Bytecode

**Байт-код Java** — набор инструкций, исполняемых виртуальной машиной Java.

**Как читать .class файл**

Для начала вспомним, как создать .class файл.

javac File.java

Если открыть любой class файл в hex редакторе, то файл начинается с "магических байт" - CAFEBABE, а дальше следует полезное содержимое файла.

Для того чтобы просмотреть содержимое, можно воспользоваться стандартной утилитой

javap File

**Параметры javap:**

**javap java.lang.Object –** без параметров

**javap -p java.lang.Object** – параметр -p показывает также приватные поля и методы

**javap -v java.lang.Object** – параметр -v показывает подробную информацию (verbose), такую, как размер стека и аргументов, версии и т.д.

Что содержит .class файл

**ClassFile {**

**u4 magic;**

**u2 minor\_version;**

**u2 major\_version;**

**u2 constant\_pool\_count;**

**cp\_info constant\_pool[constant\_pool\_count-1];**

**u2 access\_flags;**

**u2 this\_class;**

**u2 super\_class;**

**u2 interfaces\_count;**

**u2 interfaces[interfaces\_count];**

**u2 fields\_count;**

**field\_info fields[fields\_count];**

**u2 methods\_count;**

**method\_info methods[methods\_count];**

**u2 attributes\_count;**

**attribute\_info attributes[attributes\_count];**

**}**

**u1, u2, and u4** - размер полей

**cp\_info, field\_info, method\_info, attribute\_info** - специальные таблицы, о которых рассказывается ниже

**magic** - магическая константа (0xCAFEBABE), который определяет формат файла.

**minor\_version, major\_version** - версия формата .class файла.

**constant\_pool\_count и constant\_pool** - длина пула констант и сам пул констант ( Пул констант - таблица для записи различный текстовых констант, имен интерфейсов, классов, полей и другие константы, на которые в дальнейшем будут ссылки в процессе выполнения, раздел Constant pool при выводе javap -v)

**access\_flags** - набор флагов (public, abstract, enum и т.д.)

**this\_class** - ссылка на пул констант, которая определяет данный класс

**super\_class** - ссылка на пул констант, которая определяет родительский класс

**interfaces\_count и interfaces** - количество интерфейсов, которые реализуют класс и ссылки на пул констант для этих интерфейсов

**fields\_count и field**s - информация по полям

**methods\_count и methods** - информация по методам

**attributes\_count и attributes** - информация по атрибутам

**access\_flags**

|  |  |  |
| --- | --- | --- |
| **Flag Name (Имя флага)** | **Value (Значение)** | **Interpretation (Интерпретация)** |
| ACC\_PUBLIC | 0x0001 | public, виден за пределами пакета |
| ACC\_FINAL | 0x0010 | final; не может быть наследован |
| ACC\_SUPER | 0x0020 | Обработка методов супер-класса когда вызывается инструкция invokespecial. |
| ACC\_INTERFACE | 0x0200 | Интерфейс, не класс |
| ACC\_ABSTRACT | 0x0400 | Абстрактный, не может быть наследован |
| ACC\_SYNTHETIC | 0x1000 | Синтетический, не из кода |
| ACC\_ANNOTATION | 0x2000 | Аннотация |
| ACC\_ENUM | 0x4000 | Enum |
| ACC\_MODULE | 0x8000 | Модуль, не класс или интерфейс |

Сигнатуры (descriptor) для полей имеют следующий формат:

* B - byte
* C - char
* D - double
* F - float
* I - int
* J - long
* S - short
* Z - boolean
* Lимя\_класса - ccылочный тип (как в примере - Ljava/lang/String)
* [ - массив, например long[] будет [J

Часть инструкций привязаны к типу, с которым они работают, так например iconst\_1 - работает с integer, lconst\_1 - с long. Соответствие префикса от типа:

* a - reference
* b - byte
* c - character
* d - double
* i - integer
* f - float
* l - long
* s - short

Вызовы методов

**invokestatic** - используется для вызова статического метода.

**invokespecial -** используется для прямого вызова методов объекта текущего класса, конструкторов и методов родительского класса.

**invokevirtual** - используется для вызова методов класса, при этом используется динамический поиск какой метод вызывать, основываясь на классе (Dynamic Dispatch). Так как методы могут быть переопределены, то сначала проверяется наличие метода в переданном классе, потом в родительском и так далее.

**invokeinterface** - используется для вызова методов интерфейса (Dynamic Dispatch). При этом первым параметром передается ссылка на объект.

