org.objectweb.asm

Interface MethodVisitor

All Known Implementing Classes:

AdviceAdapter, AnalyzerAdapter, ASMifierMethodVisitor, CheckMethodAdapter, EmptyVisitor, GeneratorAdapter, LocalVariablesSorter, MethodAdapter, MethodNode, SAXCodeAdapter, TraceMethodVisitor

---

`public interface` **`MethodVisitor`**

A visitor to visit a Java method. The methods of this interface must be called in the following order: [`visitAnnotationDefault`] (`visitAnnotation` | `visitParameterAnnotation` | `visitAttribute`)* [`visitCode` (`visitFrame` | `visit`*X*Insn | `visitLabel` | `visitTryCatchBlock` | `visitLocalVariable` | `visitLineNumber`)* `visitMaxs`] `visitEnd`. In addition, the `visit`*X*Insn and `visitLabel` methods must be called in the sequential order of the bytecode instructions of the visited code, `visitTryCatchBlock` must be called *before* the labels passed as arguments have been visited, and the `visitLocalVariable` and `visitLineNumber` methods must be called *after* the labels passed as arguments have been visited.

Author:

Eric Bruneton

Method Summary

`AnnotationVisitor`	**`visitAnnotation`**`(String desc, boolean visible)` Visits an annotation of this method.
`AnnotationVisitor`	**`visitAnnotationDefault`**`()` Visits the default value of this annotation interface method.
`void`	**`visitAttribute`**`(Attribute attr)` Visits a non standard attribute of this method.
`void`	**`visitCode`**`()` Starts the visit of the method's code, if any (i.e. non abstract method).
`void`	**`visitEnd`**`()` Visits the end of the method.
`void`	**`visitFieldInsn`**`(int opcode, String owner, String name, String desc)` Visits a field instruction.
`void`	**`visitFrame`**`(int type, int nLocal, Object[] local, int nStack, Object[] stack)` Visits the current state of the local variables and operand stack elements.
`void`	**`visitIincInsn`**`(int var, int increment)` Visits an IINC instruction.
`void`	**`visitInsn`**`(int opcode)` Visits a zero operand instruction.

`void`	`visitIntInsn(int opcode, int operand)` Visits an instruction with a single int operand.
`void`	`visitJumpInsn(int opcode, Label label)` Visits a jump instruction.
`void`	`visitLabel(Label label)` Visits a label.
`void`	`visitLdcInsn(Object cst)` Visits a LDC instruction.
`void`	`visitLineNumber(int line, Label start)` Visits a line number declaration.
`void`	`visitLocalVariable(String name, String desc, String signature, Label start, Label end, int index)` Visits a local variable declaration.
`void`	`visitLookupSwitchInsn(Label dflt, int[] keys, Label[] labels)` Visits a LOOKUPSWITCH instruction.
`void`	`visitMaxs(int maxStack, int maxLocals)` Visits the maximum stack size and the maximum number of local variables of the method.
`void`	`visitMethodInsn(int opcode, String owner, String name, String desc)` Visits a method instruction.
`void`	`visitMultiANewArrayInsn(String desc, int dims)` Visits a MULTIANEWARRAY instruction.
`AnnotationVisitor`	`visitParameterAnnotation(int parameter, String desc, boolean visible)` Visits an annotation of a parameter this method.
`void`	`visitTableSwitchInsn(int min, int max, Label dflt, Label[] labels)` Visits a TABLESWITCH instruction.
`void`	`visitTryCatchBlock(Label start, Label end, Label handler, String type)` Visits a try catch block.
`void`	`visitTypeInsn(int opcode, String desc)` Visits a type instruction.
`void`	`visitVarInsn(int opcode, int var)` Visits a local variable instruction.

Method Detail

visitAnnotationDefault

`AnnotationVisitor` **`visitAnnotationDefault`**`()`

Visits the default value of this annotation interface method.

Returns:

a non null visitor to the visit the actual default value of this annotation interface method. The 'name' parameters passed to the methods of this annotation visitor are ignored. Moreover, exacly one visit method must be called on this annotation visitor, followed by visitEnd.

visitAnnotation

```
AnnotationVisitor visitAnnotation(String desc,
                                  boolean visible)
```

Visits an annotation of this method.

