20 for features

(b)uild number

New Version String: Current Format



1.8.0_121-b13

Historical “1.”
Hasn’t changed
since Java 1.0

Major JDK
version,
matches Java
SE Spec

Revision number
Hasn’t change
since 1.4.2…
even though 8
had revisions...

Non-standard
delimiter

Update number:

- ~~Odd for security, even for features~~
- Ends in ~~“5” or~~ “1” for scheduled security updates, other odds for security update + multiples of 20 for features

(b)uild number

JEP 223: New Version-String Scheme

GA version: 9

Security Update: 9.0.3, 9.0.6 (still gaps to cover unplanned security releases)

Feature update: 9.1.6 (With same level of security as 9.0.6)

Align with current industry practices, in particular Semantic Versioning
Provide a simple API for version-string parsing, validation, and comparison

Subtler changes

- Will no longer ship
  - the `jhat` Tool – experimental never supported
  - VisualVM – remains available in github
  - JavaDB – remains available as apache derby
- Other changes
  - G1 is the new default garbage collector
  - New unified JVM logging will be used by GC
  - JNLP parsing got strict

Gone, gone, gone!

[Some] Deprecated and old things are finally being removed

- Several GC combinations that were deprecated in JDK 8
- The Entire HTTP Proxying Mechanism of RMI
- Thread.stop(Throwable)
- Serialized Applets
- Unsupported Apple APIs
- Old warnings become new errors
    - E.g. Permgen settings

Important Incompatibility Expected in JDK 9

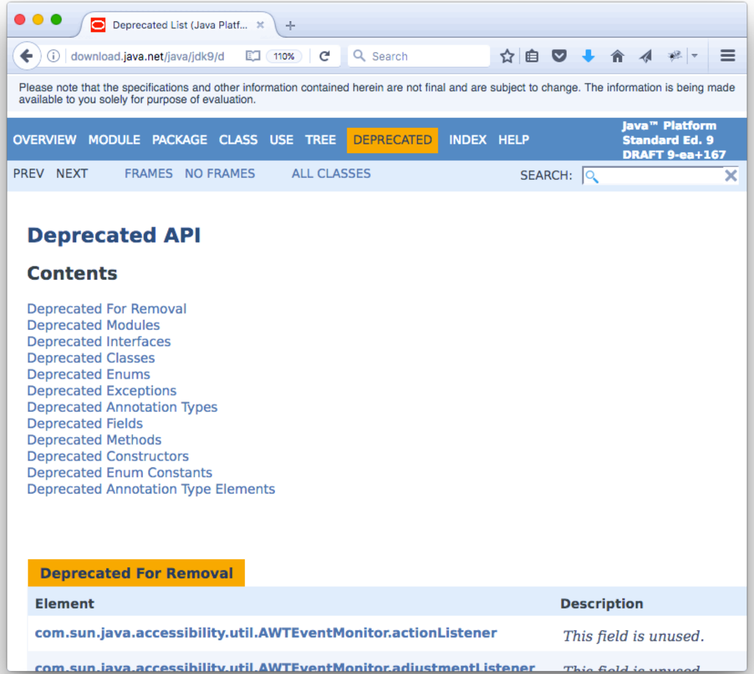
A third-party library or tool that your code relies on is affected by the changes to JDK 9.

- As part of the JDK 9 work, we are contacting developers of many common frameworks and common libraries that we have identified as affected.
- Expect most vendors with large installed base to provide updates before JDK 9 releases.
- However, until your code adopts the new versions, it is likely to be affected.

Deprecated in JDK 9

But Still There, At Least Until JDK 10

- Applets as a deployment mechanism
  - Applet viewer
  - Common DOM APIs
  - classid support
- VP6 video codec and FLV/FXM file format for JavaFX Media
- jarjar
- CMS Garbage Collector
- The `com.sun.jarsigner` package
- Java Policy Tool
- Several security APIs that are no longer current
- `jconsole`
- Doclet API


Deprecated List (Java Platf...
download.java.net/java/jdk9/d
110%
Search
OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
Java™ Platform
Standard Ed. 9
DRAFT 9-ea+167
PREV NEXT
FRAMES NO FRAMES
ALL CLASSES
SEARCH:

Please note that the specifications and other information contained herein are not final and are subject to change. The information is being made available to you solely for purpose of evaluation.

Deprecated API

Contents

- Deprecated For Removal
- Deprecated Modules
- Deprecated Interfaces
- Deprecated Classes
- Deprecated Enums
- Deprecated Exceptions
- Deprecated Annotation Types
- Deprecated Fields
- Deprecated Methods
- Deprecated Constructors
- Deprecated Enum Constants
- Deprecated Annotation Type Elements

