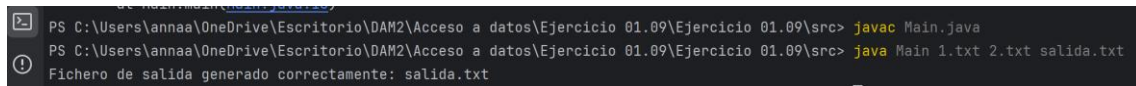


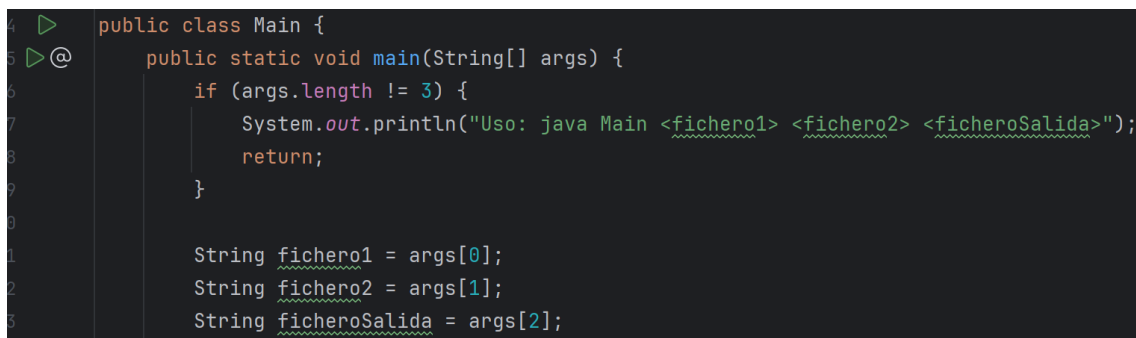
## Ejercicio 01.09

Este programa tiene como propósito combinar el contenido de dos ficheros de texto en un tercero, pero ordenando alfabéticamente todas sus líneas. A diferencia de los anteriores ejercicios, en este caso los nombres de los ficheros no se introducen mediante teclado durante la ejecución, sino que se reciben directamente como argumentos desde la línea de comandos. Esto significa que al ejecutar el programa en consola deben indicarse los tres ficheros: el primero y el segundo como entrada, y el tercero como salida.



```
PS C:\Users\annaa\OneDrive\Escritorio\DAM2\Acceso a datos\Ejercicio 01.09\Ejercicio 01.09\src> javac Main.java
PS C:\Users\annaa\OneDrive\Escritorio\DAM2\Acceso a datos\Ejercicio 01.09\Ejercicio 01.09\src> java Main 1.txt 2.txt salida.txt
Fichero de salida generado correctamente: salida.txt
```

El programa comienza comprobando si el número de argumentos proporcionados es correcto. Si no se introducen exactamente tres, muestra un mensaje de ayuda con la forma correcta de ejecución y finaliza. En caso de que se pasen los tres nombres, los guarda en las variables correspondientes: `fichero1`, `fichero2` y `ficheroSalida`.



```
public class Main {
    public static void main(String[] args) {
        if (args.length != 3) {
            System.out.println("Uso: java Main <fichero1> <fichero2> <ficheroSalida>");
            return;
        }

        String fichero1 = args[0];
        String fichero2 = args[1];
        String ficheroSalida = args[2];
    }
}
```

A continuación, se crea una lista de tipo `ArrayList` en la que se almacenarán todas las líneas leídas de los dos ficheros de entrada. Para leer el contenido se utilizan objetos `BufferedReader` junto con `FileReader`. Primero se abre el primer fichero y se recorren sus líneas, que se van añadiendo una por una a la lista. Una vez terminado, se repite el mismo proceso con el segundo fichero, de modo que al final la lista contiene las líneas de ambos archivos. Si ocurre algún error durante la lectura, como que el fichero no exista o no se pueda abrir, el programa captura la excepción y finaliza mostrando un mensaje de error.

```
List<String> lineas = new ArrayList<>();

// Leer el contenido del primer fichero
try (BufferedReader br = new BufferedReader(new FileReader(fichero1))) {
    String linea;
    while ((linea = br.readLine()) != null) {
        lineas.add(linea);
    }
} catch (IOException e) {
    System.err.println("Error al leer el fichero: " + fichero1);
    e.printStackTrace();
    return;
}

// Leer el contenido del segundo fichero
try (BufferedReader br = new BufferedReader(new FileReader(fichero2))) {
    String linea;
    while ((linea = br.readLine()) != null) {
        lineas.add(linea);
    }
} catch (IOException e) {
    System.err.println("Error al leer el fichero: " + fichero2);
    e.printStackTrace();
    return;
}
```

