JavaScript is disabled on your browser.

jminusminus

Class CLEmitter

- java.lang.Object
  - jminusminus.CLEmitter

.

```
public class CLEmitter
extends Object
```

This class provides a high level interface for creating (in-memory and file based) representation of Java classes. j-- uses this interface to produce target JVM bytecode from a j-- source program. During the pre-analysis and analysis phases, j-- produces partial (in-memory) classes for the type declarations within the compilation unit, and during the code generation phase, it produces file-based classes for the declarations.

- **Constructor Summary**

Constructors

Constructor and Description
CLEmitter(boolean toFile) Construct a CLEmitter instance.

- **Method Summary**

Methods

Modifier and Type	Method and Description
void	**addArrayInstruction**(int opcode, String type) Add an array instruction.
void	**addBranchInstruction**(int opcode,

	String label) Add a branch instruction.
void	addClass(ArrayList<String> accessFlags, String thisClass, String superClass, ArrayList<String> superInterfaces, boolean isSynthetic) Add a class or interface.
void	addClassAttribute(CLAttributeInfo attribute) Add the specified class attribute to the attribyte section of the class.
void	addCodeAttribute(CLAttributeInfo attribute) Add the specified code attribute to the attribute section of the code for the method last added.
void	addExceptionHandler(String startLabel, String endLabel, String handlerLabel, String catchType) Add an exception handler.
void	addField(ArrayList<String> accessFlags, String name, boolean isSynthetic, double d) Add a double field with initialization.
void	addField(ArrayList<String> accessFlags, String name, boolean isSynthetic, float f) Add a float field with initialization.
void	addField(ArrayList<String> accessFlags, String name, boolean isSynthetic, long l) Add a long field with initialization.
void	addField(ArrayList<String> accessFlags, String name, boolean isSynthetic, String s) Add a String type field with initialization.
void	addField(ArrayList<String> accessFlags, String name, String type, boolean isSynthetic) Add a field without initialization.
void	addField(ArrayList<String> accessFlags, String name, String type, boolean isSynthetic, int i) Add an int, short, char, byte, or boolean field with initialization.
void	addFieldAttribute(CLAttributeInfo attribute) Add the specified field attribute the attribute section of the field last added.
void	addIINCInstruction(int index, int constVal) Add an IINC instruction to increment a variable by a constant.
void	addInnerClass(ArrayList<String> accessFlags, String innerClass, String outerClass, String innerName) Add an inner class.

void	**addLabel**(String label) Add a jump label to the code section of the method being added.
void	**addLDCInstruction**(double d) Add an LDC instruction to load a double constant on the operand stack.
void	**addLDCInstruction**(float f) Add an LDC instruction to load a float constant on the operand stack.
void	**addLDCInstruction**(int i) Add an LDC instruction to load an int constant on the operand stack.
void	**addLDCInstruction**(long l) Add an LDC instruction to load a long constant on the operand stack.
void	**addLDCInstruction**(String s) Add an LDC instruction to load a String constant on the operand stack.
void	**addLOOKUPSWITCHInstruction**(String defaultLabel, int numPairs, TreeMap<Integer,String> matchLabelPairs) Add a LOOKUPSWITCH instruction -- used for switch statements.
void	**addMemberAccessInstruction**(int opcode, String target, String name, String type) Add a member (field & method) access instruction.
void	**addMethod**(ArrayList<String> accessFlags, String name, String descriptor, ArrayList<String> exceptions, boolean isSynthetic) Add a method.
void	**addMethodAttribute**(CLAttributeInfo attribute) Add the specified method attribute to the attribute section of the method last added.
void	**addMULTIANEWARRAYInstruction**(String type, int dim) Add a MULTIANEWARRAY instruction for creating multi-dimensional arrays.
void	**addNoArgInstruction**(int opcode) Add a no argument instruction.
void	**addOneArgInstruction**(int opcode, int arg) Add a one argument instruction.
void	**addReferenceInstruction**(int opcode, String type) Add a reference (object) instruction.
void	**addTABLESWITCHInstruction**(String defaultLabel, int low, int high, ArrayList<String> labels)