Deprecated For Removal

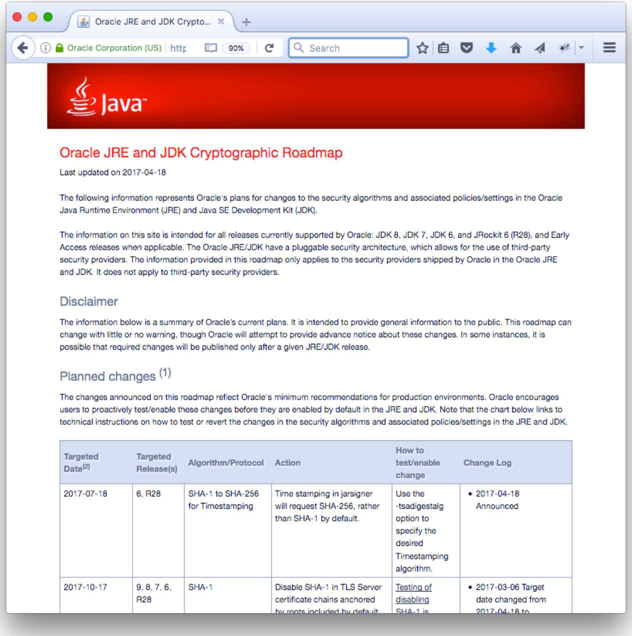
Element	Description
com.sun.java.accessibility.util.AWTEventMonitor.actionListener	*This field is unused.*
com.sun.java.accessibility.util.AWTEventMonitor.adjustmentListener	*This field is unused.*

http://download.java.net/java/jdk9/docs/api/deprecated-list.html

And expect JDK 9 to keep up to date with security

Like JDK 6, 7, and 8 now

- Default crypto values updated to increase security
- New algorithms and support for stronger keys, added
- No-longer-secure algorithms disabled-by-default
- When possible
  - Advanced warning on Java Crypto Roadmap
  - Instructions for testing before changes GA
  - Instructions for reverting changes after GA (but don't do this! Bad, no-good, idea!)



Oracle JRE and JDK Cryptographic Roadmap

Last updated on 2017-04-18

The following information represents Oracle's plans for changes to the security algorithms and associated policies/settings in the Oracle Java Runtime Environment (JRE) and Java SE Development Kit (JDK).

The information on this site is intended for all releases currently supported by Oracle: JDK 8, JDK 7, JDK 6, and JRockit 6 (R28), and Early Access releases when applicable. The Oracle JRE/JDK have a pluggable security architecture, which allows for the use of third-party security providers. The information provided in this roadmap only applies to the security providers shipped by Oracle in the Oracle JRE and JDK. It does not apply to third-party security providers.

Disclaimer

The information below is a summary of Oracle's current plans. It is intended to provide general information to the public. This roadmap can change with little or no warning, though Oracle will attempt to provide advance notice about these changes. In some instances, it is possible that required changes will be published only after a given JRE/JDK release.

Planned changes (1)

The changes announced on this roadmap reflect Oracle's minimum recommendations for production environments. Oracle encourages users to proactively test/enable these changes before they are enabled by default in the JRE and JDK. Note that the chart below links to technical instructions on how to test or revert the changes in the security algorithms and associated policies/settings in the JRE and JDK.

Targeted Date[2]	Targeted Release(s)	Algorithm/Protocol	Action	How to test/enable change	Change Log
2017-07-18	6, R28	SHA-1 to SHA-256 for Timestamping	Time stamping in jarsigner will request SHA-256, rather than SHA-1 by default.	Use the -tsadigestalg option to specify the desired Timestamping algorithm.	• 2017-04-18 Announced
2017-10-17	9, 8, 7, 6, R28	SHA-1	Disable SHA-1 in TLS Server certificate chains anchored	Testing of disabling	• 2017-03-06 Target date changed from

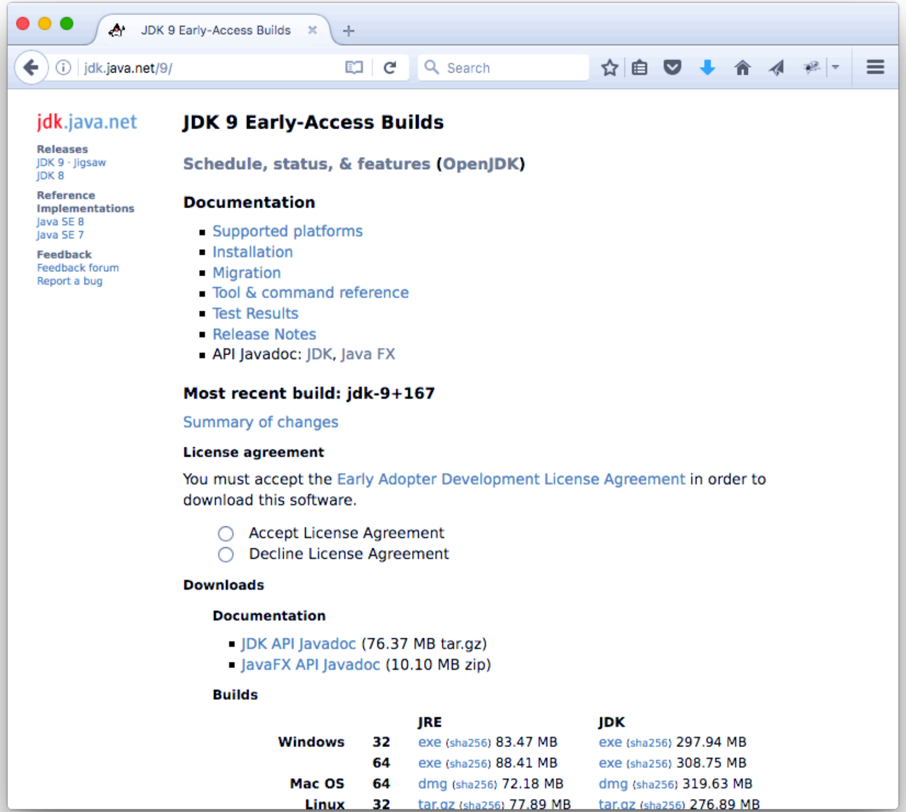
java.com/cryptoroadmap

Prepare for the Future Now!

