**Installer l’environnement de développement java**

[1. Prolégomènes 1](#_Toc109674292)

[1.1. Héritage 2](#_Toc109674293)

[1.2. Compilations d’un programme 2](#_Toc109674294)

[1.2.1. Compilation classique d’un programme 3](#_Toc109674295)

[1.2.1.1. La compilation classique 3](#_Toc109674296)

[1.2.1.2. La compilation avec l’exemple du C 3](#_Toc109674297)

[1.2.2. Java est un langage compilé et interprété 5](#_Toc109674298)

[1.2.2.1. Compilation et interprétation du langage Java 5](#_Toc109674299)

[1.2.2.2. Audience du langage Java 7](#_Toc109674300)

[1.3. Les concepts de JVM, JRE et JDK 8](#_Toc109674301)

[1.3.1. JVM 8](#_Toc109674302)

[1.3.1.1. Le principe de machine virtuelle 8](#_Toc109674303)

[1.3.1.2. Fonctionnement de la JVM 9](#_Toc109674304)

[1.3.1.3. Les fonctions principales de la JVM 10](#_Toc109674305)

[1.3.2. JRE 10](#_Toc109674306)

[1.3.3. JDK 10](#_Toc109674307)

[1.4. Historique des versions de Java 10](#_Toc109674308)

[1.4.1. Release table 10](#_Toc109674309)

[2. Installation 12](#_Toc109674310)

[2.1. Prérequis hardware 12](#_Toc109674311)

[2.1.1. Les trois types configurations hardware 13](#_Toc109674312)

[3. Tests 13](#_Toc109674313)

[3.1. Bytecode 13](#_Toc109674314)

[4. Critiques de la technologie Java 13](#_Toc109674315)

[5. Icones 13](#_Toc109674316)

[6. Links 14](#_Toc109674317)

# Prolégomènes

Avant d’attaquer les modalités d’installation de l’environnement de développement java sous Windows, il convient de rappeler le contexte global de la technologie java.

La technologie Java définit à la fois un langage de programmation orienté objet et une plateforme informatique.

La technologie Java a été développée, en 1995, par James A. Gosling chez Sun Microsystems[[1]](#footnote-1) ; cette entreprise a été rachetée par Oracle en 2009. Java est devenu donc propriété d’Oracle Corporation.

## Héritage

Java est un langage de programmation multiplateformes[[2]](#footnote-2), il dérive des langages C et C++, dont il reprend, en partie, la syntaxe.



Figure Timeline 3 langages

## Compilations d’un programme

Une des particularités principales qui différencie le langage Java des autres langages comme le C ou le C++ est la manière dont il est exécuté et compilé sur une machine.

En effet, un programme C / C++, compilé, binaire, ne fonctionne que sur la plateforme pour laquelle il a été compilé.

Pour comprendre comment fonctionne Java, il est utile de rappeler la compilation classique, avec l’exemple du C.

### Compilation classique d’un programme

#### La compilation classique

La compilation classique suit ce schéma :



Figure Compilation et exécutable

#### La compilation avec l’exemple du C

Voici un schéma qui synthétise la compilation en C :



Figure La compilation en C

Quant au compilateur Java, nommé ***javac***, il ne traduit pas **directement** le code source, du fichier ***.java***, en langage machine comme les compilateurs de C ou C++.

Il le traduit en un langage intermédiaire appelé **bytecode** représenté par les fichiers ***.class***.

Ce bytecode est ensuite interprété par un autre programme : la machine virtuelle java ou **JVM** (pour Java Virtual Machine).

Donc, un programme java contiendra, toujours, deux types de fichiers : les fichiers sources en java (extension. java) et le résultat de leur compilation en byte code (fichier d’extension .class).



Figure Composition d'un programme Java

### Java est un langage compilé et interprété

#### Compilation et interprétation du langage Java

**Java peut être considéré à la fois comme un langage compilé et interprété car** son code source est d'abord compilé en un bytecode binaire. Ce bytecode s'exécute sur la machine virtuelle Java (JVM), qui est un interpréteur et/ou un compilation juste-à-temps (*just-in-time compilation* ou *JIT compilation)[[3]](#footnote-3)*.

De plus, ce bytecode donne à Java sa portabilité : il fonctionnera sur n'importe quelle JVM correctement implémentée, quelle que soit la configuration matérielle ou logicielle de l'ordinateur.

Voici un schéma illustrant ce mécanisme :



Figure Compilation et interprétation en Java

La JVM est dépendante de la plate-forme, c’est-à-dire que sa mise en œuvre diffère d’une plate-forme à l’autre (Windows, Linux, Mac, etc…). Il en va de même pour le Java Runtime Environment (JRE) et le Java Development Kit (JDK).



Dépendants de chaque plateforme, en raison du type de système d’exploitation de la machine et/ou de processeur sous-jacent.

