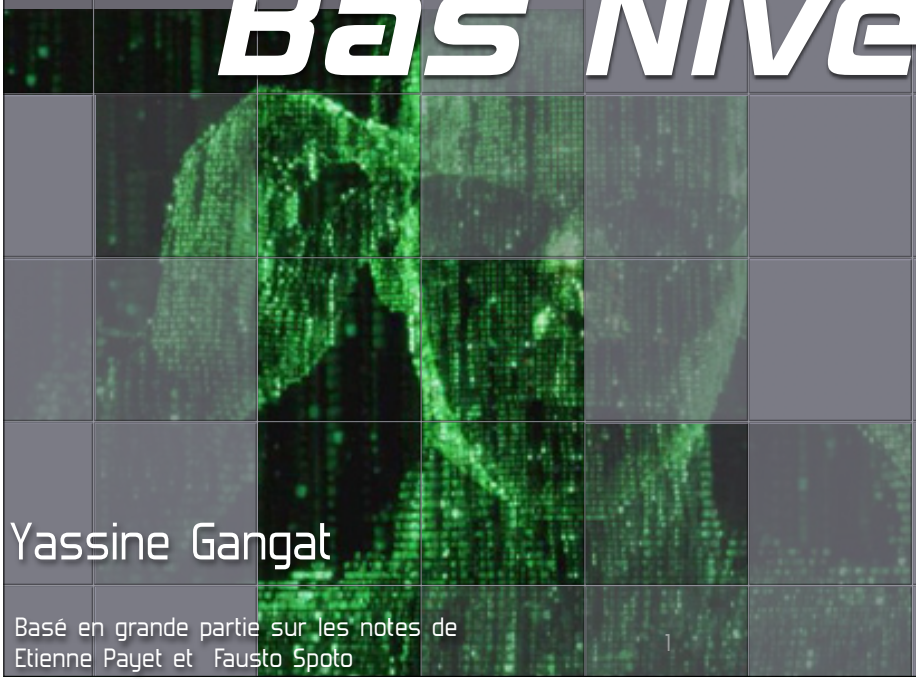
Programmation Bas Niveau



Yassine Gangat

Basé en grande partie sur les notes de Etienne Payet et Fausto Spoto

Université de La Réunion
FST - L2 Info

Introduction

Module

- Machines virtuelles (Java), bytecode, etc.
- 30h : 10h CM, 8h TD, 12h TP
- Evaluation: TP noté + exam final

Kezaco ?

- 6 générations de langages de programmation
  - **Génération 1** : Langage machine avec Instructions binaires en programmation directe
  - **Génération 2** : Instructions sous forme symbolique (mnémoniques) plus compréhensible pour l'homme, en programmation indirecte au travers un programme (assembleur)
  - **Génération 3** : Langages indépendants du processeur. Proches des langues parlées (anglais). Langages procéduraux, descriptions des opérations à effectuer pour résoudre un problème

Kezaco ?

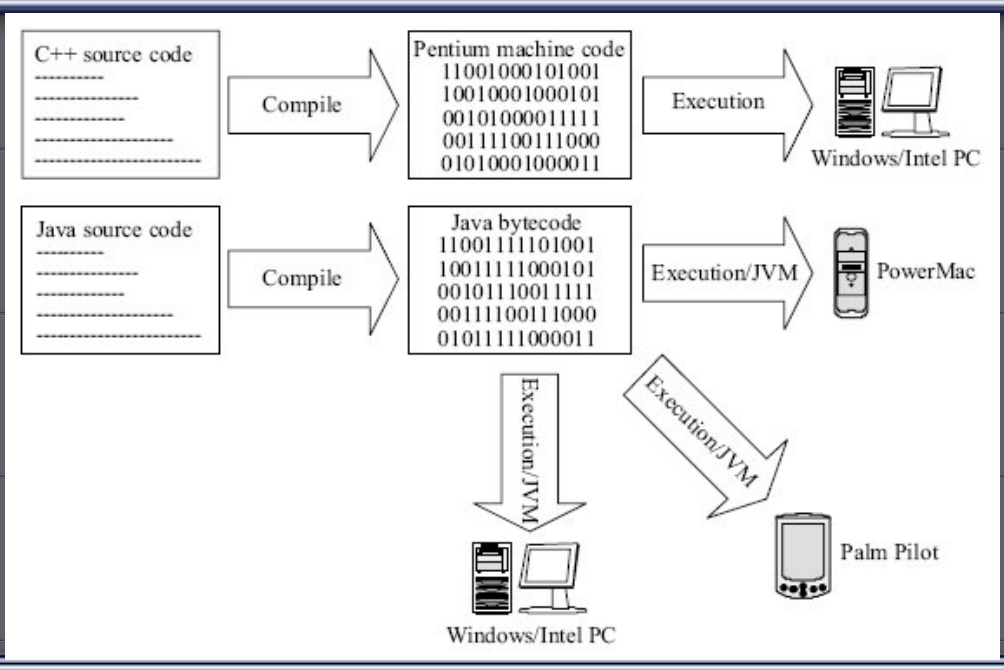
- 6 générations de langages de programmation
  - **Génération 4** : Langages descriptifs, très fortement lié à un domaine (base de données, tables de calcul)
  - **Génération 5** : Langages descriptifs pour la programmation de systèmes experts
  - **Génération 6** : Orienté objet. Toutes les informations nécessaires à la résolution d'un problème sont réunies dans un objet

