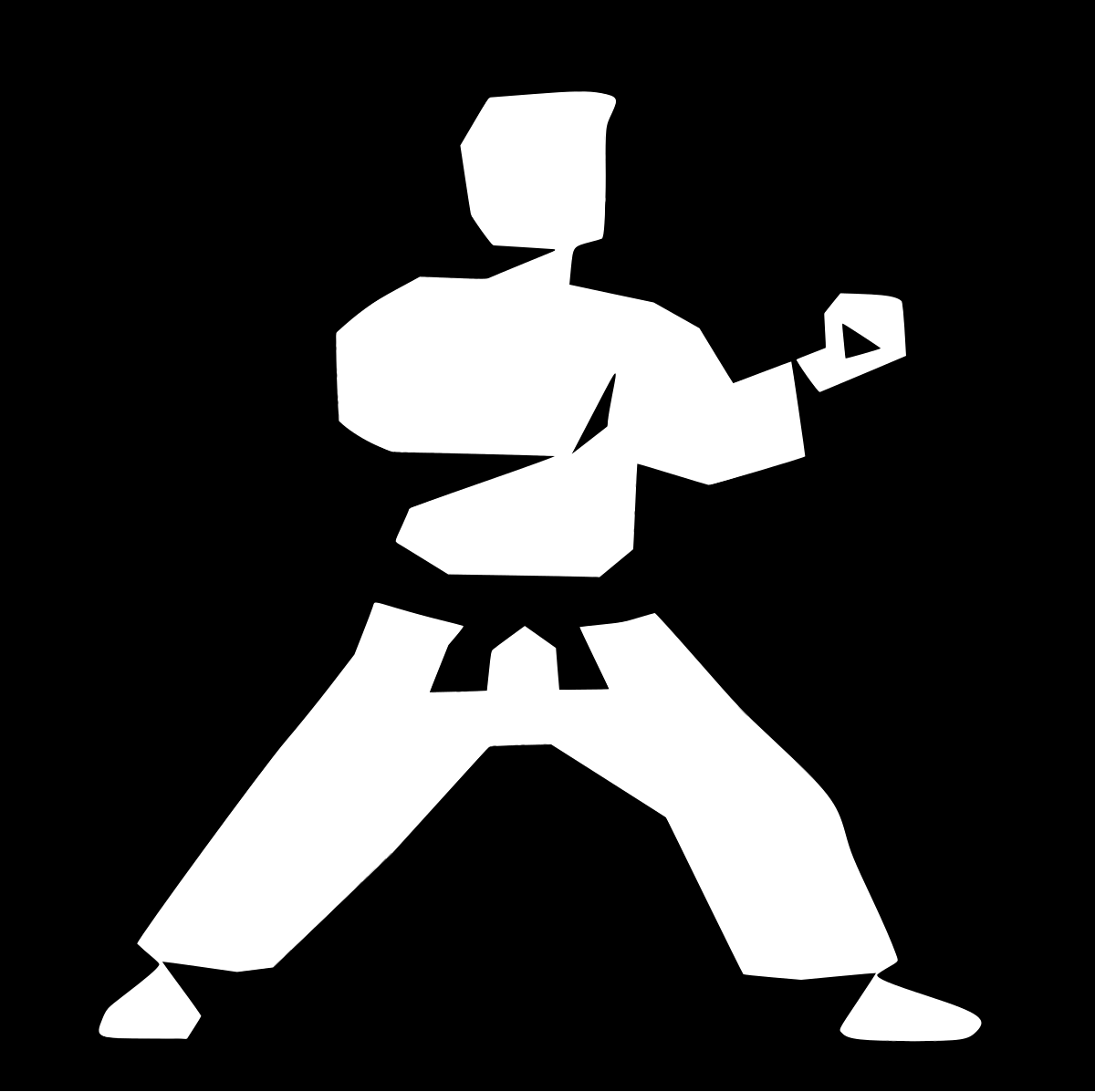
**Run Karate Test from command line**

# 

# **Installing Apache Maven**

**The installation of Apache Maven is a simple process of extracting the archive and adding the `bin` folder with the `mvn` command to the `PATH`.**

