ASSIGNMENT NO: LAB 2

SOFTWARE DESIGN AND ARCHITECTURE

TITLE: JAVA ARCHITECTURE AND IT's RELEASES

PREPARED BY: **Rehman Ghani(Fa22-Bse-099)**

Syed Muhammad Usman(Fa22-Bse-058)

Zain-ul-Abideen(Fa22-Bse-056)

SECTION NO: **5B**

SUBMITTED TO: **Mukhtiar Zamin**

DEPARTMENT OF SOFTWARE ENGINEERING

Preparation Of Report:

Rehman Ghani(Java Architecture , Release Table, JDK 1.0 to Java SE7, Java SE23 to Java SE25)

Syed Muhammad Usman(Java SE8 to JavaSE15)

Zain ul Abideen (Java SE15 to Java SE23)

JAVA ARCHITECTURE:

There are two processes in Java — Compilation and Interpretation.

- The Java source code goes to the compiler.
- The Java Compiler converts it into byte codes
- The bytes codes are then converted into machine by the JVM
- The Machine code is executed directly by the machine(Operating System)

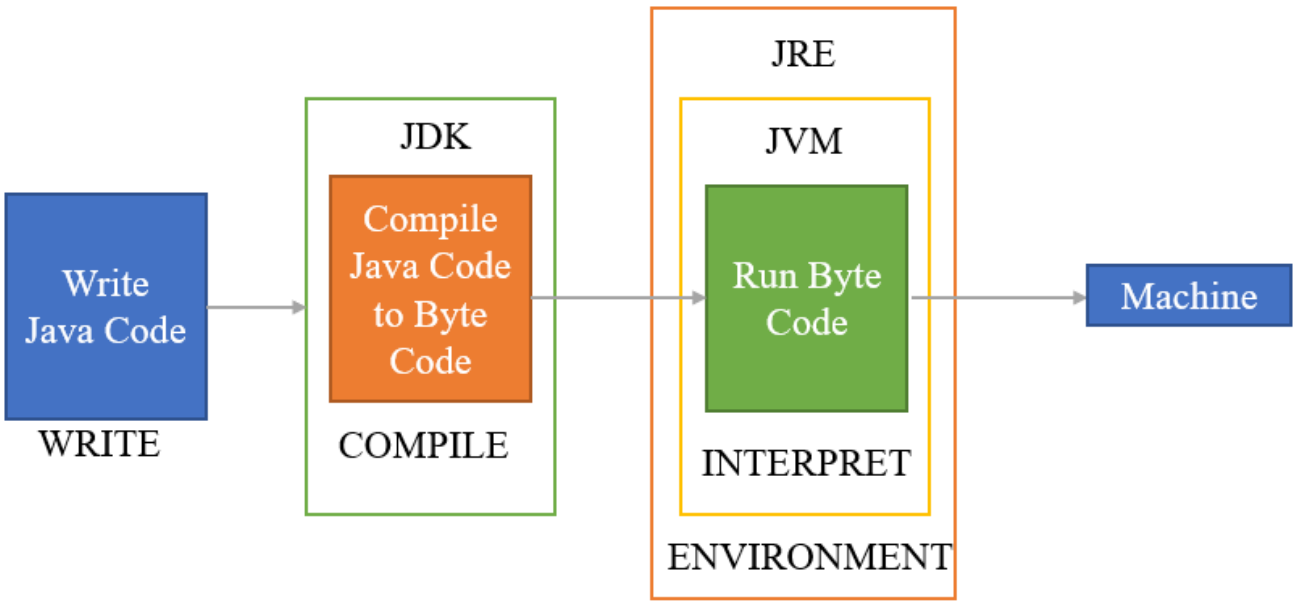
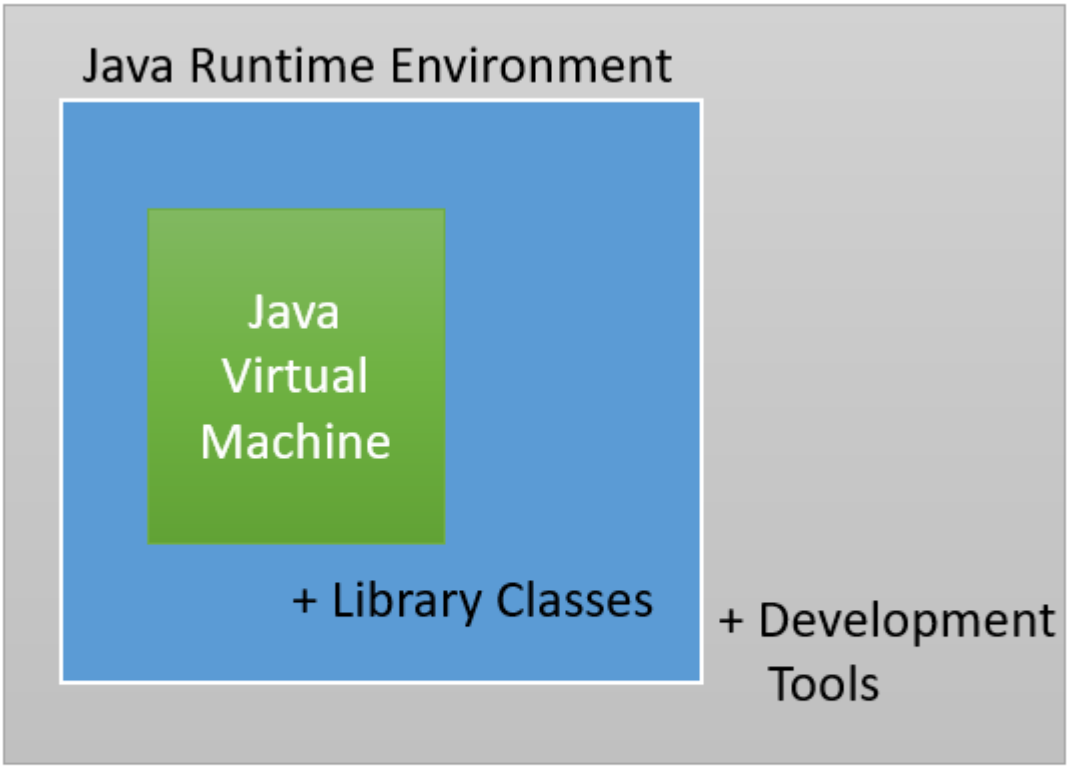


Fig: Java Architecture

Components of Java Architecture

The different components of Java architecture are

- JRE — Java Runtime Environment
- JDK — Java Development Kit
- JVM — Java Virtual Machine



JDK = JRE + Development Tools
JRE = JVM + Library Classes

Java Runtime Environment(JRE)

Java Runtime Environment provides an platform where all the applications like JVM and other runtime libraries linked together to run your Java Program. It builds a runtime environment where you can execute the Java program. The JRE also initiates the JVM for its execution. JRE has the required software and libraries to run the programs.

Java Development Kit (JDK)

Java Development Kit is the set of libraries, compiler, interpreter and other set of programs that will help you to build the Java program. Once JDK installed on machine then start developing, compile and run the Java program. You cannot compile Java program without JDK installed on machine. Once you compile the code with JDK tools, you can get an Byte Code File.

