Conception de systèmes
intégrés matériels et logiciels B2

jean-michel Douin,douin@cnam.fr

Cnam, Décembre 1996

La machine Java

Une première approche

Version du 19 Mars 1998

Bibliographie

- [LY96]T.Lindholm,F.Yellin. The Java Virtual machine Specification. The Java Series Addison Wesley. 1996.
- The VM specification.http://java.sun.com:81/docs/books/vmspec/html
- Présentation PowerPoint de Axel Kramer http://www.well.com/user/axel
- www.gamelan.com, recherche de: "Java Virtual Machine"
- La machine Kaffe de Tim Wilkinson, http://www.sarc.city.ac.uk/~tim/kaffe

Interpréteurs et machine à pile

- N. Wirth.Algorithms+Data Structures=Programs,Chap 5 pp 280-347. Prentice Hall. 1976.(La machine P-code).
- N.Wirth. LILITH Modula workstation.Rapport ETH n° xxxxxxxx 1982. (La machine M-code).

Processeurs Java

- PicoJava: The Java Virtual Machine in Hardware. M.Tremblay Sun Microelectronics. support de l'exposé effectué à JavaOne (voir également microJava et ultraJava)
- Java Based Devices from Mitsubishi,M32R/D. E. Nguyen. exposé JavaOne
- voir Digital StrongARM,...

Processeurs basés sur une machine à pile

- D.A.P.Mitchell,J.A.Thomson,G.A.Manson,G.R.Brookes.Inside the Transputer.BlackWell Scientific Publications. 1990
- ST20450, 32 bit microprocessor. Doc SGS-Thomson, May 1995. http://www.st.com/....

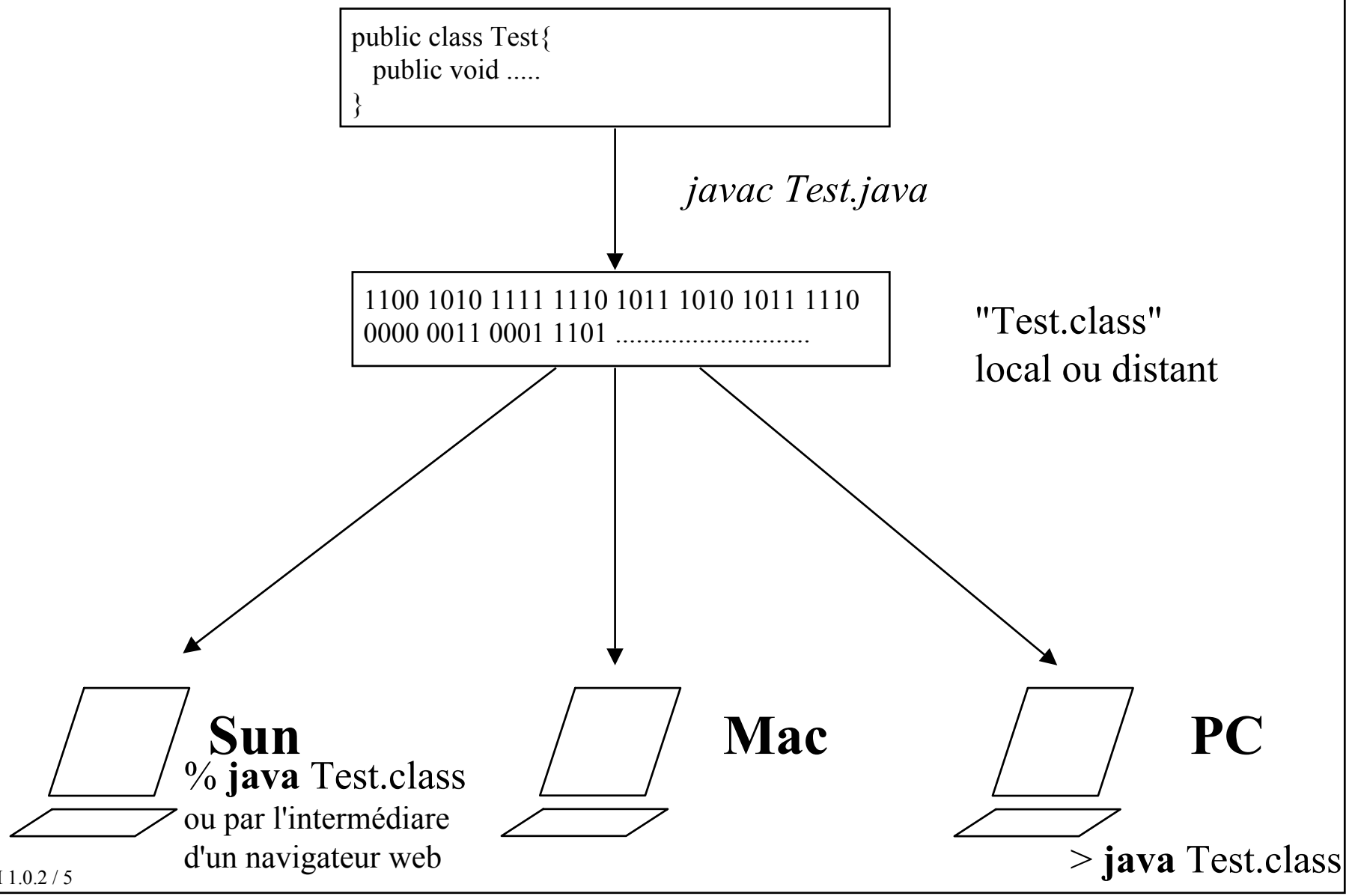
Sommaire

- Java le langage vers la machine
- Architecture de la machine virtuelle
- Le fichier généré ".class"
- Le jeu d'instructions
- Un interpréteur en C
  - Chargeur, éditeur de liens, Initialisation de la machine
  - Création, affectation des champs d'instances ou de classe
  - Segment d'activation, méthodes statiques et virtuelles,
  - Traitements des exceptions (machine et utilisateur)
- Une première implantation en VHDL
- Annexes : statistiques, un désassembleur, un chargeur-interpréteur de "bytecode"

Java le langage vers la machine

- Classes, Variables de classes
  - "Constructeurs" de classe
- Instances, Variables d'instances
- Invocation de méthodes, héritage
  - Méthodes de classes
  - Méthodes virtuelles et finales
  - Constructeurs, (destructeur et ramasse-miettes)
- Interfaces
- Exceptions
- Thread
- Appel de code natif

Objectifs


public class Test{
public void .....
}
javac Test.java
1100 1010 1111 1110 1011 1010 1011 1110
0000 0011 0001 1101 .....
"Test.class"
local ou distant
Sun
% java Test.class
ou par l'intermédiare
d'un navigateur web
Mac
PC
> java Test.class

Architecture de la machine virtuelle

- Types de données
- Les registres
- La pile d'exécution et la mémoire
  - constant pool,
  - Interface,field,methods
- Le ramasse miettes

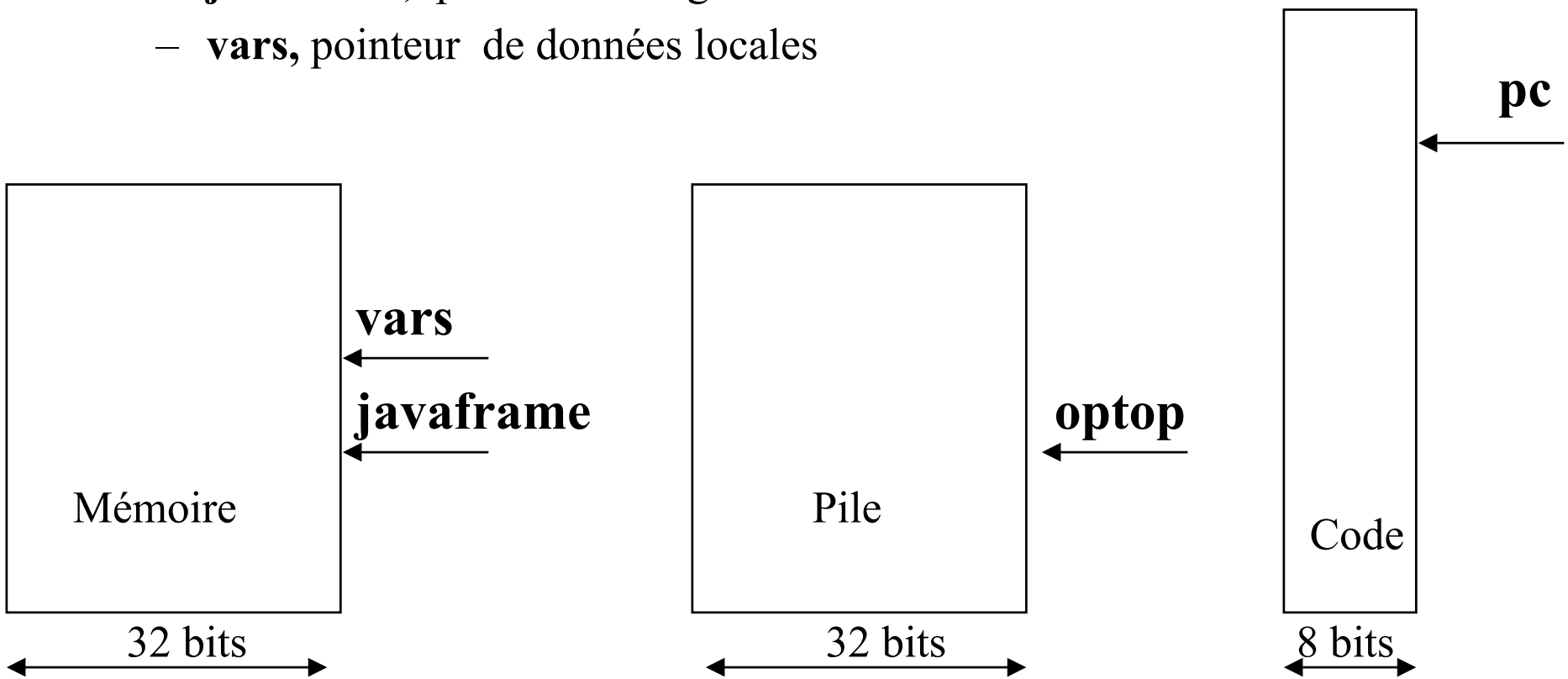
Types de données

- **byte :** 8 bits en complément à 2
- **short :** 16 bits en complément à 2
- **int :** 32 bits en complément à 2
- **long :** 64 bits en complément à 2
- **char :** 16 bits format Unicode
- **float :** 32 bits IEEE 754
- **double :** 64 bits IEEE 754
- **"adresse" :** 32 bits

- En général la taille des mots d'une machine virtuelle Java est de 32 bits

Les registres

- 4 registres
  - **pc**, compteur ordinal
  - **optop**, pointeur de pile
  - **javaframe**, pointeur de segment d'activation
  - **vars,** pointeur de données locales



La pile d'exécution et la mémoire

Aspects dynamiques

- Une pile d'exécution en mots de 32 bits,
  - Exemple : iadd == > push(pop() + pop()); ***(optop)***
- Une pile de segments d'activation
  - Appels de méthodes ***(java frame & vars)***
- Une zone mémoire est réservée pour les instances
  - Pas de déallocation programmée, gèrée par un ramasse-miettes

Aspects "statiques"

- Une zone mémoire des méthodes de chaque classe ***(pc)***
- Une zone mémoire des variables de classe

Le fichier généré ".class"

- Prémisses
- Format du fichier généré
- Le constant pool
- Informations sur l'interface(s) utilisée(s) par cette classe
- Description des champs des variables de classe ou d'instances
- Description des méthodes
- Description de ce fichier

:- Table des symboles, sans choix d'implantation ...

:- (résultat de la compilation après un "front-end")

Prémisses

- Compatibilité binaire chapitre 13 Java Specification
- Gestion de projets et internet
- Quelles sont les contraintes de compatibilité binaire ?

- ? Peut-on ajouter de nouvelles méthodes ou variables d'instance d'une classe tout en garantissant l'exécution d'une application antérieure ?

- ==> Forme symbolique du .class

Prémisses

- ajout de nouveaux champs
- ajout de nouvelles classes dans un package
- modification du type de déclaration
- modification du graphe d'héritage
- ...

Format du ".class"

- ClassFile {

– u4 magic; – u2 minor_version; – u2 major_version;	*Entête du fichier*
– u2 constant_pool_count; – cp_info *constant_pool;	*Constantes et signatures*
– u2 access_flags; – u2 this_class; – u2 super_class;	*"type" de la classe* *son nom,* *le nom de la super-classe*
– u2 interfaces_count; – u2 *interfaces;	*Liaison avec les interfaces*
– u2 fields_count; – field_info *fields;	*Variables de la classe ou d'instances*
– u2 method_count; – method_info *methods;	*Les méthodes*
– u2 attributes_count; – attribute_info *attributes;	*Description de ce fichier*

}

Le constant_pool

- *cp_info *constant_pool;*
- typedef struct {
- u1 tag;
- u1 *info;
- }**cp_info;**

```
#define CONSTANT_Class              7
#define CONSTANT_Fieldref           9
#define CONSTANT_Methodref          10
#define CONSTANT_String             8
#define CONSTANT_Integer            3
#define CONSTANT_Float              4
#define CONSTANT_Long               5
#define CONSTANT_Double             6
#define CONSTANT_InterfaceMethoderef 11
#define CONSTANT_NameAndType        12
#define CONSTANT_Asciz              1
#define CONSTANT_Utf8               1
```

- Exemple :
  - **si** pool_constant[i] est un entier
  - assert(pool_constant[i].tag == 3);
  - pool_constant[i]->info == valeur de cet entier constant

```
typedef struct{
        u1 tag;
        u4 bytes;
}CONSTANT_Integer_info;
```

u1 : un octet, u4 : 4 octets

Un exemple

```
class bulbe{
  public static void main( String Args[]){
    int [] n = new int[6];
    n[0]=0;n[1]=2;n[2]=1;n[3]=3;n[4]=4;n[5]=1;


    boolean sorted = false;
    while(!sorted){
      sorted = true;
      for(int i = 0; i < 5; i++){
        if (n[i] > n[i + 1]){
          int temp = n[i];
          n[i] = n[i + 1];
          n[i + 1] = temp;
          sorted = false;
        }
      }
    }}
```

Un exemple de constant_pool

- pool_count : 31
- [1] tag: 7 name_index: 9
- [2] tag: 7 name_index: 20
- [3] tag: 10 class_index: 2 name_and_type_index: 4
- [4] tag: 12 class_index: 24 descriptor_index: 28
- [5] tag: 1 length: 4 this
- [6] tag: 1 length: 1 Z
- [7] tag: 1 length: 13 **ConstantValue**
- [8] tag: 1 length: 7 Lbulbe;
- [9] tag: 1 length: 5 bulbe
- [10] tag: 1 length: 18 **LocalVariableTable**
- [11] tag: 1 length: 4 temp
- [12] tag: 1 length: 10 **Exceptions**
- [13] tag: 1 length: 10 bulbe.java
- [14] tag: 1 length: 15 **LineNumberTable**
- [15] tag: 1 length: 1 I
- [16] tag: 1 length: 10 **SourceFile**
- [17] tag: 1 length: 14 **LocalVariables**
- [18] tag: 1 length: 4 **Code**
- [19] tag: 1 length: 4 Args
- [20] tag: 1 length: 16 java/lang/Object
- [21] tag: 1 length: 4 main

Suite du constant_pool

- [22] tag: 1 length: 22 ([Ljava/lang/String;)V
- [23] tag: 1 length: 4 trie
- [24] tag: 1 length: 6 <init>
- [25] tag: 1 length: 6 sorted
- [26] tag: 1 length: 1 n
- [27] tag: 1 length: 2 [I
- [28] tag: 1 length: 3 ()V
- [29] tag: 1 length: 1 i
- [30] tag: 1 length: 19 [Ljava/lang/String;

- pool_constant[0] est réservé

access_flag, this_class, super_class

- #define ACC_PUBLIC 0x0001 /* accessible par tous */
- #define ACC_FINAL 0x0010 /* pas de sous-classe */
- #define ACC_SUPER 0x0020 /* obsolète */
- #define ACC_INTERFACE 0x0200 /* c'est une interface */
- #define ACC_ABSTRACT 0x0400 /* ne peut-être instancié */

- this_class
  - Indice dans le constant_pool, (nom de la classe)
    Indice 1 pour l'exemple (tag 7)
- super_class
  - Indice dans le constant_pool, (nom de la classe),
    Indice 2 pour l'exemple (tag 7)

  (toutes les classes héritent de java/lang/Object)

field_info

- typedef struct{
- u2 access_flags;
- u2 name_index; /* indices */
- u2 descriptor_index; /* dans le constant_pool */
- u2 attributes_count;
- ConstantValue_attribute *attributes;
- }**field_info;**

