

Introduction à Java



dernière modification : 11/12/2024 01:09

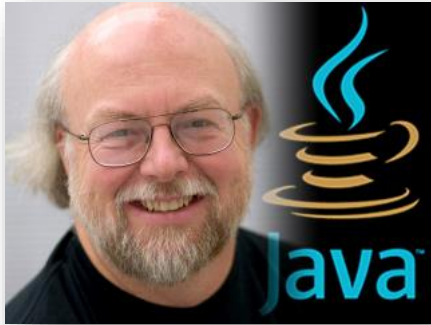


This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

© Ph. Genoud – Université Grenoble Alpes

JAVA c'est quoi ?

- Une technologie développée par SUN Microsystems™ lancée en 1995 - rachetée par Oracle en 2009
 - Un langage de programmation
 - Dans un des premiers papiers* sur le langage JAVA, James Gosling et Henry Mc Gilton le décrivent comme suit :



« *Java : a simple, **object-oriented**, distributed, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language* »

* White Paper :The Java Language Environment - James Gosling, Henry McGilton - May 1996
<https://www.oracle.com/technetwork/java/langenv-140151.html>

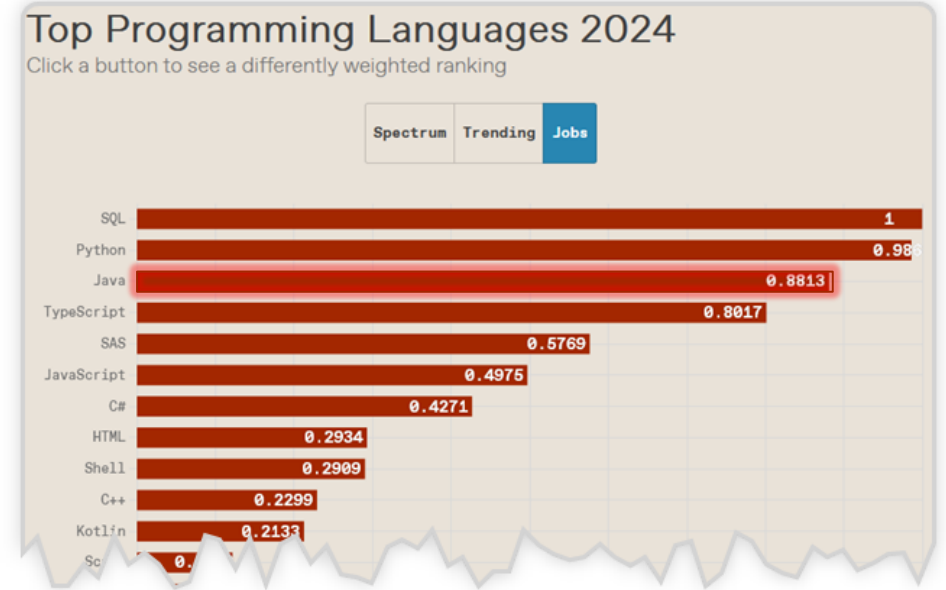
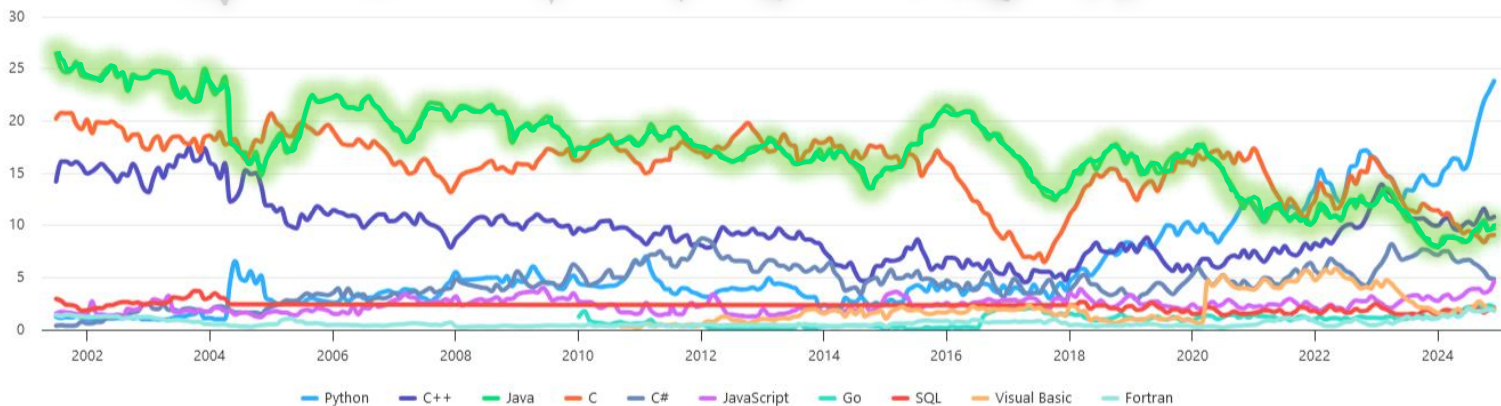
- Une plateforme, environnement logiciel dans lequel les programmes java s'exécutent.
 - Machine Virtuelle Java (JVM)
 - Environnement d'exécution (Java Run Time Environment)

JAVA c'est quoi ?

- Technologie présente dans de très nombreux domaines d'application : des serveurs d'applications aux téléphones portables et cartes à puces
- Dominante dans les années 2000, elle demeure toujours d'actualité

TIOBE index Dec. 2024 [↗](#)

Dec 2024	Dec 2023	Change	Programming Language	Ratings	Change
1	1		Python	23.84%	+9.98%
2	3	▲	C++	10.82%	+0.81%
3	4	▲	Java	9.72%	+1.73%
4	2	▼	C	9.10%	-2.34%
5	5		C#	4.87%	-2.43%
6					+1.7%



Stackoverflow developer survey 2024 [↗](#)




JAVA c'est quoi ?

- PYPL <https://pypl.github.io/PYPL.html>



- https://github.com/readme/featured/java-programming-language?utm_source=github&utm_medium=referral&utm_campaign=&scid=&utm_content=octoverse



Don't call it a comeback: Why Java is still champ

FAR FROM DEAD, THE PERPETUALLY-POPULAR LANGUAGE IS UP TO SPEED AND READY FOR THE FUTURE.

MIKE MELANSON // AUGUST 9, 2022

Le langage Java

Mon premier programme Java (pas très objet...)

Le code de la classe doit être enregistré dans un fichier de même nom (casse comprise) que la classe `HelloWorld.java`



3

1

Tout code java doit être défini à l'intérieur d'une classe

Le point d'entrée pour l'exécution est la méthode `main()`

4

5

Java est un langage **typé statiquement** (toute variable ou fonction doit être typée) contrairement à JavaScript (typage dynamique)

```
public class HelloWorld {  
    public static void main(String[] args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.println("Hello World !");  
            i++;  
        }  
    }  
}
```

2

La description de la classe est effectuée à l'intérieur d'un bloc `{ }`

Compilation :

`javac HelloWorld.java`

Exécution :

`java HelloWorld`



`HelloWorld.java`

`javac`



`HelloWorld.class`

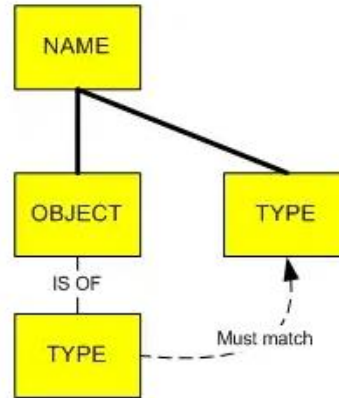
`java`

```
Hello World !  
Hello World !  
Hello World !  
Hello World !  
Hello World !
```

Typage statique/Typage dynamique

typage statique

- lorsqu'une variable est créée un type lui est associé
- le type de la variable ne peut être changé
- seules des valeurs compatibles avec le type de la variable peuvent lui être affectées



java

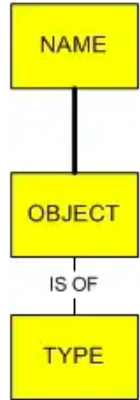
```
int number;  
✓ number = 10;  
✗ number = "hello";  
number = false;
```

autres exemples :
C, C++, C#, FORTRAN...

- nécessite en général déclaration explicite des types
- vérification des types peut être faite avant l'exécution (compilation) → diminue risques d'erreurs d'exécution
- permet de produire du code plus optimisé (au moment de l'exécution car la vérification de type a déjà été effectuée)

typage dynamique

- le type d'une variable est associé à la valeur qui lui est affectée
- le type de la variable peut être changé au cours de l'exécution



JavaScript

```
let number;  
✓ number = 10;  
number = "hello";  
number = false;
```

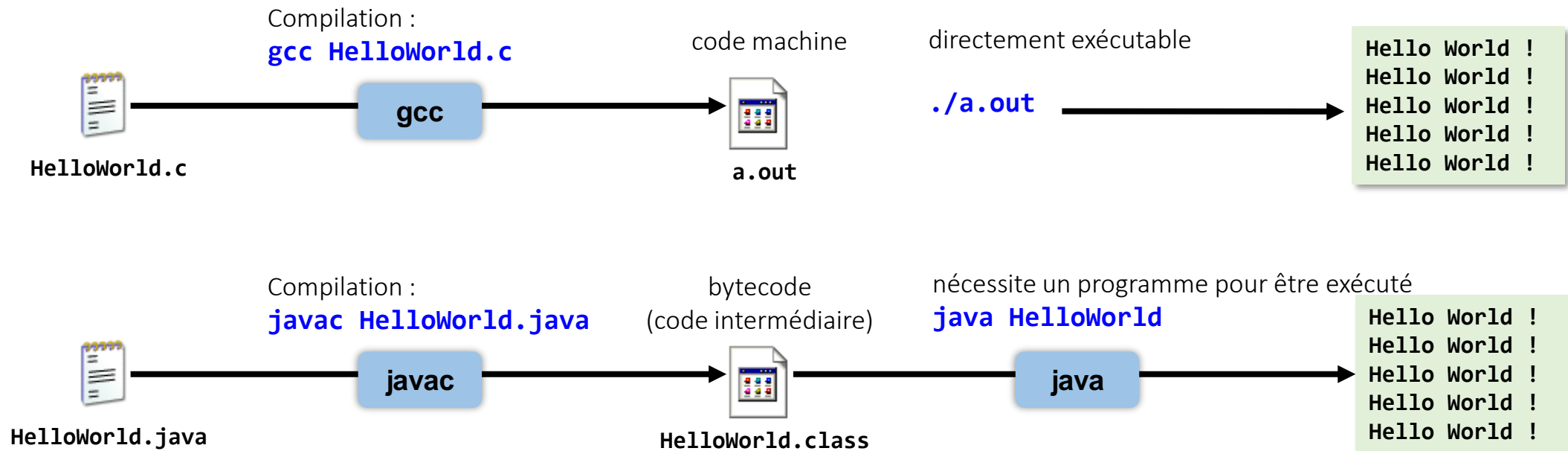
autres exemples :
Python, Ruby, PHP, Lisp...

- ne nécessite pas de déclaration explicite des types
- vérification des types faite à l'exécution → risque d'erreurs à l'exécution
- tendance à une exécution plus lente car les informations de type pour chaque variable doivent être récupérées au moment de l'exécution.