- **java** : it is the launcher for all the java applications.
- **javac** : compiler of the java programming languages.
- **javadoc**: it is the API documentation generator.
- **jar**: creates and manage all the JAR files.

Java Virtual Machine(JVM)

JVM is the interpreter that executes the byte codes into Machine code(instructions). The beautiful quality is WORA(Write Once Run Anywhere). This means code can runs its applications on any platform. This makes Java as Platform Independent.
In a nutshell, JVM performs the following functions:

- Loads the code
- Verifies the code
- Executes the code
- Provides runtime environment

The below architecture depicts the architecture of the JVM. Let's see the each element in detail:

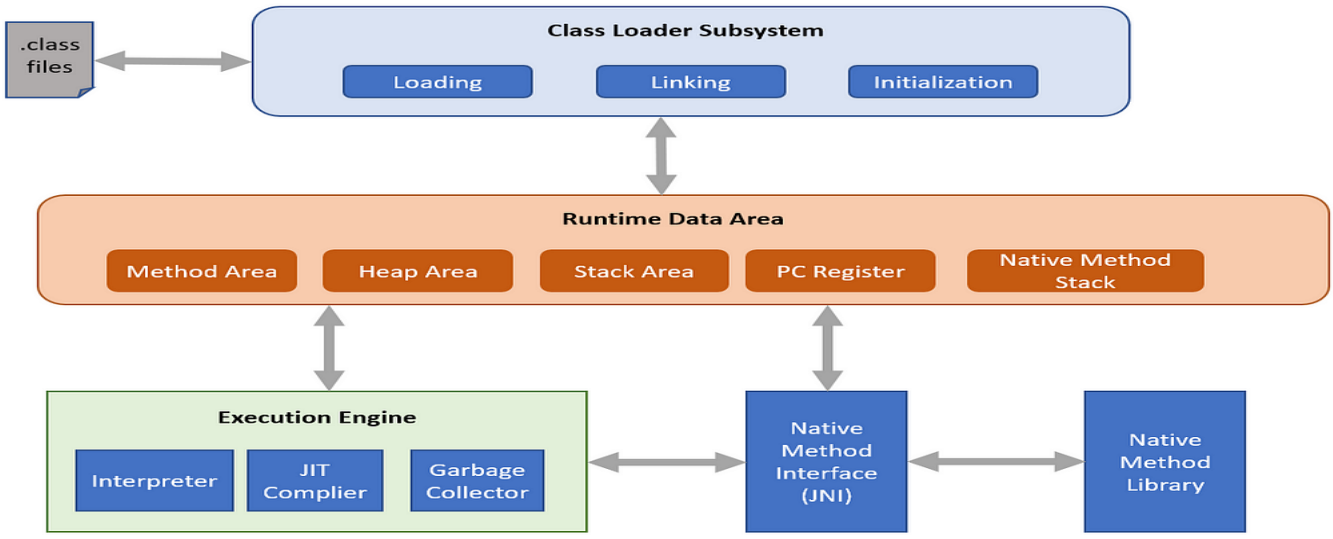


Fig: Components of JVM

Refrence : https://devaraj-durairaj.medium.com/java-architecture-and-components-febd83b3adfc

JAVA VERSION HISTORY:

The Java language has undergone several changes since JDK 1.0 as well as numerous additions of classes and packages to the standard library. Since J2SE 1.4, the evolution of the Java language has been governed by the Java Community Process (JCP), which uses *Java Specification Requests* (JSRs) to propose and specify additions and changes to the Java platform. The language is specified by the *Java Language Specification* (JLS); changes to the JLS are managed under JSR 901. In September 2017, Mark Reinhold, chief Architect of the Java Platform, proposed to change the release train to "one feature release every six months" rather than the then-current two-year schedule.[1][2] This proposal took effect for all following versions, and is still the current release schedule.

In addition to the language changes, other changes have been made to the Java Class Library over the years, which has grown from a few hundred classes in JDK 1.0 to over three thousand in J2SE 5. Entire new APIs, such as Swing and Java2D, have been introduced, and many of the original JDK 1.0 classes and methods have been deprecated, and very few APIs have been removed (at least one, for threading, in Java 22[3]). Some programs allow the conversion of Java programs from one version of the Java platform to an older one (for example Java 5.0 backported to 1.4) (see Java backporting tools).

Regarding Oracle's Java SE support roadmap,[4] Java SE 23 is the latest version, while versions 21, 17, 11 and 8 are the currently supported long-term support (LTS) versions, where Oracle Customers will receive Oracle Premier Support. Oracle continues to release no-cost public Java 8 updates for development[4] and personal use indefinitely. Oracle also continues to release no-cost public Java 17 LTS updates for all users, including commercial and production use until September 2024.[5]

In the case of OpenJDK, both commercial long-term support and free software updates are available from multiple organizations in the broader community.[6]

Java 23 was released on September 17, 2024.

RELEASE TABLE

Java version overview					
Version	**Type**	**Class file format version**[7]	**Release date**	**End of public updates (free)**	**End of extended support (paid)**
JDK 1.0		45[8]	23rd January 1996	May 1996	—
JDK 1.1		45	18th February 1997	October 2002	—

J2SE 1.2		46	4th December 1998	November 2003	—
J2SE 1.3		47	8th May 2000	March 2006	—
J2SE 1.4		48	13th February 2002	October 2008	—
J2SE 5.0 (1.5)		49	30th September 2004	October 2009	—
Java SE 6 (1.6)		50	11th December 2006	April 2013 for Oracle December 2018 for Azul[9]	December 2016 for Red Hat[10] October 2018 for Oracle[11] December 2027 for Azul[9] March 2028 for BellSoft Liberica[12]
Java SE 7 (1.7)		51	28th July 2011	July 2015 for Oracle July 2022 for Azul[9]	June 2020 for Red Hat[10] July 2022 for Oracle[13] December 2027 for Azul[9] March 2028 for BellSoft Liberica[12]
Java SE 8 (1.8)	LTS	52	18th March 2014	April 2019 for Oracle November 2026 for Eclipse Temurin[14] November 2026 for Red Hat[10] November 2026 for Azul[9] December 2030 for Amazon Corretto[15]	December 2030 for Oracle[4] December 2030 for Azul[9] March 2031 for BellSoft Liberica[12]
Java SE 9 (1.9)		53	21st September 2017	March 2018	—
Java SE 10 (1.10)		54	20th March 2018	September 2018	—
Java SE 11	LTS	55	25th September 2018	April 2019 for Oracle September 2027 for Microsoft Build of OpenJDK[16] October 2024 for Red Hat[10] October 2027 for	January 2032 for Azul[9] March 2032 for BellSoft Liberica[12] October 2027 for

