

# Java (programming language)

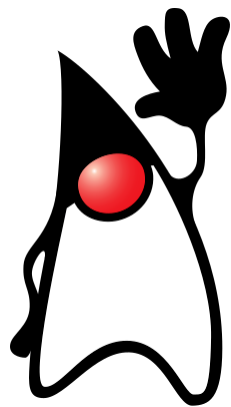
**Java** is a [high-level](#), [general-purpose](#), [memory-safe](#), [object-oriented programming language](#). It is intended to let [programmers write once, run anywhere \(WORA\)](#),<sup>[18]</sup> meaning that [compiled](#) Java code can run on all platforms that support Java without the need to recompile.<sup>[19]</sup> Java applications are typically compiled to [bytecode](#) that can run on any [Java virtual machine](#) (JVM) regardless of the underlying [computer architecture](#). The [syntax](#) of Java is similar to [C](#) and [C++](#), but has fewer [low-level](#) facilities than either of them. The Java runtime provides dynamic capabilities (such as [reflection](#) and runtime code modification) that are typically not available in traditional compiled languages.

Java gained popularity shortly after its release, and has been a popular programming language since then.<sup>[20]</sup> Java was the third most popular programming language in 2022 according to [GitHub](#).<sup>[21]</sup> Although still widely popular, there has been a gradual decline in use of Java in recent years with [other languages using JVM](#) gaining popularity.<sup>[22]</sup>

Java was designed by [James Gosling](#) at [Sun Microsystems](#). It was released in May 1995 as a core component of Sun's [Java platform](#). The original and [reference implementation](#) Java [compilers](#), virtual machines, and class libraries were released by Sun under [proprietary licenses](#). As of May 2007, in compliance with the specifications of the [Java Community Process](#), Sun had [relicensed](#) most of its Java technologies under the [GPL-2.0-only](#) license. [Oracle](#), which bought Sun in 2010, offers its own [HotSpot](#) Java Virtual Machine. However, the official [reference implementation](#) is the [OpenJDK](#) JVM, which is open-source software used by most developers and is the default JVM for almost all Linux distributions.

[Java 24](#) is the version current as of March 2025. Java 8, 11, 17, and 21 are [long-term support](#) versions still under maintenance.

# History



Duke, the Java mascot



James Gosling, the creator of Java, in 2008

James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991.<sup>[23]</sup> Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time.<sup>[24]</sup> The language was initially called *Oak* after an *oak* tree that stood outside Gosling's office. Later the project went by the name *Green* and was finally renamed *Java*, from *Java coffee*, a type of coffee from *Indonesia*.<sup>[25]</sup> Gosling designed Java with a C/C++-style syntax that system and application programmers would find familiar.<sup>[26]</sup>




Paradigm	Multi-paradigm: generic, object-oriented (class-based), functional, imperative, reflective, concurrent
Designed by	James Gosling
Developer	Oracle Corporation
First appeared	May 23, 1995 <sup>[1]</sup>
Stable release	Java SE 24 <sup>[2][3]</sup>  / 18 March 2025
Typing discipline	Static, strong, safe, nominative, manifest
Memory management	Automatic garbage collection
Filename extensions	.java, .class, .jar, .jmod, .war
Website	<div>oracle.com/java/ (<a href="http://oracle.com/java/">http://oracle.com/java/</a>)</div> <div>java.com (<a href="http://java.com">http://java.com</a>)</div>

Sun Microsystems released the first public implementation as Java 1.0 in 1996.<sup>[27]</sup> It promised [write once, run anywhere](#) (WORA) functionality, providing no-cost run-times on popular [platforms](#). Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major [web browsers](#) soon incorporated the ability to run [Java applets](#) within web pages, and Java quickly became popular. The Java 1.0 compiler was re-written [in Java](#) by [Arthur van Hoff](#) to comply strictly with the Java 1.0 language specification.<sup>[28]</sup> With the advent of Java 2 (released initially as J2SE 1.2 in December 1998 – 1999), new versions had multiple configurations built for different types of platforms. [J2EE](#) included technologies and APIs for enterprise applications typically run in server environments, while J2ME featured APIs optimized for mobile applications. The desktop version was renamed J2SE. In 2006, for marketing purposes, Sun renamed new J2 versions as [Java EE](#), [Java ME](#), and [Java SE](#), respectively.

In 1997, Sun Microsystems approached the [ISO/IEC JTC 1](#) standards body and later the [Ecma International](#) to formalize Java, but it soon withdrew from the process.<sup>[29][30][31]</sup> Java remains a [de facto standard](#), controlled through the [Java Community Process](#).<sup>[32]</sup> At one time, Sun made most of its Java implementations available without charge, despite their [proprietary software](#) status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System.

On November 13, 2006, Sun released much of its Java virtual machine (JVM) as [free and open-source software](#) (FOSS), under the terms of the [GPL-2.0-only](#) license. On May 8, 2007, Sun finished the process, making all of its JVM's core code available under [free software](#)/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.<sup>[33]</sup>

Sun's vice-president Rich Green said that Sun's ideal role with regard to Java was as an [evangelist](#).<sup>[34]</sup> Following [Oracle Corporation](#)'s acquisition of Sun Microsystems in 2009–10, Oracle has described itself as the steward of Java technology with a relentless commitment to fostering a community of participation and transparency.<sup>[35]</sup> This did not prevent Oracle from filing a lawsuit against Google shortly after that for using Java inside the [Android SDK](#) (see the [Android](#) section).

<div>dev.java (<a href="http://dev.java">http://dev.java</a>)</div>
<div>Influenced by</div>
<div>CLU,<sup>[4]</sup> Simula67,<sup>[4]</sup> Lisp,<sup>[4]</sup> Smalltalk,<sup>[4]</sup> Ada 83, C++,<sup>[5]</sup> C#,<sup>[6]</sup> Eiffel,<sup>[7]</sup> Mesa,<sup>[8]</sup> Modula-3,<sup>[9]</sup> Oberon,<sup>[10]</sup> Objective-C,<sup>[11]</sup> UCSD Pascal,<sup>[12][13]</sup> Object Pascal<sup>[14]</sup></div>
<div>Influenced</div>
<div>Ada 2005, ArkTS, BeanShell, C#, Chapel,<sup>[15]</sup> Clojure, ECMAScript, Fantom, Gambas,<sup>[16]</sup> Groovy, Hack,<sup>[17]</sup> Haxe, J#, JavaScript, JS++, Kotlin, PHP, Python, Scala, Seed7, Vala</div>
<div> <a href="#">Java Programming</a> at Wikibooks</div>

On April 2, 2010, James Gosling resigned from [Oracle](#).<sup>[36]</sup>

In January 2016, Oracle announced that Java run-time environments based on JDK 9 will discontinue the browser plugin.<sup>[37]</sup>

Java software runs on most devices from laptops to [data centers](#), [game consoles](#) to scientific [supercomputers](#).<sup>[38]</sup>

[Oracle](#) (and others) highly recommend uninstalling outdated and unsupported versions of Java, due to unresolved security issues in older versions.<sup>[39]</sup>

## Principles

There were five primary goals in creating the Java language:<sup>[19]</sup>

1. It must be simple, [object-oriented](#), and familiar.
2. It must be [robust](#) and secure.
3. It must be architecture-neutral and portable.
4. It must execute with high performance.
5. It must be [interpreted](#), [threaded](#), and [dynamic](#).

