

CIFP César Manrique.

Programación 1º de Desarrollo de Aplicaciones Web

Profesor: José David Díaz Díaz

Actividades de la Unidad 9: Ficheros de texto



Esta obra está licenciada bajo la Licencia Creative Commons Atribución 4.0 Internacional.
Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by/4.0/> o
envíe una carta a Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Índice

Actividades.....	1
Desarrollo.....	2
Actividades de comprobación.....	2
10.1. Una excepción en Java:.....	2
10.2. Una excepción comprobada es:.....	2
10.3. Cuando llegamos al final de un flujo de entrada de tipo FileReader, el método read():....	2
10.4. La palabra reservada finally:.....	3
10.5. Un flujo de tipo BufferedReader:.....	3
10.6. La clase Scanner:.....	3
10.7. Para cambiar de línea al escribir en el flujo salida de tipo BufferedWriter debemos ejecutar:.....	4
10.8. Nos tenemos que asegurar de que todos los flujos abiertos deben cerrarse antes de que termine la aplicación	4
10.9. Los flujos se cierran:.....	4
10.10. Apertura de flujos con recursos:.....	5
Actividades de Aplicación.....	6
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.....	6
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.....	9
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.....	11
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.....	12
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.....	13
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	14
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.....	16
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.....	18
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.....	20

Actividades

Actividades de la Unidad 9: Fichero de texto.

En este documento se detallan las actividades a realizar. Se entregará al profesor en la plataforma digital dos ficheros. Un primer fichero pdf con todas las actividades a realizar, el nombre del fichero será “unidad2 + nombre del alumno.pdf”. Añadir en el fichero pdf por cada actividad de programación dos capturas de pantalla, una del código y otra de su ejecución. También en el fichero pdf copiar todas las preguntas y las respuestas correctas de las actividades de comprobación. Además, entregar un segundo fichero comprimido con todos los códigos fuentes de cada actividad de programación realizada.

Todas las actividades resueltas se deberán de analizar y no se entregarán.

A continuación, detallamos las actividades a realizar:

- **Actividades propuestas.** No realizar ninguna.
- **Actividades de comprobación.** Realizarlas todas. Copiar todas las preguntas y sus respuestas correctas.
- **Actividades de aplicación.** Realizar las siguientes **10.11, 10.12, 10.13, 10.14, 10.15, 10.16, 10.17, 10.18 y 10.20** .
- **Actividades de ampliación.** No realizar ninguna.

[Volver al índice](#)

Desarrollo

Actividades de comprobación.

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.

[Volver al índice](#)

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.

[Volver al índice](#)

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.

[Volver al índice](#)

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.

[Volver al índice](#)

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.

[Volver al índice](#)

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.

[Volver al índice](#)

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 ()`

[Volver al índice](#)

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.

[Volver al índice](#)

10.9. Los flujos se cierran:

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

[Volver al índice](#)

10.10. Apertura de flujos con recursos:

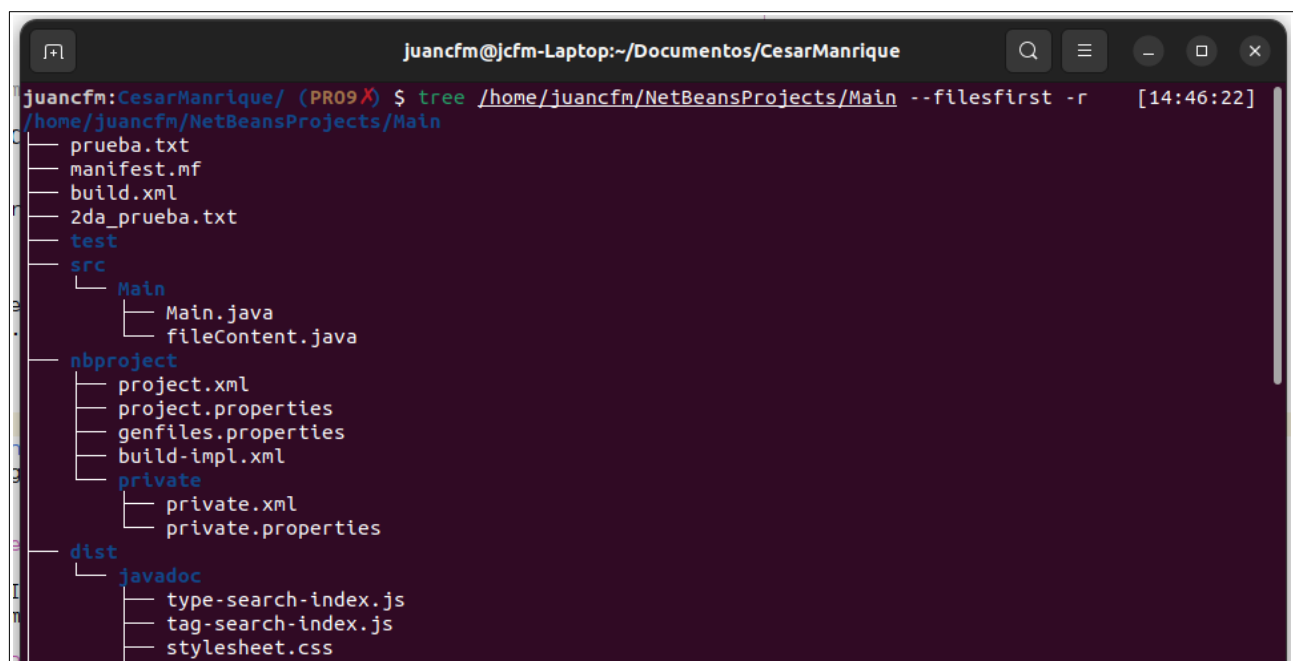
- a) Consiste en abrir flujos asociados con varios archivos a la vez.
- b) Es abrir archivos recurriendo a una tabla.
- c) Es una nueva forma de abrir flujos en Java, que permite prescindir del cierre explícito de los archivos y del método `close()`.
- d) Consiste en abrir flujos sin peligro de que se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

