DOCUMENTACIÓN TAREA 2.2

-DÍA 1: A primera vista esta me parece que es la practica que más cambia y que más pide de momento, ya que es una ampliación de la anterior en la que se pide que cuando se dividen los bloques en vez de guardarlos en un ".txt" se pide guardarlos en un archivo ".XML" con una estructura determinada pero dinámica en función del bloque ademas de pedir algunos datos calculados que puede que den algunos problemas.

Para comenzar he repasado como debe ser la estructura de un archivo ".XML" y como se forman ademas de descargarme un programa llamado "XML Copy Editor" el cual permite la visualización de estos archivos ademas de generar documentos ".DTD" a partir de los ".XML" y viceversa.

Usando la librería DOM en java he conseguido que mi aplicación me genere los archivos ".XML" con la estructura que yo deseaba ademas de generar un fichero por cada bloque. Esto lo he conseguido metiendo la el código de escritura XML en la función de escritura que previamente usaba para generar los ".txt".

-DÍA 2: Hoy me he propuesto generar esos ficheros con la estructura del nombre que exige la practica y el primer problema que me he encontrado ha sido que el campo ciclo tiene valor tan solo en la segunda linea y el fichero se genera según los datos de la primera linea (cabecera), con lo cual de alguna manera tenia que conseguir el dato de la segunda linea y volver a la primera y comenzar a escribir o averiguar el valor del dato antes de escribirlo. David me ha explicado que para ello se puede usar programación concurrente pero también me ha dicho que es un mundo muy complejo y que no me meta, con lo cual yo había pensado en usar punteros de alguna manera para conseguir el dato antes de comenzar a escribir pero no conseguía entender demasiado bien esto y decidí ir por el camino facil y estructurar mi programa en 2 bucles principales, uno de lectura para obtener todos los datos necesarios y almacenarlos, y otro bucle de escritura donde volcar todos los datos previamente almacenados en el fichero que se vaya a generar. Al final del día he conseguido crear esta estructura y hacer que funcione correctamente, generando así los nombre de los ficheros.

-DÍA 3: Mi primer pensamiento hoy ha sido obtener otro dato calculado más, en este caso el numero de lineas de cada bloque, para ello he pensado en almacenarlo en un "array" usando como indice el numero de cabecera, el cual obtengo en el bucle de escritura, el problema es que para generar el "array" hace falta declarar su tamaño y este lo averiguo tras el bucle de escritura con lo cual he pensado en hacer otro bucle más para obtener especificamente todos los datos calculados excepto el numero de cabeceras que hay en el fichero ademas de el numero de lineas totales que lo averiguaría en el primer bucle. Una vez modificada esta estructura y generando el

“array” correctamente he conseguido obtener este dato. Para evitar el uso de otro bucle más investigue el “arraylist” e intente implementarlo pero no conseguí entenderlo del todo bien y me decidí a usar un “array” los cuales entiendo mejor y me podrían ahorrar algunos problemas más adelante.

-DÍA 4: Usando la misma filosofía del día anterior he conseguido obtener el resto de datos calculados y también he implementado la opción de que si no hay ningun campo que se llame “IMPORTE” que no muestre el dato “TOTAL_IMPORTE” dentro del documento “.XML”. Tras el día de hoy lo único que me queda es conseguir el valor de cada dato dentro de cada campo y volcarlo en el documento.

-DÍA 5: A la hora de almacenar el valor de todos los datos en un “array” se me ocurrió la idea de usar un “array tridimensional” (con tres indices). Para ello un indice sería el numero del campo, otro sería el numero de filas (en función del numero del campo), y el último índice el numero de columna (en función del numero de campo). Con la estructura creada parecía que se almacenaba correctamente, pero no sabía muy bien como mostrar el valor de ese “array” con lo cual abandone esa idea por que me quede pillado y no conseguí ningun avance en este día con lo cual intentaré usar un “array bidimensional” o alguna otra estructura.

-DÍA 7: Viendo que el día anterior no he conseguido ningun resultado me plantee que de alguna manera la función de escritura leyese linea por linea el fichero y los mostrase, pero ademas de encontrarme con varios errores solo me rellenaba los datos de la última linea, algo similar a lo que me pasaba en la practica anterior cuando no usaba el “true” dentro del “buffered writer” con lo cual esto me ha hecho pensar que quizas exista algo similar en la librería “DOM”.

-DÍA 8: Tras ver que no existía nada similar al “true” del “buffered writer” volví a intentar crear un “array tridimensional”, solo que esta vez si que he encontrado por internet como mostrar el contenido de este y he conseguido finalizar la practica. A continuación muestro una captura del “array tridimensional”.


```
int numCab = 0;
int[] conLinBloq = new int[resultado + 1];
int[] conCamposBloq = new int[resultado + 1];
int[] posicionImporte = new int [resultado + 1];
double[] importeTotal = new double [resultado + 1];


//BUCLE PARA OBTENER EL NUMERO DE LINEAS DE CADA BLOQUE, DE CAMPOS,
//POSICION DEL IMPORTE EN LOS BLOQUES Y EL IMPORTE TOTAL.
while (linea3 != null) {
    linea3 = lector3.readLine();
    if (linea3 == null){
        lector3.close();
        break;
    }

    numCab += Trata_Ficheros.trataFicheros(linea3,nombre);
    conLinBloq[numCab] += Trata_Ficheros.nLineaFichero(linea3,nombre);
    conCamposBloq[numCab] += Trata_Ficheros.nCamposFichero(linea3,nombre);
    posicionImporte[numCab] += Trata_Ficheros.posicionImporte(linea3, nombre, numCab);
    importeTotal[numCab] += Trata_Ficheros.importeTotalBloques(linea3, nombre, posicionImporte, numCab);
}


int numCab4 = 0;
int[] conLinBloq2 = new int[resultado + 1];
int[] conCamposBloq2 = new int[resultado + 1];
String [][][] valCampo = new String [resultado + 1] [conLinBloq[resultado]] [conCamposBloq[resultado]];


while (linea4 != null) {
    linea4 = lector4.readLine();

    if (linea4 == null){
        lector4.close();
        break;
    }

    numCab4 += Trata_Ficheros.trataFicheros(linea4,nombre);
    conLinBloq2[numCab4] += Trata_Ficheros.nLineaFichero(linea4,nombre);
    conCamposBloq2[numCab4] += Trata_Ficheros.nCamposFichero(linea4,nombre);
    valCampo [numCab4] [conLinBloq2[numCab4]-1] = Trata_Ficheros.valorCampo(linea4, conCamposBloq2, numCab4);


}
```