Chapitre 1 : Introduction à la programmation java

M. André Bernard Simel YOUM

UCAO- ISAET (THIES)
Licence 3, Informatique de Gestion
Semestre 5

2024-2025

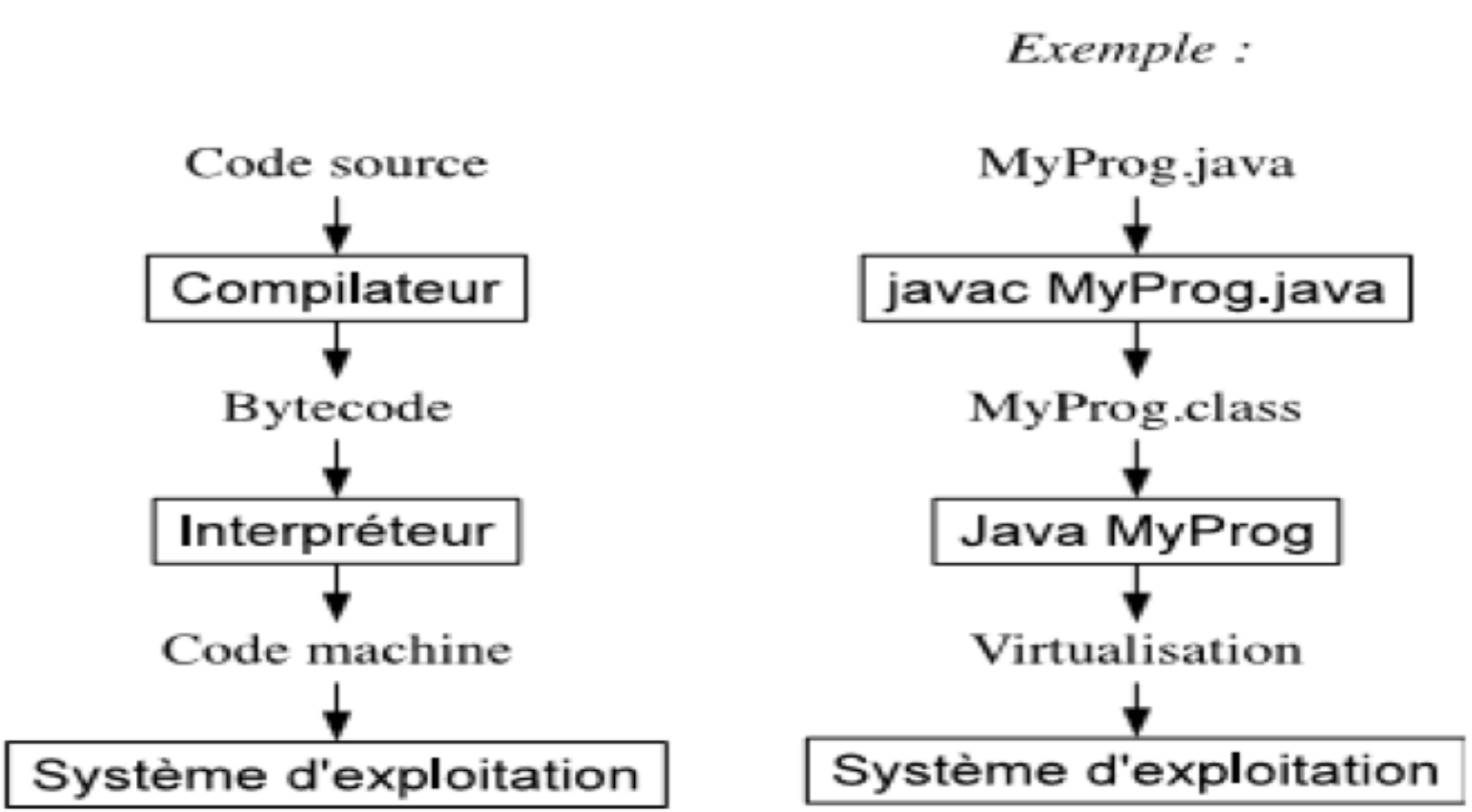
Plan

Introduction

- Le langage Java est un langage généraliste de programmation synthétisant les principaux langages existants lors de sa création en 1995 par Sun Microsystems.
- Il a trois principaux avantages :
  - **Modulaire :** on peut écrire des portions de code génériques, c'est à dire utilisables par plusieurs applications ;
  - **Rigoureux :** la plupart des erreurs se produisent à la compilation et non à l'exécution ;
  - **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.



Compilation et Interprétation

1. **Compilation**
   - La compilation s'effectue par la commande **javac**, suivie d'un ou plusieurs noms de fichiers contenant le code source de classes Java.
   - Par exemple, **javac MyProg.java** compilera le fichier **MyProg.java**
   - Le résultat de cette compilation est un fichier nommé **MyProg.class** contenant le bytecode correspondant au fichier source compilé.
2. **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).
   - Par exemple, **java MyProg**

Kit de développement java

Trois choix sont possibles :.

1. **Choix 1 : Le JDK de Sun (http ://java.sun.com)**
   - référence pour développer en Java.
   - comporte tous les outils indispensables à la réalisation et à l'exécution d'un programme Java.
   - totalement gratuit
2. **Choix 2 : Freeware ou shareware**
   - offrent une interface graphique au JDK.
   - des outils supplémentaires (générateurs d'interface...).
   - exemple : Eclipse, NetBeans
3. **Choix 3 : Environnement de développement** professionnel
   - très puissants, très chers.
   - exemples : Visual J++ de Microsoft, JBuilder 2

JDK / JRE

1. **JDK (Java Development Kit)**
   - pour développer les applications Java.
   - inclut la JRE.
2. **JRE (Java Runtime Environment)**
   - pour exécuter les applications Java.

