Eléments du Langage Java

1. Notion de Machine Virtuelle JVM

Java est un langage de programmation OO adapté à la distribution d'applications sur Internet et s'intégrant au Web.

Java est conçu pour qu'il soit indépendant de la plate forme d'exécution des programmes.

Java est développé par SUN Microsystems, il intègre :

- un langage de programmation,
- une machine virtuelle (JVM) ou un interpréteur (JDK),
- un ensemble d'outils (jdb, javadoc),
- un ensemble de classes standards dans des API,

Les outils dans le JDK :

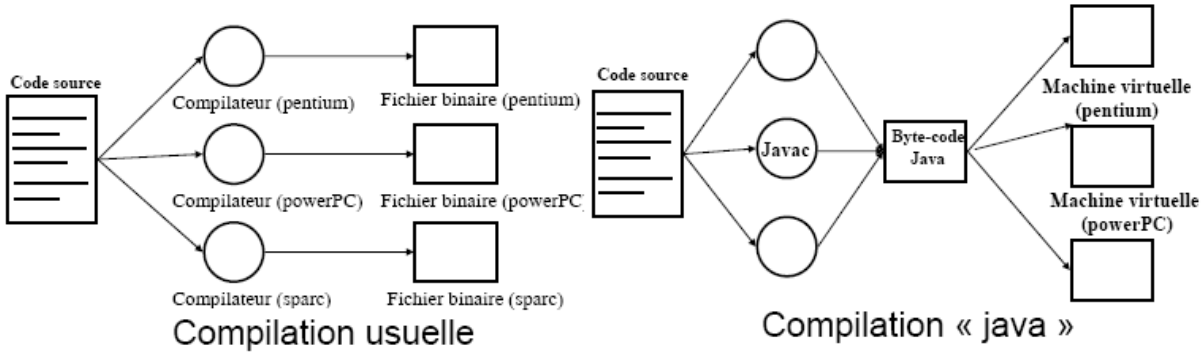
- javac : le compilateur java
- java : la machine virtuelle
- javadoc : le générateur de documents
- jdb : le débogueur java
- javap : le désassembleur java, permet de convertir un fichier .class à un fichier .java

Java n'est pas gourmand en mémoire : la JVM n'a besoin que de 215KO pour s'exécuter.

En java le code source n'est pas traduit directement en langage machine (natif). Il est d'abord traduit dans un langage appelé « bytecode » qui est un langage d'une machine virtuelle dont le langage a été défini par Sun, ce langage est indépendant de la plate forme d'exécution du programme.

Le bytecode s'exécute par une JVM qui est un interpréteur qui lit chaque instruction du bytecode du programme .class, la traduit dans le langage du processeur de l'ordinateur et lance son exécution.

Les systèmes qui veulent exécuter des programmes java doivent fournir un tel interpréteur. Actuellement tous les systèmes ont une JVM (tous les UNIX, Windows, MacOS…).



2. Caractéristiques du langage Java

Java : A simple, object-oriented, network-savy, interpreted, robust, secure, architecture neutral, portable, highperformance, multithreaded, dynamic language.

– **Java™ c'est** : un langage, une machine virtuelle, un ensemble de classes standards (réparties dans plusieurs API), des outils (JDB, javadoc). . .

– **Simple** : reprise de la syntaxe du C/C++, sans les pointeurs, ni l'héritage multiple de classes, mais avec un ramasse-miettes,

– **Orienté-objet** : tout est classe dérivant de java.lang.Object, à l'exception de quelques types primitifs,

– **Interprété** : au niveau d'une JVM, après production de byte code,

– **Robuste** : mécanisme d'exceptions, langage fortement typé, ramasse-miettes, pas d'accès direct à la mémoire

par pointeur,

– **Sécurisé** : pas d'accès direct aux ressources physiques, compilateur "exigeant", JVM Verifier (vérificateur de byte code), Class Loader et Security Manager (système de fichiers, accès réseau),

– **Indépendant** du matériel et portable : la JVM s'intercale entre le hardware et les applications/applets,

– **Performant** : un peu plus à chaque nouvelle version de JVM,

– **Distribué** : API réseau standard conséquente, mécanismes de RMI, JavaIDL (pour intégration et adéquation à CORBA), JNDI (Java™ Naming and Directory Interface, pour les services d'annuaires),