[Volver al índice](#)

Actividades de Aplicación.

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.

Para realizar las pruebas de este ejercicio se colocaron dos ficheros de prueba en la raíz del proyecto para no tener que estar indicando la ruta. Se adjunta captura de pantalla con la ruta y la ubicación de los archivos de prueba así como la ubicación de los archivos con el código de las clases usadas. Recordar que estos ejercicios los realizo sobre un equipo con Ubuntu 22.04 con lo que las rutas difieren de las proporcionadas por un equipo con Windows.



```
juancfm@jcfm-Laptop:~/Documentos/CesarManrique
juancfm:CesarManrique/ (PROJ) $ tree /home/juancfm/NetBeansProjects/Main --filesfirst -r [14:46:22]
/home/juancfm/NetBeansProjects/Main
├── prueba.txt
├── manifest.mf
├── build.xml
├── 2da_prueba.txt
├── test
├── src
│   └── Main
│       ├── Main.java
│       └── fileContent.java
├── nbproject
│   ├── project.xml
│   ├── project.properties
│   ├── genfiles.properties
│   ├── build-impl.xml
│   └── private
│       ├── private.xml
│       └── private.properties
└── dist
    └── javadoc
        ├── type-search-index.js
        ├── tag-search-index.js
        └── stylesheet.css
```

Actividades de la Unidad 9: Ficheros de texto

```
package Main;
import java.util.Scanner;

/**
 * @author juancfm
 */
public class Main {

    /**
     * 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.
     */
    String texto;

    public static void main(String[] args) {

        try (Scanner s = new Scanner(System.in)) {
            System.out.println("El presente programa le solicitará el nombre de un
            + "de un fichero de texto y le mostrará su contenido por "
            + "pantalla.");
            System.out.println(x: "\nPulse la tecla Enter para continuar");
            texto = s.nextLine();

            System.out.println("\nPor favor introduzca el nombre del "
            + "fichero a leer: ");
            texto = s.nextLine();

            fileContent file = new fileContent();

            System.out.println(x: file.fileContent(texto));
        } catch (Exception e) {
            System.err.println(x: e.getMessage());
        }
        System.exit(status:0);
    }
}
```

```
package Main;
import java.io.*;

/**
 * @author juancfm
 */
public class fileContent {

    String result = "";
    BufferedReader bR = null;

    public String fileContent(String texto) {

        texto = texto.trim();
        if (texto.isEmpty()) {
            texto = "prueba.txt";
        }

        try {
            bR = new BufferedReader(new FileReader(fileName: texto));
            String line;

            while ((line = bR.readLine()) != null) {
                result = result + line + "\n";
            }
        } catch (IOException e) {
            System.err.println("Error al leer el fichero: " + e.getMessage());
        } finally {
            if (bR != null) {
                try {
                    bR.close();
                } catch (Exception e) {
                    System.err.println("Segunda Exception: " + e);
                }
            }
        }

        return result;
    }
}
```

Output

Main (run) × Main (run) #2 ×

```
run:
El presente programa le solicitará el nombre de un fichero de texto y le mostrará su contenido por pantalla.

Pulse la tecla Enter para continuar

Por favor introduzca el nombre del fichero a leer:

Esta es una prueba.
Esta es la segunda línea de la prueba.

BUILD SUCCESSFUL (total time: 9 seconds)
```

Output

Main (run) × Main (run) #2 ×

```
run:
El presente programa le solicitará el nombre de un fichero de texto y le mostrará su contenido por pantalla.

Pulse la tecla Enter para continuar

Por favor introduzca el nombre del fichero a leer:
2da_prueba.txt
Esta es otra prueba (2da prueba).
Esta es la segunda línea de la 2da prueba.

BUILD SUCCESSFUL (total time: 12 seconds)
```