	Add a TABLESWITCH instruction -- used for switch statements.
CLFile	**clFile**() Return the CLFile instance corresponding to the class built by this emitter.
CLConstantPool	**constantPool**() Return the constant pool of the class being built.
String	**createLabel**() Construct and return a unique jump label.
void	**destinationDir**(String destDir) Set the destination directory for the class file to the specified value.
boolean	**errorHasOccurred**() Has an emitter error occurred up to now?
static void	**initializeByteClassLoader**() Set a new ByteClassLoader for loading classes from byte streams.
int	**pc**() Return the pc (location counter).
Class	**toClass**() Return the class being constructed as a Java Class instance.
void	**write**() Write out the class to the file system as a .class file if toFile is true.

- **Methods inherited from class java.lang.Object**
  clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- **Constructor Detail**

  - **CLEmitter**

  publicCLEmitter(booleantoFile)

  Construct a CLEmitter instance.

  Parameters:

  toFile - if true, the in-memory representation of the class file will be written to the file system. Otherwise, it won't be saved as a file.

- **Method Detail**

  - **destinationDir**

  publicvoiddestinationDir(StringdestDir)

  Set the destination directory for the class file to the specified value.

  Parameters:

  destDir - destination directory.

- **errorHasOccurred**

```
publicbooleanerrorHasOccurred()
```

Has an emitter error occurred up to now?

Returns:

true or false.

- **addClass**

```
publicvoidaddClass(ArrayList<String>accessFlags,
           StringthisClass,
           StringsuperClass,
           ArrayList<String>superInterfaces,
           booleanisSynthetic)
```

Add a class or interface. This method instantiates a class file representation in memory, so must be called prior to methods that add information (fields, methods, instructions, etc.) to the class.

Parameters:

`accessFlags` - the access flags for the class or interface.
`thisClass` - fully qualified name of the class or interface in internal form.
`superClass` - fully qualified name of the parent class in internal form.
`superInterfaces` - list of direct super interfaces of this class or interface as fully qualified names in internal form.
`isSynthetic` - whether the class or interface is synthetic.

- **addInnerClass**

```
publicvoidaddInnerClass(ArrayList<String>accessFlags,
                Stringinnerclass,
                StringouterClass,
                StringinnerName)
```


Add an inner class. Note that this only registers the inner class with its parent and does not create the class.

Parameters:

`accessFlags` - access flags for the inner class.
`innerClass` - fully qualified name of the inner class in internal form.
`outerClass` - fully qualified name of the outer class in internal form.
`innerName` - simple name of the inner class.

- **addField**

```
publicvoidaddField(ArrayList<String>accessFlags,
           Stringname,
           Stringtype,
           booleanisSynthetic)
```

Add a field without initialization.

Parameters:

`accessFlags` - access flags for the field.
`name` - name of the field.
`type` - type descriptor of the field.
`isSynthetic` - is this a synthetic field?

- **addField**

```
publicvoidaddField(ArrayList<String>accessFlags,
           Stringname,
           Stringtype,
           booleanisSynthetic,
```

```
            inti)
```

Add an int, short, char, byte, or boolean field with initialization. If the field is final, the initialization is added to the constant pool. The initializations are all stored as ints, where boolean true and false are 1 and 0 respectively, and short, char, and byte must be cast to int.

Parameters:

`accessFlags` - access flags for the field.
`name` - name of the field.
`type` - type descriptor of the field.
`isSynthetic` - is this a synthetic field?
`i` - int value.

- **addField**

```
publicvoidaddField(ArrayList<String>accessFlags,
            Stringname,
            booleanisSynthetic,
            floatf)
```

Add a float field with initialization. If the field is final, the initialization is added to the constant pool.

Parameters:

`accessFlags` - access flags for the field.
`name` - name of the field.
`isSynthetic` - is this a synthetic field?
`f` - float value.

- **addField**

```
publicvoidaddField(ArrayList<String>accessFlags,
            Stringname,
            booleanisSynthetic,
            longl)
```

Add a long field with initialization. If the field is final, the initialization is added to the constant pool.

Parameters:

`accessFlags` - access flags for the field.
`name` - name of the field.
`isSynthetic` - is this a synthetic field?
`l` - long value.

