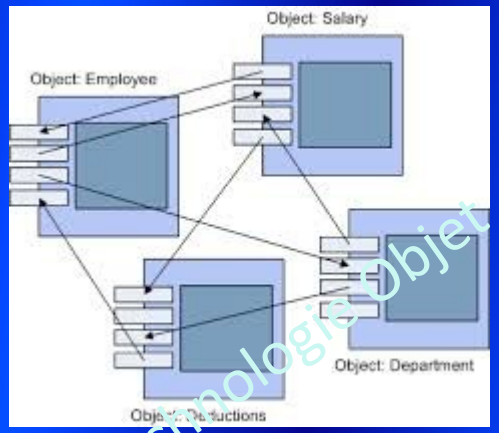
Chapitre 3

La Machine Virtuelle JAVA : JVM



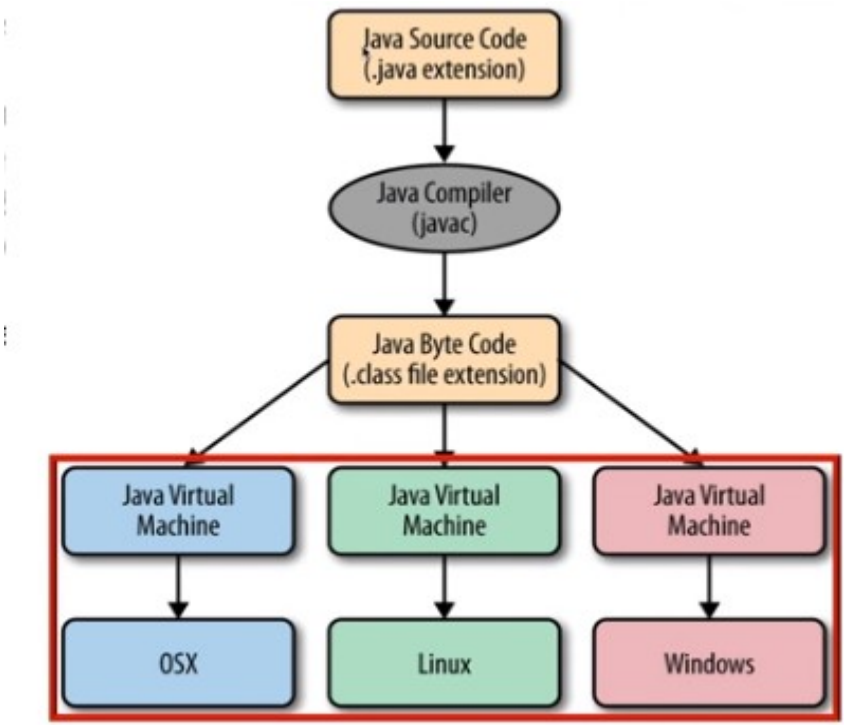
Module : Technologie objet
Master 1 IRC

Mr A. Dekhinet
Université de BATNA
Département d'Informatique

- ✓ JVM : Java Virtual Machine (Portable, Sure, Robuste)
- ✓ Elle fournit une abstraction de la machine réelle : Un programme qui exécute un programme
- ✓ Elle exécute un ByteCode (Code intermédiaire) : Code indépendant de la machine
  - ↳ Portabilité des programme JAVA : WORA (Write Once, Run Anywhere)
- ✓ La JVM supporte actuellement plusieurs langage : Ada, C, LISP, Python
- ✓ Elle existe en deux versions : Client et Serveur
- ✓ Comme toute machine, elle a son propre jeu d'instruction et manipule des zones mémoires durant l'exécution (run time)
- ✓ Byte Code : Code Operation sur 8 bit (1 Byte)
- ✓ Jeu d'instruction : Mélange RISC et CISC
- ✓ La JVM est une machine a pile