**invokedynamic -** вызов динамически-вычисляемых call sites.

Справочник основных opcode

**aaload (0x32) , baload (0x33), caload (0x34), daload (0x31), iaload (0x2e), faload (0x30), laload (0x2f), saload (0x35)** - загрузка ссылки из массива (a\* - ссылок, b\* - byte или boolean, c\* - char, d\* - double, i\* - integer, f\* - float, l\* - long, s\* - short).

**aastore (0x53) , bastore (0x54), castore (0x55), dastore (0x52), iastore (0x4а), fastore (0x51), lastore (0x50), sastore (0x56) -** записать элемент в массив (a\* - ссылок, b\* - byte или boolean, c\* - char, d\* - double, i\* - integer, f\* - float, l\* - long, s\* - short).

**aconst\_null (0x1) -** положить на стек null.

**aload (0x30) , dload (0x18), iload (0x15), fload (0x17), lload (0x16)** - загрузить значение из локальной переменной под индекcом index (a\* - ссылок, d\* - double, i\* - integer, f\* - float, l\* - long).

**aload\_{x} , dload\_{x}, iload\_{x}, fload\_{x} , lload\_{x}** - загрузить значение из локальной переменной под индекcом x (a\* - ссылок, d\* - double, i\* - integer, f\* - float, l\* - long).

**anewarray (0xbd )** - создает массив объектов.

**areturn (0xb0), dreturn (0xaf), ireturn (0xac), freturn (0xae) , lreturn (0xad)** - возвращает значение со стека из метода (a\* - ссылок, d\* - double, i\* - integer, f\* - float, l\* - long).

**arraylength (0xbe)** - получить длину массива.

**astore (0x3a) , dstore (0x39), istore (0x36), fstore (0x38), lstore (0x37) -** записать значение с вершины стека в локальную переменную под индекcом index (a\* - для ссылок, d\* - double, i\* - integer, f\* - float, l\* - long).

**astore\_{x} , dstore\_{x}, istore\_{x}, fstore\_{x} , lstore\_{x}** - загрузить значение из локальной переменной под индекcом {x} (a\* - ссылок, d\* - double, i\* - integer, f\* - float, l\* - long).

**athrow (0xbf)** - Выбрасывает исключение objectref.

**bipush (0x10), sipush (0x11)** - кладет значение на стек (b\* - byte, s\* - short).

**breakpoint (0xca)** - проверяет, является ли объект переданным типом. Если нет, то выбрасывает - ClassCastException.

**d2f (0x90), d2i (0x8e), d2l (0x8f), f2d (0x8d), f2i(0x8b), f2l (0x8c), i2b (0x91), i2c(0x92), i2d(0x87), i2f(0x86), i2l(0x85), i2s(0x93), l2d(0x8a), l2f(0x89), l2i(0x88) -** конвертация значений (d - double, i - int, f - float, s - short, c - char, l - long).

**dadd (0x63), fadd(0x62), iadd(0x60), ladd(0x61), ddiv(0x6f), dmul(0x6b), drem (0x73), dsub (0x67), idiv(0x6c), imul (0x68), irem (0x70), isub (0x64), fdiv(0x6e), fmul (0x6a), frem (0x72), fsub (0x66), ldiv(0x6d), lmul (0x69), lrem (0x71), lsub (0x65) - \*add -** cкладывает два значения, \*div - делит два значения, \*mul - умножает два значения, \*rem - остаток от деления, \*sub – вычитание. (d - double, i - int, f - float, l - long).

**dcmpg (0x98), dcmpl(0x97), fcmpg(0x96), fcmpl(0x95), lcmp(0x94)** - сравнивают 2 значения. Если первое больше, то на стек кладется 1, если равны - 0, если первое меньше, то кладется -1. (\*cmpg и \*cmpl различаются в работе с NaN, первое кладет 1, второе - "-1"), (d - double, f - float, l - long).

**dconst\_0 (0xe),  dconst\_1(0xf), lconst\_0(0x9), lconst\_1(0xa), fconst\_0(0xb), fconst\_1(0xc), fconst\_2(0xd), iconst\_m1 (0x3), iconst\_1(0x4), iconst\_2 (0x5), iconst\_3 (0x6), iconst\_4 (0x7), iconst\_5 (0x8)** - кладет значение на стек (\*\_0 -> 0, \*\_1 -> 1, \*\_2 -> 2, \*\_3 -> 3, \*\_4 -> 4, \*\_5 -> 5, \*\_m1 -> -1), (d\* - double, i\* - int, f\* - float, l\* - long).

