**Compiling and running Groovy**

* we’ve used Groovy in direct ­mode, where the code is directly executed without producing any executable files.
* A second way of using Groovy: compiling it to Java bytecode and running it as regular Java application code within a JVM, called precompiled mode.

**Note:**

Both ways execute Groovy inside a JVM eventually, and both ways compile the Groovy code to Java bytecode. The major difference is when that compilation occurs and whether the resulting classes are used in memory or stored on disk.

**Compiling Groovy with groovy**

Compiling Groovy is straightforward because it comes with a compiler called groovyc. The groovyc compiler generates at least one class file for each Groovy source file compiled.

As an example, you can compile file.groovy to normal Java bytecode by running groovyc on the script file like so:

groovyc -d classes file.groovy

In this case, the groovyc compiler outputs Java class files to a directory named classes, which you told it to do with the -d flag

If the directory specified with –d doesn’t exist, it’s created. When you’re running the compiler, the name of each generated class file is printed to the console.

**Running a compiled Groovy script with Java**

Running a compiled Groovy program is identical to running a compiled Java program, with the added requirement of having the embeddable groovy-all-\*.jar file in your JVM’s classpath, which will ensure that all of Groovy’s third-party dependencies will be resolved automatically at runtime.

java -cp %GROOVY\_HOME%/embeddable/groovy-all-2.4.13.jar; MyGroovyScrpt