Type de langages

- Langages des générations 1 et 2: **langages de bas niveau** (orienté machine) - Proche du matériel
- Langages des générations 3 à 6 : **langages de haut niveau** (orienté problème) - Eloigné du matériel

Machines Virtuelles

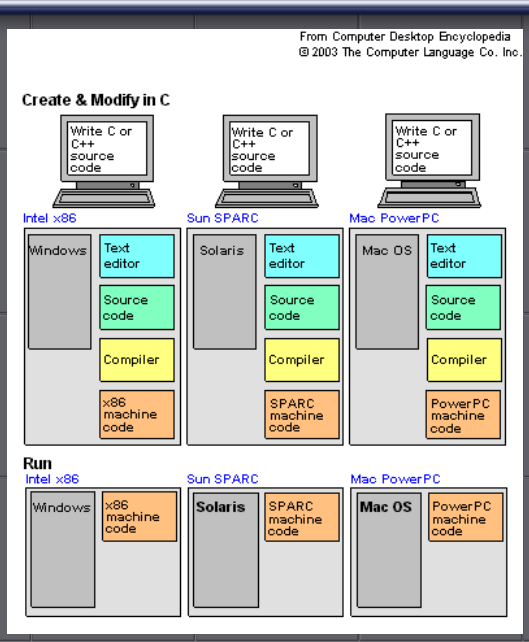
Exemple

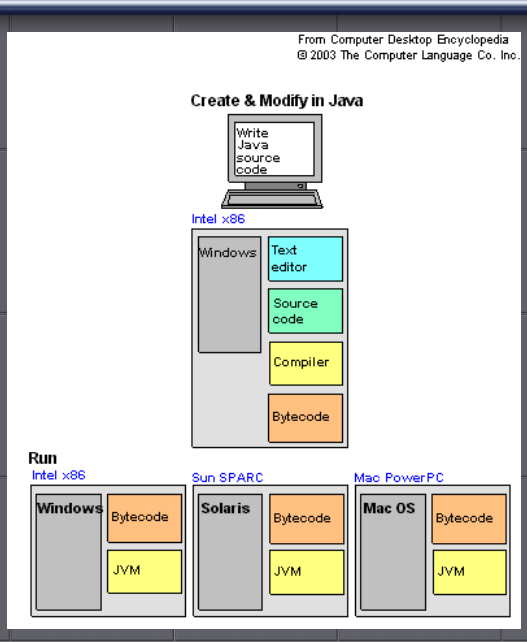


Définition

- **Machine virtuelle de haut niveau:**
  - programme qui exécute un autre programme avec le fonctionnement et la structure d'une machine physique traditionnelle
  - La VM s'exécute elle sur une machine physique et est spécifique à chaque plateforme.

