



# JAVA VERSION API

Miami JVM Group (MJUG)

<https://www.meetup.com/miami-java-user-group/>

Thursday, October 17, 2024



# EUGENIO ALVAREZ



A South Florida software engineering professional. Experienced in organizational design, software design, construction, and deployment. Extensive knowledge of Java. Proponent of Unit testing. An advocate for Agile Software Engineering methods using Kanban and Scrum.

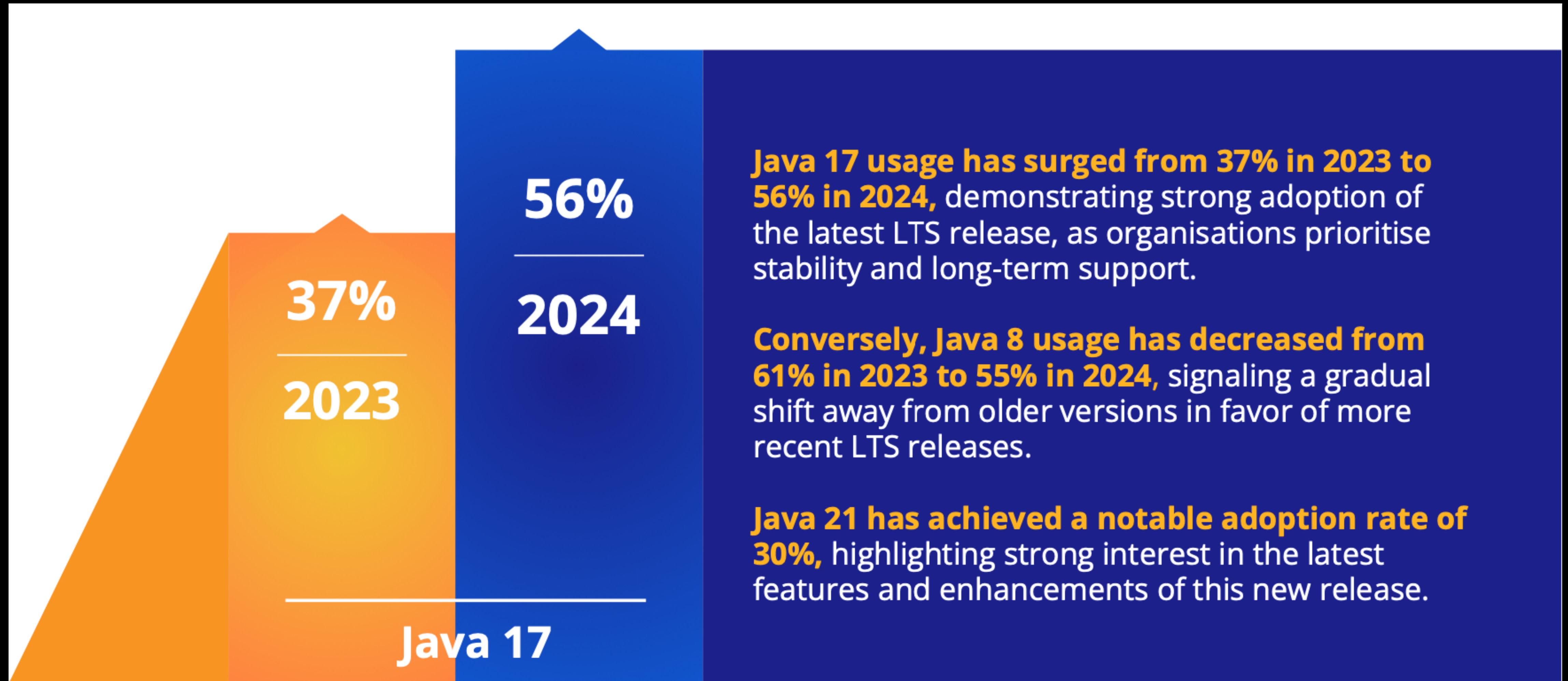
[www.linkedin.com/in/ealvarez](https://www.linkedin.com/in/ealvarez)

# WHAT THIS MEETING IS ABOUT

The Java Version API (`java.lang.Runtime.Version`) was introduced in version 9. The API allows numeric runtime version information to be retrieved inside a running Java application. We will cover use cases and introduce a Runtime Version backport for older versions of Java (8, 7, & 6) that illustrates one use case.