- typedef struct{
- u2 attribute_name_index;
- u4 attribute_length;
- u2 constantvalue_index;
- } **ConstantValue_attribute;**

Lecture des descripteurs de "Field"

- *FieldType ::= BaseType* | *ObjectType* | *ArrayType*
- *BaseType*
  - **B** byte
  - **C** char
  - **D** double
  - **F** float
  - **I** int
  - **J** long
  - **S** short
  - **Z** boolean
- *ObjectType*
  - **L**<classname>**;**
- *ArrayType*
  - **[** table

Exemples :
double m[] [] --> **[[D**

Strings Args[] --> **[Ljava/lang/String;**

Field

- Fields
  - recense tous les champs d'une classe
  - Statiques
    - fields[i].access_flag & ACC_STATIC == ACC_STATIC
  - ou locaux à chaque instance

  - Note d'implantation :
    - Les types B,C,F,I,L et [occupent un mot machine (32 bits)
    - Les types D et J occupent 2 mots

method_info

- typedef struct{
- u2 access_flags;
- u2 name_index;
- u2 descriptor_index;
- u2 attributes_count;
- Code_attribute *attributes;
- }**method_info;**

method_info.Code_attribute

```
typedef struct{
        u2 start_pc;
        u2 end_pc;
        u2 handler_pc;
        u2 catch_type;
      } exception_info;
```

```
typedef struct{
        u2 attribute_name_index;
        u4 attribute_length;
        u2 max_stack;
        u2 max_locals;
        u4 code_length;
        u1 *code;
        u2 exception_table_length;
        exception_info *exception_table;
        u2 attributes_count;
        attribute_info *attributes;
      } Code_attribute;
```

Sur l'exemple

analyse de 'methods'

- method_count: 2
- method.access_flags: 0x9 /* main */
- method.name_index: 21
- method.descriptor_index: 22
- method.attributes_count: 1
- attribute_name_index: 18
- attribute_length: 297
- code : 10,6,bc,a,.....3e,b1,

[18] tag: 1 length: 4 Code
[21] tag: 1 length: 4 main
[22] tag: 1 length: 22 ([Ljava/lang/String;)V

- method.access_flags: 0x1 /* <init> */
- method.name_index: 24
- method.descriptor_index: 28
- method.attributes_count: 1
- attribute_name_index: 18
- attribute_length: 47
- code : 2a,b7,0,3,b1,

[24] tag: 1 length: 6 <init>
[28] tag: 1 length: 3 ()V

Lecture des descripteurs de "method"

- *MethodDescriptor ::= **(** FieldType ***)** ReturnDescriptor*
- *ReturnDescriptor ::= FieldType **V***
    - **V** si le type retourné est void

Exemples :
Object m(int i, double d, Thread T)
--> **(IDLjava/lang/Thread;)Ljava/lang/Object;**

void main(String Args[]) --> **([Ljava/lang/String;)V**

méthodes d'intialisation

- <init>V
  - Constructeur par défaut de chaque instance
  - Sur l'exemple "bulbe.<init>V" est présent, son code déclenche l'appel du constructeur de la classe java.lang.Object

- <clinit>V
  - méthode d'initialisation d'une classe
  - Affectations de variables de classe .
  - (exécutée une seule fois)

method_info.Code_attribute.attributes

```
typedef struct{         /* p 115 */
        u2 attribute_name_index;
        u4 attribute_length;
        u2 line_number_table_length;
        line_number_info *line_number_table;
      }LineNumberTable_attribute;
```

--> informations destinées au débogueur

```
typedef struct{                 /* p 116 */
        u2 attribute_name_index;
        u4 attribute_length;
        u2 local_variable_table_length;
        local_variable_info *local_variable_table;
      }LocalVariableTable_attribute;
typedef struct{
        u2 start_pc;
        u2 length;
        u2 name_index;
        u2 descriptor_index;
        u2 index;
      }local_variable_info;
```

ClassFile.attributes

- typedef struct{
- u2 attribute_name_index;
- u4 attribute_length;
- u2 sourcefile_index;
- } **SourceFile_attribute;**

- Sur l'exemple
- analyse de 'attributes'
- attributes_count: 1
- source_attribute.name_index : 16
- source_attribute.length : 2
- source_attribute.sourcefile_index : 13

constant_pool
[13] tag: 1 length: 10 bulbe.java
[16] tag: 1 length: 10 SourceFile

Le jeu d'instructions

- Machine à pile
- Instructions
  - Gestion de la pile constantes et variables
  - Gestion des tableaux
  - Opérations sur la pile
  - Instructions arithmétiques et logiques
  - Opérations de conversions
  - Instructions de contrôles
  - Appels et retours de méthodes
  - Instructions sur les instances
  - Exceptions et Instructions "_quick"

Machine à pile : Un premier exemple

```
class Exemple {
  public static void main( String args[]) {
    int compte=1, resultat=1;
    int valeur = 1;

      compte = compte + 1;
      resultat = resultat * compte;
  }
}
                /* Exemple.main.([Ljava/lang/String;)V */
 /*  0*/ iconst_1,          empiler ( 1)
 /*  1*/ istore_1,          compte = depiler()          compte=1
 /*  2*/ iconst_1,          empiler ( 1)
 /*  3*/ istore_2,          resultat = depiler()        resultat=1
 /*  4*/ iload_1,           empiler ( compte )
 /*  5*/ iconst_1,          empiler ( 1)
 /*  6*/ iadd,              empiler ( depiler() + depiler())
 /*  7*/ istore_1,          compte = depiler()          compte=compte + 1
 /*  8*/ iload_2,           empiler ( resultat )
 /*  9*/ iload_1,           empiler ( compte )
 /* 10*/ imul,              empiler ( depiler() * depiler())
 /* 11*/ istore_2,          resultat = depiler()   resultat=resultat*compte
 /*12*/ returnn,
```

Format d'une instruction

- Le code opération des instructions est sur 8 bits
  - exemple : istore,

0x36					

- Une instruction possède de 0 à 4 opérandes*
  - exemple : istore

0x36	0x01				

  - [iconst0, istore,0,dup,istore,1]
  -

0x03	0x36	0	0x59	0x36	0x01

- Manipulation implicite de la pile
  - --> un code plus compact / machine à registre

* il y a des quelques exceptions (3) à cette règle ! (switch, wide)

Format d'une instruction,[LY96] p 275

- Operation Store int into local variable
- Format

istore
index

- Forms *istore* = 54(0x36)
- Stack …,value => …
- Description : .....

Syntaxe des instructions

- Txxx
  - T indique le type des opérandes

- **i**store
  - Rangement d'un entier
- **l** long
- **d** double
- **f** float
- **c** char
- **a** object

Gestion de la Pile:Constantes et variables

- Lecture d'une variable locale placée sur la pile
  - ***i****load,****i****laod_<n>, ****l****load,lload_<n>, ****f****load, ****f****load_<n>,* ***d****load,****d****load_<n>, ****a****load,****a****load_<n>*
- Ecriture d'une variable locale depuis la pile
  - *istore,istore_<n>,lstore,lstore_<n>,fstore, fstore_<n>, dstore,dstore_<n>, astore,astore_<n>*
- Lecture d'une constante placée sur la pile
  - *bipush,sipush,ldc,ldc_w,ldc2_w,aconst_null,iconst_m1, iconst_<i>,lconst_<l>,dconst_<d>,*
- Accès avec déplacement supérieur à 16 bits
  - *wide*

<n> valeur de l'opérande, 0 à 5

Gestion des tableaux

- Création de tableaux
  - *newarray, anewarray,multianewarray*
- Accès en lecture
  - *baload,caload,salolad,iaload,laload,faload,daload,aaload*
- Accès en écriture
  - *bastore,castore,sastore,iastore,lastore,fastore,dastore, aatore*
- longueur du tableau
  - *arraylength*

Opérations sur la pile

- retrait du sommet
  - *pop, pop2*
- duplication du sommet de pile
  - *dup,dup,dup_x1,dup2_x1,dup2_x2*
- échange du sommet de pile et du sous-sommet
  - *swap*

Instuctions arithmétiques et logiques

- Addition/soustraction
  - *iadd,ladd,fadd,dadd/isub,lsub,fsub,dsub*
- multiplication,division
  - *imul,fmul,dmul,lmul,idiv,fdiv,ddiv,ldiv,irem,frem,drem,lrem*
- complément
  - *ineg,fneg,dneg,lneg,*
- décalage
  - *ishl,ishr,iushr,lshl,lshr,lushr*
- Opérations
  - *ior,lor,iand,land,ixor,lxor*
- Incrément de variable locale
  - *iinc*

Opérations de conversions

- int en long, float ou double
  - *i2l,i2f,i2d*
- long en float ou double
  - *l2f,l2d*
- float en double
  - *f2d*

Instructions de contrôles

- Sauts conditionnels
  - *ifeq,iflt,ilfe,ifgt,ifge,ifnull,ifnonnull,if_icmpeq,if_icmpne, if_icmplt,if_icmpgt,if_icmple,if_icmpge,if_acmpeq,if_acmpne, lcmp,fcmpl,fcmpg,dcmpl,dcmpg*
- Sauts inconditionnels
  - *goto,goto_w,jsr,jsr_w,ret*
- Accès indirect
  - *tableswitch,lookupswitch*

Appels de méthodes & valeurs retournées

- Appel d'une méthode (par défaut virtuelle en java)
  - *invokevirtual*
- Appel d'une méthode implantée par une interface
  - *invokeinterface*
- Appel d'une méthode de la classe super ou privée
  - *invokespecial*
- Appel d'une méthode de classe
  - *invokestatic*
- Valeurs retournées
  - *ireturn,lreturn,dreturn,areturn, return*(void)

Instructions sur les instances

- Création d'un objet
  - *new*
- Accès aux champs d'instances
  - *getfield,putfield,getstatic,putstatic*
- Accès aux champs des variables de classe
  - *getstatic,putstatic*
- Contrôles de propriétés (changement de type)
  - *instanceof, checkcast*

Exceptions et Synchronisation

- Levée d'une exception
  - *athrow*
- Synchronisation de blocs d'instruction
  - *monitorexit,monitorenter*

Instructions "_quick" & JIT

- Une optimisation en JIT(Just in Time)
- Réécriture (à la volée) de certaines instructions,
  - Les accès au "constant pool" sont résolus une seule fois

- JIT(Just in Time) en code natif : machine Kaffe de Tim Wilkinson
  - KAFFE v0.5.6 - A JIT and interpreting virtual machine to run Java(tm)* code
  - JIT en code natif (i386)

Instructions "_quick"

- case ldc_quick, case ldc_w_quick, case ldc2_w_quick
- case anewarray_quick, case multianewarray_quick
- case putfield_quick, case putfield2_quick
- case getfield_quick, case getfield2_quick
- case putstatic_quick, case putstatic2_quick
- case getstatic_quick, case getstatic2_quick : /* 212 */
- case getfield_quick_w : /* 227 */
- case invokevirtual_quick : /* 214 */
- case invokenonvirtual_quick : /* 215 */
- case invokesuper_quick : /* 216 */
- case invokestatic_quick : /* 217 */
- case invokeinterface_quick : /* 218 */
- case invokevirtualobject_quick : /* 219 */
- case invokevirtual_quick_w : /* 226 */
-

- case new_quick : /* 221 */
- case checkcast_quick : /* 224 */
- case instanceof_quick : /* 225 */

Table 3.1 page 73 [LY96], types et instructions

opcode	byte	short	int	long	float	double	char	reference
Tipush	bipush	sipush						
Tconst			iconst	lconst	fconst	dconst		aconst
Tload			iload	lload	fload	dload		aload
Tstore			istore	lstore	fstore	dstore		astore
Tinc			iinc					
Taload	baload	saload	iaload	laload	faload	daload	caload	aaload
Tastore	bastore	sastore	iastore	lastore	fastore	dastore	castore	aastore
Tadd			iadd	ladd	fadd	dadd		
Tsub			isub	lsub	fsub	dsub		
Tmul			imul	lmul	fmul	dmul		
Tdiv			idiv	ldiv	fdiv	ddiv		
Trem			irem	lrem	frem	drem		
Tneg			ineg	lneg	fneg	dneg		
Tshl			ishl	lshl				
Tshr			ishr	lshr				
Tushr			iushr	lushr				
Tand			iand	land				
Tor			ior	lor				
Txor			ixor	lxor				
i2T	i2b	i2s		i2l	i2f	i2d		
l2T			l2i		l2f	l2d		
f2T			f2i	f2l		f2d		
d2T			d2i	d2l	d2f			
Tcmp				lcmp				
Tcmpl					fcmpl	dcmpl		
Tcmpg					fcmpg	dcmpg		
if_TcmpOP		if_icmpOP				if_acmpOP		
Treturn			ireturn	lreturn	freturn	dreturn		areturn

Un interpréteur en C

- Sommaire
  - Algorithme général
  - Structures de données
    - Registres
    - Implantations des classes et méthodes (aspects statiques)
    - Environnement et pile d'exécution (aspects dynamiques)
  - Objets et instances
    - Accès aux champs
    - Appels de méthodes virtuelles (par défaut en Java)
    - Le ramasse miettes

Un interpréteur en C

- Algorihme principal

  – **do**{

```
ir = next();   /* lecture de l'instruction */
switch (ir) {
    case(....
    case(iadd) : ipush( ipop() + ipop()); break;
    case(....
    case(iand) :  ipush( ipop() & ipop()); break;
    case(....
    case(....
```

  – } **while**(true);

Quels registres ?

- pc
- vars
- javaframe
- optop

- cl : *Numéro de la classe courante*
- me : *Numéro de la méthode courante issue de cette classe*

- cc : *méta-registre : désigne la structure de la classecourante*

Un interpréteur en C, syntaxe

- lecture mémoire et incrémentation du *pc* *pc*
  - u1 next(void);
  - u2 next2(void);
  - u4 next4(void);

- Opérations sur la pile "typées" (par le nom) *optop*
  - int_t **i**pop(void); **f**pop,**d**pop,**l**pop,**a**pop
  - void **i**push(int_t); **f**push,**d**push,**l**push,**a**push

- Opérations sur la mémoire "typées" *vars*
  - int_t iGetLocalVariable(void); fGetLocalVariable,dGet..
  - void iSetLocalVariable(int_t); fSetLocalVariable,....