1) Download JDK 9 EA and JDK 9-Enabled IDE (for example, NetBeans Dev builds).
2) Identify code and dependencies (libraries) that have problematic dependencies:
   - Using `jdeps`
   - Testing
3) Update your own code and eliminate dependencies to removed, deprecated, or hidden/internal APIs.
   - As a stopgap measure, use command-line options to export hidden APIs.
4) Find alternative libraries that work with JDK 9.

Download JDK 9 EA

- Early access builds of JDK 9 available for testing
- Periodic updates, so check frequently for newer builds.
  - See “Summary of changes” on each build.



jdk.java.net

Releases
JDK 9 · Jigsaw
JDK 8

Reference Implementations
Java SE 8
Java SE 7

Feedback
Feedback forum
Report a bug

JDK 9 Early-Access Builds

Schedule, status, & features (OpenJDK)

Documentation

- Supported platforms
- Installation
- Migration
- Tool & command reference
- Test Results
- Release Notes
- API Javadoc: JDK, Java FX

Most recent build: jdk-9+167

Summary of changes

License agreement

You must accept the Early Adopter Development License Agreement in order to download this software.

○ Accept License Agreement
○ Decline License Agreement

Downloads

Documentation

- JDK API Javadoc (76.37 MB tar.gz)
- JavaFX API Javadoc (10.10 MB zip)

Builds

		JRE	JDK
Windows	**32**	exe (sha256) 83.47 MB	exe (sha256) 297.94 MB
	64	exe (sha256) 88.41 MB	exe (sha256) 308.75 MB
Mac OS	**64**	dmg (sha256) 72.18 MB	dmg (sha256) 319.63 MB
Linux	**32**	tar.gz (sha256) 77.89 MB	tar.gz (sha256) 276.89 MB

http://jdk.java.net/9/

Identify Problematic Dependencies

Use Java Dependency Analysis Tool (`jdeps`)

- Available since JDK 8
- Best results from the version in JDK 9 EA
- Option to find internal dependencies

```
tzupdater-2.0.3-2015b $ jdeps tzupdater.jar
tzupdater.jar -> java.base
   com.sun.tools.tzupdater                -> com.sun.tools.tzupdater.utils       tzupdater.jar
(...)
   com.sun.tools.tzupdater                -> java.util.regex                     java.base
   com.sun.tools.tzupdater                -> java.util.zip                       java.base
   com.sun.tools.tzupdater                -> sun.util.calendar                   JDK internal API (java.base)
   com.sun.tools.tzupdater                -> tools.javazic                       tzupdater.jar
(...)
   com.sun.tools.tzupdater.utils          -> java.util                           java.base
   com.sun.tools.tzupdater.utils          -> sun.util.calendar                   JDK internal API (java.base)
   tools.javazic                          -> java.io                             java.base
(...)
```

https://wiki.openjdk.java.net/display/JDK8/Java+Dependency+Analysis+Tool

Stopgap: Expose Internal APIs

But come back and fix!

Sample command for earlier dev version of Netbeans

```
$ bin/netbeans --jdkhome ~/jdk9ea --add-exports java.desktop/sun.awt=ALL-UNNAMED --
add-exports java.base/jdk.internal.jrtfs=ALL-UNNAMED --add-exports
java.desktop/java.awt.peer=ALL-UNNAMED --add-exports
java.desktop/com.sun.beans.editors=ALL-UNNAMED --add-exports
java.desktop/sun.awt.im=ALL-UNNAMED  --add-exports
java.desktop/com.sun.java.swing.plaf.gtk=ALL-UNNAMED --add-exports
java.management/sun.management=ALL-UNNAMED
```

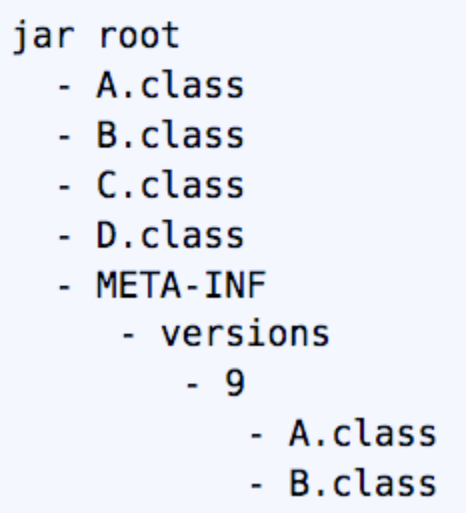
Tools for new paradigm

JEP 238: Multi-Release JAR Files

tools / jar

Extend the JAR file format to allow multiple, Java-release-specific versions of class files to coexist in a single archive

Write JDK-version-specific variants of the same code into a single jar file



```
jar root
  - A.class
  - B.class
  - C.class
  - D.class
  - META-INF
     - versions
        - 9
           - A.class
           - B.class
```

JEP 295: Ahead-of-Time Compilation

hotspot / compiler

Compile Java classes to native code prior to launching the virtual machine.

Improve the start-up time of both small and large Java applications, with at most a limited impact on peak performance.

AOT compilation is done by a new tool, jaotc

For the initial release, the only supported module is java.base.

