CiDEMA
CONSEIL • DEVELOPPEMENT • OPTIMISATION

EPF
Ecole d'ingénieurs

La Programmation Orienté Objet

MODULE 1 : JAVA ?

Sommaire

Présentation de JAVA

Le **langage Java** est un **langage généraliste** de programmation synthétisant les principaux langages existants lors de sa création en 1995 par *Sun Microsystems*.

Il permet une **programmation orientée-objet** (à l'instar de SmallTalk et C++), **modulaire** (langage ADA) et reprend une syntaxe très proche de celle du langage C.

Outre son orientation objet, le langage Java a l'avantage d'être **modulaire** (on peut écrire des portions de code génériques, c-à-d utilisables par plusieurs applications), **rigoureux** (la plupart des erreurs se produisent à la compilation et non à l'exécution) et **portable** (un même programme compilé peut s'exécuter sur différents environnements).

En contrepartie, les applications Java ont le défaut d'être plus lentes à l'exécution que des applications programmées en C par exemple.

Environnement JAVA

Java est un langage interprété, ce qui signifie qu'un programme compilé n'est pas directement exécutable par le système d'exploitation mais il doit être interprété par un autre programme, qu'on appelle interpréteur. La figure suivante illustre ce fonctionnement.

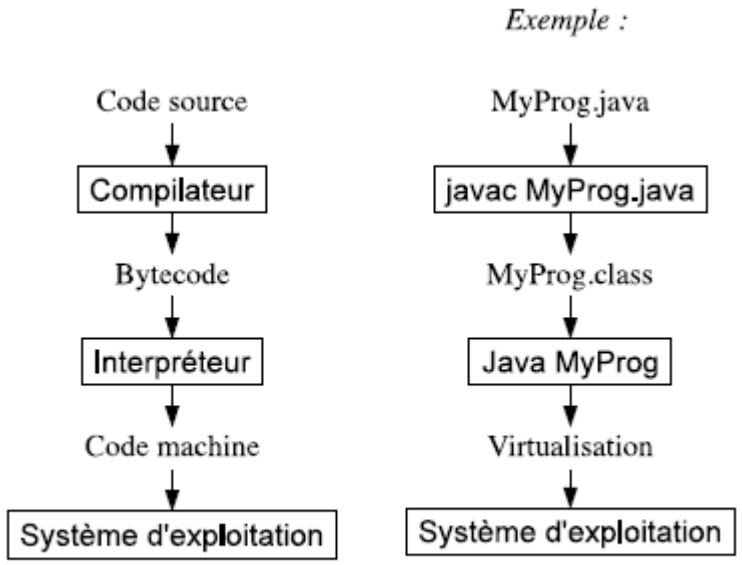
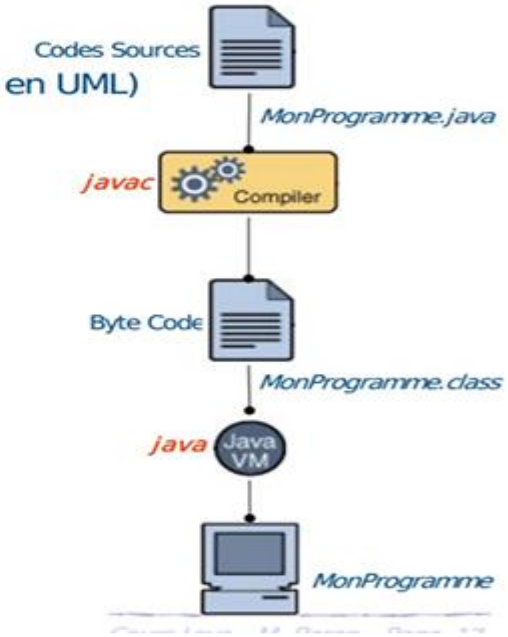