Exemple





Avantages

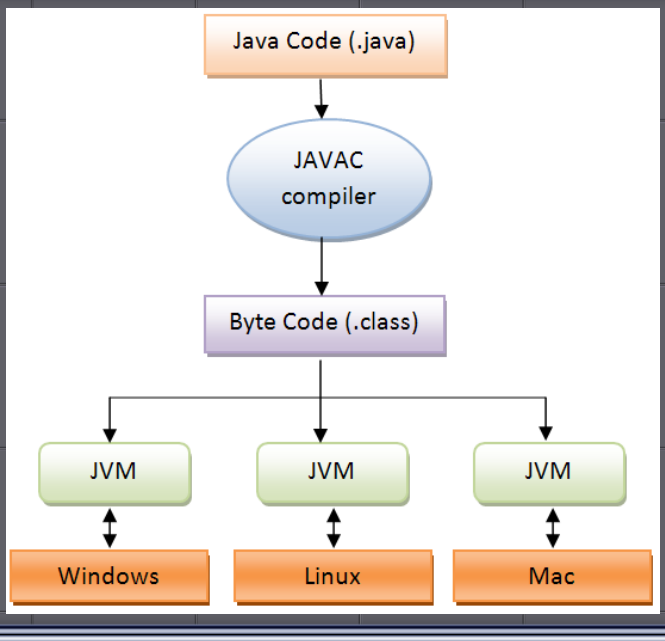
- Portabilité
- Richesse bibliothèque
- Moins de limitations
- Facilité de MAJ
- Sécurité

Inconvénients

- Vitesse d'éxécution
- Impossibilité de contrôler le materiel directement

Et nous ?

- Bytecode !!!



Bytecode

Définition

- Opcode: (operation code)
  - code d'une opération (add, sub, etc.) du langage machine
- Bytecode:
  - opcode de la taille d'un byte (octet)
  - opcode d'une VM
  - par extension, langage d'une machine virtuelle
- Ex: Java Virtual Machine

Java sans Java ?

- Bytecode sans Java : (Jython, etc.)
- Assembleur de bytecode : (Jasmin, Oolong)

Assembleur

- **Langage d'assemblage** (Langage assembleur ou encore assembleur): forme plus lisible par un humain du langage machine, associant un mnémonique à chaque opcode (ex: add, sub, etc.)
- **Assembleur**: programme qui convertit le mnémonique en langage machine

Jasmin

- http://jasmin.sourceforge.net/

Structure JVM

Lire un .class

- javap -c nomDeFichier
  - -c : pour avoir le détail des méthodes
  - nomDeFichier : un fichier .class (ne pas ajouter l'extension)
  - A tester ;)

Types de bases

Type	Descrip de type - mnémonique	Descrip de type - methodes	
byte	b	B	entier signés sur 8 bits (-2^7 à 2^7-1)
short	s	S	entier signés sur 16 bits (-2^15 à 2^15-1)
int	i	I	entier signés sur 32 bits
long	l	J	entier signés sur 64 bits
char	c	C	entiers ≥ 0 sur 16 bits (UNICODE : 0 à 65535)
float	f	F	flottants 32 bits
double	d	D	flottants 64 bits
class	a	L...;	référence à un objet

Piles

- La pile est l'endroit où sont rangées les frames (cadres) d'invocation des méthodes. Elle est gérée par les trois pointeurs frame, vars et optop. Il y a une pile par thread.
- ex :
  - empiler 5
  - emplier 10
  - addition
- Notation polonaise inverse (notation post-fixée)

NPI

- 3 x (4 + 7) s'écrit:
  - 4 [Ent] 7 + 3 x
  - 3 [Ent] 4 [Ent] 7 + x

qq bytecode

Type	mnémonique
Addition	iadd, ladd, fadd, dadd
Soustraction	isub, lsub, fsub, dsub
Multiplication	imul, lmul, fmul, dmul
Division	idiv, ldiv, fdiv, ddiv
Reste div. entière	irem, lrem

Jasmin

Projet Jasmin

- http://jasmin.sourceforge.net/
  - 1996
  - télécharger jasmin-2.4.zip (1.4 MB)
  - Exemples : Helloworld.j, Count.j, etc.
- java -jar jasmin.jar examples/Count.j
  - crée un fichier binaire Count.class (non lisilble directement) qui peut tourner sur une JVM
- java examples/Count
  - Attention : on n'écrit pas le .class