				Eclipse Temurin[14] October 2027 for Azul[9] January 2032 for Amazon Corretto[15] January 2032 for Azul[9]	Red Hat[10] January 2032 for Oracle[4]
Java SE 12		56	19th March 2019	September 2019	—
Java SE 13		57	17th September 2019	March 2020	—
Java SE 14		58	17th March 2020	September 2020	—
Java SE 15		59	16th September 2020	March 2021	—
Java SE 16		60	16th March 2021	September 2021	—
Java SE 17	LTS	61	14th September 2021	September 2024 for Oracle[4] September 2027 for Microsoft Build of OpenJDK[16] October 2027 for Eclipse Temurin[14] October 2027 for Red Hat[10] October 2029 for Amazon Corretto[15] September 2029 for Azul[9]	September 2029 for Oracle[4] March 2030 for BellSoft Liberica[12]
Java SE 18		62	22nd March 2022	September 2022	—
Java SE 19		63	20th September 2022	March 2023	—
Java SE 20		64	21st March 2023	September 2023	—
Java SE 21	LTS	65	19th September 2023	September 2028 for Oracle[4] September 2028 for Microsoft Build of OpenJDK[16] December 2029 for Red Hat[10] December 2029 for Eclipse Temurin[14] October 2030 for Amazon Corretto[15]	September 2031 for Oracle[4] March 2032 for BellSoft Liberica[12]

				September 2031 for Azul[9]	
Java SE 22		66	19th March 2024	September 2024	—
Java SE 23		67	17th September 2024	March 2025 for Oracle September 2032 for Azul[9]	—
Java SE 24		68	March 2025	September 2025	—
Java SE 25	LTS	69	September 2025	September 2030 for Oracle[4]	September 2033 for Oracle[4] March 2034 for BellSoft Liberica[12]
Legend: Old version, not maintained Old version, still maintained **Latest version** Future release					

JDK 1.0

JDK 1.0	
Released	January 23, 1996 (28 years ago)

The first version was released on January 23, 1996.[17][18] The first stable version, JDK 1.0.2, is called Java 1.[18]

JDK 1.1

JDK 1.1	
Released	February 19, 1997 (27 years ago)

Major additions in the release on February 19, 1997 included:[19]

- extensive retooling of the Abstract Window Toolkit (AWT) event model
- inner classes added to the language
- JavaBeans
- Java Database Connectivity (JDBC)
- Java remote method invocation (RMI) and serialization

- reflection which supported Introspection only, no modification at runtime was possible. (The ability to modify objects reflectively was added in J2SE 1.2, by introducing the AccessibleObject class and its subclasses such as the Field class.)
- Just-in-time compilation (JIT) on Microsoft Windows platforms, produced for JavaSoft by Symantec
- Internationalization and Unicode support originating from Taligent[20]

J2SE 1.2

J2SE 1.2	
Codename	Playground
Released	December 8, 1998 (26 years ago)

The release on December 8, 1998 and subsequent releases through J2SE 5.0 were rebranded retrospectively **Java 2** and the version name "J2SE" (Java 2 Platform, Standard ion) replaced JDK to distinguish the base platform from J2EE (Java 2 Platform, Enterprise ion) and J2ME (Java 2 Platform, Micro ion). This was a very significant release of Java as it tripled the size of the Java platform to 1520 classes in 59 packages. **Major additions included:**

- strictfp keyword (by JVM 17 an obsolete keyword, should not be used in new code)
- The Swing graphical API was integrated into the core classes.
- Sun's JVM was equipped with a JIT compiler for the first time.
- Java plug-in
- Java IDL, an IDL implementation for CORBA interoperability
- Collections framework

J2SE 1.3

J2SE 1.3	
Codename	Kestrel
Released	May 8, 2000 (24 years ago)

The most notable changes in the May 8, 2000 release were:

- HotSpot JVM included (the HotSpot JVM was first released in April 1999 for the J2SE 1.2 JVM)
- RMI was modified to support optional compatibility with CORBA.

- Java Naming and Directory Interface (JNDI) included in core libraries (previously available as an extension)
- Java Platform Debugger Architecture (JPDA)
- JavaSound
- Synthetic proxy classes

Java 1.3 is the last release of Java to officially support Microsoft Windows 95.[24]

J2SE 1.4

J2SE 1.4	
Codename	Merlin
Released	February 6, 2002 (22 years ago)
Support ended	
Public	October 2008
Paid	February 2013

The February 6, 2002 release was the first release of the Java platform developed under the Java Community Process as JSR 59. **Major changes included:**

- Language changes
  - assert keyword (specified in JSR 41)
- Library improvements
  - Regular expressions modeled after Perl regular expressions
  - Exception chaining allows an exception to encapsulate original lower-level exception
  - Internet Protocol version 6 (IPv6) support
  - Non-blocking I/O (named NIO) (specified in JSR 51)
  - Logging API (specified in JSR 47)
  - Image I/O API for reading and writing images in formats like JPEG and PNG
  - Integrated XML parser and XSLT processor (JAXP) (specified in JSR 5 and JSR 63)
  - Integrated security and cryptography extensions (JCE, JSSE, JAAS)
  - Java Web Start included (Java Web Start was first released in March 2001 for J2SE 1.3) (specified in JSR 56)
  - Preferences API (java.util.prefs)

Public support and security updates for Java 1.4 ended in October 2008. Paid security updates for Oracle customers ended in February 2013.[27]

Java SE 5

Java SE 5	
Codename	Tiger
Released	September 30, 2004 (20 years ago)
Support ended	
Public	November 2009
Paid	April 2015

The release on September 30, 2004 was originally numbered 1.5, which is still used as the internal version number. The number was changed to "better reflect the level of maturity, stability, scalability and security of the J2SE".[28] This version was developed under JSR 176.

Java SE 5 entered its end-of-public-updates period on April 8, 2008; updates are no longer available to the public as of November 3, 2009. Updates were available to paid Oracle customers until May 2015.[4]

Tiger added a number of significant new language features:

- Generics: provides compile-time (static) type safety for collections and eliminates the need for most typecasts (type conversion) (specified by JSR 14)
- Metadata: also called annotations; allows language constructs such as classes and methods to be tagged with additional data, which can then be processed by metadata-aware utilities (specified by JSR 175)
- Autoboxing/unboxing: automatic conversions between primitive types (such as int) and primitive wrapper classes (such as Integer) (specified by JSR 201)
- Enumerations: the enum keyword creates a typesafe, ordered list of values (such as Day.MONDAY, Day.TUESDAY, etc.); previously this could only be achieved by non-typesafe constant integers or manually constructed classes (typesafe enum pattern) (specified by JSR 201)
- Varargs: the last parameter of a method can now be declared using a type name followed by three dots (e.g. void drawtext(String... lines)); in the calling code any number of parameters of that type can be used and they are then placed in an array to be passed to the method, or alternatively the calling code can pass an array of that type
- Enhanced for each loop: the for loop syntax is extended with special syntax for iterating over each member of either an array or any Iterable, such as the standard Collection classes (specified by JSR 201)

- Improved semantics of execution for multi-threaded Java programs; the new Java memory model addresses issues of complexity, effectiveness, and performance of previous specifications[31]
- Static imports

There were also the following improvements to the standard libraries:

- Automatic stub generation for RMI objects
- Swing: New skinnable look and feel, called synth
- The concurrency utilities in package java.util.concurrent[32]
- Scanner class for parsing data from various input streams and buffers

Java 5 is the last release of Java to officially support Microsoft Windows 98 and Windows ME,[33] while Windows Vista was the newest version of Windows that Java SE 5 was supported on prior to Java 5 going end-of-life in October of 2009.[27]

Java 5 Update 5 (1.5.0_05) is the last release of Java to work on Windows 95 (with Internet Explorer 5.5 installed) and Windows NT 4.0.[34]

Java 5 was first available on Apple Mac OS X 10.4 (Tiger)[35] and was the default version of Java installed on Apple Mac OS X 10.5 (Leopard).