AOT compilation of any other JDK module, or of user code, is experimental

Pro tip: Use jaotc on the production machine, not in the development one

From "One JRE to rule them all" to customized runtimes

jlink: a great tool for ~~embedded devices~~ cloud deployments

Then

- Minimize download size
- Java programs should share a system JRE
- 300 Mb is too large!
- Serious applications require one large app container

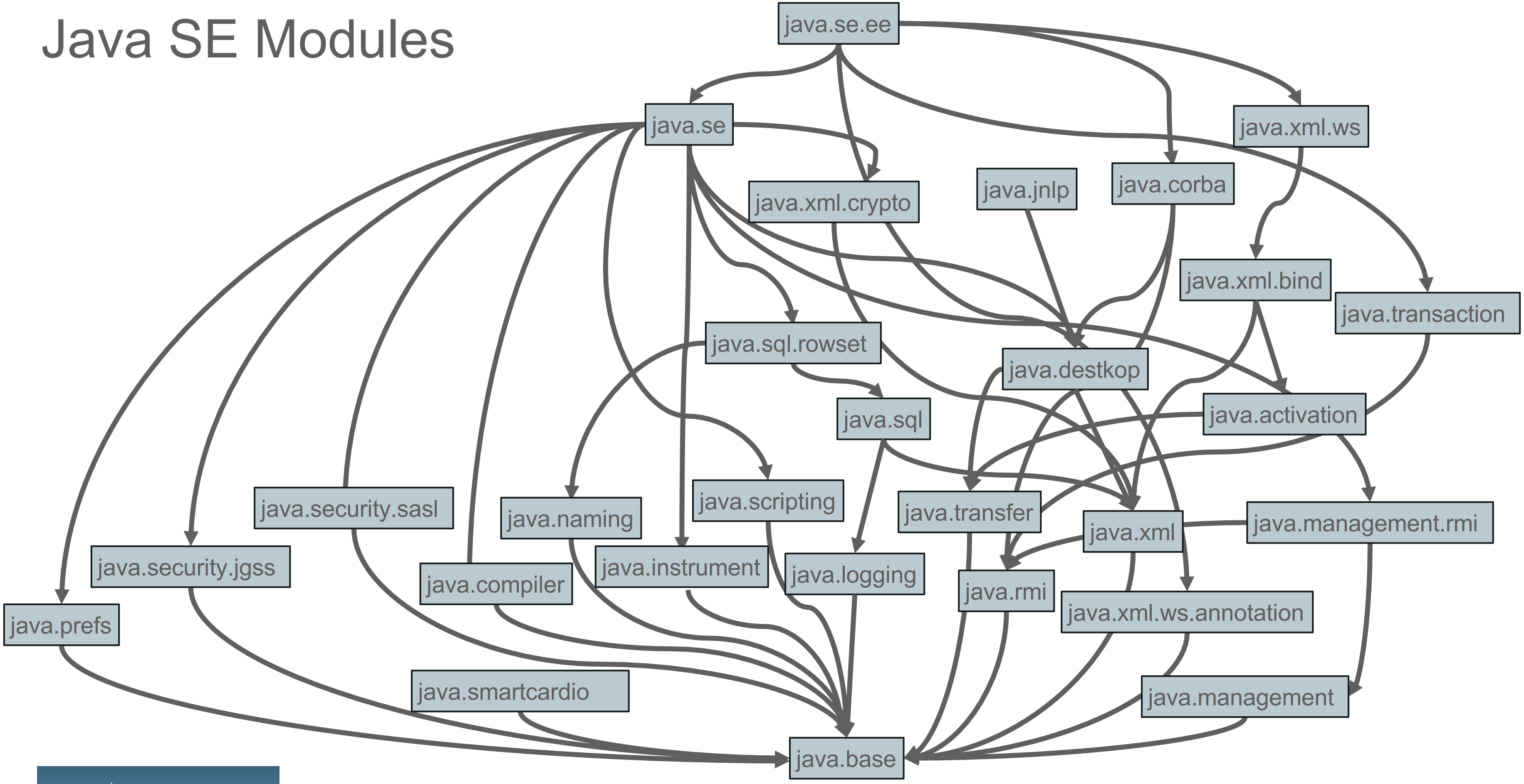
Now

- Minimize interdependencies
- Each application has its own java runtime (and container)
- I have games over 600 Mb and over 20 Gb of media in my cell phone
- Serious systems require hundreds or thousands of instance of self-contained components and services that spawn and die as they will