  a = b + c

  push b, push c, add , store a

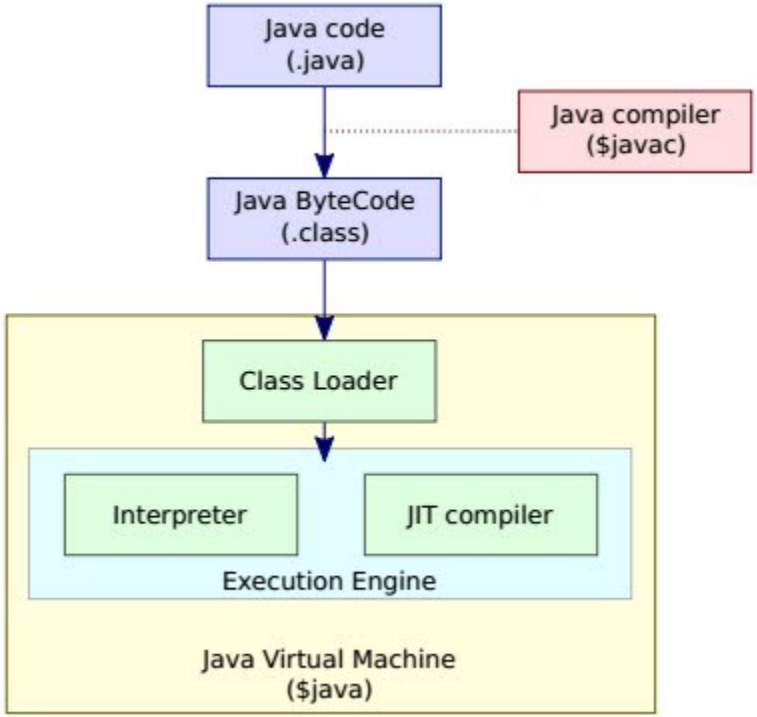


- ✓ Execution Engine : Execute le Byte code
- ✓ Class Loader : Charge le Byte code contenu dans le fichier classe dans le Runtime
- ✓ Interpreter : Interprete le Byte code

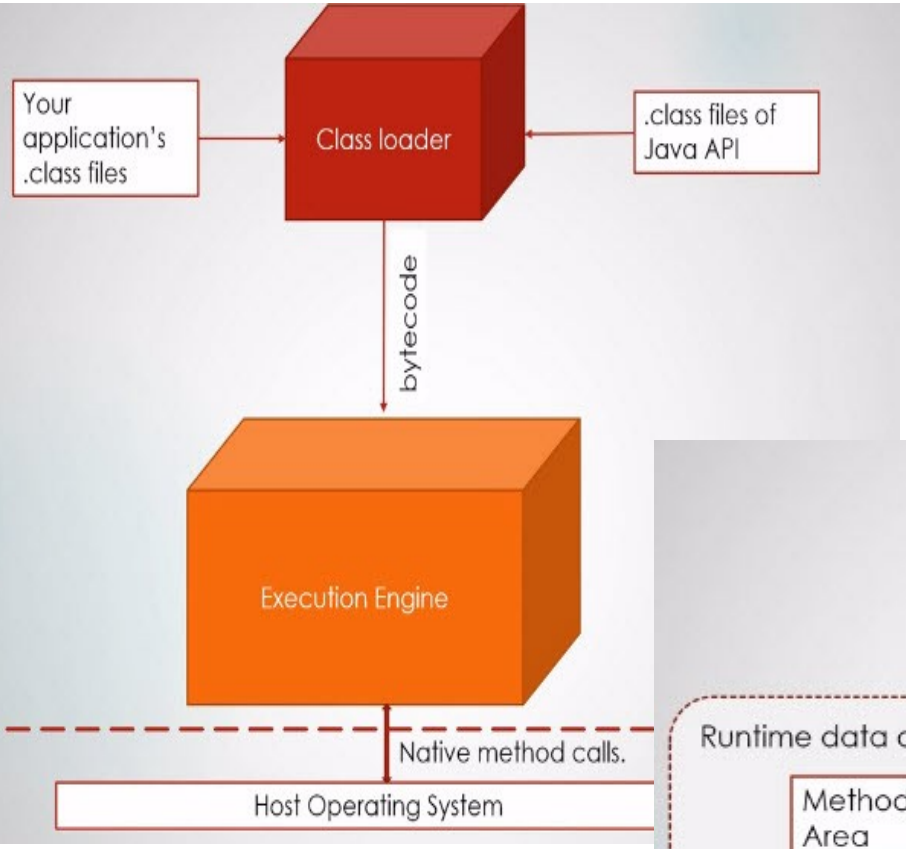
JIT (Just In Time) : Compile le Byte code en instructions natives (langage machine physique cible) a la volé