Parameters:
`desc` - the class descriptor of the annotation class.
`visible` - `true` if the annotation is visible at runtime.

Returns:
a non null visitor to visit the annotation values.

visitParameterAnnotation

```
AnnotationVisitor visitParameterAnnotation(int parameter,
                                           String desc,
                                           boolean visible)
```

Visits an annotation of a parameter this method.

Parameters:
`parameter` - the parameter index.
`desc` - the class descriptor of the annotation class.
`visible` - `true` if the annotation is visible at runtime.

Returns:
a non null visitor to visit the annotation values.

visitAttribute

```
void visitAttribute(Attribute attr)
```

Visits a non standard attribute of this method.

Parameters:
`attr` - an attribute.

visitCode

```
void visitCode()
```

Starts the visit of the method's code, if any (i.e. non abstract method).

visitFrame

```
void visitFrame(int type,
                int nLocal,
                Object[] local,
                int nStack,
                Object[] stack)
```

Visits the current state of the local variables and operand stack elements. This method must(*) be called *just before* any instruction **i** that follows an unconditionnal branch instruction such as GOTO or THROW, that is the target of a jump instruction, or that starts an exception handler block. The visited types must describe the values of the local variables and of the operand stack elements *just before* **i** is executed.

(*) this is mandatory only for classes whose version is greater than or equal to `V1_6`.

Packed frames are basically "deltas" from the state of the previous frame (very first frame is implicitly defined by the method's parameters and access flags):

- `Opcodes.F_SAME` representing frame with exactly the same locals as the previous frame and with the empty stack.
- `Opcodes.F_SAME1` representing frame with exactly the same locals as the previous frame and with single value on the stack (`nStack` is 1 and `stack[0]` contains value for the type of the stack item).
- `Opcodes.F_APPEND` representing frame with current locals are the same as the locals in the previous frame, except that additional locals are defined (`nLocal` is 1, 2 or 3 and `local` elements contains values representing added types).
- `Opcodes.F_CHOP` representing frame with current locals are the same as the locals in the previous frame, except that the last 1-3 locals are absent and with the empty stack (`nLocals` is 1, 2 or 3).
- `Opcodes.F_FULL` representing complete frame data.

Parameters:

`type` - the type of this stack map frame. Must be `Opcodes.F_NEW` for expanded frames, or `Opcodes.F_FULL`, `Opcodes.F_APPEND`, `Opcodes.F_CHOP`, `Opcodes.F_SAME` or `Opcodes.F_APPEND`, `Opcodes.F_SAME1` for compressed frames.
`nLocal` - the number of local variables in the visited frame.
`local` - the local variable types in this frame. This array must not be modified. Primitive types are represented by `Opcodes.TOP`, `Opcodes.INTEGER`, `Opcodes.FLOAT`, `Opcodes.LONG`, `Opcodes.DOUBLE`,`Opcodes.NULL` or `Opcodes.UNINITIALIZED_THIS` (long and double are represented by a single element). Reference types are represented by String objects, and uninitialized types by Label objects (this label designates the NEW instruction that created this uninitialized value).
`nStack` - the number of operand stack elements in the visited frame.
`stack` - the operand stack types in this frame. This array must not be modified. Its content has the same format as the "local" array.

visitInsn

```
void visitInsn(int opcode)
```

Visits a zero operand instruction.

Parameters:

