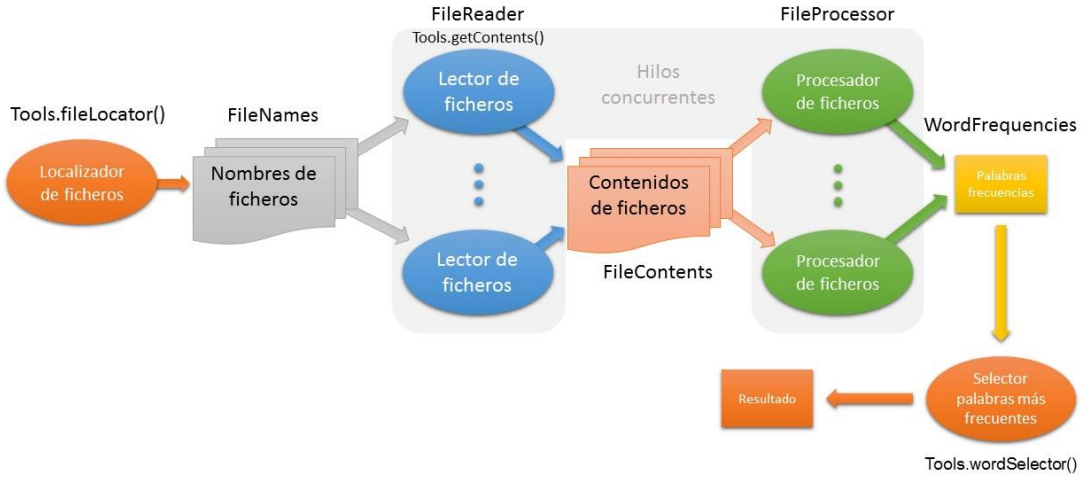
Procesamiento de ficheros de texto

Se desea construir en Java un sistema que procese de forma concurrente conjuntos de ficheros de texto. El objetivo del sistema es: dado un conjunto de ficheros, procesarlos y obtener la lista de las 10 palabras más frecuentes, ordenadas de mayor a menor.



Estructura general

Los ficheros a procesar, que contienen texto sin formato, se encuentran en el directorio **datos** del *HOME* del usuario actual.

El sistema realiza las siguientes acciones:

- Identifica el conjunto de ficheros a procesar —ya implementado en **Tools.fileLocator()**— y almacena sus nombres en un objeto de la clase **FileNames**.
- Varios hilos de la clase **FileReader** van tomando uno a uno los nombres de fichero de un objeto **FileNames** y se van encargando de leer los correspondientes ficheros y añadir su contenido a un objeto de la clase **FileContents**.
- Varios hilos de la clase **FileProcessor** van leyendo uno a uno el contenido de los ficheros de un objeto **FileContents** y lo procesan, separando las palabras y contabilizando su frecuencia, actualizando el resultado sobre un "contador" común de la clase **WordFrequencies**.
- Se ordenan y seleccionan las 10 palabras más frecuentes partiendo de un objeto **WordFrequencies** —ya implementado en **Tools.wordSelector()**.

Para construir el sistema se deben crear distintas clases que representen los distintos procesos y contenedores de datos que maneja el sistema. En concreto se tienen las siguientes clases, en las que se debe mantener la interfaz suministrada.

Clase FileNames

Un objeto de esta clase almacena temporalmente los nombres de los ficheros a procesar. El método **Tools.fileLocator()** se encarga de añadir nombres de ficheros, y los hilos **FileReader** se encargan de consumirlos. El objeto de la clase **FileNames** dispone de un método para establecer cuándo ha terminado su función, dejando entonces de aceptar más nombres de ficheros, así como no hacer que se espere por más nombres.

