**Introduction**

This booklet had the intent to summary a brief java’s history and then show the release versions of Java platform and some of the major enhancements of each one of them.

**History of Java**

Java is an object-oriented programming language. Java was developed by James Gosling and colleagues at Sun Microsystems in the 1990s.  
  
Java was started as a project called “Oak” by James Gosling in June 1991. The goal was to implement a virtual machine that is much simpler than C/C++.  
Java was developed with the goal to implement “Write Once, Run Anywhere” programming model.

The original name of the programming language was Oak. Later on, it was discovered that there is already a programming language named Oak.  
As the story goes, after a lot of discussions, the development team took a break and went out for Coffee. That’s where the name “Java” was proposed and finalized by the team.

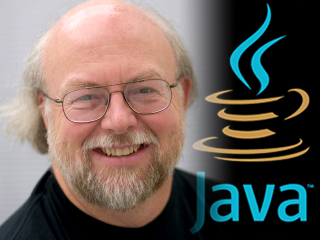


Figura 1 James Gosling the father of JAva

**Philosophy of Java**

The Java programming language was built on the following five philosophies.  
It will use the Object-oriented programming methodology

The same program should be executable on multiple operating systems.  
Built-in support for using computer networks.  
Designed to execute code from the remote sources securely.  
It should be easy to use, take the good features of Object-oriented programming.

**References**

*Wikipedia  
Oracle*

**Versions**

To understanding part of how new features are added to Java we need comprehend the JCP and JSR (<https://jcp.org/en/introduction/overview>).

See description of JCP and JSR extracted from the JCP’s site:

**The JCP program** holds the responsibility for the development of Java technology. As an open, inclusive organization of active members and non-member public input, it primarily guides the development and approval of Java technical specifications. Anyone can register and join the JCP and have a part in its process, and you don't even have to join to contribute as a public participant.  
   
The work of the Java Community under the JCP program helps to ensure Java technology's standard of stability and cross-platform compatibility, enabling it to operate on hundreds of millions of devices, from desktop computers to consumer electronics to industrial robots.  
   
Just as important, the JCP program continually grows the platform's specification portfolio to meet the emerging technology needs of developers and organizations globally who depend on Java technology.

**Java Specification Requests (JSRs)** are the actual descriptions of proposed and final specifications for the Java platform. At any one time there are numerous JSRs moving through the review and approval process.

A simple way to stay up to date and track the JSRs in each stage of review is to join the [Mailing List](https://jcp.org/en/participation/mail). As a Mailing List member, you can automatically receive emails on JSRs as they move through the stages in review.

Of course, at any time, you can see a list of all JSRs, by visiting the [Java Specification Requests list](https://jcp.org/en/jsr/all). Using the links near the top of the page, you can sort the list by JSR ID number, Title, and the Spec Lead Company Name.

**Warning**

On the next section we are seeing just a summary of changes from the language’s point of view. For see the changelogs in details see the link bellow:

[https://www.oracle.com/java**/**technologies/javase/jdk-relnotes-index.html](https://www.oracle.com/java/technologies/javase/jdk-relnotes-index.html)

**Summary: JDK1.0 until JDK7**

**JDK Beta**

*Developed by James Gosling at Sun Microsystems on 1995.*

**JDK 1.0**  
*Released on January 23, 1996*

**JDK 1.1**

*Released on February 19, 1997*

*Some of major additions:*

*Inner classes added to the language*

*JavaBeans*

*JDBC*

*RMI*

*Reflection for introspection only*

**J2SE 1.2**

*Codename* ***Playground***

*Released on December 8, 1998*

*Born the Java2*

*Some of major additions:*

*Collections framework*

*Swing*

*JIT Compiler*

**J2SE 1.3**

*Released around on May 8, 2000*

*Codename Kestrel*

*Some of major additions:*

*JNDI*

*JPDA*

**J2SE 1.4**

*Codename* ***Merlin***

*Released February 6, 2002*

*Some of major additions:*

*Assert keyword*

*Regular Expressions Improvements*

*Exception chaining*

*Non-blocking IO*

*Logging API*

*Image IO API*

*XML Parsers*

*XSLT Processors*

*Java Web Start*

*Preferences API*

**J2SE 5.0**

*Codename* ***Tiger***

*Released September 30, 2004*

*Some of major additions:*

*For-each loop*

*Autoboxing*

*Annotations*

*Generics*

*Enumerations*

*Varargs*

*Static Imports*

**Java SE 6**

*Codename* ***Mustang***

*Released December 11, 2006*

*Some of major additions:*

*Scripting Language Support (Rhino)*

*Java Compiler API*

**Java SE 7**

*Codename* ***Dolphin***

*Released around July 7, 2011*

*Some of major additions:*

*Scripting Language Support (Rhino)*

*JVM support for dynamic languages*

*Strings in switch*

*Automatic resource management in try-statement*

*Catching multiple exception*

*New IO API*

**Java SE 8 until Java SE 15**

Now we’ll starting see the releases in a little more detailed. However, we are focus more in either that changes that somehow had impacted on language itself or the explicitly language’s changes.

That means that we won’t see the dozens of improvements and new features regard the Garbage Collector, JVM internal and others that don’t impact directly the syntax.

**Java SE 8**

Codename **Dolphin**

Released March 18, 2014

**Some of major additions:**

Lambda expressions

forEach() method

default and static method in interfaces

Functional Interfaces

Stream API

New Date Time API

Nashorn

Remove the permanent Generation

1 Java memory Management for JVM

The major change in memory management was the replacement of the Permanent Generation by the Metaspace.