## Versions

As of November 2024, Java 8, 11, 17, and 21 are supported as [long-term support](#) (LTS) versions, with Java 25, releasing in September 2025, as the next scheduled LTS version.<sup>[40]</sup>

Oracle released the last zero-cost public update for the [legacy](#) version [Java 8](#) LTS in January 2019 for commercial use, although it will otherwise still support Java 8 with public updates for personal use indefinitely. Other vendors such as [Adoptium](#) continue to offer free builds of OpenJDK's long-term support (LTS) versions. These builds may include additional security patches and bug fixes.<sup>[41]</sup>

Major release versions of Java, along with their release dates:

Version	Date
JDK <a href="#">Beta</a>	1995
JDK 1.0	January 23, 1996 <sup>[42]</sup>
JDK 1.1	February 19, 1997
J2SE 1.2	December 8, 1998
J2SE 1.3	May 8, 2000
J2SE 1.4	February 6, 2002
J2SE 5.0	September 30, 2004
Java SE 6	December 11, 2006
Java SE 7	July 28, 2011
Java SE 8 (LTS)	March 18, 2014
Java SE 9	September 21, 2017
Java SE 10	March 20, 2018
Java SE 11 (LTS)	September 25, 2018 <sup>[43]</sup>
Java SE 12	March 19, 2019
Java SE 13	September 17, 2019
Java SE 14	March 17, 2020
Java SE 15	September 15, 2020 <sup>[44]</sup>
Java SE 16	March 16, 2021
Java SE 17 (LTS)	September 14, 2021
Java SE 18	March 22, 2022
Java SE 19	September 20, 2022
Java SE 20	March 21, 2023
Java SE 21 (LTS)	September 19, 2023 <sup>[45]</sup>
Java SE 22	March 19, 2024
Java SE 23	September 17, 2024
Java SE 24	18 March 2025 <sup>[46]</sup>

## Editions

---

Sun has defined and supports four editions of Java targeting different application environments and segmented many of its [APIs](#) so that they belong to one of the platforms. The platforms are:

- [Java Card](#) for smart-cards.<sup>[47]</sup>
- [Java Platform, Micro Edition](#) (Java ME) – targeting environments with limited resources.<sup>[48]</sup>

- [Java Platform, Standard Edition](#) (Java SE) – targeting workstation environments.<sup>[49]</sup>
- [Java Platform, Enterprise Edition](#) (Java EE) – targeting large distributed enterprise or Internet environments.<sup>[50]</sup>

The [classes](#) in the Java APIs are organized into separate groups called [packages](#). Each package contains a set of related [interfaces](#), classes, subpackages and [exceptions](#).

Sun also provided an edition called [Personal Java](#) that has been superseded by later, standards-based Java ME configuration-profile pairings.

## Execution system

---

### Java JVM and bytecode

One design goal of Java is [portability](#), which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate run time support. This is achieved by compiling the Java language code to an intermediate representation called [Java bytecode](#), instead of directly to architecture-specific [machine code](#). Java bytecode instructions are analogous to machine code, but they are intended to be executed by a [virtual machine](#) (VM) written specifically for the host hardware. [End-users](#) commonly use a [Java Runtime Environment](#) (JRE) installed on their device for standalone Java applications or a web browser for [Java applets](#).

Standard libraries provide a generic way to access host-specific features such as graphics, [threading](#), and [networking](#).

The use of universal bytecode makes porting simple. However, the overhead of [interpreting](#) bytecode into machine instructions made interpreted programs almost always run more slowly than native [executables](#). [Just-in-time](#) (JIT) compilers that compile byte-codes to machine code during runtime were introduced from an early stage. Java's Hotspot compiler is actually two compilers in one; and with [GraalVM](#) (included in e.g. Java 11, but removed as of Java 16) allowing [tiered compilation](#).<sup>[51]</sup> Java itself is platform-independent and is adapted to the particular platform it is to run on by a [Java virtual machine](#) (JVM), which translates the [Java bytecode](#) into the platform's machine language.<sup>[52]</sup>

### Performance

Programs written in Java have a reputation for being slower and requiring more memory than those written in [C++](#).<sup>[53][54]</sup> However, Java programs' execution speed improved significantly with



the introduction of [just-in-time compilation](#) in 1997/1998 for [Java 1.1](#),<sup>[55]</sup> the addition of language features supporting better code analysis (such as inner classes, the `StringBuilder` class, optional assertions, etc.), and optimizations in the Java virtual machine, such as [HotSpot](#) becoming Sun's default JVM in 2000. With Java 1.5, the performance was improved with the addition of the `java.util.concurrent` package, including [lock-free](#) implementations of the [ConcurrentMaps](#) and other multi-core collections, and it was improved further with Java 1.6.

## Non-JVM

Some platforms offer direct hardware support for Java; there are micro controllers that can run Java bytecode in hardware instead of a software Java virtual machine,<sup>[56]</sup> and some [ARM](#)-based processors could have hardware support for executing Java bytecode through their [Jazelle](#) option, though support has mostly been dropped in current implementations of ARM.

## Automatic memory management

Java uses an [automatic garbage collector](#) to manage memory in the [object lifecycle](#). The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the [unreachable memory](#) becomes eligible to be freed automatically by the garbage collector. Something similar to a [memory leak](#) may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use.<sup>[57]</sup> If methods for a non-existent object are called, a [null pointer](#) exception is thrown.<sup>[58][59]</sup>

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the [stack](#) or explicitly allocated and deallocated from the [heap](#). In the latter case, the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a [memory leak](#) occurs.<sup>[57]</sup> If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable or crash. This can be partially remedied by the use of [smart pointers](#), but these add overhead and complexity. Garbage collection does not prevent [logical memory](#) leaks, i.e. those where the memory is still referenced but never used.<sup>[57]</sup>

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style [pointer arithmetic](#), where object addresses can be arithmetically manipulated (e.g. by adding or subtracting an offset). This allows the garbage collector to relocate referenced objects and ensures type safety and security.

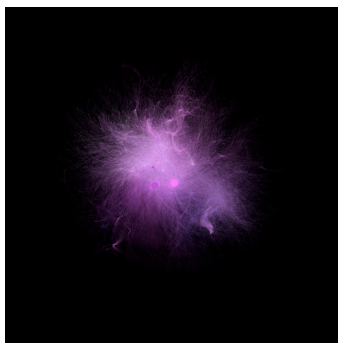
As in C++ and some other object-oriented languages, variables of Java's [primitive data types](#) are either stored directly in fields (for objects) or on the [stack](#) (for methods) rather than on the heap, as is commonly true for non-primitive data types (but see [escape analysis](#)). This was a conscious decision by Java's designers for performance reasons.

Java contains multiple types of garbage collectors. Since Java 9, HotSpot uses the [Garbage First Garbage Collector](#) (G1GC) as the default.<sup>[60]</sup> However, there are also several other garbage collectors that can be used to manage the heap, such as the Z Garbage Collector (ZGC) introduced in Java 11, and Shenandoah GC, introduced in Java 12 but unavailable in Oracle-produced OpenJDK builds. Shenandoah is instead available in third-party builds of OpenJDK, such as [Eclipse Temurin](#). For most applications in Java, G1GC is sufficient. In prior versions of Java, such as Java 8, the [Parallel Garbage Collector](#) (<https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/parallel.html>) was used as the default garbage collector.

Having solved the memory management problem does not relieve the programmer of the burden of handling properly other kinds of resources, like network or database connections, file handles, etc., especially in the presence of exceptions.

## Syntax

---



This dependency graph of the Java Core classes was created with [jdeps](#) and [Gephi](#).



The syntax of Java is largely influenced by [C++](#) and [C](#). Unlike C++, which combines the syntax for structured, generic, and object-oriented programming, Java was built almost exclusively as an object-oriented language.<sup>[19]</sup> All code is written inside classes, and every data item is an object, with the exception of the primitive data types, (i.e. integers, floating-point numbers, [boolean values](#), and characters), which are not objects for performance reasons. Java reuses some popular aspects of C++ (such as the `printf` method).

Unlike C++, Java does not support [operator overloading](#)<sup>[61]</sup> or [multiple inheritance](#) for classes, though multiple inheritance is supported for [interfaces](#).<sup>[62]</sup>

Java uses [comments](#) similar to those of C++. There are three different styles of comments: a single line style marked with two slashes (`//`), a multiple line style opened with `/*` and closed with `*/`, and the [Javadoc](#) commenting style opened with `/**` and closed with `*/`. The Javadoc style of commenting allows the user to run the Javadoc executable to create documentation for the program and can be read by some [integrated development environments](#) (IDEs) such as [Eclipse](#) to allow developers to access documentation within the IDE.