Actividades de la Unidad 9: Ficheros de texto

Output

▶▶ Main (run) × Main (run) #2 ×

▶▶ run:
El presente programa le solicitará el nombre de un fichero de texto y le mostrará su contenido por pantalla.
Pulse la tecla Enter para continuar

Por favor introduzca el nombre del fichero a leer:
XXXXXXX
Error al leer el fichero: XXXXXX (No existe el archivo o el directorio)

BUILD SUCCESSFUL (total time: 8 seconds)
|

[Volver al índice](#)

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.

```
package Main;
import java.util.Scanner;

/**
 *
 * @author juancfm
 */
public class Main {

    public static void main(String[] args) {

        /**
         * 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.
         */
        String nombre = "1";
        String pausa;
        String pathname = "datos.txt";
        int edad = 0;

        try (Scanner s = new Scanner(System.in)) {
            System.out.println("El presente programa le solicitará un nombre "
                + "y una edad, estos datos se guardarán en un fichero de "
                + "texto llamado \"datos.txt\"");
            System.out.println("Si este fichero existe, deben añadirse al "
                + "final en una nueva línea, y en caso de no existir, "
                + "debe crearse.");
            System.out.println(x: "¡Pulse la tecla Enter para continuar!");
            pausa = s.nextLine();

            System.out.println("\nPor favor, introduzca el nombre de la persona"
                + " que será agregada al fichero: ");
            nombre = s.nextLine();

            System.out.println("\nPor favor, introduzca la edad de la persona"
                + " que será agregada al fichero: ");
            edad = s.nextInt();

        } catch (Exception e) {
            System.err.println(x: e.getMessage());
        }

        Archivo.escribirDatos(pathname, nombre, edad);

        System.out.println(x: "Gracias por usar este programa.");
        System.exit(status: 0);
    }
}
```

```
package Main;
import java.io.*;

/**
 *
 * @author juancfm
 */
class Archivo {

    static void escribirDatos(String pathname, String nombre, int edad) {

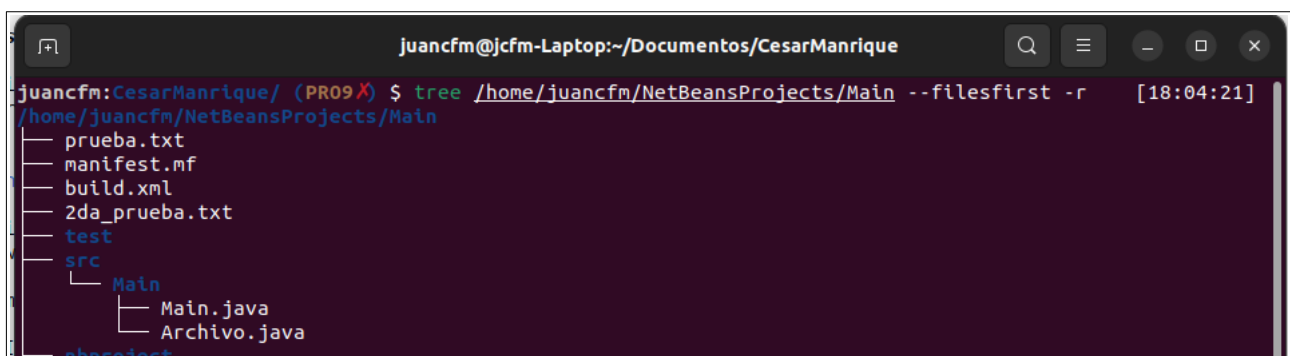
        File archivo = new File(pathname);
        BufferedWriter bw = null;

        try {
            bw = new BufferedWriter(new FileWriter(file:archivo, append:true));

            bw.write(nombre + ", " + edad + ".");
            bw.newLine();

            System.out.println(x: "Los datos se han guardado correctamente.");

        } catch (IOException e) {
            System.out.println("Ha ocurrido un error al guardar los datos. "
                + e.getMessage());
        } finally {
            if (bw != null) {
                try {
                    bw.close();
                } catch (IOException e) {
                    System.err.println("No se ha podido cerrar la conexión: "
                        + e.getMessage());
                }
            }
        }
    }
}
```



```
juancfm@jcfm-Laptop:~/Documentos/CesarManrique
juancfm:CesarManrique/ (PROJ) $ tree /home/juancfm/NetBeansProjects/Main --filesfirst -r [18:04:21]
/home/juancfm/NetBeansProjects/Main
├── prueba.txt
├── manifest.mf
├── build.xml
├── 2da_prueba.txt
├── test
├── src
│   ├── Main
│   │   ├── Main.java
│   │   └── Archivo.java
└── nbproject
```