`opcode` - the opcode of the instruction to be visited. This opcode is either NOP, 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, IALOAD, LALOAD, FALOAD, DALOAD, AALOAD, BALOAD, CALOAD, SALOAD, IASTORE, LASTORE, FASTORE, DASTORE, AASTORE, BASTORE, CASTORE, SASTORE, POP, POP2, DUP, DUP_X1, DUP_X2, DUP2, DUP2_X1, DUP2_X2, SWAP, IADD, LADD, FADD, DADD, ISUB, LSUB, FSUB, DSUB, IMUL, LMUL, FMUL, DMUL, IDIV, LDIV, FDIV, DDIV, IREM, LREM, FREM, DREM, INEG, LNEG, FNEG, DNEG, ISHL, LSHL, ISHR, LSHR, IUSHR, LUSHR, IAND, LAND, IOR, LOR, IXOR, LXOR, I2L, I2F, I2D, L2I, L2F, L2D, F2I, F2L, F2D, D2I, D2L, D2F, I2B, I2C, I2S, LCMP, FCMPL, FCMPG, DCMPL, DCMPG, IRETURN, LRETURN, FRETURN, DRETURN, ARETURN, RETURN, ARRAYLENGTH, ATHROW, MONITORENTER, or MONITOREXIT.

---

visitIntInsn

```
void visitIntInsn(int opcode,
                  int operand)
```

Visits an instruction with a single int operand.

Parameters:

`opcode` - the opcode of the instruction to be visited. This opcode is either BIPUSH, SIPUSH or NEWARRAY.
`operand` - the operand of the instruction to be visited.
When opcode is BIPUSH, operand value should be between Byte.MIN_VALUE and Byte.MAX_VALUE.
When opcode is SIPUSH, operand value should be between Short.MIN_VALUE and Short.MAX_VALUE.
When opcode is NEWARRAY, operand value should be one of `Opcodes.T_BOOLEAN`, `Opcodes.T_CHAR`, `Opcodes.T_FLOAT`, `Opcodes.T_DOUBLE`, `Opcodes.T_BYTE`, `Opcodes.T_SHORT`, `Opcodes.T_INT` or `Opcodes.T_LONG`.

---

visitVarInsn

```
void visitVarInsn(int opcode,
                  int var)
```

Visits a local variable instruction. A local variable instruction is an instruction that loads or stores the value of a local variable.

Parameters:

`opcode` - the opcode of the local variable instruction to be visited. This opcode is either ILOAD, LLOAD, FLOAD, DLOAD, ALOAD, ISTORE, LSTORE, FSTORE, DSTORE, ASTORE or RET.
`var` - the operand of the instruction to be visited. This operand is the index of a local variable.

---

visitTypeInsn

```
void visitTypeInsn(int opcode,
                   String desc)
```

Visits a type instruction. A type instruction is an instruction that takes a type descriptor as parameter.

Parameters:

`opcode` - the opcode of the type instruction to be visited. This opcode is either NEW, ANEWARRAY, CHECKCAST or INSTANCEOF.
`desc` - the operand of the instruction to be visited. This operand is must be a fully qualified class name in internal form, or the type descriptor of an array type (see `Type`).

---

visitFieldInsn

```
void visitFieldInsn(int opcode,
                    String owner,
                    String name,
                    String desc)
```

Visits a field instruction. A field instruction is an instruction that loads or stores the value of a field of an object.

Parameters:

`opcode` - the opcode of the type instruction to be visited. This opcode is either GETSTATIC, PUTSTATIC, GETFIELD or PUTFIELD.
`owner` - the internal name of the field's owner class (see `getInternalName`).
`name` - the field's name.
`desc` - the field's descriptor (see `Type`).

---

visitMethodInsn

```
void visitMethodInsn(int opcode,
                     String owner,
                     String name,
                     String desc)
```

Visits a method instruction. A method instruction is an instruction that invokes a method.

Parameters:

`opcode` - the opcode of the type instruction to be visited. This opcode is either INVOKEVIRTUAL, INVOKESPECIAL, INVOKESTATIC or INVOKEINTERFACE.
`owner` - the internal name of the method's owner class (see `getInternalName`).
`name` - the method's name.
`desc` - the method's descriptor (see `Type`).

---

visitJumpInsn

```
void visitJumpInsn(int opcode,
                   Label label)
```

Visits a jump instruction. A jump instruction is an instruction that may jump to another instruction.

Parameters:

`opcode` - the opcode of the type instruction to be visited. This opcode is either IFEQ, IFNE, IFLT, IFGE, IFGT, IFLE, IF_ICMPEQ, IF_ICMPNE, IF_ICMPLT, IF_ICMPGE, IF_ICMPGT, IF_ICMPLE, IF_ACMPEQ, IF_ACMPNE, GOTO, JSR, IFNULL or IFNONNULL.
`label` - the operand of the instruction to be visited. This operand is a label that designates the instruction to which the jump instruction may jump.