## Hello world

The following is a simple example of a ["Hello, World!" program](#) that writes a message to the [standard output](#):

```
1 public class Example {
2     public static void main(String[] args) {
3         System.out.println("Hello World!");
4     }
5 }
```

## Special classes

---

### Applet

Java applets were programs embedded in other applications, mainly in web pages displayed in web browsers. The Java applet API was deprecated with the release of Java 9 in 2017.<sup>[63][64]</sup>

## Servlet

[Java servlet](#) technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. Servlets are [server-side](#) Java EE components that generate responses to requests from [clients](#). Most of the time, this means generating [HTML](#) pages in response to [HTTP](#) requests, although there are a number of other standard servlet classes available, for example for [WebSocket](#) communication.

The Java servlet API has to some extent been superseded (but still used under the hood) by two standard Java technologies for web services:

- the [Java API for RESTful Web Services](#) (JAX-RS 2.0) useful for AJAX, JSON and REST services, and
- the [Java API for XML Web Services](#) (JAX-WS) useful for [SOAP Web Services](#).

Typical implementations of these APIs on Application Servers or Servlet Containers use a standard servlet for handling all interactions with the [HTTP](#) requests and responses that delegate to the web service methods for the actual business logic.

## JavaServer Pages

JavaServer Pages ([JSP](#)) are [server-side](#) Java EE components that generate responses, typically [HTML](#) pages, to [HTTP](#) requests from [clients](#). JSPs embed Java code in an HTML page by using the special [delimiters](#) `<%` and `%>`. A JSP is compiled to a Java *servlet*, a Java application in its own right, the first time it is accessed. After that, the generated servlet creates the response.<sup>[65]</sup>

## Swing application

[Swing](#) is a graphical user interface library for the Java SE platform. It is possible to specify a different look and feel through the [pluggable look and feel](#) system of Swing. Clones of [Windows](#), [GTK+](#), and [Motif](#) are supplied by Sun. [Apple](#) also provides an [Aqua](#) look and feel for [macOS](#). Where prior implementations of these looks and feels may have been considered lacking, Swing in Java SE 6 addresses this problem by using more native [GUI widget](#) drawing routines of the underlying platforms.<sup>[66]</sup>

## JavaFX application

JavaFX is a [software platform](#) for creating and delivering [desktop applications](#), as well as [rich web applications](#) that can run across a wide variety of devices. JavaFX is intended to replace [Swing](#) as the standard [graphical user interface](#) (GUI) library for [Java SE](#), but since JDK 11 JavaFX has not been in the core JDK and instead in a separate module.<sup>[67]</sup> JavaFX has support for [desktop computers](#) and [web browsers](#) on [Microsoft Windows](#), [Linux](#), and [macOS](#). JavaFX does not have support for native OS look and feels.<sup>[68]</sup>

## Generics

In 2004, [generics](#) were added to the Java language, as part of J2SE 5.0. Prior to the introduction of generics, each variable declaration had to be of a specific type. For container classes, for example, this is a problem because there is no easy way to create a container that accepts only specific types of objects. Either the container operates on all subtypes of a class or interface, usually `Object`, or a different container class has to be created for each contained class. Generics allow compile-time type checking without having to create many container classes, each containing almost identical code. In addition to enabling more efficient code, certain runtime exceptions are prevented from occurring, by issuing compile-time errors. If Java prevented all runtime type errors (`ClassCastException`s) from occurring, it would be [type safe](#).

In 2016, the type system of Java was proven [unsound](#) in that it is possible to use generics to construct classes and methods that allow assignment of an instance of one class to a variable of another unrelated class. Such code is accepted by the compiler, but fails at run time with a class cast exception.<sup>[69]</sup>

## Criticism

Criticisms directed at Java include the implementation of generics,<sup>[70]</sup> speed,<sup>[53]</sup> the handling of unsigned numbers,<sup>[71]</sup> the implementation of floating-point arithmetic,<sup>[72]</sup> and a history of security vulnerabilities in the primary Java VM implementation [HotSpot](#).<sup>[73]</sup> Developers have criticized the complexity and verbosity of the Java Persistence API (JPA), a standard part of Java EE. This has led to increased adoption of higher-level abstractions like Spring Data JPA, which aims to simplify database operations and reduce boilerplate code. The growing popularity of such frameworks suggests limitations in the standard JPA implementation's ease-of-use for modern Java development.<sup>[74]</sup>

# Class libraries

---

The [Java Class Library](#) is the [standard library](#), developed to support application development in Java. It is controlled by [Oracle](#) in cooperation with others through the [Java Community Process](#) program.<sup>[75]</sup> Companies or individuals participating in this process can influence the design and development of the APIs. This process has been a subject of controversy during the 2010s.<sup>[76]</sup> The class library contains features such as:

- The core libraries, which include:
  - [Input/output](#) (I/O or IO)<sup>[77]</sup> and [non-blocking I/O](#) (NIO), or IO/NIO<sup>[78]</sup>
  - [Networking](#)<sup>[79]</sup> (new [user agent](#) (HTTP client) since Java 11<sup>[80]</sup>)
  - [Reflective programming](#) (reflection)
  - [Concurrent computing](#) (concurrency)<sup>[77]</sup>
  - [Generics](#)
  - Scripting, Compiler
  - [Functional programming](#) ([Lambda](#), streaming)
  - [Collection libraries](#) that implement [data structures](#) such as [lists](#), [dictionaries](#), [trees](#), [sets](#), [queues](#) and [double-ended queue](#), or [stacks](#)<sup>[81]</sup>
  - [XML Processing](#) (Parsing, Transforming, Validating) libraries
  - [Security](#)<sup>[82]</sup>
  - [Internationalization and localization](#) libraries<sup>[83]</sup>
- The integration libraries, which allow the application writer to communicate with external systems. These libraries include:
  - The [Java Database Connectivity](#) (JDBC) [API](#) for database access
  - [Java Naming and Directory Interface](#) (JNDI) for lookup and discovery
  - [Java remote method invocation](#) (RMI) and [Common Object Request Broker Architecture](#) (CORBA) for distributed application development
  - [Java Management Extensions](#) (JMX) for managing and monitoring applications
- [User interface](#) libraries, which include:
  - The (heavyweight, or [native](#)) [Abstract Window Toolkit](#) (AWT), which provides [GUI](#) components, the means for laying out those components and the means for handling

events from those components

- The (lightweight) [Swing](#) libraries, which are built on AWT but provide (non-native) implementations of the AWT widgetry
- APIs for audio capture, processing, and playback
- [JavaFX](#)
- A platform dependent implementation of the Java virtual machine that is the means by which the bytecodes of the Java libraries and third-party applications are executed
- Plugins, which enable [applets](#) to be run in web browsers
- [Java Web Start](#), which allows Java applications to be efficiently distributed to [end users](#) across the Internet
- Licensing and documentation

## Documentation

---

Javadoc is a comprehensive documentation system, created by [Sun Microsystems](#). It provides developers with an organized system for documenting their code. Javadoc comments have an extra asterisk at the beginning, i.e. the delimiters are `/**` and `*/`, whereas the normal multi-line comments in Java are delimited by `/*` and `*/`, and single-line comments start with `//`.<sup>[84]</sup>

## Implementations

---

[Oracle Corporation](#) owns the official implementation of the Java SE platform, due to its acquisition of [Sun Microsystems](#) on January 27, 2010. This implementation is based on the original implementation of Java by Sun. The Oracle implementation is available for [Windows](#), [macOS](#), [Linux](#), and [Solaris](#). Because Java lacks any formal standardization recognized by [Ecma International](#), ISO/IEC, ANSI, or other third-party standards organizations, the Oracle implementation is the [de facto standard](#).

The Oracle implementation is packaged into two different distributions: The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs and is intended for end users, and the [Java Development Kit](#) (JDK), which is intended for software developers and includes development tools such as the [Java compiler](#), [Javadoc](#), [Jar](#), and a [debugger](#). Oracle has also released [GraalVM](#), a high performance Java dynamic compiler and interpreter.

[OpenJDK](#) is another Java SE implementation that is licensed under the GNU GPL. The implementation started when Sun began releasing the Java source code under the GPL. As of Java SE 7, OpenJDK is the official Java reference implementation.

The goal of Java is to make all implementations of Java compatible. Historically, Sun's trademark license for usage of the Java brand insists that all implementations be *compatible*. This resulted in a legal dispute with [Microsoft](#) after Sun claimed that the Microsoft implementation did not support [Java remote method invocation](#) (RMI) or [Java Native Interface](#) (JNI) and had added platform-specific features of their own. Sun sued in 1997, and, in 2001, won a settlement of US\$20 million, as well as a court order enforcing the terms of the license from Sun.<sup>[85]</sup> As a result, Microsoft no longer ships Java with [Windows](#).

