Manual técnico conversor

-Por Daniel Álvarez Morales

ÍNDICE

1.- Menú y muestras de directorio..... 2
2.- Excepciones..... 2
3.- Extensiones..... 3
4.- Escritura del fichero..... 4
5.- Conversiones..... 4
5.1.-Formas admisibles..... 4
5.2.- Los 4 pasos para realizar conversiones en general..... 5
6.-Detalles adicionales..... 10

1.- Menú y muestras de directorio

-El menú se compone por un bucle while y un booleano como salida y la información de las carpetas se controlan con “if" si es nulo se exige un directorio sino se lee dicho directorio.

```java
public boolean verRutaDirectorio(File directorio) {
    boolean esNull = true;
    if (directorio != null) {
        // Path ruta = directorio.toPath();

        System.out.println("Carpeta seleccionada: " + directorio.getAbsolutePath());
        return esNull = false;

    } else {
        System.out.println("Selecciona primero un directorio para ver su ruta.");
        return esNull = true;
    }
}
```

2.- Excepciones

-Algunas excepciones existen en el main para controlar que el usuario introduzca letras o números raros en los scanners tipo int (InputMismatchException).

```java
try {
    System.out.print("Seleccione una opción: ");

    opcion = sc.nextInt();
    sc.nextLine();

    volverIntroducir = true;

} catch (InputMismatchException e) {
    System.out.println("Error, entrada incorrecta, asegurate de introducir un numero del menú.");
    sc.nextLine();
    e.printStackTrace();
}
```

-Con esta práctica el programa no se interrumpirá de golpe y será más robusto a fallos.

-La mayoría de excepciones que sirven para detectar errores se lanzan, se detectan y tratan en el main. Simplificando así el control de excepciones.

```
try {

    gestion.conversion(opcion, archivoSeleccionado, directorioSeleccionado,
            gestion.getArchivoTextoGestion(), nombreArchivoSalida);

} catch (IOException e) {
    System.out.println("Error a la hora de convertir fichero o al escribir fichero");
    e.printStackTrace();
}
```

3.- Extensiones

-La extensión se obtiene con el método de la imagen “detectoExtension” que trabaja con un string pasado por parámetro.

```
public String detectorExtension(String nombreArchivo) {
    String[] partes = nombreArchivo.split("\\.");

    String extension = partes[partes.length - 1].toLowerCase();

    return extension;
}
```

-Y con un switch case se llevará a su correspondiente conversion según su extensión y su opción asignada por el usuario que solo pueden ser dos opciones en contraparte de su extensión.

```
switch (extensionAconvertir) {

    case "xml":
        System.out.println("Convirtiendo '" + extension + "' a  '" + extensionAconvertir + "'  ");

        Files.createFile(rutaYSalida);
        List<String> textoConvertido = conversionCsvAxml(texto);
        escribirFichero(rutaYSalida, textoConvertido);

        break;
```

4.- Escritura del fichero

-La escritura se desarrolla en la función “escribirFichero”.

```java
public void escribirFichero(Path ruta, List<String> contenidoPorLinea) throws IOException {
    Files.write(ruta, contenidoPorLinea, StandardOpenOption.APPEND);
}
```

-Los parámetros son ruta (el path del archivo convertido) y contenidoPorLinea (el texto convertido).

5.- Conversiones

-Las funciones de las conversiones son diversas y densas.
Se basan sobre todo en un texto **con una forma que tiene ser concreta** por parámetro que será procesado , reconstruido y devuelto en una estructura dinámica del mismo tipo llamado “textoWork”.

```java
public List<String> conversionCsvAxml(List<String> texto) {

    List<String> textoWork = new ArrayList<>();
```

5.1.-Formas admisibles

-Formas concretas que tienen que tener los archivos para convertir con éxito:

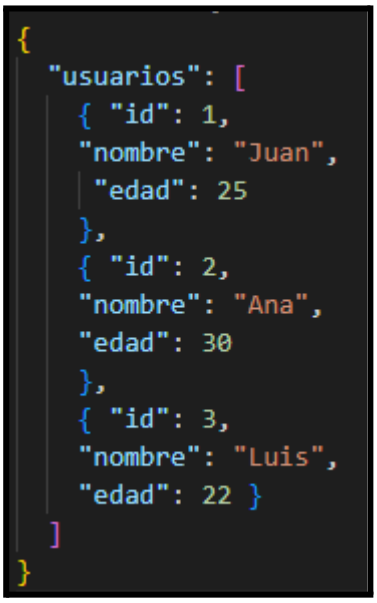
a) Forma archivo vcs:

```
id,nombre,edad
1,Juan,25
2,Ana,30
3,Luis,22
```

b) Forma archivo xml:

```xml
<usuarios>
 <usuario>
     <id>1</id>
     <nombre>Juan</nombre>
     <edad>25</edad>
 </usuario>
</usuarios>
```

c) Forma archivo json:



-Al convertirse, si se vuelven a convertir tienen que tener dicha forma mostrada anteriormente, sino las líneas no se procesarán como es previsto y no se imprimirán en el fichero correctamente.