Syntaxe Jasmin

- Exemple
  - **bipush 5**

    5
  - **bipush 3**

    3

    5
  - **iadd**

    8

Syntaxe Jasmin

- Exemple
  - **bipush 4**

    4
  - **bipush 3**

    3

    4
  - **imul** (12) / **idiv** (1) / **isub** (1) / **irem** (1)

Syntaxe Jasmin

- Commentaire
  - point-virgule précédé d'un caractère blanc (espace, tab, retour à la ligne)
- Exemple

```
; addition de deux nbre
bipush 4 ; 1ère val
bipush 3 ; 2ème val
imul ; multiplication
```


Syntaxe Jasmin

- Format général d'une instruction
  - mnémonique paramètre(s) ; commentaire
- Directives
  - commence par un point
  - exemple:
    - .class public MaClass ;pour indiquer que le fichier définit la class MaClass
    - .limit stack 2 ;fixe la taille maxi de la pile à 2

Quelques exemples

- Expressions arithmétiques (à faire au tableau ^^)...

Rappel

- Java : 4 type d'entiers
  - long (64): -9,223,372,036,854,775,808 à 9 , 223,372,036,854,775,807
  - int (32): -2,147,483,648 to 2,147,483,647
  - short (16): - 32,768 to 32,767
  - byte (8): - 128 to 127

Syntaxe

- b: byte
  - bipush 37 -> ok
  - bipush 300 -> erreur
- s: short
  - sipush 300 -> ok
  - sipush 50000 -> erreur

Constante

- ldc arg: load constant (byte, short, int et float)
  - ldc 50000, ldc 37, ldc 300, ldc 3.14 -> ok
- ldc2_w arg: load constant (long et double)
  - ldc2_w 3.1415
- xconst_n: (pas d'arg) avec x = type, et
  - n = valeur entre -1 (m1) et 5 pour int : iconst_0
  - n = valeur entre 0 et 1 pour long/double
  - n = valeur entre 0 et 2 pour float
- aconst_null

Affichage en Java

- java.lang.System.out.println (8)
  - java.lang: package
  - System.out: champ statique «out» de la classe system du package
  - println: nom de la méthode
  - (8): paramètres
- javap java.lang.System
  - public static final java.io.PrintStream out;

Affichage en Bytecode

- getstatic java/lang/System/out Ljava/io/PrintStream
  - getstatic : obtenir un champ statique
  - java/lang/System/out : le champ statique
  - Ljava/io/PrintStream : le type de champ statique
- bipush 8
- invokevirtual java/io/PrintStream/println (I) V
  - invokevirtual : appel de méthode
  - java/io/PrintStream/println : méthode
  - (I) V : I = type du paramètre (int), V = type du résulat (void)

Exemple

- getstatic java/lang/System/out Ljava/io/PrintStream;
- bipush 4
- bipush 3
- idiv
- invokevirtual java/io/PrintStream/println(I)V

Conversion de type

- En Java:
  - 2+3.1 -> Ok, 2 est converti automatiquement (promotion de type) en 2.0 puis l'addition est faite
- En Java Bytecode:
  - pas de promotion de type, il faut explicitement faire les conversions avec
    - i2f, i2d, i2l, i2b, i2c, i2s
    - l2f, l2d, l2i
    - f2l, f2d, f2i
    - d2l, d2f, d2i

Squelette simple prg

```
.class public examples/addition
.super java/lang/Object
.method public static main([Ljava/lang/String;)V
    ; VOTRE CODE ICI
    return
.end method
```


Fixer la taille pile

- Directive «.limit stack n» au début de main

Dupliquer sommet pile

- **dup**
- **dup2**
- dup_x1
- dup_x2
- dup2_x1
- dup2_x2

Manipulation pile

- pop (supprime)
- pop2
- swap (permute)

Etiquettes - Label

- nomlabel :
  - nomlabel suivi du caractère :
  - nomlabel = séquence de caractères ne commençant pas par un chiffre, sans = ; . " - , différent d'un nom d'une instruction/directive
  - Sert à nommer un point précis du programme
  - Utilisé dans les instructions branchements

Branchements

- Pour aller à un point précis (label) du programme
- goto <label>
  - ex de boucle infinie
    - Label1 :
    - goto Label1

