JAR Files in Java

An Overview with Examples

RAHULRAJ P

What is jar file in java ?

JAR (Java Archive) is a file format used to combine and compress multiple files into a single package. It is based on the popular ZIP format, which is commonly used for compressing and storing files. JAR files are mainly used in Java to package and compress related files, such as code, resources, and libraries, into one archive.

Executable JAR files:

An executable JAR (Java Archive) file is a special type of JAR file that can run a Java application directly, without needing additional setup or compilation. It combines all the necessary resources, such as compiled Java classes, dependencies, and application-specific files, into a single file with a .jar extension.

To make a JAR file executable, it must include a manifest file located at META-INF/MANIFEST.MF. This file specifies the main class, which is the starting point of the application. When the JAR file is run, the code in the main class is executed.

Why Use JAR Files ?

- ⭐ **Convenience:** Combine multiple files into a single archive for easy distribution.
- ⭐ **Compression:** Reduces file size for storage and transfer.
- ⭐ **Portability:** JAR files can be used across different platforms.
- ⭐ **Dependency Management:** Simplifies the inclusion of libraries in Java projects.

```
rahul-ts426@rahul-ts426 Java % javac Main.java
rahul-ts426@rahul-ts426 Java % java Main
This is Main Class
rahul-ts426@rahul-ts426 Java % ls
Main.class      Main.java
rahul-ts426@rahul-ts426 Java %
```

I have a Java file named Main.java. After compiling it with javac Main.java, I will get a Main.class file, which contains the bytecode needed for execution.

Basic JAR Commands

1.Creating a Jar file:

> jar cf <jar-file> <input-file>

- *c* - Create a new jar file
- *f* - Specify the output file Main.jar

```
rahul-ts426@rahul-ts426 Java % jar cf Main.jar Main.class
rahul-ts426@rahul-ts426 Java % ls
Main.class      Main.jar        Main.java
rahul-ts426@rahul-ts426 Java %
```

2.Extracting files from a JAR file:

> jar xf <jar-file>

- *x* - Extract the contents of the jar file
- *f* - Specify the output file name Main.jar

```
rahul-ts426@rahul-ts426 Java % jar xf Main.jar
rahul-ts426@rahul-ts426 Java % ls
META-INF        Main.class      Main.jar        Main.java
rahul-ts426@rahul-ts426 Java %
```

```
rahul-ts426@rahul-ts426 Java % cd META-INF
rahul-ts426@rahul-ts426 META-INF % ls
MANIFEST.MF
rahul-ts426@rahul-ts426 META-INF %
```

After running the command jar xf Main.jar, it extracts the contents of the JAR file, including the META-INF/ directory, which contains metadata like the MANIFEST.MF file.

3.Listing the contents of a JAR file:

> jar tf <jar-file>

- *t* - Lists the contents of the JAR file.
- *f* - Specify the output file name Main.jar to be listed.

```
rahul-ts426@rahul-ts426 Java % jar tf Main.jar
META-INF/
META-INF/MANIFEST.MF
Main.class
rahul-ts426@rahul-ts426 Java %
```

After running the command jar tf Main.jar, it lists the contents of the JAR file, including META-INF/, META-INF/MANIFEST.MF, and Main.class.

4.Adding files to an existing JAR file:

```
jar uf <jar-file> <input-file>
```

- *u* - Updates an existing JAR file by adding or modifying files in it.
- *f* - Specify the output file name Main.jar.

```
rahul-ts426@rahul-ts426 Java % javac sample.java
rahul-ts426@rahul-ts426 Java % java sample
This is Sample Class
rahul-ts426@rahul-ts426 Java % ls
META-INF        Main.class      Main.jar        Main.java       sample.class    sample.java
rahul-ts426@rahul-ts426 Java %
```

I create a new Java file named Sample.java. After compiling it with javac Sample.java, I get a Sample.class file.

```
rahul-ts426@rahul-ts426 Java % jar uf Main.jar sample.class
rahul-ts426@rahul-ts426 Java % jar tf Main.jar
META-INF/
META-INF/MANIFEST.MF
Main.class
sample.class
rahul-ts426@rahul-ts426 Java %
```

After updating the JAR file with jar uf Main.jar Sample.class and listing its contents with jar tf Main.jar, the JAR file will display the updated contents, including Sample.class.

5.Extracting specific files from a JAR file:

```
jar xf <jar-file> <input-file>
```

- *x* - Extract specific files or the entire contents of the JAR file.
- *f* - Specify the output file name Main.jar.

```
rahul-ts426@rahul-ts426 Java % javac Example.java
rahul-ts426@rahul-ts426 Java % java Example
This is Example Class
rahul-ts426@rahul-ts426 Java % ls
Example.class   Example.java    META-INF        Main.class      Main.jar        Main.java       sample.class    sample.java
```

I create a new Java file named Example.java. After compiling it with javac Example.java, I get a Example.class file.

```
rahul-ts426@rahul-ts426 Java % jar uf Main.jar Example.class

rahul-ts426@rahul-ts426 Java % jar tf Main.jar
META-INF/
META-INF/MANIFEST.MF
Main.class
sample.class
Example.class
rahul-ts426@rahul-ts426 Java %
```

First, update the JAR file using jar uf Main.jar Example.class, then verify the contents with jar tf Main.jar

```
rahul-ts426@rahul-ts426 Java % jar xf Main.jar Example.class
rahul-ts426@rahul-ts426 Java % ls
Example.class   Example.java    META-INF        Main.class      Main.jar        Main.java       sample.class    sample.java
rahul-ts426@rahul-ts426 Java %
```

6.Running an executable a JAR file:

> java -jar <jar-file>

```
rahul-ts426@rahul-ts426 Java % java -jar Main.jar
no main manifest attribute, in Main.jar
rahul-ts426@rahul-ts426 Java %
```

The error means that the JAR file (Main.jar) does not have a main class specified in its manifest file (META-INF/MANIFEST.MF). The Java runtime cannot determine which class contains the main method to execute.

```
rahul-ts426@rahul-ts426 Java % touch manifest.txt
rahul-ts426@rahul-ts426 Java % vi manifest.txt
rahul-ts426@rahul-ts426 Java % ls
Example.class   Example.java    META-INF        Main.class      Main.jar        Main.java       manifest.txt    sample.class    sample.java
rahul-ts426@rahul-ts426 Java %
```

```
Main-Class: Main
```

Create a manifest.txt file and set the content to Main-Class: Main.

```
rahul-ts426@rahul-ts426 Java % jar cfm Main.jar manifest.txt Main.class
rahul-ts426@rahul-ts426 Java % java -jar Main.jar
This is Main Class
rahul-ts426@rahul-ts426 Java %
```

After setting Main-Class: Main in the manifest.txt file, rebuild the JAR file, and then run the command java -jar Main.jar. If Main.java contains a main method with print statements, it will display the output specified in the main method.

THANK YOU