

## Tutoriel n°1 – Création et utilisation d'une class ZIP

### Introduction

Ce petit dossier est une technique qui vous permettra d'utiliser les bibliothèques de SUN natives au JDK 5 ( ou supérieur ). Vous trouverez la documentation principalement sur le site de SUN. La démonstration ci-après vous montrera comment utiliser ces bibliothèques et de pouvoir « zipper » ou « dézipper » un fichier, un dossier, ou une multitude de fichiers et de dossiers.

### Définition de la classe ZIP

#### *Définition des attributs de la classe:*

```
private String pathZip; // Permet de connaître le chemin du futur ZIP
private List<File> listFilesToZip; // Liste de fichiers à «zipper»
private int level = 9; // Niveau de compression
private ZipOutputStream zout; // Flux de sortie
public static final String ZIP_EXTENSION = ".zip" ; //Extension du fichier
```

#### *Création du constructeur des Getters et Setters :*

```
// Constructeur de la classe
public FileZip(String pathZip) {
    setPathZip(pathZip);
    listFilesToZip = new ArrayList<File> ();
} // FileZip
```

La fonction «setPathZip()» mettra en place le chemin vers lequel le Zip sera enregistré.

```
public void setPathZip(String pathZip) {
    this.pathZip = pathZip;
    try {
        if(null != zout)
            zout = null;
        zout = new ZipOutputStream(new FileOutputStream(pathZip));
    } catch (FileNotFoundException ex) {
        System.exit(-1);
    }
} // setPathZip
```

«setLevel» Montre le niveau de compression à appliquer lorsque l'on souhaitera enregistrer notre Zip. Le niveau de compression d'un Zip est toujours compris de 0 à 9. [ 9 par défaut ]

```
public void setLevel(int level) throws IllegalArgumentException {
    if(1 < level || level > 9)
        throw new IllegalArgumentException("Level must be between 1 and 9");
    this.level = level;
} // setLevel
```

## Tutoriel n°1 – Création et utilisation d'une class ZIP

### Les fonctions «put» et «remove»

Ces fonctions permettent de fournir à la liste «listFilesToZip» des fichiers et / ou dossiers à zipper ou non.

```
public void put(String path) throws FileNotFoundException {
    put(new File(path));
} // put

public void put(File path) throws FileNotFoundException {
    if(!path.exists())
        throw new FileNotFoundException("File not found : "+path.getName());
    listFilesToZip.add(path);
} // put
```

Deux signatures au cas où...

```
public void remove(int index) throws IndexOutOfBoundsException {
    listFilesToZip.remove(index);
} // remove

public void remove(Object o) throws NullPointerException {
    listFilesToZip.remove(o);
} // remove
```

### Enregistrement des fichiers / dossiers dans le flux de sortie

Pour l'enregistrement des données dans le zip nous procéderons de la manière suivante :

- On récupère d'une part le fichier à enregistrer dans un objet **FileInputStream** afin d'en effectuer la lecture et de l'enregistrer sur le flux de sortie.
- On lit ensuite ce fichier par tranche de 1 mo
- On écrit ensuite les données lues directement sur le flux de sortie

```
private void writeStream(File file) throws FileNotFoundException,
    IOException {
    FileInputStream fis = new FileInputStream(file);
    byte [] buff = new byte[ALLOCATE_DEFAULT];
    int octetReaded = -1;
    while( -1 != (octetReaded = fis.read(buff)) ) {
        zout.write(buff, 0, buff.length);
    }
} // writeStream
```

### Les fonctions d'exécutions et de traitements

```
private void zipFile(File file, String currentDir) {
    if(file.isDirectory()) {
```

Tutoriel n°1 – Création et utilisation d'une class ZIP

```

currentDir = currentDir.equals(".") ? file.getName() : currentDir
+ File.separator + file.getName();
try {
    File [] list = FileUtil.getList(file);
    for(File _file : list)
        zipFile(_file, currentDir);
} catch (IOException ex)
    System.out.println("ERREUR : "+file.getName());
} else {
    try {
        currentDir = ".".equals(currentDir) ? "" : currentDir +
            File.separator;
        zout.putNextEntry(new ZipEntry( currentDir + file.getName()));
        writeStream(file);
        zout.closeEntry();
    } catch (IOException ex) {}
}
} // zipFile

```

La précédente fonction sert à zipper des fichiers. Si un repertoire lui ai passé en paramètre, alors la fonction est réursive.

La fonction suivante est l'execution du fichier Zip. Autrement dit, l'enregistrement physique du fichier qui sera fait sur le disque dur ou autres part.

```

public void execute() throws IOException {
    String currentDir = ".";
    zout.setLevel(level);
    for(File file : listFilesToZip) {
        zipFile(file, currentDir);
        currentDir = ".";
    }
    zout.flush();
    zout.close();
} // execute

```

**Exemple d'utilisation de la classe ZIP**

Voici un exemple assez simple d'utilisation de la classe ZIP :

```

FileZip zip = new FileZip("dossier.zip");
zip.put(new File("C:\monFichier.txt"));
zip.put(new File("C:\unDossier"));
zip.setLevel(8);
zip.execute();

```

Et voilà, en cinq lignes vous pouvez créer votre zip.

Voilà la fin de ce premier tutoriel.