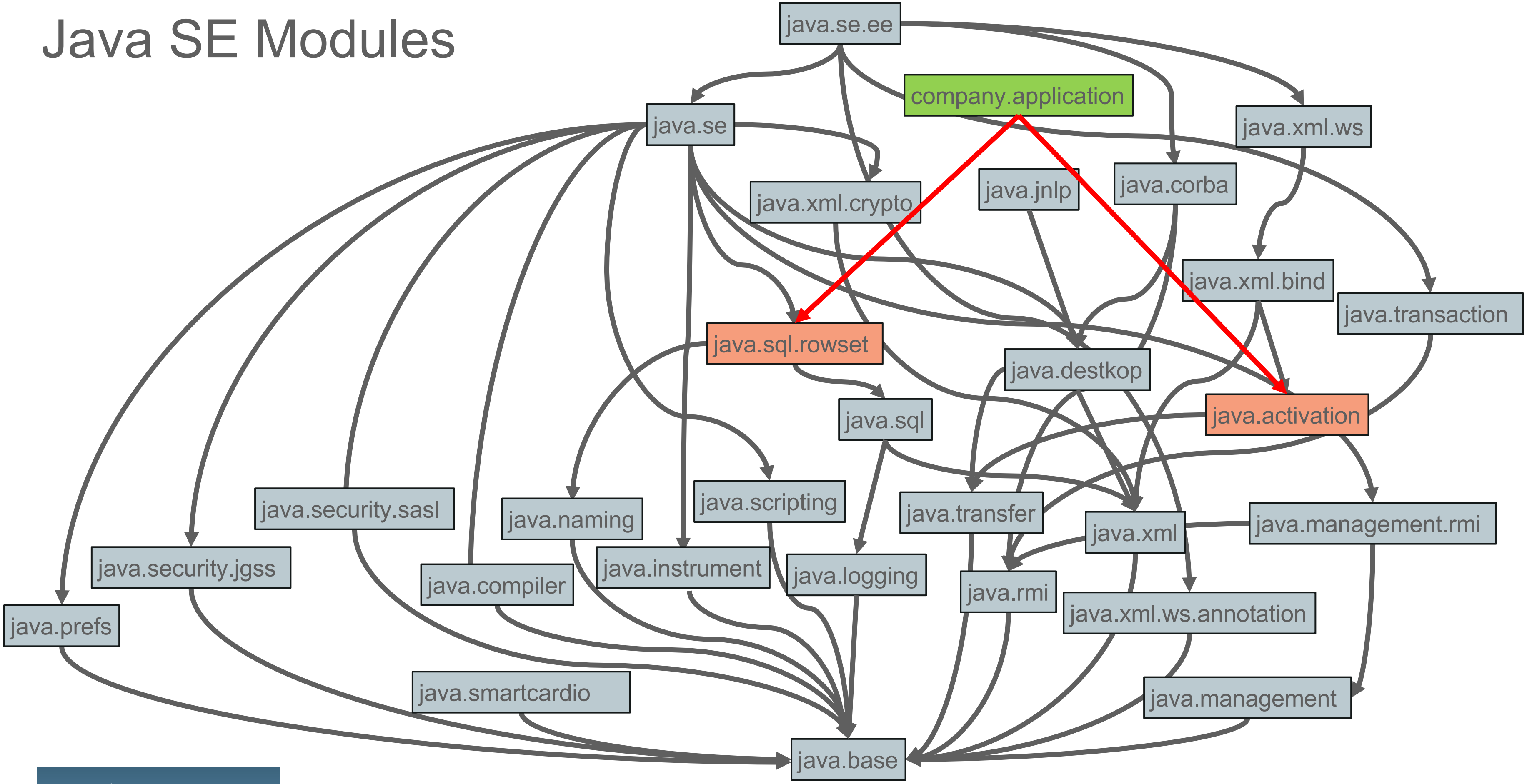
JDK 9 Modules

java.activation
java.base
java.compiler
java.corba
java.datatransfer
java.desktop
java.instrument
java.jnlp
java.logging
java.management
java.management.rmi
java.naming
java.prefs
java.rmi
java.scripting
java.se
java.se.ee
java.security.jgss
java.security.sasl
java.smartcardio
java.sql
java.sql.rowset
java.transaction
java.xml
java.xml.bind
java.xml.crypto
java.xml.ws
java.xml.ws.annotation
javafx.base
javafx.controls
javafx.fxml
javafx.graphics
javafx.media
javafx.swing
javafx.web
jdk.accessibility
jdk.attach
jdk.charsets
jdk.compiler
jdk.crypto.cryptoki
jdk.crypto.ec
jdk.dynalink
jdk.editpad
jdk.httpserver
jdk.jartool
jdk.javadoc
jdk.jcmd
jdk.jconsole
jdk.jdeps
jdk.jdi
jdk.jdwp.agent
jdk.jlink
jdk.jshell
jdk.jsobject
jdk.jstatd
jdk.localedata
jdk.management
jdk.management.agent
jdk.naming.dns
jdk.naming.rmi
jdk.net
jdk.pack
jdk.packager
jdk.packager.services
jdk.policytool
jdk.rmic
jdk.scripting.nashorn
jdk.sctp
jdk.security.auth
jdk.security.jgss
jdk.snmp
jdk.xml.dom
jdk.zipfs
jdk.incubator.httpclient