**Detailed steps are:**

* **Ensure JAVA\_HOME environment variable (system variable) is set and points to your JDK installation ( jdk folder )**
* **Extract distribution archive in any directory**

**Alternatively use your preferred archive extraction tool.**

* **Add the bin directory of the created directory apache-maven-version to the PATH environment variable**
* **Confirm with mvn -v in a new shell. The result should look similar to**

1. **Apache Maven version (cecedd343002696d0abb50b32b541b8a6ba2883f)**
2. **Maven home: bin path**
3. **Java version:13.0.1, vendor: Oracle Corporation**
4. **Java home: jdk path**
5. **Default locale: en\_US, platform encoding: UTF-8**
6. **OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"**

## 

## **Windows Tips**

* **Check environment variable value e.g.**

1. **echo %JAVA\_HOME%**
2. **C:\Program Files\Java\jdk13.0.1**

* **Adding to PATH: Add the unpacked distribution’s bin directory to your user PATH environment variable by opening up the system properties (WinKey + Pause), selecting the “Advanced” tab, and the “Environment Variables” button, then adding or selecting the *PATH* variable in the user variables with the value C:\Program Files\apache-maven-version\bin. The same dialog can be used to set JAVA\_HOME to the location of your JDK, e.g. C:\Program Files\Java\jdk13.0.1**
* **Open a new command prompt (Winkey + R then type cmd) and run mvn -v to verify the installation.**

