**Demo on Zip creation file and directory level**

1. **Creating ZIP archive:**

ZIP is an archive file format that supports lossless data compression. A ZIP file may contain one or more files or directories that may have been compressed. The ZIP file format permits several compression algorithms, though DEFLATE is the most common.

1. **ZipEntry:**

You have to create a zip entry for each file using the **java.util.zip.ZipEntry** class and before you write the data to stream, you must first put the zip entry object using the **putNextEntry()** method of ZipOutputStream. Once this is done, you can write the data and close the stream.

1. **compressing and archiving**

Since compressing and archiving old log file is an essential housekeeping job in any Java application environment, a Java programmer should know how to compress files in .zip format and then how to read them programmatically if required. The JDK provides full support to create and read ZIP files in Java.

1. **Code for creating zip file**

public class ZipDirectory {

public static void main(String[] args) throws IOException {

String sourceFile = "zipTest";

FileOutputStream fos = new FileOutputStream("dirCompressed.zip");

ZipOutputStream zipOut = new ZipOutputStream(fos);

File fileToZip = new File(sourceFile);

zipFile(fileToZip, fileToZip.getName(), zipOut);

zipOut.close();

fos.close();

}

private static void zipFile(File fileToZip, String fileName, ZipOutputStream zipOut) throws IOException {

if (fileToZip.isHidden()) {

return;

}

if (fileToZip.isDirectory()) {

if (fileName.endsWith("/")) {

zipOut.putNextEntry(new ZipEntry(fileName));

zipOut.closeEntry();

} else {

zipOut.putNextEntry(new ZipEntry(fileName + "/"));

zipOut.closeEntry();

}

File[] children = fileToZip.listFiles();

for (File childFile : children) {

zipFile(childFile, fileName + "/" + childFile.getName(), zipOut);

}

return;

}

FileInputStream fis = new FileInputStream(fileToZip);

ZipEntry zipEntry = new ZipEntry(fileName);

zipOut.putNextEntry(zipEntry);

byte[] bytes = new byte[1024];

int length;

while ((length = fis.read(bytes)) >= 0) {

zipOut.write(bytes, 0, length);

}

fis.close();

}

}

1. **Code for reading zip file**

private static void readUsingZipInputStream() throws IOException {

BufferedInputStream bis = new BufferedInputStream(new FileInputStream(FILE\_NAME));

final ZipInputStream is = new ZipInputStream(bis);

try {

ZipEntry entry;

while ((entry = is.getNextEntry()) != null) {

System.out.printf("File: %s Size %d Modified on %TD %n", entry.getName(), entry.getSize(), new Date(entry.getTime()));

extractEntry(entry, is);

}

} finally {

is.close();

}

}

/\*

\* Utility method to read data from InputStream

\*/

private static void extractEntry(final ZipEntry entry, InputStream is) throws IOException {

String exractedFile = OUTPUT\_DIR + entry.getName();

FileOutputStream fos = null;

try {

fos = new FileOutputStream(exractedFile);

final byte[] buf = new byte[BUFFER\_SIZE];

int read = 0;

int length;

while ((length = is.read(buf, 0, buf.length)) >= 0) {

fos.write(buf, 0, length);

}

} catch (IOException ioex) {

fos.close();

}

}