**Installer l’environnement de développement java**

[1. Prolégomènes 2](#_Toc110806035)

[1.1. Héritage 2](#_Toc110806036)

[1.2. Compilations d’un programme 3](#_Toc110806037)

[1.2.1. Compilation classique d’un programme 3](#_Toc110806038)

[1.2.1.1. La compilation classique 3](#_Toc110806039)

[1.2.1.2. La compilation avec l’exemple du C 4](#_Toc110806040)

[1.2.2. Java est un langage compilé et interprété 5](#_Toc110806041)

[1.2.2.1. Compilation et interprétation du langage Java 5](#_Toc110806042)

[1.2.2.2. Audience du langage Java 8](#_Toc110806043)

[1.3. Les concepts de JVM, JRE et JDK 8](#_Toc110806044)

[1.3.1. JVM 9](#_Toc110806045)

[1.3.1.1. Le principe de machine virtuelle 9](#_Toc110806046)

[1.3.1.2. Fonctionnement de la JVM 9](#_Toc110806047)

[1.3.1.3. Les fonctions principales de la JVM 10](#_Toc110806048)

[1.3.2. JRE 10](#_Toc110806049)

[1.3.2.1. Présentation du JRE 10](#_Toc110806050)

[1.3.2.2. JRE versus JDK 10](#_Toc110806051)

[1.3.3. JDK 10](#_Toc110806052)

[1.3.3.1. Définition du JDK 11](#_Toc110806053)

[1.3.3.2. Classification de JDK JRE JVM 12](#_Toc110806054)

[1.4. Le choix du JDK 12](#_Toc110806055)

[1.4.1. Rappel du contexte historique de Java 12](#_Toc110806056)

[1.4.1.1. Timeline de Java 12](#_Toc110806057)

[1.4.1.2. Les fournisseurs pour OpenJDK 14](#_Toc110806058)

[2. Installation 14](#_Toc110806059)

[2.1. Prérequis hardware 14](#_Toc110806060)

[2.1.1. Architecture de l’ordinateur 15](#_Toc110806061)

[2.1.1.1. Architecture de von Newman 15](#_Toc110806062)

[2.1.1.2. Architecture 32 bits versus 64 bits 15](#_Toc110806063)

[2.1.1.3. Pourquoi passer à un processeur 64 bits ? 15](#_Toc110806064)

[2.1.2. Les trois types configurations hardware 15](#_Toc110806065)

[2.2. Procédure d’installation 15](#_Toc110806066)

[2.3. Verbatim ! 15](#_Toc110806067)

[2.3.1. Installation du Java SE 19](#_Toc110806068)

[2.3.2. Sha256 20](#_Toc110806069)

[2.3.2.1. Programme en C 20](#_Toc110806070)

[2.3.2.2. Checksum et Sha512 21](#_Toc110806071)

[2.3.3. Installation de l'IDE Eclipse 21](#_Toc110806072)

[2.3.3.1. Configuration de l’environnement de développement 21](#_Toc110806073)

[3. Tests 21](#_Toc110806074)

[3.1. Bytecode 21](#_Toc110806075)

[3.2. Programme de test 21](#_Toc110806076)

[3.3. Junit 21](#_Toc110806077)

[4. De Java 8 à Java 18 : 8 années de nouvelles fonctionnalités 21](#_Toc110806078)

[4.1.1. Release table 22](#_Toc110806079)

[5. Critiques de la technologie Java 24](#_Toc110806080)

[6. Icones 24](#_Toc110806081)

[7. Links 24](#_Toc110806082)

# 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

#### Présentation du 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.

#### JRE versus JDK

* Le JRE est nécessaire, pour **tous type d’utilisateur,** pour faire tourner les applications Java ; tandis que le JDK est nécessaire, **au développeur,** pour coder des applications Java.
* Le JRE ne contient aucun outil de développement, contrairement au JDK.
* Les JRE sont disponibles pour une gamme de plates-formes beaucoup plus large que celle de JDK.
* JRE et JDK sont donc, **tous les deux** **dépendants** de la plateforme.

### JDK

Voyons la définition du JDK, ainsi que ses différents fournisseurs.

#### Définition du JDK

Prenons la définition de Wikipédia :

****Le Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être **compilé**, transformé en bytecode destiné à la machine virtuelle Java.

Il existe plusieurs éditions de JDK, selon la plate-forme Java considérée (et bien évidemment la version de Java ciblée) :

JSE pour la Java Standard Edition ;

JEE, sigle de Java Enterprise Edition ;

JME 'Micro Edition', destinée au marché mobiles .[[11]](#footnote-11)

Le JDK contient donc des outils permettant de développer des applications Java.