Public support and security updates for Java 1.5 ended in November 2009. Paid security updates for Oracle customers ended in April 2015.

Versioning change

This version introduced a new versioning system for the Java language, although the old versioning system continued to be used for developer libraries:

Both version numbers "1.5.0" and "5.0" are used to identify this release of the Java 2 Platform Standard ion. Version "5.0" is the product version, while "1.5.0" is the developer version. The number "5.0" is used to better reflect the level of maturity, stability, scalability and security of the J2SE.

—*"Version 1.5.0 or 5.0?", Java release notes*[36]

This correspondence continued through later releases (Java 6 = JDK 1.6, Java 7 = JDK 1.7, and so on).

Java SE 6

Java SE 6	
Codename	Mustang

Released	November 11, 2006 (18 years ago)
	Support ended
Public	February 2013

As of the version released on December 11, 2006, Sun replaced the name "J2SE" with **Java SE** and dropped the ".0" from the version number.[37] Internal numbering for developers remains 1.6.0.[38]

This version was developed under JSR 270.

During the development phase, new builds including enhancements and bug fixes were released approximately weekly. Beta versions were released in February and June 2006, leading up to a final release that occurred on December 11, 2006.

Major changes included in this version:

- Support for older Win9x versions dropped; unofficially, Java 6 Update 7 was the last release of Java shown to work on these versions of Windows.[*citation needed*] This is believed[*by whom?*] to be due to the major changes in Update 10.
- Scripting Language Support (JSR 223): Generic API for tight integration with scripting languages, and built-in Mozilla JavaScript Rhino integration.
- Dramatic performance improvements for the core platform,[41][42] and Swing.
- Improved Web Service support through JAX-WS (JSR 224).
- JDBC 4.0 support (JSR 221).
- Java Compiler API (JSR 199): an API allowing a Java program to select and invoke a Java Compiler programmatically.
- Upgrade of JAXB to version 2.0: Including integration of a StAX parser.
- Support for pluggable annotations (JSR 269).[43]
- Many GUI improvements, such as integration of SwingWorker in the API, table sorting and filtering, and true Swing double-buffering (eliminating the gray-area effect).
- JVM improvements include: synchronization and compiler performance optimizations, new algorithms and upgrades to existing garbage collection algorithms, and application start-up performance.

Java 6 can be installed to Mac OS X 10.5 (Leopard) running on 64-bit (Core **2** Duo and higher) processor machines.[44] Java 6 is also supported by both 32-bit and 64-bit machines running Mac OS X 10.6 (Snow Leopard).

Java 6 reached the end of its supported life in February 2013, at which time all public updates, including security updates, were scheduled to be stopped.[45][46] Oracle released two more updates to Java 6 in March and April 2013, which patched some security vulnerabilities.[47][48]

Java 6 updates

After Java 6 release, Sun, and later Oracle, released several updates which, while not changing any public API, enhanced end-user usability or fixed bugs.[49]

Java SE 7

Java SE 7	
Codename	Dolphin[86]
Released	July 28, 2011 (13 years ago)
Support ended	
Public	April 2015
Paid	June 2022

Java 7 was a major update that launched on July 7, 2011[87] and was made available for developers on July 28, 2011.[88] The development period was organized into thirteen milestones; on June 6, 2011, the last of the thirteen milestones was finished.[88][89] On average, 8 builds (which generally included enhancements and bug fixes) were released per milestone. The feature list at the OpenJDK 7 project lists many of the changes.

Additions in Java 7 include:[90]

- JVM support for dynamic languages, with the new invokedynamic bytecode under JSR-292,[91] following the prototyping work currently done on the Multi Language Virtual Machine
- Compressed 64-bit pointers[92] (available in Java 6 with -XX:+UseCompressedOops)[93]
- Project Coin language features:[94]
    - Strings in switch[95]
    - Automatic resource management in try-statement aka *try-with-resources statement*[96]
    - Improved type inference for generic instance creation, aka *the diamond operator* <>[97]
    - Simplified varargs method declaration[98]
    - Binary integer literals[99]
    - Allowing underscores in numeric literals[100]
    - Catching multiple exception types and rethrowing exceptions with improved type checking[101]
- Concurrency utilities under JSR 166[102]

- New file I/O library (defined by JSR 203) adding support for multiple file systems, file metadata and symbolic links. The new packages are java.nio.file, java.nio.file.attribute and java.nio.file.spi[103][104]
- Timsort is used to sort collections and arrays of objects instead of merge sort
- Library-level support for elliptic curve cryptography algorithms
- An XRender pipeline for Java 2D, which improves handling of features specific to modern GPUs
- New platform APIs for the graphics features originally implemented in version 6u10 as unsupported APIs[105]
- Enhanced library-level support for new network protocols, including SCTP and Sockets Direct Protocol
- Upstream updates to XML and Unicode
- Java deployment rule sets[106]

Lambda (Java's implementation of lambda functions), Jigsaw (Java's implementation of modules), and part of Coin were dropped from Java 7, and released as part of Java 8 (except for Jigsaw, which was released in Java 9).[107][108]

Java 7 was the default version to download on java.com from April 2012 until Java 8 was released.[109]

Java 7 updates

Oracle issued public updates to the Java 7 family on a quarterly basis[110] until April 2015 when the product reached the end of its public availability.[111] Further updates for JDK 7, which continued until July 2022, are only made available to customers with a support contract.[112]

By: **USMAN**

Java SE 8

	LTS version
Released	March 18, 2014 (10 years ago)
# of JEPs	8

Java 8 was released on March 18, 2014,[148][149] and included some features that were planned for Java 7 but later deferred.[150]

Work on features was organized in terms of JDK Enhancement Proposals (JEPs).[151]

- JSR 335, JEP 126: Language-level support for lambda expressions (officially, lambda expressions; unofficially, closures) under Project Lambda[152] and default methods (virtual extension methods)[153][154][155] which can be used to add methods to interfaces without breaking existing implementations. There was an ongoing debate in the Java community on whether to add support for lambda expressions.[156][157] Sun later declared that lambda expressions would be included in Java and asked for community input to refine the feature.[158] Supporting lambda expressions also enables functional-style operations on streams of elements, such as MapReduce-inspired transformations on collections. Default methods can be used by an author of an API to add new methods to an interface without breaking the old code using it. Although it was not their primary intent,[153] default methods can also be used for multiple inheritance of behavior (but not state).
- JEP 174: Project Nashorn, a JavaScript runtime which can run JavaScript code embedded within applications
- JEP 104: Annotation on Java types
- Unsigned integer arithmetic[159]
- JEP 120: Repeating annotations
- JEP 150: Date and time API
- JEP 178: Statically-linked JNI libraries
- JEP 153: Launch JavaFX applications (direct launching of JavaFX application JARs)
- JEP 122: Remove the permanent generation

Java 8 is not supported on Windows XP[160] but as of JDK 8 update 25, it can still be installed and run under Windows XP.[161] Previous updates of JDK 8 could be run under XP by downloading archived zip format file and unzipping it for the executable. The last version of Java 8 could run on XP is update 251.