Gestion de la Pile: Constantes

```
•     case bipush : /* 16 */
•                         byte1 = next();
•                         ipush((int_t) byte1);
•                         break;
•
```

The Java Virtual Machine Documentation SUN 15 Mars 1995
2 The Virtual Machine Instruction Set page 7

bipush

Push one-byte signed integer

Syntax : *bipush* = 16
byte1

Stack: ... => ..., *value*

byte1 is interpreted as a signed 8-bit *value*. This *value* is expanded to an integer and pushed onto the operand stack

Gestion de la Pile: Constantes

- case **sipush** : /* 17 */
- byte1 = next(); byte2 = next();
- ipush(((int_t) byte1 << 8) | (0xFF & byte2));
- break;

- case **ldc** : /* 18 */
- indexbyte1 = next();
- pushconstantpool(indexbyte1);
- break;
-
- /*
- **ldc_w,ldc2_w,aconst_null,iconst_m1,**
- **iconst_0,iconst_1,iconst_2,iconst_3,iconst_4,iconst_5,**
- **lconst_0,lconst_1,fconst_0,fconst_1,fconst_2,**
- **dconst_0,dconst_1,**
- */

Gestion de la Pile: Variables(1)

- case **iload** : /* 21 */
- uindex = next();
- ipush(iGetLocalVariable(uindex));
- break;
-
- case **iload_0** : /* 26 */
- ipush(iGetLocalVariable(0));
- break;
-
- /*
- **iload_1,iload_2,iload_3,**
- **lload,lload_0 :,lload_1,lload_2,lload_3,**
- **fload,fload_0,fload_1,fload_2,fload_3,**
- **dload,dload_0,dload_1,dload_2,dload_3,**
- **aload,aload_0,aload_1,aload_2,aload_3**
- */

Gestion de la Pile: Variables(2)

- case **iload** : /* 21 */
- uindex = next();
- ipush(iGetLocalVariable(uindex));
- break;
-
- case **iload_0** : /* 26 */
- ipush(iGetLocalVariable(0));
- break;

- case **aload** : /* 25 */
- uindex = next();
- apush(aGetLocalVariable(uindex));
- break;

Gestion de la Pile: Variables(3)

```
case istore    :  /* 54 */
        uindex = next();
        iSetLocalVariable(uindex,ipop());
        break;

case istore_0  :  /* 59 */
        iSetLocalVariable(0,ipop());
        break;

/*istore_1,istore_2,istore_3 */

case lstore    :  /* 55 */
        uindex = next();
        lSetLocalVariable(uindex,lpop());
        break;

case lstore_0  :  /* 63 */
        lSetLocalVariable(0,lpop());
        break;

/* lstore_1,lstore_2,lstore_3  */
```

Un petit exemple

- int i,j,k;
- i = 2;
- j = i;
- ..
- iconst_2,
- istore_0,
- iload_0,
- istore_1,

voir également en Annexe 1

Gestion de la Pile: Variables(4)

- case **fstore** : /* 56 */
- uindex = next();
- fSetLocalVariable(uindex,fpop());
- break;
-
- /* **fstore_0,fstore_1,fstore_2,fstore_3** */
- /* **dstore,dstore_0,dstore_1,dstore_2,dstore_3** */
- /* **astore,astore_0,astore_1,astore_2,astore_3** */

- case **iinc** : /* 132 */
- vindex = next();
- constant = next();
- iSetLocalVariable(vindex,iGetLocalVariable(vindex)+constant);
- break;

Gestion des tableaux

```
case newarray       :  /* 188 */
        atype = next();
        size = ipop();
        apush(AllocArray(atype,size));
        break;

/* anewarray,multianewarray,arraylength */

case iaload         :  /* 46 */
        index  = ipop();
        arrayref  = apop();
        if (arrayref == null)
            GenerateException(NullPointerException);
        if (index < 0 || index >= arrayref->borne)
            GenerateException(ArrayIndexOutOfBoundsException);
        zonei = (int_t *) arrayref->zone;
        ipush(zonei[index]);
        break;

/* laload,faload,daload,aaload,baload,caload,saload */
```

Gestion des tableaux(2)

```
case lastore        :  /* 80 */
        lvalue = lpop();
        index  = ipop();
        arrayref  = apop();
        if (arrayref == null)
            GenerateException(NullPointerException);
        if (index < 0 || index > arrayref->borne)
            GenerateException(ArrayIndexOutOfBoundsException);
        zonel = (long_t *) arrayref->zone;
        zonel[index] = lvalue;
        break;

/* iastore,fastore,dastore,aastore,bastore,castore,
sastore */
```

Exemple d'instructions sur les tableaux

```
int[] Tablei =new int[10];

for (int i=0;i<10;i++){
  Tablei[i] = i;
}
for (int i=0;i<10;i++){
  Tablei[i] = Tablei[i]*Tablei[i];
}

char[] Tablec =new char[10];
for (int i=0;i<10;i++){
  Tablec[i] = (char)(i+'0');
}
```

Le code généré

- /*0*/ bipush, 10,
- /*2*/ newarray, 10,/*int*/
- /*4*/ astore_1,
- /*5*/ iconst_0,
- /*6*/ istore_2,
- /*7*/ gotoo, 0x00, 17,
- /*10*/ aload_1,
- /*11*/ iload_2,
- /*12*/ iload_2,
- /*13*/ iastore,
- /*14*/ iinc, 2, 1,
- /*17*/ iload_2,
- /*18*/ bipush, 10,
- /*20*/ if_icmplt,0xFF,10-20,
- /*23*/ iconst_0,
- /*24*/ istore_3,
- /*25*/ gotoo, 0x00, 41,
- /*28*/ aload_1,
- /*29*/ iload_3,
- /*30*/ aload_1,
- /*31*/ iload_3,
- /*32*/ iaload,
- /*33*/ aload_1,
- /*34*/ iload_3,
- /*35*/ iaload,
- /*36*/ imul,
- /*37*/ iastore,
- /*38*/ iinc, 3, 1,
- /*41*/ iload_3,
- /*42*/ bipush, 10,
- /*44*/ if_icmplt,0xFF,28-44,
- /*47*/ bipush,10,
- /*49*/ newarray,5,/*char*/
- /*51*/ astore,4,
- /*53*/ iconst_0,
- /*54*/ istore,5,
- /*56*/ gotoo,0x00,73-56,
- /*59*/ aload,4,
- /*61*/ iload,5,
- /*63*/ iload,5,
- /*65*/ bipush,48,
- /*67*/ iadd,
- /*68*/ int2char,
- /*69*/ castore,
- /*70*/ iinc, 5,1,
- /*73*/ iload,5,
- /*75*/ bipush,10,
- /*77*/ if_icmplt,0xFF,59,

Opérations sur la pile

```
case pop     :  /* 87 */
        any = ipop();
        break;

case pop2    :  /* 88 */
        any1 = ipop();any2 = ipop();
        break;

case dup     :  /* 89 */
        any = ipop();
        ipush(any);ipush(any);
        break;

case dup2    :  /* 92 */
        any1 = ipop(); any2 = ipop();
        ipush(any2);ipush(any1);
        ipush(any2);ipush(any1);
        break;

/* dup_x1,dup2_x1,dup_x2,dup2_x2,swap */
```

Opérations arithmétiques(1)

- case **iadd** : /* 96 */
- ivalue2 = ipop();
- ivalue1 = ipop();
- ipush(ivalue1 + ivalue2);
- break;
- /* **ladd,fadd,dadd** */

- case **isub** : /* 100 */
- ivalue2 = ipop();
- ivalue1 = ipop();
- ipush(ivalue1 - ivalue2);
- break;
- /* lsub,fsub,dsub */
-
- case **imul** : /* 104 */
- ivalue2 = ipop();
- ivalue1 = ipop();
- ipush(ivalue1 * ivalue2);
- break;
- /* **lmul,fmul,dmul** */

Opérations arithmétiques(2)

```
case idiv :  /* 108 */
        ivalue2 = ipop();
        ivalue1 = ipop();
        if (ivalue2 == 0L)
          GenerateException(ArithmeticException);
        ipush( ivalue1 / ivalue2);
        break;

/* ldiv,fdiv,ddiv,irem,frem,lrem,drem */

case ineg :  /* 116 */
        ivalue = ipop();
        ipush(-(ivalue));
        break;

/* lneg,fneg,dneg  */
```

Opérations logiques

```
case ishl :  /* 120 */
        ivalue2 = ipop() & 0x001F; ivalue1 = ipop();
        ipush(ivalue1 << ivalue2);
        break;

case ishr :  /* 122 */
        ivalue2 = ipop() & 0x001F; ivalue1 = ipop();
        if (ivalue1 < 0)
          ipush(0x8000 | (ivalue1 >> ivalue2));
        else
          ipush(ivalue1 >> ivalue2);
        break;
/* iushr,lshl,lshr,lushr */

case iand :  /* 126 */
        ivalue2 = ipop(); ivalue1 = ipop();
        ipush(ivalue1 & ivalue2);
        break;
/* land,ior,lor,ixor,lxor */
```

Opérations de conversions

- /* **i2l** */
-
- case **i2f** : /* 133 */
- ivalue = ipop();
- fpush((float_t)ivalue);
- break;
-
- case **i2d** : /* 134 */
- ivalue = ipop();
- dpush((double_t)ivalue);
- break;

- /* **l2i,l2f,l2d,f2i,f2l,f2d,d2i,d2l,d2f,**
- **int2byte,int2char,int2short** */

Instructions de contrôles(1)

```
case ifeq      :  /* 153 */
        ivalue = ipop();
        branchbyte1 = next();branchbyte2 = next();
        if (ivalue == 0L){
          offset = ((int)branchbyte1 << 8) | branchbyte2;
          pc = pc_i + offset;
        }
        break;

case iflt      :  /* 155 */
        ivalue = ipop();
        branchbyte1 = next();branchbyte2 = next();
        if (ivalue < 0L){
          offset = ((int)branchbyte1 << 8) | branchbyte2;
          pc = pc_i + offset;
        }
        break;

/* ifle,ifne,ifgt,ifge */
/* note : pc_i a la valeur du pc à la lecture de
         l'instruction courante (ici l'instruction de saut)*/
```

Instructions de contrôles(2)

```
case if_icmpeq :  /* 159 */
    ivalue2 = ipop(); ivalue1 = ipop();
    branchbyte1 = next();branchbyte2 = next();
    if (ivalue1 == ivalue2){
      offset = ((int)branchbyte1 << 8) | branchbyte2;
      pc = pc_i + offset;
    }
    break;

/* if_icmpne,if_icmplt,if_icmpgt,if_icmple,if_icmpge,
lcmp */
```

Instructions de contrôles(3)

```
case fcmpl     :  /* 149 */
        fvalue2 = fpop(); fvalue1 = fpop();
        if (fNaN(fvalue1) || fNaN(fvalue2)) ipush(-1L);
        else if (fvalue1 > fvalue2) ipush(1L);
        else if (fvalue1 == fvalue2) ipush(0L);
        else ipush(-1L);
        break;

/* fcmpg */

case dcmpl     :  /* 151 */
        dvalue2 = dpop(); dvalue1 = dpop();
        if (dvalue1 < dvalue2) ipush(1L);
        else if (dvalue1 == dvalue2) ipush(0L);
        else ipush(-1L);
        break;

/* dcmpg,if_acmpeq,if_acmpne,ifnull,ifnonnull */
```

Instructions de contrôles(4)

```
case gotoo    :  /* 167 */
    branchbyte1 = next();branchbyte2 = next();
    offset = ((int)branchbyte1 << 8) | branchbyte2;
    pc = pc_i + offset;
    break;

case goto_w   :  /* 200  */
    loffset = next4();
    pc = pc_i + loffset;
    break;

/* jsr_w,jsr */
```

Exemple : le source en JAVA

```
class jvm_1 {
  public static void main( String args[]) {

    int i, j;

    i = 1;
    j = 2;
    i = j;
    if (i != j) i = i*i;
    while (j < 10){
      j++;
    }

  }
}
```

Exemple : les instructions

- iconst_1, /* i = 1; */
- istore_1,
- iconst_2, /* i = 2; */
- istore_2,
- iload_2, /* i = j; */
- istore_1,
- iload_1, /* if (i != j) */
- iload_2,
- if_icmpeq, 0x00, 0x0D,
- iload_1, /* i = i*i; */
- iload_1,
- imul,
- istore_1,
- gotoo, 0x00, 0x06, /* while .. */
- iinc, 2, 1, /* j++; */
- iload_2,
- bipush, 10,
- if_icmplt, 0xFF, 0xFA, /* while (j < 10) */
- returnn

Un autre exemple, le source Java

```
class jvm_2 {
  public static void main( String args[]) {
    int i, j;
    double d, d1;
    int k,l;

    i = j = 3;
    d = d1 = 2.0;
    k = l = 3;
    if (j == k)
      d = d * (double) k;

    k = k +5;
    for( i=1;i<k;i++){
      d = d * d1;
    }
  }}
```

Le code

- iconst_3,
- dup,
- istore_2,
- istore_1,
- ldc2w, 0x00, 0x05,
- /* #5 <Double 2.000000> */
- dup2,
- dstore, 5,
- dstore_3,
- iconst_3,
- dup,
- istore, 8,
- istore, 7,
- iload_2,
- iload, 7,
- if_icmpne, 0x00, 0x09,
- dload_3,
- iload, 7,
- i2d,
- dmul,
- dstore_3,
- iload, 7,
- iconst_5,
- iadd,
- istore, 7,
- iconst_1,
- istore_1,
- gotoo, 0x00, 0x0B,
- dload_3,
- dload, 5,
- dmul,
- dstore_3,
- iinc, 1, 1,
- iload_1,
- iload, 7,
- if_icmplt, 0xFF, 0xF5,

Exemple : le tri bulle...

```
public class bulbe{
  public static void main( String Args[]){
    int [] n = new int[6];
    n[0]=5;n[1]=3;n[2]=1;n[3]=2;n[4]=4;n[5]=6;

    boolean sorted = false;
    while(!sorted){
      sorted = true;
      for(int i = 0; i < 5; i++){
        if (n[i] > n[i + 1]){
          int temp = n[i];
          n[i] = n[i + 1];
          n[i + 1] = temp;
          sorted = false;
        }
      }
    }
    boolean trie = true;
    if (n[0]!=0) trie = false;
    if (n[1]!=1) trie = false;
    if (!(n[2]==2  || n[3]==3 || n[4]==4 || n[5]==5)) trie = false;
  }}
```

Exemple : les instructions(1)

- /*0*/ bipush, 6,
- /*2*/ newarray, 10,
- /*4*/ astore_1,
- /*5*/ aload_1,
- /*6*/ iconst_0,
- /*7*/ iconst_5,
- /*8*/ iastore,
- /*9*/ aload_1,
- /*10*/ iconst_1,
- /*11*/ iconst_3,
- /*12*/ iastore,
- /*13*/ aload_1,
- /*14*/ iconst_2,
- /*15*/ iconst_1,
- /*16*/ iastore,
- /*17*/ aload_1,
- /*18*/ iconst_3,
- /*19*/ iconst_2,
- /*20*/ iastore,
- /*21*/ aload_1,
- /*22*/ iconst_4,
- /*23*/ iconst_4,
- /*24*/ iastore,
- /*25*/ aload_1,
- /*26*/ iconst_5,
- /*27*/ bipush, 6,
- /*29*/ iastore,
- /*30*/ iconst_0,
- /*31*/ istore_2,
- /*32*/ gotoo,0,83,
- /*35*/ iconst_1,
- /*36*/ istore_2,
- /*37*/ iconst_0,
- /*38*/ istore_3,
- /*39*/ gotoo,0,78,
- /*42*/ aload_1,
- /*43*/ iload_3,
- /*44*/ iaload,
- /*45*/ aload_1,
- /*46*/ iload_3,
- /*47*/ iconst_1,
- /*48*/ iadd,
- /*49*/ iaload,
- /*50*/if_icmple,0,75

Exemple : Les instructions(2)

- /*53*/ aload_1,
- /*54*/ iload_3,
- /*55*/ iaload,
- /*56*/ istore, 4,
- /*58*/ aload_1,
- /*59*/ iload_3,
- /*60*/ aload_1,
- /*61*/ iload_3,
- /*62*/ iconst_1,
- /*63*/ iadd,
- /*64*/ iaload,
- /*65*/ iastore,
- /*66*/ aload_1,
- /*67*/ iload_3,
- /*68*/ iconst_1,
- /*69*/ iadd,
- /*70*/ iload, 4,
- /*72*/ iastore,
- /*73*/ iconst_0,
- /*74*/ istore_2,
- /*75*/ iinc, 3, 1,
- /*78*/ iload_3,
- /*79*/ iconst_5,
- /*80*/if_icmplt,0xFF,42-80,
- /*83*/ iload_2,
- /*84*/ ifeq,0xFF,35-84,
- /*87*/ iconst_1,
- /*88*/ istore_3,
- /*89*/ aload_1,
- /*90*/ iconst_0,
- /*91*/ iaload,
- /*92*/ ifeq,0,97-92,
- /*95*/ iconst_0,
- /*96*/ istore_3,
- /*97*/ aload_1,
- /*98*/ iconst_1,
- /*99*/ iaload,
- /*100*/ iconst_1,
- /*101*/ if_icmpeq,0,5,
- /*104*/ iconst_0,
- /*105*/ istore_3,
- /*106*/ aload_1,
- /*107*/ iconst_2,
- /*108*/ iaload,
- /*109*/ iconst_2,
- /*110*/if_icmpeq,0,26,
- /*113*/ aload_1,
- /*114*/ iconst_3,
- /*115*/ iaload,
- /*116*/ iconst_3,
- /*117*/ if_icmpeq,0,136
- /*120*/ aload_1,
- /*121*/ iconst_4,
- /*122*/ iaload,
- /*123*/ iconst_4,
- /*124*/ if_icmpeq,0,136
- /*127*/ aload_1,
- /*128*/ iconst_5,
- /*129*/ iaload,
- /*130*/ iconst_5,
- /*131*/ if_icmpeq,0,136
- /*134*/ iconst_0,
- /*135*/ istore_3,
- /*136*/ returnn,

Accès direct, ("switch/case")

```
        case tableswitch :  /* 170 */
            index = ipop();
            while (((long)pc % 4) != 0) pc++;
            default_offset = next4();
            low  = next4();
            high = next4();
            if (index < low || index > high)
              pc = pc_i + default_offset;
            else{
              index = index -low;
              pc = pc + (index*4);
              offset4 = next4();
              pc = pc_i + offset4;
            }
            break;

    /* lookupswitch:   171 */
```

Exemple de switch

```
class jvm_3 {

  public static void main ( String args[]) {
    int i=0,j=0;

    i = 3;
    switch(i){
      case 1 : j = 1;
      case 2 : j = 2;
      case 3 : j = 3;
      case 4 : j = 4;
      default : j = 5;
    }
  ....
```

Exemple tableswitch

- /*0*/ iconst_0,
- /*1*/ istore_1,
- /*2*/ iconst_0,
- /*3*/ istore_2,
- /*4*/ iconst_3,
- /*5*/ istore_1,
- /*6*/ iload_1,
- /*7*/ tableswitch,
- 0x00, 0x00, 0x00,0x25,/* 1 to 4: 44 */
- 0x00, 0x00, 0x00,0x01,/* 1: 36 */
- 0x00, 0x00, 0x00,0x04,
- 0x00, 0x00, 0x00,0x1D,
- 0x00, 0x00, 0x00,0x1F,/* 2: 38 */
- 0x00, 0x00, 0x00,0x21,/* 3: 40 */
- 0x00, 0x00, 0x00,0x23,/* 4: 42 */

- /*36*/ iconst_1,
- /*37*/ istore_2,
- /*38*/ iconst_2,
- /*39*/ istore_2,
- /*40*/ iconst_3,
- /*41*/ istore_2,
- /*42*/ iconst_4,
- /*43*/ istore_2,
- /*44*/ iconst_5,
- /*45*/ istore_2,
- ...

