Técnico Superior en Desarrollo de Aplicaciones Web

Práctica: Ficheros en JAVA - MB

Lee bien todo lo que se indica en este enunciado y procura respetarlo.

Si tienes alguna duda o crees que hay alguna incoherencia, avisa al profesor para que te la solucione.

Normas de Realización

a) El ejercicio se realizará en clase en el tiempo indicado para ello. No se va a dejar tiempo para acabarlo en casa.

b) Los ejercicios deben realizarse en archivos .java separados . Cada archivo se nombrará como corresponda en cada ejercicio. **Indica tu nombre y apellidos como comentario en la primera línea de código de cada ejercicio.**

c) Puedes consultar tus apuntes y fuentes de internet para realizar el ejercicio.

d) No se pueden utilizar herramientas que generen código para completar la práctica. Si se detecta dicho uso, se considerará plagio.

e) Sólo se permite usar las herramientas del lenguaje que se han dado hasta este momento. (*Recuerda: yo evalúo competencias, no sólo conocimientos*).

f) **Se tendrá muy en cuenta la legibilidad del código**: sangrado de lineas correcto, uso de comentarios y nombre de variables adecuado.

Copia o plagio

Tal y como se indica en los estatutos del centro, está totalmente prohibido copiar código de otro/a compañero/a y/o de internet, incluidas herramientas que generan código. Si se detecta y se demuestra este tipo de comportamiento, la calificación de la actividad será de 0 puntos.

Entrega

El ejercicio puntuará para la evaluación práctica del 2º Trimestre de la asignatura.

- Al finalizar el ejercicio debes entregar en la plataforma un archivo comprimido en .zip .rar o.7z con todos los ficheros **.java** de los ejercicios que completes. Nombra ese archivo con tu primer apellido y con tu nombre y el modelo de práctica que realices: Ej: `Ramos_JoseFrancisco_MB.`

- **Recuerda:** se puntúa lo que se entrega en la plataforma. Además, si un ejercicio no compila, ese ejercicio tendrá 0 puntos.

Muy Importante

Hay un ejemplo de ejecución al final de cada enunciado. **Respeta la salida de mensajes de cara al usuario como se indica.** Si no entiendes algo, pregunta. No te inventes nada.

Todos los archivos de texto que vayan a leerse o a crearse **deben estar en el mismo paquete que el archivo .java** correspondiente (tal y como se ha visto en clase). Si se colocan en otro sitio, el ejercicio será incorrecto.

Para resolver esta práctica, crea un **paquete llamado T2P3** y genera ahí los archivos necesarios (tanto .java como los de texto).

No pongas triste a tu profesor:

Ya que tu profesor se ha molestado en explicarte lo mejor posible cada enunciado, en ponerte ejemplos de ejecución junto con pantallazos de posibles salidas de cada ejercicio y en decirte qué es importante, no pongas triste a tu profe y LEE BIEN TODO. Respeta exactamente lo que se te pide. **Si eso no ocurre y tu profe se pone triste, te penalizará con 0,5 puntos por cada elemento no respetado.**

EJERCICIO 1 *(3 puntos)*

Realiza un programa que pida al usuario dos números: L y N. Esos números deben ser enteros y mayores a 2. Si no se cumple alguna de esas condiciones, el número en cuestión debe volver a pedirse.

A continuación debe crear el archivo `numeros.txt` y llenarlo de la siguiente forma:

- El archivo debe tener tantas líneas como indique el valor L.
- En cada linea deben aparecer separados por coma (,) tantos números aleatorios entre 3 y 20 como indique el valor N

Si el fichero se crea correctamente, debe indicarse por pantalla.

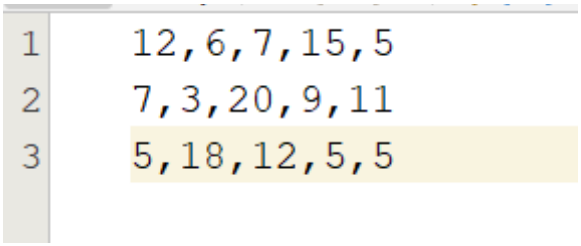
Ejemplo de ejecución:

```
Numero de lineas del fichero: -8
Valores mayores a 2!!
Numero de líneas del fichero: 3

Cantidad de números por linea: muchos
ERROR: debes introducir un número!!
Cantidad de números por linea: 5


El fichero numeros.txt se ha generado con éxito.
```

Un posible contenido del fichero `numeros.txt` según el ejemplo anterior sería:



3 filas con 5 números por fila.