From October 2014, Java 8 was the default version to download (and then again the download replacing Java 9) from the official website.[162] "Oracle will continue to provide Public Updates and auto updates of Java SE 8, Indefinitely for Personal Users".[163]

Java 8 updates

Java SE 9

Java SE 9	
Released	September 21, 2017 (7 years ago)
# of JEPs	9
Support ended	

Public	March 2018

Java SE 9 was made available on September 21, 2017[242] due to controversial acceptance of the current implementation of Project Jigsaw by Java Executive Committee[243] which led Oracle to fix some open issues and concerns and to refine some critical technical questions. In the last days of June 2017, Java Community Process expressed nearly unanimous consensus on the proposed Module System scheme.[244]

- JSR 376: Modularization of the JDK under Project Jigsaw (Java Platform Module System)[108]
- JavaDB was removed from JDK[245]
- JEP 193: Variable handles, define a standard means to invoke the equivalents of various java.util.concurrent.atomic and sun.misc.Unsafe operations
- JEP 213: Milling Project Coin, allow @SafeVarargs on private instance methods; Allow effectively-final variables to be used as resources in the try-with-resources statement; Allow diamond with anonymous classes if the argument type of the inferred type is denotable; Complete the removal, begun in Java SE 8, of underscore from the set of legal identifier names; Support for private methods in interfaces
- JEP 222: jshell: The Java Shell (Read-Eval-Print Loop): JShell is a REPL command-line interface for the Java language.[246]
- JEP 254: Compact Strings
- JEP 263: HiDPI graphics: automatic scaling and sizing
- JEP 266: More concurrency updates, it includes a Java implementation of Reactive Streams,[247] including a new Flow class[248] that included the interfaces previously provided by Reactive Streams[249]
- JEP 268: XML catalogs
- JEP 282: jlink: The Java Linker, create a tool that can assemble and optimize a set of modules and their dependencies into a custom run-time image. It effectively allows to produce a fully usable executable including the JVM to run it
- JEP 295: Ahead-of-Time Compilation, ahead-of-time compilation provided by GraalVM

The first Java 9 release candidate was released on August 9, 2017.[250] The first stable release of Java 9 was on September 21, 2017.[251]

History

At JavaOne 2011, Oracle discussed features they hoped to release for Java 9 in 2016.[252] Java 9 should include better support for multi-gigabyte heaps, better native code integration, a different default garbage collector (G1, for "shorter response times")[253] and a self-tuning JVM.[254] In early 2016, the release of Java 9 was rescheduled for March 2017[255] and later again postponed four more months to July 2017.[256]

Java 9 updates

Table of Java 9 updates show

Java SE 10

Java SE 10	
Released	March 20, 2018 (6 years ago)
# of JEPs	12
Support ended	
Public	September 2018

OpenJDK 10 was released on March 20, 2018, with twelve new features confirmed.[262] Among these features were:

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

The first of these JEP 286 *Local-Variable Type Inference*, allows the var keyword to be used for local variables with the actual type calculated by the compiler. Due to this change, developers can do the following instead of manually specifying the variable's type:

```
var list = new ArrayList<String>(); // infers ArrayList<String>
var stream = list.stream();         // infers Stream<String>
```

Java 10 updates

[ ]

Table of Java 10 updates show

Java SE 11

[_]

Java SE 11	
LTS version	
Released	September 25, 2018 (6 years ago)
# of JEPs	17
Removal(s)	
Notable	Java applets, Java Web Start, JavaFX, JavaEE, and CORBA modules

JDK 11 was released on September 25, 2018 and the version is currently open for bug fixes. It offers LTS, or Long-Term Support. Among others, Java 11 includes a number of new features, such as:[268]

- JEP 181: Nest-Based Access Control
- JEP 309: Dynamic Class-File Constants
- JEP 315: Improve Aarch64 Intrinsics
- JEP 318: Epsilon: A No-Op Garbage Collector
- JEP 320: Remove the Java EE and CORBA Modules
- JEP 321: HTTP Client (Standard)
- JEP 323: Local-Variable Syntax for Lambda Parameters
- JEP 324: Key Agreement with Curve25519 and Curve448
- JEP 327: Unicode 10
- JEP 328: Flight Recorder
- JEP 329: ChaCha20 and Poly1305 Cryptographic Algorithms
- JEP 330: Launch Single-File Source-Code Programs
- JEP 331: Low-Overhead Heap Profiling
- JEP 332: Transport Layer Security (TLS) 1.3
- JEP 333: ZGC: A Scalable Low-Latency Garbage Collector (Experimental)
- JEP 335: Deprecate the Nashorn JavaScript Engine
- JEP 336: Deprecate the Pack200 Tools and API

A number of features from previous releases were dropped; in particular, Java applets and Java Web Start are no longer available. JavaFX, Java EE and CORBA modules have been removed from JDK.[269]

Java 11 updates

[ ]

Table of Java 11 updates show

Java SE 12

[ ]

Java SE 12	
Released	March 19, 2019 (5 years ago)
# of JEPs	8
Addition(s)	
Preview(s)	Enhanced switch statements
Support ended	
Public	September 2019

JDK 12 was released on March 19, 2019. Among others, Java 12 includes a number of new features, such as:[316]

- JEP 189: Shenandoah: A Low-Pause-Time Garbage Collector (Experimental)
- JEP 230: Microbenchmark Suite
- JEP 325: Switch Expressions (Preview)
- JEP 334: JVM Constants API
- JEP 340: One AArch64 Port, Not Two
- JEP 341: Default CDS Archives
- JEP 344: Abortable Mixed Collections for G1
- JEP 346: Promptly Return Unused Committed Memory from G1

The preview feature JEP 325 extends the switch statement so it can also be used as an expression, and adds a new form of case label where the right hand side is an expression. No

break statement is needed. For complex expressions a yield statement can be used. This becomes standard in Java SE 14.

```
int ndays = switch(month) {
   case JAN, MAR, MAY, JUL, AUG, OCT, DEC -> 31;
   case APR, JUN, SEP, NOV -> 30;
   case FEB -> {
      if (year % 400 == 0) yield 29;
      else if (year % 100 == 0) yield 28;
      else if (year % 4 == 0) yield 29;
      else yield 28; }
};
```

Java 12 updates

[ ]

Table of Java 12 updates show

Java SE 13

[ ]