5.2.- Los 4 pasos para realizar conversiones en general

A) Obtener texto

-En este caso lo obtiene después de leer es como el programa selecciona los archivos y su respectivo texto.

```
List<String> lectura = gestion.leerFichero(nombreFichero, directorioSeleccionado);
archivoSeleccionado = nombreFichero;
```

-Se almacena en la lista de String que tiene la clase “Gestion_Conversor”.

```
public Gestion_Conversor() {

    archivoTextoGestion = new ArrayList<>();
}
```

-Luego se invoca el método llamado “conversión” que se dedica a convertir ficheros.

```
gestion.conversion(opcion, archivoSeleccionado, directorioSeleccionado,
        gestion.getArchivoTextoGestion(), nombreArchivoSalida);
```

-Dependiendo de la extensión se llamará a su procedimiento correspondiente,

```
if (extension.equals("xml")) {
    switch (extensionAconvertir) {
        case "csv":
            System.out.println("Convirtiendo '" + extension + "' a  '" + extensionAconvertir + "'  ");
            Files.createFile(rutaYSalida);
            List<String> textoConvertido = conversionXmlAcsv(texto);
            escribirFichero(rutaYSalida, textoConvertido);

            break;
```

B) Manejar vacíos (El programa puede cometer el error de coger líneas vacías)

-El programa cuando recorre la estructura de datos verifica líneas vacías para evitar excepciones o errores de Arrays, debido a que suele usar splitters en arrays tipo String.

```
// 2. Reagrupar hasta el siguiente objeto
// Empezamos desde la segunda línea (índice 1) para saltar <usuarios>
for (int i = 1; i < texto.size(); i++) {
    String linea = texto.get(i).trim();

    // Verificar si la línea está vacía o nula
    if (linea.isEmpty()) {
        continue; // Ignorar líneas vacías
    }

    if (i != 1 && linea.equals(lineaElemento)) {
        break; // cuando encontramos el segundo <usuario>, paramos
    }
```

C) Dividir entre valores y claves [Por ejemplo Edad (Clave)→ 19 (Valor)]

-El programa usa estructuras auxiliares para llevar a cabo sus procesamientos.
Es decir, una estructura auxiliar que recoge sólo las claves y otra estructura que recoge sólo los valores.

```
// Reagrupando cabecera en estructura dinámica arraylist
List<String> auxCabecera = new ArrayList<>();
```

(Conversión xml a csv)

```
// 1. Guardar la línea que marca el inicio de un objeto (ej: <usuario>)
String lineaElemento = texto.get(1).trim();
```

(Conversión xml a csv)

-Luego de que verifique si está vacío e itera hasta que encuentre el fin del primer elemento es decir hasta que se repite el inicio de la caja de elementos(<Usuario> hasta el otro <Usuario>).

-Lo reprocesa y extrae la primera partes de los splitter de “<>” que es donde están las claves que le interesa para armar la cabecera del csv.
-Guardándolo finalmente en otro auxiliar llamado “auxCabeceraDef”.

```
// 3. Extraer solo las líneas de atributos
List<String> auxCabeceraDef = new ArrayList<>();

for (int i = 1; i < auxCabecera.size() - 1; i++) {
    String linea = auxCabecera.get(i);
    String[] partes = linea.split("[<>]");

    // Verificar que la línea contenga más de un elemento antes de acceder
    if (partes.length > 1) {
        auxCabeceraDef.add(partes[1]); // Esto es el nombre de la etiqueta: id, nombre...
    }
}
```

(Conversión xml a csv)

-Después de la cabecera procesa y recoge los atributos del xml para montar el csv

```java
// Reagrupando datos en estructura estática (cuerpo del csv)
List<String> auxCuerpo = new ArrayList<>();
int saltoLinea = 0;
for (String linea : texto) {
    String[] partes = linea.split("<[^>]+>"); // Divide por cualquier etiqueta
    for (int i = 0; i < partes.length; i++) {
        // Verificar si la parte está vacía antes de agregarla
        if (partes[i].trim().equals("")) {
            continue; // Esta línea elimina elementos vacíos o espaciados
        }

        saltoLinea++;
        if (saltoLinea == numeroAtributos) {
            auxCuerpo.add(partes[i].trim());
            saltoLinea = 0;
        } else {
            auxCuerpo.add(partes[i].trim() + ",");
        }
    }
}
```