Actividades de la Unidad 9: Ficheros de texto

```
Output
Main (run) x Main (run) #2 x
run:
El presente programa le solicitará un nombre y una edad, estos datos se guardarán en un fichero de texto llamado "datos.txt"
Si este fichero existe, deben añadirse al final en una nueva línea, y en caso de no existir, debe crearse.

Pulse la tecla Enter para continuar

Por favor introduzca el nombre de la persona que será agregada al fichero:
Pedro

Por favor introduzca la edad de la persona que será agregada al fichero:
35
Los datos se han guardado correctamente.
Gracias por usar este programa.
BUILD SUCCESSFUL (total time: 24 seconds)
```

```
juancfm@jcfm-Laptop: ~/Documentos/CesarManrique
28 directories, 63 files
juancfm:CesarManrique/ (PR09 X) $ tree /home/juancfm/NetBeansProjects/Main --filesfirst -r [18:04:27]
/home/juancfm/NetBeansProjects/Main
├── prueba.txt
├── manifest.mf
├── datos.txt
├── build.xml
├── 2da_prueba.txt
├── test
├── src
│   └── Main
│       ├── Main.java
│       └── Archivo.java
```

```
juancfm:CesarManrique/ (PR09 X) $ cd /home/juancfm/NetBeansProjects/Main [18:06:47]
juancfm:Main/ $ cat datos.txt [18:08:12]
Pedro, 35.
```

[Volver al índice](#)

10.13. Implementa un programa que lea dos listas de números enteros no ordenados de sen dos 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.

```
package Main;

import java.io.*;
import java.util.*;

/**
 * @author juancfm
 */
public class Main {

    /**
     * 10.13. Implementa un programa que lea dos listas de números enteros
     * no ordenados desde dos 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.
     */

    String fichero1 = "file1.txt";
    String fichero2 = "file2.txt";
    String ficheroSalida = "output.txt";

    // Leer números de los archivos
    List<Integer> numeros1 = leerNumeros( nombreArchivo: fichero1);
    List<Integer> numeros2 = leerNumeros( nombreArchivo: fichero2);

    // Unir las listas de números
    List<Integer> numerosAgrupados = new ArrayList<>( numeros1);
    numerosAgrupados.addAll( numeros2);

    // Ordenar la lista de números
    Collections.sort( numerosAgrupados);

    // Guardar la lista ordenada en un archivo
    guardarNumeros( nombreArchivo: ficheroSalida, numeros: numerosAgrupados);
}
```

```
public static List<Integer> leerNumeros(String nombreArchivo) {
    List<Integer> numeros = new ArrayList<>();
    try (BufferedReader bR
        = new BufferedReader(new FileReader( fileName: nombreArchivo))) {
        String line;
        while ((line = bR.readLine()) != null) {
            try {
                int numero = Integer.parseInt( line.trim());
                numeros.add( numero);
            } catch (NumberFormatException e) {
                System.err.println("Error al parsear número en archivo "
                    + nombreArchivo + ": " + e.getMessage());
            }
        }
    } catch (IOException e) {
        System.err.println("Error al leer archivo " + nombreArchivo + ": "
            + e.getMessage());
    }
    return numeros;
}

public static void guardarNumeros(String nombreArchivo,
    List<Integer> numeros) {
    try (BufferedWriter bW
        = new BufferedWriter(new FileWriter( fileName: nombreArchivo))) {
        for (int numero : numeros) {
            bW.write( Integer.toString( numero));
            bW.newLine();
        }
        System.out.println("Archivo " + nombreArchivo
            + " guardado correctamente.");
    } catch (IOException e) {
        System.err.println("Error al guardar archivo " + nombreArchivo +
            ": " + e.getMessage());
    }
}
```

```
juancfm@jcfm-Laptop:~/NetBeansProjects/Main
juancfm:~/Main/ $ cat file1.txt
1
2
23
3
45
67
34
21
45
26
juancfm:~/Main/ $
```

```
juancfm@jcfm-Laptop:~/NetBeansProjects/Main
juancfm:~/Main/ $ cat file2.txt
3
4
3
12
juancfm:~/Main/ $
```

```
Output
Main (run) × Main (run) #2 ×
run:
Archivo output.txt guardado correctamente.
BUILD SUCCESSFUL (total time: 0 seconds)
```