– **Multithreads** : nativement et de façon simple, géré directement par la JVM,

– **Dynamique** : chargement des classes en cours d'exécution, à la demande, pas de phase d'édition de liens,

– **Simplifie la réutilisation**

3. Conventions d'écritures

- Pourquoi :
  - 80% du coût d'un logiciel est dans sa maintenance,
  - difficile d'avoir la maintenance par la même personne qui a écrit le programme,
  - permettre la lisibilité du programme,
  - réutilisation
- Concerne :
  - nom des fichiers et leurs organisations,
  - indentation,
  - commentaires,
  - déclarations,
  - instructions,
  - espacements,
  - noms.
- Quelques conventions [Stéphane Frénot - stephane.frenot@insa-lyon.fr]
  - un fichier .java par classe
  - entête classique : description...
  - commentaires : JavaDoc
    - Début de bloc : /** .... **/
    - Balises : @author, @version, @see, @param, @return,
  - Convention de nommage
    - paquetages minuscule
    - classes MajusculePourLaPremiereLettreDeChaqueMot
    - méthodes minusculePourLaPremiereCommeLaClasseAprès
    - constantes MAJUSCULE_AVEC_SOULIGNE
    - accesseur d'un attribut y de type X X getY()
    - accesseur de modification void setY(X valeur)

4. Syntaxe du langage

4.1 Syntaxe

- Structure d'une classe
- Les types primitifs et les valeurs littérales
- Les opérateurs
- constantes
- variables

- Structures conditionnelles . if/if...else/if...else if . ? : . switch	Structures iterative . for, . while, . do...while	- Instructions de saut . break . continue . return . throw

4.2 Le mot clé static

- ✓ Permet de déclarer un attribut ou une méthode de classe (qui concerne tous les objets de la classe et non pas un objet spécifique).
- ✓ Un attribut ou une méthode statique est invoqué par le nom de la classe.
- ✓ Syntaxiquement il peut être aussi invoqué par une instance de la classe mais ceci est déconseillé vu qu'il induit une ambigüité sur le fait qu'il est statique ou non.
- ✓ Dans une méthode statique on ne peut utiliser que les attributs statiques.
- ✓ Dans une méthode non statique on peut utiliser les attributs statiques et non statiques.
- ✓ Exemple

Pour la classe Stylo chaque stylo possède sa propre couleur, cependant si on veut compter le nombre de stylos instanciés, ce nombre concerne tous les stylos en même temps donc il est statique.

```
public class Stylo{
        private String marque ;
        private static int nbStylo=0;
        public Stylo(String couleur){
                this.couleur=couleur ;
                nbStylo++;
        }
        Public static int getNbStylo(){
                Return nbStylo ;
        }
}
System.out.println("Nb Stylo : "+Stylo.getNbStylo());        //l'appel
Stylo s;
s=new Stylo("Bleu");
System.out.println("Nb Stylo : "+s.getNbStylo()); //correct mais déconseillé
```

4.3 Le mot clé final

Final peut avoir plusieurs significations, ceci dépond de la déclaration qui l'utilise :

- Devant la déclaration d'une variable de type simple,
- Devant la déclaration d'un objet,
- Devant une méthode,
- Devant une classe

A- Devant la déclaration d'une variable de type simple

La variable déclarée devient une constante.

```
final int MAX = 10 ;
MAX = 11 ;
```

B- Devant la déclaration d'un objet

La référence de l'objet devient une constante, cependant les valeurs des attributs de l'objet peuvent changer

```
final Stylo s = new Stylo() ;
s = new Stylo();
s.ecrire(« Texte»);  // correct et le niveauEncre change
```

4.4 Les tableaux

A- Déclaration	**B- Réservation de mémoire**
type[] nomTableau ; Ou type nomTableau[] ;	nomTableau = new type[taille] ;
C- Accés	**D- Exemples**
type val = nomTableau[indice] ; nomTableau[indice]=val ;	int[] tVal ; tVal = new int[10] ; tval[0]=0 ; tval[1]=1 ; Stylo[] tStylo ; tSylo = new Stylo[10] ; tSylo[0] = new Stylo() ;