Figure Dépendance de la plateforme

Mais toutes les JVM peuvent exécuter le même bytecode java. Car Java est indépendant de la plateforme.



Figure Indépendance de la plateforme de Java

C’est l’approche « write once and run anywhere » [[4]](#footnote-4), en français « écrire une fois et exécuter partout ».



Figure 8 write once and run anywhere (WORA)

Cette indépendance de la plate-forme est l'une des caractéristiques qui ont fait de Java l'une des plates-formes de programmation les plus utilisées.

#### Audience du langage Java

En 2019, Java était l'un des langages de programmation les plus populaires, en particulier pour les applications Web client-serveur, avec **9 millions de développeurs[[5]](#footnote-5)**.

En effet selon l’index TIOBE Java est classé troisième :



Tableau Index Tiobe juillet 2022

**[[6]](#footnote-6)**

Toujours selon cette même source, on constate que le langage Java a occupé la **première place** de 2002 à 2017 :



Tableau Index Tiobe sur 10 ans

Précisons les concepts sur lesquels repose la technologie Java.

## Les concepts de JVM, JRE et JDK

En effet, il convient de présenter les notions de JRE, JVM et JDK , que tout développeur Java se doit de connaitre.

### JVM

Avant de détailler la JVN, il y a lieu de présenter le concept de machine virtuelle, dans le cas général.

#### Le principe de machine virtuelle

Prenons la définition de Wikipédia :

****En informatique, une machine virtuelle[[7]](#footnote-7) est une illusion d'un appareil informatique créée par un logiciel d'émulation. Le logiciel d'émulation simule la présence de ressources matérielles et logicielles telles que la mémoire, le processeur, le disque dur, voire le système d'exploitation et les pilotes, permettant d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée.

Un des intérêts des machines virtuelles est de pouvoir s'abstraire des caractéristiques de la machine physique utilisée (matérielles et logicielles — notamment système d'exploitation), permettant une forte portabilité des logiciels[[8]](#footnote-8)

**L'usage de machines virtuelles est l'un des principes fondamentaux de la technologie Java**.

Dans le cas particulier de Java la machine virtuelle prend le nom de : Java Virtual Machine (JVM).

Donc, la JVM est une machine **abstraite** c’est une machine virtuelle, elle n’existe pas physiquement.

#### Fonctionnement de la JVM

Il existe donc, une étape intermédiaire entre le code l’interprété et celui qui est compilé: la JVM.

* La JVM est un programme, qui permet d’isoler l’application qu’il doit faire tourner, du matériel et même du système d’exploitation.
* Le programme n’a aucun accès aux spécificités du matériel, l’ensemble de ses besoins lui étant fourni par la JVM.
* Ainsi, tout programme conçu pour cette machine virtuelle Java pourra fonctionner sur n’importe quel système d’exploitation, du moment que la dite machine virtuelle Java existe pour cet OS en question.
* Le programme fonctionnant depuis la JVM a déjà subi une première phase de compilation pour le transformer non pas en langage machine propre à l’ordinateur, mais dans un langage "machine virtuelle": le **bytecode**.
  + Ensuite la machine virtuelle compile ce bytecode à la volée juste au moment de son utilisation (technologie JIT, Just in Time).
  + La JVM sert, donc, à exécuter du code managé : code dans un langage intermédiaire : le bytecode.
* **La machine virtuelle ne connaît pas le langage Java : elle ne connaît que le bytecode qui est issu de la compilation de codes sources écrits en Java.**

#### Les fonctions principales de la JVM

Donc, pour résumer, la JVM  :

* Charge le code
* Vérifie le code
* Interprète ce code
* Compile ce bytecode à la volée
* Exécute le code
* Fournit l’environnement d’exécution

### JRE

Le Java Runtime Environment, c’est **l’implémentation** de JVM[[9]](#footnote-9). Donc le JRE **dépend** du type de la plateforme.

Il est utilisé pour fournir **l’environnement d’exécution des programmes Java**. Il se compose d’un ensemble de bibliothèques et d’autres fichiers Java nécessaires à la JVM lors de l’exécution.

Il existe de nombreuses implémentations de JVM[[10]](#footnote-10) , celles-ci sont open source :

* HotSpot, la principale implémentation de référence de Java VM
* Eclipse OpenJ9 d’IBM J9, pour Windows, Linux, macOS.

Le JRE ne contient aucun outil de développement, contrairement au JDK.