Manipulation des variables

- Les identificateurs de variables ou de méthodes acceptent les caractères a..z, A..Z, $, _ ainsi que les caractères 0..9 s'ils ne sont pas le premier caractère de l'identificateur.
- La syntaxe de déclaration des variables : **type variable ;**
- Le caractère de fin d'une déclaration et d'une instruction est **" ;"**
  Exemple : a=c+b ;
- Les commentaires d'une ou plusieurs lignes se situent entre les symboles "/*" et "*/"
  /*exemple de commentaire sur une ligne*/
  /*exemple de commentaire
  sur plusieurs lignes*/
- Les commentaires d'une seule ligne peuvent se faire en commençant la ligne par "//"
  //exemple de commentaire sur une ligne

Types primitifs

Le tableau suivant liste l'ensemble des types primitifs de données de Java.

Type	Valeurs	Tailles en octets
boolean	True ou false	Non spécifiée
byte	Entier signé	1
char	Caractère	2
short	Entier signé	2
int	Entier signé	4
long	Entier signé	8
float	Réel signé	4
double	Réel signé	8

Tableaux et Matrices

- Une variable est déclarée comme un tableau dès lors que des crochets sont présents soit après son type, soit après son identificateur. Les deux syntaxes suivantes sont acceptées pour déclarer un tableau :
  1. **type** [] nomTableau ;
     Exemple : int [] monTableau ;
  2. **type** nomTableau [] ;
     Exemple : float Tab [] ;
- Un tableau a toujours une taille fixe qui doit être précisée avant l'affectation de valeurs à ses indices, de la manière suivante :
  - **type** [] nomTableau=**new type** [taille] ;
    Exemple : int [] monTableau=**new int**[20] ;
- La taille de ce tableau est disponible dans une variable **length** appartenant au tableau et accessible par **nomTableau.length**.
- On peut également créer des matrices ou des tableaux à plusieurs dimensions en multipliant les crochets (**ex : int[][] nomMatrice ;**)

Chaines de caractères

- Les chaînes de caractères ne sont pas considérées en Java comme un type primitif ou comme un tableau.
- On utilise une classe particulière, nommée **String**, fournie dans le package **java.lang**.
  Exemples :
  String s1="Bonjour" ;
  String s2="tout le monde" ;
  String s3=s1+s2 ;

Opérateurs (1/2)

Une liste des opérateurs disponibles en Java est présentée par ordre de priorité décroissante dans le tableau suivant :

Pr.	Opérateur	Syntaxe	Résultat	Signification
1	++	++<ari>	<ari>	pré incrémentation
		<ari>++	<ari>	post incrémentation
	–	–<ari>	<ari>	pré décrémentation
		<ari>–	<ari>	post décrémentation
	+	+<ari>	<ari>	signe positif
	-	-<ari>	<ari>	signe négatif
	!	!<boo>	<boo>	complément logique
	(type)	(type)<val>	<val>	changement de type
2	*	<ari>*<ari>	<ari>	multiplication
	/	<ari>/<ari>	<ari>	division
	%	<ari>%<ari>	<ari>	reste de la division
3	+	<ari>+<ari>	<ari>	addition
	-	<ari>-<ari>	<ari>	soustraction
	+	<str>+<str>	<str>	concaténation
4	<<	<ent> << <ent>	<ent>	décalage de bits à gauche
	>>	<ent> >> <ent>	<ent>	décalage de bits à droite

Opérateurs (2/2)

5	<	<ari> < <ari>	<boo>	inférieur à
	<=	<ari> <= <ari>	<boo>	inférieur ou égal à
	>	<ari> > <ari>	<boo>	supérieur à
	>=	<ari> >= <ari>	<boo>	supérieur ou égal à
	instanceof	<val>instanceof<cla>	<boo>	test de type
6	==	<val>==<val>	<boo>	égal à
	!=	<val>!=<val>	<boo>	différent de
7	&	<ent>&<ent>	<ent>	ET bit à bit
		<boo>&<boo>	<boo>	ET booléen
8	^	<ent>^<ent>	<ent>	OU exclusif bit à bit
		<boo>^<boo>	<boo>	OU exclusif booléen
9	|	<ent>|<ent>	<ent>	OU bit à bit
		<boo>|<boo>	<boo>	OU booléen
10	&&	<boo>&&<boo>	<boo>	ET logique
11	||	<boo>||<boo>	<boo>	OU logique
12	?:	<boo>?<ins>:<ins>	<ins>	si...alors...sinon
13	=	<var>=<val>	<val>	assignation

Légende

<ari> valeur arithmétique
<boo> valeur booléenne
<cla> classe
<ent> valeur entière
<ins> instruction
<str> chaîne de caractères (String)
<val> valeur quelconque
<var> variable

Structures de contrôle (1/2)

Les structures de contrôle permettent d'exécuter un bloc d'instructions, soit plusieurs fois **(instructions itératives)** soit selon la valeur d'une expression **(instructions conditionnelles ou de choix multiple)**. Dans tous ces cas, un bloc d'instruction est

- soit une instruction unique ;
- soit une suite d'instructions commençant par une accolade ouvrante "{" et se terminant par une accolade fermante "}".

Structures de contrôle (2/2)

1. Test
   if (<condition>) <bloc1> [**else** <bloc2>]
   ou
   <condition> **?**<instruction1> **:**<instruction2>
   ou
   switch(<expression>){
   case v1 : In1 ;**break** ;
   case v2 : In2 ;**break** ;
   case v3 : In3 ;}
2. Boucle while
   while (<condition>) <bloc>
3. Boucle do..while
   do a++
   while (a != b) ;