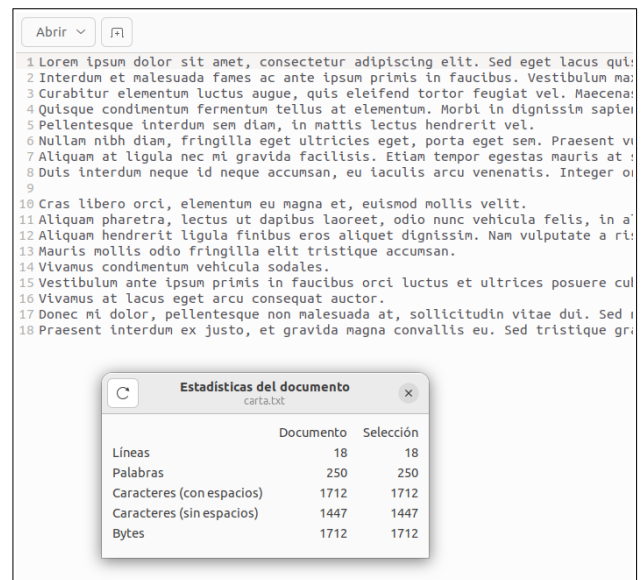
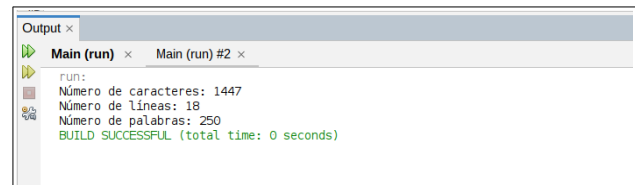
```
juancfm@jcfm-Laptop:~/NetBeansProjects/Main
juancfm:~/Main/ $ cat output.txt
1
2
3
3
4
5
12
21
23
26
34
45
67
juancfm:~/Main/ $
```

```
juancfm:~/Main/ $ tree /home/juancfm/NetBeansProjects/Main --filesfirst -r
/home/juancfm/NetBeansProjects/Main
├── prueba.txt
├── output.txt
├── mainfest.nf
├── file2.txt
├── file1.txt
├── datos.txt
├── build.xml
├── 2da_prueba.txt
├── test
└── src
    └── Main.java
```

[Volver al índice](#)

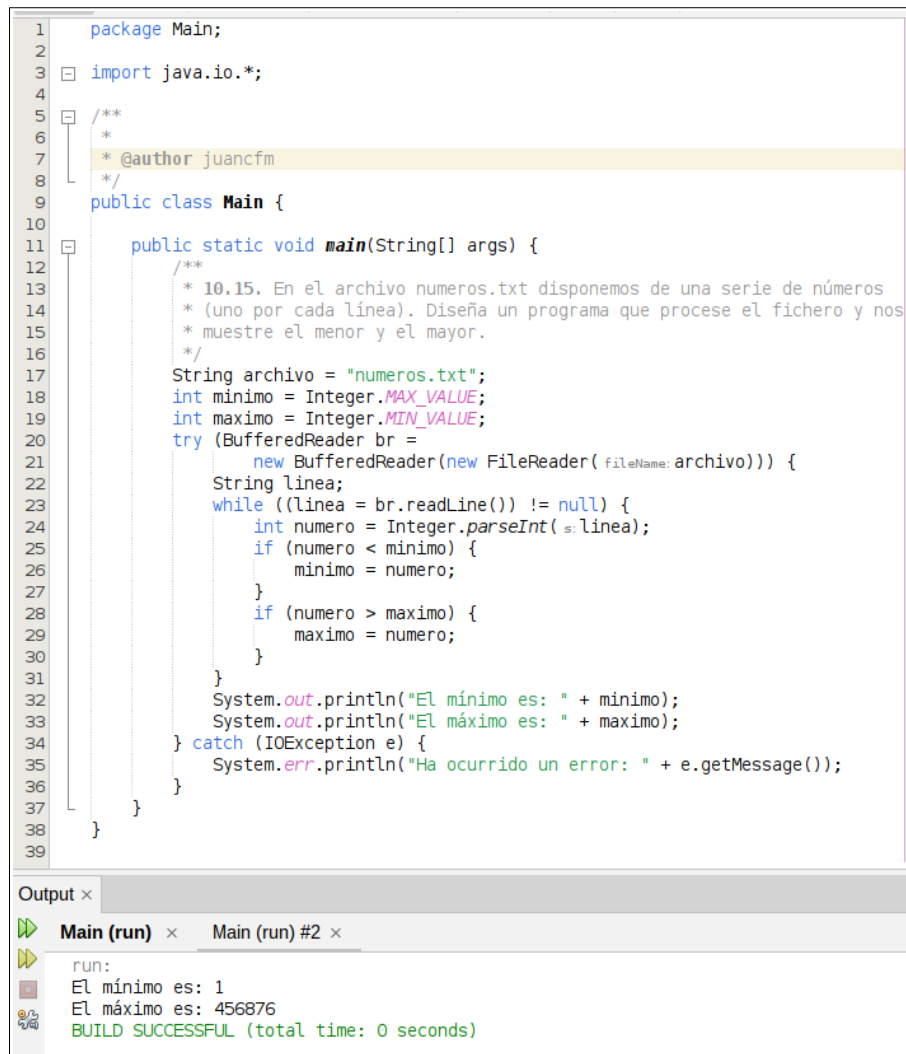
10.14. Escribe un programa que lea un fichero de texto llamado carta.txt. Tenemos que contar los caracteres, las líneas y las palabras. Para simplificar supondremos que cada palabra está separada de otra por un único espacio en blanco o por un cambio de línea.

```
1 package Main;
2
3 import java.io.*;
4
5 /**
6  *
7  * @author juancfm
8  */
9 public class Main {
10
11     public static void main(String[] args) {
12         /**
13          * 10.14. Escribe un programa que lea un fichero de texto llamado
14          * carta.txt. Tenemos que contar los caracteres, las líneas y las
15          * palabras. Para simplificar supondremos que cada palabra está separada
16          * de otra por un único espacio en blanco o por un cambio de línea.
17          */
18         String filename = "carta.txt";
19         int numCaracteres = 0;
20         int numLineas = 0;
21         int numPalabras = 0;
22
23         try (BufferedReader br =
24             new BufferedReader(new FileReader(filename))) {
25             String line;
26             while ((line = br.readLine()) != null) {
27                 numLineas++;
28                 if (!line.trim().isEmpty()) {
29                     String[] palabras = line.trim().split(regex: "\\s+");
30                     numPalabras += palabras.length;
31                     for (String palabra : palabras) {
32                         numCaracteres += palabra.length();
33                     }
34                 }
35             }
36         } catch (IOException e) {
37             System.err.println("Error al leer el archivo: " + e.getMessage());
38         }
39
40         System.out.println("Número de caracteres (sin espacios en blanco: "
41             + numCaracteres);
42         System.out.println("Número de líneas: " + numLineas);
43         System.out.println("Número de palabras: " + numPalabras);
44     }
45 }
46
47 }
```