Java SE 13	
Released	September 17, 2019 (5 years ago)
# of JEPs	5
Addition(s)	
Preview(s)	Enhanced switch statements, text blocks
Support ended	
Public	March 2023[321]

JDK 13 was released on September 17, 2019. Java 13 includes the following new features, as well as "hundreds of smaller enhancements and thousands of bug fixes".[322]

- JEP 350: Dynamic CDS Archives

- JEP 351: ZGC: Uncommit Unused Memory
- JEP 353: Reimplement the Legacy Socket API
- JEP 354: Switch Expressions (Preview)
- JEP 355: Text Blocks (Preview)

JEP 355 *Text Blocks* allows multiline string literals:

```
String html = """
            <html lang="en">
               <body>
                  <p>Hello, world</p>
               </body>
            </html>
            """;
```

Java 13 updates

[ ]

Table of Java 13 updates show

Java SE 14

[ ]

Java SE 14	
Released	March 17, 2020 (4 years ago)
# of JEPs	16
Addition(s)	
Notable	Helpful NullPointerExceptions, enhanced switch statements
Preview(s)	Pattern matching for instanceof, records, text blocks
Incubating	jpackager, Foreign memory access

Removal(s)	
Notable	Remove Concurrent Mark Sweep garbage collector
Support ended	
Public	September 2020

JDK 14 was released on March 17, 2020. Java 14 includes the following new features, as well as "hundreds of smaller enhancements and thousands of bug fixes".[327]

- JEP 305: Pattern Matching for instanceof (Preview)
- JEP 343: Packaging Tool (Incubator)
- JEP 345: NUMA-Aware Memory Allocation for G1
- JEP 349: JFR Event Streaming
- JEP 352: Non-Volatile Mapped Byte Buffers
- JEP 358: Helpful NullPointerExceptions
- JEP 359: Records (Preview)
- JEP 361: Switch Expressions (Standard)
- JEP 362: Deprecate the Solaris and SPARC Ports
- JEP 363: Remove the Concurrent Mark Sweep (CMS) Garbage Collector
- JEP 364: ZGC on macOS
- JEP 365: ZGC on Windows
- JEP 366: Deprecate the ParallelScavenge + SerialOld GC Combination
- JEP 367: Remove the Pack200 Tools and API
- JEP 368: Text Blocks (Second Preview)
- JEP 370: Foreign-Memory Access API (Incubator)

JEP 305, *Pattern Matching for instanceof* simplifies the common case of an instanceof test being immediately followed by cast, replacing

```
if (obj instanceof String) {
    String s = (String)obj;
    System.out.println(s.length());
}
```

with

```
if (obj instanceof String s) {
    System.out.println(s.length());
}
```

JEP 359 *Records* allows easy creation of simple immutable Tuple-like classes.[328]

```
record Point(int x, int y) { }
Point p = new Point(3, 4);
```

```
System.out.println(p.x());
```

Java 14 updates

Java SE 15

Java SE 15	
Released	September 15, 2020 (4 years ago)
# of JEPs	14
Addition(s)	
Notable	Hidden classes, ZGC (garbage collector), Shenandoah (garbage collector), text blocks
Preview(s)	Sealed classes, pattern matching of instanceof, records
Incubating	Foreign-memory access
Removal(s)	
Notable	JavaScript engine, Solaris and SPARC ports
Support ended	
Public	March 2023[321]

JDK 15 was released on September 15, 2020. Java 15 adds e.g. support for multi-line string literals (aka Text Blocks). The Shenandoah and Z garbage collectors (latter sometimes abbreviated ZGC) are now ready for use in production (i.e. no longer marked experimental). Support for Oracle's Solaris operating system (and SPARC CPUs) is dropped (while still available in e.g. Java 11). The Nashorn JavaScript Engine is removed. Also removed some root CA certificates.

- JEP 339: Edwards-Curve Digital Signature Algorithm (EdDSA)
- JEP 360: Sealed Classes (Preview)

- JEP 371: Hidden Classes
- JEP 372: Remove the Nashorn JavaScript Engine
- JEP 373: Reimplement the Legacy DatagramSocket API
- JEP 374: Disable and Deprecate Biased Locking
- JEP 375: Pattern Matching for instanceof (Second Preview)
- JEP 377: ZGC: A Scalable Low-Latency Garbage Collector
- JEP 378: Text Blocks
- JEP 379: Shenandoah: A Low-Pause-Time Garbage Collector
- JEP 381: Remove the Solaris and SPARC Ports
- JEP 383: Foreign-Memory Access API (Second Incubator)
- JEP 384: Records (Second Preview)
- JEP 385: Deprecate RMI Activation for Removal

JEP 360 *Sealed Classes* adds sealed classes and interfaces that restrict which other classes or interfaces may extend or implement them. Only those classes specified in a permits clause may extend the class or interface.

```
package com.example.geometry;

public abstract sealed class Shape
    permits Circle, Rectangle, Square {...}
```

Together with records, sealed classes are sum types. They work well with other recent features like records, switch expressions, and pattern matching for instance-of. They all form part of a system for "Pattern matching in Java" first discussed by Gavin Bierman and Brian Goetz, in September 2018.[334]

By: ZAIN

Java 15 updates
Java SE 16

Java SE 16	
Released	March 16, 2021 (3 years ago)
# of JEPs	17
Addition(s)	

Notable	Windows/AArch64 Port, jpackager, pattern matching for instanceof, records
Preview(s)	Sealed classes
Incubating	Foreign linker, Foreign-memory access
	Support ended
Public	September 2021

JDK 16 was released on March 16, 2021. Java 16 removes Ahead-of-Time compilation (and Graal JIT) options.[340] The Java implementation itself was and is still written in C++, while as of Java 16, more recent C++14 (but still not e.g. C++17 or C++20) is allowed. The code was also moved to GitHub, dropping Mercurial as the source control system.

- JEP 338: Vector API (Incubator)
- JEP 347: Enable C++14 Language Features
- JEP 357: Migrate from Mercurial to Git
- JEP 369: Migrate to GitHub
- JEP 376: ZGC: Concurrent Thread-Stack Processing
- JEP 380: Unix-Domain Socket Channels
- JEP 386: Alpine Linux Port – not yet stable
- JEP 387: Elastic Metaspace
- JEP 388: Windows/AArch64 Port
- JEP 389: Foreign Linker API (Incubator)
- JEP 390: Warnings for Value-Based Classes
- JEP 392: Packaging Tool
- JEP 393: Foreign-Memory Access API (Third Incubator)
- JEP 394: Pattern Matching for instanceof
- JEP 395: Records
- JEP 396: Strongly Encapsulate JDK Internals by Default
- JEP 397: Sealed Classes (Second Preview)

Java 16 updates

Table of Java 16 updates show

Java SE 17

Java SE 17	
LTS version	
Released	September 14, 2021 (3 years ago)
# of JEPs	14
Addition(s)	
Notable	macOS/AArch64 Port, sealed classes
Preview(s)	Switch pattern matching
Incubating	Vector API, Foreign function & memory API
Removal(s)	
Notable	AOT compiler, RMI activation, strictfp keyword made obsolete (JEP 306)

JDK 17 was released in September 2021.[346] Java 17 is the 2nd long-term support (LTS) release since switching to the new 6-month release cadence (the first being Java 11).