Le langage Java

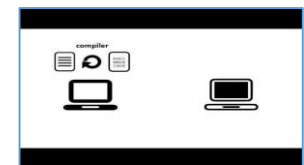
Un langage compilé / interprété

- Contrairement à d'autres langages compilés (C, FORTRAN, Pascal...) le code java compilé n'est pas directement exécutable, il est exécuté au travers d'un autre programme : **java**



<https://www.youtube.com/watch?v=11f45REi3k4>

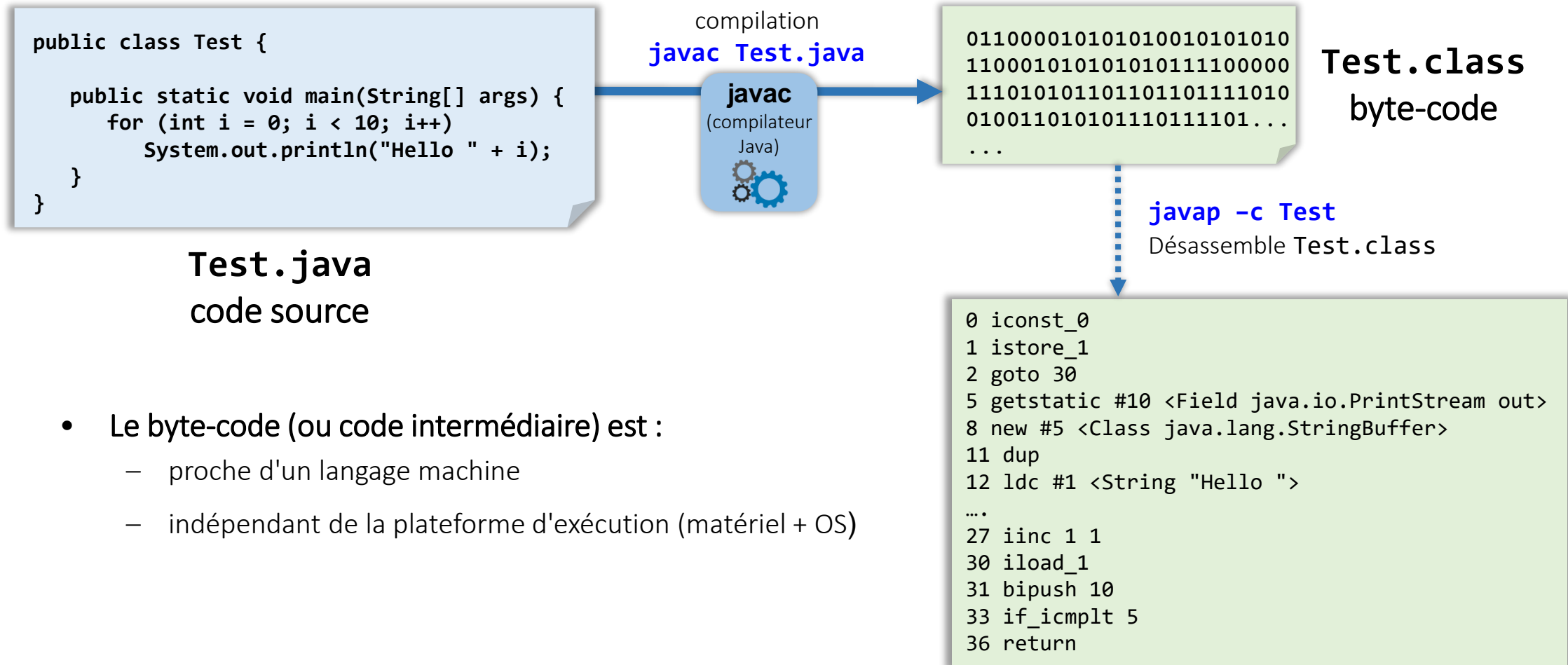
Compiler and Interpreter: Compiled Language vs Interpreted Programming Languages
durée 6 min



Le langage Java

Un langage compilé / interprété

- Compilation d'un programme JAVA : génération de byte-code



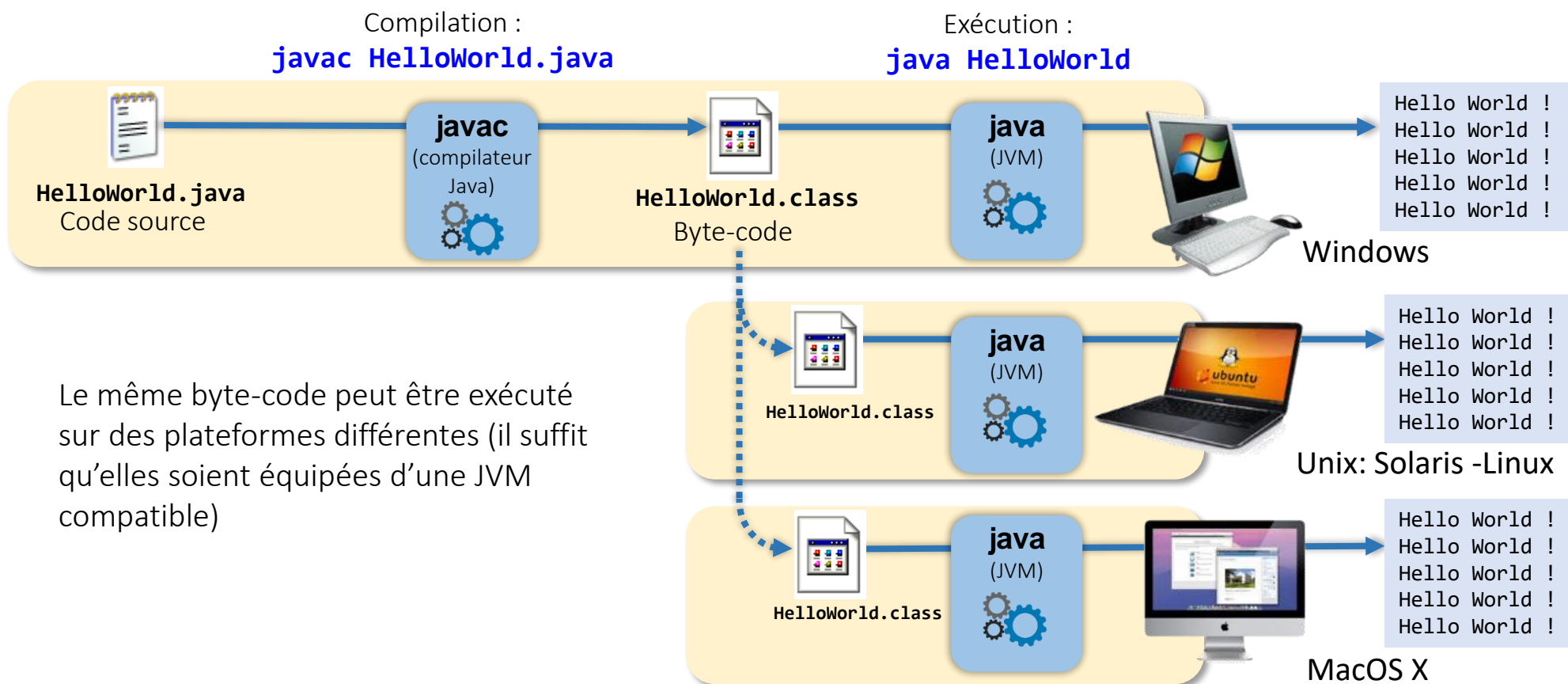
- Le byte-code (ou code intermédiaire) est :
 - proche d'un langage machine
 - indépendant de la plateforme d'exécution (matériel + OS)

La machine virtuelle Java

Exécution d'un programme Java compilé



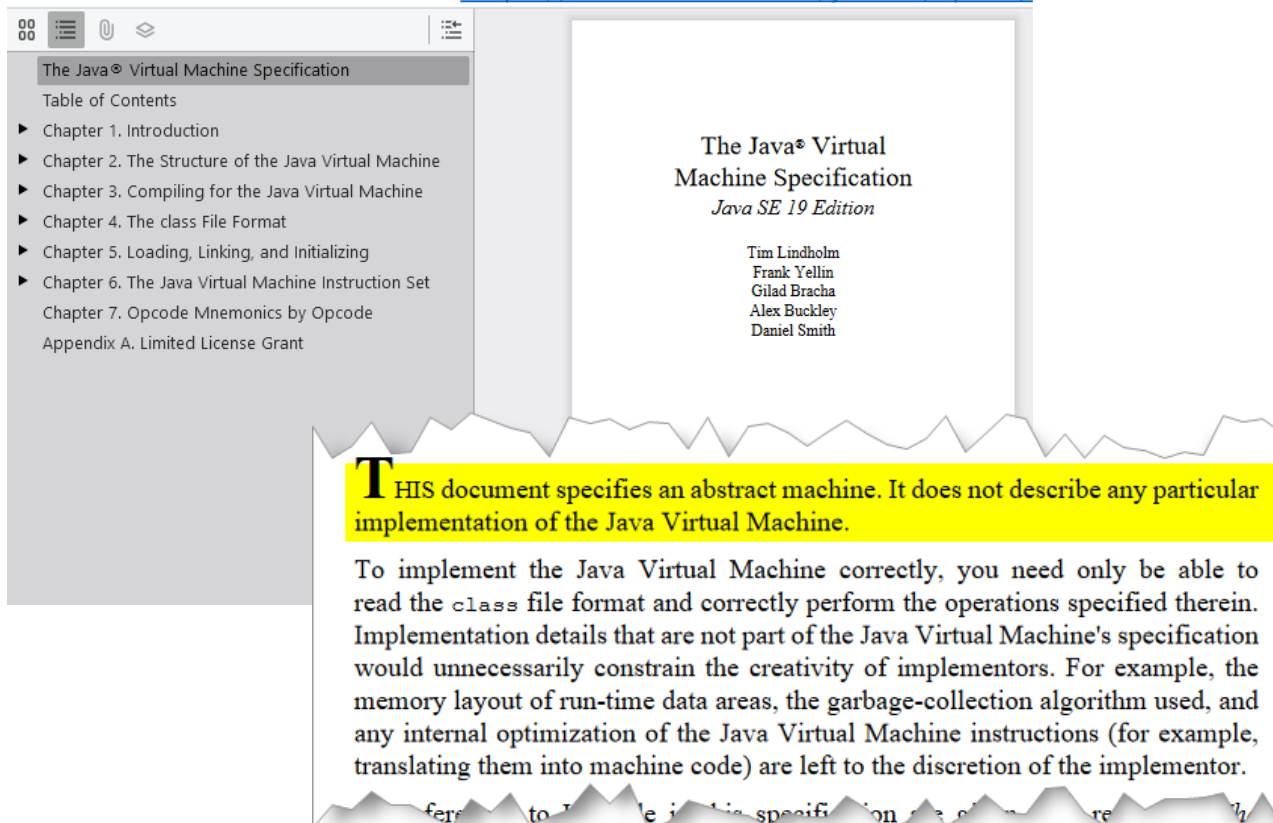
- byte-code assure la portabilité des programmes Java
 - langage d'une Machine Virtuelle
 - à l'exécution un interpréteur (JVM) simule cette machine virtuelle



La machine virtuelle java

- fonctions
 - permettre d'exécuter des programmes java sur n'importe quel environnement (matériel + système d'exploitation)
 - gérer et optimiser l'utilisation de la mémoire
- une spécification de la JVM

<https://docs.oracle.com/javase/specs/>



- différentes implémentations (open source ou propriétaires)
 - HotSpot (Open JDK) : implémentation de référence
 - GraalVM (Oracle)
 - Eclipse OpenJ9 (IBM)
 - Azul Zing
 - ...

<https://www.fineconstant.com/posts/comparing-jvm-performance/>

La machine virtuelle java

Utilisation pour d'autres langages

- Alors qu'elle était autrefois réservée à Java, la JVM est aujourd'hui suffisamment flexible et puissante pour prendre en charge de nombreux autres langages.

