Ejercicios Ficheros de texto

1. Programa Java que lee texto por teclado y lo escribe en un fichero de texto llamado datos.txt. El proceso consiste en leer una línea de texto por teclado y escribirla en el fichero. Este proceso se repite hasta que se introduce por teclado la cadena FIN. La cadena FIN que indica el final de lectura no se debe escribir en el fichero.

2. Programa que leer por teclado líneas de texto y las añade al final del ficheros datos.txt. Para resolverlo vamos a modificar el programa anterior para que añada texto al fichero datos.txt, es decir, al volver a ejecutar el programa el contenido anterior del fichero no se pierde y el contenido nuevo se añade al final.

3. Programa Java que lee el contenido del fichero datos.txt creado en el ejemplo anterior y lo muestra por pantalla. El proceso consiste en leer una línea del fichero y mostrarla por pantalla. El proceso se repite hasta que se llegue al final del fichero y no hayan más líneas que leer. Cuando esto ocurre el método readLine() devuelve *null*.

4. Mostrar por pantalla el contenido del fichero de texto datos.txt pero en este caso lo vamos a leer carácter a carcater. El proceso consiste en leer un carácter del fichero y mostrarlo por pantalla. Este proceso se repite hasta que no queden más caracteres que leer en el fichero, es decir, hasta que se alcance el finall del fichero. En este caso el método read() devuelve -1.

5. Programa que lee línea a línea el contenido del fichero datos.txt utilizando la clase Scanner.

Se utiliza el **método hasNext()** de Scanner para saber si quedan más datos que leer en el fichero. Este método devuelve *false* si se ha llegado al final del fichero y *true* en caso contrario.

6. Disponemos de un fichero de texto llamado enteros.txt que contiene números enteros. El siguiente programa lee los números y los muestra. Muestra también la cantidad de números leídos y su suma.

Por ejemplo, si el fichero enteros.txt contiene los siguientes números:
323 34 234 990 22 3 1
5463 28 34 0 7

El programa mostrará por pantalla:

323
34
234
990
22
3
1

5463
28
34
0
7
Número leídos: 12
Suma 7139

Se utilizará el **método hasNextInt()** de Scanner para saber si quedan más enteros que leer en el fichero. El método hasNextInt() devuelve *true* cuando lo siguiente que se va a extraer del fichero es un entero y devuelve false en caso contrario.
La lectura acaba cuando no quedan más enteros (se ha llegado al final del fichero) o cuando encuentra un carácter no válido como entero.

Por ejemplo, si el contenido del fichero enteros.txt es el siguiente:

323 34 KKK 234 990 22 3 1
5463 28 34 0 7

El programa mostrará por pantalla:
323
34
Número leídos: 2
Suma 357

7. Disponemos de un fichero de texto llamado enteros.txt que contiene los siguientes números enteros separados por espacios en blanco o comas:
34,45,23 8, 9
12 23
El siguiente programa Java lee el contenido del fichero y muestra los números. Muestra también la cantidad de números leídos y su suma.
El programa lee líneas completas del fichero y las pasa a un StringTokenizer del que se extraen los números.

8. Programa que obtiene la línea de mayor tamaño y la de menor tamaño dentro de un fichero de texto.

Para resolver este ejercicio se utiliza la clase Scanner para leer el fichero. La lectura se realiza línea a línea hasta que se alcance el final del fichero. Para determinar cuál es la de mayor longitud y cuál es la menor, se lee la primera línea del fichero y se toma como la línea mayor y

la menor. A continuación se leen el resto de líneas y para cada una se compara su tamaño con la mayor y menor actuales.
El nombre del fichero se selecciona de forma gráfica utilizando la clase JfileChooser.

9. Programa Java para buscar una palabra o una cadena en un fichero de texto.

El programa pedirá que se introduzca una palabra o un texto por teclado y realizará su búsqueda por todo el archivo. Se mostrará por pantalla el número de línea y el contenido de la línea del fichero que contiene la cadena buscada. Si la cadena buscada aparece en varias líneas se mostrarán todas ellas. Si el fichero no contiene el texto buscado se mostrará un mensaje indicándolo.

