CHAPTER 11

■ ■ ■

Packaging Modules

QUESTIONS AND EXERCISES

1. What formats can you use to package your modules?

Answer:

You can package your modules in the following formats:

- Exploded directory
- Modular JAR format
- JMOD format
- JIMAGE format

2. What is a multi-release JAR?

Answer:

A multi-release JAR is a JAR that contains the same release of a library – offering the same APIs – for multiple JDK versions.

3. Describe the structure of a multi-release JAR.

Answer:

The multi-release JAR extends the existing directory structure for a JAR. A JAR contains a root directory where all its contents reside. It contains a `META-INF` directory that is used to store metadata about the JAR. Typically, a JAR contains a `META-INF/MANIFEST.MF` file containing its attributes. In a multi-release JAR, the `META-INF` directory contains a `versions`

sub-directory, which may contain many sub-directories—each of them named the same as the JDK major version. For example, for classes specific to JDK 9, there may be the `META-INF/versions/9` directory and, for classes specific to JDK 10, there may be a directory called `META-INF/versions/10`, etc. A typical MRJAR may have the following entries:

```
- jar-root
  - <common-classes>
- META-INF
  - MANIFEST.MF
  - versions
    - 9
      - <classes for JDK 9>
    - 10
      - <classes for JDK 10>
```

4. What happens when a multi-release JAR is used on a JDK version (for example JDK8) that does not understand multi-release JARs?

   Answer:

   when a multi-release JAR is used on a JDK version (for example JDK 8), which does not understand multi-release JARs, the multi-release JAR is treated as a JAR. The classes from the `/META-INF/versions` directory are ignored. All classes from the root directory are used.

5. Describe the search order when a resource is looked up in a multi-release JAR.

   Answer:

   The search for a resource or class file in a multi-release JAR (MRJAR) uses the following rules:

   - The major version of the JDK is determined for the environment in which the MRJAR is being used. Suppose the major version of the JDK is `N`.
   - To locate a resource or a class file named `R`, the platform-specific sub-directory under the `META-INF/versions` directory is searched starting at the directory for version `N`.
   - If `R` is found in sub-directory `N`, it is returned. Otherwise, sub-directories for versions lower than `N` are searched. This process continues for all sub-directories under the `META-INF/versions` directory.
   - When `R` is not found in the `META-INF/versions/N` sub-directories, the root directory of the MRJAR is searched for `R`.

6. Describe the limitations of a multi-release JAR.

Answer:

The boot loader does not support multi-release JARs..

7. What is the name of the attribute that is present in the `META-INF\MANIFEST.MF` file for a multi-release JAR?

Answer:

The name of the attribute that is present in the `META-INF\MANIFEST.MF` file for a multi-release JAR is `Multi-Release` and it is set to `true`.

8. What is the `jmod` tool and where is it located?

Answer:

The `jmod` is be used to create JMOD files, list the contents of JMOD files, print the description of a module, and record hashes of the modules used in a JMOD file. The tool is in the `JDK_HOME\bin` directory.

9. What is the JMOD format and how is it better than the JAR format.

Answer:

JDK 9 introduced a new format, called JMOD, to package modules.

JMOD files are designed to handle more content types than JAR files can. JMOD files can package native code, configuration files, native commands, and other kinds of data.

10. Java supports three phases from JDK 9: compile-time, link time, and runtime. In what phases is the JMOD format supported?

Answer:

JMOD format is supported at compile-time and link time.

11. Suppose you have a JMOD file named `jdojo.test.jmod`. Write the command using the `jmod` tool to describe the module stored in this JMOD file.

Answer:

```
jmod describe jdojo.test.jmod
```

12. What is the location of the JDK modules in JMOD format?

Answer:

All JDK modules in JMOD format are stored in the `JDK_HOME/jmods` directory.