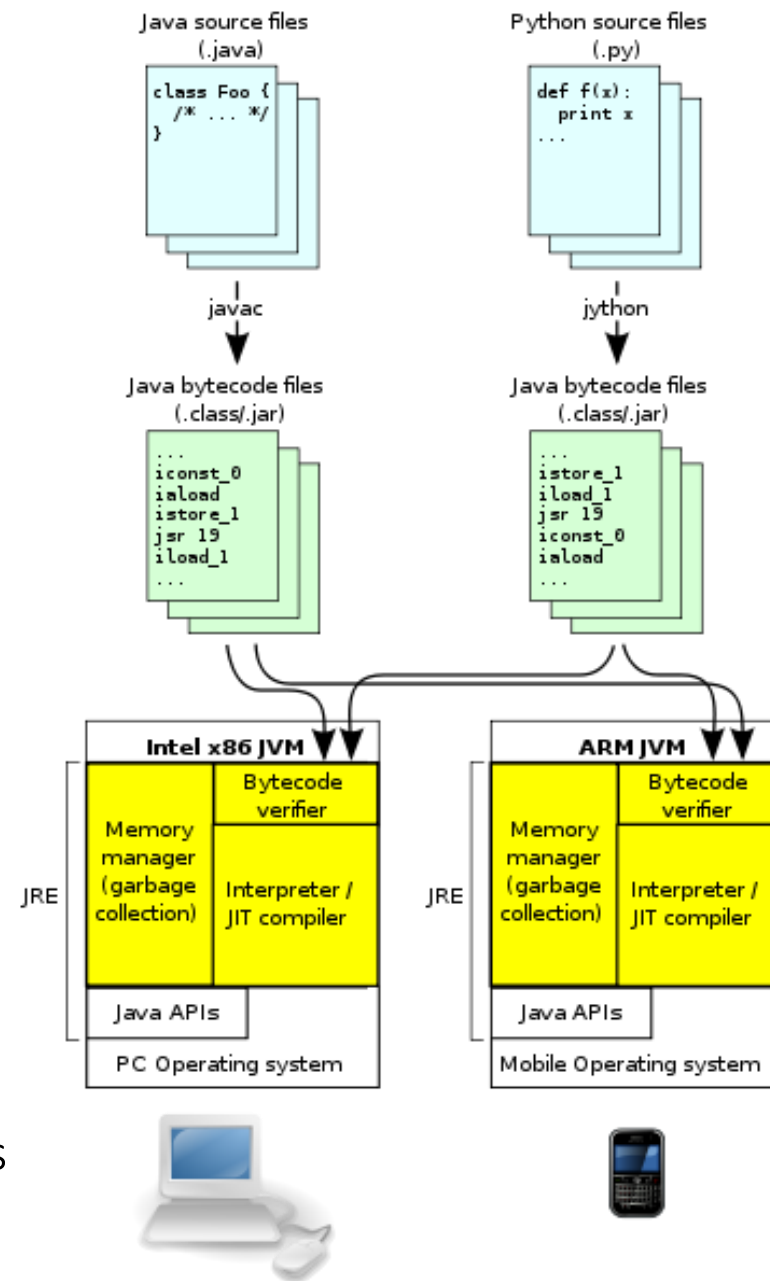
Versions of non-JVM languages

Language	On JVM
Erlang	Erjang
JavaScript	Rhino
Pascal	Free Pascal
PHP	Quercus
Python	Jython
REXX	NetRexx ^[3]
Ruby	JRuby
Tcl	Jacl

Languages designed expressly for JVM

Language
BBj
Clojure
Fantom
Groovy
Kotlin
MIDletPascal
Scala
Kawa

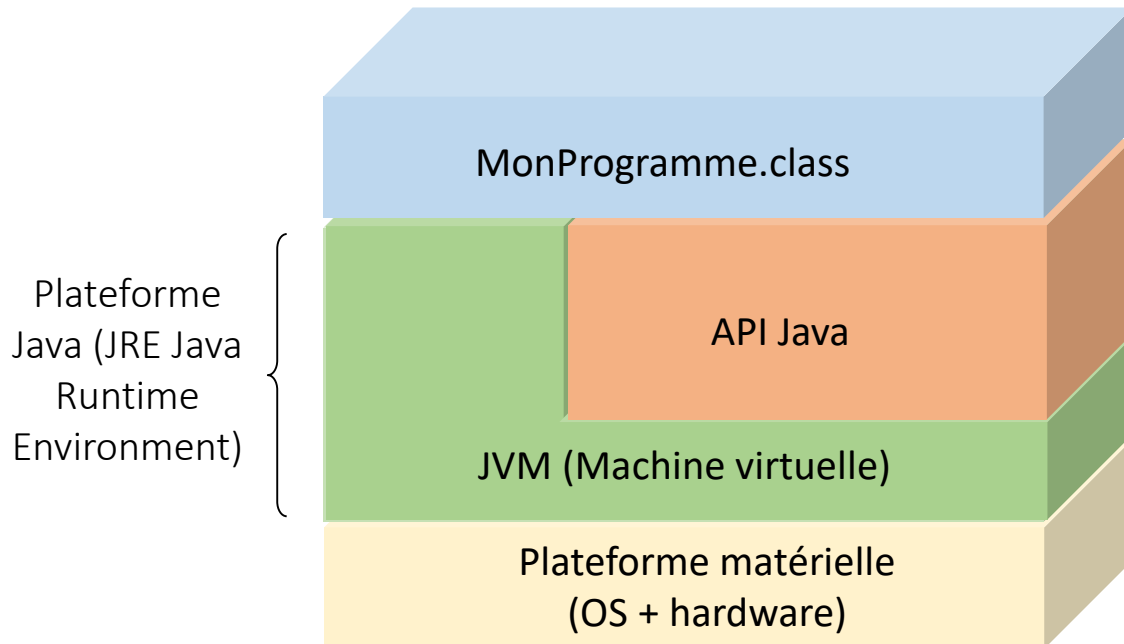
- même s'ils ne codent pas en Java, les programmeur conservent l'accès au très vaste écosystème de bibliothèques Java.



https://en.wikipedia.org/wiki/Da_Vinci_Machine

La plateforme Java

- Plateforme
 - Environnement matériel et/ou logiciel dans lequel un programme s'exécute.
 - La plupart des plateformes sont la combinaison d'un OS et du matériel sous-jacent (*MS Windows + Intel, Linux + Intel, Solaris + Sparc, Mac Os X + Intel*)
- La plateforme Java est entièrement logicielle et s'exécute au dessus des plateformes matérielles

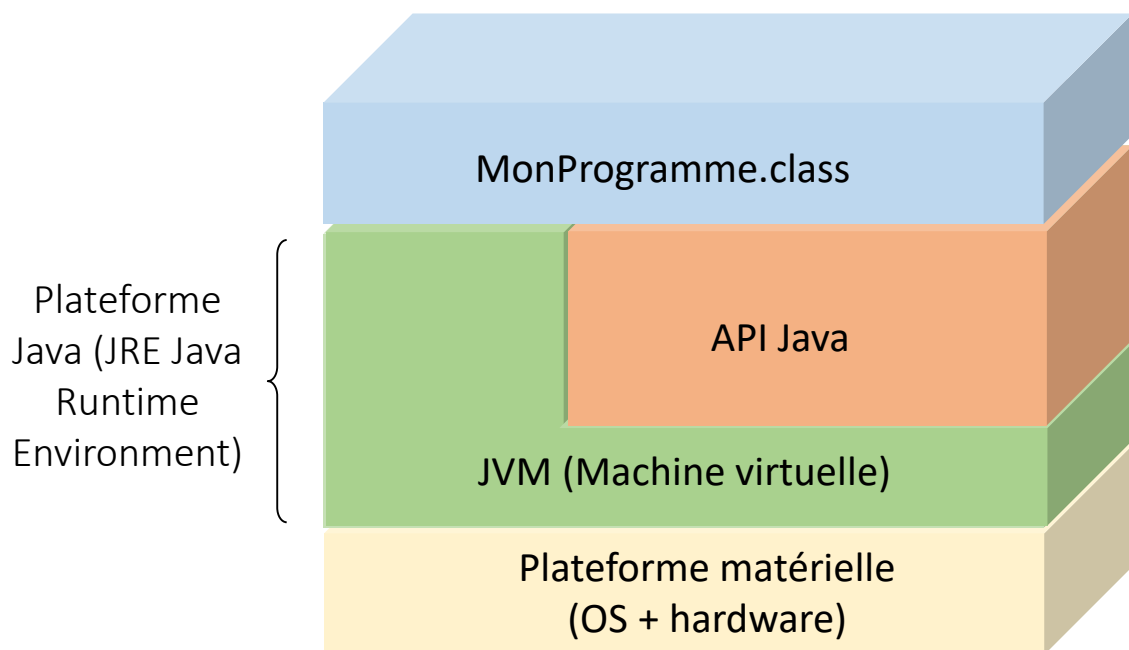


API (Application Programming Interface) Java :
Bibliothèques Java standards sur lesquelles le programmeur peut s'appuyer pour écrire son code.

La plateforme Java

API Java

- API Java
 - (très) vaste collection de composants logiciels (classes et interfaces)
 - organisée en bibliothèques (*packages*)
 - offre de nombreux services de manière standard (indépendamment de la plateforme matérielle)



Swing	Java 2D	AWT	Accessibility		User Interface Toolkits	
Drag and Drop	Input Methods	Image I/O	Print Service	Sound		
IDL	JDBC	JNDI	RMI	RMI-IIOP	Scripting	Integration librairies
Beans	Security	Serialization	Extension Mechanism		Other Base Libraries	
JMX	XML JAXP	Networking	Override Mechanism			
JNI	Date and Time	Input/Output	Internationalization			
lang and util						
Math	Collections	Ref Objects	Regular Expressions		Lang and util base libraries	
Logging	Management	Instrumentation	Concurrency Utilities			
Reflection	Versioning	Preferences API	JAR	Zip		



Programmer en Java nécessite une bonne connaissance de l'API. Attention à la courbe d'apprentissage (learning curve) qui est peut être longue

La plateforme Java

Les différentes éditions de Java

- 3 éditions de Java



Standard Edition JSE

Fourni les compilateurs, outils, runtimes, et APIs standards pour écrire, déployer, et exécuter des applications dans la langage de programmation Java



Enterprise Edition JEE

Destinée au développement d'applications « d'entreprise » («*business applications*») robustes et interopérables.

Construit au dessus de JSE, elle propose des APIs supplémentaires pour le développement et le déploiement d'applications distribuées et articulées autour du web.



Micro Edition JME

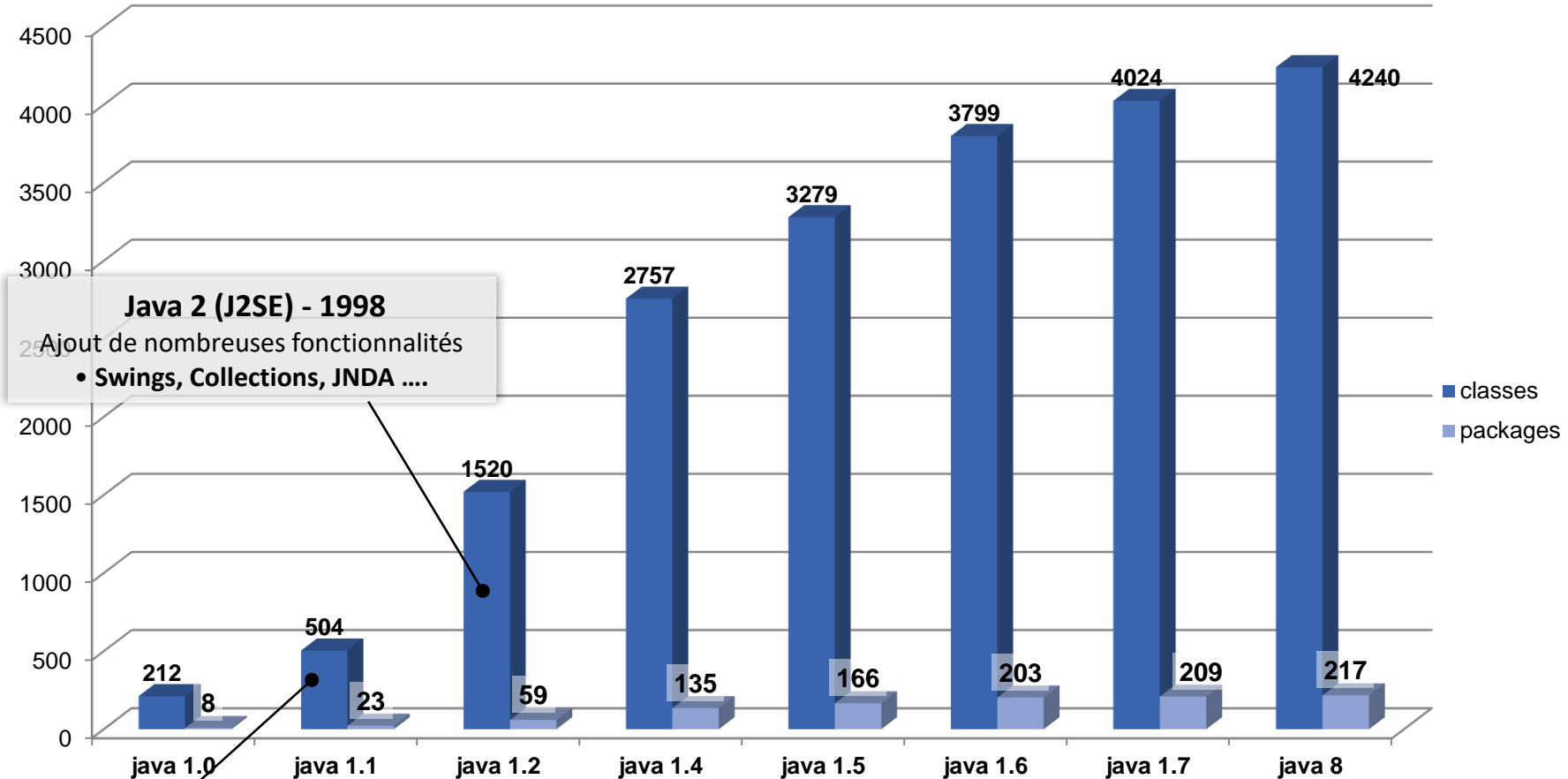
Environnement d'exécution basé sur un ensemble réduit d'API et optimisé pour les dispositifs « légers » :