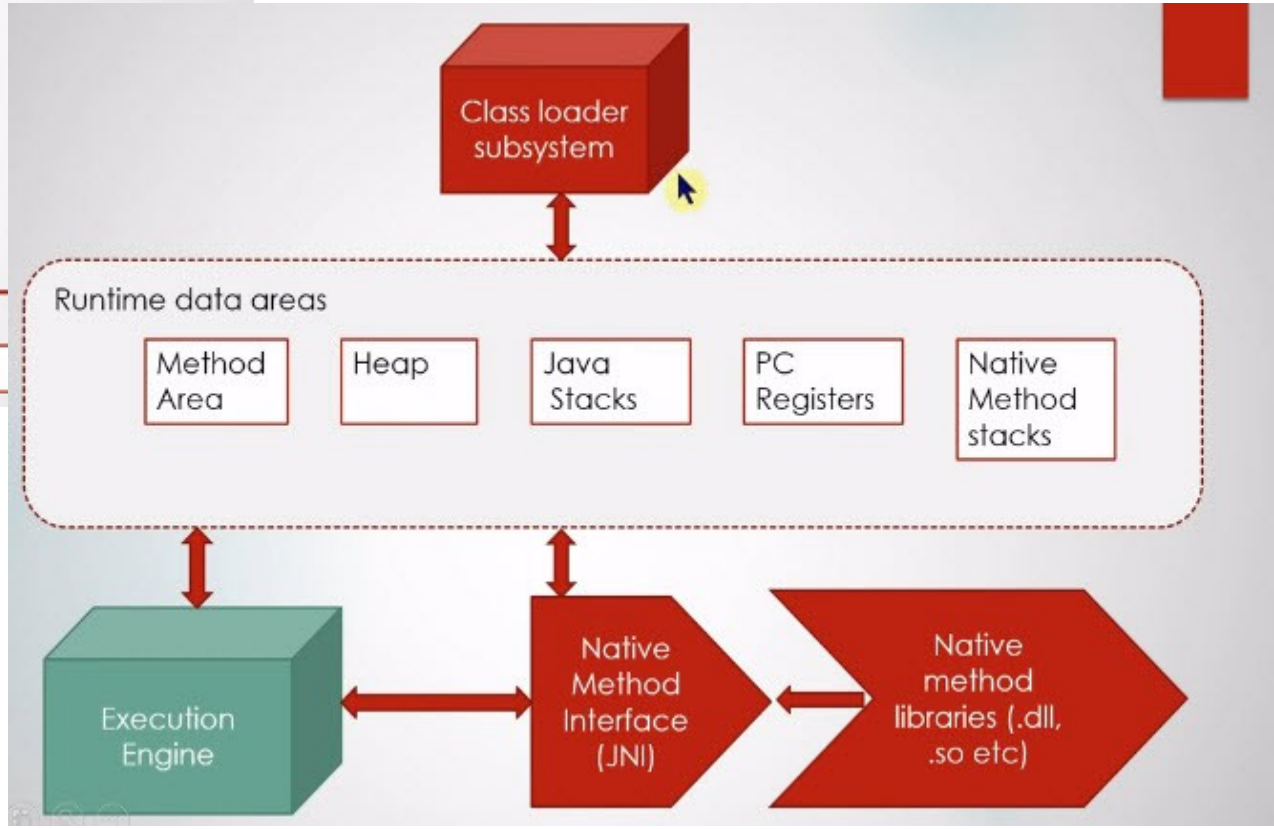
Certaines JVM utilise une approche mixte : HotSpot JVM de SUN (Oracle)

