



# Programmation Orientée Objet Java



## Introduction

v1.1 - 20.02.2024



# Plan

---

- Java
  - C'est quoi ?
  - Bref historique
  - La plateforme Java
  - La Machine Virtuelle Java
  - Java : Quelques définitions
  - Le langage Java
  - Java : Spécificités du langage
  - Compilation
  - Execution
  - Debuggage
  - JAR
- La Programmation Orientée Objet (POO)
  - Bref historique
  - Concepts
  - Langages
  - Modélisation

# Java



# Java : C'est quoi ?

---

**Java** est le nom d'une technologie mise au point par Sun Microsystems (racheté par Oracle en 2010).

**Java** permet de produire des logiciels indépendants de toute architecture matérielle.

Cette technologie s'appuie sur différents éléments qui, par abus de langage, sont souvent tous appelés Java :

- Ne pas confondre avec JavaScript (JS), un langage de programmation "Web"
- Le **langage Java** est un langage de Programmation Orientée Objet
- Un programme compilé en **bytecode** Java s'exécute dans un environnement d'exécution Java (**JRE**) qui émule une machine virtuelle, dite **Java Virtual Machine**

# Java : Bref historique

- 1995 : JDK 1.0
- 1997 : JDK 1.1 (-> 1.1.8)
- 1998 : JDK 1.2 (-> 1.2.2)
- 2000 : J2SE 1.3 (-> 1.3.1)
- 2002 : J2SE 1.4 (Java -> 1.4.2)
- 2004 : J2SE 5 (Java 1.5)
- 2006 : Java SE 6 (Java 1.6) + passage en Open Source avec l'OpenJDK
- 2011 : Java SE 7 (Java 1.7)
- 2014 (mars) : Java SE 8 (Java 1.8) *support étendu jusqu'en janvier 2019*

Oracle indique sortir 2 releases majeures de Java par an (OpenJDK) : mars et septembre  
et une version LTS (Long Term Support) (OracleJDK) tous les 3 ans

- |  |  |
|--|--|
| • 2017 (septembre) : Java SE 9 (Java 1.9) <a href="#">[OpenJDK en mars 2018]</a> |  |
| • 2018 (mars) : Java SE 10 (Java 1.10) <a href="#">[OpenJDK en sept. 2018]</a>   | 2018 (septembre) : Java SE 11 (Java 1.11) -> LTS                                 |
| • 2019 (mars) : Java SE 12 (Java 1.12) <a href="#">[OpenJDK en sept. 2019]</a>   | 2019 (septembre) : Java SE 13 (Java 1.13) <a href="#">[OpenJDK en mars 2020]</a> |
| • 2020 (mars) : Java SE 14 (Java 1.14) <a href="#">[OpenJDK en sept. 2020]</a>   | 2020 (septembre) : Java SE 15 (Java 1.15) <a href="#">[OpenJDK en mars 2021]</a> |
| • 2021 (mars) : Java SE 16 (Java 1.16) <a href="#">[OpenJDK en sept. 2021]</a>   | 2021 (septembre) : Java SE 17 (Java 1.17) -> LTS                                 |
| • 2022 (mars) : Java SE 18 (Java 1.18)   | 2022 (septembre) : Java SE 19 (Java 1.19)  |
| • 2023 (22 mars) : Java SE 20 (Java 1.20)  | 2023 (septembre) : <b>Java SE 21</b> (Java 1.21)                                 |
| • 2024 (?? mars) : Java SE 22 (Java 1.22)  |  |

# L'évolution de Java

Plateforme	Java			J2SE			Java SE					
Version	1.0	1.1	1.2	1.3	1.4	5.0	6	7	8	9	10	11
Nombre de packages	8	23	59	76	135	166	202	209	217	315	314	223
Nombre de classes	201	503	1520	1840	2990	3280	3780	4024	4240	6005	6002	4410
Nombre de modules										98	99	71

[https://en.wikipedia.org/wiki/Java\\_version\\_history](https://en.wikipedia.org/wiki/Java_version_history)