EJERCICIO 2 *(3 puntos)*

Haciendo uso del fichero `numeros.txt` creado en el ejercicio anterior (si no has realizado el ejercicio anterior, el profesor te facilitará uno para este ejercicio), calcula los siguientes valores:

- El mayor número de todo el ficheros.
- El menor número de todo el fichero.
- La suma de todos los números del fichero.
- La media (con decimales) de todos los números del fichero.

Fijate bien en el ejemplo siguiente y respeta los mensajes de interacción con el usuario.

Para el fichero `numeros.txt` del ejemplo anterior, tendríamos

```
Obteniendo datos del fichero numeros.txt...
Mayor valor: 20
Menor valor: 3
Suma total: 140
Media: 9.33
```

Nota: si sabes definir funciones en JAVA, puedes ordenar el código usando funciones.

EJERCICIO 3 *(4 puntos)*

Introducción

La tabla Unicode es un estándar internacional de codificación de caracteres que asigna un número único a cada carácter. Así, por ejemplo, para los siguientes caracteres tenemos sus números asociados:

- A: 65, B: 66, C: 67, … Z:90
- a: 97, b:98, c: 99, … z:122
- 0: 48, 1: 49, 2: 50, ...9: 57
- Salto de línea: 10
- Espacio en blanco: 32

La tabla Unicode permite que los caracteres sean representados de manera consistente en diferentes sistemas informáticos y lenguajes de programación, incluido JAVA.

Objetivo

Con la información anterior, crea un programa que lea un fichero de texto indicado por el usuario y genere otro fichero usando un algoritmo de codificación sencillo que se explica más adelante.

Funcionamiento

Lo primero que debe hacer el programa es pedir al usuario el nombre (y la extensión) del fichero que quiere codificar.

Después debe crear el fichero de salida. Este fichero tendrá el mismo nombre que el fichero de entrada y como extensión `.cod`

A continuación, irá leyendo el contenido del fichero de entrada, codificando lo que encuentre (según el algoritmo que se explican mas adelante) y volcando el resultado en el fichero de salida.

El usuario debe ir enterándose de todo lo que va pasando (ver ejemplo de ejecución).

Al final deben existir dos archivos: el archivo con el texto original (legible) y el archivo .cod con el texto codificado.

Algoritmo de codificación

El algoritmo de codificación que vamos a usar es muy sencillo y hace uso de los valores de la tabla Unicode que aparecen en la Introducción:

- Se codifica cada carácter
- Los saltos de línea se mantienen y no reciben ningún cambio.
- Las letras (tanto minúsculas como mayúsculas) se mueven un valor hacia adelante en la tabla unicode: la 'a' pasa a ser la 'b', la 'b' pasa a ser la 'c', ... Esto debe hacerse de forma circular, es decir, la 'z' debe pasar a ser la 'a', no el carácter siguiente a la 'z' (que no será una letra, cuidado!).
- Los espacios en blanco serán sustituidos por el carácter ']' (corchete de cierre):*Esa carácter debe aparecer tal cual en el código, no quiero que en esta parte del algoritmo me pongas el valor de este carácter en la tabla unicode.*
- Todos los dígitos que aparezcan deben ser sustituidos por la letra 'Ç' (C con cedilla mayúscula). *Esa carácter debe aparecer tal cual en el código, no quiero que en esta parte del algoritmo me pongas el valor de este carácter en la tabla unicode.*
- Cualquier otro carácter que aparezca en el texto original será movido 5 posiciones hacia atrás en la tabla unicode. Pej, si aparece el carácter 64 (@) en el texto original, será cambiado por el carácter 59 (;)

Nota: si sabes definir funciones en JAVA, puedes ordenar el código usando funciones..

Ejemplo de ejecución

```
Introduzca fichero a codificar: texto.txt
Generando archivo texto.cod...

Archivo texto.cod generado con éxito.
```

Archivo texto.txt	Archivo texto.cod
Se iba la luz con el traspuesto Apolo y la sombra a los brutos de la tierra obligaba al reposo: yo tan solo me disponia a sostener la guerra 1764, ya de la compasion, ya del camino que se trazo mi mente que no yerra.	Tf]jcb]mb]mva]dpo]fm]usbtqvftup]Bqpm z]mb]tpncsb]b]mpt]csvupt]ef]mb]ujfss pcmjhbcb]bm]sfqptp5]zp]ubo]tpm nf]ejtqpojb]b]tptufofs]mb]hvfssb]ÇÇÇÇ zb]ef]mb]dpnqbtjpo']zb]efm]dbnjo rvf]tf]usbap]nj]nfouf]rvf]op]zfssb