Branchements

- ifTest <label>
  - Va au label si le Test est vrai... (byte, short, int, char)
  - Test =
    - Entre 2 var: _icmpgt (>), _icmplt (<), _icmpge (≥), _icmple (≤), _icmpeq (=), _icmpne (#)
    - Entre var et 0: gt (>), lt (<), ge (≥), le (≤), eq (=), ne (#)

Branchements

- long : lcmp
  - condition : v1 et v2 long
  - resultat: v1 et v2 retiré de la pile et remplacé par:
    - 1 si v1 > v2
    - 0 si v1 = v2
    - -1 si v1 < v2
- (à utiliser avec ifgt, iflt, ifeq, ...)
- float : fcmpg fcmpl - double : dcmpg dcmpl

Variables locales

- Stockage de données de la JVM
  - Pile
  - Tableau de variables locales (à la méthode en cours d'éxecution) : 65,536 local variables
    - Limitable par «.limit locals 5»

Variables locales

- Tableau de variables locales (à la méthode en cours d'éxecution) indexé par des entiers
  - Indice première case : 0
  - istore, lstore, fstore, dstore
    - retire la valeur au sommet de la pile et la stocke dans une variable locale
  - iload, lload, fload, dload
    - copie le contenu d'une variable au sommet de la pile

Exemple

- iconst_2
- istore 0
- .
- .
- .
- iload 0

Notes

- Double et Long prennent l'équivalent de 2 variables
- De 0 à 3: on peut utiliser «fload_3» au lieu de «fload 3»
- Variables non typés
  - le slot 2 peut être int au début, float au milieu et 1/2 double à la fin du prg)

Char

- Caractères : UNICODE
  - getstatic java/lang/System/out Ljava/io/PrintStream;
  - ldc 65 ; equivalent de «A» en unicode
  - invokevirtual java/io/PrintStream/println(C)V

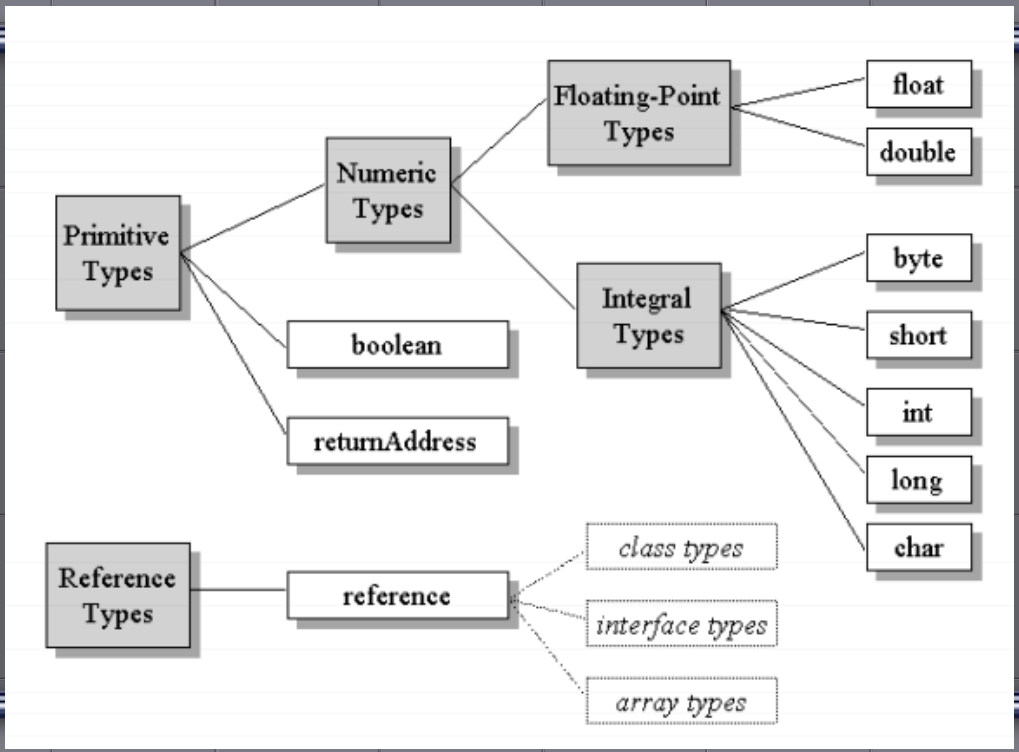
String

- Chaîne de caractères
  - classe string (java.lang.String)
    - liste des méthodes avec «javap java.lang.String»
  - constantes avec les " "
    - ldc "coucou"
  - Affichage avec Ljava/lang/String;
    - invokevirtual java/io/PrintStream/println(Ljava/lang/String;)V
  - Stockage
    - aload, astore

