Contenido

[Introducción 1](#_Toc527286004)

[Antecedentes 1](#_Toc527286005)

[Entorno de Desarrollo 1](#_Toc527286006)

[Enunciado Practico 1](#_Toc527286007)

[Explicación del problema a resolver 1](#_Toc527286008)

[Solución 2](#_Toc527286009)

[Descripción y justificación de la solución 2](#_Toc527286010)

[Código fuente de la aplicación e interfaces 2](#_Toc527286011)

[Explicación 3](#_Toc527286012)

[Algoritmos 3](#_Toc527286013)

[Clases 3](#_Toc527286014)

[Estructuras de datos 4](#_Toc527286015)

[Listado de pruebas 4](#_Toc527286016)

[Entrada 4](#_Toc527286017)

[Ejecución 4](#_Toc527286018)

[Salida 4](#_Toc527286019)

[Conclusiones 4](#_Toc527286020)

# Introducción

## Antecedentes

El programa ha sido diseñado en un ordenador con sistema operativo Windows 7 con 8Gb de memoria RAM, se ha necesitado la creación de varias carpetas y ficheros

## Entorno de Desarrollo

El entorno de Desarrollo ha sido java con eclipse.

# Enunciado Practico

## Explicación del problema a resolver

* Supóngase que en el pc existe una carpeta Usuarios que contiene una subcarpeta por cada uno de ellos, desde Usuario1 hasta Usuario5. A su vez, cada carpeta contiene al menos dos subcarpetas y ficheros en cada una de ellas.
* Crear un programa en java que cree varios procesos en paralelo.
* Cada uno de ellos debe listar el contenido de la carpeta del usuario que recibe como argumento y escribir en un fichero de salida los archivos listados.
* El final de la ejecución devuelve un fichero por cada carpeta de usuario.
* Una vez finalizados todos los hijos el padre imprimirá los ficheros por pantalla secuencialmente.

# Solución

## Descripción y justificación de la solución

Se ha tenido que crear una carpeta padre que contenga otras cinco carpetas y a su vez cada una de estas carpetas contiene dos ficheros y dos carpetas más.

La utilización de arrays es debido a un ahorro de espacio en el código y mejor utilización de los recursos

## Código fuente de la aplicación e interfaces

**import** java.io.\*;

**public** **class** Trabajo3 {

**public** **static** **final** String ***RUTA*** = "C:\\Users\\Alberto\\Desktop\\Usuarios\\";

**public** **static** **final** String ***USU***[] = { "Usuario1", "Usuario2", "Usuario3", "Usuario4", "Usuario5" };

**public** **static** **final** **int** ***NUM\_USU*** = 5;

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

ProcessBuilder pb[] = **new** ProcessBuilder[***NUM\_USU***];

File fOutUsu[] = **new** File[***NUM\_USU***];

File fErr[] = **new** File[***NUM\_USU***];

Process pr[] = **new** Process[***NUM\_USU***];

**for** (**int** i = 0; i <= ***NUM\_USU*** - 1; i++) {

pb[i] = **new** ProcessBuilder("cmd", "/C", "cd " + ***RUTA*** + ***USU***[i] + "& dir");

fOutUsu[i] = **new** File(***RUTA*** + ***USU***[i] + "salida.txt");

fErr[i] = **new** File(***RUTA*** + ***USU***[i] + "error.txt");

pb[i].redirectOutput(fOutUsu[i]);

pb[i].redirectError(fErr[i]);

pb[i].start();

}

**for** (**int** i = 0; i <= ***NUM\_USU*** - 1; i++) {

pr[i] = pb[i].start();

}

**for** (**int** i = 0; i <= ***NUM\_USU*** - 1; i++) {

pr[i].waitFor();

}

FileReader f[] = **new** FileReader[***NUM\_USU***];

BufferedReader linea[] = **new** BufferedReader[***NUM\_USU***];

String cadena = " ";

**try** {

**for** (**int** i = 0; i <= ***NUM\_USU*** - 1; i++) {

f[i] = **new** FileReader(***RUTA*** + ***USU***[i] + "salida.txt");

linea[i] = **new** BufferedReader(f[i]);

cadena = linea[i].readLine();

**while** (cadena != **null**) {

System.***out***.println(cadena);

cadena = linea[i].readLine();

}

f[i].close();

linea[i].close();

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

# Explicación

## Algoritmos

La utilización de varios for es debida a la necesidad de repetir el mismo código un número repetido de veces concretas, mientras que la utilización del while es debido a que no sabemos el número de líneas que va a tener los ficheros

## Clases

Nada más que se ha utilizada una clase principal

## Estructuras de datos

La estructura de datos en su mayoría es con arrays ya que así se puede manejar dodo con suma rapidez

# Listado de pruebas

## Entrada

No hay datos de entrada que analizar lo único que se ha podido analizar es si existen o no los directorios a los cuales nos referimos

## Ejecución

Las pruebas en ejecución se han vasado en comprobar que no exista desbordamiento y que los bucles funcionen bien.

## Salida

Las pruebas a la hora de la salida de datos han sido comprobar que los ficheros existen.

# Conclusiones

Un buen repaso para la utilización de ficheros.