Ces principaux composants sont :

* **JRE** : environnement d’exécution des programmes Java, qui contient la **JVM**.
* Le compilateur : ***javac***
* L’interpréteur/chargeur : ***java***
* Le débogueur : ***jdb***
* Le désassembleur des fichiers .class : ***javap***
* L’archiveur : ***jar***
* Le générateur de documentation : ***javadoc***

On peut synthétiser ces trois notions : JVM, JRE et JDK à l’aide d’un schéma.

#### Classification de JDK JRE JVM



Figure Classification de JDK JRE JVM

****Il est possible d’installer plusieurs versions du JDK sur la même machine[[12]](#footnote-12).

Dans le cas où l’on désire tester les performances de chaque édition du JDK, par exemple.

## Le choix du JDK

Tous les JDK ne sont pas open source. Pour bien comprendre cette situation, il y a lieu de rappeler l’historique de Java, pour ensuite présenter les principales distributions des JDK, disponibles actuellement.

### Rappel du contexte historique de Java

Les principales dates qui ont marquées l’histoire de la technologie Java.

#### Timeline de Java

* 1982 Fondation de Sun Microsystems[[13]](#footnote-13), en Californie, par Andy Bechtolsheim, Bill Joy,Vinod Khosla et Scott McNealy.
* 1995 Lancement du langage Java, développé par James Gosling et Patrick Naughton au sein de Sun Microsystems.
* 1996 JDK 1.0
* 1997 JDK 1.1
* 1998 J2SE[[14]](#footnote-14) 1.2
* 2000 J2SE 1.3
* 2002 J2SE 1.4
* 2004 J2SE 5.0
* 2006 Java SE 6. Java devient open source. En effet Sun Microsystems publie le code source de Java sous la Licence publique générale GNU (GPL) .
* 2009 Sun Microsystems est rachetée[[15]](#footnote-15) par Oracle qui devient donc le nouveau propriétaire de Java.
* 2011 Java SE 7 est présenté par Oracle avec deux variantes du JDK
  + OpenJDK qui est open source. Il est maintenu et développé par Oracle, et permet aux communautés de développeurs et aux entreprises[[16]](#footnote-16) de participer à son développement.
  + Oracle JDK qui n’est pas open source.
    - Il est maintenu et développé par Oracle et est conforme aux spécifications OpenJDK.
    - Ce JDK d'Oracle fournit des extensions additionnels mais dont l'usage est payant.
    - Oracle JDK est publié sous le "Oracle Binary Code License Agreement".
* 2014 Java SE 8
* 2017 Java SE 9
* 2018-03 Java SE 10
* 2018-09 Java SE 11
* 2019-03 Java SE 12
* 2019-09 Java SE 13
* 2020-03 Java SE 14
* 2020-09 Java SE 15
* 2021-03 Java SE 16
* 2021-04 Microsoft[[17]](#footnote-17) sort sa propre distribution OpenJDK.
* 2021-09 Java SE 17

Pour développer en Java on a le choix entre des JDK open source ou soumis à une licence commerciale.

La version open source repose sur OpenJDK. En effet, OpenJDK est une implémentation officielle, gratuite et open source de la plate-forme Java SE, tel que défini par le Java Community Process[[18]](#footnote-18). et ce, depuis sa version 7. Il a été initialement publié en 2007 comme le résultat du développement que Sun Microsystems a commencé en 2006.

#### Les fournisseurs pour OpenJDK

Il existe de nombreux d’éditeurs pour OpenJDK. Ces éditions ne sont pas toutes open source. On a retenu les principales distributions.

|  |  |  |
| --- | --- | --- |
| **Fournisseur** | **Gratuite libre et open source** | **Site** |
| AdoptOpenJDK[[19]](#footnote-19) | **OUI** | <https://adoptium.net/temurin/releases> |
| Amazon Corretto | **OUI** | <https://aws.amazon.com/fr/corretto> |
| Azul Zulu | **OUI** | <https://www.azul.com/downloads> |
| IBM | **OUI** | <https://www.ibm.com/support/pages/java-sdk-downloads-eclipse> |
| Microsoft | **OUI** | <https://www.microsoft.com/openjdk> |
| OpenJDK | **OUI** | <https://openjdk.org> |
| Oracle JDK | **NON[[20]](#footnote-20)** | <https://www.oracle.com/java/technologies/downloads> |
| SapMachine | **OUI** | <https://sap.github.io/SapMachine> |

Tableau Principales distributions OpenJDK

On va choisir une version libre du JDK. Donc on optera pour une distribution de type openJDK, ou bien pour des fins de développement à une version d’Oracle.