Platform-independent Java is essential to [Java EE](#), and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications.

## Use outside the Java platform

---

The Java programming language requires the presence of a software platform in order for compiled programs to be executed.

Oracle supplies the [Java platform](#) for use with Java. The [Android SDK](#) is an alternative software platform, used primarily for developing [Android applications](#) with its own GUI system.

## Android

The Java language is a key pillar in [Android](#), an [open source mobile operating system](#). Although Android, built on the [Linux kernel](#), is written largely in C, the [Android SDK](#) uses the Java language as the basis for Android applications but does not use any of its standard GUI, SE, ME or other established Java standards.<sup>[86]</sup> The bytecode language supported by the Android SDK is incompatible with Java bytecode and runs on its own virtual machine, optimized for low-memory devices such as [smartphones](#) and [tablet computers](#). Depending on the Android version, the bytecode is either interpreted by the [Dalvik virtual machine](#) or compiled into native code by the [Android Runtime](#).

Android does not provide the full Java SE standard library, although the Android SDK does include an independent implementation of a large subset of it. It supports Java 6 and some Java 7 features, offering an implementation compatible with the standard library ([Apache Harmony](#)).



## Controversy

The use of Java-related technology in Android led to a legal dispute between Oracle and Google. On May 7, 2012, a San Francisco jury found that if APIs could be copyrighted, then Google had infringed Oracle's copyrights by the use of Java in Android devices.<sup>[87]</sup> District Judge [William Alsup](#) ruled on May 31, 2012, that APIs cannot be copyrighted,<sup>[88]</sup> but this was reversed by the United States Court of Appeals for the Federal Circuit in May 2014.<sup>[89]</sup> On May 26, 2016, the district court decided in favor of Google, ruling the copyright infringement of the Java API in Android constitutes fair use.<sup>[90]</sup> In March 2018, this ruling was overturned by the Appeals Court, which sent down the case of determining the damages to federal court in San Francisco.<sup>[91]</sup> Google filed a petition for [writ of certiorari](#) with the [Supreme Court of the United States](#) in January 2019 to challenge the two rulings that were made by the Appeals Court in Oracle's favor.<sup>[92]</sup> On April 5, 2021, the Court ruled 6–2 in Google's favor, that its use of Java APIs should be considered [fair use](#). However, the court refused to rule on the copyrightability of APIs, choosing instead to determine their ruling by considering Java's API copyrightable "purely for argument's sake."<sup>[93]</sup>

## See also

- [C#](#)
- [C++](#)
- [Dalvik](#), used in old Android versions, replaced by non-JIT [Android Runtime](#)
- [Java Heterogeneous Distributed Computing](#)
- [List of Java APIs](#)
- [List of Java frameworks](#)
- [List of JVM languages](#)
- [List of Java virtual machines](#)
- [Comparison of C# and Java](#)
- [Comparison of Java and C++](#)
- [Comparison of programming languages](#)

 **Computer  
programming portal**

## References

- Binstock, Andrew (May 20, 2015). "Java's 20 Years of Innovation" (<https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/>) . *Forbes*. Archived (<https://web.archive.org/web/20150520140000/http://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/>)

- [e.org/web/20160314102242/http://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/](http://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/)) from the original on March 14, 2016. Retrieved March 18, 2016.
2. "Oracle Releases Java 24" (<https://www.oracle.com/news/announcement/oracle-releases-java-24-2025-03-18/>) .
  3. "JDK 24" (<https://openjdk.org/projects/jdk/24/>) .
  4. Barbara Liskov with John Guttag (2000). *Program Development in Java – Abstraction, Specification, and Object-Oriented Design*. USA, Addison Wesley. ISBN 978-0-201-65768-5.
  5. Chaudhary, Harry H. (July 28, 2014). "Cracking The Java Programming Interview :: 2000+ Java Interview Que/Ans" (<https://books.google.com/books?id=0rUtBAAAQBAJ&pg=PAPA133>) . Archived (<https://web.archive.org/web/20230929040943/https://books.google.com/books?id=0rUtBAAAQBAJ&pg=PAPA133#v=onepage&q&f=false>) from the original on September 29, 2023. Retrieved May 29, 2016.
  6. Java 5.0 added several new language features (the [enhanced for loop](#), [autoboxing](#), [varargs](#) and [annotations](#)), after they were introduced in the similar (and competing) [C#](#) language. [1] (<http://www.barrycornelius.com/papers/java5/>) Archived (<https://web.archive.org/web/20110319065438/http://www.barrycornelius.com/papers/java5/>) March 19, 2011, at the Wayback Machine [2] (<http://www.levenez.com/lang/>) Archived (<https://web.archive.org/web/20060107162045/http://www.levenez.com/lang/>) January 7, 2006, at the Wayback Machine
  7. Gosling, James; McGilton, Henry (May 1996). "The Java Language Environment" (<https://www.oracle.com/technetwork/java/langenv-140151.html>) . Archived (<https://web.archive.org/web/20140506214653/http://www.oracle.com/technetwork/java/langenv-140151.html>) from the original on May 6, 2014. Retrieved May 6, 2014.
  8. Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad. "The Java Language Specification, 2nd Edition" ([https://java.sun.com/docs/books/jls/second\\_edition/html/intro.doc.html#237601](https://java.sun.com/docs/books/jls/second_edition/html/intro.doc.html#237601)) . Archived ([https://web.archive.org/web/20110805051057/http://java.sun.com/docs/books/jls/second\\_edition/html/intro.doc.html#237601](https://web.archive.org/web/20110805051057/http://java.sun.com/docs/books/jls/second_edition/html/intro.doc.html#237601)) from the original on August 5, 2011. Retrieved February 8, 2008.
  9. "The A-Z of Programming Languages: Modula-3" (<https://web.archive.org/web/20090105145818/http://www.computerworld.com.au/index.php/id%3B1422447371%3Bpp%3B3%3Bfp%3B4194304%3Bfpid%3B1>) . Computerworld. Archived from the original (<http://www.computerworld.com.au/index.php/id;1422447371;pp;3;fp;4194304;fpid;1>) on January 5, 2009. Retrieved June 9, 2010.

10. [Niklaus Wirth](#) stated on a number of public occasions, e.g. in a lecture at the Polytechnic Museum, Moscow in September 2005 (several independent first-hand accounts in Russian exist, e.g. one with an audio recording: Filippova, Elena (September 22, 2005). "[Niklaus Wirth's lecture at the Polytechnic Museum in Moscow](#)" (<http://www.delphikingdom.com/asp/viewitem.asp?catalogid=1155>) . Archived (<https://web.archive.org/web/20201201054813/http://www.delphikingdom.com/asp/viewitem.asp?catalogid=1155>) from the original on December 1, 2020. Retrieved November 20, 2011.), that the Sun Java design team licensed the Oberon compiler sources a number of years prior to the release of Java and examined it: a (relative) compactness, type safety, garbage collection, no multiple inheritance for classes – all these key overall design features are shared by Java and Oberon.
11. [Patrick Naughton](#) cites [Objective-C](#) as a strong influence on the design of the Java programming language, stating that notable direct derivatives include Java interfaces (derived from Objective-C's [protocol](#)) and primitive wrapper classes. [\[3\]](#) (<http://cs.gmu.edu/~sean/stuff/java-objc.html>) Archived (<https://web.archive.org/web/20110713014816/http://cs.gmu.edu/~sean/stuff/java-objc.html>) July 13, 2011, at the [Wayback Machine](#)
12. TechMetrix Research (1999). "[History of Java](#)" (<https://web.archive.org/web/20101229090912/http://www.fscript.org/prof/javapassport.pdf>) (PDF). *Java Application Servers Report*. Archived from the original (<http://www.fscript.org/prof/javapassport.pdf>) (PDF) on December 29, 2010. "The project went ahead under the name *green* and the language was based on an old model of [UCSD Pascal](#), which makes it possible to generate interpretive code."
13. "[A Conversation with James Gosling](#)" (<http://queue.acm.org/detail.cfm?id=1017013>) . *ACM Queue*. Vol. 2, no. 5. Association for Computing Machinery. August 31, 2004. Archived (<https://web.archive.org/web/20150716194245/http://queue.acm.org/detail.cfm?id=1017013>) from the original on July 16, 2015. Retrieved June 9, 2010.
14. The Java Language Team. [About Microsoft's 'Delegates'](#) (<https://web.archive.org/web/20120627043929/http://java.sun.com/docs/white/delegates.html>) (White Paper). JavaSoft, Sun Microsystems, Inc. Archived from the original (<http://java.sun.com/docs/white/delegates.html>) on June 27, 2012. "In the summer of 1996, Sun was designing the precursor to what is now the event model of the AWT and the JavaBeans component architecture. Borland contributed greatly to this process. We looked very carefully at Delphi Object Pascal and built a working prototype of bound method references in order to understand their interaction with the Java programming language and its APIs."

