

Enfoque adoptado a la resolución de la Tarea 1

He pensado en que la resolución de la Tarea 1, puede ser almacenar en un array todos las palabras que haya en el fichero que estamos leyendo, para así comparar cada valor del array con la palabra clave que introduzcamos, si coincide, se incrementaría una variable específicamente creada para contar las coincidencias, y esta se mostraría. He pensado que al tratarse de manipulación de archivos tabulados es más sencillo realizar este ejercicio, ya que cada campo está bien remarcado, y NO van a existir filas con columnas irregulares (1 fila con 3 columnas, 2 fila con 5, etc) para ser almacenado en un array, mi forma de ver el ejercicio no puede encontrar cadenas dentro palabras, quiero decir, no puedo buscar "labr" dentro de la palabra "palabra" pero si puedo encontrar la palabra "palabra" dentro de una frase. Respecto a la Orientación a objetos aun no se como orientarlo, quizá cada línea del fichero siendo un objeto.

He aprendido:

- Uso básico de ShellScript (variables y comandos)
- Que `\\s` en Java se utiliza como "espacio en blanco" y `\\s+` como "varios espacios en blanco".
- He usado `.split(,)` para separar una variable String en una array de palabras separadas por comas en este caso.

Navegando por Internet ha llegado a mi interés el tema de las listas, he visto que las listas son bastantes más flexibles para el enfoque que le quería dar a la tarea, ya que yo calculaba previamente el número de filas y columnas para averiguar la capacidad del array donde quería almacenar las palabras, pero con las listas no tengo que indicar capacidad total, ni tengo que sobrescribir nada ya que puedo ir aumentando los valores de la lista cuando quiera, he usado fundamentos básicos de listas como: `.get()` o `.add()`, y también trasteado otras opciones muy útiles que trae como `.size()` que dice el tamaño de la lista (similar a `length` en los array), o `.contains("Gema")` que me devuelve true si encuentra la palabra "Gema" en la lista almacenada o false si no.

Cabe mencionar que antes de empezar esta tarea no he tenido experiencia con el procesado de ficheros y ahora tengo un poco más de idea.

Primer enfoque POO

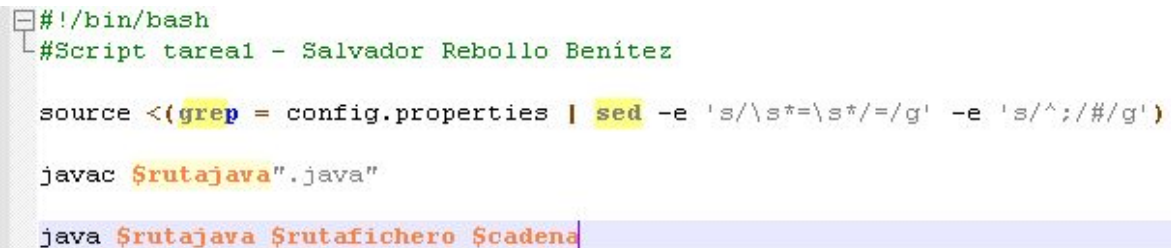
Mi primera versión de este proyecto enfoca la programación orientada a objetos de una manera bastante simple, los objetos únicamente tienen un atributo de instancia (tipo String) en el cual almaceno el nombre del fichero tabulado (.csv) en cuestión, he incluido un par de método funcionales que me devuelven el número de repeticiones halladas de una palabra clave pasada como parámetro. Uno de los métodos únicamente me devuelve una variable tipo int con todas las repeticiones, el otro me imprime un par de líneas diciéndome de forma visual cuantas repeticiones ha hallado. Por otra parte el `toString` que he hecho me imprime el fichero en cuestión.

Enfoque dado de uso de parámetros y argumentos

Mi primer borrador consistía en usar variables (las cuales tenía que cambiar manualmente y por lo tanto entrar al código del script cada vez que lo quisiera hacer). El enfoque dado a posteriori ha tenido una pequeña mejora, la cual consiste en que el script leerá un fichero extra de propiedades (en mi caso llamado “**config.properties**”, aunque por lo que he visto hay más formatos de ficheros clave=valor como el .ini o .conf) dicho fichero contiene los parámetros que el script va a coger, y los consigo pasar de la siguiente manera:

```
#!/bin/bash
#Script tarea1 - Salvador Rebollo Benítez

source <(grep = config.properties | sed -e 's/\s*=\s*/=/g' -e 's/^\s*;/#/g')
```



```
javac $rutajava\".java\"
java $rutajava $ruta fichero $cadena
```

Para que mi script consiguiera leer el fichero properties use la cuarta línea (source) y la ayuda de las pipes (tuberías) para concatenar comandos linux.

1. Primero con grep: Busca todas las líneas halladas en el fichero que contengan el caracter “=”, así me llevare solo las declaraciones de variables (que es lo que me interesa)
2. El uso de sed (sin profundizar demasiado en expresiones regulares) nos permite, de una forma cómoda, borrar líneas, registros o sustituir cadenas de caracteres dentro de las líneas. La expresión “-e” nos proporciona una forma de sencilla de separar los comandos del sed (aquí es utilizado para separar 2 expresiones regulares).
 - a. La primera expresión regular usada sirve para quitar los espacios que pudiera haber alrededor del signo “=”, si por ejemplo tuviera escrito en el fichero de parámetros las líneas así: “clave = parámetro” a bash no le gusta eso, así que me quitaría los espacios y quedaría así: “clave=parámetro”.
 - b. El uso de la segunda expresión regular cambiará los ; por # sólo cuando una línea empiece por ; Todo eso se evaluará para extraer las variables.

Con esta implementación desde el script puedo usar las variables como si fueran parte de ese mismo archivo, pero no existen ahí, si no en el fichero de parámetros. Luego simplemente uso las variables como argumentos para compilar y ejecutar el programa java.