Actividad resuelta 10.1

Escribir el método

```
Integer leerEntero(),
```

que pide un entero por consola, lo lee del teclado y lo devuelve. Si la cadena introducida por consola no tiene el formato correcto, muestra un mensaje de error y vuelve a pedirlo.

Actividad resuelta 10.2

Leer el archivo de texto Main.java de uno de los proyectos que hayamos terminado y mostrarlo por pantalla.

Solución

Actividad resuelta 10.3

Repetir la Actividad resuelta 10.2 usando un flujo de texto con búfer.

Solución

```
String texto = "";
```

Actividad resuelta 10.4

Crear con un editor el fichero de texto *NumerosReales.txt* en la carpeta del proyecto de NetBeans actual y escribir en él una serie de números reales separados por espacios simples.

Implementar un programa que acceda a *NumerosReales.txt*, lea los números y calcule la suma y la media aritmética, mostrando los resultados por pantalla.

Actividad propuesta 10.2

Pide por teclado el nombre, la edad (`int`) y la estatura en metros (`double`) de un deportista. Introduce los datos en una sola línea y léelos con un objeto `Scanner`. Muestra los resultados por pantalla.

Actividad resuelta 10.5

Crear con un editor un archivo de texto con un conjunto de números reales, uno por línea. Abrirlo con un flujo de texto para lectura y leerlo línea a línea. Convertir las cadenas leídas en números de tipo `double` por medio de `Scanner`, y mostrar al final la suma de todos ellos.

Solución

Actividad resuelta 10.6

Crear con un editor el fichero de texto *Enteros.txt* en la carpeta del proyecto actual de NetBeans y escribir en él una serie de enteros separados por secuencias de espacios y tabuladores, incluso en líneas distintas, tal como:

```
2   3  45    73
123      4  21
```

Implementar un programa que acceda a *Enteros.txt* con un objeto Scanner a través de un flujo de entrada, lea los números y calcule su suma y su media aritmética, mostrando los resultados por pantalla.

Actividad propuesta 10.3

Crea con un editor el fichero de texto *Jugadores.txt* en la carpeta del proyecto de NetBeans actual y escribe en él los nombres, edades y estaturas de los jugadores de un equipo, cada uno en una línea.

```
juan 22 1.77
luis 22  1.80
pedro 20 1.73
...
```

Implementa un programa que lea del fichero los datos, muestre los nombres y calcule la media de la edad y de las estaturas, mostrándolas por pantalla.

Actividad resuelta 10.7

Como ejemplo, vamos a guardar en un fichero el texto,

> "En un lugar de La Mancha,
> de cuyo nombre no quiero acordarme"

La primera línea, carácter a carácter, y la segunda, en una sola sentencia.

10.1. Una excepción en Java:

a) Se produce cuando un disco está defectuoso.
b) Es un valor único de una variable.
c) Se arroja al sistema cuando se produce una condición anómala durante la ejecución de un programa.
d) Tiene lugar cuando un código es sintácticamente incorrecto.

10.2. Una excepción comprobada es:

a) Una excepción que hemos reparado.
b) Una excepción que no detiene la ejecución del programa.
c) Una excepción previsible, que el propio compilador nos obliga a gestionar.
d) Una excepción muy conocida.

10.3. Cuando llegamos al final de un flujo de entrada de tipo `FileReader`, el método `read()`:

a) Muestra el mensaje: End of File
b) Devuelve `null`.
c) Produce una excepción `EOFException`
d) Devuelve –1.

10.4. La palabra reservada `finally`:

a) Termina la ejecución de un programa.
b) Termina la ejecución de un método, forzando el `return`.
c) En una estructura `try-catch`, fuerza la ejecución de su bloque antes de que se ejecute una sentencia `return` e independientemente de si se produce o no una excepción.
d) Indica el final de un método.

10.5. **Un flujo de tipo `BufferedReader`:**

a) Crea un archivo de texto con búfer.

b) Solo sirve para leer cadenas de caracteres.

c) Nos permite acceder a archivos binarios.

d) Accede a un archivo de texto para lectura con búfer.

10.6. **La clase `Scanner`:**

a) Solo permite leer texto de cualquier flujo de texto.

b) Permite digitalizar imágenes.

c) Permite leer y analizar texto de cualquier flujo de entrada de texto.

d) Solo nos permite leer de la consola.