Jrocket JVM de Oracle utilise JIT



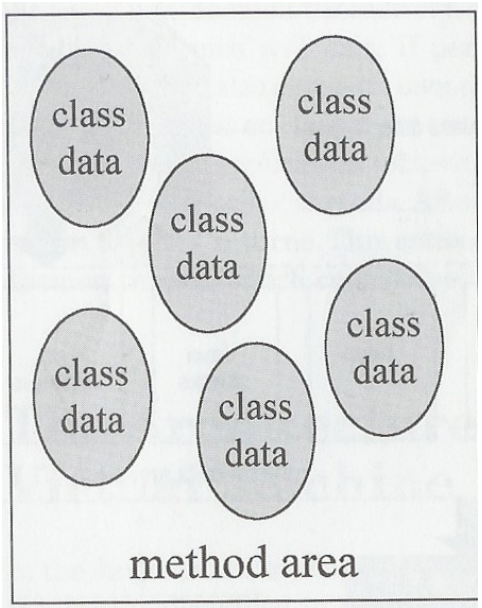
Modèle Mémoire : Run Time

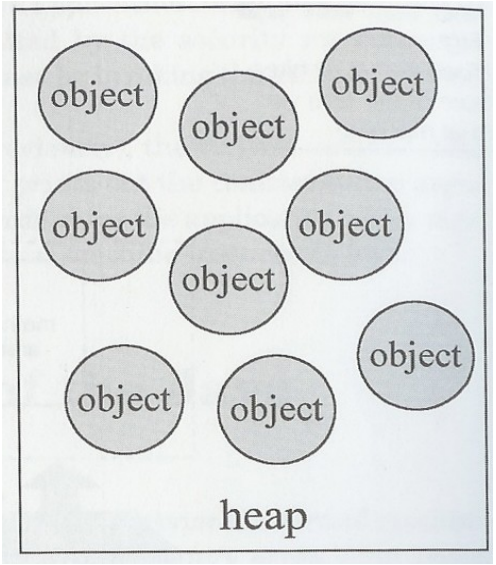


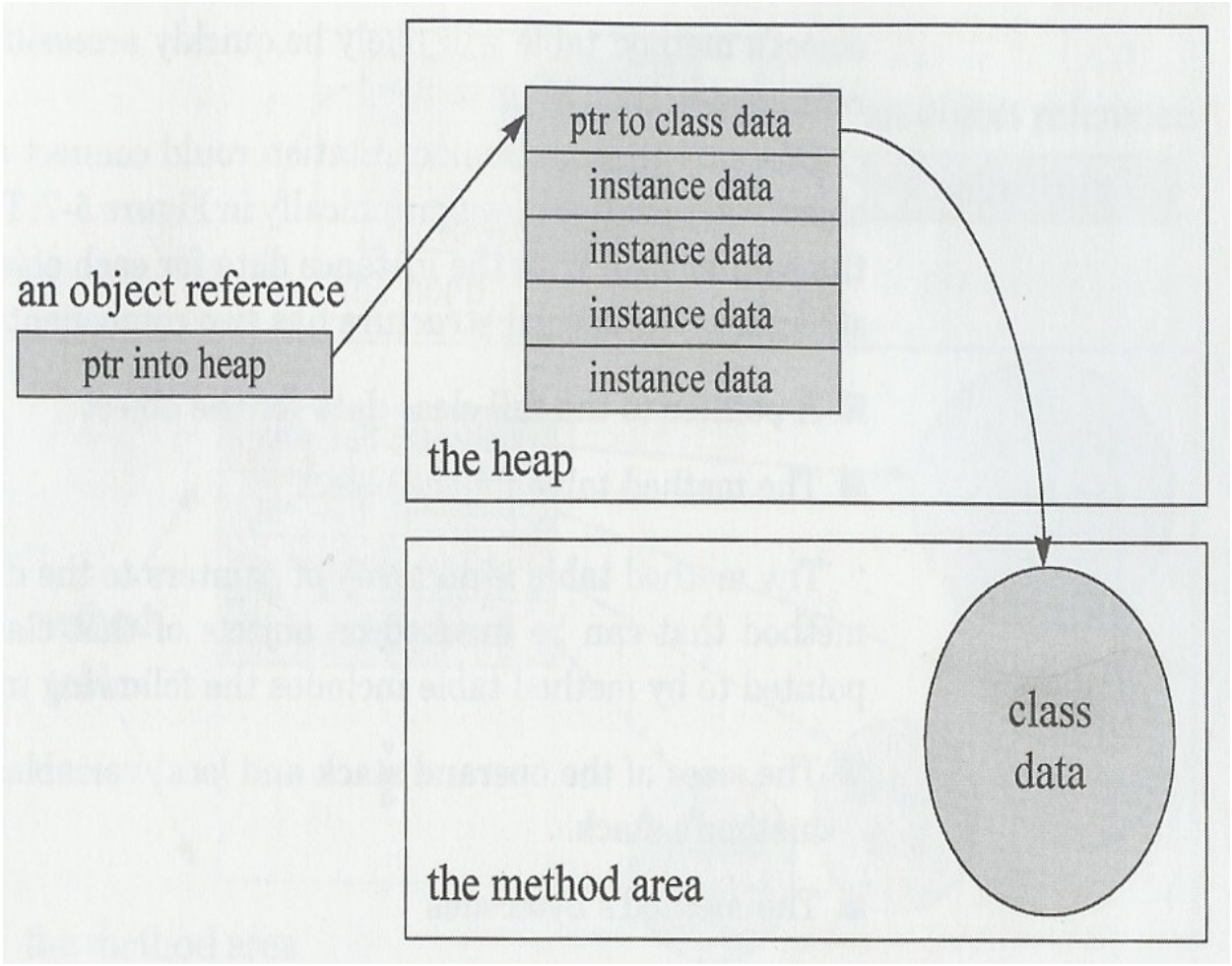


- ✓ Method area
  - Une zone par JVM : Partagée
  - Zone des méthodes contenant le byte code, les constantes, …
- ✓ Heap (Tas)
  - Une zone par JVM : Partagée
  - Taille du tas paramétrable par –Xms<nm> m megabyte (-Xms128m)
  - Contenant les objets instances des classes (new)
  - Gérée en Ramasse – miettes (Garbage collection)
- ✓ Registres PC (Program Counter)
  - Compteur ordinal
  - Un compteur par thread
- ✓ Stack (Pile)
  - Une pile par thread
  - Taille de la pile paramétrable par –Xss<nk> n kbytes (-Xss400k)
  - Contient les blocs d'activation (Frame) : Etats courant du thread
  - Bloc engendré par l'invocation des méthodes
  - Le Bloc contient : Variables locales, Paramètres, Opérandes des instructions Byte code, …
  - A un frame est associé trois registres : frame, optop, vars
  - Empilement et Dépilement de frames
- ✓ Pile pour méthodes natives non Byte code, écrite dans le langage natif de la machine