- JEP 306: Restore Always-Strict Floating-Point Semantics
- JEP 356: Enhanced Pseudo-Random Number Generators
- JEP 382: New macOS Rendering Pipeline
- JEP 391: macOS/AArch64 Port
- JEP 398: Deprecate the Applet API for Removal
- JEP 403: Strongly Encapsulate JDK Internals

- JEP 406: Pattern Matching for switch (Preview)
- JEP 407: Remove RMI Activation
- JEP 409: Sealed Classes
- JEP 410: Remove the Experimental AOT and JIT Compiler
- JEP 411: Deprecate the Security Manager for Removal
- JEP 412: Foreign Function & Memory API (Incubator)
- JEP 414: Vector API (Second Incubator)
- JEP 415: Context-Specific Deserialization Filters

JEP 406 extends the pattern matching syntax used in instanceof operations to switch statements and expressions. It allows cases to be selected based on the type of the argument, null cases and refining patterns

```
Object o = ...;
return switch (o) {
      case null      -> "Null";
      case String s   -> "String %s".formatted(s);
      case Long l     -> "long %d".formatted(l);
      case Double d   -> "double %f".formatted(d);
      case Integer i && i > 0                    // refining patterns
                  -> "positive int %d".formatted(i);
      case Integer i && i == 0
                  -> "zero int %d".formatted(i);
      case Integer i && i < 0
                  -> "negative int %d".formatted(i);
      default        -> o.toString();
};
```

Java 17 updates

Java SE 18

Released	March 22, 2022 (2 years ago)
# of JEPs	9
	Addition(s)
Notable	UTF by default Javadoc code snippets
Preview(s)	Switch pattern matching

Incubating	Vector API Foreign function & memory API
	Removal(s)
Notable	Deprecated finalization for removal
	Support ended
Public	September 2022

JDK 18 was released on March 22, 2022.[371]

- JEP 400: UTF-8 by Default
- JEP 408: Simple Web Server
- JEP 413: Code Snippets in Java API Documentation
- JEP 416: Reimplement Core Reflection with Method Handles
- JEP 417: Vector API (Third Incubator)
- JEP 418: Internet-Address Resolution SPI
- JEP 419: Foreign Function & Memory API (Second Incubator)
- JEP 420: Pattern Matching for switch (Second Preview)
- JEP 421: Deprecate Finalization for Removal

Java 18 updates

[ ]

Table of Java 18 updates show

Java SE 19

[ ]

Java SE 19	
Released	September 20, 2022 (2 years ago)
# of JEPs	7

Addition(s)	
Preview(s)	Foreign function & memory API Switch pattern matching
Incubating	Vector API Structured concurrency
Support ended	
Public	March 2023

JDK 19 was released on 20 September 2022.[380]

- JEP 405: Record Patterns (Preview)
- JEP 422: Linux/RISC-V Port
- JEP 424: Foreign Function & Memory API (Preview)
- JEP 425: Virtual Threads (Preview)
- JEP 426: Vector API (Fourth Incubator)
- JEP 427: Pattern Matching for switch (Third Preview)
- JEP 428: Structured Concurrency (Incubator)

JEP 405 allows record patterns, extending the pattern matching capabilities of instanceof operators, and switch expressions, to include record patterns that explicitly refer to the components of the record.

```
record Rectangle(int x, int y, int w, int h) {}

int area(Object o) {
   if (o instanceof Rectangle(int x, int y, int w, int h)) {
      return w * h;
   }
   return 0;
}
```

Such patterns can include nested patterns, where the components of records are themselves records, allowing patterns to match more object graphs.

Java 19 updates

Java SE 20

Java SE 20

Released	March 21, 2023 (21 months ago)
# of JEPs	7
Addition(s)	
Incubating	Scoped values
Support ended	
Public	September 2023

Java 20 was released on 21 March 2023.[386] All JEPs were either incubators or previews.

- JEP 429: Scoped Values (Incubator)
- JEP 432: Record Patterns (Second Preview)
- JEP 433: Pattern Matching for switch (Fourth Preview)
- JEP 434: Foreign Function & Memory API (Second Preview)
- JEP 436: Virtual Threads (Second Preview)
- JEP 437: Structured Concurrency (Second Incubator)
- JEP 438: Vector API (Fifth Incubator)

Java 20 updates

Table of Java 20 updates show

Java SE 21

Java SE 21	
LTS version	
Released	September 19, 2023 (15 months ago)

# of JEPs	15
	Addition(s)
Notable	Record patterns, pattern matching for switch, virtual threads
Preview(s)	String templates, unnamed classes and main methods
Incubating	Vector API

Java 21 was released on 19 September 2023.[392] The 32-bit version of Java for Windows on x86 was deprecated for removal with this release. The following JEPs were added, including eight JEPs that graduated from the incubating and preview stages, compared to Java 20 which only had previewing and incubating JEPs. Java 21 introduces features first previewed in Java 17 (pattern matching for switch statements) and Java 19 (record patterns). All JEPs added with Java 21 include the following:

1. JEP 430: String Templates (Preview)
2. JEP 431: Sequenced Collections
3. JEP 439: Generational ZGC
4. JEP 440: Record Patterns
5. JEP 441: Pattern Matching for switch
6. JEP 442: Foreign Function & Memory API (Third Preview)
7. JEP 443: Unnamed Patterns and Variables (Preview)
8. JEP 444: Virtual Threads
9. JEP 445: Unnamed Classes and Instance Main Methods (Preview)
10. JEP 446: Scoped Values (Preview)
11. JEP 448: Vector API (Sixth Incubator)
12. JEP 449: Deprecate the Windows 32-bit x86 Port for Removal
13. JEP 451: Prepare to Disallow the Dynamic Loading of Agents
14. JEP 452: Key Encapsulation Mechanism API
15. JEP 453: Structured Concurrency (Preview)

JEP 445, previewing unnamed classes, allows for a barebones Main class without boilerplate code:

```
void main() {
    System.out.println("Hello, World!");
}
```

instead of :

```java
public class HelloWorld {
   public static void main(String  args) {
      System.out.println("Hello, World!");
   }
}
```

Java 21 updates

Java SE 22

Java SE 22	
Released	March 19, 2024 (9 months ago)
# of JEPs	12
Addition(s)	
Notable	Foreign function and memory API, unnamed variables and patterns
Preview(s)	Structured concurrency, string templates
Incubating	Vector API
Support ended	
Public	September 2024

Java 22 was released on March 19, 2024.[400][401] The following features, or JEPs, were added with this release:

1. JEP 423: Region Pinning for G1
2. JEP 447: Statements before super(...) (Preview)
3. JEP 454: Foreign Function & Memory API
4. JEP 456: Unnamed Variables & Patterns
5. JEP 457: Class-File API (Preview)
6. JEP 458: Launch Multi-File Source-Code Programs
7. JEP 459: String Templates (Second Preview)
8. JEP 460: Vector API (Seventh Incubator)
9. JEP 461: Stream Gatherers (Preview)