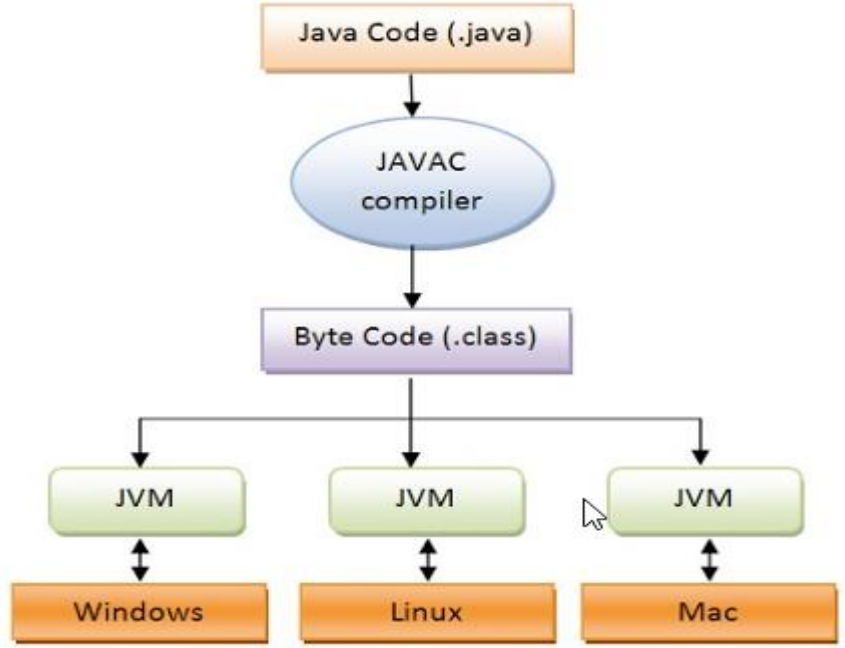
FIGURE 1.1 – Interprétation du langage

Environnement JAVA

Un programmeur Java écrit son code source, sous la forme de classes, dans des fichiers dont l'extension est `.java`.
Ce code source est alors compilé par le compilateur `javac` en un langage appelé *bytecode* et enregistre le résultat dans un fichier dont l'extension est `.class`.
Le bytecode ainsi obtenu n'est pas directement utilisable. Il doit être interprété par la *machine virtuelle* de Java qui transforme alors le code compilé en code machine compréhensible par le système d'exploitation.
C'est la raison pour laquelle Java est un langage portable : le bytecode reste le même quelquesoit l'environnement d'exécution

- Création du code source
  - A partir des spécifications (par exemple en UML)
  - Outil : éditeur de texte, IDE
- Compilation en Byte-Code
  - A partir du code source
  - Outil : compilateur Java
- Diffusion sur l'architecture cible
  - Transfert du Byte-Code seul
  - Outils : réseau, disque, etc
- Exécution sur la machine cible
  - Exécution du Byte-Code
  - Outil : Machine Virtuelle Java





Compilation

La compilation s'effectue par la commande javac suivie d'un ou plusieurs nom de fichiers contenant le code source de classes Java. Par exemple, javac MyProg.java compile la classe My-Prog dont le code source est situé dans le fichier MyProg.java. La compilation nécessite souvent la

précision de certains paramètres pour s'effectuer correctement, notamment lorsque le code source fait référence à certaines classes situées dans d'autres répertoires que celui du code compilé. Il faut alors ajouter l'option -classpath suivie des répertoires (séparés par un ; sous Windows et : sous Unix) des classes référencées.

Par exemple :

javac -classpath /prog/exos1:/cours MyProg.java

compilera le fichier MyProg.java si celui-ci fait référence à d'autres classes situées dans les répertoires /prog/exos1 et /cours. Le résultat de cette compilation est un fichier nommé My-Prog.class contenant le bytecode correspondant au source compilé. Ce fichier est créé par défaut dans le répertoire où la compilation s'est produite. Il est cependant fortement souhaitable de ne pas mélanger les fichiers contenant le code source et ceux contenant le bytecode. Un répertoire de destination où sera créé le fichier MyProg.class peut être précisé par l'option -d, par exemple :

javac -d /prog/exos1 -classpath /cours MyProg.java

Interprétation

Le bytecode obtenu par compilation ne peut être exécuté qu'à l'aide de l'interpréteur. L'exécution s'effectue par la commande `java` suivie du nom de la classe à exécuter (sans l'extension `.class`).

Comme lors de la compilation, il se peut que des classes d'autres répertoires soient nécessaires.

Il faut alors utiliser l'option `-classpath` comme dans l'exemple qui suit :

```
java -classpath /prog/exos1:/cours MyProg
```

Documentation

La documentation JAVA est riche

JAVADOC :

https://docs.oracle.com/en/java/javase/22/