JAR FILES ARE ZIP FILES USED TO AGGREGATE JAVA CLASSES AROUND EFFICIENTLY

LET'S SAY WE HAVE WRITTEN A LOT OF JAVA CODE - A LOT OF CLASSES, AND A LOT OF INTERFACES

QUESTION

HOW WILL WE NOW MAKE THAT CODE AVAILABLE TO OTHER PROGRAMMERS TO USE IN THEIR OWN CODE?

ANSWER

WE COLLECT ALL OF THE CLASSES, RESOURCES (TEXT OR IMAGE FILES OUR CODE USES) AND METADATA (ANNOTATIONS ETC) INTO A SINGLE, COMPRESSED FILE, CALLED A

JAR FILE

JAR FILES CAN BE CREATED FROM A COMMAND-LINE TOOL THAT COMES ALONG WITH THE JAVA COMPILER

Creating a JAR File

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

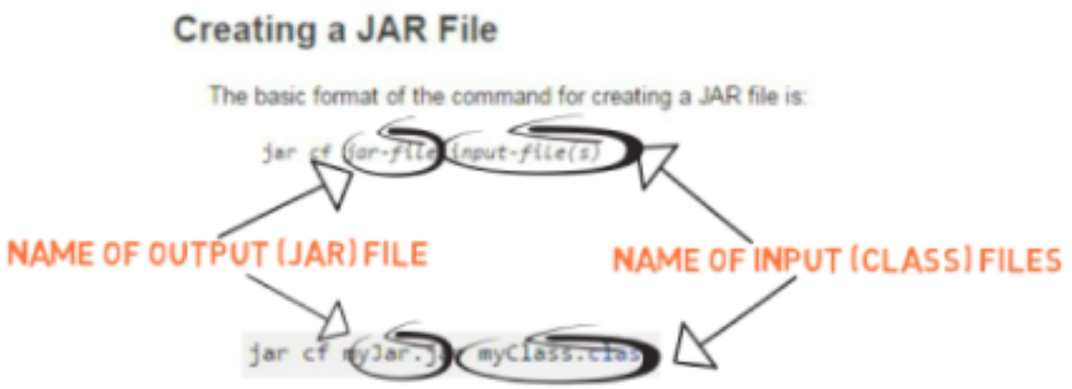
```
jar cf jar-file input-file(s)
```

NAME OF OUTPUT (JAR) FILE

NAME OF INPUT (CLASS) FILES

```
jar cf myJar.jar myClass.class
```

JAR FILES CAN BE CREATED FROM A COMMAND-LINE TOOL THAT COMES ALONG WITH THE JAVA COMPILER



THE JAR FILE IS SIMILAR TO A ZIP FILE, IN FACT THE .JAR FORMAT IS BUILT ON THE .ZIP FORMAT.

THE .CLASS FILES, IN TURN, ARE PRODUCED WITH JAVA SOURCE CODE (THE .JAVA FILES) ARE COMPILED BY JAVA

THE .CLASS FILES ARE BYTECODE REPRESENTATIONS OF OUR SOURCE CODE (.JAVA) FILES

REMEMBER HOW WE MENTIONED THAT JAVA CODE IS ARCHITECTURE-INDEPENDENT, BECAUSE IT IS COMPILED INTO SOMETHING CALLED **BYTECODE?**

HOW CAN JAR FILES BE USED?

1. THEY CAN BE USED TO WRITE CODE

IF PROGRAMMER WRITES CODE THAT YOU WOULD LIKE TO USE:

A. GET A JAR FILE OF THAT CODE

B. ADD A REFERENCE TO THAT JAR FILE IN YOUR JAVA PROJECT

C. USE THE PUBLIC CLASSES AND TYPES IN THAT JAR FILE DIRECTLY IN YOUR CODE

2. THEY CAN BE DIRECTLY EXECUTED

A JAR FILE CAN EXPLICITLY BE MARKED AS EXECUTABLE WHILE BEING CREATED

JAVA CODE IS EXECUTED EITHER THROUGH AN IDE, OR USING A COMMAND LINE TOOL

2. THEY CAN BE DIRECTLY EXECUTED

A JAR FILE CAN EXPLICITLY BE MARKED AS EXECUTABLE WHILE BEING CREATED

JAVA CODE IS EXECUTED EITHER THROUGH AN IDE, OR USING A COMMAND LINE TOOL

IN ORDER FOR EITHER THE IDE OR THE JAVA COMMAND LINE TOOL TO KNOW WHERE TO START EXECUTING, AN "ENTRY POINT" MUST BE SPECIFIED

AN ENTRY POINT IS SIMPLY THE CLASS IN WHICH THE MAIN METHOD WILL BE EXECUTED

TO SPECIFY THIS ENTRY POINT, SOMETHING MORE THAN A MERE ZIP-FILE-LIKE ARCHIVE OF FILES IS NEEDED..

THAT SOMETHING IS CALLED

THE MANIFEST

JAR FILES ARE A LOT SMARTER THAN ZIP FILES

JAR FILES CAN BE

EXECUTED

ELECTRONICALLY SIGNED

VERSION CONTROLLED

SEALED

("SEALING" A JAR MEANS THAT ALL CLASSES IN THAT PACKAGE MUST BE IN THE SAME JAR FILE)

SET SECURITY ATTRIBUTES

ALL OF THESE ARE MADE POSSIBLE BY HAVING THE JAR FILE CONTAIN

METADATA

THE MANIFEST CONTAINS METADATA, I.E. DATA ABOUT THE FILES PACKAGED IN THE JAR FILE

THE MANIFEST CONTAINS METADATA,
I.E. DATA ABOUT THE FILES PACKAGED
IN THE JAR FILE

WHEN A JAR FILE IS CREATED,
SOMETHING CALLED A DEFAULT MANIFEST IS CREATED TOO

ADDITIONAL INFORMATION CAN BE ADDED
TO THE DEFAULT MANIFEST USING THE JAR
TOOL

```
jar cfm jar-file manifest-addition input-file(s)
```

NAME (AND PATH) OF A TEXT FILE
WITH ADDITIONAL INFORMATION TO BE
ADDED TO THE MANIFEST

(THERE IS SOME FINE PRINT AROUND THE
FORMAT AND ENCODING OF THIS
MANIFEST FILE - IT NEEDS TO BE UTF-8
ENCODED)

NOW, FOR INSTANCE TO MAKE A JAR FILE EXECUTABLE -

CREATE A MANIFEST FILE (SAY "MANIFEST.TXT"), WITH THE FOLLOWING LINE IN IT

```
Main-Class: MyPackage.MyClass
```

THEN RUN THE JAR TOOL POINTING TO THIS MANIFEST.TXT FILE

```
jar cfm MyJar.jar Manifest.txt MyPackage/*.class
```

THE RESULTING JAR FILE CAN THEN BE EXECUTED DIRECTLY

```
java -jar MyJar.jar
```

```
java -jar MyJar.jar
```

THIS WILL CAUSE JAVA TO DO THE FOLLOWING

A. FIND THE MANIFEST OF THE JAR FILE

B. FIND THE LINE IN THE MANIFEST FILE SPECIFYING THE MAINCLASS

C. FIND THE "PUBLIC STATIC VOID MAIN" METHOD
IN THAT CLASS

D. START EXECUTING CODE FROM THERE