- Carte à puce (smart cards)
- Téléphones mobiles
- Objets connectés



La plateforme Java JSE

Evolution de l'API

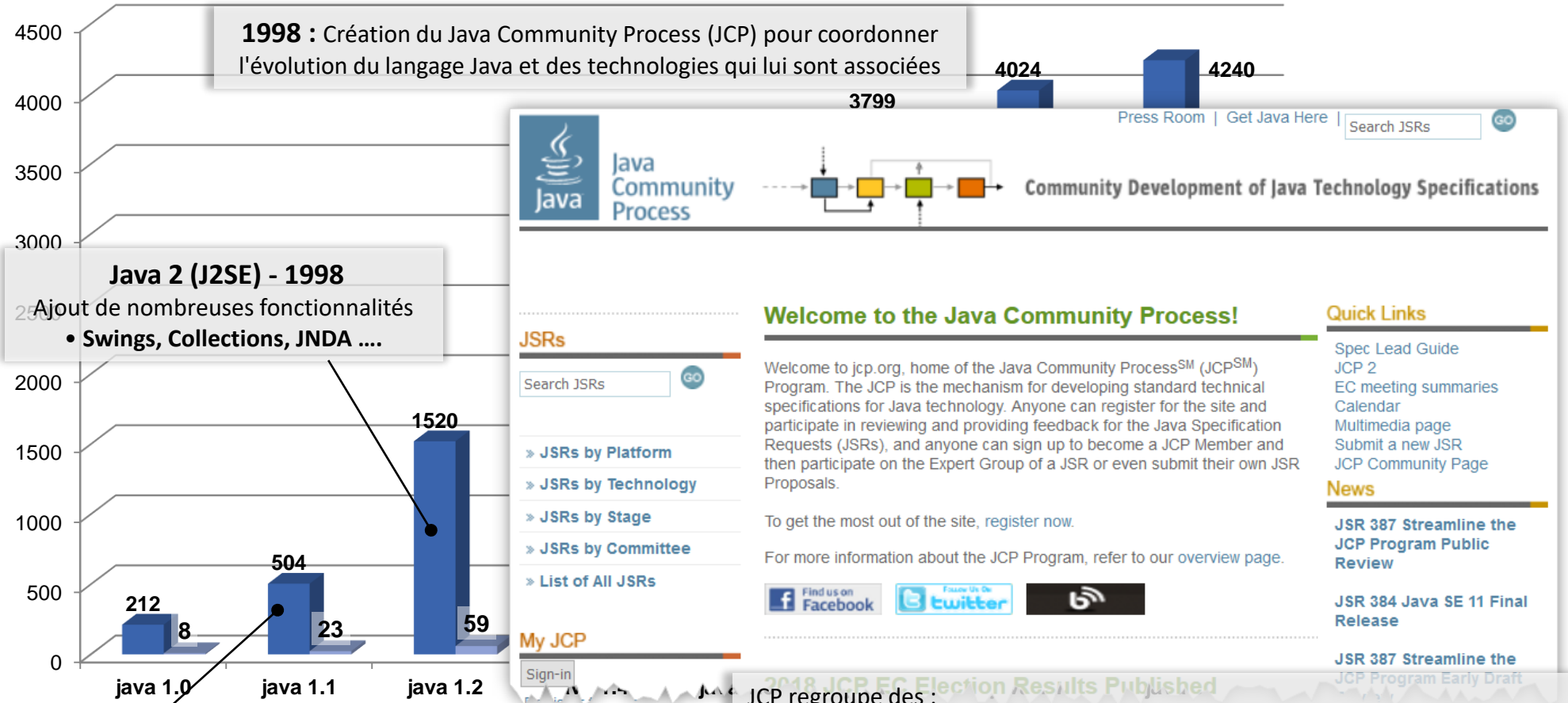


Java 2 (J2SE) - 1998
Ajout de nombreuses fonctionnalités
• Swings, Collections, JNDA

Java 1.1 – 1997
• Modification du langage : classes internes
• Modification du modèle événementiel pour awt
Java Beans
• Amélioration des performances de la VM