# JAVA VERSION MIGRATION



Source: 2024 Jakarta EE Developer Survey Report, October 2024 <https://outreach.eclipse.foundation/jakarta-ee-developer-survey-2024>

# AGENDA

- Compatibility (bytecode versus source code)
- Quickly switching between different Java versions
- Review Java Version API
- What are the use cases for using the Version API
- Backported Java Version API to older versions (8, 7 & 6)
- Testing between versions
- Continuous version updates versus a finish line mentality


# COMPATIBILITY BETWEEN JAVA VERSIONS

## BYTECODE VERSUS SOURCE CODE

- Bytecode compatibility
- Source code compatibility

# BYTECODE COMPATIBILITY

# JAVA CLASS FILE / BYTECODE VERSIONS

The Java Version Almanac  
**javaalmanac.io**

GitHub

Find Java Resources

The Java Version Almanac / Java Bytecode /

Feedback on this page?

## Class File Versions

Each JDK Release comes with its class file version. Class files are backward compatible. But class files compiled for newer JDK releases cannot be executed and will result in an `UnsupportedClassVersionError`.

The following table lists class file versions for each JDK release. Follow the linked JDK information to find download options for a suitable JDK.

JDK Version	Class File Version
Java 1.0	45.0
Java 1.1	45.3
Java 1.2	46.0
Java 1.3	47.0
Java 1.4	48.0
Java 5	49.0
Java 6	50.0
Java 7	51.0
Java 8	52.0
Java 9	53.0
Java 10	54.0
Java 11	55.0
Java 12	56.0
Java 13	57.0
Java 14	58.0
Java 15	59.0
Java 16	60.0
Java 17	61.0
Java 18	62.0
Java 19	63.0
Java 20	64.0
Java 21	65.0
Java 22	66.0
Java 23	67.0
Java 24	68.0

Data Source

## Magic Number Class File Version

```
00000000: cafe babe 0000 0041 003b 0a00 0200 0307 .....A.;.....
00000010: 0004 0c00 0500 0601 0010 6a61 7661 2f6c .....java/l
00000020: 616e 672f 4f62 6a65 6374 0100 063c 696e ang/Object...<in
00000030: 6974 3e01 0003 2829 5609 0008 0009 0700 it>...()V.....
00000040: 0a0c 000b 000c 0100 106a 6176 612f 6c61 .....java/la
00000050: 6e67 2f53 7973 7465 6d01 0003 6f75 7401 ng/System...out.
00000060: 0015 4c6a 6176 612f 696f 2f50 7269 6e74 ..Ljava/io/Print
00000070: 5374 7265 616d 3b0a 000e 000f 0700 100c Stream;.....
00000080: 0011 0012 0100 116a 6176 612f 6c61 6e67 .....java/lang
00000090: 2f52 756e 7469 6d65 0100 0776 6572 7369 /Runtime...versi
000000a0: 6f6e 0100 1d28 294c 6a61 7661 2f6c 616e on...()Ljava/lan
000000b0: 672f 5275 6e74 696d 6524 5665 7273 696f g/Runtime$Versio
000000c0: 6e3b 0a00 1400 1507 0016 0c00 1700 1801 n;.....
000000d0: 0013 6a61 7661 2f69 6f2f 5072 696e 7453 ..java/io/PrintS
000000e0: 7472 6561 6d01 0007 7072 696e 746c 6e01 tream...println.
```

Class File Version is two 16-bit numbers 0041 0000 or 65.0

Above example: Hex 0041 is 65

65 - 44 = 21

Example Bytecode for Java Version 21

Source: Java Almanac <https://javaalmanac.io/bytecode/versions/>



# JAVA BYTECODE COMPATIBILITY

- Bytecode compatibility refers to the ability of bytecode compiled in an older Java version (e.g., JDK 8) to run on a newer Java runtime (e.g., JDK 11).
- Older runtimes do not support newer class file bytecode features
- `java.lang.UnsupportedClassVersionError` if you try running a newer class with older runtime.

# JAVA BYTECODE COMPATIBILITY

- Bytecode backward compatibility historically much better than source code compatibility.
- Biggest issue:
  - Java runtime may have missing or changed APIs



# SOURCE COMPATIBILITY

# JAVA SOURCE CODE VERSION COMPATIBILITY

- Source compatibility is generally more work than byte code compatibility
- First deploy byte code compiled with previous version
- Compiled Java 8 code deployed Java 11, 17, 21 runtimes



## COMPILE TIME SOURCE COMPATIBILITY

- Since Java 1.4 compiler introduced source code and byte code options for backward compatibility
- `javac -bootclasspath jdk8/jre/lib/rt.jar -source 8 -target 8 Foo.java`

## SOURCE CODE EXAMPLE

- `String _ = "Hello World";`
- Above compiles with Java 7 and older.
- Java 7 bytecode works with Java 23
- Does not compile with Java 8 or newer