### JDK

## Historique des versions de Java

<https://en.wikipedia.org/wiki/Java_version_history>

Java 8, 11, 17

### Release table

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Release date** | **End of Free Public Updates**[**[3]**](https://en.wikipedia.org/wiki/Java_version_history#cite_note-auto9-3)[**[8]**](https://en.wikipedia.org/wiki/Java_version_history#cite_note-8)[**[9]**](https://en.wikipedia.org/wiki/Java_version_history#cite_note-9)[**[10]**](https://en.wikipedia.org/wiki/Java_version_history#cite_note-10) | **Extended Support Until** |
| JDK Beta | 1995 | ? | ? |
| JDK 1.0 | January 1996 | ? | ? |
| JDK 1.1 | February 1997 | ? | ? |
| J2SE 1.2 | December 1998 | ? | ? |
| J2SE 1.3 | May 2000 | ? | ? |
| J2SE 1.4 | February 2002 | October 2008 | February 2013 |
| Java SE 5 | September 2004 | November 2009 | April 2015 |
| Java SE 6 | December 2006 | April 2013 | December 2018 December 2026 for Azul[[11]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Azul-11) |
| Java SE 7 | July 2011 | July 2019 | July 2022 |
| Java SE 8 (LTS) | March 2014 | **March 2022 for Oracle (commercial)** December 2030 for Oracle (non-commercial) December 2030 for Azul May 2026 for IBM Semeru[[12]](https://en.wikipedia.org/wiki/Java_version_history" \l "cite_note-IBM_Semeru-12) At least May 2026 for Eclipse Adoptium At least May 2026 for Amazon Corretto | December 2030[[13]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-:1-13) |
| Java SE 9 | September 2017 | March 2018 for OpenJDK | — |
| Java SE 10 | March 2018 | September 2018 for OpenJDK | — |
| Java SE 11 (LTS) | September 2018 | September 2026 for Azul October 2024 for IBM Semeru[[12]](https://en.wikipedia.org/wiki/Java_version_history" \l "cite_note-IBM_Semeru-12) At least October 2024 for Eclipse Adoptium At least September 2027 for Amazon Corretto At least October 2024 for Microsoft[[14]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Microsoft"-14)[[15]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-15) | September 2026 September 2026 for Azul[[11]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Azul-11) |
| Java SE 12 | March 2019 | September 2019 for OpenJDK | — |
| Java SE 13 | September 2019 | March 2020 for OpenJDK | — |
| Java SE 14 | March 2020 | September 2020 for OpenJDK | — |
| Java SE 15 | September 2020 | March 2021 for OpenJDK March 2023 for Azul[[11]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Azul-11) | — |
| Java SE 16 | March 2021 | September 2021 for OpenJDK | — |
| Java SE 17 (LTS) | September 2021 | September 2029 for Azul At least September 2027 for Microsoft[[14]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-Microsoft"-14) At least September 2027 for Eclipse Adoptium | September 2029 or later September 2029 for Azul |
| **Java SE 18** | March 2022 | September 2022 for OpenJDK and Adoptium | — |
| Java SE 19 | September 2022 | March 2023 for OpenJDK | — |
| Java SE 20 | March 2023 | September 2023 for OpenJDK | — |
| Java SE 21 (LTS) | September 2023 | September 2028 | September 2031[[13]](https://en.wikipedia.org/wiki/Java_version_history#cite_note-:1-13) |
| **Legend:**  Old version  Older version, still maintained  **Latest version**  Future release | | | |

<https://en.wikipedia.org/wiki/Java_version_history>

# Installation

## Prérequis hardware

Si l’on veut pouvoir faire du développement, dans n’importe quel langage, il faut disposer d’un ordinateur en conséquence.

En matière de développement, on peut considérer trois types configurations matérielles.

### Les trois types configurations hardware

# Tests

Après avoir installé , il convient de vérifier le bon fonctionnement de l’installation.

## Bytecode

Décompilateur

***javap***

# Critiques de la technologie Java

# Icones

****

****

****

# Links

JAVA: INSTALLATION

•

On aura besoin de trois prérequis

•

Java: https://www.java.com/fr/download/

•

Le JDK: https://www.oracle.com/java/technologies/javase downloads.html

•

Télécharger l’IDE que vous souhaitez

•

Eclipse: https://www.eclipse.org/downloads/

•

Netbeans : https://netbeans.apache.org/download/index.html

•

Intelij : https://www.jetbrains.com/fr fr/idea/

1. https://fr.wikipedia.org/wiki/Sun\_Microsystems [↑](#footnote-ref-1)
2. Logiciels(OS) ou matériels(x86,ARM). [↑](#footnote-ref-2)
3. <https://www.ibm.com/docs/fr/sdk-java-technology/8?topic=reference-jit-compiler> [↑](#footnote-ref-3)
4. En abrégé : WOA [↑](#footnote-ref-4)
5. <https://en.wikipedia.org/wiki/Java_(programming_language)> [↑](#footnote-ref-5)
6. <https://www.tiobe.com/tiobe-index/> [↑](#footnote-ref-6)
7. VM en abrégé. [↑](#footnote-ref-7)
8. <https://fr.wikipedia.org/wiki/Machine_virtuelle> [↑](#footnote-ref-8)
9. Il contient donc la JVM. [↑](#footnote-ref-9)
10. <https://en.wikipedia.org/wiki/List_of_Java_virtual_machines> [↑](#footnote-ref-10)