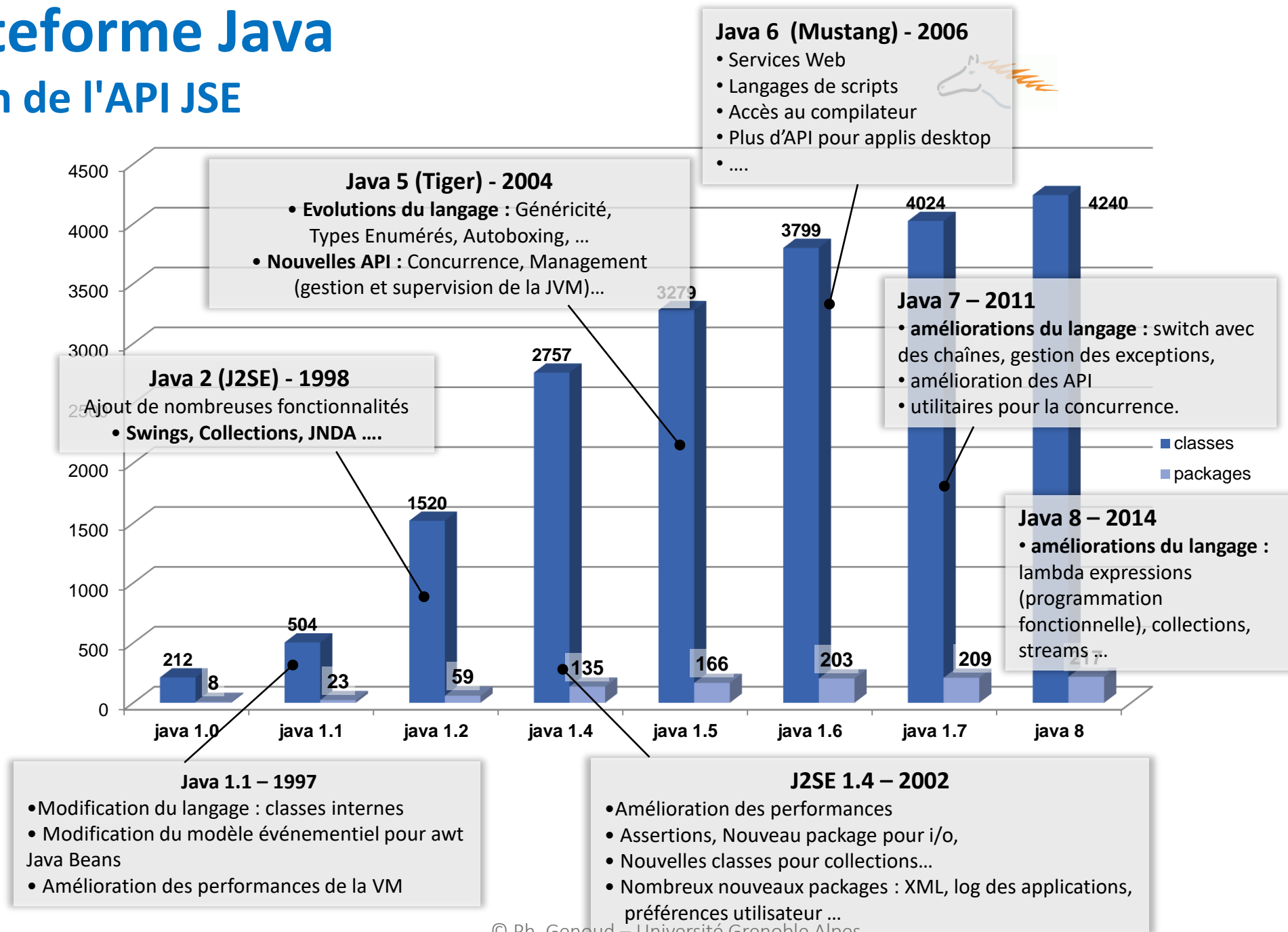
La plateforme Java

Evolution de l'API JSE



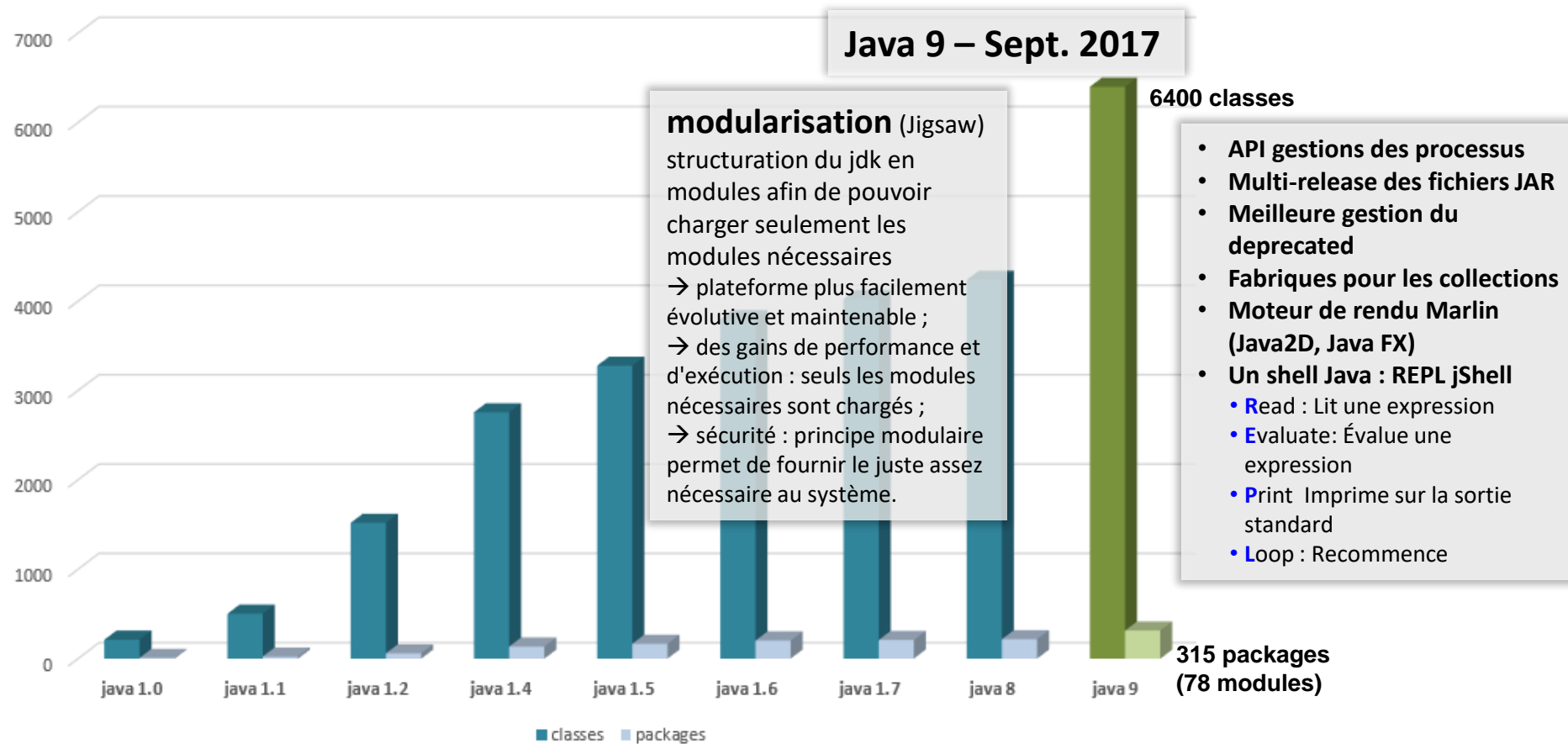
La plateforme Java

Evolution de l'API JSE



La plateforme Java

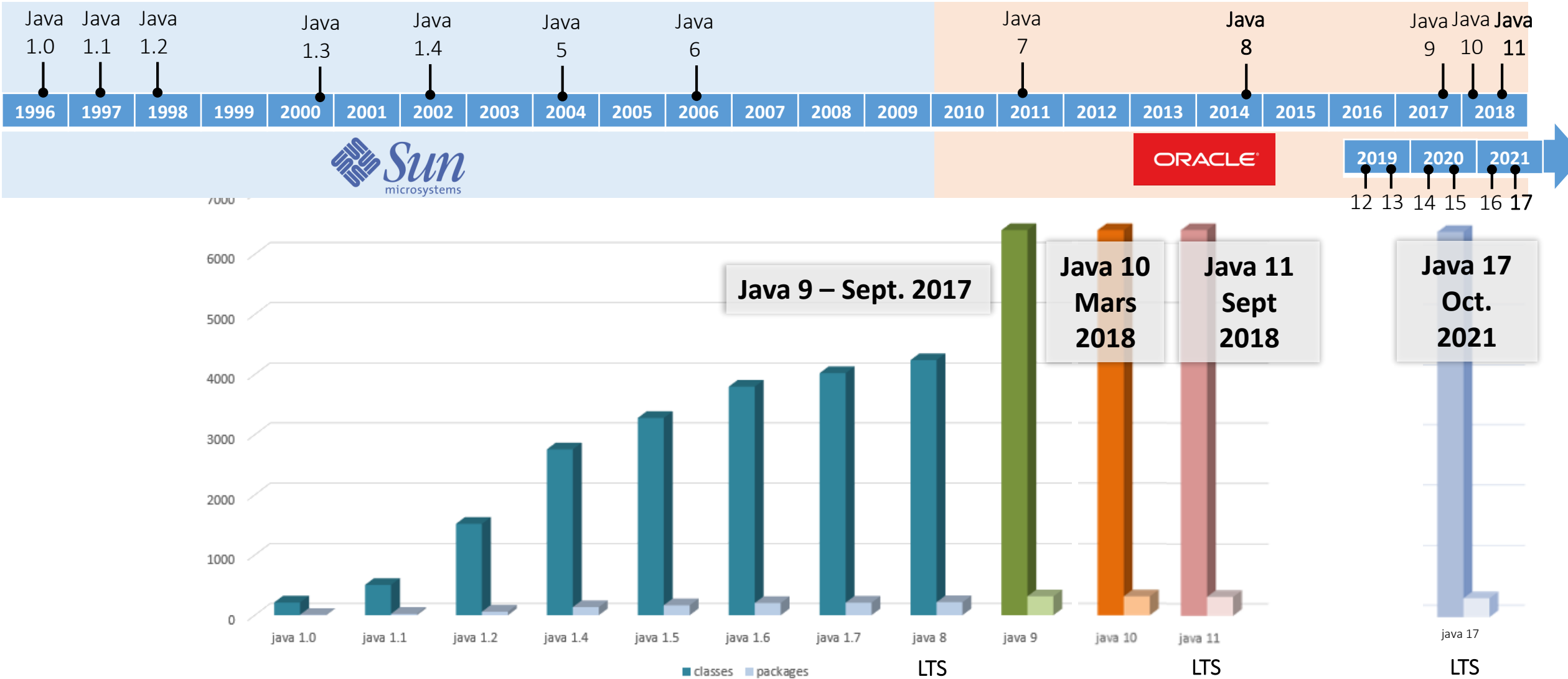
Evolution de l'API JSE



La plateforme Java

Evolution de l'API JSE

Depuis Java 9, Oracle JCP adopte une politique de release tous les 6 mois. Seules les releases LTS (Long Term Support) sont maintenues sur plusieurs années



JSE : Java Standard Edition

Organisation générale

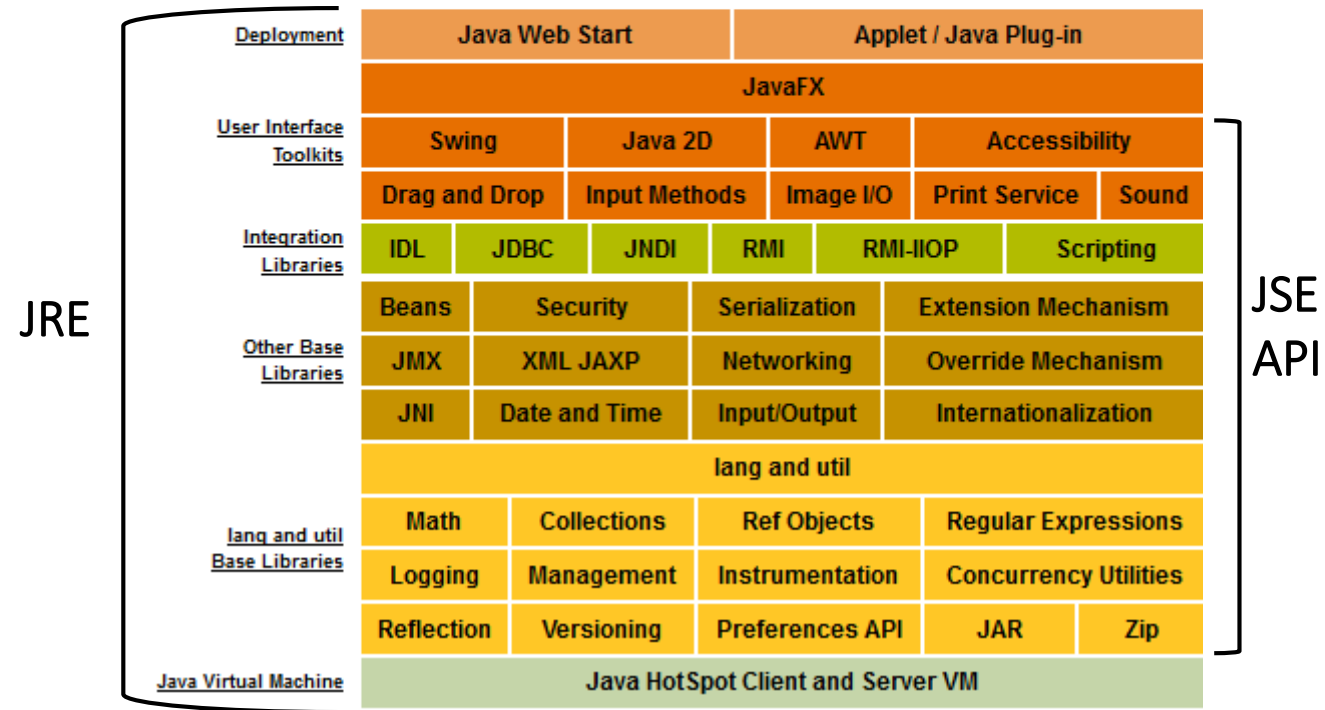
<u>User Interface Toolkits</u>	Swing	Java 2D	AWT	Accessibility		
	Drag and Drop	Input Methods	Image I/O	Print Service	Sound	
<u>Integration Libraries</u>	IDL	JDBC	JNDI	RMI	RMI-IIOP	Scripting
<u>Other Base Libraries</u>	Beans	Security	Serialization	Extension Mechanism		
	JMX	XML JAXP	Networking	Override Mechanism		
	JNI	Date and Time	Input/Output	Internationalization		
	lang and util					
<u>lang and util Base Libraries</u>	Math	Collections	Ref Objects	Regular Expressions		
	Logging	Management	Instrumentation	Concurrency Utilities		
	Reflection	Versioning	Preferences API	JAR	Zip	

JSE API

JSE API : Java Application Programming interface

JSE : Java Standard Edition

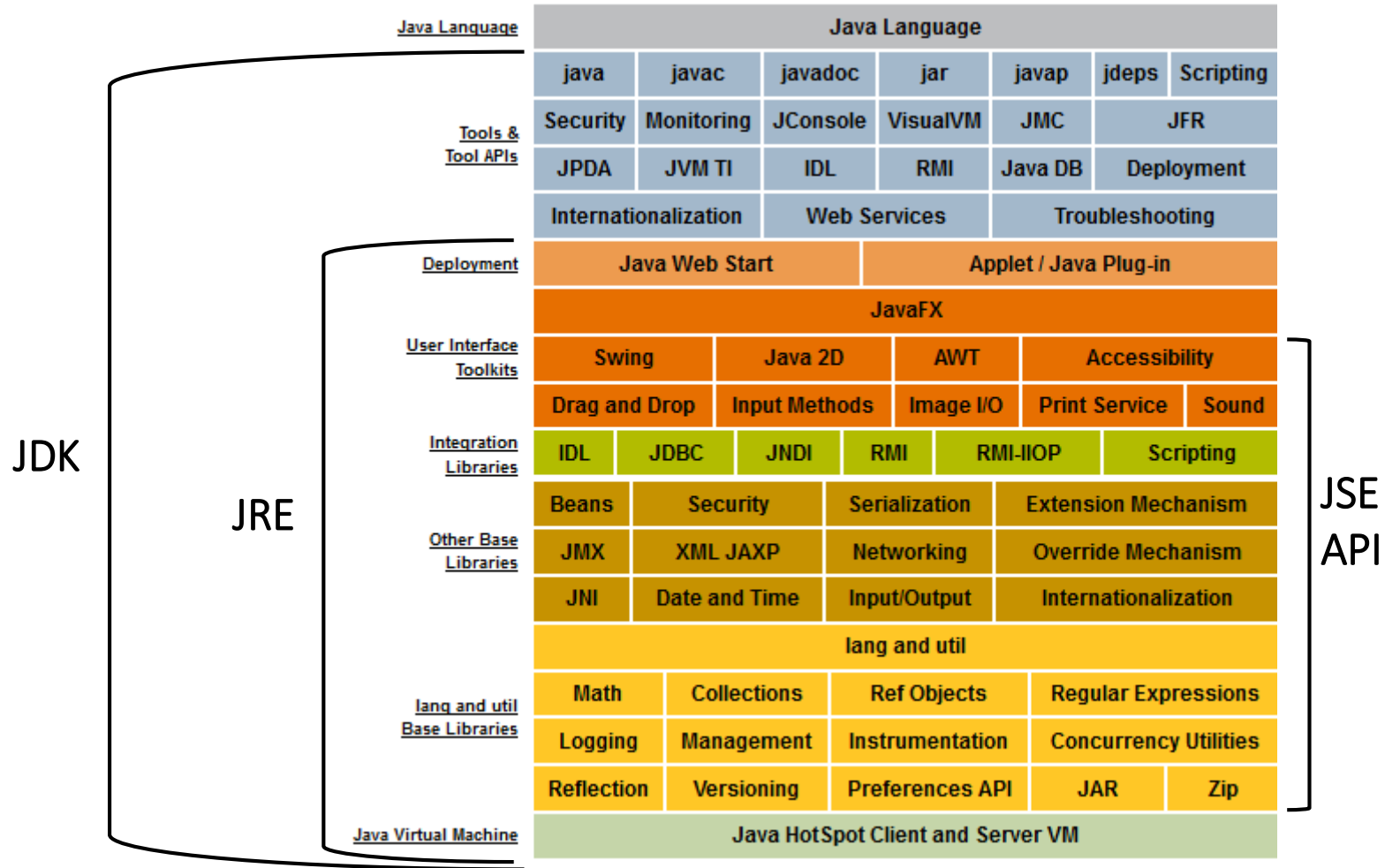
Organisation générale



JRE (Java Runtime Environment) pour l'exécution de code java compilé

JSE : Java Standard Edition

Organisation générale



JDK (Java Developer's Kit) outils de base pour le développement d'applications Java

Distributions du JDK

- Oracle JDK

<https://www.oracle.com/technetwork/java/javase/overview/index.html>

The screenshot shows the Oracle website's 'Java SE at a Glance' page. At the top, there's a navigation bar with 'ORACLE' and various menu items like 'Products', 'Industries', 'Resources', 'Support', 'Events', 'Developer', and 'Partners'. Below this, a breadcrumb trail reads 'Java / Technologies / Java SE'. The main content area features the Java logo and the heading 'Java SE at a Glance'. A sub-heading reads 'Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.' A 'General FAQs' button is visible. The page is divided into three columns: 'What's New' (highlighting Java Platform, Standard Edition 17), 'Know More' (with links for Downloads, Documentation, Community, and Training), and 'Advanced Management Console' (describing the AMC for system administrators).

récupéré le 4/01/2022

- Open JDK

<https://openjdk.java.net/>

The screenshot shows the OpenJDK website. The main heading is 'OpenJDK' in large orange and blue letters. To the left is a vertical navigation menu with categories like 'OpenJDK FAQ', 'Installing', 'Contributing', 'Sponsoring', 'Developers' Guide', 'Vulnerabilities', 'JDK GA/EA Builds', 'Mailing lists', 'Wiki', 'IRC', 'Bylaws', 'Census', 'Legal', 'JEP Process', 'Source code', 'Mercurial', 'GitHub', 'Tools', 'Mercurial', 'Git', 'jreg harness', 'Groups (overview)', 'Adoption', 'Build', 'Client Libraries', 'Compatibility & Specification', 'Review', 'Compiler', 'Conformance', 'Core Libraries', 'Governing Board', 'HotSpot', 'IDE Tooling & Support', 'Internationalization', 'JMX', 'Members', 'Networking', 'Porters', 'Quality', 'Security', 'Serviceability', 'Vulnerability', 'Web', 'Projects (overview)', 'Amber', 'Annotations Pipeline', '2.0', 'Audio Engine', 'Build Infrastructure', 'CfA', 'Caciocavallo', 'Closures', 'Code Tools', 'Cin'. The main content area features three sections, each with a Java mascot icon: 'What is this?' (describing the open-source implementation), 'Download' (providing links to binaries for Linux, macOS, and Windows), and 'Learn' (listing active community projects like Amber, Loom, Panama, and Valhalla). A 'Hack' section encourages users to contribute to the community.