<https://javapapers.com/core-java/java-features-and-history/>

Features :

- Java 8 : <https://javapapers.com/java/java-8-features/>
- Java 11 : <https://www.techgeeknext.com/java/java11-features>
- Java 15 : <https://www.techgeeknext.com/java/java15-features>
- Java 16 : <https://www.techgeeknext.com/java/java16-features>
- Java 17 : <https://www.techgeeknext.com/java/java17-features>

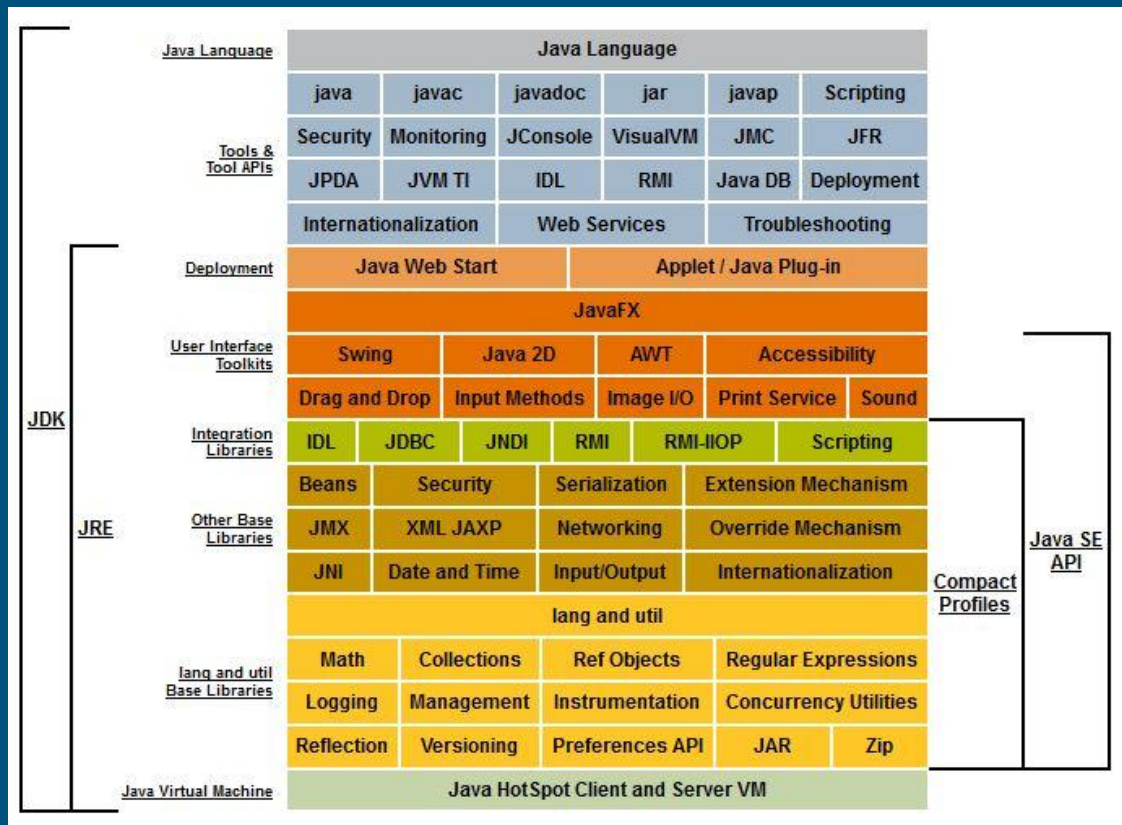
# La plateforme Java

---

La **plateforme Java** correspond à la **machine virtuelle Java** à laquelle sont adjointes diverses spécifications d'API :

- **Java** Platform, **Standard Edition (Java SE)** contient les API de base et est destiné aux ordinateurs de bureau,
- **Java** Platform, **Enterprise Edition (Java EE)** contient, en plus du précédent, des API orientées entreprise. Version destinée aux serveurs,
- **Java** Platform, **Micro Edition (Java ME)** est destiné aux appareils mobiles tels que assistants personnels ou smartphones,
- La **Java FX** Edition (ou **Java FX**) est orientée Rich Internet Application (RIA).