[Volver al índice](#)

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.

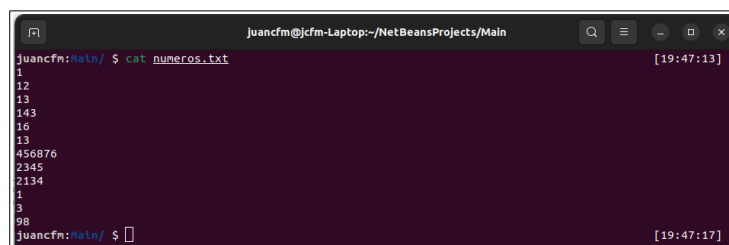


```
1 package Main;
2
3 import java.io.*;
4
5 /**
6  *
7  * @author juancfm
8  */
9 public class Main {
10
11     public static void main(String[] args) {
12         /**
13          * 10.15. En el archivo numeros.txt disponemos de una serie de números
14          * (uno por cada línea). Diseña un programa que procese el fichero y nos
15          * muestre el menor y el mayor.
16          */
17         String archivo = "numeros.txt";
18         int minimo = Integer.MAX_VALUE;
19         int maximo = Integer.MIN_VALUE;
20         try (BufferedReader br =
21             new BufferedReader(new FileReader(archivo))) {
22             String linea;
23             while ((linea = br.readLine()) != null) {
24                 int numero = Integer.parseInt(linea);
25                 if (numero < minimo) {
26                     minimo = numero;
27                 }
28                 if (numero > maximo) {
29                     maximo = numero;
30                 }
31             }
32             System.out.println("El mínimo es: " + minimo);
33             System.out.println("El máximo es: " + maximo);
34         } catch (IOException e) {
35             System.err.println("Ha ocurrido un error: " + e.getMessage());
36         }
37     }
38 }
39
```

Output x

Main (run) x Main (run) #2 x

run:
El mínimo es: 1
El máximo es: 456876
BUILD SUCCESSFUL (total time: 0 seconds)



```
juancfm@jcfm-Laptop:~/NetBeansProjects/Main
juancfm:~$ cat numeros.txt
1
12
13
143
16
13
456876
2345
2134
1
3
98
juancfm:~$
```

[Volver al índice](#)