Métodos	**Descripción**
void **addName**(String fileName)	Almacena un nuevo nombre de fichero en el objeto.
String **getName**()	Extrae del objeto y devuelve un nombre de fichero. Cuando no hay ficheros disponibles el hilo solicitante debe esperar a que cambie la situación, salvo que además se haya llamado a **noMoreNames**(), en cuyo caso inmediatamente se devuelve **null**. Por tanto, **null** indicará que no se van a devolver más nombres de ficheros.
void **noMoreNames**()	Da lugar a que el objeto no admita más nombres de fichero.

Clase FileContents

Un objeto de esta clase se encarga de almacenar y suministrar ristras, cada una de las cuales contendrá toda la información textual de uno de los ficheros. El número de ristras, así como el tamaño total de estas, estarán limitados en el momento de la construcción del objeto. Los hilos **FileReader** se encargarán de ir añadiendo los datos al objeto, y los hilos **FileProcessor** se encargarán de sacarlos. El objeto dispone de un mecanismo para saber cuándo deja de ser funcional, consistente en que los hilos **FileReader** indican cuándo comienzan a producir contenido, y cuándo terminan, de tal forma que el objeto dejará de ser funcional cuando no quede ningún **FileReader** y no haya contenido que devolver.

Métodos	Descripción
FileContents(int maxFiles, int maxChars)	Inicializa estableciendo el número máximo de ficheros que se puede almacenar y el tamaño total máximo de las ristras que se pueden almacenar en un momento dado.
void **registerWriter**()	Se llama para indicar que hay un nuevo **FileReader** usando el objeto.
void **unregisterWriter**()	Indica que un **FileReader** ha dejado de producir contenido. Cuando el número de **FileReader** registrados pase de 1 a 0 el objeto ya no recibirá más contenido.
void **addContents**(String contents)	Llamado por hilo **FileReader** para añadir el contenido de un fichero. Si se ha alcanzado el límite de ficheros o fuera a superarse el máximo establecido para el tamaño total de contenidos, el hilo se espera hasta que cambie la situación. El límite de tamaño máximo de contenido no se aplica si no hay contenido almacenado.
String **getContents**()	Llamado por hilo **FileProcessor** para extraer el contenido de un fichero. Si no hay contenido disponible, el hilo se espera a que lo haya, salvo que además el último **FileReader** se ha desregistrado, en cuyo caso inmediatamente devuelve null.

Clase FileReader

Hilo que se encarga iterativamente de obtener un nombre de fichero de un **FileNames**, leer su contenido y almacenarlo en un **FileContents**. Termina si el **FileNames** le devuelve **null**. Para facilitar la implementación de esta clase se dispone del método **Tools.getContents**(String fname) al que, si se le pasa un nombre de fichero, devuelve su contenido.

Métodos	Descripción
FileReader(FileNames fn, FileContents fc)	Inicializa un objeto estableciendo el **FileName** y **FileContents** que usará en su ejecución.

Clase FileProcessor

Hilo que se encarga iterativamente de leer un contenido de fichero de un **FileContents** y procesarlo (separar y contabilizar sus distintas palabras), almacenando el resultado en un objeto **WordFrequencies**. Termina si el **FileContents** le devuelve **null**. Una ***palabra*** se define como una secuencia

contigua de caracteres alfanuméricos (del idioma español) delimitadaE por espaciador, símbolo o extremo del fichero.

Métodos	Descripción
FileProcesor(FileContents fc, WordFrequencies wf)	Inicializa un objeto estableciendo el **FileContents** y **WordFrequencies** que usará en su ejecución.

Clase WordFrequencies

Un objeto de esta clase almacenará las palabras encontradas y sus correspondientes frecuencias acumuladas correspondientes al conjunto de los ficheros ya procesados.

Métodos	Descripción
void **addFrequencies**(Map<String,Integer> f)	Añade al objeto actual la información de palabras/frecuencias pasada.
HashMap<String,Integer> **getFrequencies**()	Devuelve las parejas de palabras/frecuencias acumuladas.

Main.main

El programa principal debe completarse para que cree 2 hilos **FileReader** y 3 hilos **FileProcessor**, así como para que espere a que terminen para poder él mismo entonces terminar.