# Chapter 2

# Using the AsmTools

This chapter describes general principles and techniques for using the AsmTools. For detailed information about the syntax of each component and command line examples, see Appendix A (Jasm Syntax) and Appendix B (Jcod Syntax). If no command-line options are provided or they are invalid, the tools provide error messages and usage information. To get the help message, launch AsmTools without any parameters as follows:

java

-jar asmtools.jar

The help system describes how to use all of the AsmTools components and contains the following topics described in this chapter.

## Assemblers and Dissassemblers

Assembly and Dissassembly are reflexive operations.  You can feed one tool into another to achieve the same file.  For example

```
java -jar asmtools.jar jdec foo.class > foo.jcod   # produces foo.jcod
java -jar asmtools.jar jcod foo.jcod               # produces foo.class
```

For a given class foo.class, the product of dissassembly, and re-assembly is the same foo.class.

## Jasm vs. Jcod

Which format to use depends on the task you are trying to do. We can describe some generalizations of when you might wish to use the JASM format versus the JCOD format.

### Jasm

The biggest difference between the two formats is that JASM specifically focuses on representing byte-code instructions in the VM format (while providing minimal description of the structure of  the rest of the class file).  Generally, JASM is more convenient for semantic changes, like change to instruction flow.

### Jcod

JCOD provides good support for describing the structure of a class file (as well as writing incorrect bytes outside of this structure), and provides no support for specifying byte-code instructions (simply raw bytes for instructions).   JCOD is typically used for VMs to test Well-formedness of class files (eg extra or missing bytes), boundary issues, constant-pool coherence, constant-pool index coherence, attribute well-formedness, etc..

### Use Cases

Below are typical cases of usage of both formats:

JASM usages:

- To obtain an invalid class where two methods have the same signature
- To obtain an invalid class reference where an illegal type is used
- To obtain an invalid class with missing/removed instructions
- To insert profiling instructions in methods
- To obtain a class where a keyword is used as an identifier
- To check that two classes produced by different compilers are equivalent

JCOD usages:

- To examine specific parts of a classfile

- eg. constant-pool (for dependency analysis)
- constant values
- inheritance chains (super classes)
- implementation fullfillment (interface resolution)

---

## Tool Usage

Asmtools consist of five utilities:

- jasm - Generates class files from the JASM representation
- jdis - Represents class file in JASM format
- jcoder - Generates class files from the JCOD representation
- jdec - Represents class file in JCOD format
- jcdec - Represents JavaCard cap and exp files in JCOD format

Each utility can be invoked from the command line as shown below:

```
$ java -jar asmtools.jar UTILITY [options] File1 ...
```

or

```
$ java -cp asmtools.jar  com.sun.asmtools.UTILITY.Main [options] File1 ...
```

Each utility supports own set of options

**Note -** See the following sections for the options associated with each tool.

---

### Jasm

jasm is an assembler that accepts a text file based on the JASM Specification, and produces a .class file for use with a Java Virtual Machine.

#### Usage:

$ java -jar asmtools.jar jasm [*options*] filename.jasm

or

$ java -cp asmtools.jar com.sun.asmtools.jasm.Main [*options*] filename.jasm

#### Options:

| option | description |
|---|---|
| `-version` | Print jasm tool version |
| -d *destdir* | Specifies a directory to place resulting .class files. If a destdir is not provided, the .class file will be written in the current directory. |
| `-g` | Add debug information to .class file |
| `-nowrite` | Do not write resulting .class files. This option may be used to verify the integrity of your source jasm file |
| `-strict` | Consider warnings as errors. |
| `-nowarn` | Do not print warnings. |
| -cv *major.minor* | Set the operating class file version (by default 45.3). |

**Note -** If the optional class attribute 'version' defines (in source of class) the class file version, then it overrides default class file version set by -cv option.

---

#### Description:

To use jasm, specify the filename of the .jasm file you wish to develop a .class file from.