10. JEP 462: Structured Concurrency (Second Preview)
11. JEP 463: Implicitly Declared Classes and Instance Main Methods (Second Preview)
12. JEP 464: Scoped Values (Second Preview)

An API related to Java's threading implementation, java.lang.Thread.countStackFrames, was removed.[3][402]

Java SE 23

Java SE 23	
Released	September 17, 2024 (3 months ago)
# of JEPs	12
Addition(s)	
Notable	Markdown documentation comments
Preview(s)	Primitive types in Patterns, instanceof, and switch, Class-File API, Stream Gatherers, Module import declarations, Implicitly declared classes and instance main methods, structured concurrency, scoped values, flexible constructor bodies
Incubating	Vector API

Java 23 was released on September 17, 2024,[403][404][405] with the following JEPs:

1. JEP 455: Primitive Types in Patterns, instanceof, and switch (Preview)
2. JEP 466: Class-File API (Second Preview)
3. JEP 467: Markdown Documentation Comments
4. JEP 469: Vector API (Eighth Incubator)
5. JEP 473: Stream Gatherers (Second Preview)
6. JEP 471: Deprecate the Memory-Access Methods in sun.misc.Unsafe for Removal
7. JEP 474: ZGC: Generational Mode by Default
8. JEP 476: Module Import Declarations (Preview)
9. JEP 477: Implicitly Declared Classes and Instance Main Methods (Third Preview)
10. JEP 480: Structured Concurrency (Third Preview)

11. JEP 481: Scoped Values (Third Preview)
12. JEP 482: Flexible Constructor Bodies (Second Preview)

The String Templates preview feature was removed in Java 23 due to issues with the design of the feature.[406]

JAVA SE 24

Releasing	March 2025 (3 months' time)
# of JEPs	19

As of November 2024, the specification for Java 24 has not yet been finalized. Java 24 is scheduled for release in March 2025.[407]

The following JEPs have been proposed for this release but have not yet been confirmed and are still in their review period:[408]

1. JEP 496: Quantum-Resistant Module-Lattice-Based Key Encapsulation Mechanism
2. JEP 497: Quantum-Resistant Module-Lattice-Based Digital Signature Algorithm

The following JEPs have been targeted to this release of Java SE:[408]

1. JEP 404: Generational Shenandoah (Experimental)
2. JEP 450: Compact Object Headers (Experimental) (formerly known as *Project Lilliput*)
3. JEP 472: Prepare to Restrict the Use of JNI
4. JEP 475: Late Barrier Expansion for G1
5. JEP 478: Key Derivation Function API (Preview)
6. JEP 479: Remove the Windows 32-bit x86 Port
7. JEP 483: Ahead-of-Time Class Loading & Linking
8. JEP 484: Class-File API
9. JEP 485: Stream Gatherers
10. JEP 486: Permanently Disable the Security Manager
11. JEP 487: Scoped Values (Fourth Preview)
12. JEP 488: Primitive Types in Patterns, instanceof, and switch (Second Preview)
13. JEP 489: Vector API (Ninth Incubator)
14. JEP 490: ZGC: Remove the Non-Generational Mode
15. JEP 491: Synchronize Virtual Threads without Pinning
16. JEP 492: Flexible Constructor Bodies (Third Preview)
17. JEP 493: Linking Run-Time Images without JMODs
18. JEP 494: Module Import Declarations (Second Preview)
19. JEP 495: Simple Source Files and Instance Main Methods (Fourth Preview)

JAVA SE 25

Java SE 25	
Releasing	September 2025 (9 months' time)
# of JEPs	0

As of December 2024, the specification for Java 25 has not yet been finalized. Java 25 is scheduled for release in September 2025.[409]

Future features

- Project Valhalla: *Value classes*, whose objects lack identity, but can in certain cases get an improved memory layout (with less indirection), or have their allocation optimized away entirely.
- Project Panama:
  - *Improved interoperability with native code*, to enable Java source code to call functions and use data types from other languages, in a way that is easier and has better performance than today (this part of Project Panama is getting stabilized in Java 22 under JEP 454: Foreign Function & Memory API).
  - Vector API, a portable and relatively low-level abstraction layer for SIMD programming. Its stabilization is dependent on Project Valhalla.
- Project Lilliput: *Reduce the size of Java object headers.* First down to 64 bits, and then down to 32 bits.
- Reducing startup time and warm-up time (time to peak performance) in JIT mode:
  - Project CRaC enables making snapshots of whole JVM (together with the running application) and restoring it with necessary adjustments (reopening files, sockets, etc).
  - Project Leyden, among other things, will allow partial or (in the long term) full AOT compiling, reducing overall dynamism (by adopting so called "closed-world constraints") to reduce dynamic compiling overhead.
- Project Babylon aims to extend the Java language's reach to alternative programming models with an enhancement to its reflective programming abilities, called code reflection (i.e., reflection over code itself). The stated main goal is to run Java code on GPUs, with SQL and other programming models as secondary targets.

Implementations Of JAVA

The officially supported Java platform, first developed at Sun and now stewarded by Oracle, is Java SE. Releases are based on the OpenJDK project, a free and open-source project with an open development model. Other Java implementations exist, however—in part due to Java's early history as proprietary software. In contrast, some implementations were created to offer some benefits over the standard implementation, often the result of some area of academic or corporate-sponsored research. Many Linux distributions include builds of

OpenJDK through the IcedTea project started by Red Hat, which provides a more straightforward build and integration environment.

Visual J++ and the Microsoft Java Virtual Machine were created as incompatible implementations. After the *Sun v. Microsoft* lawsuit, Microsoft abandoned it and began work on the .NET platform. In 2021, Microsoft started distributing compatible "Microsoft Build of OpenJDK" for Java 11 first then also for Java 17. Their builds support not only Windows, but also Linux and macOS.

Other proprietary Java implementations are available, such as Azul's Zing. Azul offers certified open source OpenJDK builds under the Zulu moniker.

Prior to the release of OpenJDK, while Sun's implementation was still proprietary, the GNU Classpath project was created to provide a free and open-source implementation of the Java platform. Since the release of JDK 7, when OpenJDK became the official reference implementation, the original motivation for the GNU Classpath project almost completely disappeared, and its last release was in 2012.

The Apache Harmony project was started shortly before the release of OpenJDK. After Sun's initial source code release, the Harmony project continued, working to provide an implementation under a lax license, in contrast to the protective license chosen for OpenJDK. Google later developed Android and released it under a lax license. Android incorporated parts of the Harmony project, supplemented with Google's own Dalvik virtual machine and ART. Apache Harmony has since been retired, and Google has switched its Harmony components with equivalent ones from OpenJDK.

Both Jikes and Jikes RVM are open-source research projects that IBM developed.

Several other implementations exist that started as proprietary software but are now open source. IBM initially developed OpenJ9 as the proprietary J9[410] but has since relicensed the project and donated it to the Eclipse Foundation. JRockit is a proprietary implementation that was acquired by Oracle and incorporated into subsequent OpenJDK versions.