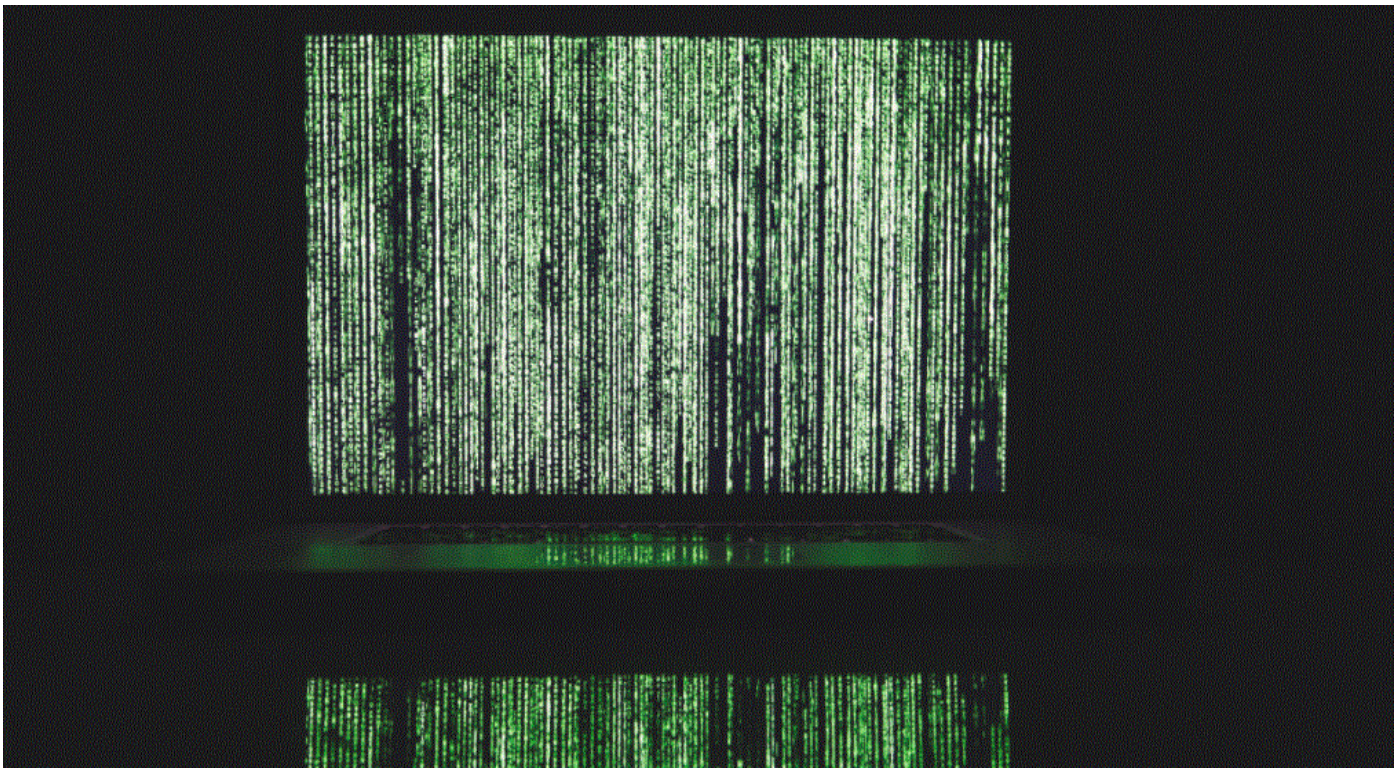
Demystify Jar File



The Java™ Archive (JAR) file format enables you to bundle multiple files into a single archive file. Typically, a JAR file contains the class files and associated metadata and resources (text, images, etc.) into one file to distribute application software or libraries on the Java platform.

Using JAR Files: The Basics

JAR files are packaged with the ZIP file format, so you can use them for tasks such as lossless data compression, archiving, decompression, and archive unpacking. These tasks are among the most common uses of JAR files, and you can realize many JAR file benefits using only these basic features.

To perform basic tasks with JAR files, you use the Java Archive Tool provided as part of the Java Development Kit (JDK). Because the Java Archive tool is invoked by using the jar command, this tutorial refers to it as ‘the Jar tool’.

Common JAR file operations

Operation	Command
To create a JAR file	jar cf jar-file input-file(s)
To view the contents of a JAR file	jar tf jar-file
To extract the contents of a JAR file	jar xf jar-file
To extract specific files from a JAR file	jar xf jar-file archived-file(s)
To run an application packaged as a JAR file (requires the Main-class manifest header)	java -jar app.jar

Creating a JAR File

The basic format of the command for creating a JAR file is:

```
jar cf jar-file input-file(s)
```

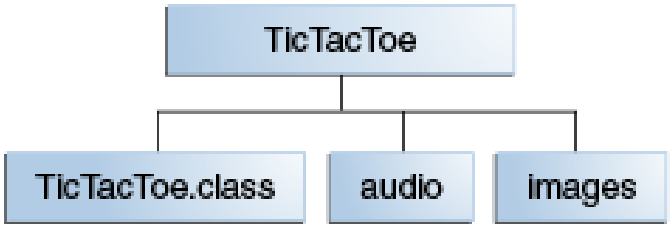
The options and arguments used in this command are:

- The c option indicates that you want to create a JAR file.
- The f option indicates that you want the output to go to a file rather than to stdout.
- jar-file is the name that you want the resulting JAR file to have. You can use any filename for a JAR file. By convention, JAR filenames are given a .jar extension, though this is not required.
- The input-file(s) argument is a space-separated list of one or more files that you want to include in your JAR file. The input-file(s) argument can contain the wildcard * symbol. If any of the “input-files” are directories, the contents of those directories are added to the JAR archive recursively.

The c and f options can appear in either order, but there must not be any space between them.

An Example

Let us look at an example. A simple TicTacToe application.



The audio and images subdirectories contain sound files and GIF images used by the applet.

You can obtain all these files from jar/examples directory when you download the entire Tutorial online. To package this demo into a single JAR file named TicTacToe.jar, you would run this command from inside the TicTacToe directory:

```
jar cvf TicTacToe.jar TicTacToe.class audio images
```

The audio and images arguments represent directories, so the Jar tool will recursively place them and their contents in the JAR file. The generated JAR file TicTacToe.jar will be placed in the current directory.

The Jar tool will accept arguments that use the wildcard * symbol. As long as there weren't any unwanted files in the TicTacToe directory, you could have used this alternative command to construct the JAR file:

```
jar cvf TicTacToe.jar *
```