Refer to Appendix A (Jasm Syntax) documentation for information on the structure of the `.jasm` file.

---

## Jdis

*jdis* is a disassembler that accepts a `.class` file, and prints the plain-text translation of `jasm` source file to the standard output.

### Usage:

$ java -jar asmtools.jar jdis [*options*] filename.class

or

$ java -cp asmtools.jar com.sun.asmtools.jdis.Main [*options*] filename.class

### Options:

| option | description |
|---|---|
| `-version` | Print jdis tool version |
| -g | Generate a detailed output format. Constants from constant pool are printed, and instructions in methods are preceded with source line numbers (if attribute LineNumberTable is available) and with bytecode program counters. |
| -s1 | Generate source lines in comments. Commented lines of the source file, from which given .class file is obtained, are printed above the corresponding instruction. Both attributes LineNumberTable and SourceFile must be available. The source file should be placed in the current working directory. |
| -hx | Generate floating-point constants in hexadecimal format. |

### Description:

To use jdis, specify a *filename*`.class` that you wish to disassemble.
You may redirect standard output to a *filename* `.jasm` file. Jdis will disassemble a `.class` file and create a resultant `.jasm` source file.

Refer to Appendix A (Jasm Syntax) documentation for information on the structure of the resultant `.jasm` file.

---

## Jcoder

*jcod*er is a low-level assembler that accepts text based on the Jcod Specification. and produces a `.class` file for use with a Java Virtual Machine. Jcod's primary use is as a tool for producing specialized tests for testing a JVM implementation.

### Usage:

$ java -jar asmtools.jar jcoder [*options*] filename.jcod

or

$ java -cp asmtools.jar com.sun.asmtools.jcoder.Main [*options*] filename.jcod

### Options:

| option | description |
|---|---|
| `-version` | Print jcoder tool version |
| -d destdir | Specifies a directory to place resulting .class files. If a destdir is not provided, the .class file will be written in the current directory. |
| -nowrite | Do not write resulting .class files. This option may be used to verify the integrity of your source jcod file. |

### Description:

To use jcod, specify the *filename*`.jcod` file you wish to develop a `.class`file from.

Refer to Appendix B (Jcod Syntax) documentation for information on the structure of the `.jcod` file.

---

## Jdec

*jdec* is a low-level disassembler that accepts `.class` file and prints a plain text of `jcod` source file to the standard output.

### Usage:

$ java -jar asmtools.jar jdec [*options*] filename.class [> filename.jcod]

or

$ java -cp asmtools.jar com.sun.asmtools.jdec.Main [*options*] filename.class [> filename.jcod]

### Options:

| option | description |
|---|---|
| `-version` | Print jdec tool version |
| -g | Generate a detailed output format. |

### Description:

To use jdec, specify a *`filename.class`* that you wish to disassemble.
You may redirect standard output to a *`filename.jcod`* file. *jdec* will disassemble `.class` file and create a resultant `.jcod` plain source file.

Refer to Appendix B (Jcod Syntax) documentation for information on the structure of the resultant `.jcod` file.

---

## Jcdec

*jcdec* is a low-level disassembler that accepts `.class` file and prints a plain text of `jcod` source file to the standard output.

### Usage:

$ java -jar asmtools.jar jcdec [*options*] filename.exp | filename.cap [> filename.jcod]

or

$ java -cp asmtools.jar com.sun.asmtools.jcdec.Main [*options*] filename.exp | filename.cap [> filename.jcod]

### Options:

| option | description |
|---|---|
| `-version` | Print jcdec tool version |
| -g | Generate a detailed output format. |

### Description:

To use jcdec, specify a *`filename.exp`* or *`filename.cap`* that you wish to disassemble.
You may redirect standard output to a *`filename.jcod`* file. *jcdec* will disassemble the file and create a resultant `.jcod` plain source file.

Refer to Appendix B (Jcod Syntax) documentation for information on the structure of the resultant `.jcod` file.

---

Chapter 3