- **addField**

```
publicvoidaddField(ArrayList<String>accessFlags,
            Stringname,
            booleanisSynthetic,
            doubled)
```

Add a double field with initialization. If the field is final, the initialization is added to the constant pool.

Parameters:

`accessFlags` - access flags for the field.
`name` - name of the field.
`isSynthetic` - is this a synthetic field?
`d` - double value.

- **addField**

```
publicvoidaddField(ArrayList<String>accessFlags,
            Stringname,
```

```
            booleanisSynthetic,
            Strings)
```

Add a String type field with initialization. If the field is final, the initialization is added to the constant pool.

Parameters:

`accessFlags` - access flags for the field.
`name` - name of the field.
`isSynthetic` - is this a synthetic field?
`s` - String value.

- **addMethod**

```
publicvoidaddMethod(ArrayList<String>accessFlags,
            Stringname,
            Stringdescriptor,
            ArrayList<String>exceptions,
            booleanisSynthetic)
```

Add a method. Instructions can subsequently be added to this method using the appropriate methods for adding instructions.

Parameters:

`accessFlags` - access flags for the method.
`name` - name of the method.
`descriptor` - descriptor specifying the return type and the types of the formal parameters of the method.
`exceptions` - exceptions thrown by the method, each being a name in fully qualified internal form.
`isSynthetic` - whether this is a synthetic method?

- **addExceptionHandler**

```
publicvoidaddExceptionHandler(StringstartLabel,
                  StringendLabel,
                  StringhandlerLabel,
                  StringcatchType)
```


Add an exception handler.

Parameters:

`startLabel` - the exception handler is active from the instruction following this label in the code section of the current method being added ...
`endLabel` - to the instruction following this label. Formally, the handler is active while the program counter is within the interval [startLabel, endLabel).
`handlerLabel` - the handler begins with instruction following this label.
`catchType` - the exception type that this exception handler is designated to catch, as a fully qualified name in internal form. If null, this exception handler is called for all exceptions; this is used to immplement "finally".

- **addNoArgInstruction**

```
publicvoidaddNoArgInstruction(intopcode)
```

Add a no argument instruction. Following instructions can be added using this method:

Arithmetic Instructions:

`IADD, LADD, FADD, DADD, ISUB, LSUB, FSUB, DSUB, IMUL,`

LMUL, FMUL, DMUL, IDIV, LDIV, FDIV, DDIV, IREM, LREM, FREM, DREM, INEG, LNEG, FNEG, DNEG

Array Instructions:

IALOAD, LALOAD, FALOAD, DALOAD, AALOAD, BALOAD, CALOAD, SALOAD, IASTORE, LASTORE, FASTORE, DASTORE, AASTORE, BASTORE, CASTORE, SASTORE, ARRAYLENGTH

Bit Instructions:

ISHL, ISHR, IUSHR, LSHL, LSHR, LUSHR, IOR, LOR, IAND, LAND, IXOR, LXOR

Comparison Instructions:

DCMPG, DCMPL, FCMPG, FCMPL, LCMP

Conversion Instructions:

I2B, I2C, I2S, I2L, I2F, I2D, L2F, L2D, L2I, F2D, F2I, F2L, D2I, D2L, D2F

Load Store Instructions:

ILOAD_0, ILOAD_1, ILOAD_2, ILOAD_3, LLOAD_0, LLOAD_1, LLOAD_2, LLOAD_3, FLOAD_0, FLOAD_1, FLOAD_2, FLOAD_3, DLOAD_0, DLOAD_1, DLOAD_2, DLOAD_3, ALOAD_0, ALOAD_1, ALOAD_2, ALOAD_3, ISTORE_0, ISTORE_1, ISTORE_2, ISTORE_3, LSTORE_0, LSTORE_1, LSTORE_2, LSTORE_3, FSTORE_0, FSTORE_1, FSTORE_2, FSTORE_3, DSTORE_0, DSTORE_1, DSTORE_2, DSTORE_3, ASTORE_0, ASTORE_1, ASTORE_2, ASTORE_3, ICONST_0, ICONST_1, ICONST_2, ICONST_3, ICONST_4, ICONST_5, ICONST_M1, LCONST_0, LCONST_1, FCONST_0, FCONST_1, FCONST_2, DCONST_0, DCONST_1, ACONST_NULL, WIDE (added automatically where necesary)

Method Instructions:

IRETURN, LRETURN, FRETURN, DRETURN, ARETURN, RETURN

Stack Instructions:

POP, POP2, DUP, DUP_X1, DUP_X2, DUP2, DUP2_X1, DUP2_X2, SWAP

Miscellaneous Instructions:

NOP, ATHROW, MONITORENTER, MONITOREXIT

The opcodes for instructions are defined in CLConstants class.

