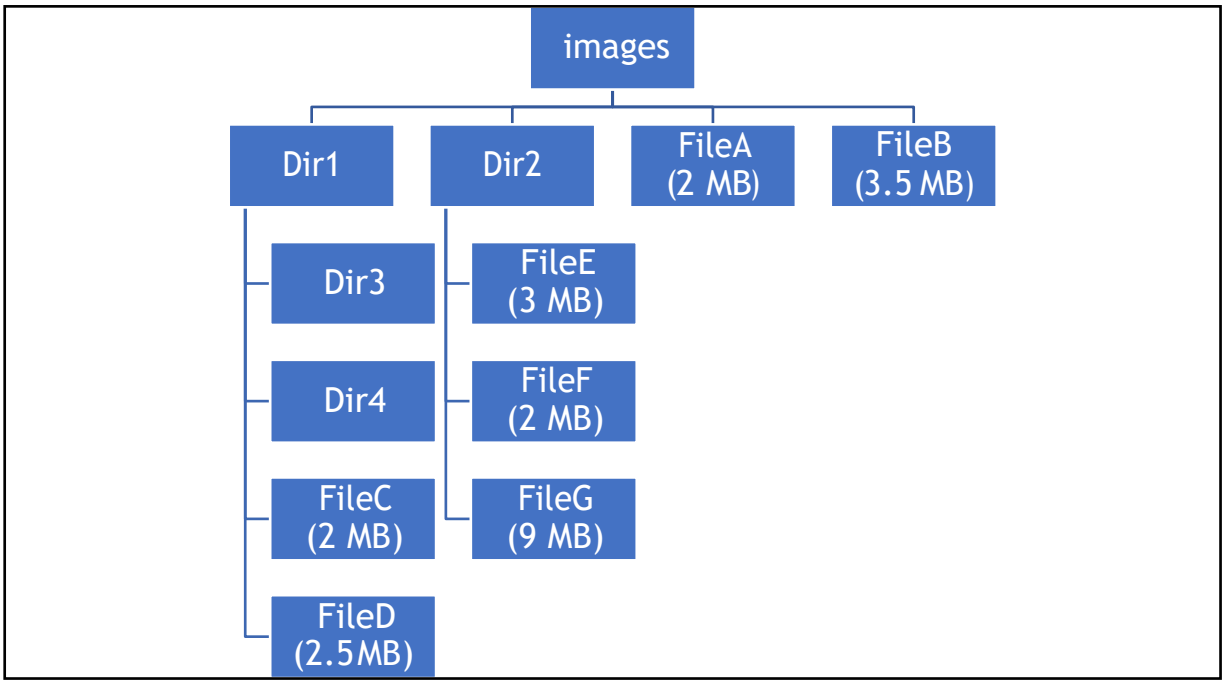
Code Test

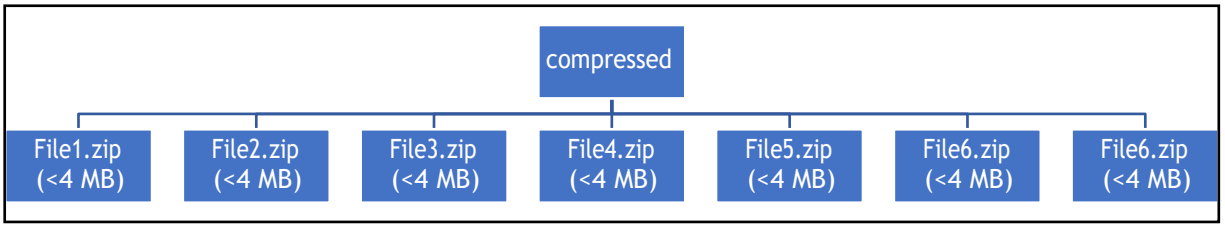
Design and Implement a service that compresses files and folders into a set of compressed files such that of each compressed file doesn't exceed a maximum size. The same program can be used for decompressing the files that it has generated earlier. The output of the decompression should be identical to the original input of the compression process.

When compression is needed, the program to provide REST that takes 3 parameters:

1. Path of Input directory. This directory will contain some files and folders.
2. Path of Output directory. This is where the program should write the compressed files to.
3. Maximum compressed size per file expressed in MB.
4. e.g. POST /compress expected params (inputDir=images, outputDir=compressed, maxSize=4)
5. So in output directory there must be N zipped files of size not more than 4 MB.