10.7. **Para cambiar de línea al escribir en el flujo `salida` de tipo `BufferedWriter` debemos ejecutar:**

a) salida.write("\n")

b) salida.write("\r\n")

c) salida.write("newLine")

d) salida.newLine()

10.8. Nos tenemos que asegurar de que todos los flujos abiertos deben cerrarse antes de que termine la aplicación...

a) Porque se quedarían abiertos hasta que se apague el ordenador.
b) Porque otra aplicación podría alterarlos.
c) Porque se deben liberar los recursos asociados, como los archivos. Además, podrían quedar caracteres del búfer sin escribir.
d) Porque se pueden borrar datos de un archivo.

10.9. Los flujos se cierran:

a) Con el método `close()`.
b) Apagando el ordenador.
c) Abortando el programa.
d) Con el método `cerrar()`.

10.10. Apertura de flujos con recursos:

a) Consiste en abrir flujos asociados con varios archivos a la vez.
b) Es abrir archivos recurriendo a una tabla.
c) Es una nueva forma de abrir flujos en Java, que permite prescindir del cierre explícito de los archivos y del método `close()`.
d) Consiste en abrir flujos sin peligro de que se produzcan excepciones.

10.11. Escribe un programa que solicite al usuario el nombre de un fichero de texto y muestre su contenido en pantalla. Si no se proporciona ningún nombre de fichero, la aplicación utilizará por defecto *prueba.txt*.

10.12. Diseña una aplicación que pida al usuario su nombre y edad. Estos datos deben guardarse en el fichero *datos.txt*. Si este fichero existe, deben añadirse al final en una nueva línea, y en caso de no existir, debe crearse.

10.13. Implementa un programa que lea dos listas de números enteros no ordenados de sendos archivos con un número por línea, los reúna en una lista única y los guarde en orden creciente en un tercer archivo, de nuevo uno por línea.

10.14. Escribe un programa que lea un fichero de texto llamado *carta.txt*. Tenemos que contar los caracteres, las líneas y las palabras. Para simplificar supondremos que cada palabra está separada de otra por un único espacio en blanco o por un cambio de línea.

10.15. En el archivo *numeros.txt* disponemos de una serie de números (uno por cada línea). Diseña un programa que procese el fichero y nos muestre el menor y el mayor.

10.16. Un libro de firmas es útil para recoger los nombres de todas las personas que han pasado por un determinado lugar. Crea una aplicación que permita mostrar el libro de firmas e insertar un nuevo nombre (comprobando que no se encuentre repetido). Llamaremos al fichero *firmas.txt*.

10.17. En Linux disponemos del comando *more*, al que se le pasa un fichero y lo muestra poco a poco: cada 24 líneas. Implementa un programa que funcione de forma similar.

10.18. Escribe la función `Integer[] leerEnteros(String texto)`, al que se le pasa una cadena y devuelve una tabla con todos los enteros que aparecen en ella.

10.19. Un encriptador es una aplicación que transforma un texto haciéndolo ilegible para aquellos que desconocen el código. Diseña un programa que lea un fichero de texto, lo codifique y cree un nuevo archivo con el mensaje cifrado. El alfabeto de codificación se encontrará en el fichero *codec.txt*. Un ejemplo de codificación de alfabeto es:

Alfabeto: a b c d e f g h i j k l m n o p q r s t u v w x y z

Cifrado: e m s r c y j n f x i w t a k o z d l q v b h u p g

10.20. Algunos sistemas operativos disponen de la orden `comp`, que compara dos archivos y nos dice si son iguales o distintos. Diseña esta orden de forma que, además, nos diga en qué línea y carácter se encuentra la primera diferencia. Utiliza los ficheros *texto1.txt* y *texto2.txt*.

10.21. Diseña una pequeña agenda, que muestre el siguiente menú:

1. Nuevo contacto.
2. Buscar por nombre.
3. Mostrar todos.
4. Salir.

En ella, guardaremos el nombre y el teléfono de un máximo de 20 personas.

La opción 1 nos permitirá introducir un nuevo contacto siempre y cuando la agenda no esté llena, comprobando que el nombre no se encuentra insertado ya.

La opción 2 muestra todos los teléfonos que coinciden con la cadena que se busca. Por ejemplo, si tecleamos «`Pe`», mostrará el teléfono de `Pe`dro, de `Pe`pe y de `Pe`tunia.

La opción 3 mostrará un listado con toda la información (nombres y teléfonos) ordenados alfabéticamente por el nombre.