# La plateforme Java





# La Machine Virtuelle Java

---

La **Machine Virtuelle Java** [Java Virtual Machine=JVM], qu'est-ce ?

Elle permet d'exécuter des programmes écrits en langage Java indépendants de tout processeur et de tout système d'exploitation en interprétant le **bytecode** ou en le compilant à la volée en langage machine.

La portabilité est dépendante de la qualité de portage des JVM sur chaque OS. Si un appareil informatique embarque une JVM, un programme Java “pourra” s’y exécuter.

Les logiciels écrits en **Java** sont ainsi très facilement portables

# Java : Quelques définitions

---

## Bytecode ?

C'est le code généré par le compilateur Java (javac) et qui est exécuté par la machine virtuelle Java.

## JRE ?

Java Runtime Environment

## JDK ?

Java Development Kit = JRE + outils de développement

## ~~J2SE~~ -> Java SE ?

Java 2 Standard Edition (J2SE), plateforme avec les API et bibliothèques de bases, devenue Java SE

## ~~J2EE~~ -> Java EE ?

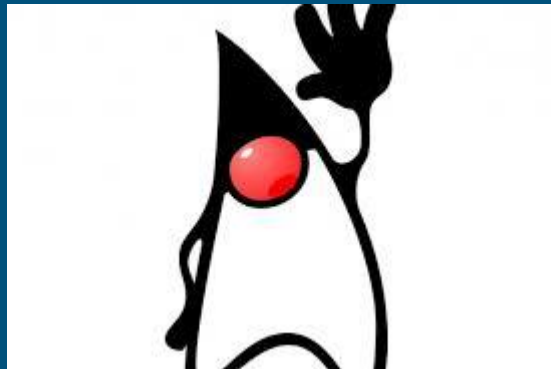
Java 2 Enterprise Edition (J2EE), extension avec des technologies pour le développement d'applications d'entreprise, devenue Java EE

# Le langage Java

---

## Le langage Java ?

C'est un langage de programmation informatique orienté objet créé par **James Gosling**, **Mike Sheridan** et **Patrick Naughton**, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.



Duke, la mascotte Java

# Le langage Java : Popularité

Classement PYPL (PopularitY of Programming Language)





















Classement des langages les plus “populaires” basé sur les recherches sur Internet

<http://pypl.github.io/PYPL.html>

Classement TIOBE

Tableau des 20  
langages  
informatiques les  
plus populaires

<https://www.tiobe.com/tiobe-index/>

Feb 2024	Feb 2023	Change	Programming Language	Ratings	Change
1	1		 Python	15.16%	-0.32%
2	2		 C	10.97%	-4.41%
3	3		 C++	10.53%	-3.40%
4	4		 Java	8.88%	-4.33%
5	5		 C#	7.53%	+1.15%
6	7	▲	 JavaScript	3.17%	+0.64%
7	8	▲	 SQL	1.82%	-0.30%
8	11	▲	 Go	1.73%	+0.61%
9	6	▼	 Visual Basic	1.52%	-2.62%
10	10		 PHP	1.51%	+0.21%
11	24	▲	 Fortran	1.40%	+0.82%
12	14	▲	 Delphi/Object Pascal	1.40%	+0.45%
13	13		 MATLAB	1.26%	+0.27%
14	9	▼	 Assembly language	1.19%	-0.19%
15	18	▲	 Scratch	1.18%	+0.42%
16	15	▼	 Swift	1.16%	+0.23%
17	33	▲	 Kotlin	1.07%	+0.76%
18	20	▲	 Rust	1.05%	+0.35%
19	30	▲	 COBOL	1.01%	+0.60%
20	16	▼	 Ruby	0.99%	+0.17%