Notes

- 5 types de bases
  - int (32-bit signed integer)
  - long (64-bit signed integer)
  - float (32-bit floating-point number)
  - double (64-bit double precision floating-point number)
  - reference

Notes



Notes

- reference (astore, aload)
  - référence à un objet (Car, Rectangle, String, etc...)
  - = pointeur

Méthodes

- Chaque méthode dispose de sa propre pile et tableau de variable locale
- Définition
  - .method public static nom(type_paramètres) type_résultat
  - .limit stack 99
  - .limit locals 99
  - ; VOTRE CODE
  - return
  - .end method

Méthodes

- Les paramètres
  - stockés dans les var locales de la méthodes
- Appel à une méthode
  - Placer les paramètres au sommet de la pile
  - invokestatic nomClass/ nomMethod(type_param)type_resul
  - Le résultat (s'il y en a un) est stocké au sommet de la pile, en remplacement des paramètres

Tableaux (Arrays)

- Tableau= suite de case mémoire adjacentes
- En JBC:
  - Tous les éléments d'un tableau sont du même type
  - L'indice du premier élement est 0
  - Les tableaux ont une taille fixe et sont de la forme :

4 octets
(32 Bits)

Tableaux

- Création d'un tableau: newarray <type_elements>
  - la taille doit être spécifié avant, les éléments sont initialisés à une valeur par défaut (ex. pour int: 0)
    - bipush 6
    - newarray int
  - Place en haut de la pile une référence vers le tableau

6	int	0	0	0	0	0	0

Tableaux

- arraylength
  - Prend la référence d'un tableau au sommet de la pile et la remplace par la taille du tableau
- bastore, aastore, fastore, dastore, etc.
  - Ecrit une valeur dans le tableau (voir exemple)
- baload, aaload, faload, daload, etc.
  - Lis une valeur dans le tableau (voir exemple)

Exemples

- bipush 6
- newarray float
- getstatic java/lang/System/out Ljava/io/PrintStream;
- swap
- arraylength
- invokevirtual java/io/PrintStream/println(I)V

Exemples

- bipush 6
- newarray float ; création d'un tableau de 6 floats
- astore 0 ; pour garder l'adresse en mémoire du tableau
- aload 0 ; pour faire le rappel du tableau
- iconst_2
- ldc 7.0
- fastore ; stockage de 7.0 dans la case 2
- ...
- aload 0 ; pour faire le rappel du tableau
- iconst_2
- faload ; pour mettre au sommet de la pile la valeur de la case 2.

Tableaux & méthodes

- [type_éléments
  - .method public static afficherTableauFloat ([F)V
  - .method public static creerTableauFloat (IF)[F
  - .method public static main([Ljava/lang/String;)V

Paramètres en LC

- % java monFichier test1 test2

- .method public static main([Ljava/lang/String;)V
- .limit stack 5
- .limit locals 10
- getstatic java/lang/System/out Ljava/io/PrintStream;
- aload 0
- iconst_1
- aaload
- invokevirtual java/io/PrintStream/println(Ljava/lang/String;)V
- return
- .end method

A vous de jouer

- TP2 à refaire avec des paramètres en ligne de commandes pour
  - Paques
  - Factorielle
  - Fibonacci
  - Schéma

Références

A lire...

- http://flylib.com/books/en/2.354.1.1/1/
- http://www.artima.com/insidejvm/ed2/index.html
- http://www.javafr.com/tutoriaux/JAVA-BYTECODE-COMPRENDRE-RESULTAT-VOS-COMPILATIONS_110.aspx
- http://www.cs.sjsu.edu/~pearce/modules/lectures/co/jvm/jasmin/
- http://en.wikipedia.org/wiki/Java_bytecode_instruction_listings
- http://andrei.gmxhome.de/bytecode/index.html

Fin

After this there is not turning back

You can take the red pill, wake up in your bed and believe what ever you want to believe or you can take the blue pill, stay in wonder-land and I will show you how deep the holerabbit goes