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 .

```
1 package Main;
2
3 import java.io.*;
4 import java.util.Scanner;
5
6 /**
7  *
8  * @author juancfm
9  */
10 public class Main {
11
12     public static void main(String[] args) {
13
14         /**
15          * 10.16. Un libro de firmas es útil para recoger los nombres de todas
16          * las personas que han pasado por un determinado lugar. Crea una
17          * aplicación que permita mostrar el libro de firmas e insertar un nuevo
18          * nombre (comprobando que no se encuentre repetido). Llamaremos al
19          * fichero firmas.txt .
20          */
21
22         // Comprobamos si el archivo existe
23         File archivo = new File("firmas.txt");
24         if (!archivo.exists()) {
25             try {
26                 archivo.createNewFile();
27                 System.out.println("El archivo firmas.txt ha sido creado.\n");
28             } catch (IOException e) {
29                 System.out.println("No se pudo crear el archivo firmas.txt.");
30                 return;
31             }
32
33             LibroFirmas libro = new LibroFirmas();
34
35             try (Scanner scanner = new Scanner(System.in)) {
36                 OUTER:
37                 while (true) {
38                     System.out.println("¿Qué quieres hacer?");
39                     System.out.println("1. Mostrar firmas");
40                     System.out.println("2. Agregar firma");
41                     System.out.println("3. Salir");
42                     String opcion = scanner.nextLine();
43                     switch (opcion) {
44                         case "1" -> {
45                             System.out.println("Firmas en el libro:");
46                             libro.mostrarFirmas();
47                         }
48                         case "2" -> {
49                             System.out.println("Ingresa el nombre:");
50                             String nombre = scanner.nextLine();
51                             libro.agregarFirma(nombre);
52                         }
53                         case "3" -> {
54                             break OUTER;
55                         }
56                         default -> System.out.println("Opción inválida.");
57                     }
58                 }
59                 System.out.println("#####");
60                 System.out.println("¡Gracias por usar este programa!");
61                 System.out.println("#####");
62             }
63         }
64     }
65 }
```

```
1 package Main;
2 import java.io.*;
3 import java.util.ArrayList;
4
5 /**
6  *
7  * @author juancfm
8  */
9 class LibroFirmas {
10     private final ArrayList<Firma> firmas;
11     private static final String NOMBRE_ARCHIVO = "firmas.txt";
12
13     public LibroFirmas() {
14         this.firmas = new ArrayList<Firma>();
15         leerFirmas();
16     }
17
18     public void mostrarFirmas() {
19         for (Firma firma : firmas) {
20             System.out.println(firma.getNombre());
21         }
22     }
23
24     public boolean agregarFirma(String nombre) {
25         boolean status = true;
26         if (buscarFirma(nombre)) {
27             System.out.println("El nombre ya se encuentra en el libro.");
28             status = false;
29         } else {
30             Firma firma = new Firma(nombre);
31             firmas.add(firma);
32             escribirFirma(firma);
33             System.out.println("Firma agregada al libro.");
34         }
35         return status;
36     }
37
38     private boolean buscarFirma(String nombre) {
39         boolean status = false;
40         for (Firma firma : firmas) {
41             if (firma.getNombre().equalsIgnoreCase(nombre)) {
42                 status = true;
43             }
44         }
45         return status;
46     }
47
48     private void leerFirmas() {
49         try {
50             try (BufferedReader bR =
51                 new BufferedReader(new FileReader(NOMBRE_ARCHIVO))) {
52                 String linea = bR.readLine();
53                 while (linea != null) {
54                     Firma firma = new Firma(linea);
55                     firmas.add(firma);
56                     linea = bR.readLine();
57                 }
58             }
59         } catch (IOException e) {
60             System.out.println("Error al leer el archivo de firmas: "
61                 + e.getMessage());
62         }
63     }
64 }
```

```
1 package Main;
2
3 /**
4  *
5  * @author juancfm
6  */
7 class Firma {
8     private final String nombre;
9
10     public Firma(String nombre) {
11         this.nombre = nombre;
12     }
13
14     public String getNombre() {
15         return nombre;
16     }
17 }
18
```

```
64 private void escribirFirma(Firma firma) {
65     try {
66         try (FileWriter fW =
67             new FileWriter(NOMBRE_ARCHIVO, append: true)) {
68             fW.write(firma.getNombre() + "\n");
69         }
70     } catch (IOException e) {
71         System.out.println("Error al escribir en el archivo de firmas: "
72             + e.getMessage());
73     }
74 }
75
76
77 }
```

Actividades de la Unidad 9: Ficheros de texto

```
run:
El archivo firmas.txt ha sido creado.

¿Qué quieres hacer?
1. Mostrar firmas
2. Agregar firma
3. Salir
2
Ingresa el nombre:
Juan
Firma agregada al libro.
¿Qué quieres hacer?
1. Mostrar firmas
2. Agregar firma
3. Salir
2
Ingresa el nombre:
Juan
El nombre ya se encuentra en el libro.
```

```
El nombre ya se encuentra en el libro.
¿Qué quieres hacer?
1. Mostrar firmas
2. Agregar firma
3. Salir
2
Ingresa el nombre:
Pedro
Firma agregada al libro.
¿Qué quieres hacer?
1. Mostrar firmas
2. Agregar firma
3. Salir
1
Firmas en el libro:
Juan
Pedro
¿Qué quieres hacer?
1. Mostrar firmas
2. Agregar firma
3. Salir
3
#####
Gracias por usar este programa.
#####
BUILD SUCCESSFUL (total time: 53 seconds)
```

[Volver al índice](#)