Por último, la opción 4 guarda todos los datos de la agenda en el archivo *agenda.txt*.

La próxima vez que se ejecute la aplicación, si hay datos guardados, se cargarán en memoria.

10.22. Crea con un editor de texto el fichero *deportistas.txt*, donde se recogen los datos de un grupo de deportistas, uno en cada línea. Aparecerá el nombre completo, seguido de la edad, el peso y la estatura. La primera línea será el encabezamiento con los nombres de los campos. El documento tendrá la siguiente forma:

Nombre	Edad	Peso	Estatura
Juan Pedro Pérez Gómez	25	70,5	1,80
Ana Ruiz del Val	23	60	1,75

...

Implementa un programa donde se cree un flujo de texto de entrada, a partir del cual, usando un objeto `Scanner`, se leerán los datos de los deportistas, que se mostrarán por pantalla. Al final aparecerán los valores medios de la edad, el peso y la estatura.

ACTIVIDADES FINALES

10.23. Con el fichero *deportistas.txt* de la Actividad de aplicación 10.22, implementa una aplicación que lea los datos de los deportistas y los guarde en otros tres ficheros, uno con los nombres y las edades, otro con los nombres y los pesos y el tercero con los nombres y las estaturas.

10.24. Implementa una aplicación que mantenga un registro de las temperaturas máxima y mínima diarias medidas en una estación meteorológica. Los datos se guardarán en un archivo de texto con el siguiente formato:

Fecha	Temperatura máxima	Temperatura mínima
2020-01-15	12	–1
2020-01-16	15	2

...

Al arrancar la aplicación aparecerá un menú con las opciones:

1. Registrar nueva temperatura.
2. Mostrar historial de registros.
3. Salir.

El historial de registros mostrará todos los datos registrados junto con el máximo valor de las temperaturas máximas y el mínimo de las temperaturas mínimas.

10.25. Repite la Actividad de aplicación 10.21, pero guardando los datos en un fichero XML.

10.26. Repite la Actividad de aplicación 10.24, pero guardando los datos en un fichero XML.

Actividades de ampliación

Actividades de ampliación

10.27. Repite la Actividad de aplicación 10.14, pero sabiendo que una palabra puede no estar separada de otra solo por un espacio en blanco; también puede ser un tabulador, punto, coma o punto y coma.

10.28. Diseña un programa al que se le proporcione el nombre de un fichero de texto y una cadena. Debemos buscar todas las ocurrencias de la cadena en el fichero.

10.29. Escribe un programa que pida el nombre de un fichero de texto que contenga código fuente en Java. El programa debe crear un nuevo fichero que tenga como nombre el mismo del fichero original con el prefijo «sin_comentarios_». El nuevo fichero tendrá como contenido el código fuente sin ningún tipo de comentarios.

10.30. En ocasiones, es necesario particionar un fichero de texto de gran tamaño en otros ficheros más pequeños. Crea una aplicación a la que se le proporciona un fichero de texto y un tamaño. Este puede estar en bytes, kilobytes o megabytes. La aplicación debe fragmentar el fichero original en tantos ficheros del tamaño especificado como necesite. Los nombres de estos ficheros serán idénticos al nombre original con el prefijo «parte999_», donde «999» es el número del volumen. Para indicar el tamaño se escribirá una cantidad seguida de b (bytes), k (kilobytes) o m (megabytes).

10.31. Se pretende mantener los datos de los clientes de un banco en un archivo de texto. De cada cliente se guardará: DNI, nombre completo, fecha de nacimiento y saldo. Implementa una aplicación que arranque mostrando en el menú:

1. Alta cliente.
2. Baja cliente.
3. Listar clientes.
4. Salir.

Implementa la clase `Cliente` con los atributos referidos. Nada más arrancar la aplicación se leerán del archivo los datos de los clientes construyendo los objetos `Cliente` de todos ellos, que se irán insertando en una tabla de clientes. Cuando se dé de alta uno nuevo, se creará el objeto correspondiente y se insertará en la tabla por orden de DNI. Para eliminar un cliente, se pedirá el DNI y se eliminará de la tabla. Al listar los clientes, se mostrará el DNI, el nombre, el saldo y la edad de todos ellos, así como el saldo máximo, el mínimo y el promedio del conjunto de los clientes. Al cerrar la aplicación, se guardarán en el archivo los datos actualizados con el mismo formato.