Accès direct, ("switch/case")

```
case lookupswitch:  /* 171 */
    key = ipop();
    while (((long)pc % 4) != 0) pc++;
    default_offset = next4();
    npairs = next4();
    index = matched = 0;
    do{

      match  = next4();
      loffset = next4();
      matched = key == match;
      index++;
    }while( (index <= npairs) && (key > match) && !matched);
    if (matched)
      pc = pc_i + loffset;
    else
      pc = pc_i + default_offset;
    break;
```

Exemple de switch

```
class jvm_3 {

  public static void main ( String args[]) {
    int i=0,j=0;

   switch(i){
     case 20 : j = 1;
     case 2 : j = 2;
     case 100 : j = 3;
     case 4 : j = 4;
     default : j = 5;
    }
  ....
```

Exemple lookupswitch

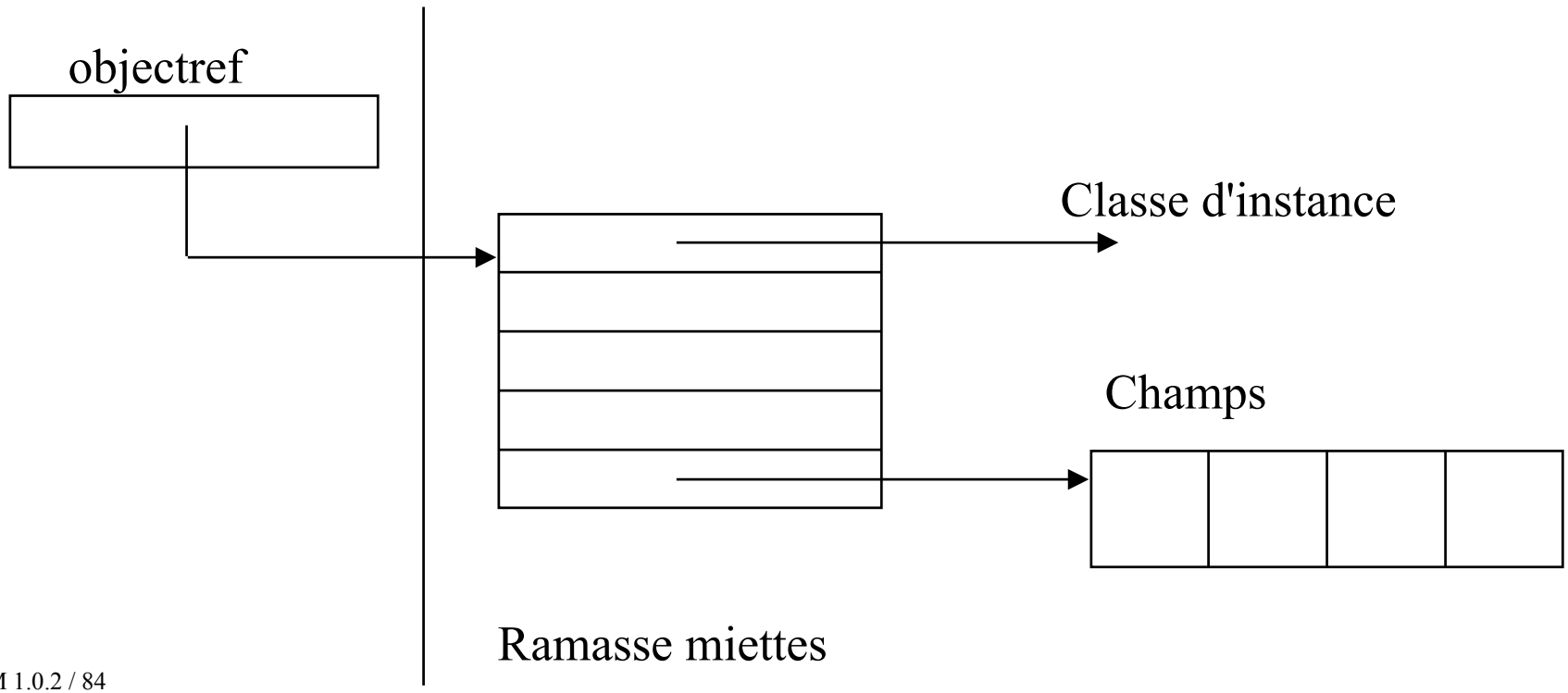
- /* 46*/ iload_1,
- /* 47*/ lookupswitch,
- 0, 0, 0,49,/* default : 96*/
- 0, 0, 0, 4,/* npairs : 4*/
- 0, 0, 0, 2, 0, 0, 0,43,/*<2-90>*/
- 0, 0, 0, 4, 0, 0, 0,47,/*<4-94>*/
- 0, 0, 0,20, 0, 0, 0,41,/*<20-88>*/
- 0, 0, 0,100, 0, 0, 0,45,/*<100-92>*/
- /* 88*/ iconst_1,
- /* 89*/ istore_2,
- /* 90*/ iconst_2,
- /* 91*/ istore_2,
- /* 92*/ iconst_3,
- /* 93*/ istore_2,
- /* 94*/ iconst_4,
- /* 95*/ istore_2,
- /* 96*/ iconst_5,
- /* 97*/ istore_2,

Gestion des variables de classe

```
case putstatic:  /* 179 */ /* et getstatic */
        indexbyte1 = next(); indexbyte2 = next();
        index = ((unsigned int)indexbyte1<<8 | indexbyte2);

       if(!FieldFound(cc,index,&ClassNum,&FieldNum)){
         if((int)ClassNum == VOIDNUMBER){
           GenerateException(Env,ClassNotFoundException);
         }else if(FieldNum == VOIDNUMBER){
           GenerateException(Env,NoSuchFieldError);
         }
        }else{

        pop_static_fields(ClassNum,FieldNum);

      /* JIT */
        *pc_i = putstatic_quick; pc_i++;
        *pc_i = (u1) ClassNum; pc_i++;
        *pc_i = (u1) FieldNum;
        }

        break;
```

Gestion des variables d'instances

- case **neww** : /* 187 */
- indexbyte1 = next(); indexbyte2 = next();
- index = ((unsigned int)indexbyte1<<8 | indexbyte2);
- objectref = AllocObject(cc,index);
- apush(objectref);
- break;



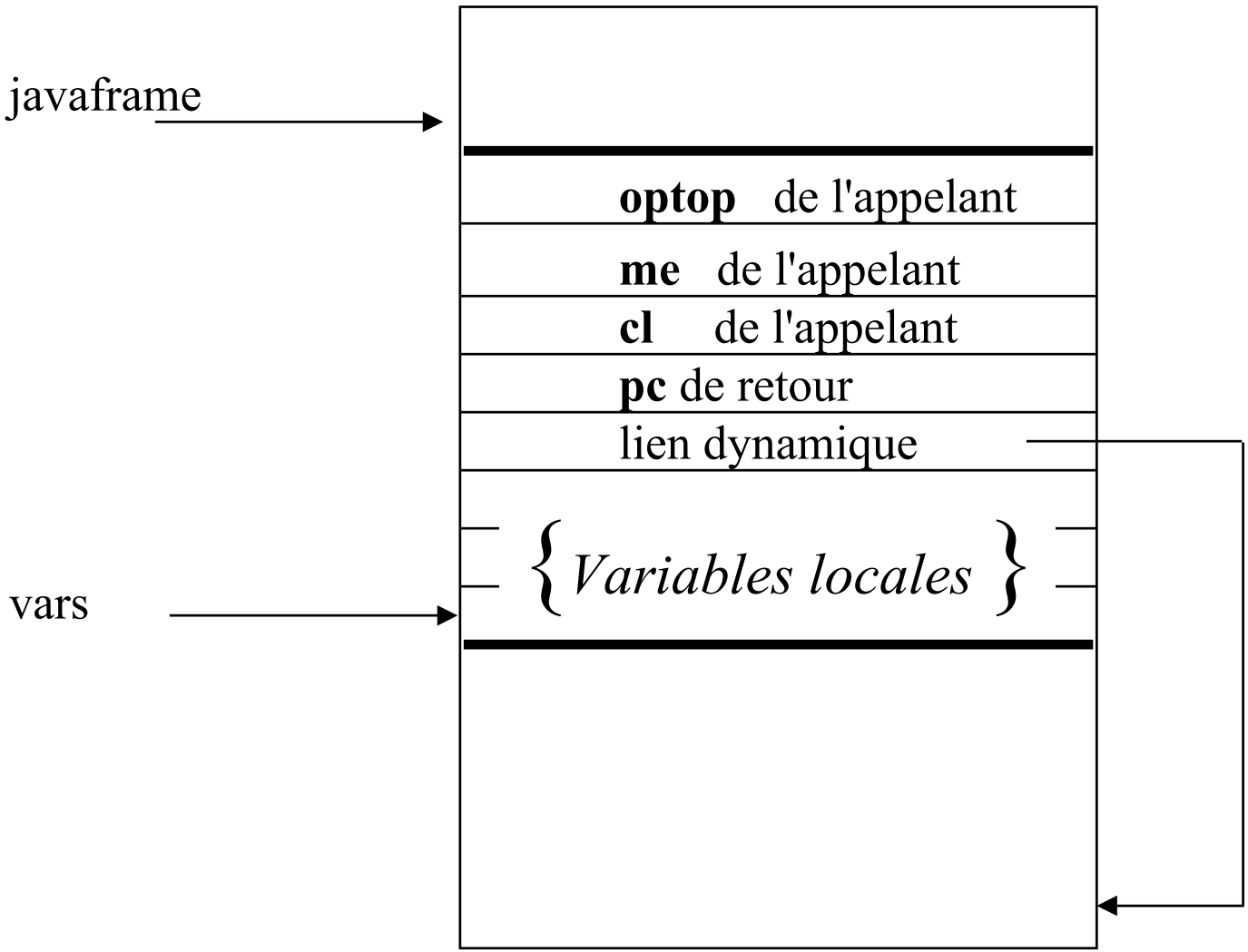
Gestion des variables d'instances

```
case putfield :  /* 181 */ /* case getfield : 180 */
    indexbyte1 = next(); indexbyte2 = next();
    index = ((unsigned int)indexbyte1<<8 | indexbyte2);

    if(!FieldFound(cc,index,&ClassNum,&FieldNum)){
      if((int)ClassNum == VOIDNUMBER){
        GenerateException(Env,ClassNotFoundException);
      }else if(FieldNum == VOIDNUMBER){
        GenerateException(Env,NoSuchFieldError);
      }
    }else{
    objectref = apop();
    if (objectref == null)
      GenerateException(Env,NullPointerException);

    push_local_fields(ClassNum,objectref,FieldNum);


    /* JIT */*pc_i = getfield_quick; pc_i++;
    /* JIT */*pc_i = (u1) ClassNum; pc_i++;
    /* JIT */*pc_i = (u1) FieldNum;
    };break;
```

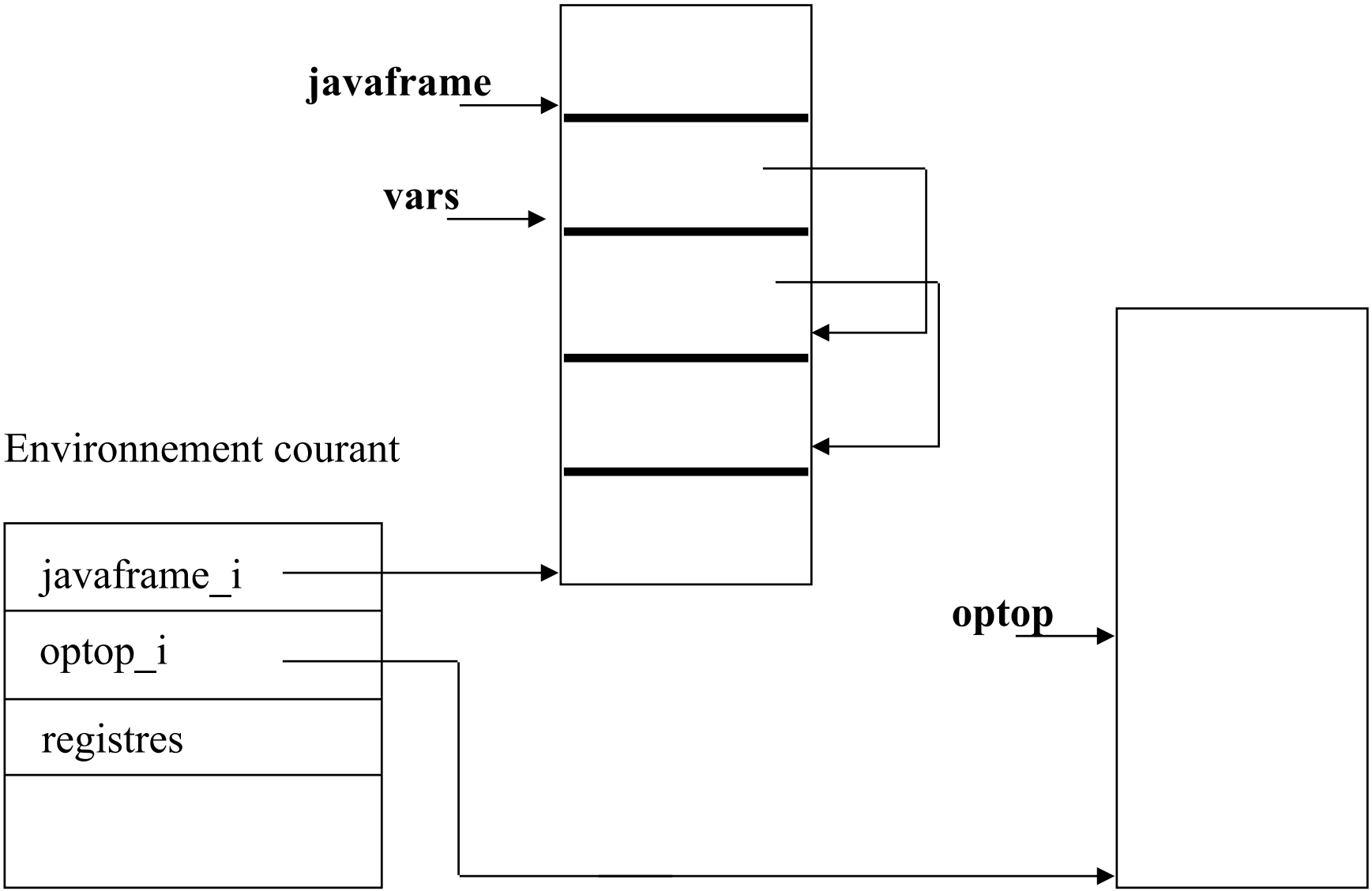
Invocation de méthodes de classe

```
case invokestatic     :  /* 184 */
        indexbyte1 = next(); indexbyte2 = next();
        index = ((unsigned int)indexbyte1<<8 | indexbyte2);

        if(!ClassFound(cc,index,&ClassNum,&MethodNum)){
          if((int)ClassNum==VOIDNUMBER){
            GenerateException(Env,ClassNotFoundException);
          }else if((int)MethodNum==VOIDNUMBER){
            GenerateException(Env,NoSuchMethodError);
          }
        }else{

        /* JIT */
        *pc_i = invokestatic_quick; pc_i++;
        *pc_i = (u1) ClassNum; pc_i++;
        *pc_i = (u1) MethodNum;

        mark_static( Env, ClassNum, MethodNum);

        break;
              /* case invokespecial : case invokeinterface */
```

Segment d'activation



Environnement d'exécution



Segment d'activation, mark, unmark

```
void mark_static( environment *Env, unsigned int ClassNum,
                  unsigned int MethodNum){
 stack_item *oldjavaframe=javaframe;
 stack_item *optop_initial;
 unsigned int ArgNum;

 ArgNum = cc->methods[me].attributes[0].max_locals;
 FrameStackOverflowTest(ArgNum+5);javaframe = javaframe + ArgNum;
 javaframe->s    = oldjavaframe; javaframe++;   /* lien dynamique */
 javaframe->addr = pc;   javaframe++;/* adresse du code de retour */
 javaframe->i    = cl;   javaframe++;       /* la classe courante */
 javaframe->i    = me;   javaframe++;      /* la place pour optop */
 vars  = javaframe;

 pop_local_variables(ClassNum,MethodNum);
 optop_initial->s    = optop; /* le sommet de pile apres recopie LY p 330 */

 cl = ClassNum; /* affectation des registres */
 me = MethodNum;
 cc = class_table(ClassNum);
 pc = cc->methods[MethodNum].attributes[0].code; }
```