Java SE Modules

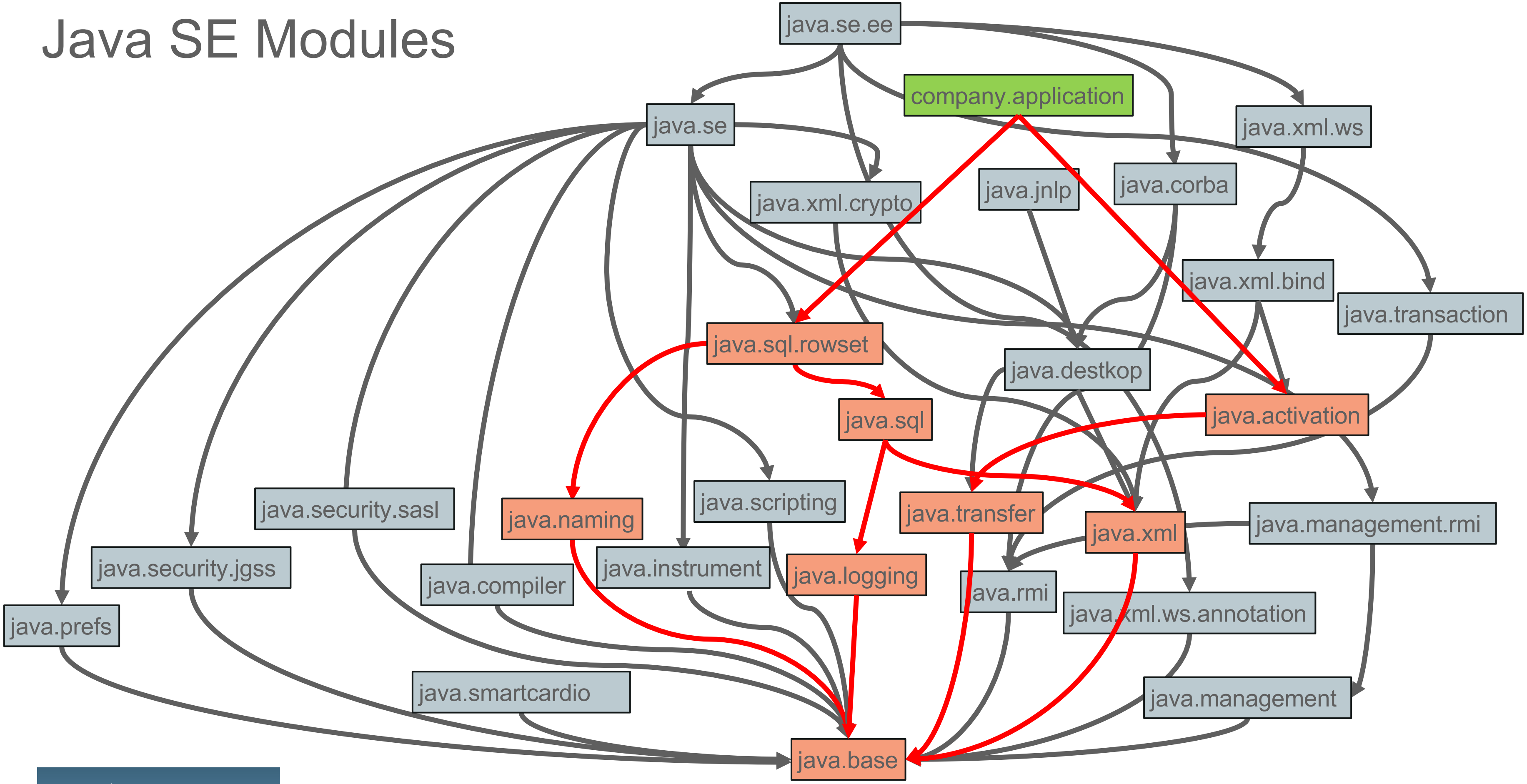


Java SE Modules



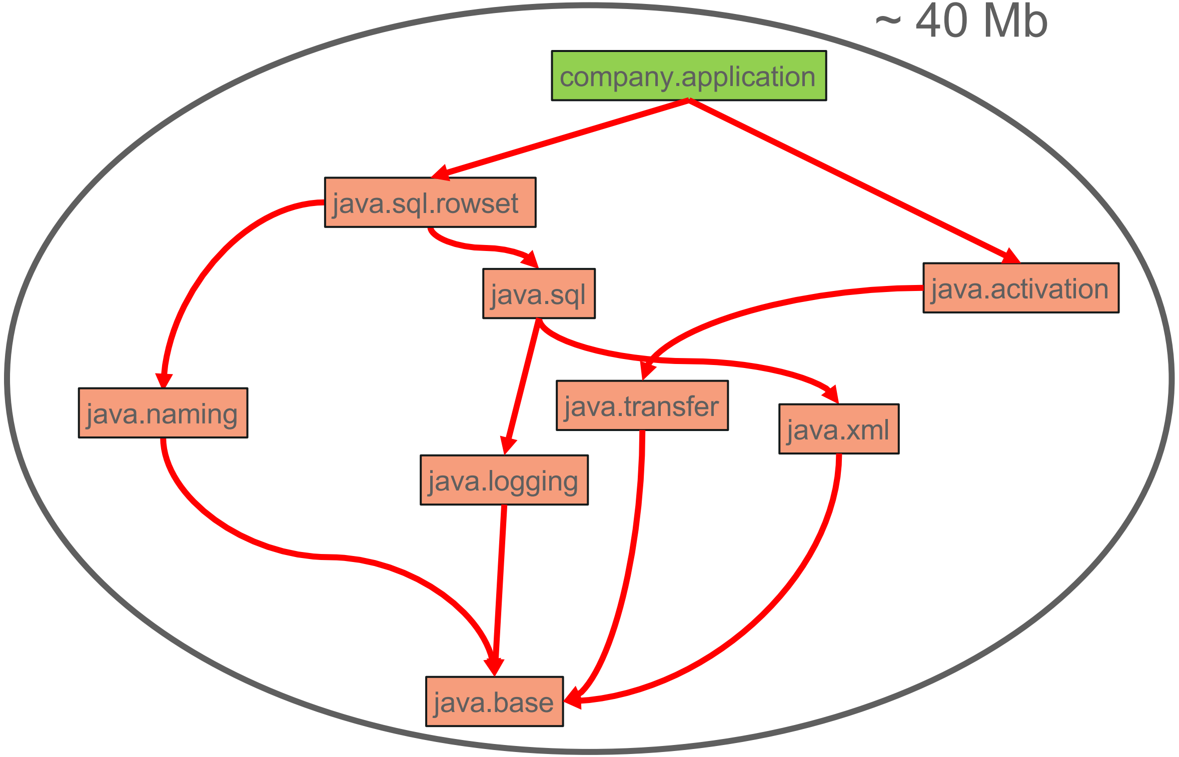
java.se.ee
company.application
java.se
java.xml.ws
java.xml.crypto
java.jnlp
java.corba
java.xml.bind
java.transaction
java.sql.rowset
java.destkop
java.sql
java.activation
java.security.sasl
java.naming
java.scripting
java.transfer
java.xml
java.management.rmi
java.security.jgss
java.instrument
java.compiler
java.logging
java.rmi
java.xml.ws.annotation
java.prefs
java.smartcardio
java.management
java.base

