# jar包打包流程及其相关说明

1. jar包打包流程：
2. 编写若干“.java”为后缀名的类文件，每个文件第一行为“package PACKAGENAME;”， 将这些文件放在一个叫“PACKAGENAME”的同名文件夹下

【如果文件夹名与package名不同会导致class需要相互调用时不能编译】

1. 运行cmd，用cd命令进入“PACKAGENAME”文件夹目录上一层
2. 将package中的文件编译成.class文件：

“javac -d DIRECTORYPATH PACKAGENAME\FILE1.java PACKAGENAME\FILE2.java......”

1. 若FILE2.java需要调用FILE1.java的class时编译顺序很重要，需要先编译FILE1.java
2. “-d DIRECTORYPATH”意味着将生成的.class文件放在指定位置，d为directory缩写
3. 上述命令效果为：在“DIRECTORYPATH”下创建了一个PACKAGENAME的文件夹，文件夹里面放了所有编译成功的.class文件

【一般.java文件（第一行不是“package PACKAGENAME;”开头的）将成功的.class 文件直接放在“DIRECTORYPATH”下而不创建一个文件夹】

1. “DIRECTORYPATH”可以为“.”意味着在当前目录下实施操作

【如果需要创建的文件夹重名则进行合并操作；即执行命令后所有第一行为“package PACKAGENAME;”的.java和.class文件都会在“PACKAGENAME”文件夹下】

1. 生成jar包：

“jar cvf JARNAME.jar DIRECTORYPATH/\*”

【这个“DIRECTORYPATH”与3中的“DIRECTORYPATH”一般不是一个路径】

【“JARNAME”：jar包名与“PACKAGENAME”：package名可以不同】

1. “c”：创建新jar包
2. “v”：在标准输出中生成详细输出

（在cmd中显示添加了什么文件到jar包中，压缩率是多少，可省略）

1. “f”：指定jar包名
2. “\*”表示将所有文件都压缩进jar包中

【除了“DIRECTORYPATH/\*”当然也可以是

“DIRECTORYPATH1/FILE1.EXTENSION1 DIRECTORYPATH2/FILE2/EXTENSION2 .......”】

1. 效果：在当前目录下生成一个“JARNAME.jar”的jar包，然后根据绝对路径（不包括磁盘名）创建相应文件夹和其中的文件，如果有重名，则生成的jar包不能使用
2. 绝对路径与相对路径：
3. 绝对路径：DIRECTORYPATH=磁盘:/文件夹名/子文件夹名/......
4. 相对路径：DIRECTORYPATH=文件夹名/子文件夹名/.....

即从当前目录（cmd中即最左边“>”之前路径）开始找

1. 为什么引入“javac -d”：

这是因为以“package PACKAGENAME;”开头的文件在类有相互调用时不能直接在当前 目录下直接编译生成.class文件

1. 以“package PACKAGENAME;”开头的文件即使生成了.class文件仍不能直接运行怎么办
2. 运行cmd，用cd命令进入存放.class文件夹（PACKAGENAME）的上一层目录
3. “java PACKAGENAME.CLASSFILENAME”（没有.class后缀名）即可运行

【这里不能使用绝对路径】