Valeurs retournées et retour de S/P

```
        case ireturn :  /* 172 */

            unmark( Env);
            break;

    /* lreturn,freturn,dreturn */

        case ret     :  /* 169 */
            uindex = next();
            pc = (u1 *) aGetLocalVariable( uindex);
            break;


    /* areturn, returnn */
```

Segment d'activation, unmark

```
static void unmark( environment *Env){ /* optop reste en l'etat */

  javaframe--;  /* optop reste en l'etat (retour de fonction) */
  javaframe--; me = javaframe->i;
  javaframe--; cl = javaframe->i;
  javaframe--; pc = javaframe->addr;
  javaframe--; javaframe = vars = javaframe->s;
  cc = class_table(cl);
 }
```

Invocations de méthodes

```
case invokevirtual    :  /* 182 */

    indexbyte1 = next(); indexbyte2 = next();
    index = ((unsigned int)indexbyte1<<8 | indexbyte2);

    if(!ClassFound(cc,index,&ClassNum,&MethodNum)){
      if((int)ClassNum==VOIDNUMBER){
        GenerateException(Env,ClassNotFoundException);
      }else if((int)MethodNum==VOIDNUMBER){
        GenerateException(Env,NoSuchMethodError);
      }
    }else{

    mark_virtual( Env, ClassNum, MethodNum, index);
   /* JIT */ /* pas pour l'instant */
         /* des l'implantation de la tmv */

    };break;
```

Invocation de méthodes d'instance

```
static void mark_virtual( environment  *Env,unsigned int ClassNum,
                          unsigned int MethodNum, unsigned int index){

  sauvegarde des registres

  pop_local_variables(ClassNum,MethodNum);


  {    /* recherche de la methode virtuelle */
  objectref_t thiss;   /* thiss pour this */
  thiss = aGetLocalVariable(0);  /* par convention voir LY */
  search_virtual_method(thiss,ClassNumInitial,index,
                        &ClassNum,&(int)MethodNum);

  if((int)MethodNum == VOIDNUMBER){
    GenerateException(Env,NoSuchMethodError);
  }
  }
affectation des registres
}
```

Exemple : la classe Complexe

```
class Complexe{

  Complexe(){
    Reel = 0.0;
    Imag = 0.0;
  }

  Complexe(double PartieReelle, double PartieImaginaire){
    Reel = PartieReelle;
    Imag = PartieImaginaire;
  }


  public Complexe Plus( Complexe C){
    return new Complexe(Reel + C.Reel, Imag + C.Imag);
  }

  .....
    private double Reel;
    private double Imag;
  }
```

La classe Complexe : le "bytecode"

- **/* Complexe.<init>.()V */** *Complexe()*

- /* 0*/ aload_0,
- /* 1*/ invokespecial, 0, 9,/* java/lang/Object.<init>()V */
- /* 4*/ aload_0,
- /* 5*/ dconst_0,
- /* 6*/ putfield, 0,18,/* Complexe.Reel D */
- /* 9*/ aload_0,
- /* 10*/ dconst_0,
- /* 11*/ putfield, 0,14,/* Complexe.Imag D */
- /* 14*/ returnn,

- /* ici appel du constructeur de la classe object sur cette
- instance ligne 1 */

La classe Complexe : le "bytecode"

- /* Complexe.<init>.(DD)V */

  Complexe(double PartieReelle, double PartieImaginaire)

- /* 0*/ aload_0,
- /* 1*/ invokespecial, 0, 9,/* java/lang/Object.<init>()V */
- /* 4*/ aload_0,
- /* 5*/ dload_1,
- /* 6*/ putfield, 0,18,/* Complexe.Reel D */
- /* 9*/ aload_0,
- /* 10*/ dload_3,
- /* 11*/ putfield, 0,14,/* Complexe.Imag D */
- /* 14*/ returnn,

La classe Complexe : le "bytecode"

- **/* Plus.(LComplexe;)LComplexe; */**

- /* 0*/ new, 0, 5,/* Complexe */
- /* 3*/ dup,
- /* 4*/ aload_0,
- /* 5*/ getfield, 0,18,/* Complexe.Reel D */
- /* 8*/ aload_1,
- /* 9*/ getfield, 0,18,/* Complexe.Reel D */
- /* 12*/ dadd,
- /* 13*/ aload_0,
- /* 14*/ getfield, 0,14,/* Complexe.Imag D */
- /* 17*/ aload_1,
- /* 18*/ getfield, 0,14,/* Complexe.Imag D */
- /* 21*/ dadd,
- /* 22*/ invokespecial, 0,11,/* Complexe.<init>(DD)V */
- /* 25*/ areturn,

Complexe : le constant pool(1)

- [1] tag: 8 string_index: 75
- [2] tag: 7 name_index: 68
- [3] tag: 7 name_index: 43
- [4] tag: 7 name_index: 74
- [5] tag: 7 name_index: 63
- [6] tag: 7 name_index: 32
- [7] tag: 7 name_index: 44
- [8] tag: 7 name_index: 51
- [9] tag: 10 class_index: 4 name_and_type_index: 26
- [10] tag: 10 class_index: 2 name_and_type_index: 30
- [11] tag: 10 class_index: 5 name_and_type_index: 23
- [12] tag: 10 class_index: 2 name_and_type_index: 26
- [13] tag: 10 class_index: 8 name_and_type_index: 29
- [14] tag: 9 class_index: 5 name_and_type_index: 25
- [15] tag: 10 class_index: 8 name_and_type_index: 22
- [16] tag: 9 class_index: 3 name_and_type_index: 28
- [17] tag: 10 class_index: 2 name_and_type_index: 21
- [18] tag: 9 class_index: 5 name_and_type_index: 24
- [19] tag: 10 class_index: 2 name_and_type_index: 31
- [20] tag: 10 class_index: 6 name_and_type_index: 27
- [21] tag: 12 class_index: 59 descriptor_index: 70
- [22] tag: 12 class_index: 35 descriptor_index: 48
- [23] tag: 12 class_index: 72 descriptor_index: 41
- [24] tag: 12 class_index: 38 descriptor_index: 60
- [25] tag: 12 class_index: 58 descriptor_index: 60
- [26] tag: 12 class_index: 72 descriptor_index: 50

Complexe : le constant pool(2)

- [27] tag: 12 class_index: 71 descriptor_index: 36
- [28] tag: 12 class_index: 56 descriptor_index: 64
- [29] tag: 12 class_index: 72 descriptor_index: 36
- [30] tag: 12 class_index: 49 descriptor_index: 67
- [31] tag: 12 class_index: 49 descriptor_index: 46
- [32] tag: 1 length: 19 java/io/PrintStream
- [33] tag: 1 length: 15 LineNumberTable
- [34] tag: 1 length: 13 ConstantValue
- [35] tag: 1 length: 7 valueOf
- [36] tag: 1 length: 21 (Ljava/lang/String;)V
- [37] tag: 1 length: 7 EstEgal
- [38] tag: 1 length: 4 Reel
- [39] tag: 1 length: 10 Exceptions
- [40] tag: 1 length: 17 TestComplexe.java
- [41] tag: 1 length: 5 (DD)V
- [42] tag: 1 length: 10 SourceFile
- [43] tag: 1 length: 16 java/lang/System
- [44] tag: 1 length: 12 TestComplexe
- [45] tag: 1 length: 18 LocalVariableTable
- [46] tag: 1 length: 27 (D)Ljava/lang/StringBuffer;
- [47] tag: 1 length: 6 Ecrire
- [48] tag: 1 length: 21 (D)Ljava/lang/String;
- [49] tag: 1 length: 6 append
- [50] tag: 1 length: 3 ()V
- [51] tag: 1 length: 16 java/lang/String
- [52] tag: 1 length: 4 this

Complexe : le constant pool(3)

- [52] tag: 1 length: 4 this
- [53] tag: 1 length: 16 PartieImaginaire
- [54] tag: 1 length: 10 LComplexe;
- [55] tag: 1 length: 3 ()D
- [56] tag: 1 length: 3 out
- [57] tag: 1 length: 22 (LComplexe;)LComplexe;
- [58] tag: 1 length: 4 Imag
- [59] tag: 1 length: 8 toString
- [60] tag: 1 length: 1 D
- [61] tag: 1 length: 1 C
- [62] tag: 1 length: 12 PartieReelle
- [63] tag: 1 length: 8 Complexe
- [64] tag: 1 length: 21 Ljava/io/PrintStream;
- [65] tag: 1 length: 4 Code
- [66] tag: 1 length: 14 LocalVariables
- [67] tag: 1 length: 44 (Ljava/lang/String;)Ljava/lang/StringBuffer;
- [68] tag: 1 length: 22 java/lang/StringBuffer
- [69] tag: 1 length: 8 ToString
- [70] tag: 1 length: 20 ()Ljava/lang/String;
- [71] tag: 1 length: 5 print
- [72] tag: 1 length: 6 <init>
- [73] tag: 1 length: 13 (LComplexe;)Z
- [74] tag: 1 length: 16 java/lang/Object
- [75] tag: 1 length: 4 + i
- [76] tag: 1 length: 4 Plus

Le module de test

```
public class TestComplexe{

  public static void main( String Args[]){

    Complexe C0 = new Complexe();
    Complexe C1, C2, C3;

    System.out.println(" test de la classe Complexe");
    C1 = new Complexe(1.0,2.0);
    C2 = new Complexe(3.0,4.0);

    System.out.print("C1 = ");C1.Ecrire(); System.out.println(".");
    System.out.print("C2 = ");C2.Ecrire(); System.out.println(".");

    C1 = C2;

    if (C1.EstEgal(C2)){
      System.out.println("       C1 est egal a C2");
    } else {
      System.out.println(" C1 est different de C2");
    }
```

La classe TestComplexe : le "bytecode"

- **/* main.([Ljava/lang/String;)V */**

- /* 0*/ new, 0,12,/* Complexe */
- /* 3*/ dup,
- /* 4*/ invokespecial, 0,17,/* Complexe.<init>()V */
- /* 7*/ astore_1,
- /* 8*/ getstatic, 0,21,/* java/lang/System.out Ljava/io/PrintStream; */
- /* 11*/ ldc, 7,/* string: test de la classe Complexe */
- /* 13*/ invokevirtual, 0,19,/* java/io/PrintStream.println(Ljava/lang/String;)V */
- /* 16*/ new, 0,12,/* Complexe */
- /* 19*/ dup,
- /* 20*/ dconst_1,
- /* 21*/ ldc2_w, 0,30,/* double:2.000000 */
- /* 24*/ invokespecial, 0,16,/* Complexe.<init>(DD)V */
- /* 27*/ astore_2,
- /* 28*/ new, 0,12,/* Complexe */
- /* 31*/ dup,
- /* 32*/ ldc2_w, 0,35,/* double:3.000000 */
- /* 35*/ ldc2_w, 0,25,/* double:4.000000 */
- /* 38*/ invokespecial, 0,16,/* Complexe.<init>(DD)V */
- /* 41*/ astore_3,

La classe TestComplexe : le "bytecode"

- /* 42*/ getstatic, 0,21,/* java/lang/System.out Ljava/io/PrintStream; */
- /* 45*/ ldc, 4,/* string: C1 = */
- /* 47*/ invokevirtual, 0,24,/* java/io/PrintStream.print(Ljava/lang/String;)V */
- /* 50*/ aload_2,
- /* 51*/ invokevirtual, 0,18,/* Complexe.Ecrire()V */
- /* 54*/ getstatic, 0,21,/* java/lang/System.out Ljava/io/PrintStream; */
- /* 57*/ ldc, 1,/* string: . */
- /* 59*/ invokevirtual, 0,19,/* java/io/PrintStream.println(Ljava/lang/String;)V */
- /* 62*/ getstatic, 0,21,/* java/lang/System.out Ljava/io/PrintStream; */
- /* 65*/ ldc, 6,/* string: C2 = */
- /* 67*/ invokevirtual, 0,24,/* java/io/PrintStream.print(Ljava/lang/String;)V */
- /* 70*/ aload_3,
- /* 71*/ invokevirtual, 0,18,/* Complexe.Ecrire()V */
- /* 74*/ getstatic, 0,21,/* java/lang/System.out Ljava/io/PrintStream; */
- /* 77*/ ldc, 1,/* string: . */
- /* 79*/ invokevirtual, 0,19,/* java/io/PrintStream.println(Ljava/lang/String;)V */
- /* 82*/ aload_3,
- /* 83*/ astore_2,

Exceptions déclenchées par la machine

- bloc try/catch
- champ exceptions du .class, par méthode

```
typedef struct{
        u2 start_pc;
        u2 end_pc;
        u2 handler_pc;
        u2 catch_type;
      } exception_info;

typedef struct{.....
        u4 code_length;
        u1 *code;
        u2 exception_table_length;
        exception_info *exception_table;
        u2 attributes_count;
        attribute_info *attributes;
      } Code_attribute;
```

Exemple : except.java

```
class except {

  public static  void Write(int i){ }

  public static void main( String args[]) {
    int i=0,j=0;

    try{
      i++;
      i = i /j;

    }catch( ArithmeticException e){
       Write(1);
    }
    Write(2);
    }
}
```

Exemple : except.class

- /* **except.main.([Ljava/lang/String;)V** */
- /* 0*/ iconst_0,
- /* 1*/ istore_1,
- /* 2*/ iconst_0,
- /* 3*/ istore_2,
- /* 4*/ iinc, 1, 1,/*(0x1),(0x1)*/
- /* 7*/ iload_1,
- /* 8*/ iload_2,
- /* 9*/ idiv,
- /* 10*/ istore_1,
- /* 11*/ gotoo, 0, 8,/*(8 -> 19)*/
- /* 14*/ pop,
- /* 15*/ iconst_1,
- /* 16*/ invokestatic, 0, 4,/* except.Write(I)V */
- /* 19*/ iconst_2,
- /* 20*/ invokestatic, 0, 4,/* except.Write(I)V */
- /* 23*/ returnn,
- /* exception : start_pc, end_pc, handler_pc, catch_type */
- /* _table[0]   4 ,   11 ,   14 ,   1   */

Exceptions programmées

- instruction athrow

finally : Finally.java

```
class Finally {
  public static  void Write(int i){ }


  public static void main( String args[]) {
    int i=0,j=0;
    try{
      i++;
      i = i /j;

    }catch( ArithmeticException e){
       Write(12121212);
    } finally {
      Write(23232323);
    }

      Write(34343434);
    }
}
```

finally : Finally.class

- /* **except.main.([Ljava/lang/String;)V** */
- /* 0*/ iconst_0, ... /* 11*/ gotoo, 0,12,/*(12 -> 23)*/
- /* 14*/ pop,
- /* 15*/ ldc, 3,/* int:12121212 *//* int:(0xB8F47C) */
- /* 17*/ invokestatic, 0, 8,/* Finally.Write(I)V */
- /* 20*/ gotoo, 0, 3,/*(3 -> 23)*/
- /* 23*/ **jsr, 0,12,/*(0x0),(0xC)*/**
- /* 26*/ gotoo, 0,18,/*(18 -> 44)*/
- /* 29*/ astore_3,
- /* 30*/ **jsr, 0, 5,/*(0x0),(0x5)*/**
- /* 33*/ aload_3,
- /* 34*/ athrow,
- /* 35*/ astore, 4,
- /* 37*/ ldc, 2,/* int:23232323 *//* int:(0x1627F43) */
- /* 39*/ invokestatic, 0, 8,/* Finally.Write(I)V */
- /* 42*/ **ret, 4,**
- /* 44*/ ldc, 1,/* int:34343434 *//* int:(0x20C0A0A) */
- /* 46*/ invokestatic, 0, 8,/* Finally.Write(I)V */
- /* 49*/ returnn,

Le code en mémoire

- Structures de données
  - Une table de classes (limitée à 256/JIT)
  - Chaque classe possède une liste de méthodes (limitée à 256)

    ==> Chaque méthode est identifiée par un numéro de classe et un numéro de méthode.