Parameters:

opcode - opcode of the instruction.

- **addOneArgInstruction**

```
publicvoidaddOneArgInstruction(intopcode,
                               intarg)
```

Add a one argument instruction. Wideable instructions are widened if necessary by adding a WIDE instruction before the instruction. Following instructions can be added using this method:

Load Store Instructions:

```
ILOAD, LLOAD, FLOAD, DLOAD, ALOAD, ISTORE, LSTORE, FSTORE,
DSTORE, ASTORE, BIPUSH, SIPUSH
```

Flow Control Instructions:

```
RET
```

The opcodes for instructions are defined in CLConstants class.

Parameters:

`opcode` - opcode of the instruction.
`arg` - the argument. For the instructions that deal with local variables, the argument is the local variable index; for BIPUSH and SIPUSH instructions, the argument is the constant byte or short value.

- **addIINCInstruction**

```
publicvoidaddIINCInstruction(intindex,
                    intconstVal)
```

Add an IINC instruction to increment a variable by a constant. The instruction is widened if necessary by adding a WIDE instruction before the instruction.

Parameters:

`index` - local variable index.
`constVal` - increment value.

- **addMemberAccessInstruction**

```
publicvoidaddMemberAccessInstruction(intopcode,
                                     Stringtarget,
                                     Stringname,
                                     Stringtype)
```


Add a member (field & method) access instruction. Following instructions can be added using this method:

Field Instructions:

```
GETSTATIC, PUTSTATIC, GETFIELD, PUTFIELD
```

Method Instructions:

```
INVOKEVIRTUAL, INVOKESPECIAL, INVOKESTATIC, INVOKEINTERFACE,
INVOKEDYNAMIC
```

The opcodes for instructions are defined in CLConstants class.

Parameters:

`opcode` - opcode of the instruction.
`target` - fully qualified name in internal form of the class to which the member belongs.
`name` - name of the member.
`type` - type descriptor of the member.

- **addReferenceInstruction**

```
publicvoidaddReferenceInstruction(intopcode,
                                  Stringtype)
```

Add a reference (object) instruction. Following instructions can be added using this method:

```
NEW, CHECKCAST, INSTANCEOF
```

The opcodes for instructions are defined in CLConstants class.

Parameters:

`opcode` - opcode of the instruction.
`type` - reference type in internal form.

- **addArrayInstruction**

```
publicvoidaddArrayInstruction(intopcode,
                              Stringtype)
```

Add an array instruction. Following instructions can be added using this method:

```
NEWARRAY, ANEWARRAY
```

The opcodes for instructions are defined in CLConstants class.

Parameters:

`opcode` - opcode of the instruction.
`type` - array type. In case of NEWARRAY, the primitive types are specified as: "Z" for boolean, "C" for char, "F" for float, "D" for double, "B" for byte, "S" for short, "I" for int, "J" for long. In case of ANEWARRAY, reference types are specified in internal form.

- **addMULTIANEWARRAYInstruction**

```
publicvoidaddMULTIANEWARRAYInstruction(Stringtype,
                                       intdim)
```

Add a MULTIANEWARRAY instruction for creating multi-dimensional arrays.

Parameters:

`type` - array type in internal form.
`dim` - number of dimensions.

- **addBranchInstruction**

```
publicvoidaddBranchInstruction(intopcode,
                               Stringlabel)
```

Add a branch instruction. Following instructions can be added using this method:

```
IFEQ, IFNE, IFLT, IFGE, IFGT, IFLE, IF_ICMPEQ, IF_ICMPNE,
IF_ICMPLT, IF_ICMPGE, IF_ICMPGT, IF_ICMPLE, IF_ACMPEQ,
IF_ACMPNE, GOTO, JSR, IF_NULL, IF_NONNULL, GOTO_W, JSR_W
```

The opcodes for instructions are defined in CLConstants class.