# **Running Apache Maven**

**The syntax for running Maven is as follows:**

**mvn [options] [<goal(s)>] [<phase(s)>]**

**mvn -help**

**Options:**

**-am,--also-make If project list is specified, also**

**build projects required by the**

**list**

**-amd,--also-make-dependents If project list is specified, also**

**-e,--errors Produce execution error messages**

**-emp,--encrypt-master-password <arg> Encrypt master security password**

**-ep,--encrypt-password <arg> Encrypt server password**

**-f,--file <arg> Force the use of an alternate POM**

**file (or directory with pom.xml)**

**-gs,--global-settings <arg> Alternate path for the global**

**settings file**

**-l,--log-file <arg> Log file where all build output**

**will go (disables output color)**

**-q,--quiet Quiet output - only show errors**

**settings file**

**-T,--threads <arg> Thread count, for instance 2.0C**

**where C is core multiplied**

**-U,--update-snapshots Forces a check for missing**

**releases and updated snapshots on**

**remote repositories**

**-v,--version Display version information**

**-V,--show-version Display version information**

**WITHOUT stopping build**

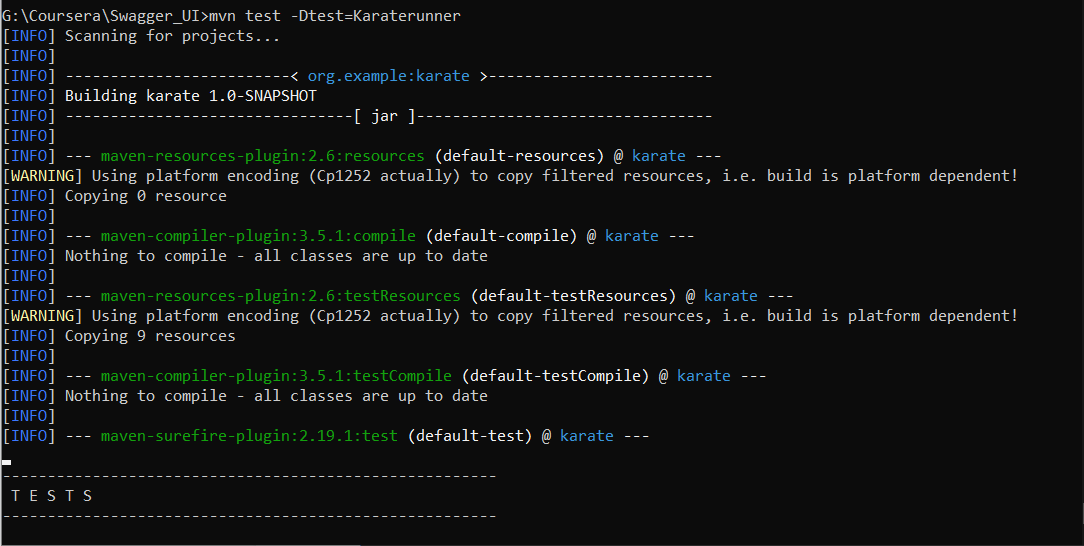
## **Command Line**

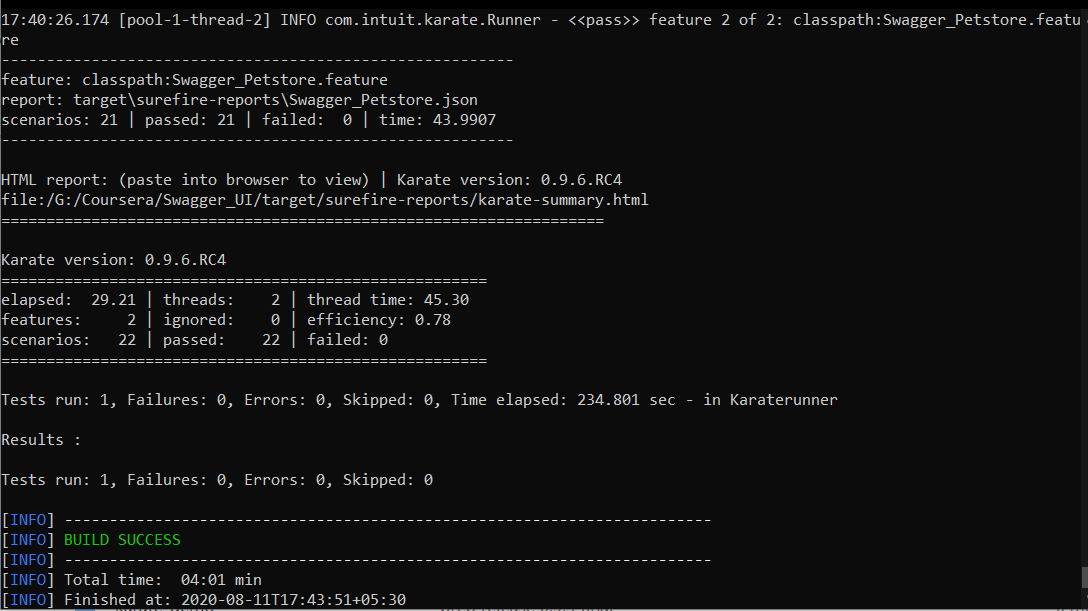
**Normally in dev mode, you will use your IDE to run a \*.feature file directly or via the companion 'runner' JUnit Java class. When you have a 'runner' class in place, it would be possible to run it from the command-line as well.**

**Note that the mvn test command only runs test classes that follow the \*Test.java** [**naming convention**](https://github.com/intuit/karate/#naming-conventions) **by default. But you can choose a single test to run like this:**

**mvn test -Dtest=Runner\_Class\_filename**

**When your Java test "runner" is linked to multiple feature files, which will be the case when you use the recommended** [**parallel runner**](https://github.com/intuit/karate/#parallel-execution)**, you can narrow down your scope to a single feature (or even directory) via the command-line, useful in dev-mode. Note how even** [**tags**](https://github.com/intuit/karate/#tags) **to exclude (or include) can be specified using the** [**Karate options**](https://github.com/intuit/karate/#karate-options)**.**

**mvn test -Dkarate.options="--tags ~@ignore classpath:feature\_file\_path" -Dtest=Runner\_Class\_filename**

****