Proposition de J.I.T.

invokestatic, indexbyte1, indexbyte2,

- indexbyte1 et indexbyte2 accès au constant_pool

invokestatic_quick, NumerodeClasse, NumerodeMéthode

- accès direct au code associé

Démarrage de la machine

- ClassNum = add_class(argv[1]); /* chargement de la classe argv[1] */
- createInterpreter(&Main); /* creation d'un interpreteur Java */
- /* Appel du constructeur par defaut de cette classe */
- ConsNum = search_class_method(ClassNum,&Cl,"<init>","()V");
- initializeInterpreter(&Main,ClassNum,ConsNum);
- objectref = execute_constructor(&Main, ClassNum, ConsNum);
- MethodNum = search_class_method(ClassNum,&Cl,"main","([Ljava/lang/String;)V");
- initializeInterpreter(&Main,ClassNum,MethodNum);
- execute(&Main,ClassNum, MethodNum, AllocArgs(argv,argn));
- destroyInterpreter(&Main);

interpreter.h

- void createInterpreter(interpreter *);
- void destroyInterpreter(interpreter *);

- void initializeInterpreter(interpreter *Inter,
- unsigned int ClassNum,
- unsigned int ConsNum);

- void execute (interpreter *Inter,
- unsigned int ClassNum,
- unsigned int MethodNum,
- objectref_t Args);

- void execute_class_init(interpreter *Inter,
- unsigned int ClassNum,
- unsigned int ConsNum);

- objectref_t execute_constructor (interpreter *Inter,
- unsigned int ClassNum,
- unsigned int ConsNum);

Gestion des instances

- Table pour chaque champ < taille, déplacement>
- Identification de la classe d'instance
- Ramasse miettes

Gestion des Threads

- Prise en compte par le systéme hôte
- reste à implanter
  - Les requêtes de synchronisation (synchronized) par l'implantation des moniteurs(au sens hoare)

- Un objet peut devenir un moniteur
  - synchronized(new Object()){
  - wait();
  - }

Premières conclusions(1)

- Implantation facilitée par : (+)
  - héritage simple
  - 2 types de mots machine :
    - 32 bits Objet et types prédéfini
    - 64 bits pour double et long (dommage ....)
  - Pas de référence en Java
    - pas d'adressage programmée
    - Passage de paramètre par référence n'existe pas
- ...

Premières conclusions(2)

- Recopie des paramètres
  - assurée par la machine ?
- Grand nombre de classe (contre)
  - Gestion de cache traditionnelle peu adaptée
- Temps-réél (contre)
  - voir Kelvin Nilsen et JavaTime (Berkeley)
- ...

Statistiques et évolution

- Premières constatations
  - prépondérance de :
    - aload_0 chargement de **this** sur la pile
    - getfield accès aux champs de l'objet en cours (getfield_quick)
    - putfield " " "
    - invokevirtual
- …

Annexe 0.1

- CHRT222.java, un automate d'états déclenché pendant 2*10 secondes
  - 6 états, (dont 2 automates de 2 états en //)
- 2 styles de génération de code en Java de l'automate
  - 1) à l'aide de *if else...*
  - 2) à l'aide d'un *switch-case*
  - *trace d'exécution sur PowerPC :*

on declenche l'automate pendant env. 10 sec
if_style : CHRT22if
*L'automate est declenche : **28109** fois en if_style*
on declenche l'automate pendant env. 10 sec
case_style : CHRT22sw
*L'automate est declenche : **29663** fois en case_style*

Annexe 0.1 : occurrence des instructions

- iconst_0; 3(0x3);460648
- iconst_1; 4(0x4);576952
- iconst_2; 5(0x5);115548
- iconst_3; 6(0x6);231106
- bipush; 16(0x10); 226446
- ldc; 18(0x12); 8
- ldc2_w; 20(0x14); 57772
- iload; 21(0x15); 57774
- dload; 24(0x18);115544
- iload_1; 27(0x1b); 57772
- iload_2; 28(0x1c); 57772
- iload_3; 29(0x1d); 57772
- **aload_0;** 42(0x2a);1759797
- aload_1; 43(0x2b); 28109
- aload_2; 44(0x2c); 29663
- iaload; 46(0x2e);662047
- istore; 54(0x36); 2
- dstore; 57(0x39); 57774
- istore_3; 62(0x3e); 57772
- iastore ; 79(0x4f);173340
- ...

•iflt;155(0x9b); 57772
•ifle ;158(0x9e); 57772
•if_icmpne ;160(0xa0);255311
•if_icmplt ;161(0xa1);231096
•gotoo ;167(0xa7); 87439
•tableswitch ;170(0xaa); 29663
•lookupswitch;171(0xab);117098
•ireturn ;172(0xac); 57772
•returnn ;177(0xb1); 8
•getstatic ;178(0xb2); 10
•putstatic ;179(0xb3); 3
•getfield ;180(0xb4); 69
•putfield ;181(0xb5); 34
•invokevirtual;182(0xb6); 57782
•invokespecial ;183(0xb7); 9
•invokestatic ;184(0xb8); 57774
•neww ;187(0xbb); 5
•newarray ;188(0xbc); 8
•**getfield_quick** ;206(0xce);1644152
•putfield_quick ;207(0xcf);288858
•invokesuper_quick ;216(0xd8); 2

aload_0 et getfield_quick représentent 20 % chacune des occurrences

Annexe 0.1 : occurrence des classes

- java/lang/Object; 3
- java/lang/InOut; 10
- java/lang/System1; 57784 // induit appel de currentTimeMillis()
- CHRT222; 2
- CHRT22if ; 28110
- CHRT22sw ; 29664

–Comparatif

trace d'exécution sur PowerPC JavaRunner JDK 1.02:

on declenche l'automate pendant env. 10 sec

if_style : CHRT22if

*L'automate est declenche : **108601** fois en if_style*

on declenche l'automate pendant env. 10 sec

case_style : CHRT22sw

*L'automate est declenche : **116669** fois en case_style*

Annexe 0.1 : désassemblage : occurrence *if*

- iconst_0; 3(0x3); 43
- iconst_1; 4(0x4); 32
- iconst_2; 5(0x5); 8
- iconst_3; 6(0x6); 9
- dconst_0; 14(0xe); 1
- bipush; 16(0x10); 33
- ldc ; 18(0x12); 16
- ldc2_w ; 20(0x14); 8
- iload ; 21(0x15); 5
- dload; 24(0x18); 8
- iload_1; 27(0x1b); 2
- iload_2 ; 28(0x1c); 2
- iload_3 ; 29(0x1d); 2
- dload_1; 39(0x27); 1
- dload_3; 41(0x29); 1
- aload_0 ; 42(0x2a); 93
- aload_1 ; 43(0x2b); 1
- aload_2 ; 44(0x2c); 1
- iaload; 46(0x2e); 26
- istore ; 54(0x36); 3
- ....

- iastore ; 79(0x4f); 26
- ...
- dcmpl ;151(0x97); 2
- ifeq ;153(0x99); 3
- ifne ;154(0x9a); 7
- iflt ;155(0x9b); 2
- ifle ;158(0x9e); 1
- if_icmpeq ;159(0x9f); 6
- if_icmpne ;160(0xa0); 9
- if_icmplt ;161(0xa1); 5
- gotoo ;167(0xa7); 11
- lookupswitch ;171(0xab); 2
- ireturn ;172(0xac); 14
- returnn ;177(0xb1); 5
- getstatic ;178(0xb2); 22
- getfield ;180(0xb4); 73
- putfield ;181(0xb5); 21
- invokevirtual ;182(0xb6); 24
- invokespecial ;183(0xb7); 5
- invokestatic;184(0xb8); 4
- neww ;187(0xbb); 2
- newarray ;188(0xbc); 4

Annexe 0.1 : désassemblage:occurrence *switch*

- iconst_0; 3(0x3); 79
- iconst_1; 4(0x4); 61
- iconst_2; 5(0x5); 14
- iconst_3; 6(0x6); 18
- dconst_0; 14(0xe); 1
- bipush; 16(0x10); 59
- ldc; 18(0x12); 23
- ldc2_w ; 20(0x14); 8
- iload ; 21(0x15); 5
- dload; 24(0x18); 8
- iload_1 ; 27(0x1b); 4
- iload_2 ; 28(0x1c); 4
- iload_3; 29(0x1d); 4
- dload_1; 39(0x27); 1
- dload_3; 41(0x29); 1
- aload_0; 42(0x2a); 181
- aload_1; 43(0x2b); 1
- aload_2 ; 44(0x2c); 1
- iaload; 46(0x2e); 49
- istore; 54(0x36); 3
- dstore ; 57(0x39); 7
- ...

•iastore; 79(0x4f); 52
•dup; 89(0x59); 10
•iadd; 96(0x60); 8 .....
•dcmpl ;151(0x97); 2
•ifeq ;153(0x99); 6
•ifne ;154(0x9a); 14
•iflt ;155(0x9b); 2
•ifle ;158(0x9e); 2
•if_icmpeq ;159(0x9f); 12
•if_icmpne ;160(0xa0); 12
•if_icmplt ;161(0xa1); 9
•gotoo ;167(0xa7); 25
•tableswitch ;170(0xaa); 1
•lookupswitch ;171(0xab); 6
•ireturn ;172(0xac); 28
•returnn ;177(0xb1); 6
•getstatic;178(0xb2); 29
•putstatic ;179(0xb3); 0
•getfield ;180(0xb4); 143
•putfield ;181(0xb5); 42
•invokevirtual ;182(0xb6); 31
•invokespecial;183(0xb7); 6

Annexe 0.2

- TableTst.java, un tri par sélection en récursif et itératif 13000 appels, tri de 20 elts

- iconst_0; 3(0x3); 12
- iconst_1; 4(0x4); 520
- bipush; 16(0x10); 1323
- sipush; 17(0x11); 3
- ldc; 18(0x12); 960
- iload; 21(0x15); 63019
- iload_1; 27(0x1b); 3014
- iload_2; 28(0x1c); 42154
- iload_3 ; 29(0x1d); 61915
- dload_1; 39(0x27); 2
- dload_3; 41(0x29); 2
- aload_0; 42(0x2a); 84930
- aload_1; 43(0x2b); 8
- aload_2; 44(0x2c); 440
- iaload; 46(0x2e); 81312
- istore; 54(0x36); 1647
- istore_1; 60(0x3c); 8
- istore_2; 61(0x3d); 440
- istore_3; 62(0x3e); 1439

•iinc ;132(0x84); 40914
•if_icmp;160(0xa0); 20090
•if_icmplt ;161(0xa1); 744
•if_icmpge;162(0xa2); 220
•if_icmple;164(0xa4); 60488
•gotoo;167(0xa7); 228
•ireturn;172(0xac); 1098
•returnn;177(0xb1); 243
•getstatic ;178(0xb2); 9
•putstatic;179(0xb3); 3
•getfield;180(0xb4); 19
•putfield;181(0xb5); 4
•invokevirtual;182(0xb6); 832
•invokespecial;183(0xb7); 12
•invokestatic;184(0xb8); 9
•neww;187(0xbb); 9
•newarray;188(0xbc); 2
•getfield_quick;206(0xce); 83573
•putfield_quick;207(0xcf); 444
•getstatic_quick;210(0xd2); 153
•invokesuper_quick;216(0xd8); 665

Annexe 0.3

- Statistiques : extraites de "stat.jvm", pour Sieve.java(Score de 2 !,JDK : 15)

- iconst_0; 3(0x3); 30007
- iconst_1; 4(0x4); 16383
- iconst_3; 6(0x6); 3798
- dconst_0; 14(0xe); 1
- sipush; 17(0x11); 2
- ldc; 18(0x12); 6
- ldc2_w; 20(0x14); 3
- iload; 21(0x15); 93796
- dload; 24(0x18); 7
- iload_0; 26(0x1a); 66565
- iload_2; 28(0x1c); 76926
- iload_3; 29(0x1d); 33796
- aload_0; 42(0x2a); 8
- aload_1; 43(0x2b); 62762
- baload; 51(0x33); 16382
- istore; 54(0x36); 33802
- dstore; 57(0x39); 5
- istore_0; 59(0x3b); 1
- istore_2; 61(0x3d); 4
- istore_3; 62(0x3e); 3798

•astore_1; 76(0x4c); 1
•bastore; 84(0x54); 46380
•dup; 89(0x59); 3
•iadd; 96(0x60); 41393
•dsub; 103(0x67); 2
•ddiv; 111(0x6f); 2
•iinc ; 132(0x84); 36564
•dcmpl; 151(0x97); 2
•ifeq; 153(0x99); 16382
•iflt; 155(0x9b); 2
•if_icmpeq; 159(0x9f); 1
•if_icmple; 164(0xa4); 66564
•gotoo; 167(0xa7); 3803
•returnn; 177(0xb1); 11
•getstatic; 178(0xb2); 9
•putstatic; 179(0xb3); 3
•invokevirtual;182(0xb6); 9
•invokespecial;183(0xb7); 11
•invokestatic;184(0xb8); 4
•neww; 187(0xbb); 3
•newarray; 188(0xbc); 1

Annexe 1

- Petit essai de compilation
  - Expression infixée en postfixée + Générateur de code

(limité à "istore,0",iadd,imul,idiv,isub et iload_<n> n[0..5]

voir fichiers : parse.h,.c, generate.h,c et tstparse.c

- ((3+4)*5/3+(3*1))/2
- iconst_3
- iconst_4
- iadd
- iconst_5
- imul
- iconst_3
- idiv
- iconst_3
- iconst_1
- imul
- iadd
- iconst_2
- idiv
- istore_0

- 1+2*3/4*(2+4)
- iconst_1
- iconst_2
- iconst_3
- imul
- iconst_4
- idiv
- iconst_2
- iconst_4
- iadd
- imul
- istore_0

Annexe 2 : Interpreteur en C

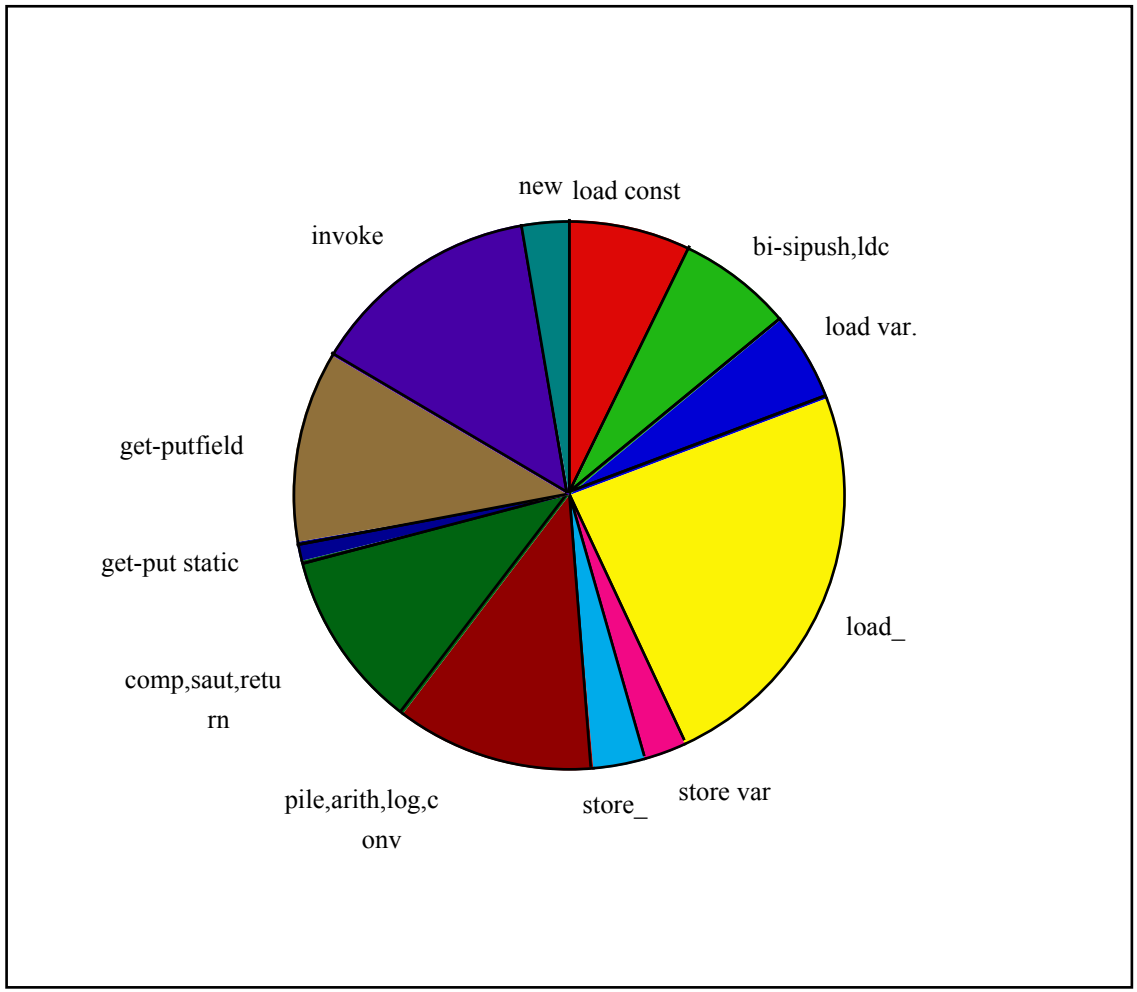
- Un interpréteur de bytecode
  - classes.c /* gestion des classes de l'application */
  - classfie.h,.c /* gestion du .class */
  - constpoo.h,.c /* gestion du contant_pool */
  - exceptio.h,.c /* exception, uniquement issue de la machine */
  - int64.h,.c /* le type long en Java, les entiers sur 64 bits */
  - interpre.h,.c /* l'interpreteur de bytecode */
  - jvm.c /* le module "main" */
  - memalloc.h,.c /* en attendant le ramasse miettes */
  - object.h,.c /* gestion des objets, allocation ... */
  - signatur.h,.c /* la signature "L<>;DL... " en nombre d'octets */
  - statisti.h,.c /* statistiques */
  - types.h,.c /* définition avec vérification des types de la machine */
  - opcodes.h, constant.h
  - soit 6500 lignes de C , sur PPC 47Ko de Code, et 67 KO de données