Java SE Modules



Java Custom Runtime

Includes the Modular Application



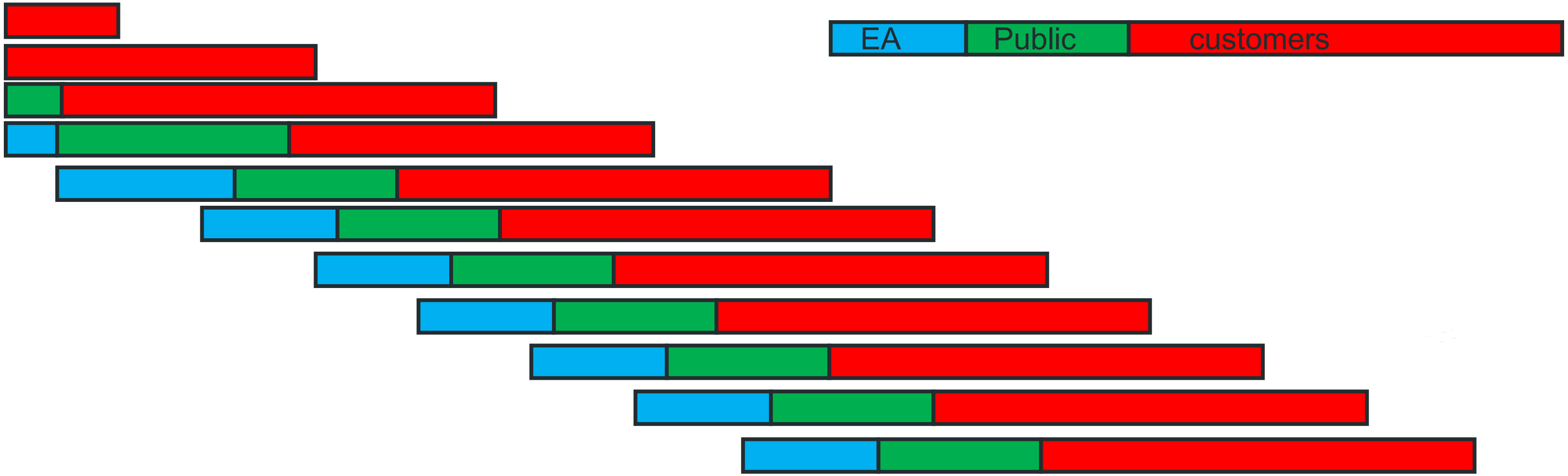
Custom Image
~ 40 Mb

Java™

ORACLE®

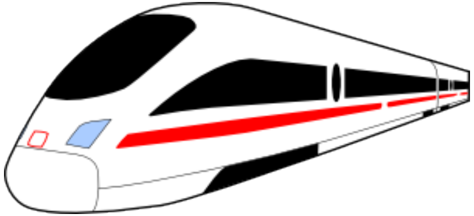
Changing the release model

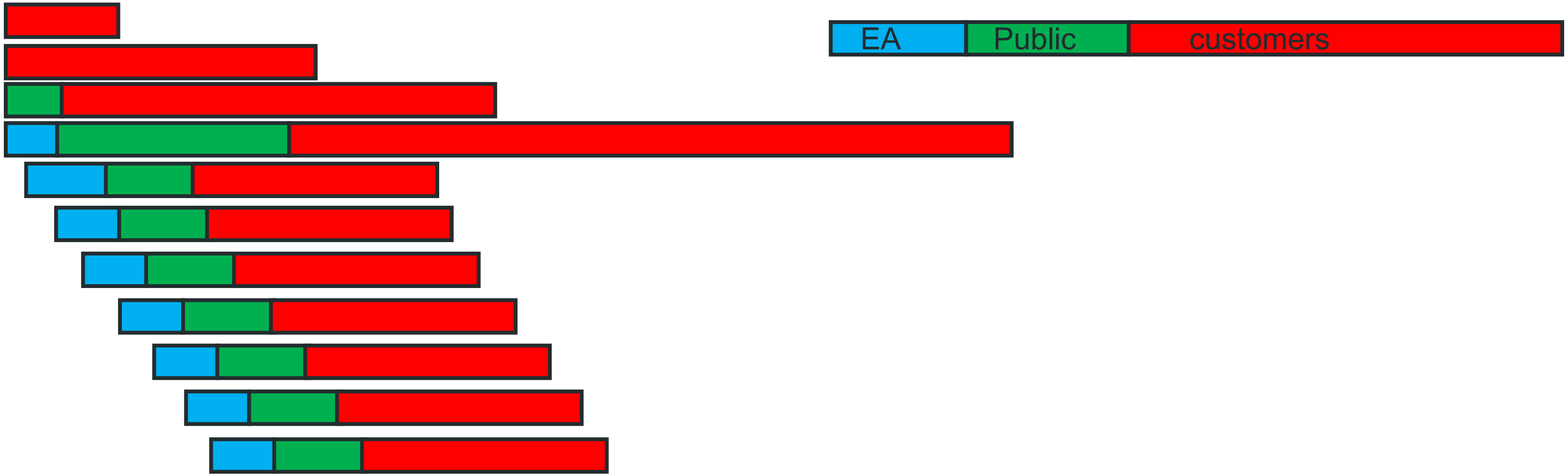
Features determine the release schedule



Faster (and shorter) releases

Constant schedule, features board when ready

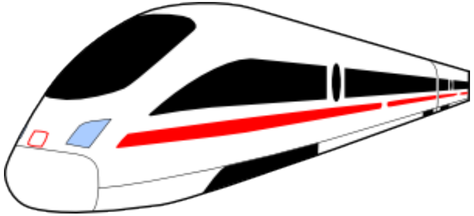


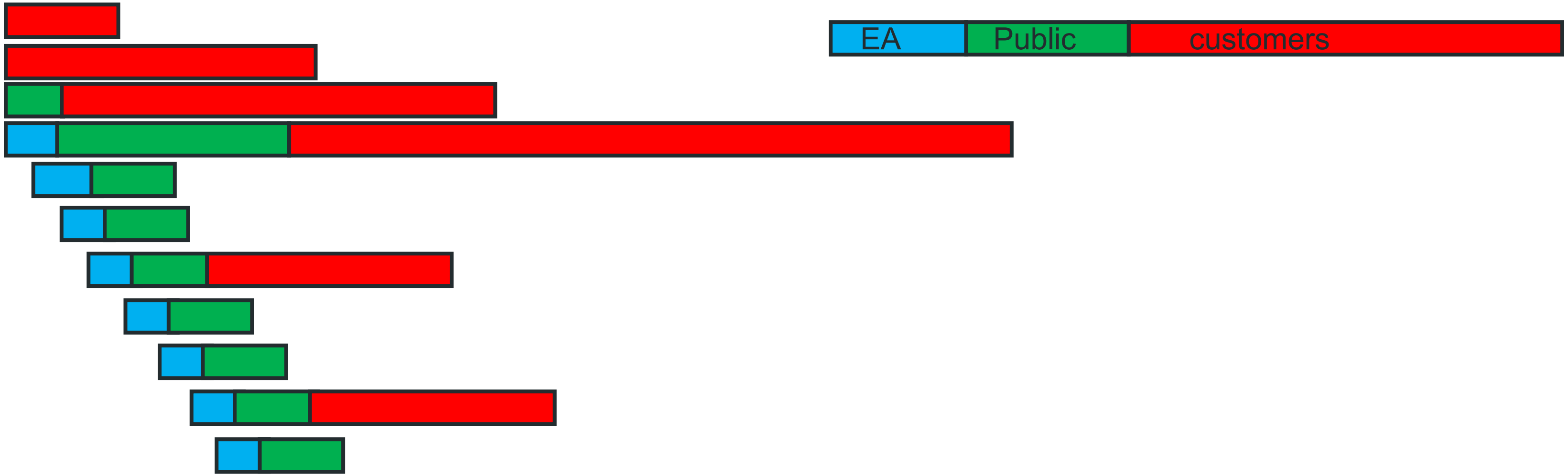


For illustration only nothing has been decided!