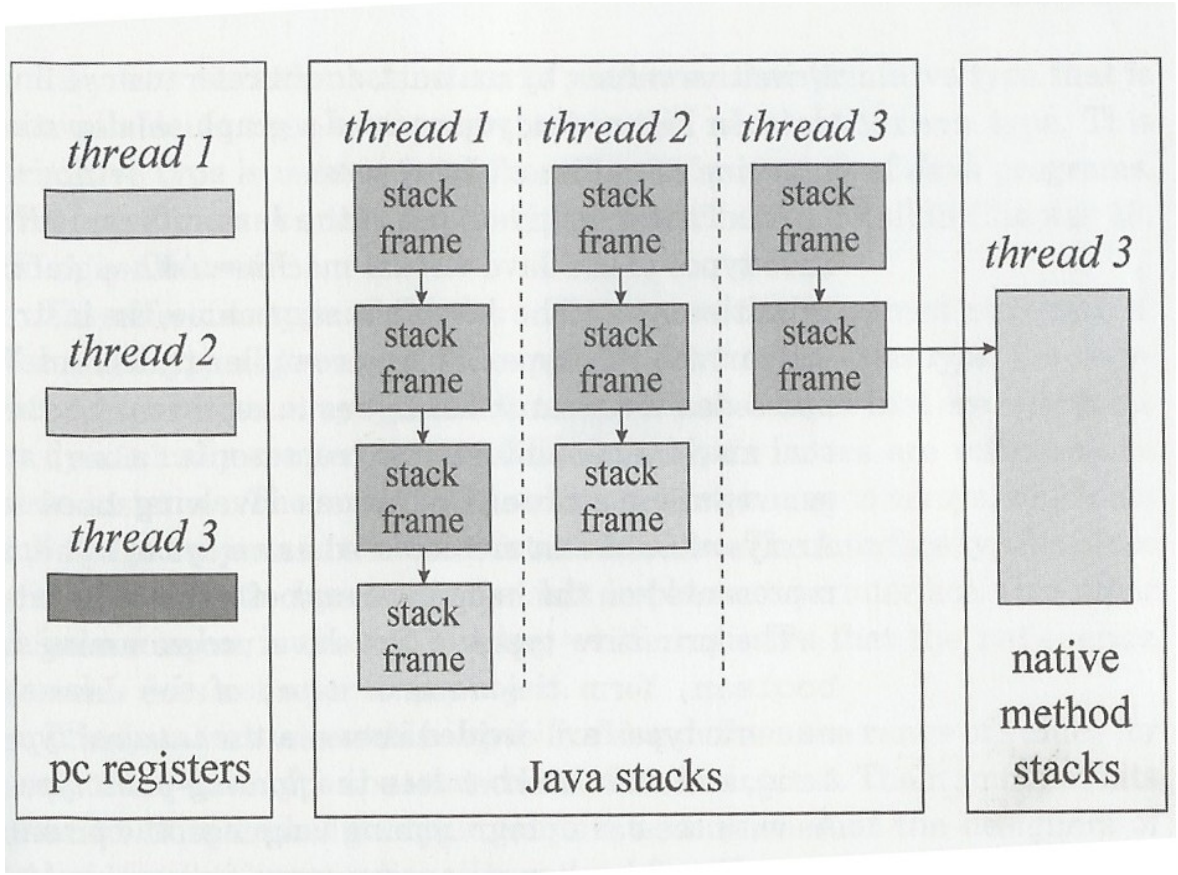
Modèle Mémoire : Run Time







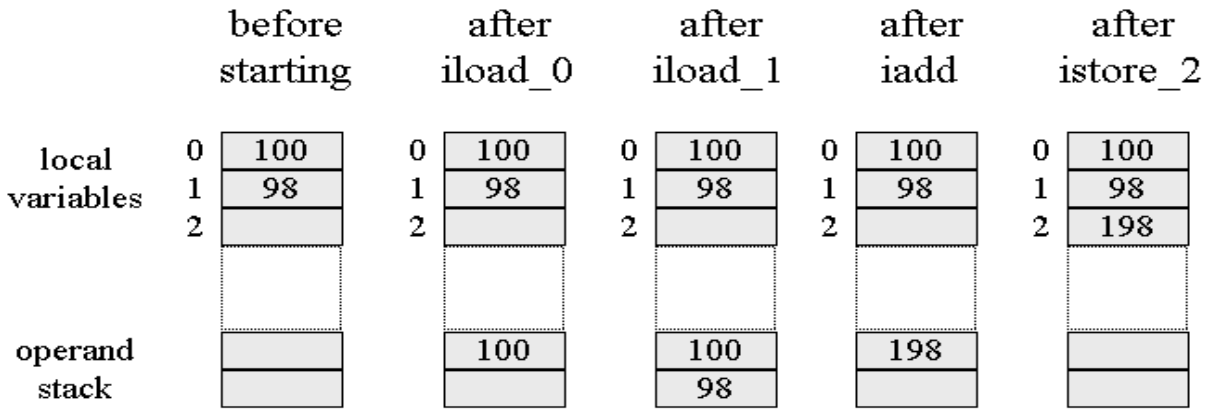
Modèle Mémoire : Run Time



Stack Frame
(frame)