Annexe 3 : désassembleur

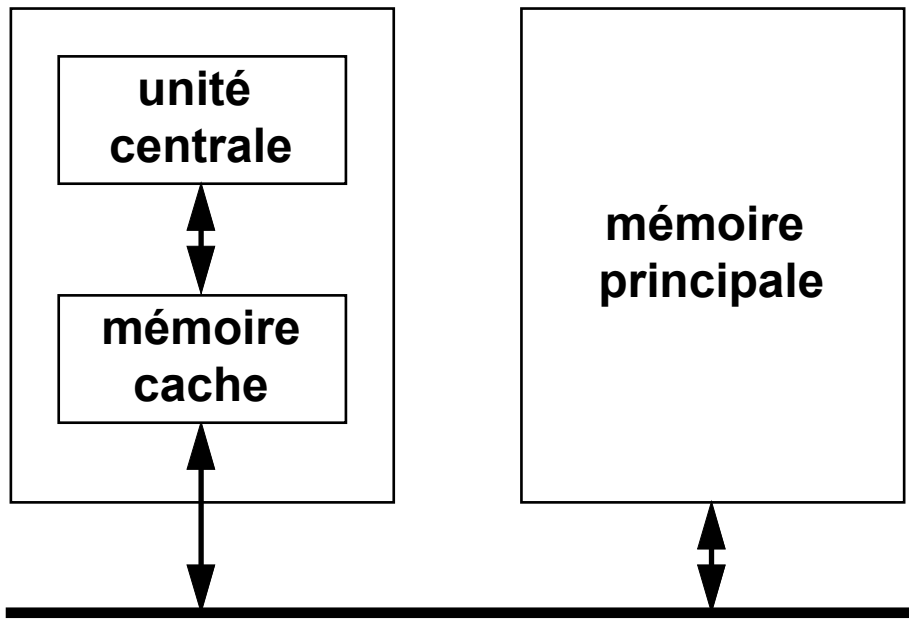
- Un désassembleur de bytecode
  - Voirs fichiers :
    - decode.c /* syntaxe C */ dec_vhdl.c /* syntaxe VHDL */ et dec_java
    - classfile.h,.c
    - constpool.h,.c
    - mnemonic.h,.c
    - opcodes.h, types.h, constant.h
    - statdeco.h,.c

Annexe 4 :Statistiques de désassemblage

- Analyse **statique** du code sur 130 fichiers ".class" extraits de : The Java Tutorial, OOP for the internet. Mary Campione,Kathy Walrath Addison Wesley. Java series



Annexe 5 : Cache sur les instructions



Un test "approprié" pour le cache

```
  public static void main( String args[]){
    int table[] = new int [40];

    table[0] = 10; table[1] = 22;
    for( int i = 2; i < 40; i++){
      table[i] = table[i-1] + table[i-2];
    }
    // ...
  }
}
```

Le "byteCode" du test

- /* 0*/ bipush,40,
- /* 2*/ newarray,10, /* int */
- /* 4*/ astore_1,
- /* 5*/ aload_1,
- /* 6*/ iconst_0,
- /* 7*/ bipush,10,
- /* 9*/ iastore,
- /* 10*/ aload_1,
- /* 11*/ iconst_1,
- /* 12*/ bipush,22,
- /* 14*/ iastore,
- /* 15*/ iconst_2,
- /* 16*/ istore_2,
- /* 17*/ gotoo, 0,20,/*(20 -> 37)*/
- /* 20*/ aload_1,
- /* 21*/ iload_2,
- /* 22*/ aload_1,
- /* 23*/ iload_2,
- /* 24*/ iconst_1,
- /* 25*/ isub,
- /* 26*/ iaload,
- /* 27*/ aload_1,
- /* 28*/ iload_2,
- /* 29*/ iconst_2,
- /* 30*/ isub,
- /* 31*/ iaload,
- /* 32*/ iadd,
- /* 33*/ iastore,
- /* 34*/ iinc, 2, 1,/*(0x2),(0x1)*/
- /* 37*/ iload_2,
- /* 38*/ bipush,40,
- /* 40*/ if_icmplt,255,236,/*(-20 -> 20)*/
- /* 43*/ returnn,

Statistiques : usage du Cache et de la pile

- Statisques Cache de taille 32 : write_back
- NbReadCache = 947
- NbWriteCache = 0
- NbReadMemory = 44
- NbWriteMemory = 0
- NbSubstitution = 12

- Sur le même exemple statistique sur l'usage de la pile
- Nombre de depiler 610
- Nombre d'opérations empiler 612
- Profondeur maximale de la pile 8

Annexe 6 : Nouvelles statistiques

- La gestion de cache en Java comme exemple ...
  - 5 Classes

Le module de Test correspond à l'exécution de l'annexe 5

usage des classes par :

put-get/static,invoke-special/virtual/static

java/lang/Object ; 4

java/lang/InOut ; 13

java/lang/System1 ; 13

TstJVM ; 4

Memory ; 90

Cache ; 1946

CacheData ; 32

Fibonacci_main ; 93

util/jvm/OpCodes ; 2

Annexe 6 : Occurrence :cache.java

- iconst_m1 ; 2(0x2); 992
- iconst_0; 3(0x3); 991
- iconst_1; 4(0x4); 1097
- iconst_2; 5(0x5); 3
- iconst_3; 6(0x6); 3
- iconst_4; 7(0x7); 4
- iconst_5; 8(0x8); 4
- bipush; 16(0x10); 282
- sipush; 17(0x11); 103
- ldc; 18(0x12); 161
- ldc_w; 19(0x13); 75
- iload; 21(0x15); 876
- iload_1; 27(0x1b); 18076
- iload_2; 28(0x1c); 51977
- iload_3; 29(0x1d); 948
- aload_0; 42(0x2a); 54754
- aload_1; 43(0x2b); 2
- aload_2; 44(0x2c); 9
- aaload; 50(0x32); 33677
- baload; 51(0x33); 89
- istore; 54(0x36); 4
- ....

•iinc;132(0x84); 16686
•ifeq;153(0x99); 18254
•if_icmpne;160(0xa0); 16986
•if_icmplt ;161(0xa1); 16648
•gotoo;167(0xa7); 1077
•lookupswitch;171(0xab); 738
•ireturn;172(0xac); 1989
•returnn;177(0xb1); 89
•getstatic;178(0xb2); 16
•putstatic;179(0xb3); 5
•getfield;180(0xb4); 35
•putfield;181(0xb5); 21
•invokevirtual;182(0xb6); 1056
•invokespecial;183(0xb7); 13
•invokestatic;184(0xb8); 2
•neww;187(0xbb); 37
•newarray;188(0xbc); 2
•anewarray;189(0xbd); 2
•arraylength;190(0xbe); 46
•getfield_quick;206(0xce); 86154
•putfield_quick;207(0xcf); 1203
•getstatic_quick;210(0xd2); 88
•invokesuper_quick;216(0xd8); 1053

Annexe 7 : Proplog

- Proplog en chaînage avant comme exemple ...
  - Computing with logic, Logic programming with Prolog
  - David Maier & David S. Warren, The Benjamin/cummings
  - 11 règles, question ?- h.
    - f :- b, d, e.
    - a :- d,g.
    - a :- c,f.
    - x :- b.
    - e :- d.
    - h :- a,x.
    - d :- c.
    - a :- x,c.
    - d :- x,b.
    - b.
    - c.

Annexe 7 : Nouvelles statistiques

- Proplog en chainage Avant
  - 8 Classes

usage des classes par :
put-get/static,invoke-special/virtual/static

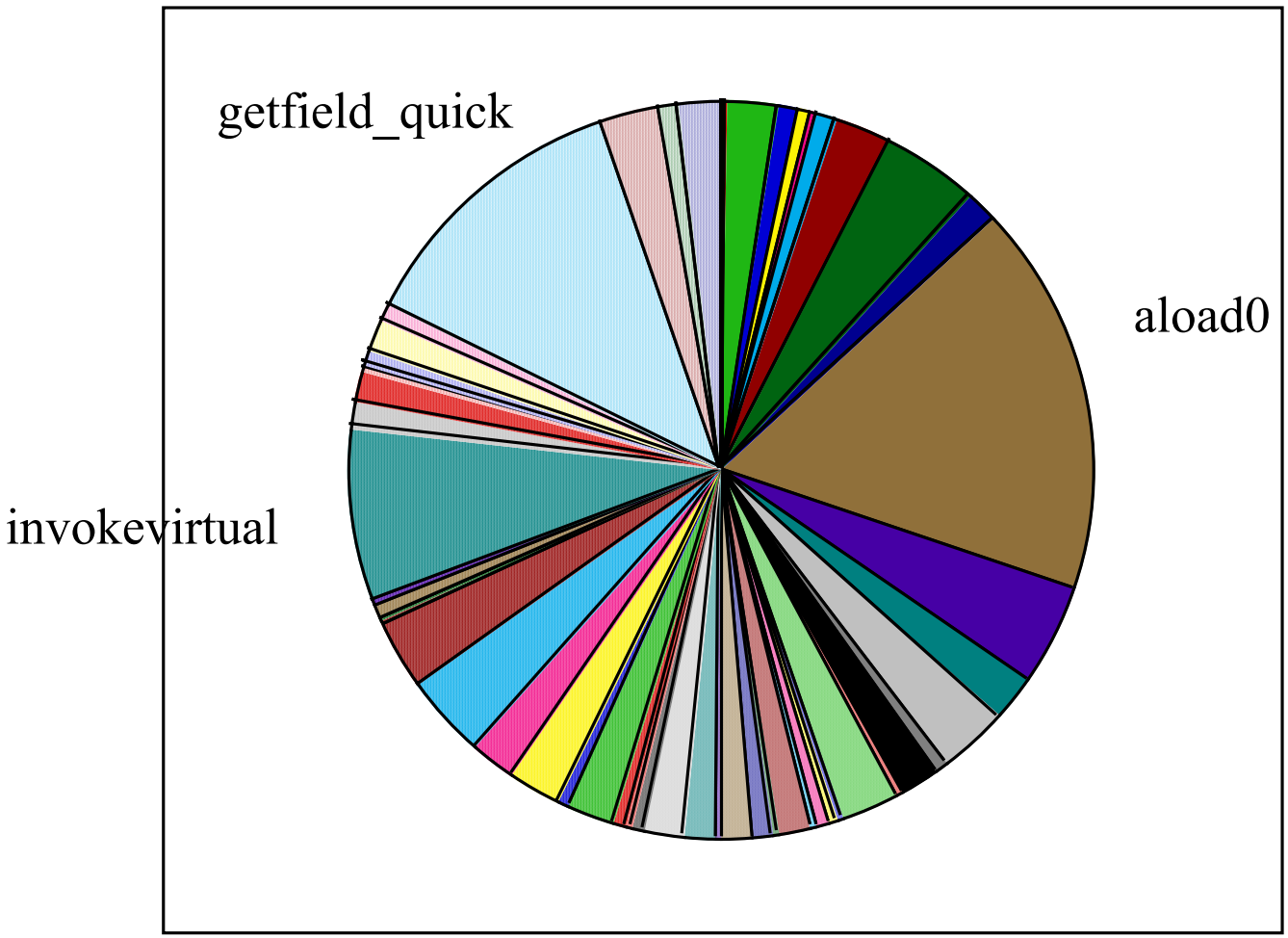
java/lang/Object ; 9
java/lang/InOut ; 63
java/lang/System1 ; 63
Proplog0 ; 22
ClausesList ; 97
SymbolsList ; 125
Symbol ; 103
String1 ; 85
SymbolNode ; 164
Clause ; 71
Proplog ; 2

Annexe 7 : Occurrence :Proplog0.java

- aconst_null; 1(0x1); 33
- iconst_0; 3(0x3); 169
- iconst_1; 4(0x4); 68
- bipush; 16(0x10); 29
- ldc; 18(0x12); 35
- iload ; 21(0x15); 62
- iload_1; 27(0x1b); 179
- iload_2; 28(0x1c); 312
- iload_3; 29(0x1d); 116
- aload_0; 42(0x2a); 1290
- aload_1; 43(0x2b); 350
- aload_2; 44(0x2c); 141
- aload_3; 45(0x2d); 227
- aaload; 50(0x32); 60
- ....
- dup;89(0x59); 116
- iadd;96(0x60); 11
- iinc;132(0x84); 118
- ifeq;153(0x99); 115
- ifne;154(0x9a); 44
- if_icmpeq;159(0x9f); 29
- if_icmpne;160(0xa0); 29

- if_icmplt;161(0xa1); 148
- if_icmpge;162(0xa2); 58
- gotoo;167(0xa7); 153
- ireturn;172(0xac); 161
- areturn;176(0xb0); 265
- returnn;177(0xb1); 229
- getstatic ;178(0xb2); 25
- putstatic;179(0xb3); 5
- getfield;180(0xb4); 37
- putfield;181(0xb5); 17
- invokevirtual;182(0xb6); 590
- invokespecial;183(0xb7); 87
- invokestatic;184(0xb8); 2
- neww;187(0xbb); 105
- newarray;188(0xbc); 30
- anewarray;189(0xbd); 1
- arraylength;190(0xbe); 28
- ifnull;198(0xc6); 103
- ifnonnull;199(0xc7); 68
- getfield_quick;206(0xce); 928
- putfield_quick;207(0xcf); 192
- getstatic_quick ;210(0xd2); 54
- invokesuper_quick;216(0xd8);142

Annexe 7 : Occurrences : Proplog0



Annexe 8

- Analyse en détail d'un .class
  - Le source Java
  - Le fichier .class brut
  - l'analyse des principaux champs
    - Constant pool
    - Interface
    - Fields
    - Methods
      - Method_attribute
        - Code_attribute
        - LineNumberTable attribute, LocalVariableTable attribute
    - SourceFile_attribute

- http://java.sun.com:81/docs/books/vmspec/html

Annexe 8 : le Source, UneClasse.java

```
public class UneClasse extends Object{
  public static void main(String[] args) {
     for(int i=0; i < args.length; i++){
       UneClasse e = new UneClasse(args[i]);
       e.print();
     }
  }

  private String str;
  UneClasse( String s){  this.str = s;}

  void print(){
    System.out.print(enTete);
    System.out.println(this);
  }

  public String toString(){   return str;  }

  private static String enTete;
  static{
    enTete = "argument : ";
  }}
```

Annexe 8 : le fichier, UneClasse.class

CA FE BA BE 00 03 00 2D 00 3B 08 00 23 07 00 2B 07 00 2C 07
00 32 07 00 1F 09 00 03 00 11 0A 00 03 00 10 0A 00 05 00 12
09 00 04 00 15 0A 00 02 00 0F 0A 00 03 00 14 09 00 03 00 0E
0A 00 05 00 13 0C 00 33 00 1E 0C 00 30 00 36 0C 00 30 00 2A
0C 00 2F 00 1E 0C 00 1A 00 2A 0C 00 16 00 19 0C 00 1A 00 36
0C 00 29 00 31 01 00 07 70 72 69 6E 74 6C 6E 01 00 04 74 68
69 73 01 00 14 28 29 4C 6A 61 76 61 2F 6C 61 6E 67 2F 53 74
72 69 6E 67 3B 01 00 15 28 4C 6A 61 76 61 2F 6C 61 6E 67 2F
4F 62 6A 65 63 74 3B 29 56 01 00 05 70 72 69 6E 74 01 00 0D
43 6F 6E 73 74 61 6E 74 56 61 6C 75 65 01 00 0B 4C 55 6E 65
43 6C 61 73 73 65 3B 01 00 12 4C 6F 63 61 6C 56 61 72 69 61
62 6C 65 54 61 62 6C 65 01 00 12 4C 6A 61 76 61 2F 6C 61 6E
67 2F 53 74 72 69 6E 67 3B 01 00 13 6A 61 76 61 2F 69 6F 2F
50 72 69 6E 74 53 74 72 65 61 6D 01 00 0A 45 78 63 65 70 74
..... !