Worldwide, Feb 2024 :				
Rank	Change	Language	Share	1-year trend
1		Python	28.11 %	+0.6 %
2		Java	15.52 %	-1.0 %
3		JavaScript	8.57 %	-1.0 %
4	▲	C/C++	6.92 %	+0.1 %
5	▼	C#	6.73 %	-0.1 %
6	▲	R	4.75 %	+0.7 %
7	▼	PHP	4.57 %	-0.6 %
8		TypeScript	2.78 %	-0.0 %
9		Swift	2.75 %	+0.5 %
10		Objective-C	2.37 %	+0.1 %
11		Rust	2.23 %	+0.3 %
12		Go	2.04 %	+0.1 %
13		Kotlin	1.75 %	-0.1 %
14		Matlab	1.64 %	-0.1 %
15	▲▲	Ada	1.08 %	+0.2 %
16	▲▲▲	Dart	0.98 %	+0.2 %
17	▼	Ruby	0.98 %	-0.0 %
18		Powershell	0.94 %	+0.1 %
19	▼▼▼▼	VBA	0.88 %	-0.1 %
20	▲▲	Lua	0.85 %	+0.3 %
21	▼	Scala	0.59 %	-0.0 %
22	▲	Abap	0.57 %	+0.1 %
23	▲	Julia	0.56 %	+0.1 %
24	▼▼▼	Visual Basic	0.51 %	-0.1 %
25		Perl	0.33 %	-0.0 %
26		Groovy	0.33 %	-0.0 %
27	▲	Haskell	0.26 %	-0.1 %
28	▼	Cobol	0.22 %	-0.1 %
29		Delphi/Pascal	0.19 %	+0.2 %

© Pierre Carboneille, 2023

# Java : Spécificités du langage

---

Le **langage Java** et ses caractéristiques :

- Langage Orienté Objet
- Compilé mais interprété dans une JVM  
-> Slogan SUN : **WORA** = “**W**rite **O**nce, **R**un **A**newhere”
- Typage fort et statique
- Pas de gestion mémoire implicite (pas de notion de pointeur)
- Héritage simple (pas d'héritage multiple)

**James Gosling** décrit le **langage Java** comme “A simple, object oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high performance, multithreaded, dynamic language.”

# Java : Compilation

---

La compilation est l'étape qui consiste à transformer un "code source" en un code machine (binaire) compréhensible par l'ordinateur.

Cette étape est réalisée par un outil nommé "compilateur".

En Java, le compilateur est **javac** (java compiler)

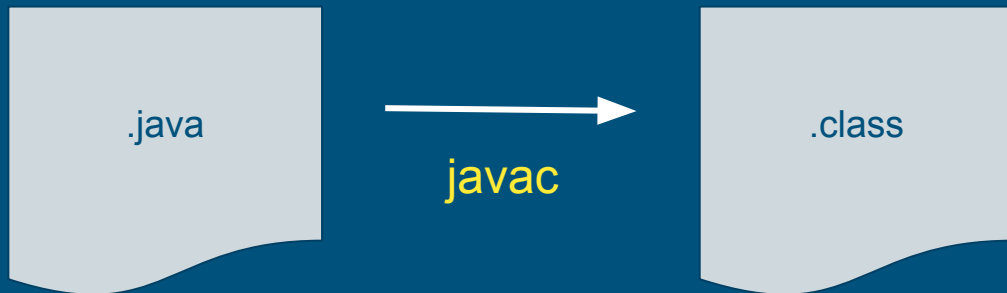
MAIS il ne compile pas en langage machine interprétable directement par l'OS, mais en **bytecode**, un meta langage interprété par la JVM (code assembleur de la JVM).

# Java : Compilation

---

L'outil **javac** a donc la lourde tâche :

- d'analyser le code Java,
- vérifier sa validité,
- le compiler -> transformer chaque fichier **.java** en fichier **.class**



# Java et Bytecode

---

## Code Java

```
public static void main(String[] args) {  
    int a = 1;  
    int b = 2;  
    int c = a + b;  
}
```

## Bytecode

```
public static void main(java.lang.String[]);  
descriptor: ([Ljava/lang/String;)V  
flags: (0x0009) ACC_PUBLIC, ACC_STATIC  
Code:  
stack=2, locals=4, args_size=1  
0: iconst_1  
1: istore_1  
2: iconst_2  
3: istore_2  
4: iload_1  
5: iload_2  
6: iadd  
7: istore_3  
8: return  
...
```



# Exécution

---

L'exécution d'un programme Java après sa compilation est réalisée par la JVM (Java Virtual Machine).

