Upgrade Java Auto Modules

Pack common-cli and others with jlink

Onkobu Tanaake

2021-10-03T10:00:00

Java 9 added project JigSaw or modules. The entire JDK is not longer a large block. Instead it consists of modules. And since Java WebStart has been deprecated and removed with JDK 11 there was the need for a better shipping format. With Java WebStart or even so called Applets it was necessary to download at least a JRE or Java Runtime Environment first to run any Java program.

So a new tool called `jlink` was added. In combination with module declaration a developer can pack all necessary modules including the JDK ones into a single ZIP-archive. This also includes a run script. When you download and unpack this ZIP you can run it right away.

Some of the Java libraries I use are a bit outdated and probably never become fully fledged Java modules. But I still want to pack them as part of my programs. The invocation of `jlink` fails with an error message that auto modules cannot be analyzed correctly. It simply bails out and there is no ZIP to distribute.

```
Error: jdk.tools.jlink.plugin.PluginException: module-info.class
```

The new tools and features suffer from long standing bugs. Namely JDK-8229396 [https://bugs.openjdk.java.net/browse/JDK-8229396] tells the story of `jdeps` that was long broken. Finally JDK-8130047 [https://bugs.openjdk.java.net/browse/JDK-8130047] shows that `jlink` might never get fixed. So clever people out there found a way to re-package a JAR to become a distinct module instead:

1. Auto-generate the modules-info.java through `jdeps`
2. compile the file with `javac`
3. Update the JAR with the compiled module definition
4. update Maven signatures (signed modules cannot be patched)

One of my soliton scripts does all that for you. Have a look at Codeberg.org [https://codeberg.org/onkobu/soliton/src/branch/master/bin/to-explicit.sh].