**dneg (0x77),  ineg (0x74), fneg(0x76), lneg(0x75)** - возвращает обратное значение. (d - double, i - int, f - float, l - long).

**dup(0x59) -** дублирует верх стека.

**dup\_x1(0x5a) -** дублирует верх стека и вставляет его на 1 или 2 значения ниже.

**dup\_x2(0x5b) -** дублирует верх стека и вставляет его на 2 или 3 значения ниже.

**dup2 (0x5c) –** дублирует 1 или 2 значения с верха стека.

**dup2\_x1(0x5d)** - дублирует 1 или 2 значения с вершины стека и вставляет его на 2 или 3 значения ниже.

**dup2\_x2 (0x5e) -** дублирует 1 или 2 значения с вершины стека и вставляет его на 2, 3 или 4 значения ниже.

**getfield (0xb4) -** положить на стек значение из поля.

**getstatic (0xb2)** - положить на стек значение из статического поля.

**goto (0xa7), goto\_w (0xc8)** – прыгнуть.

**iand (0x7e), land (0x7f), ior (0x80), lor(0x81), ixor(0x82), lxor(0x83) -** выполнить для \*and - &, для \*or - |, для \*xor - логическое XOR (l\* - для long, i\* - для int).

**if\_acmpeq (0xa5), if\_acmpne(0xa6)** - прыгнуть, если ссылки ссылаются на один и тот же объект (для if\_acmpeq), не ссылаются на один и тот же объект (для if\_acmpne).

**if\_icmpeq (0x9f), if\_icmpne(0xa0), if\_icmplt(0xa1), if\_icmpge(0xa2), if\_icmpgt(0xa3), if\_icmple(0xa4)** - прыгнуть, если (value1 и value2 - int):

* if\_icmpeq - если value1 == value2
* if\_icmpne - если value1 ≠ value2
* if\_icmplt - если value1 < value2
* if\_icmple - если value1 ≤ value2
* if\_icmpgt - если value1 > value2
* if\_icmpge - если value1 ≥ value2

**ifeq (0x99), ifne(0x9a), iflt(0x9b), ifge(0x9c), ifgt(0x9d), ifle(0x9e)** - прыгнуть, если (value- int):

* ifeq - если value == 0
* ifne - если value ≠ 0
* iflt - если value < 0
* ifle - если value ≤ 0
* ifgt - если value > 0
* ifge - если value ≥ 0

**ifnonnull (0xc7), ifnull(0xc6)** - прыгнуть, если (value- объект):

* ifnull - если value == null
* ifnonnull - если value ≠ null

**iinc (0x84) -** увеличить локальную переменную (int) на константу.

**impdep1 (0xfe), impdep2(0xff)** - кладет на стек 1, если объект данного типа, если нет - то кладет 0.

**iushr (0x7c), lushr(0x7d) -** логический сдвиг вправо. (i\* - int, l\* - long).

**ishl (0x78), lshl(0x79), ishr(0x7a), lshr(0x7b)** - сдвиг влево - для \*shl , вправо - для \*shr. (i\* - int, l\* - long).

**jsr (0xa8), jsr\_w(0xc9)** – для class файла версии 51 или старше не используются. Использовались для реализации try-finally.

**ldc(0x12), ldc\_w(0x13), ldc2\_w(0x14) –** К=кладет константу на стек. ldc\_w - использует широкий индекс, ldc2\_w используется для long и double.

**lookupswitch(0xab)** - используется для перехода к строчке в соответствии с таблицей соответствия. См. реализацию switch.

**monitorenter(0xc2), monitorexit(0xc3) -** взятие и отпуск монитора.

**multianewarray(0xc5)** - создает многоразмерный массив.

**new (0xbb)** - создает новый объект, не вызывая его конструктор.

**newarray (0xbc)** - создает новый массив.

**nop (0x0) -** не делает ничего.

**pop (0x57)** - выбрасывает 1 элемент с вершины стека.

**pop2(0x58)** - выбрасывает 1 или 2 элемента с вершины стека.

**putfield (0xb5 ) -** вставить в поле объекта значение с верха стека.

**putstatic (0xb3 )** - вставить в статическое поле значение с верха стека.

**ret (0xa9) -** возвращает void из метода.

**swap (0xb1)** - меняет местами два верхних элемента стека.

**tableswitch (0xaa) -** одна из реализаций switch.

**wide (0xc4)** - расширяет индекс локальной переменной дополнительными байтами.