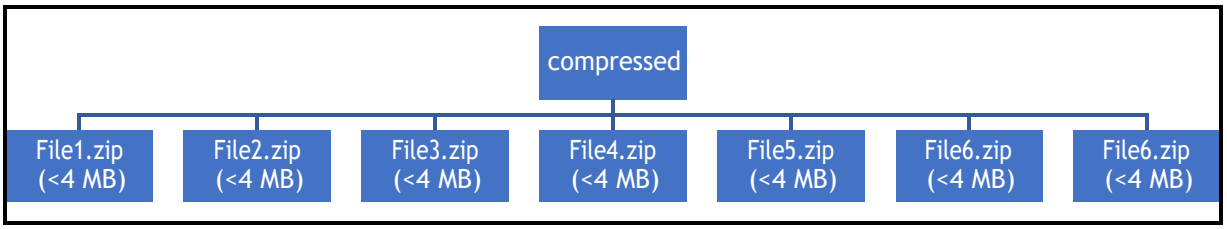
Sample Input Directory



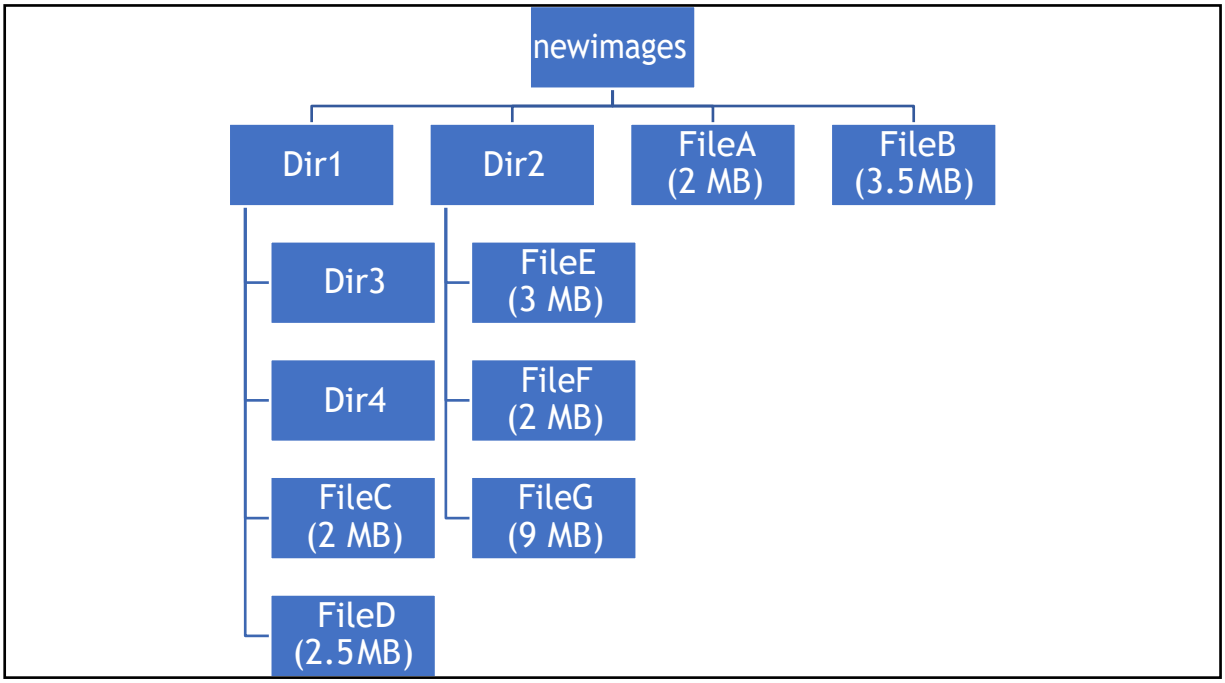
Sample Output Directory

When decompression is needed, the program to provide REST that takes 2 parameters:

1. Path of Input directory. This directory contains the compressed files that were generated by your program.
2. Path of Output directory. This is where the program should put the decompressed files & folders.
3. e.g. POST /decompress expected params (inputDir=compressed, outputDir=newimages)
4. So this newimages directory must me same to same as images directory even with Mth level of sub-directories.



Sample Input Directory



Sample Output Directory

Consider the following when writing the code:

1. Some files may be greater than the JVM memory. (in case you are using Java)
2. Some input files (even when compressed) may be greater than the maximum size allowed for output files. (e.g. what is file is having size more than 4 MB)
3. You can use zip for compression algorithm (in case you are using JDK's implementation **don't use a third party library** for it) but design the program to allow support different compression algorithms in the future.
4. When compressing, generate as few files as possible. You are not required to generate the absolute minimum number of compressed files but doing so would be a BONUS.
5. Another BONUS to make the compression process run in parallel to speed up the compression for single directory. (add another param to REST API for no. of parallel executers)

Please write your production-quality code in Java.

Note: These files and directories are to be present on your server only.