**Case 1: Run Jar**

Assume current directory is C:\Users\gewin\testlab\jar-console-app\target, I can run it successfully with:

java -jar jar-console-app-1.0-SNAPSHOT.jar

**Manifest.MF in jar-console-app-1.0-SNAPSHOT.jar**

Manifest-Version: 1.0

Built-By: gewin

**Class-Path: external/find-screen-service-1.0.jar external/find-screen-service-1.1.jar**

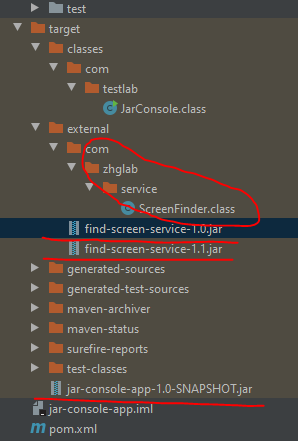
Created-By: Apache Maven 3.6.1

Build-Jdk: 1.8.0\_221

Main-Class: com.testlab.JarConsole

**Fig. 2**

**Directory Structure**



**Fig 1**

**Explanation:**

In this case, find-screen-service-1.0.jar will actually be loaded for ScreenFinder class which is referenced from JarConsole class.

**Case 2. Run class file**

assume current dir: C:\Users\gewin\testlab\jar-console-app\target\classes>

java -classpath .;../\* com.testlab.JarConsole // success

This is JVM can probe jar-console-app-1.0-SNAPSHOT.jar and acquires Class-Path in its MANIFEST.MF which quoted in “Fig. 2” and find the ScreenFinder class, If I delete jar-console-app-1.0-SNAPSHOT.jar or rename external folder to external1, then above java execution will fail.

If I just delete jar-console-app-1.0-SNAPSHOT.jar(keeping external folder name unchanged), then

java -classpath .;../external/\* com.testlab.JarConsole

will succeed.

**Note**: in current state:

java -classpath ../external/\* com.testlab.JarConsole

will fail with “Error: Could not find or load main class com.testlab.JarConsole” as JVM only use ../external/\* as class path.

If I change the path to C:\Users\gewin\testlab\jar-console-app\target>, with the same directory structure as Fig. 1, then only below java exeution will succeed.

java -classpath ./classes;./external/\* com.testlab.JarConsole

or

java -classpath classes;external/\* com.testlab.JarConsole

\*\*\*In case 1, although there are 2 version of find-screen-service-1.0.jar in the class path, but in the MANIFEST.MF, external/find-screen-service-1.0.jar is before external/find-screen-service-1.1.jar specified in Class-Path, in such case, the find-screen-service-1.0.jar is used because it is specified first in the class path. This occurrence is in accordance with Class-Path Attibute(in “[Jar File Specification](https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jar.html" \l "classpath)”) and the actual ClassLoader used is “[UrlClassLoader](https://docs.oracle.com/javase/6/docs/api/java/net/URLClassLoader.html)”(The URLs will be searched in the order specified for classes and resources after first searching in the parent class loader)), in UrlClassLoader Api documentation it is specified that the package loading order is in the “specified order for classes and resources”, so whenever first ScreenFinder class is found, its belonging package will be used.

**Case 3. Run Jar with directory in Class-Path**

If extract find-screen-service-1.0.jar under external folder like illustrated in Fig 1., then Manifest.MF Class-Path can be specified as bleow:

Class-Path: external\

When such change was made to Class-Path in MANIFEST.MF, and created jar-console-app-1.0-SNAPSHOT.jar, I can run it under C:\Users\gewin\testlab\central-test-lab\jar-console-app\target successfully with

java -jar jar-console-app-1.0-SNAPSHOT.jar

This case illustrated Class-Path can be directory but it will be used to search for .class file(not .jar file).

The git repo for this test is here.