# DEPLOYMENT STRATEGIES

# DEPLOYMENT VERSION COMPATIBILITY

- Critical to be able to rollback very quickly in all environments
- Blue-green deployments ideal
- Load balancer running new and older runtimes
- At a minimum, enable quick runtime switching in all environments



# QUICK JAVA DEPLOYMENT OPTIONS

- update-alternatives (Linux / WSL)
- sdkman (Linux / WSL / macOS)
- jEnv (Linux / WSL / macOS)
- Custom scripts

# ALTERNATIVES (LINUX / WSL)

- Install java versions using favorite package manager
- `update-alternatives —list java`
- `sudo update-alternatives —config java`
- `java -version`

Source: update-alternatives man page <https://man7.org/linux/man-pages/man1/update-alternatives.1.html>

# SDKMAN (LINUX / WSL / MACOS)

- `sdk install java 21.0.4`
- `sdk list`
- `sdk use java 21.0.4`
- `sdk default java 21.0.4`

Source: SdkMan website <https://sdkman.io/>



# JENV (LINUX / WSL / MACOS)

- `jenv add /path/to/jdk`
- `jenv versions`
- `jenv global jdkversion`
- `jenv local jdkversion`

Source: jEnv website <https://www.jenv.be/>

# CUSTOM SCRIPT (LINUX / WSL / MACOS)

- Options include:
  - JAVA\_HOME environment variable method along with changing user path
  - Symbolic links to globally change which java version is in the path

# VERSION API



# REVIEW JAVA VERSION API

- Introduced in Java 9
  - `Runtime.version().major()`
  - `Runtime.version().minor()`
  - `Runtime.version().security()`
- Replaced in Java 10
  - `Runtime.version().feature()`
  - `Runtime.version().interim()`
  - `Runtime.version().update()`

Source: Java 10 JavaDoc <https://docs.oracle.com/javase/10/docs/api/java/lang/Runtime.Version.html>

## USE CASES FOR THE VERSION API

- Prevent execution in wrong runtime
- Access new feature runtime
- Work around bug

## JAVA VERSION CHECK USE CASE

- Prevent execution in wrong runtime
- When bytecode is deployed separately from Java runtime.



## USE CASES FOR THE VERSION API

- Access new runtime features
- Introspection is used to access newer features than the source code.

## USE CASES FOR THE VERSION API

- Work around bug(s)
- System level feature may be used based on version to work around a behavior in a specific version

# VERSION API BACKPORT

## VERSION API BACKPORT DETAILS

- Reimplements API in versions 8, 7, 6
- Uses introspection in version 9 and newer
  - `RuntimeVersion.version().feature()`
  - `RuntimeVersion.version().interim()`
  - `RuntimeVersion.version().update()`
  - `RuntimeVersion.version().version()`
  - `RuntimeVersion.version().parse()`

Source: GitHub Repository: <https://github.com/MiamiJUG>



# TESTING BETWEEN VERSIONS

- Ability to switch between versions is key to developer testing
- Integration test environment must test multiple Java version when migrating

# CONTINUOUS VERSION UPDATES

- Java version X or Y is not the finish line
- Continuous updating is necessary to keep up

“To improve is to change; to be perfect is to change often”  
Winston Churchill

Source: Churchill By Himself: The Definitive Collection of Quotations <https://www.amazon.com/Churchill-Himself-Definitive-Collection-Quotations/dp/1586489577>



# Thank You

Miami JVM Group (MJUG)

<https://www.meetup.com/miami-java-user-group/>

[www.linkedin.com/in/ealvarez](http://www.linkedin.com/in/ealvarez)

# REFERENCES

- Oracle JDK Migration Guide (Java 11)
- <https://docs.oracle.com/en/java/javase/11/migrate/index.html>
- 2024 Jakarta EE Developer Survey Report, October 2024
- <https://outreach.eclipse.foundation/jakarta-ee-developer-survey-2024>
- Java System Properties (Java 17)
- <https://docs.oracle.com/en/java/javase/17/docs/api/system-properties.html>
- Java Class File Versions
- <https://javaalmanac.io/bytecode/versions/>
- Java 10 Runtime.Version JavaDoc
- <https://docs.oracle.com/javase/10/docs/api/java/lang/Runtime.Version.html>

# REFERENCES

- Update-alternatives man page
- <https://man7.org/linux/man-pages/man1/update-alternatives.1.html>
- SDKMAN website
- <https://sdkman.io/>
- jEnv website
- <https://www.jenv.be/>
- Churchill By Himself: The Definitive Collection of Quotations
- <https://www.amazon.com/Churchill-Himself-Definitive-Collection-Quotations/dp/1586489577>