(Conversión xml a csv)

D) Insertar en la estructura definitiva “textoWork” (el archivo convertido)

-Con un bucle for el programa inserta la cabecera del csv es decir una sola línea

```java
// Contando atributos para hacer saltos al insertar
for (String etiqueta : auxCabeceraDef) {
    numeroAtributos++;
}

// Reagrupando datos en estructura estática (cuerpo del csv)
List<String> auxCuerpo = new ArrayList<>();
int saltoLinea = 0;
for (String linea : texto) {
    String[] partes = linea.split("<[^>]+>"); // Divide por cualquier etiqueta
    for (int i = 0; i < partes.length; i++) {
        // Verificar si la parte está vacía antes de agregarla
        if (partes[i].trim().equals("")) {
            continue; // Esta línea elimina elementos vacíos o espaciados
        }

        saltoLinea++;
        if (saltoLinea == numeroAtributos) {
            auxCuerpo.add(partes[i].trim());
            saltoLinea = 0;
        } else {
            auxCuerpo.add(partes[i].trim() + ",");
        }
    }
}
```

(Conversión xml a csv)

-Con un bucle for y contadores externos con la intención de iterar a medida del csv, el programa inserta el cuerpo del csv es decir los atributos.

```
// Contando atributos para hacer saltos al insertar
for (String etiqueta : auxCabeceraDef) {
    numeroAtributos++;
}
```

(Conversión xml a csv)

```
// Insertando cuerpo
String cuerpo = "";
int saltoPagina = 0;

for (String string : auxCuerpo) {
    cuerpo += string;
    saltoPagina++;

    if (saltoPagina == numeroAtributos) {
        textoWork.add(cuerpo);
        cuerpo = "";
        saltoPagina = 0;
    }
}
```

(Conversión xml a csv)

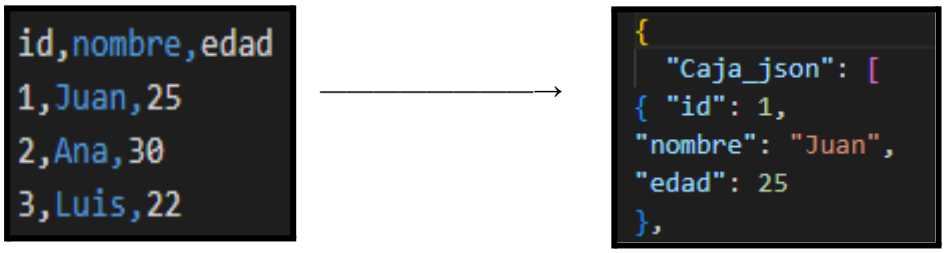
-Y finalmente lo devuelve para que la función “escribirFichero” cumpla su cometido.

```
return textoWork;
```

(Conversión xml a csv)

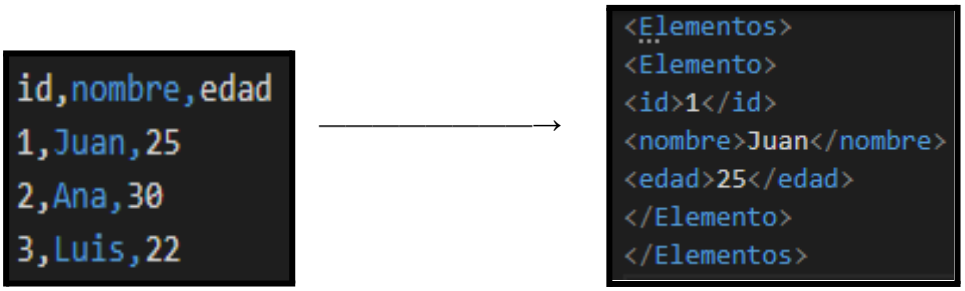
6.-Detalles adicionales

-También cuando el programa pasa un csv o xml a json el nombre de la caja del json lo asigna un nombre estándar que tiene el programa para los json "Caja_json", ya que por ejemplo el csv no posee un nombre de tabla como tal sino que son solo tuplas.



(Fichero original csv) *(Fichero json de origen csv)*

-Lo mismo ocurre cuando por ejemplo un csv o json pasa a xml, para las etiquetas padres del xml el programa les asigna “<Elementos><Elemento></Elemento></Elementos> ”.



(Fichero original csv) *(Fichero xml de origen csv)*