15. "Chapel spec (Acknowledgements)" (<http://chapel.cray.com/spec/spec-0.98.pdf>) (PDF). Cray Inc. October 1, 2015. Archived (<https://web.archive.org/web/20160205114946/http://chapel.cray.com/spec/spec-0.98.pdf>) (PDF) from the original on February 5, 2016. Retrieved January 14, 2016.
16. "Gambas Documentation Introduction" (<http://gambaswiki.org/wiki/doc/intro?nh&l=en>) . Gambas Website. Archived (<https://web.archive.org/web/20171009041815/http://gambaswiki.org/wiki/doc/intro?nh&l=en>) from the original on October 9, 2017. Retrieved October 9, 2017.
17. "Facebook Q&A: Hack brings static typing to PHP world" (<http://www.infoworld.com/article/2610885/facebook-q-a--hack-brings-static-typing-to-php-world.html>) . *InfoWorld*. March 26, 2014. Archived (<https://web.archive.org/web/20150213220946/http://www.infoworld.com/article/2610885/facebook-q-a--hack-brings-static-typing-to-php-world.html>) from the original on February 13, 2015. Retrieved January 11, 2015.
18. "Write once, run anywhere?" (<http://www.computerweekly.com/Articles/2002/05/02/186793/write-once-run-anywhere.htm>) . Computer Weekly. May 2, 2002. Archived (<https://web.archive.org/web/20210813193857/https://www.computerweekly.com/feature/Write-once-run-anywhere>) from the original on August 13, 2021. Retrieved July 27, 2009.
19. "1.2 Design Goals of the Java Programming Language" (<https://www.oracle.com/technetwork/java/intro-141325.html>) . Oracle. January 1, 1999. Archived (<https://web.archive.org/web/20130123204103/http://www.oracle.com/technetwork/java/intro-141325.html>) from the original on January 23, 2013. Retrieved January 14, 2013.
20. Melanson, Mike (August 9, 2022). "Don't call it a comeback: Why Java is still champ" (<https://github.com/readme/featured/java-programming-language>) . *GitHub*. Archived (<https://web.archive.org/web/20230825195416/https://github.com/readme/featured/java-programming-language>) from the original on August 25, 2023. Retrieved October 15, 2023.
21. "The top programming languages" (<https://octoverse.github.com/2022/top-programming-languages>) . *The State of the Octoverse*. GitHub. Archived (<https://web.archive.org/web/20230802203718/https://octoverse.github.com/2022/top-programming-languages>) from the original on August 2, 2023. Retrieved October 15, 2023.
22. McMillan, Robert (August 1, 2013). "Is Java Losing Its Mojo?" (<https://www.wired.com/2013/01/java-no-longer-a-favorite/>) . *Wired*. Archived (<https://web.archive.org/web/20170215115409/https://www.wired.com/2013/01/java-no-longer-a-favorite/>) from the original on February 15, 2017. Retrieved October 15, 2023.

23. Byous, Jon (c. 1998). "Java technology: The early years" (<https://web.archive.org/web/20050420081440/http://java.sun.com/features/1998/05/birthday.html>) . *Sun Developer Network*. Sun Microsystems. Archived from the original (<https://java.sun.com/features/1998/05/birthday.html>) on April 20, 2005. Retrieved April 22, 2005.
24. Object-oriented programming "The History of Java Technology" (<https://web.archive.org/web/20100210225651/http://www.java.com/en/javahistory/>) . *Sun Developer Network*. c. 1995. Archived from the original (<http://www.java.com/en/javahistory/>) on February 10, 2010. Retrieved April 30, 2010.
25. Murphy, Kieron (October 4, 1996). "So why did they decide to call it Java?" (<https://www.infoworld.com/article/2077265/so-why-did-they-decide-to-call-it-java-.html>) . *JavaWorld*. Archived (<https://web.archive.org/web/20200713234202/https://www.infoworld.com/article/2077265/so-why-did-they-decide-to-call-it-java-.html>) from the original on July 13, 2020. Retrieved July 13, 2020.
26. Kabutz, Heinz; *Once Upon an Oak* (<http://www.artima.com/weblogs/viewpost.jsp?thread=7555>) Archived (<https://web.archive.org/web/20070413072630/http://www.artima.com/weblogs/viewpost.jsp?thread=7555>) April 13, 2007, at the *Wayback Machine*. Artima. Retrieved April 29, 2007.
27. "JAVASOFT SHIPS JAVA 1.0" (<https://web.archive.org/web/20070310235103/http://www.sun.com/smi/Press/sunflash/1996-01/sunflash.960123.10561.xml>) . Archived from the original (<http://www.sun.com/smi/Press/sunflash/1996-01/sunflash.960123.10561.xml>) on March 10, 2007. Retrieved May 13, 2018.
28. *Object-oriented Programming with Java: Essentials and Applications* (<https://books.google.com/books?id=rXGMFYXFDwMC>) . Tata McGraw-Hill Education. p. 34.
29. "JSG – Java Study Group" (<http://www.open-std.org/JTC1/SC22/JSG/>) . *open-std.org*. Archived (<https://web.archive.org/web/20060825082008/http://www.open-std.org/JTC1/SC22/JSG/>) from the original on August 25, 2006. Retrieved August 2, 2006.
30. "Why Java Was – Not – Standardized Twice" (<http://www.computer.org/csdl/proceedings/hicss/2001/0981/05/09815015.pdf>) (PDF). Archived (<https://web.archive.org/web/20140113101235/http://www.computer.org/csdl/proceedings/hicss/2001/0981/05/09815015.pdf>) (PDF) from the original on January 13, 2014. Retrieved June 3, 2018.
31. "What is ECMA—and why Microsoft cares" (<https://www.zdnet.com/news/what-is-ecma-and-why-microsoft-cares/298821>) . *ZDNet*. Archived (<https://web.archive.org/web/20140506215226/http://www.zdnet.com/news/what-is-ecma-and-why-microsoft-cares/298821>) from the original on May 6, 2014. Retrieved May 6, 2014.



32. "Java Community Process website" (<http://www.jcp.org/en/home/index>) . Jcp.org. May 24, 2010. Archived (<https://web.archive.org/web/20060808070528/http://www.jcp.org/en/home/index>) from the original on August 8, 2006. Retrieved June 9, 2010.
33. "JAVAONE: Sun – The bulk of Java is open sourced" (<http://grnlight.net/index.php/programming-articles/115-javaone-sun-the-bulk-of-java-is-open-sourced>) . GrnLight.net. Archived (<https://web.archive.org/web/20140527220942/http://grnlight.net/index.php/programming-articles/115-javaone-sun-the-bulk-of-java-is-open-sourced>) from the original on May 27, 2014. Retrieved May 26, 2014.
34. "Sun's Evolving Role as Java Evangelist" (<http://onjava.com/pub/a/onjava/2002/04/17/evangelism.html>) . O'Reilly Media. Archived (<https://web.archive.org/web/20100915162748/http://onjava.com/pub/a/onjava/2002/04/17/evangelism.html>) from the original on September 15, 2010. Retrieved August 2, 2009.
35. "Oracle and Java" (<https://web.archive.org/web/20100131091008/http://www.oracle.com/us/technologies/java/>) . *oracle.com*. Oracle Corporation. Archived from the original (<https://www.oracle.com/us/technologies/java/>) on January 31, 2010. Retrieved August 23, 2010.

