# Java CLI JDK command line

**Main.java** source → `javac` → **Main.class** bytecode → `jar` → **app.jar** archive → `java -jar` → **JVM** running app

## CONCEPTS

Source becomes bytecode with javac, bytecode runs on the JVM with java. Tools like jar, jshell and jcmd all ship in the JDK.

Class path separator is : on Linux and macOS, ; on Windows.

A JRE-only install has java but no javac, jar or jshell: install a JDK.

## RUN

**`java -version`**
Print version and exit

**`java <File>.java [args]`**
Compile in memory and run

**`java -cp <path> <MainClass> [args]`**
Run a class

**`java -jar <file.jar> [args]`**
Run a JAR's main class

**`java -Dkey=value ...`**
Set a system property

**`java -ea ...`**
Enable assertions

## RUN (cont.)

**`java -m <module>/<MainClass>`**
Run a module

## CLASS PATH & MODULES

**`-cp "lib/*:out"`**
Add all JARs in lib, plus out

**`--module-path <dir> / -p <dir>`**
Where application modules live

**`--add-opens <mod>/<pkg>=ALL-UNNAMED`**
Open a package to reflection

## MEMORY & JVM FLAGS

**`-Xmx<size>`**
Maximum heap

**`-Xms<size>`**
Initial heap

**`-Xss<size>`**
Thread stack size

**`-XX:MaxRAMPercentage=<n>`**
Heap as share of RAM

**`-XX:+HeapDumpOnOutOfMemoryError`**
Write a heap dump on OOM

## COMPILE (JAVAC)

**`javac <File>.java`**
Compile one file

**`javac -d <dir> <files>`**
Put class files in dir

**`javac -cp <path> <files>`**
Compile against libraries

**`javac --release <n> <files>`**
Target a Java version

**`javac -Xlint:all <files>`**
Enable all warnings

**`javac -encoding UTF-8 <files>`**
Source file encoding

**`javac @<argfile>`**
Options and files from a file

## PACKAGE (JAR)

**`jar --create --file <f.jar> -C <dir> .`**
Pack a directory

**`jar --create --file <f.jar> --main-class <Class> -C <dir> .`**
Runnable JAR

**`jar -tf <f.jar>`**
List contents

**`jar -xf <f.jar> [<entry>]`**
Extract all or one entry

## JSHELL

**`jshell`**
Start the REPL

**`jshell --class-path <path>`**
Start with libraries

**`/vars`**
Show variables

**`/methods`**
Show methods

**`/list`**
Show snippets with IDs

## DIAGNOSE

**`jcmd`**
List running JVMs

**`jps -l`**
PIDs with main class or JAR

**`jcmd <pid> Thread.print`**
Thread dump

**`jcmd <pid> GC.heap_info`**
Heap usage summary

**`jcmd <pid> GC.class_histogram`**
Objects per class

**`jcmd <pid> GC.heap_dump <file>`**
Write a heap dump

**`jcmd <pid> JFR.start duration=60s filename=<f.jfr>`**
Record a flight recording

## DIAGNOSE (cont.)

**`jcmd <pid> help`**
Commands this JVM knows

## ENVIRONMENT

**`JAVA_HOME`**
JDK install directory

**`JAVA_TOOL_OPTIONS`**
Options for every JVM

## RECIPES

### Build and run a small app

```
javac -d out $(find src -
name '*.java')
jar --create --file app.jar
--main-class
com.example.Main -C out .
java -jar app.jar
```

### Run with libraries

```
javac -cp "lib/*" -d out
src/Main.java
java -cp "out:lib/*" Main
```

### Diagnose a stuck app

```
jps -l
jcmd <pid> Thread.print -l >
threads.txt
jcmd <pid> GC.heap_info
```

### Capture memory failures

```
java -Xmx512m \
  -XX:
+HeapDumpOnOutOfMemoryError
\
  -XX:HeapDumpPath=/tmp/
heap.hprof \
  -jar app.jar
```