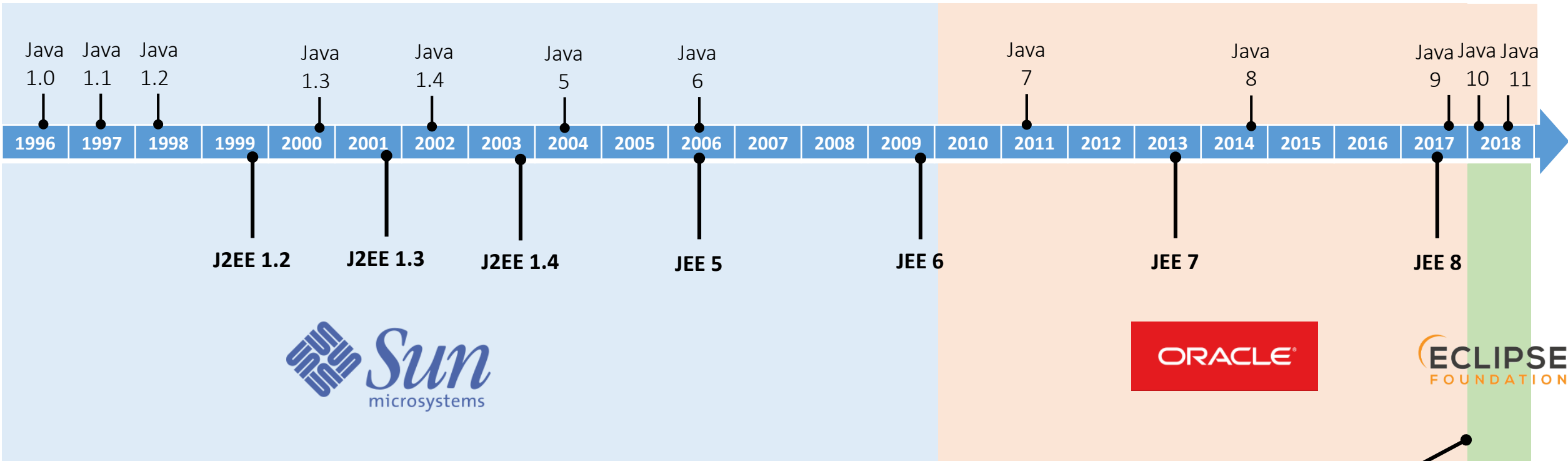
récupéré le 4/01/2022

JEE Java Enterprise Edition

- Java EE (Java Enterprise Edition), anciennement J2EE (Java 2 Enterprise Edition) et maintenant Jakarta EE est un ensemble de spécifications étendant JSE pour le développement d'applications d'entreprise.
<https://jakarta.ee/specifications/platform/>
- La spécification JEE définit
 - Un ensemble d'API pour les applications d'entreprises (applications réparties, services web, mapping Objet/Relationnel pour la persistance des données...),
 - Un plate-forme (Java EE Platform) à base de composants pour héberger et exécuter les applications (serveurs Java EE)
 - Une suite de tests (*Java EE Compatibility Test Suite*) pour vérifier la compatibilité des implémentations
 - Une implémentation de référence (*Java EE Reference Implementation*): [GlassFish](#) ;

JEE Java Entreprise Edition - Historique

- Evolution de la spécification JEE par rapport à évolution du langage Java (JSE)



Fin 2017 Oracle cède JEE à la fondation Eclipse. Java étant une marque déposée, Java EE a été renommé EE4J (Eclipse Enterprise for Java) pour finalement devenir **Jakarta EE**

JEE Spécifications

- <https://jakarta.ee/specifications/platform/>

Différentes versions

Home / Specifications / Jakarta EE Platform

Jakarta EE Platform

- [Jakarta EE Platform 10](#)
Release of the Jakarta EE 10 Platform
- [Jakarta EE Platform 8](#)
Jakarta EE Platform 8 defines a standard platform for ...
- [Jakarta EE Platform 9](#)
Jakarta EE Platform 9 defines a standard platform for ...
- [Jakarta EE Platform 9.1](#)
Jakarta EE Platform 9.1 defines a standard platform for ...

Home / Specifications / Jakarta EE Platform / Jakarta EE Platform 8

Pour une version

Jakarta EE Platform 8

The Jakarta EE Platform defines a standard platform for hosting Jakarta EE applications.

- [Jakarta EE Platform 8 Specification Document](#) (PDF)
- [Jakarta EE Platform 8 Specification Document](#) (HTML)
- [Jakarta EE Platform 8 Javadoc](#)
- [Jakarta EE Platform 8 TCK](#) (sig.sha.pub)
- Maven coordinates
 - jakarta.platform:jakarta.jakartaee-api:jar:8.0.0
- Compatible Implementations used for ratification.
 - Eclipse Glassfish 5.1

Compatible Implementations

- [Jakarta EE 8 Compatible Implementations](#)

Spécifications

Jakarta EE Platform

- Table of Contents
- Copyright
- ▶ Eclipse Foundation Specification License
- ▶ Chapter 1. Introduction
- ▶ Chapter 2. Platform Overview
- ▶ Chapter 3. Security
- ▶ Chapter 4. Transaction Management
- ▶ Chapter 5. Resources, Naming, and Injection
- ▶ Chapter 6. Application Programming Interface
- ▶ Chapter 7. Interoperability
- ▶ Chapter 8. Application Assembly and Deployment
- ▶ Chapter 9. Profiles
- ▶ Chapter 10. Application Clients
- ▶ Chapter 11. Service Provider Interface
- ▶ Chapter 12. Compatibility and Migration
- ▶ Chapter 13. Future Directions
- ▶ Appendix A: Previous Version Deployment Descriptors
- ▶ Appendix B: Java EE 8 and Jakarta EE 8 Comparison
- ▶ Appendix C: Revision History
- ▶ Appendix D: Related Documents

JAKARTA EE

Jakarta EE Platform

Jakarta EE Platform Team, <https://projects.eclipse.org/projects/ee4j/jakartaee-platform>

8 August 2019

APIs (Javadoc)

All Classes

Packages

- javax.annotation
- javax.annotation.security
- javax.annotation.sql
- javax.batch.api
- javax.batch.api.chunk
- javax.batch.api.chunk.listener
- javax.batch.api.listener
- javax.batch.api.partition
- javax.batch.operations

All Classes

- AbortedException
- AbortProcessingException
- AbstractBatchlet
- AbstractCheckpointAlgorithm
- AbstractChunkListener
- AbstractClearableCredential
- AbstractItemProcessListener
- AbstractItemReader
- AbstractItemWriter

Jakarta EE 8 Specification APIs

Package	Description
javax.annotation	This package defines the common annotations.
javax.annotation.security	This package contains the security common annotations.
javax.annotation.sql	
javax.batch.api	Provides APIs used by Batch artifacts.
javax.batch.api.chunk	Provides APIs used by artifacts in a Batch chunk step.

Implémentations (open source ou commerciales)

- Apache TomEE
- Eclipse GlassFish
- FUJITSU Software Enterprise Application Platform
- IBM WebSphere Liberty
- ORACLE WebLogic Server
- payara® COMMUNITY
- WildFly
- Red Hat JBoss Enterprise Application Platform
- Oracle WebLogic Server 14c
- Payara Server Community
- WildFly
- JBoss Enterprise Application Platform

Java EE - APIs

- JEE inclus de nombreuses API couvrant de nombreuses fonctionnalités
 - pages web dynamiques, lecture écriture transactionnelle dans des BD, files d'attentes distribuées...
 - **Web**
 - Servlet: gestion de requêtes HTTP. API de bas niveau, les autres spécifications JEE s'appuient sur elle,
 - JSP (Java Server Pages) : pages web dynamiques,
 - JSF (Java Server Faces): construction d'interfaces utilisateur à partir de composants;
 - EL (Unified Expression Language): langage simple pour lier des composants (JSP ou JSF) avec des objets Java,
 - WebSocket: gestion de connexions WebSocket.
 - **Services Web**
 - Java API for XML Web Services: pour la création de services web SOAP,
 - Java API for RESTful Web Services: support pour créer des services web conformes aux principes architecturaux REST (Representational State Transfer);
 - Java API for JSON Processing: gestion d'informations encodées au format JSON format;;
 - Java API for JSON Binding: conversion d'informations JSON en classes et objets Java et inversement,
 - Java Architecture for XML Binding: transformation de XML en objets Java,

Java EE - APIs

– Spécifications d'entreprise

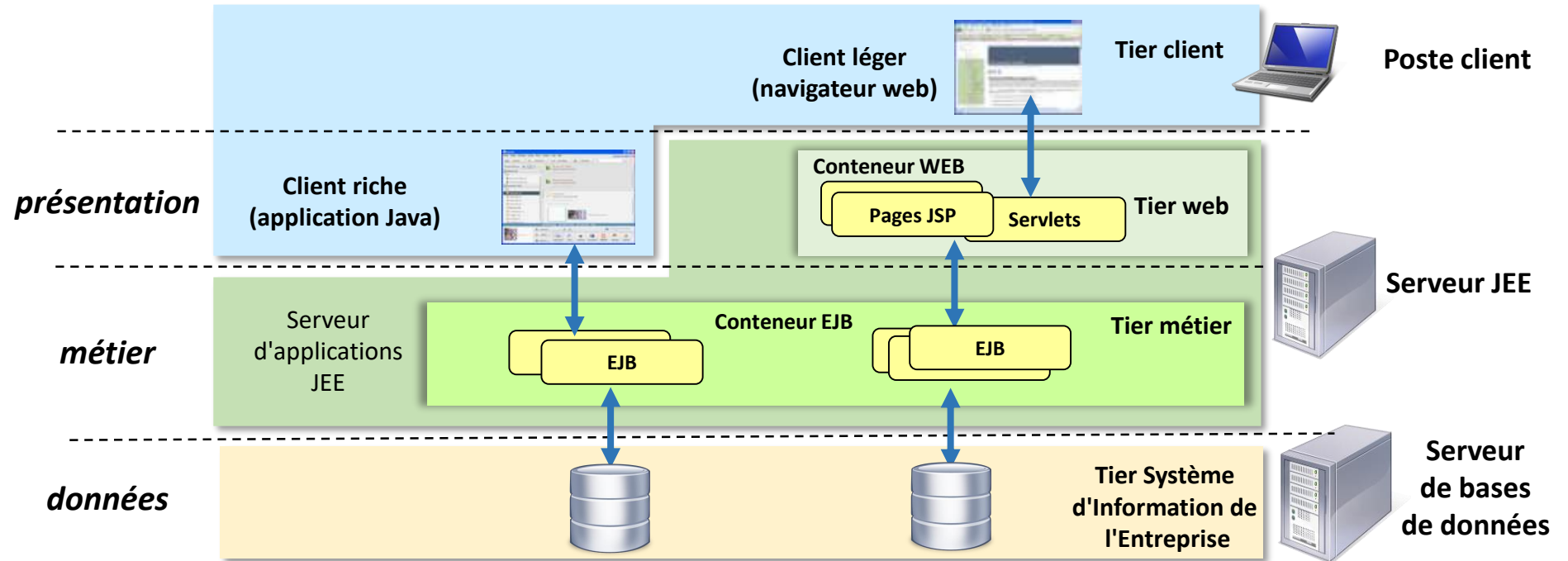
- EJB (Enterprise JavaBean) ensemble d'APIs légères (lightweigh) pour la définition d'objets métiers (Entreprise Beans) pris en charge par un container d'objets (EJB container)
 - transactions (en utilisant JTA), appels de procédures à distance (RMI ou RMI-IIOP), gestion de la concurrence, injection de dépendances, control d'accès.
- JPA (Java Persistence API) : ORM (Object-Relational Mapping) entre tables d'un SGBDR et des classes Java.
- JTA (Java Transaction API) : interaction avec le support transactionnel offert par JEE with the transaction support offert par Java EE.
- JMS (Java Message Service) : création, envoi, réception de messages d'un système de messagerie d'entreprise.