"Oracle has been a leading and substantive supporter of Java since its emergence in 1995 and takes on the new role as steward of Java technology with a relentless commitment to fostering a community of participation and transparency."
36. Gosling, James (April 9, 2010). "Time to move on..." ([https://web.archive.org/web/20101105031239/http://nighthacks.com/roller/jag/entry/time\\_to\\_move\\_on](https://web.archive.org/web/20101105031239/http://nighthacks.com/roller/jag/entry/time_to_move_on)) *On a New Road*. Archived from the original ([http://nighthacks.com/roller/jag/entry/time\\_to\\_move\\_on](http://nighthacks.com/roller/jag/entry/time_to_move_on)) on November 5, 2010. Retrieved November 16, 2011.
37. Topic, Dalibor. "Moving to a Plugin-Free Web" ([https://blogs.oracle.com/java-platform-group/entry/moving\\_to\\_a\\_plugin\\_free](https://blogs.oracle.com/java-platform-group/entry/moving_to_a_plugin_free)) . Archived ([https://web.archive.org/web/20160316164325/https://blogs.oracle.com/java-platform-group/entry/moving\\_to\\_a\\_plugin\\_free](https://web.archive.org/web/20160316164325/https://blogs.oracle.com/java-platform-group/entry/moving_to_a_plugin_free)) from the original on March 16, 2016. Retrieved March 15, 2016.
38. "Learn About Java Technology" (<http://www.java.com/en/about/>) . Oracle. Archived (<https://web.archive.org/web/20111124090716/http://www.java.com/en/about/>) from the original on November 24, 2011. Retrieved November 21, 2011.
39. "Why should I uninstall older versions of Java from my system?" ([https://www.java.com/en/download/faq/remove\\_olderversions.xml](https://www.java.com/en/download/faq/remove_olderversions.xml)) . Oracle. Archived ([https://web.archive.org/web/20180212011608/https://java.com/en/download/faq/remove\\_olderversions.xml](https://web.archive.org/web/20180212011608/https://java.com/en/download/faq/remove_olderversions.xml)) from the original on February 12, 2018. Retrieved September 24, 2021.



40. "Oracle Java SE Support Roadmap" (<https://www.oracle.com/java/technologies/java-se-support-roadmap.html>) . Oracle. September 13, 2021. Archived (<https://web.archive.org/web/20210919090451/https://www.oracle.com/java/technologies/java-se-support-roadmap.html>) from the original on September 19, 2021. Retrieved September 18, 2021.
41. "Temurin™ Support; Adoptium" (<https://adoptium.net/support/>) . *adoptium.net*. Archived (<https://web.archive.org/web/20240329061257/https://adoptium.net/support/>) from the original on March 29, 2024. Retrieved March 29, 2024.
42. "JAVASOFT SHIPS JAVA 1.0" (<https://web.archive.org/web/20070310235103/http://www.sun.com/smi/Press/sunflash/1996-01/sunflash.960123.10561.xml>) . *sun.com*. Archived from the original (<http://www.sun.com/smi/Press/sunflash/1996-01/sunflash.960123.10561.xml>) on March 10, 2007. Retrieved February 5, 2008.
43. Chander, Sharat. "Introducing Java SE 11" (<https://blogs.oracle.com/java-platform-group/introducing-java-se-11>) . *oracle.com*. Archived (<https://web.archive.org/web/20180926093144/https://blogs.oracle.com/java-platform-group/introducing-java-se-11>) from the original on September 26, 2018. Retrieved September 26, 2018.
44. Chander, Sharat (September 15, 2020). "The Arrival of Java 15!" (<https://blogs.oracle.com/java-platform-group/the-arrival-of-java-15>) . Oracle. Archived (<https://web.archive.org/web/20200916092332/https://blogs.oracle.com/java-platform-group/the-arrival-of-java-15>) from the original on September 16, 2020. Retrieved September 15, 2020.
45. "JDK 21" (<https://openjdk.org/projects/jdk/21/>) . *openjdk.org*. Archived (<https://web.archive.org/web/20230920173515/https://openjdk.org/projects/jdk/21/>) from the original on September 20, 2023. Retrieved September 20, 2023.
46. "JDK 24" (<https://openjdk.org/projects/jdk/24/>) .
47. "Java Card Overview" (<https://www.oracle.com/technetwork/java/embedded/javacard/overview/>) . *Oracle Technology Network*. Oracle. Archived (<https://web.archive.org/web/20150107034738/http://www.oracle.com/technetwork/java/embedded/javacard/overview/>) from the original on January 7, 2015. Retrieved December 18, 2014.
48. "Java Platform, Micro Edition (Java ME)" (<https://www.oracle.com/technetwork/java/embedded/javame/>) . *Oracle Technology Network*. Oracle. Archived (<https://web.archive.org/web/20150104210546/http://www.oracle.com/technetwork/java/embedded/javame/>) from the original on January 4, 2015. Retrieved December 18, 2014.

49. "Java SE" (<https://www.oracle.com/technetwork/java/javase/overview/>) . *Oracle Technology Network*. Oracle. Archived (<https://web.archive.org/web/20141224184532/http://www.oracle.com/technetwork/java/javase/overview/>) from the original on December 24, 2014. Retrieved December 18, 2014.
50. "Java Platform, Enterprise Edition (Java EE)" (<https://www.oracle.com/technetwork/java/javaee/overview/>) . *Oracle Technology Network*. Oracle. Archived (<https://web.archive.org/web/20141217155326/http://www.oracle.com/technetwork/java/javaee/overview/>) from the original on December 17, 2014. Retrieved December 18, 2014.
51. "Deep Dive Into the New Java JIT Compiler – Graal | Baeldung" (<https://www.baeldung.com/graal-java-jit-compiler>) . *www.baeldung.com*. August 6, 2021. Archived (<https://web.archive.org/web/20211028165746/https://www.baeldung.com/graal-java-jit-compiler>) from the original on October 28, 2021. Retrieved October 13, 2021.
52. "Is the JVM (Java Virtual Machine) platform dependent or platform independent? What is the advantage of using the JVM, and having Java be a translated language?" (<http://www.programmerinterview.com/index.php/java-questions/jvm-platform-dependent/>) . Programmer Interview. Archived (<https://web.archive.org/web/20150119144223/http://www.programmerinterview.com/index.php/java-questions/jvm-platform-dependent/>) from the original on January 19, 2015. Retrieved January 19, 2015.
53. Jelovic, Dejan. "Why Java will always be slower than C++" ([https://web.archive.org/web/20080211111923/http://www.jelovic.com/articles/why\\_java\\_is\\_slow.htm](https://web.archive.org/web/20080211111923/http://www.jelovic.com/articles/why_java_is_slow.htm)) . Archived from the original ([http://www.jelovic.com/articles/why\\_java\\_is\\_slow.htm](http://www.jelovic.com/articles/why_java_is_slow.htm)) on February 11, 2008. Retrieved February 15, 2008.
54. Hundt, Robert. "Loop Recognition in C++/Java/Go/Scala" (<https://days2011.scala-lang.org/sites/days2011/files/ws3-1-Hundt.pdf>) (PDF). Archived (<https://web.archive.org/web/20111116151424/https://days2011.scala-lang.org/sites/days2011/files/ws3-1-Hundt.pdf>) (PDF) from the original on November 16, 2011. Retrieved July 12, 2012.
55. "Symantec's Just-In-Time Java Compiler To Be Integrated into Sun JDK 1.1" ([https://web.archive.org/web/20100628171748/http://www.symantec.com/about/news/release/article.jsp?prid=19970407\\_03](https://web.archive.org/web/20100628171748/http://www.symantec.com/about/news/release/article.jsp?prid=19970407_03)) . Archived from the original ([http://www.symantec.com/about/news/release/article.jsp?prid=19970407\\_03](http://www.symantec.com/about/news/release/article.jsp?prid=19970407_03)) on June 28, 2010. Retrieved August 1, 2009.