Figura 2 source http://www.journaldev.com/2856/java-jvm-memory-model-memory-management-in-java

With Java 8, there is no Perm Gen, which means there is no more “java.lang.OutOfMemoryError: PermGen” space problems. Unlike Perm Gen which resides in the Java heap, Metaspace is not part of the heap. Most allocations of the class metadata are now allocated out of native memory. Metaspace by default auto increases its size (up to what the underlying OS provides), while Perm Gen always has fixed maximum size. Two new flags can be used to set the size of the metaspace, they are: “**-XX:MetaspaceSize**” and “**-XX:MaxMetaspaceSize**”. The theme behind the Metaspace is that the lifetime of classes and their metadata matches the lifetime of the classloaders. That is, as long as the classloader is alive, the metadata remains alive in the Metaspace and can’t be freed.

**Java SE 9**

Codename Dolphin

Released March 18, 2014

Some of major additions:

Java Platform Module System

Jshell

Jlink

Reactive Streams

Http2 Client

Compact Strings

**Java SE 10**

OpenJDK: Released on March 20, 2018

Major additions:

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

**Java SE 11**

Released on September 25, 2018

[JEP-181: Nest-Based Access Control](https://openjdk.java.net/jeps/181)

[JEP-309: Dynamic Class-File Constants](https://openjdk.java.net/jeps/309)

[JEP-315: Improve Aarch64 Intrinsics](https://openjdk.java.net/jeps/315)

[JEP-318: Epsilon: A No-Op Garbage Collector](https://openjdk.java.net/jeps/318)

[JEP-320: Remove the Java EE and CORBA Modules](https://openjdk.java.net/jeps/320)

[JEP-321: HTTP Client (Standard)](https://openjdk.java.net/jeps/321)

[JEP-323: Local-Variable Syntax for Lambda Parameters](https://openjdk.java.net/jeps/323)

[JEP-324: Key Agreement with Curve25519 and Curve448](https://openjdk.java.net/jeps/324)

[JEP-327: Unicode 10](https://openjdk.java.net/jeps/327)

[JEP-328: Flight Recorder](https://openjdk.java.net/jeps/328)

[JEP-329: ChaCha20 and Poly1305 Cryptographic Algorithms](https://openjdk.java.net/jeps/329)

[JEP-330: Launch Single-File Source-Code Programs](https://openjdk.java.net/jeps/330)

[JEP-331: Low-Overhead Heap Profiling](https://openjdk.java.net/jeps/331)

[JEP-332: Transport Layer Security (TLS) 1.3](https://openjdk.java.net/jeps/332)

[JEP-333: ZGC: A Scalable Low-Latency Garbage Collector (Experimental)](https://openjdk.java.net/jeps/333)

[JEP-335: Deprecate the Nashorn JavaScript Engine](https://openjdk.java.net/jeps/335)

[JEP-336: Deprecate the Pack200 Tools and API](https://openjdk.java.net/jeps/336)

**Java SE 12**

Released on March 19, 2019. [JEP-189: Shenandoah: A Low-Pause-Time Garbage Collector (Experimental)](https://openjdk.java.net/jeps/189)

* [JEP-230: Microbenchmark Suite](https://openjdk.java.net/jeps/230)
* [JEP-325: Switch Expressions (Preview)](https://openjdk.java.net/jeps/325)
* [JEP-334: JVM Constants API](https://openjdk.java.net/jeps/334)
* [JEP-340: One AArch64 Port, Not Two](https://openjdk.java.net/jeps/340)
* [JEP-341: Default CDS Archives](https://openjdk.java.net/jeps/341)
* [JEP-344: Abortable Mixed Collections for G1](https://openjdk.java.net/jeps/344)
* [JEP-346: Promptly Return Unused Committed Memory from G1](https://openjdk.java.net/jeps/346)

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".[[339]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-339)

* [JEP-350: Dynamic CDS Archives](https://openjdk.java.net/jeps/350)
* [JEP-351: ZGC: Uncommit Unused Memory](https://openjdk.java.net/jeps/351)
* [JEP-353: Reimplement the Legacy Socket API](https://openjdk.java.net/jeps/353)
* [JEP-354: Switch Expressions (Preview)](https://openjdk.java.net/jeps/354)
* [JEP-355: Text Blocks (Preview)](https://openjdk.java.net/jeps/355)

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".[[344]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-344)

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

Java SE 15

Released on September 15, 2020

[JEP-339: Edwards-Curve Digital Signature Algorithm (EdDSA)](https://openjdk.java.net/jeps/339)

* [JEP-360: Sealed Classes (Preview)](https://openjdk.java.net/jeps/360)
* [JEP-371: Hidden Classes](https://openjdk.java.net/jeps/371)
* [JEP-372: Remove the Nashorn JavaScript Engine](https://openjdk.java.net/jeps/372)
* [JEP-373: Reimplement the Legacy DatagramSocket API](https://openjdk.java.net/jeps/373)
* [JEP-374: Disable and Deprecate Biased Locking](https://openjdk.java.net/jeps/374)
* [JEP-375: Pattern Matching for instanceof (Second Preview)](https://openjdk.java.net/jeps/375)
* [JEP-377: ZGC: A Scalable Low-Latency Garbage Collector](https://openjdk.java.net/jeps/377)
* [JEP-378: Text Blocks](https://openjdk.java.net/jeps/378)
* [JEP-379: Shenandoah: A Low-Pause-Time Garbage Collector](https://openjdk.java.net/jeps/379)
* [JEP-381: Remove the Solaris and SPARC Ports](https://openjdk.java.net/jeps/381)
* [JEP-383: Foreign-Memory Access API (Second Incubator)](https://openjdk.java.net/jeps/383)
* [JEP-384: Records (Second Preview)](https://openjdk.java.net/jeps/384)
* [JEP-385: Deprecate RMI Activation for Removal](https://openjdk.java.net/jeps/385)