– Autres spécifications

- Validation (Bean Validation API) : fournit un moyen unifié pour définir des contraintes sur les objets métiers (beans, par exemple les classes de modèle JPA model) qui peuvent être appliquées sur plusieurs couches (persistance JPA, vues JSF).
- Batch Applications: exécution de programmes en tâches de fond.

Architecture des applications Java EE

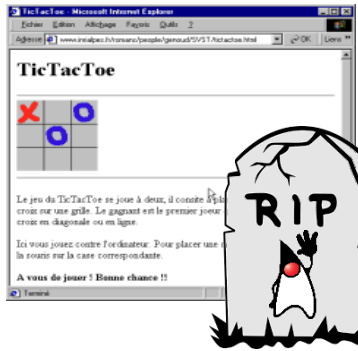
- S'appuie sur un modèle d'architecture multi-tiers (multi-couches)



- Logique de l'application :
 - Composants web (Servlet, JSP, JFS)
 - Composants métiers (EJB) : Client , Compte en Banques
- Services standards (cycle de vie des composants, multithreading, transactions, persistance...) pris en charge par les conteneurs Web et EJB du serveur d'application JEE

Exemples de programmes Java

- Les Applications indépendantes
 - Programmes autonomes (stand-alone) lancé depuis une JVM

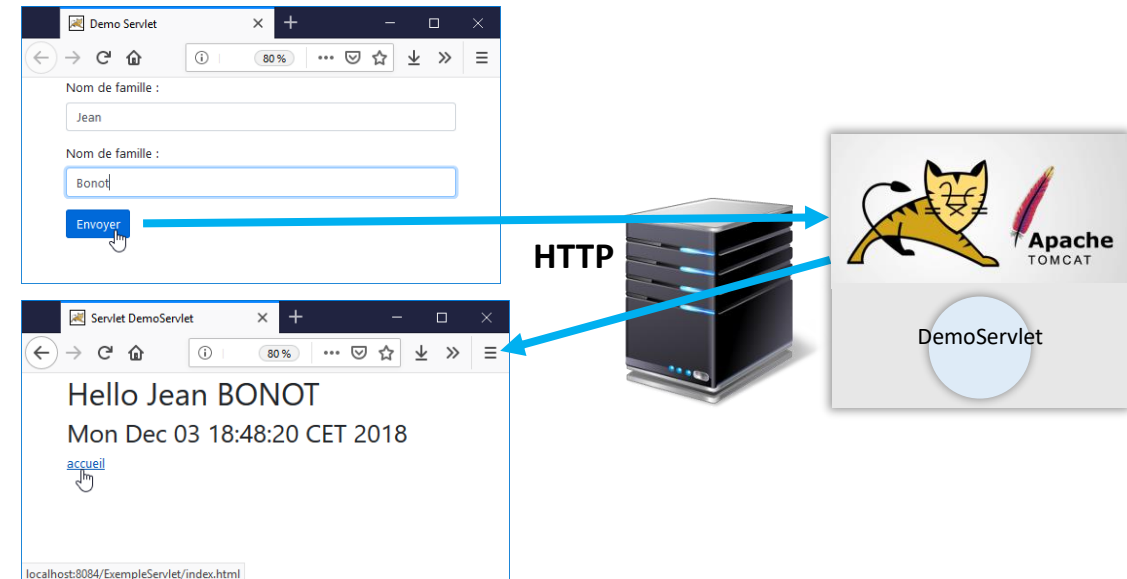
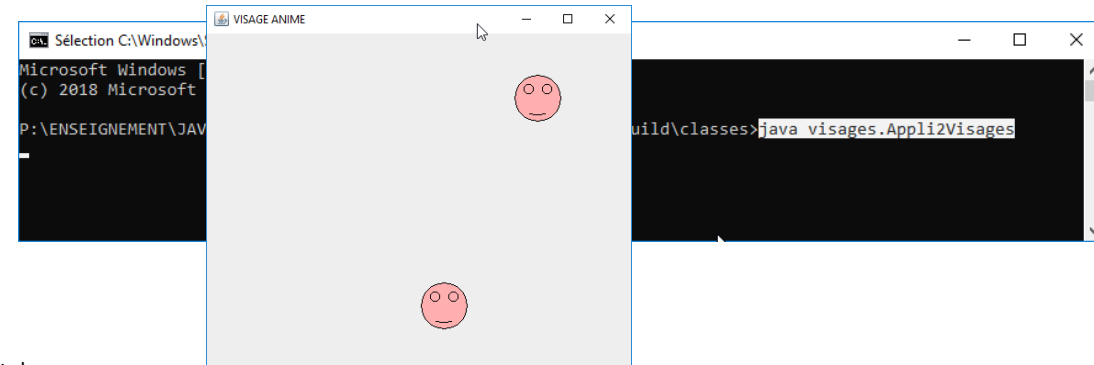


- Les Applets
 - Programmes exécutées dans l'environnement d'un navigateur Web et chargés au travers de pages HTML

- Servlets
 - Programmes exécutés par un serveur d'application (exemple Apache Tomcat conteneur Web JEE)

Seuls diffèrent les contextes d'invocation et d'exécution

- Les droits des applications, applets servlets ne sont pas les mêmes



Exemples de programmes Java

Application indépendante

Application est définie par un ensemble de classes dont **une** jouera le rôle de **classe principale**



Appli2Visages.java
La classe "principale"



VisageRond.java
Défini un type d'objet Visage



Dessin.java
Gère la zone de dessin

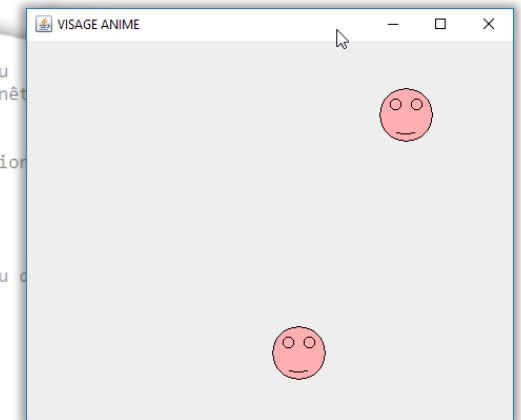
- Application doit posséder une classe principale
 - classe possédant une méthode de signature **public static void main(String[] args)**
 - cette méthode servira de point d'entrée pour l'exécution **java Appli2Visages**

Tableau de chaînes de caractères (équivalent à `argv`, `argv` du C)

Exécute le code défini dans la méthode `main` contenue dans le fichier `Appli2Visages.class`

```
1 package visages;
2
3 import javax.swing.JFrame;
4 import javax.swing.WindowConstants;
5
6 /**
7  * Animation d'un visage dans une fenêtre graphique.
8  * Un visage est dessiné à l'intérieur d'une fenêtre et se déplace. Chaque fois
9  * que l'un des bords est atteint, le visage change de direction.
10  * @author Philippe Genoud
11  */
12 public class Appli2Visages {
13
14     public static void main(String[] args) {
15
16         // la fenêtre graphique
17         JFrame laFenetre = new JFrame("VISAGE ANIME");
18         laFenetre.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
19         laFenetre.setSize(512,512);
20
21         // créé la zone de dessin et la place dans la fenêtre
22         Dessin d = new Dessin();
23         laFenetre.add(d);
24
25         // affiche la fenêtre
26         laFenetre.setVisible(true);
27
28         // creation d'un objet VisageRond
29         VisageRond v1 = new VisageRond();
30         // on rajoute cet objet la zone de dessin
31         d.ajouterObjet(v1);
32         // création d'un deuxième visage
33         VisageRond v2 = new VisageRond(d.getLargeur()/2,d.getHauteur()/2);
34         v2.setDy(-5);
35         // ajout de cet objet à la zone de dessin
36         d.ajouterObjet(v2);
37     }
38 }
```

```
37
38 // la boucle d'animation
39 // c'est une boucle infinie, le programme devra être interrompu
40 // par CTRL-C ou en cliquant dans le cas de fermeture de la fenê
41 while (true) {
42     // les visages effectuent un déplacement élémentaire
43     // en rebondissant sur les bords et en changeant d'expression
44     v1.deplacerAvecRebond();
45     v2.deplacerAvecRebond();
46     // la zone de dessin se réaffiche
47     d.repaint();
48     // un temps de pause pour avoir le temps de voir le nouveau c
49     d.pause(50);
50 }
51 }
52 }
53 }
54 } // AppliVisage1
55 }
```



Exemples de programmes Java

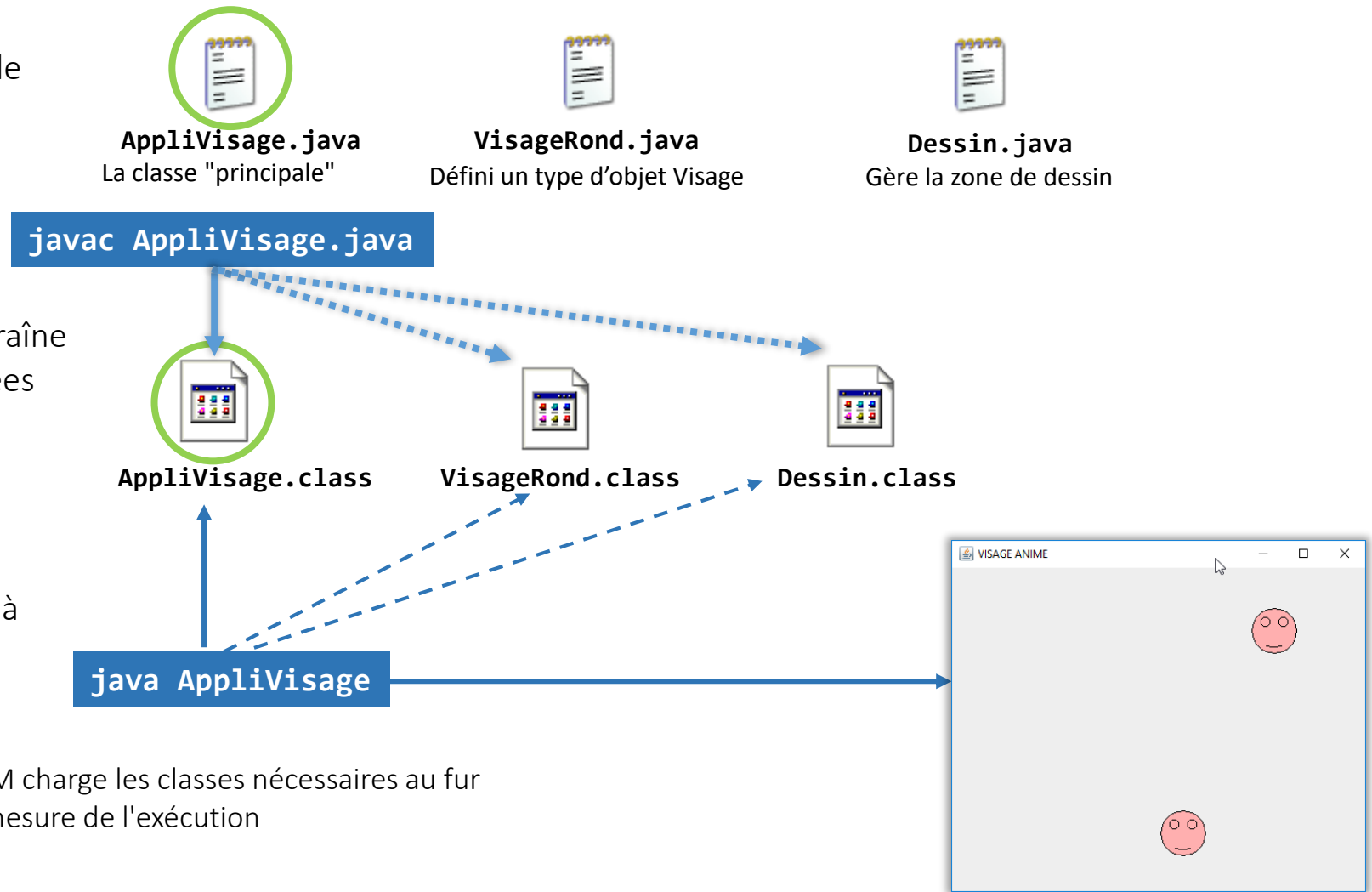
Application indépendante

Application est définie par un ensemble de classes dont **une** jouera le rôle de **classe principale**