56. Salcic, Zoran; Park, Heejong; Teich, Jürgen; Malik, Avinash; Nadeem, Muhammad (July 22, 2017). "Noc-HMP: A Heterogeneous Multicore Processor for Embedded Systems Designed in SystemJ". *ACM Transactions on Design Automation of Electronic Systems*. **22** (4): 73. doi:10.1145/3073416 (<https://doi.org/10.1145%2F3073416>) . ISSN 1084-4309 (<https://search.worldcat.org/issn/1084-4309>) . S2CID 11150290 (<https://api.semanticscholar.org/CorpusID:11150290>) .
57. Bloch 2018, p. 26-28, §Item 7: Eliminate obsolete object references.
58. "NullPointerException" (<http://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html>) . Oracle. Archived (<https://web.archive.org/web/20140506214735/http://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html>) from the original on May 6, 2014. Retrieved May 6, 2014.
59. "Exceptions in Java" (<http://www.artima.com/designtechniques/exceptions.html>) . Artima.com. Archived (<https://web.archive.org/web/20090121152332/http://www.artima.com/designtechniques/exceptions.html>) from the original on January 21, 2009. Retrieved August 10, 2010.
60. "Java HotSpot™ Virtual Machine Performance Enhancements" (<https://docs.oracle.com/javase/7/docs/technotes/guides/vm/performance-enhancements-7.html>) . Oracle.com. Archived (<https://web.archive.org/web/20170529071720/http://docs.oracle.com/javase/7/docs/technotes/guides/vm/performance-enhancements-7.html>) from the original on May 29, 2017. Retrieved April 26, 2017.
61. "Operator Overloading (C# vs Java)" (<http://msdn.microsoft.com/en-us/library/ms228498%28v=vs.90%29.aspx>) . *C# for Java Developers*. Microsoft. Archived ([https://web.archive.org/web/20150107190007/http://msdn.microsoft.com/en-us/library/ms228498\(v=vs.90\).aspx](https://web.archive.org/web/20150107190007/http://msdn.microsoft.com/en-us/library/ms228498(v=vs.90).aspx)) from the original on January 7, 2015. Retrieved December 10, 2014.
62. "Multiple Inheritance of State, Implementation, and Type" (<https://docs.oracle.com/javase/tutorial/java/landl/multipleinheritance.html>) . *The Java Tutorials*. Oracle. Archived (<https://web.archive.org/web/20141109034520/https://docs.oracle.com/javase/tutorial/java/landl/multipleinheritance.html>) from the original on November 9, 2014. Retrieved December 10, 2014.
63. "Deprecated APIs, Features, and Options" (<https://www.oracle.com/technetwork/java/javase/9-deprecated-features-3745636.html#JDK-8074165>) . Oracle. Archived (<https://web.archive.org/web/20190619200811/https://www.oracle.com/technetwork/java/javase/9-deprecated-features-3745636.html#JDK-8074165>) from the original on June 19, 2019. Retrieved May 31, 2019.

64. "Applet (Java Platform SE 7)" (<https://docs.oracle.com/javase/7/docs/api/java/applet/Applet.html>) . Docs. Oracle. Archived (<https://web.archive.org/web/20200802033524/https://docs.oracle.com/javase/7/docs/api/java/applet/Applet.html>) from the original on August 2, 2020. Retrieved May 1, 2020.
65. "What Is a JSP Page? – The Java EE 5 Tutorial" (<https://docs.oracle.com/javaee/5/tutorial/doc/bnagy.html>) . docs.oracle.com. Archived (<https://web.archive.org/web/20200802003029/https://docs.oracle.com/javaee/5/tutorial/doc/bnagy.html>) from the original on August 2, 2020. Retrieved May 1, 2020.
66. "Trail: Creating a GUI With JFC/Swing (The Java Tutorials)" (<https://docs.oracle.com/javase/tutorial/uiswing/>) . docs.oracle.com. Archived (<https://web.archive.org/web/20200429104302/https://docs.oracle.com/javase/tutorial/uiswing/>) from the original on April 29, 2020. Retrieved May 1, 2020.
67. "Removed from JDK 11, JavaFX 11 arrives as a standalone module" (<https://www.infoworld.com/article/3305073/removed-from-jdk-11-javafx-11-arrives-as-a-standalone-module.html>) . InfoWorld. September 20, 2018. Archived (<https://web.archive.org/web/20201014141716/https://www.infoworld.com/article/3305073/removed-from-jdk-11-javafx-11-arrives-as-a-standalone-module.html>) from the original on October 14, 2020. Retrieved October 13, 2020.
68. "Getting Started with JavaFX: Hello World, JavaFX Style" ([https://docs.oracle.com/javafx/2/get\\_started/hello\\_world.htm](https://docs.oracle.com/javafx/2/get_started/hello_world.htm)) . JavaFX 2 Tutorials and Documentation. Oracle. Archived ([https://web.archive.org/web/20200802013650/https://docs.oracle.com/javafx/2/get\\_started/hello\\_world.htm](https://web.archive.org/web/20200802013650/https://docs.oracle.com/javafx/2/get_started/hello_world.htm)) from the original on August 2, 2020. Retrieved May 1, 2020.
69. "Java and Scala's Type Systems are Unsound" (<https://raw.githubusercontent.com/namin/unsound/master/doc/unsound-oopsla16.pdf>) (PDF). Archived (<https://web.archive.org/web/20161128174902/https://raw.githubusercontent.com/namin/unsound/master/doc/unsound-oopsla16.pdf>) (PDF) from the original on November 28, 2016. Retrieved February 20, 2017.
70. Arnold, Ken (June 27, 2005). "Generics Considered Harmful" ([https://web.archive.org/web/20071010002142/http://weblogs.java.net/blog/arnold/archive/2005/06/generics\\_considered\\_harmful\\_1.html](https://web.archive.org/web/20071010002142/http://weblogs.java.net/blog/arnold/archive/2005/06/generics_considered_harmful_1.html)) . java.net. Archived from the original ([https://weblogs.java.net/blog/arnold/archive/2005/06/generics\\_considered\\_harmful\\_1.html](https://weblogs.java.net/blog/arnold/archive/2005/06/generics_considered_harmful_1.html)) on October 10, 2007. Retrieved September 10, 2015.
71. Owens, Sean R. "Java and unsigned int, unsigned short, unsigned byte, unsigned long, etc. (Or rather, the lack thereof)" (<https://web.archive.org/web/20090220171410/http://darksleep.com/player/JavaAndUnsignedTypes.html>) . Archived from the original (<http://darksleep.com/player/JavaAndUnsignedTypes.html>) on February 20, 2009. Retrieved July 4, 2011.

72. Kahan, William (March 1, 1998). "How Java's Floating-Point Hurts Everyone Everywhere – ACM 1998 Workshop on Java (Stanford)" (<http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf>) (PDF). Electrical Engineering & Computer Science, University of California at Berkeley. Archived (<https://web.archive.org/web/20120905004527/http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf>) (PDF) from the original on September 5, 2012. Retrieved June 4, 2011.
73. "Have you checked the Java?" (<https://web.archive.org/web/20120921140402/http://blogs.technet.com/b/mmpc/archive/2010/10/18/have-you-checked-the-java.aspx>) . Archived from the original (<http://blogs.technet.com/b/mmpc/archive/2010/10/18/have-you-checked-the-java.aspx>) on September 21, 2012. Retrieved December 23, 2011.
74. Chidester, Ashlan. *Java Persistence API, Jenkins and AWS* (<https://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9798224253951>) . ISBN 9798224253951. Retrieved September 16, 2024.
75. Cadenhead, Rogers (November 20, 2017), *Understanding How Java Programs Work* (<http://www.informit.com/articles/article.aspx?p=2832404&seqNum=4>) , archived (<https://web.archive.org/web/20210813193850/https://www.informit.com/articles/article.aspx?p=2832404&seqNum=4>) from the original on August 13, 2021, retrieved March 26, 2019
76. Woolf, Nicky (May 26, 2016). "Google wins six-year legal battle with Oracle over Android code copyright" (<https://www.theguardian.com/technology/2016/may/26/google-wins-copyright-lawsuit-oracle-java-code>) . *The Guardian*. ISSN 0261-3077 (<https://search.worldcat.org/issn/0261-3077>) . Archived (<https://web.archive.org/web/20190326203847/https://www.theguardian.com/technology/2016/may/26/google-wins-copyright-lawsuit-oracle-java-code>) from the original on March 26, 2019. Retrieved March 26, 2019.
77. Bloch 2018, pp. 1–4, § 1 Introduction.
78. "java.nio (Java Platform SE 8)" (<https://docs.oracle.com/javase/8/docs/api/java/nio/package-summary.html>) . *docs.oracle.com*.
79. "Java Networking" (<https://docs.oracle.com/javase/8/docs/technotes/guides/net/>) . *docs.oracle.com*.
80. "HttpClient (Java SE 11 & JDK 11)" (<https://docs.oracle.com/en/java/javase/11/docs/api/java.net.http/java/net/http/HttpClient.html>) . *docs.oracle.com*.
81. "Collections Framework Overview" (<http://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html>) . *Java Documentation*. Oracle. Archived (<https://web.archive.org/web/20141231132540/http://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html>) from the original on December 31, 2014. Retrieved December 18, 2014.