Considering all variables

Not all releases need to be the same length…

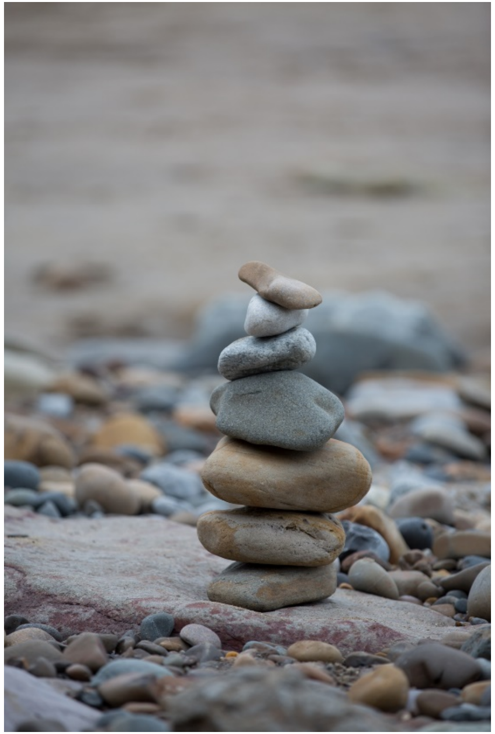




For illustration only nothing has been decided!

JDK 9 designed for the future

- In spite of all these changes…
  - Java remains committed to backward compatibility (for applications that follow the standard)
- Difficult balancing act…
  - JDK 9 has hard forward-looking choices
  - Support for JDK 8 will be extended to provide a generous migration time



Java™

ORACLE®