Mais avant d’installer l’environnement de développement Java, il faut s’assurer de disposer du matériel adéquat.

# Installation

## Prérequis hardware

Si l’on veut pouvoir faire de la conception et du développement d’applications, que ce soit dans n’importe quel langage, il faut disposer d’une machine en conséquence.

Pour bien comprendre l’enjeu, un bref rappel technique est nécessaire.

### Architecture de l’ordinateur

#### Architecture de von Newman

Architecture de von Newman + biblio (Dev/D)

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

#### Architecture 32 bits versus 64 bits

L'autre grande différence se situe au niveau de la mémoire vive – la Ram, dans le jargon. En effet, un processeur 32 bits peut en théorie gérer 232 soit 4,29 milliards d'adresses mémoire, ce qui correspond à 4 Go de Ram. En pratique, pour diverses raisons techniques, un PC à architecture 32 bits n'exploite jamais ce maximum, même quand il dispose physiquement de 4 Go de mémoire vive – selon les cas, il en gère entre 2,8 et 3,5 Go. Certes, il existe quelques subterfuges techniques pour contourner cette limite, mais elles sont tendance à ralentir le système, ce qui n'est pas le but recherché. À l'inverse, un processeur 64 bits sait nativement gérer jusqu'à 264 adresses, soit quelque 18 milliards de giga-octets ! Autant dire qui n'y a pas de réelle limite matérielle pour le Ram en 64 bits.

Schéma

#### Pourquoi passer à un processeur 64 bits ?

biblio (Dev/D)

### Les trois types configurations hardware

## Procédure d’installation

►

## Verbatim !

►environnement de développement java

Une image contenant texte

Description générée automatiquement

►pas le jre , car il contient le jdk

**►java 1.8.0 = Java SE 8**

**► java 1.8.0 => système de versionning (officiel) utilisé dans la doc Oracle ou Sun**

**►openjdk et java pour system Unix like**

**►Eclipse**

**►Eclipse version JEE**

**►Workspace**

**Une image contenant texte

Description générée automatiquement**

**-welcome page Une image contenant texte

Description générée automatiquement**

**►Perspectives**

**Une image contenant texte

Description générée automatiquement**



►3 types de commentaires

Une image contenant texte

Description générée automatiquement

►Portabilité du code java

Compilation javac

Exécution java

### Installation du Java SE

<https://www.youtube.com/watch?v=y_Ye9-s7h1w>



<https://www.youtube.com/watch?v=CIuB3qwLdbk>

Une image contenant texte

Description générée automatiquement

Mm

Step by step

<https://www.youtube.com/watch?v=CIuB3qwLdbk>

On va installer Oracle JDK, puisque c’est à des fins de développement.

Téléchargement

Certutil

### Sha256

PowerShell

Pour comparer le Sha256 => faire un programme en C pour comparer 2 chaines de caractères.

#### Programme en C

#### Checksum et Sha512

F41047493698E0E77D6AA88D490656998EE4FD634776B040CA60C69621A36C8FFB02D3657994155475A09774839F030094B043B28B3132B6ED56831B05FD73F8

### Installation de l'IDE Eclipse

#### Configuration de l’environnement de développement

# Tests

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

## Bytecode

Décompilateur

***javap***

## Programme de test

## Junit

# De Java 8 à Java 18 : 8 années de nouvelles fonctionnalités

Graphe versions : majeures / autres

<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>

# 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)
11. <https://fr.wikipedia.org/wiki/Java_Development_Kit> [↑](#footnote-ref-11)
12. Si celle-ci dispose d’assez de ressources ! [↑](#footnote-ref-12)
13. <https://fr.wikipedia.org/wiki/Sun_Microsystems> [↑](#footnote-ref-13)
14. Java 2 Standard Edition [↑](#footnote-ref-14)
15. Pour un montant de 7 400 000 000 $. [↑](#footnote-ref-15)
16. Red Hat, Azul Systems, IBM, Apple, SAP, Microsoft, … [↑](#footnote-ref-16)
17. <https://devblogs.microsoft.com/java/microsoft-deepens-its-investments-in-java/> [↑](#footnote-ref-17)
18. <https://fr.wikipedia.org/wiki/Java_Community_Process> [↑](#footnote-ref-18)
19. Devenu Adoptium [↑](#footnote-ref-19)
20. Gratuit pour un usage personnel et non commercial [↑](#footnote-ref-20)