82. "Java Security Overview" (<http://docs.oracle.com/javase/8/docs/technotes/guides/security/overview/jsoverview.html>) . *Java Documentation*. Oracle. Archived (<https://web.archive.org/web/20150103045031/http://docs.oracle.com/javase/8/docs/technotes/guides/security/overview/jsoverview.html>) from the original on January 3, 2015. Retrieved December 18, 2014.
83. "Trail: Internationalization" (<http://docs.oracle.com/javase/tutorial/i18n/>) . *The Java Tutorials*. Oracle. Archived (<https://web.archive.org/web/20141231053232/http://docs.oracle.com/javase/tutorial/i18n/>) from the original on December 31, 2014. Retrieved December 18, 2014.
84. "How to Write Doc Comments for the Javadoc Tool" (<https://www.oracle.com/technetwork/articles/java/index-137868.html>) . *Oracle Technology Network*. Oracle. Archived (<https://web.archive.org/web/20141218182906/http://www.oracle.com/technetwork/articles/java/index-137868.html>) from the original on December 18, 2014. Retrieved December 18, 2014.
85. Niccolai, James (January 24, 2001). "Sun, Microsoft settle Java lawsuit" (<https://www.infoworld.com/article/2074908/sun-microsoft-settle-java-lawsuit.html>) . *JavaWorld*. IDG News Service. Archived (<https://web.archive.org/web/20200714001541/https://www.infoworld.com/article/2074908/sun-microsoft-settle-java-lawsuit.html>) from the original on July 14, 2020. Retrieved July 13, 2020.
86. van Gurp, Jilles (November 13, 2007). "Google Android: Initial Impressions and Criticism" ([https://web.archive.org/web/20080828111808/http://www.javalobby.org/nl/archive/jlnews\\_20071130.html](https://web.archive.org/web/20080828111808/http://www.javalobby.org/nl/archive/jlnews_20071130.html)) . *Javalobby*. Archived from the original on August 28, 2008. Retrieved March 7, 2009. *"Frankly, I don't understand why Google intends to ignore the vast amount of existing implementation out there. It seems like a bad case of "not invented here" to me. Ultimately, this will slow adoption. There are already too many Java platforms for the mobile world and this is yet another one"*
87. Mullin, Joe (May 7, 2012). "Google guilty of infringement in Oracle trial; future legal headaches loom" (<https://arstechnica.com/tech-policy/news/2012/05/jury-rules-google-violated-copyright-law-google-moves-for-mistrial.ars>) . *Law & Disorder*. Ars Technica. Archived (<https://web.archive.org/web/20120508134916/http://arstechnica.com/tech-policy/news/2012/05/jury-rules-google-violated-copyright-law-google-moves-for-mistrial.ars>) from the original on May 8, 2012. Retrieved May 8, 2012.
88. Mullin, Joe (May 31, 2012). "Google wins crucial API ruling, Oracle's case decimated" (<https://arstechnica.com/tech/2012/05/google-wins-crucial-api-ruling-oracles-case-decimated/>) . *Ars Technica*. Archived (<https://web.archive.org/web/20170312065520/https://arstechnica.com/tech-policy/2012/05/google-wins-crucial-api-ruling-oracles-case-decimated/>) from the original on March 12, 2017. Retrieved June 1, 2012.



89. Rosenblatt, Seth (May 9, 2014). "Court sides with Oracle over Android in Java patent appeal" (<https://www.cnet.com/news/court-sides-with-oracle-over-android-in-java-patent-appeal/>) . *CNET*. Archived (<https://web.archive.org/web/20140510203805/http://www.cnet.com/news/court-sides-with-oracle-over-android-in-java-patent-appeal/>) from the original on May 10, 2014. Retrieved May 10, 2014.
90. Mullin, Joe (May 26, 2016). "Google beats Oracle—Android makes "fair use" of Java APIs" (<https://arstechnica.com/tech-policy/2016/05/google-wins-trial-against-oracle-as-jury-finds-android-is-fair-use/>) . *Ars Technica*. Archived (<https://web.archive.org/web/20170120164551/http://arstechnica.com/tech-policy/2016/05/google-wins-trial-against-oracle-as-jury-finds-android-is-fair-use/>) from the original on January 20, 2017. Retrieved May 26, 2016.
91. Farivar, Cyrus (March 27, 2018). " "Google's use of the Java API packages was not fair," appeals court rules" (<https://arstechnica.com/tech-policy/2018/03/googles-use-of-the-java-api-packages-was-not-fair-appeals-court-rules/>) . *Ars Technica*. Archived (<https://web.archive.org/web/20190924081919/https://arstechnica.com/tech-policy/2018/03/googles-use-of-the-java-api-packages-was-not-fair-appeals-court-rules/>) from the original on September 24, 2019. Retrieved August 6, 2019.
92. Lee, Timothy (April 23, 2019). "Google asks Supreme Court to overrule disastrous ruling on API copyrights" (<https://arstechnica.com/tech-policy/2019/01/google-asks-supreme-court-to-overrule-disastrous-ruling-on-api-copyrights/>) . *Ars Technica*. Archived (<https://web.archive.org/web/20190423084450/https://arstechnica.com/tech-policy/2019/01/google-asks-supreme-court-to-overrule-disastrous-ruling-on-api-copyrights/>) from the original on April 23, 2019. Retrieved April 23, 2019.
93. "*Google LLC v. Oracle America, Inc* 593 U. S. \_\_\_\_ (2021)" ([https://www.supremecourt.gov/opinions/20pdf/18-956\\_d18f.pdf](https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf)) (PDF). Archived ([https://web.archive.org/web/20210405140150/https://www.supremecourt.gov/opinions/20pdf/18-956\\_d18f.pdf](https://web.archive.org/web/20210405140150/https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf)) (PDF) from the original on April 5, 2021. Retrieved April 6, 2021.

## Bibliography

---

- Bloch, Joshua (2018). *Effective Java: Programming Language Guide* (Third ed.). Addison-Wesley. ISBN 978-0-13-468599-1.
- Gosling, James; Joy, Bill; Steele, Guy L. Jr.; Bracha, Gilad (2005). *The Java Language Specification* (<https://java.sun.com/docs/books/jls/>) (3rd ed.). Addison-Wesley. ISBN 0-321-24678-0. Archived (<https://web.archive.org/web/20120214061826/http://java.sun.com/docs/books/jls/>) from the original on February 14, 2012. Retrieved February 8, 2019.

- Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, Alex (2014). *The Java® Language Specification* (<https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf>) (PDF) (Java SE 8 ed.). Archived (<https://web.archive.org/web/20141021061951/http://docs.oracle.com/javase/specs/jls/se8/jls8.pdf>) (PDF) from the original on October 21, 2014. Retrieved November 18, 2014.
- Lindholm, Tim; Yellin, Frank (1999). *The Java Virtual Machine Specification* (<https://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html>) (2nd ed.). Addison-Wesley. ISBN 0-201-43294-3. Archived (<https://web.archive.org/web/20110925050249/http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html>) from the original on September 25, 2011. Retrieved February 8, 2019.

## External links

---

- OpenJDK, Oracle (<https://jdk.java.net/>)
- JDK builds, Adoptium (<https://adoptium.net/>)