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.

```
1  package Main;
2
3  import java.io.*;
4  import java.util.Scanner;
5
6  /**
7   *
8   * @author juancfm
9   */
10 public class Main {
11
12     public static void main(String[] args) {
13         /**
14          * 10.17. En Linux disponemos del comando more, al que se le pasa un
15          * fichero y lo muestra poco a poco: cada 24 líneas. Implementa un
16          * programa que funcione de forma similar.
17          */
18
19         Scanner scanner = new Scanner( source: System.in );
20
21         System.out.println( x: "Por favor ingresa el nombre del archivo: " );
22         String nombreArchivo = scanner.nextLine();
23
24         File archivo = new File( pathname: nombreArchivo );
25         try {
26             try (Scanner lectorArchivo = new Scanner( source: archivo )) {
27                 int contadorLineas = 0;
28                 while (lectorArchivo.hasNextLine()) {
29                     String linea = lectorArchivo.nextLine();
30                     System.out.println( x: linea );
31                     contadorLineas++;
32                     if (contadorLineas % 24 == 0) {
33                         System.out.println(
34                             x: "\n--- Presiona Enter para continuar ---");
35                         scanner.nextLine();
36                     }
37                 }
38             }
39         } catch (FileNotFoundException e) {
40             System.out.println( x: "El archivo no existe." );
41         }
42
43         System.out.println( x: "#####");
44         System.out.println( x: "\tGracias por usar este programa." );
45         System.out.println( x: "#####");
46
47     }
48 }
49
```

Actividades de la Unidad 9: Ficheros de texto

```
38 | }  
Output - Main (run) x  
run:  
Por favor ingresa el nombre del archivo:  
carta.txt
```

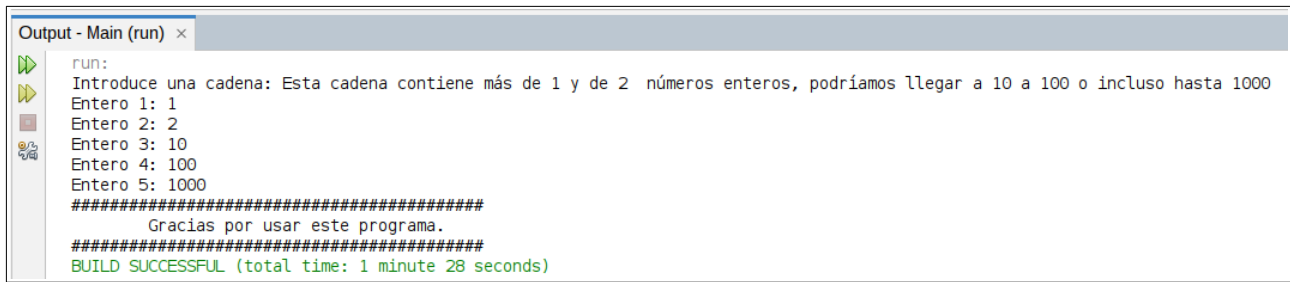
```
Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Etiam  
Vivamus at lacus eget arcu consequat auctor.  
Donec mi dolor, pellentesque non malesuada at, sollicitudin vitae dui.  
  
--- Presiona Enter para continuar ---  
  
Sed nec libero ut dui scelerisque porta bibendum eu mi.  
Praesent interdum ex justo, et gravida magna convallis eu. Sed tristique gravida rutrum.  
#####  
Gracias por usar este programa.  
#####  
BUILD SUCCESSFUL (total time: 32 seconds)
```

[Volver al índice](#)

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.

```
1 package Main;
2
3 import java.io.*;
4 import java.util.Scanner;
5
6 /**
7  *
8  * @author juancfm
9  */
10 public class Main {
11
12     public static void main(String[] args) {
13         /**
14          * 10.18. Escribe la función Integer[ ] leerEnteros(String texto ), al
15          * que se le pasa una cadena y devuelve una tabla con todos los enteros
16          * que aparecen en ella.
17          */
18
19         Scanner scanner = new Scanner( source: System.in);
20         System.out.print( s: "Introduce una cadena: ");
21         String cadena = scanner.nextLine();
22         Integer[] enteros = leerEnteros( texto: cadena);
23         int i = 0;
24         for (int entero : enteros) {
25             i++;
26             System.out.println("Entero " + i + ": " + entero);
27         }
28
29         System.out.println( x: "#####");
30         System.out.println( x: "\tGracias por usar este programa.");
31         System.out.println( x: "#####");
32     }
33
34     public static Integer[] leerEnteros(String texto) {
35         String[] palabras = texto.split( regex: "\\s+");
36         Integer[] enteros = new Integer[palabras.length];
37         int j = 0;
38         for (String palabra : palabras) {
39             try {
40                 enteros[j] = Integer.valueOf( s: palabra);
41                 j++;
42             } catch (NumberFormatException e) {
43                 // Ignorar palabras que no son enteros
44             }
45         }
46         return java.util.Arrays.copyOf( original: enteros, newLength: j);
47     }
48
49 }
50
```

Actividades de la Unidad 9: Ficheros de texto

A screenshot of a terminal window titled "Output - Main (run)". The window shows the execution of a program. The output text is as follows:

```
run:
Introduce una cadena: Esta cadena contiene más de 1 y de 2 números enteros, podríamos llegar a 10 a 100 o incluso hasta 1000
Entero 1: 1
Entero 2: 2
Entero 3: 10
Entero 4: 100
Entero 5: 1000
#####
      Gracias por usar este programa.
#####
BUILD SUCCESSFUL (total time: 1 minute 28 seconds)
```

[Volver al índice](#)

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.

[Volver al índice](#)