La JVM exécute du bytecode généré par le compilateur.

L'outil permettant de lancer un programme Java (une fois compilé), s'appelle `java`.

# Le Classpath

---

Le **classpath** permet de préciser au compilateur et à la JVM où ils peuvent trouver les classes dont ils ont besoin pour la compilation et l'exécution d'une application.

Il est défini par un ensemble de répertoires ou fichiers **.jar** dans lesquels rechercher les classes nécessaires.

Le classpath peut être défini à plusieurs niveaux :

1. Au niveau global : il faut utiliser la variable d'environnement système CLASSPATH
2. Au niveau spécifique : il faut utiliser l'option **-classpath** pour le compilateur
3. Au sein d'un script de lancement

Depuis **Java 9** (Jigsaw) avec l'apparition des modules, on peut définir un **module-path**.

# Exécution : Performance ?

---

Java n'est pas un langage ayant de très bonnes performances d'exécution.

Mais, du fait de son exécution sous forme de **bytecode** par une JVM, celui-ci est moins performant que du code natif compilé en langage machine (comme avec des langages tels que C ou C++).

Par ailleurs, le lancement de la JVM (obligatoire pour exécuter une application Java) prend également du temps.

# Exécution : Mémoire ?

---

Java est consommateur de mémoire du fait de l'utilisation d'une JVM qui doit être démarrée pour exécuter une application Java.

# Debogage

---

Il s'agit d'exécuter le programme Java en **mode debug**, c'est à dire dans un mode d'exécution contrôlée où il va être possible au développeur de :

- Exécuter le code en mode pas à pas
- Mettre des points d'arrêts (breakpoints)
- Inspecter l'état des variables
- ...

L'outil fourni avec le JDK pour faire du débogage est **jdb**

# JAR : Java ARchive

---

L'ensemble des fichiers d'une application Java (.class et autres) peuvent être regroupés dans un **fichier JAR** (.jar).

Un fichier **JAR** est en fait un fichier Zip qui peut être créé grâce à l'outil **jar**

# La Programmation Orientée Objet

---

# P00 : Bref historique

---

L'émergence de langage de P00 a lieu dans les années 1980.

Le 1er langage avec des notions "objet" est **Simula** (dans les années 60).  
Mais vraiment introduite avec le langage **SmallTalk** en 1972.



# POO : Concepts

---

- Classe
- Objet et Instance
- Les données : Propriétés/Attributs (fields)
- Les traitements/opérations : Méthodes (methods)
- Encapsulation
- Composition
- Héritage et surcharge
- Typage et Polymorphisme
- Abstraction

# POO : Langages

---

Plusieurs langages de programmation permettent de faire de la POO :

- **Java** (Groovy, Scala, Kotlin, Clojure, ...)
- C#
- C++
- Dart
- Object Pascal
- Objective-C
- Perl
- PHP
- Python
- Ruby
- Swift
- ...

Exemples de langages **NON Orientés Objets** :

- C
- Cobol
- Fortran
- Go
- Pascal



Exemple de langages **orientés objets à prototypes** :

- Javascript
- Lua

# POO : Modélisation

---

Plusieurs langages de modélisation permettent de décrire/schématiser un modèle objet.

Le plus connu est **UML** (Unified Modeling Language).

Il permet de décrire plusieurs types (14) de diagrammes :

- Diagrammes de structure
  - Diagramme de classes
  - Diagramme d'objets
  - ...
- Diagrammes de comportement/interaction
  - Diagramme des cas d'utilisation
  - Diagramme de séquence
  - ...



# Questions/Réponses

---