Parameters:

`opcode` - opcode of the instruction.
`label` - branch label.

- **addTABLESWITCHInstruction**

```
publicvoidaddTABLESWITCHInstruction(StringdefaultLabel,
                                    intlow,
                                    inthigh,
                                    ArrayList<String>labels)
```

Add a TABLESWITCH instruction -- used for switch statements.

Parameters:

`defaultLabel` - jump label for default value.
`low` - smallest value of index.
`high` - highest value of index.
`labels` - list of jump labels for each index value from low to high, end values included.

- **addLOOKUPSWITCHInstruction**

```
publicvoidaddLOOKUPSWITCHInstruction(StringdefaultLabel,
                                     intnumPairs,

TreeMap<Integer,String>matchLabelPairs)
```

Add a LOOKUPSWITCH instruction -- used for switch statements.

Parameters:

`defaultLabel` - jump label for default value.
`numPairs` - number of pairs in the match table.
`matchLabelPairs` - key match table.

- **addLDCInstruction**

```
publicvoidaddLDCInstruction(inti)
```

Add an LDC instruction to load an int constant on the operand stack.

Parameters:

`i` - int constant.

- **addLDCInstruction**

```
publicvoidaddLDCInstruction(floatf)
```


Add an LDC instruction to load a float constant on the operand stack.

Parameters:

`f` - float constant.

- **addLDCInstruction**

```
publicvoidaddLDCInstruction(longl)
```

Add an LDC instruction to load a long constant on the operand stack.

Parameters:

`l` - long constant.

- **addLDCInstruction**

```
publicvoidaddLDCInstruction(doubled)
```

Add an LDC instruction to load a double constant on the operand stack.

Parameters:

d - double constant.

- **addLDCInstruction**

```
publicvoidaddLDCInstruction(Strings)
```

Add an LDC instruction to load a String constant on the operand stack.

Parameters:

s - String constant.

- **addClassAttribute**

```
publicvoidaddClassAttribute(CLAttributeInfoattribute)
```

Add the specified class attribute to the attribyte section of the class.

Parameters:

`attribute` - class attribute.

- **addMethodAttribute**

```
publicvoidaddMethodAttribute(CLAttributeInfoattribute)
```

Add the specified method attribute to the attribute section of the method last added.

Parameters:

`attribute` - method attribute.

- **addFieldAttribute**

```
publicvoidaddFieldAttribute(CLAttributeInfoattribute)
```

Add the specified field attribute the attribute section of the field last added.

Parameters:

`attribute` - field attribute.

- **addCodeAttribute**

```
publicvoidaddCodeAttribute(CLAttributeInfoattribute)
```

Add the specified code attribute to the attribute section of the code for the method last added.

Parameters:

attribute - code attribute.

- **addLabel**

```
publicvoidaddLabel(Stringlabel)
```


Add a jump label to the code section of the method being added. A flow control instruction that was added with this label will jump to the instruction right after the label.

Parameters:

`label` - jump label.

- **createLabel**

```
publicStringcreateLabel()
```

Construct and return a unique jump label.

Returns:

unique jump label.

- **pc**

```
publicintpc()
```

Return the pc (location counter). The next instruction will be added with this pc.

Returns:

the pc.

- **constantPool**

```
publicCLConstantPoolconstantPool()
```

Return the constant pool of the class being built.

Returns:

constant pool.

- **initializeByteClassLoader**

```
public staticvoidinitializeByteClassLoader()
```

Set a new ByteClassLoader for loading classes from byte streams.

- **clFile**

```
publicCLFileclFile()
```

Return the CLFile instance corresponding to the class built by this emitter.

- **toClass**

```
publicClasstoClass()
```

Return the class being constructed as a Java Class instance.

Returns:

Java Class instance.

- **write**

```
publicvoidwrite()
```

Write out the class to the file system as a .class file if toFile is true. The destination directory for the file can be set using the destinationDir(String dir) method.