---

visitLabel

```
void visitLabel(Label label)
```

Visits a label. A label designates the instruction that will be visited just after it.

Parameters:

`label` - a `Label` object.

---

visitLdcInsn

```
void visitLdcInsn(Object cst)
```

Visits a LDC instruction.

Parameters:

`cst` - the constant to be loaded on the stack. This parameter must be a non null `Integer`, a `Float`, a `Long`, a `Double` a `String` (or a `Type` for `.class` constants, for classes whose version is 49.0 or more).

---

visitIincInsn

```
void visitIincInsn(int var,
                   int increment)
```

Visits an IINC instruction.

Parameters:

`var` - index of the local variable to be incremented.
`increment` - amount to increment the local variable by.

---

visitTableSwitchInsn

```
void visitTableSwitchInsn(int min,
                          int max,
                          Label dflt,
                          Label[] labels)
```

Visits a TABLESWITCH instruction.

Parameters:

`min` - the minimum key value.
`max` - the maximum key value.
`dflt` - beginning of the default handler block.
`labels` - beginnings of the handler blocks. `labels[i]` is the beginning of the handler block for the `min + i` key.

---

visitLookupSwitchInsn

```
void visitLookupSwitchInsn(Label dflt,
                           int[] keys,
                           Label[] labels)
```

Visits a LOOKUPSWITCH instruction.

Parameters:

`dflt` - beginning of the default handler block.
`keys` - the values of the keys.
`labels` - beginnings of the handler blocks. `labels[i]` is the beginning of the handler block for the `keys[i]` key.

---

visitMultiANewArrayInsn

```
void visitMultiANewArrayInsn(String desc,
                             int dims)
```

Visits a MULTIANEWARRAY instruction.

Parameters:

`desc` - an array type descriptor (see `Type`).
`dims` - number of dimensions of the array to allocate.

---

visitTryCatchBlock

```
void visitTryCatchBlock(Label start,
                        Label end,
                        Label handler,
                        String type)
```

Visits a try catch block.

Parameters:

`start` - beginning of the exception handler's scope (inclusive).
`end` - end of the exception handler's scope (exclusive).
`handler` - beginning of the exception handler's code.
`type` - internal name of the type of exceptions handled by the handler, or `null` to catch any exceptions (for "finally" blocks).

Throws:

`IllegalArgumentException` - if one of the labels has already been visited by this visitor (by the `visitLabel` method).

---

visitLocalVariable

```
void visitLocalVariable(String name,
                        String desc,
                        String signature,
                        Label start,
                        Label end,
                        int index)
```

Visits a local variable declaration.

Parameters:

`name` - the name of a local variable.
`desc` - the type descriptor of this local variable.
`signature` - the type signature of this local variable. May be `null` if the local variable type does not use generic types.
`start` - the first instruction corresponding to the scope of this local variable (inclusive).
`end` - the last instruction corresponding to the scope of this local variable (exclusive).
`index` - the local variable's index.

Throws:

`IllegalArgumentException` - if one of the labels has not already been visited by this visitor (by the `visitLabel` method).

---

visitLineNumber

```
void visitLineNumber(int line,
                     Label start)
```

Visits a line number declaration.

Parameters:

`line` - a line number. This number refers to the source file from which the class was compiled.

`start` - the first instruction corresponding to this line number.

Throws:

`IllegalArgumentException` - if `start` has not already been visited by this visitor (by the `visitLabel` method).

---

visitMaxs

```
void visitMaxs(int maxStack,
               int maxLocals)
```

Visits the maximum stack size and the maximum number of local variables of the method.

Parameters:

`maxStack` - maximum stack size of the method.

`maxLocals` - maximum number of local variables for the method.

---

visitEnd

```
void visitEnd()
```

Visits the end of the method. This method, which is the last one to be called, is used to inform the visitor that all the annotations and attributes of the method have been visited.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