Por ejemplo, tenemos un fichero de texto llamado TemaFicheros.txt con el siguiente contenido:

```
A partir de Java 7 se puede usar la instrucción try-with-resources.
Un resource (recurso) es un objeto que necesita ser cerrado.
Un try-with-resources asegura que estos objetos serán cerrados.
Un objeto AutoCloseable puede ser usado como recurso.
```

Si el texto a buscar es *recurso* el programa mostrará por pantalla:

```
Archivo: TemaFicheros.txt
Texto a buscar: recurso
Linea 2: Un resource (recurso) es un objeto que necesita ser cerrado.
Linea 4: Un objeto AutoCloseable puede ser usado como recurso.
```

Si el texto a buscar ahora es *array* el programa mostrará por pantalla:

```
Archivo: TemaFicheros.txt
Texto a buscar: array
array no se ha encontrado en el archivo
```

Solución:
Para resolver el ejercicio de búsqueda de texto en ficheros leeremos el fichero línea a línea utilizando la clase Scanner. Para cada línea se comprueba si contiene o no el texto buscado, si la línea contiene lo que buscamos se muestra por pantalla junto a su número de línea. Para obtener el número de línea utilizaremos un contador que se incrementa cada vez que se lea una nueva línea desde el archivo. Se utilizará una variable de tipo boolean para saber si se ha encontrado o no el texto buscado.

10. Crear un fichero de texto en Java con caracteres aleatorios.

Programa Java para crear un fichero de texto que contenga caracteres obtenidos de forma aleatoria. Los caracteres a incluir en el fichero serán las letras de la A a la Z (mayúsculas y minúsculas incluida la ñ) y espacios en blanco. El número total de caracteres a escribir se pide por teclado.

Por ejemplo, si el número de caracteres a escribir en el fichero es 30 el contenido del fichero creado podría ser este:

```
kUIdzzMVjnRÑhyUPDANgIXmnb Pkhl
```

El fichero se llamará "caracteres.txt". Se debe crear en la carpeta "ficheros" en la unidad C:

Solución:
Para crear el fichero de texto utilizaremos la clase PrintWriter:

```
PrintWriter salida = new PrintWriter("c:/ficheros/caracteres.txt")
```

Utilizaremos la clase Random para obtener números de forma aleatoria. Obtendremos números entre 0 y 255 que es el rango de caracteres de la tabla ASCII. Una vez obtenido el número comprobaremos si corresponde a una letra mayúscula o minúscula o a un espacio en blanco o a la letra ñ. Si es así se escribe en el fichero de texto.

11. Programa Java que lee un archivo de texto que contiene números de tipo *int* y *double*. El archivo a leer está formado por dos líneas. La primera línea del fichero contiene números enteros separados por espacios en blanco. La segunda línea contiene números de tipo *double* separados también por espacios en blanco.
Por ejemplo:
2 6 -1 0 5 10 1 8 2 100
5,75 -8,25 4,25
No conocemos a priori la cantidad de números que hay en cada línea del archivo.
El programa debe leer el archivo de texto, mostrar por pantalla los números enteros y su suma y a continuación mostrar por pantalla los números double y su suma.
Por ejemplo, si el archivo contiene los valores anteriores, el programa mostrará por pantalla:
Números de tipo int:
2 6 -1 0 5 10 1 8 2 100
Suma de los int: 133
Números de tipo double:
5.75 -8.25 4.25

Suma de los doubles: 1.75

Solución:

Para leer el archivo de texto utilizaremos la clase Scanner.

Sabemos que en la primera línea del archivo se encuentran los números enteros. Para leer cada número entero del archivo utilizaremos el método nextInt() de Scanner. Como no sabemos cuántos números hay en la línea, para poder leerlos todos utilizaremos el método hasNextInt() de Scanner dentro de un bucle while. El método hasNextInt() devuelve true cuando lo siguiente que se va a leer del archivo es un entero, por lo tanto podemos usarlo para saber si quedan más enteros por leer.

Cuando hasNextInt() devuelva false se habrán acabado lo números enteros y debemos empezar a leer los de tipo double. Cada double del archivo se leerá con en método nextDouble(). Utilizaremos ahora el método hasNextDouble() dentro de un bucle while para poder leerlos todos.