La compilation de la classe principale entraîne la compilation de toutes les classes utilisées
`javac ≈ make`

Pour exécuter l'application on indique à l'interpréteur java le nom de la classe principale

La JVM charge les classes nécessaires au fur et à mesure de l'exécution



Exemples programmes Java

Applet

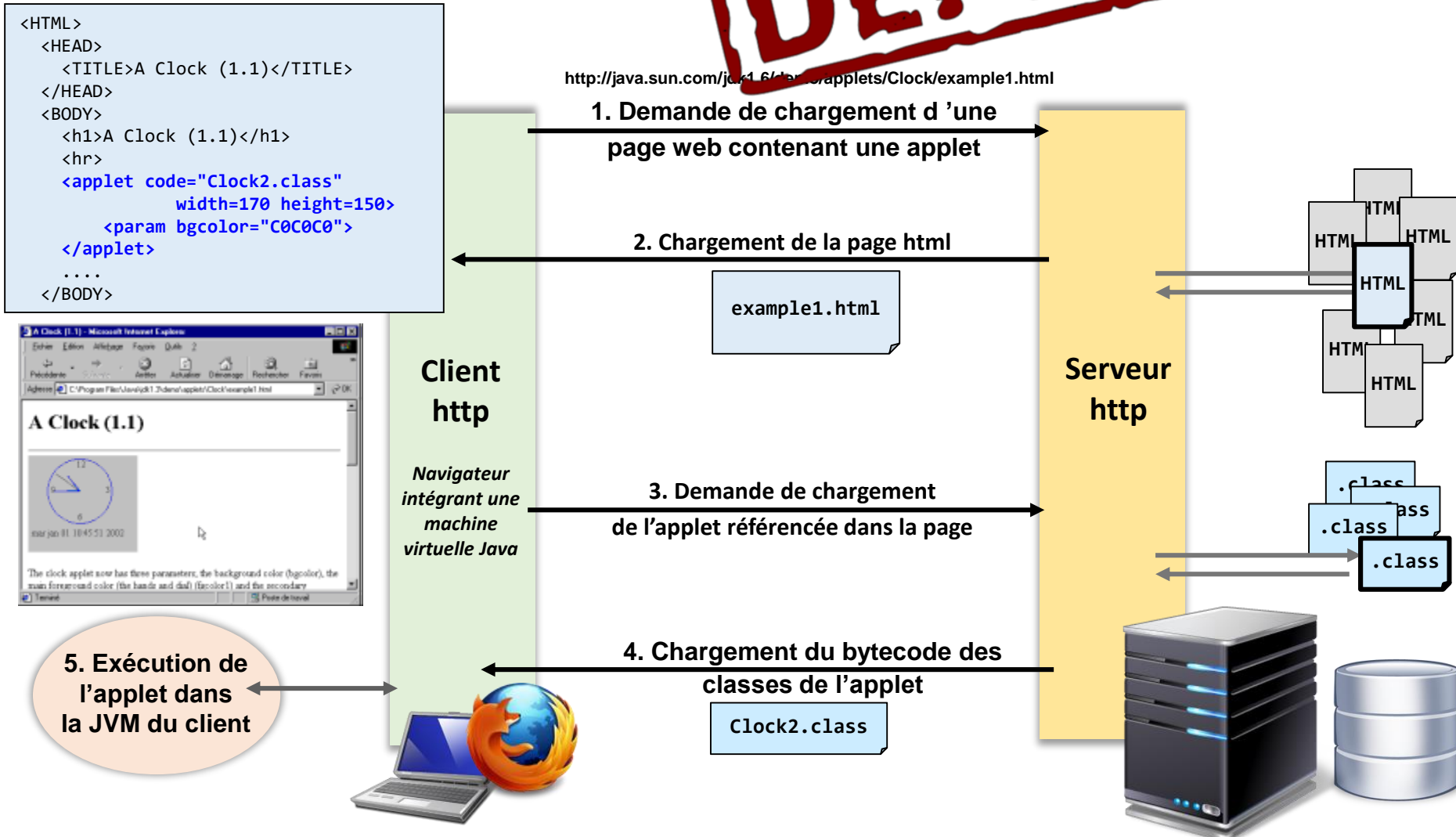
- Classe principale ne possède pas de méthode `main()`
- Hérite de `java.awt.Applet` ou `javax.swing.JApplet`
- Son bytecode réside sur un serveur http
- Elle est véhiculée vers un client http (navigateur Web) via une page html qui contient son url
- Lorsqu'un navigateur compatible Java (avec sa propre machine virtuelle java (JVM) : Java Plugin) reçoit cette page HTML, il télécharge le code de la classe et l'exécute sur le poste client
 - l'applet doit posséder un certain nombre de méthodes pour permettre cette exécution
 - `init()`, `start()`, `stop()`, `paint()`, `destroy()`



Exemple de programmes Java

Applet : Principe de fonctionnement

DEPRECATED

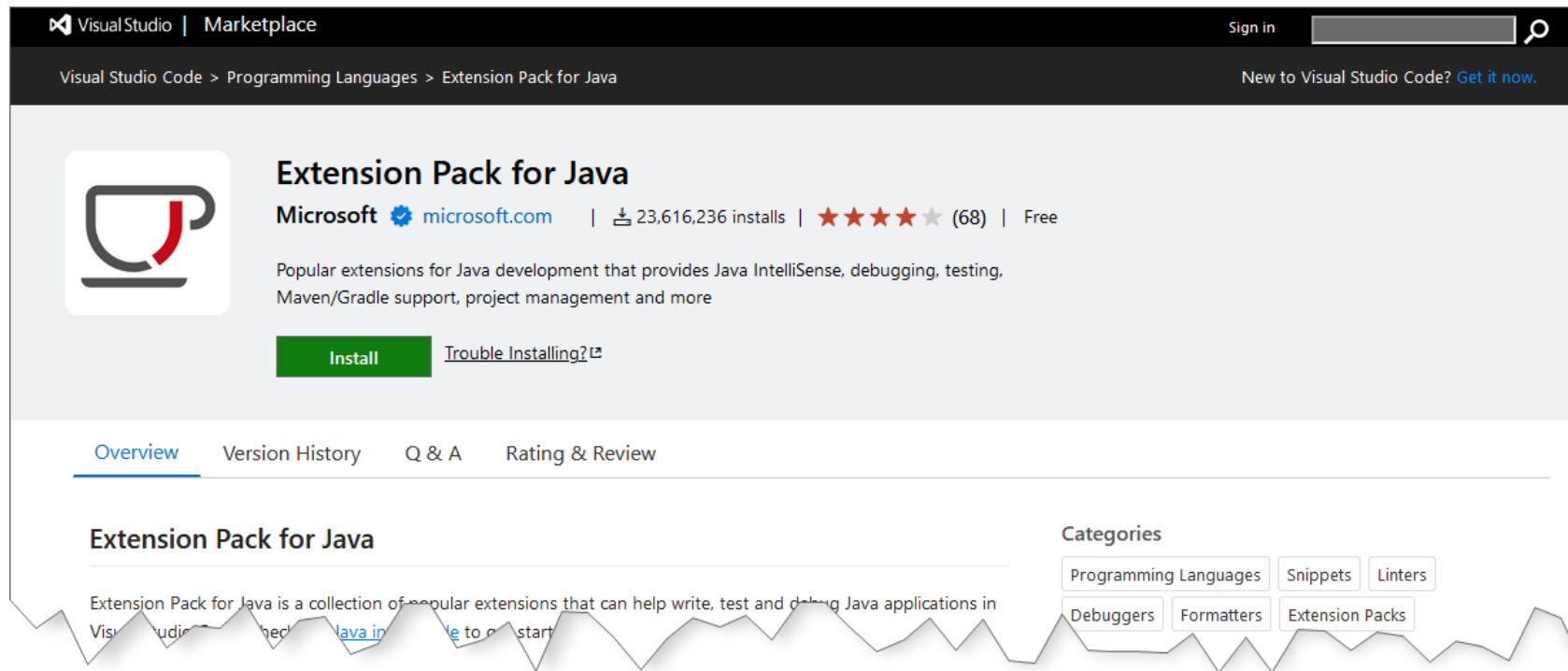


1^{ère} séance de TP

- https://lig-membres.imag.fr/genoud/teaching/PL2AI/tds/P00/sujets/tp01_jdk/tp01_jdk_21.html
- utilisation depuis ligne de commande des outils de base du JDK pour le développement d'applications Java
 - Compilation :
 - **javac**
 - Exécution :
 - **java**
 - Documentation
 - **javadoc**
 - "Archivage"
 - **jar**
- Pourquoi ne pas utiliser tout de suite un IDE Java ?
 - Être capable de travailler avec Java quelque soit l'environnement,
 - Bien comprendre les étapes de compilation puis d'exécution,
 - Comprendre des mécanismes parfois masqués par des environnements intégrés (ex. la notion de **CLASSPATH**).

2^{ème} séance de TP

- Tutoriel d'initiation à l'utilisation de Visual Studio Code pour développer en Java



1: Configuration de VSCode

2: Creation d'un projet Java

3: Compiler et exécuter un projet

4: Utilisation de l'éditeur

5: Interrompre l'exécution d'un programme

6: Générer la documentation Javadoc

7: Utiliser le debugger

7: documentation en ligne

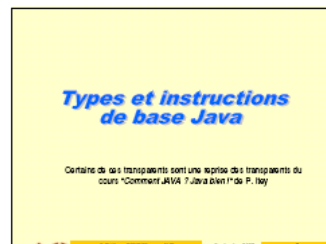
^ Haut de page

3^{ème} séance de TP

https://lig-membres.imag.fr/genoud/teaching/PL2AI/tds/P00/sujets/tp03_typesSimples/typesSimples.html

- Objectif : expérimenter les constructions de base du langage Java (types simples, instructions de contrôle, itérations) qui sont très proches syntaxiquement de celles utilisées par le langage C
- Ecriture de programmes simples
 - Thème 1 : expressions
 - Thème 2 : instructions conditionnelles
 - Thème 3 : itérations
 - Thème 4 : instanciation d'objets - envois de messages

<http://lig-membres.imag.fr/genoud/PL2AI/cours/>



Un programme en langage JAVA consiste en une série d'énoncés (statements) organisés en blocs. Ces énoncés définissent des données et des opérations sur ces données. La fin de chaque énoncé est marquée par un ;. Ce caractère indique à l'interpréteur JAVA que l'énoncé est complet, qu'il doit être exécuté avant de passer à l'énoncé suivant. L'ordre dans lequel l'interpréteur JAVA exécute les énoncés s'appelle le flot d'exécution ou flot de contrôle.

1. Types simples

type	désignation	domaine	constantes	opérations
entiers	byte entiers sur 8 bits	-128 à 127	323 constante décimale	unaire : +, -
	short entiers sur 16 bits	-2147483648 à 2147483647	0xF2 constante hexadécimale	binaire : +, -, *, /
	int entiers sur 32 bits	-9223372036854775808 à 9223372036854775807	0777 constante octale	

- 1: Types simples
- 2: Enoncés élémentaires
 - Déclaration d'une variable
 - Affectation
- 3: Flot de contrôle
 - Instruction composée
 - Instructions conditionnelles
 - Instruction itératives
 - Boucle Tantque
 - Boucle Répéter
 - Boucle Pour