… **parameter 2** **parameter 1** **Local Variables** **(vars)**
Frame data
Operand stack **(optop)**

```
iload_0      // push the int in local variable 0
iload_1      // push the int in local variable 1
iadd         // pop two ints, add them, push result
istore_2     // pop Int, store into local variable 2
```



- Les instructions de la JVM sont typées.
- Le nom de chaque opération est préfixée par une lettre indiquant le type des données qu'elle manipule :
  - 'i' : `int`
  - 'l' : `long`
  - 's' : `short`
  - 'b' : `byte`
  - 'c' : `char`
  - 'f' : `float`
  - 'd' : `double`
  - 'a' : `reference`
- Par ailleurs, les *opcodes* sont stockés sur un octet.

Lecture et écriture des variables locales

- iload, iload_<n>, lload, lload_<n>, fload, fload_<n>, dload, dload_<n>,aload, aload_<n>
  Charge une variable locale au sommet de la pile de calcul.
- istore, istore_<n>, lstore, lstore_<n>, fstore, fstore_<n>, dstore, dstore_<n>,astore, astore_<n>
  Écrit le contenu du sommet de la pile dans une variable locale.
- bipush, sipush, ldc, ldc_w, ldc2_w, aconst_null, iconst_m1, iconst_<i>, fconst_<f>, dconst_<d>
  Empile une constante au sommet de la pile.
- wide :
  Modifie le sens de l'instruction suivante : la prochaine instruction devra attendre un indice de variable locale codé sur 2 octets et non 1 seul.

Opérations arithmétiques

- Addition : iadd, ladd, fadd, dadd.
- Soustraction : isub, lsub, fsub, dsub.
- Multiplication : imul, lmul, fmul, dmul.
- Division : idiv, ldiv, fdiv, ddiv.
- Reste de la division : irem, lrem, frem, drem.
- Négation : ineg, lneg, fneg, dneg.
- Décalage : ishl, ishr, iushr, ishl, lshl, lshr, lushr.
- "Ou" sur la représentation binaire : ior, lor.
- "Et" sur la représentation binaire : iand, land.
- "Ou exclusif" sur la représentation binaire : ixor, lxor.
- Incrémentation d'une variable locale : iincr.
- Comparaison : dcmpg, dcmpl, fcmpg, fcmpl, lcmp.

- Conversions d'élargissement : i2l, i2f,i2d, l2f, l2d,f2d.
- Conversions par projection : i2b,i2c, i2s, l2i, f2i, d2i, d2f.

- Dépiler : pop, pop2.
- Dupliquer le sommet de la pile : dup, dup2, dup_x1, dup2_x1, dup_x2, dup2_x2, swap.

- Branchement conditionnel : ifeq, iflt, ifle, ifne, ifgt, ifnull, ifnonnull, if_icmpeq, if_icmpne, if_icmplt, if_icmpgt, if_icmple, if_icmpge, if_acmpne.
- Table de saut : tableswitch, lookupswitch.
- Branchement inconditionnel : goto, goto_w, jsr, jsr_w, ret.

Manipulation d'objets et de tableaux

- Création d'une nouvelle instance de classe : new.
- Création d'un nouveau tableau : newarray, anewarray, multianewarray.
- Accès aux champs d'une classe : getfield, setfield, getstatic, putstatic.
- Chargement d'un tableau sur la pile de calcul : baload, caload, saload, iaload, laload, faload, daload, aaload.
- Affectation d'une case d'un tableau : bastore, castore, sastore, iastore, lastore, fastore, dastore, aastore.
- Empile la taille d'un tableau : arraylength.
- Vérification dynamique : instanceof, checkcast.

Invocation de méthode

- Invoquer une méthode avec liaison tardive (*i.e.* en prenant en compte le type exact de l'instance considérée) : invokevirtual.
- Invoquer une méthode d'une interface dans une instance qui l'implémente : invokeinterface.
- Invoquer une initialisation d'instance, une méthode privée ou une méthode d'une classe mère : invokespecial.
- Invoquer une méthode de classse statique : invokestatic.