Annexe 8 : l'entête

CA FE BA BE 00 03 00 2D

soient

- u4 magic; **CA FE BA BE**
- u2 minor_version; **00 03**
- u2 major_version; **00 2D**

Annexe 8 : le constant pool

00 3B soient 59 entrées

- u2 constant_pool_count;
- cp_info constant_pool[constant_pool_count];

08 00 23

- Le premier élément a un tag de 8 soit CONSTANT_String

alors

CONSTANT_String_info{
  u1 tag; **08**
  u2 string_index; **00 23** *(35 en base 10)*
}

[35] tag: 1 length: 11 argument :

Annexe 8 : le constant pool [1..5]

07 00 2B

– Le deuxième élément a un tag de 7 soit CONSTANT_Class

alors

CONSTANT_Class_info{

u1 tag; **07**

u2 name_index; **00 2B** *(43)*

}

[43] tag: 1 length: 16 java/lang/Object

- suivis de **07 00 2C 07 00 32 07 00 1F**
- **[3] tag 7 name_index 00 2C** ***(44)***
- **[4] tag 7 name_index 00 32** ***(50)***
- **[5] tag 7 name_index 00 1F** ***(31)***

Annexe 8 : le constant pool [6]

09 00 03 00 11

– Le 6 ème élément a un tag de 9 soit CONSTANT_Fieldref

alors

CONSTANT_Fieldref_info{
  u1 tag; **09**
  u2 class_index; **00 03**
  u2 class_and_type_index; **00 11** ***(17)***
}

[17] tag: 12 class_index: 47 descriptor_index: 30

– *[47] tag: 1 length: 3 str*

– *[30] tag: 1 length: 18 Ljava/lang/String;*

Annexe 8 : le constant pool [7]

0A 00 03 00 10

– Le 7 ème élément a un tag de 10 soit CONSTANT_Methodref alors

CONSTANT_Methodref_info{
  u1 tag; **0A**
  u2 class_index; **00 03**
  u2 class_and_type_index; **00 10** ***(16)***
}

[3] tag: 7 name_index: 44
– *[44] tag: 1 length: 9 UneClasse*
[16] tag: 12 class_index: 48 descriptor_index: 42
– *[48] tag: 1 length: 6 <init>*
– *[42] tag: 1 length: 21 (Ljava/lang/String;)V*

Annexe 8 : le constant pool [8..13]

suivis de :

0A 00 05 00 12

- [8] tag: 10 class_index: 5 name_and_type_index: 18

09 00 04 00 15

- [9] tag: 9 class_index: 4 name_and_type_index: 21

0A 00 02 00 0F

- [10] tag: 10 class_index: 2 name_and_type_index: 15

0A 00 03 00 14

- [11] tag: 10 class_index: 3 name_and_type_index: 20

09 00 03 00 0E

- [12] tag: 9 class_index: 3 name_and_type_index: 14

0A 00 05 00 13

- [13] tag: 10 class_index: 5 name_and_type_index: 19

Annexe 8 : le constant pool [14]

0C 00 33 00 1E

– Le 14 ème élément a un tag de 12 soit CONSTANT_NameAndType

alors

CONSTANT_NameAndType{

u1 tag;	**0C**	
u2 name_index;	**00 33**	***(51)***
u2 descriptor_index;	**00 1E**	***(30)***

}

[51] tag: 1 length: 6 enTete
[30] tag: 1 length: 18 Ljava/lang/String;

Annexe 8 : le constant pool [15..21]

suivis de :

0C 00 30 00 36 0C 00 30 00 2A
0C 00 2F 00 1E 0C 00 1A 00 2A 0C 00 16 00 19 0C 00 1A 00 36
0C 00 29 00 31

- [15] tag: 12 class_index: 48 descriptor_index: 54
- [16] tag: 12 class_index: 48 descriptor_index: 42
- [17] tag: 12 class_index: 47 descriptor_index: 30
- [18] tag: 12 class_index: 26 descriptor_index: 42
- [19] tag: 12 class_index: 22 descriptor_index: 25
- [20] tag: 12 class_index: 26 descriptor_index: 54
- [21] tag: 12 class_index: 41 descriptor_index: 49

Annexe 8 : le constant pool [22]

01 00 07 70 72 69 6E 74 6C 6E

– Le 22 ème élément a un tag de 1 soit CONSTANT_Utf8

alors

CONSTANT_Utf8_info{
  u1 tag; **01**
  u2 length; **00 07**
  u1 bytes[length]; **70 72 69 6E 74 6C 6E**
  p r i n t l n
}

Annexe 8 : le constant pool [23..40]

- [23] tag: 1 length: 4 this
- [24] tag: 1 length: 20 ()Ljava/lang/String;
- [25] tag: 1 length: 21 (Ljava/lang/Object;)V
- [26] tag: 1 length: 5 print
- [27] tag: 1 length: 13 ConstantValue
- [28] tag: 1 length: 11 LUneClasse;
- [29] tag: 1 length: 18 LocalVariableTable
- [30] tag: 1 length: 18 Ljava/lang/String;
- [31] tag: 1 length: 19 java/io/PrintStream
- [32] tag: 1 length: 10 Exceptions
- [33] tag: 1 length: 14 UneClasse.java
- [34] tag: 1 length: 15 LineNumberTable
- [35] tag: 1 length: 11 argument :
- [36] tag: 1 length: 1 I
- [37] tag: 1 length: 10 SourceFile
- [38] tag: 1 length: 14 LocalVariables
- [39] tag: 1 length: 4 Code
- [40] tag: 1 length: 8 toString

Annexe 8 : le constant pool [41..58]

- [41] tag: 1 length: 3 out
- [42] tag: 1 length: 21 (Ljava/lang/String;)V
- [43] tag: 1 length: 16 java/lang/Object
- [44] tag: 1 length: 9 UneClasse
- [45] tag: 1 length: 4 main
- [46] tag: 1 length: 22 ([Ljava/lang/String;)V
- [47] tag: 1 length: 3 str
- [48] tag: 1 length: 6 <init>
- [49] tag: 1 length: 21 Ljava/io/PrintStream;
- [50] tag: 1 length: 16 java/lang/System
- [51] tag: 1 length: 6 enTete
- [52] tag: 1 length: 1 s
- [53] tag: 1 length: 8 <clinit>
- [54] tag: 1 length: 3 ()V
- [55] tag: 1 length: 1 i
- [56] tag: 1 length: 4 args
- [57] tag: 1 length: 19 [Ljava/lang/String;
- [58] tag: 1 length: 1 e

Annexe 8 : informations sur la classe

00 01 00 03 00 02

soient

- u2 access_flags; **00 01** ***(public)***
- u2 this_class; **00 03** ***(index constantpool)***
- u2 super_class; **00 02** ***(index constantpool)***

- **[3] tag 7 name_index 00 2C** ***(44)***
  - *[44] tag 1 length 9 UneClasse*
- **[2] tag 7 name_index 00 2B** ***(43)***
  - *[43] tag 1 length 16 java/lang/Object*

Annexe 8 : Interface

00 00

soient

- u2 interfaces_count; **00 00**
- u2 interfaces [interfaces_count];

Annexe 8 : Fields

00 02
00 02 00 2F 00 1E 00 00
00 0A 00 33 00 1E 00 00

soient

- u2 fields_count; **00 02**
- field_info fields[fields_count];

- field_info{
- u2 access_flags; **00 02** **00 0A**
- u2 name_index; **00 2F** **00 33**
- u2 descriptor_index; **00 1E** **00 1E**
- u2 attributes_count; **00 00** **00 00**
- attribute_info attributes[attributes_count];
- };

Annexe 8 : Fields

00 02 00 2F 00 1E 00 00

- *[47] tag: 1 length: 3 str*
- *[30] tag: 1 length: 18 Ljava/lang/String;*

00 0A 00 33 00 1E 00 00

- *[51] tag: 1 length: 6 enTete*
- *[30] tag: 1 length: 18 Ljava/lang/String;*

descriptor_index

- *FieldType ::= BaseType | ObjectType | ArrayType*
- *BaseType*
  - **B** byte **C** char
  - **D** double **F** float
  - **I** int **J** long
  - **S** short **Z** boolean
- *ObjectType* **L**<classname>**;**
- *ArrayType* **[** table

Annexe 8 : Methods

00 05

00 09 00 2D 00 2E 00 01

00 27 00 00 00 6C 00 04 00 03 00 00 00 1E 03 3C

A7 00 15 BB 00

03 59 2A 1B 32 B7 00 07 4D 2C B6 00 0B 84 01 01

1B 2A BE A1 FF EB B1

soient

- u2 methods_count; **00 05**
- method_info methods[methods_count];

- method_info{
- u2 access_flags; **00 09**
- u2 name_index; **00 2D**
- u2 descriptor_index; **00 2E**
- u2 attributes_count; **00 01**
- attribute_info attributes[attributes_count];
- };

Annexe 8 : Method_info

- method_info{
- u2 access_flags; **00 09**
- u2 name_index; **00 2D** ***(45)***
- u2 descriptor_index; **00 2E** ***(46)***
- u2 attributes_count; **00 01**
- attribute_info attributes[attributes_count];
- };

- *[45] tag: 1 length: 4 main*
- *[46] tag: 1 length: 22 ([Ljava/lang/String;)V*

Annexe 8 : attribute_info

- **00 27 00 00 00 6C**

- attribute_info{
- u2 attribute_name_index; **00 27** ***(39)***
- u4 attribute_length; **00 00 00 6C**
- u1 info[attribute_length];
- }

- *[39] tag: 1 length: 4* ***Code***

Annexe 8 : Code_attribute

- **00 04 00 03 00 00 00 1E**

- Code_attribute{
- u2 attribute_name_index;
- u4 attribute_length;
- u2 max_stack; **00 04**
- u2 max_locals; **00 03**
- u4 code_length; **00 00 00 1E**
- u1 code[code_length];

03 3C A7 00 15 BB 00 03 59 2A 1B 32 B7 00 07 4D 2C

B6 00 0B 84 01 01 1B 2A BE A1 FF EB B1 03 00

- u2 exception_table_length; **00 00**
- exception_info exception_table[exception_table_length];
- u2 attributes_count;
- attribute_info attributes[attributes_count];
- }

Annexe 8 : Code_attribute

- **03 3C A7 00 15 BB 00 03 59 2A 1B 32 B7 00 07 4D 2C**
- **B6 00 0B 84 01 01 1B 2A BE A1 FF EB B1**

```
/*   0*/ iconst_0,
/*   1*/ istore_1,
/*   2*/ gotoo, 0,21,/*(21 -> 23)*/
/*   5*/ neww, 0, 3,/* UneClasse */
/*   8*/ dup,
/*   9*/ aload_0,
/*  10*/ iload_1,
/*  11*/ aaload,
/*  12*/ invokespecial, 0, 7,/* UneClasse.<init>(Ljava/lang/String;)V */
/*  15*/ astore_2,
/*  16*/ aload_2,
/*  17*/ invokevirtual, 0,11,/* UneClasse.print()V */
/*  20*/ iinc, 1, 1,
/*  23*/ iload_1,
/*  24*/ aload_0,
/*  25*/ arraylength,
/*  26*/ if_icmplt,255,235,/*(-21 -> 5)*/
/*  29*/ returnn,
```

Annexe 8 : method_info

- *MethodDescriptor ::= **(** FieldType ***)** ReturnDescriptor*
- *ReturnDescriptor ::= FieldType **V***
    - **V** si le type retourné est void

invokespecial, 0, 7,

[7] tag: 10 class_index: 3 name_and_type_index: 16
- *[3] tag: 7 name_index: 44*
    - *[44] tag: 1 length: 9 UneClasse*

[16] tag: 12 class_index: 48 descriptor_index: 42
- *[48] tag: 1 length: 6 <init>*
- *[42] tag: 1 length: 21 (Ljava/lang/String;)V*

soit **UneClasse.<init>(Ljava/lang/String;)V**

Annexe 8 : method_info

- **invokevirtual, 0,11, /* UneClasse.print()V */**

[11] tag: 10 class_index: 3 name_and_type_index: 20
- *[3] tag: 7 name_index: 44*
  - *[44] tag: 1 length: 9 UneClasse*

[20] tag: 12 class_index: 26 descriptor_index: 54
- *[26] tag: 1 length: 5 print*
- *[54] tag: 1 length: 3 ()V*

- **neww, 0, 3,/* UneClasse */**

[3] tag: 7 name_index: 44
- *[44] tag: 1 length: 9 UneClasse*

Annexe 8 : Code_attribute; lecture

```
for( i = 0; i < methods_count;i++){
  methods[i].access_flags = readu2();
  methods[i].name_index = readu2();
  methods[i].descriptor_index = readu2();

  methods[i].attributes_count = readu2();
  methods[i].attributes = NULL;
  for(j = 0;j < methods[i].attributes_count;j++){
    name_index = readu2();
    attribute_length = readu4();
    Utf8 = Utf8_info(constant_pool,name_index);
    if (strcmp( Utf8->bytes,"Code") == 0){

      /* lecture du bytecode  */

      methods[i].attributes[0].exception_table_length = readu2();
      if (methods[i].attributes[0].exception_table_length > 0){

        /* lecture éventuelle des tables d'exception */
```

Annexe 8 : exception_info

- **exception_info**{
- u2 start_pc;
- u2 end_pc;
- u2 handler_pc;
- u2 catch_type;
- } ;

Annexe 8 : Code_attribute.attribute_info

- u2 attributes_count;
- attribute_info attributes[attributes_count];

line_number_info{
    u2 start_pc;
    u2 line_pc;
};

LineNumberTable_attribute{
    u2 attribute_name_index;
    u4 attribute_length;
    u2 line_number_table_length;
    line_number_info line_number_table[line_number_table_length];
};

Débogueur : liaison avec le source

Annexe 8 : Code_attribute.attribute_info

local_variable_info{
    u2 start_pc;
    u2 length;
    u2 name_index;
    u2 descriptor_index;
    u2 index;
};

LocalVariableTable_attribute{
    u2 attribute_name_index;
    u4 attribute_length;
    u2 local_variable_table_length;
    local_variable_info
      local_variable_table[local_variable_table_length];
};

Débogueur : Valeurs des variables en cours d'exécution

Annexe 8 : Code_attribute.attribute_info

- **00 02 00 22 00 00 00 16**
- **00 05**
- **....**

- Code_attribute{
- ....
- u2 attributes_count; **00 02**
- attribute_info attributes[attributes_count];

- }
- [34] tag: 1 length: 15 LineNumberTable

- alors

u2 attribute_name_index; **00 22**

u4 attribute_length; **00 00 00 16**

u2 line_number_table_length; **00 05**

line_number_info line_number_table[line_number_table_length];

Annexe 8 : Code_attribute.attribute_info

- **00 1D**
- **00 00 00 20**
- **00 03 ....**

- [29] tag: 1 length: 18 LocalVariableTable

- alors

  u2 attribute_name_index; **00 1D**

  u4 attribute_length; **00 00 00 20**

  u2 local_variable_table_length; **00 03**

  local_variable_info *local_variable_table; **....**

Annexe 8 : Code_attribute.attribute_info; lecture

```
methods[i].attributes[0].attributes_count = readu2();
for(k=0;k<methods[i].attributes[0].attributes_count;k++){
  name_index = readu2();
  attribute_length = readu4();
  Utf8 = Utf8_info(constant_pool,name_index);

  if (strcmp( Utf8->bytes,"LineNumberTable")==0){
    line_attribute.attribute_name_index = name_index;
    line_attribute.attribute_length = attribute_length;
    line_attribute.line_number_table_length = readu2();
      /* lecture de l'attribut : LineNumberTable */
  } else if (strcmp(Utf8->bytes,"LocalVariableTable")==0){
    local_attribute.attribute_name_index = name_index;
    local_attribute.attribute_length = attribute_length;
    local_attribute.local_variable_table_length = readu2();
     /* lecture de l'attribut LocalVariableTable*/
  } else {
     /* lecture du fichier on passe,  le futur ... */
  }
```

Annexe 8 : SourceFile_attribute

- **00 01 00 25 00 00 00 02 00 21**

- attribute_count **00 01**
- typedef struct{
- u2 attribute_name_index; **00 25** ***(37)***
- u4 attribute_length; **00 00 00 02**
- u2 sourcefile_index; **00 21** ***(33)***
- } **SourceFile_attribute;**

[37] tag: 1 length: 10 SourceFile

[33] tag: 1 length: 14 UneClasse.java