Cuando ya se tiene toda la información, el programa procede a ordenar la lista alfabéticamente. Para ello utiliza el método `Collections.sort()`, que reorganiza los elementos de la lista de menor a mayor siguiendo el orden natural de las cadenas de texto. Este paso es el que garantiza que el fichero de salida contenga los datos en el orden correcto.

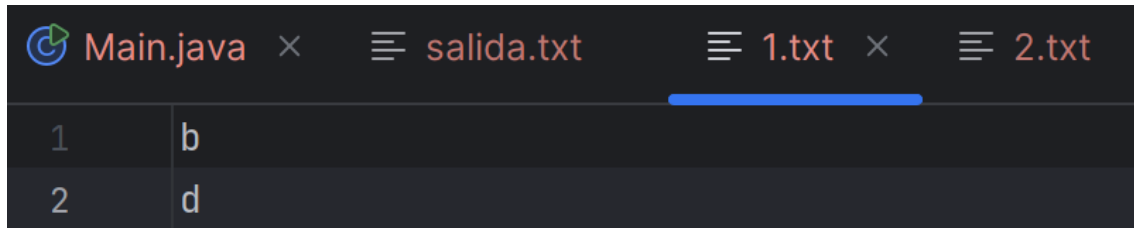
Por último, se crea un `BufferedWriter` asociado a un `FileWriter` para escribir el contenido en el fichero de salida. El programa recorre la lista ya ordenada y escribe cada línea seguida de un salto de línea, de manera que el archivo resultante conserva la estructura de texto original pero con los elementos ordenados. Si surge un problema al escribir, también se captura la excepción correspondiente para informar al usuario.

```
// Ordenar las lineas alfabéticamente
Collections.sort(lineas);

// Escribir el contenido ordenado en el fichero de salida
try (BufferedWriter bw = new BufferedWriter(new FileWriter(ficheroSalida))) {
    for (String linea : lineas) {
        bw.write(linea);
        bw.newLine();
    }
} catch (IOException e) {
    System.err.println("Error al escribir en el fichero: " + ficheroSalida);
    e.printStackTrace();
}

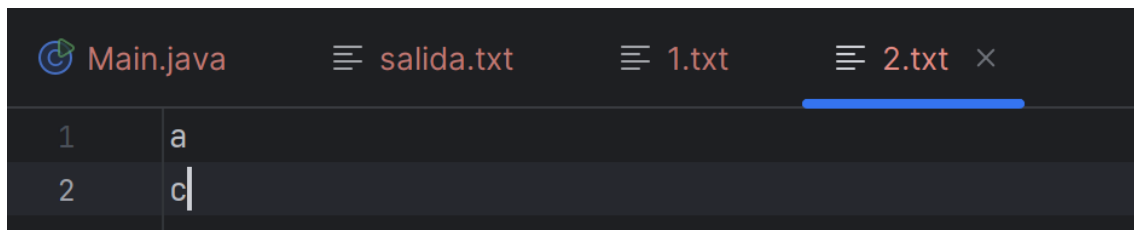
System.out.println("Fichero de salida generado correctamente: " + ficheroSalida);
}
```

El programa finaliza mostrando un mensaje en la consola que confirma que el fichero de salida se ha generado correctamente. En resumen, se trata de un programa que integra varias funcionalidades importantes: la lectura de ficheros, el uso de listas dinámicas para almacenar datos, la ordenación alfabética con colecciones y la escritura de resultados en un nuevo archivo. Además, introduce la práctica de trabajar con argumentos de línea de comandos, lo que aporta mayor flexibilidad en su uso.



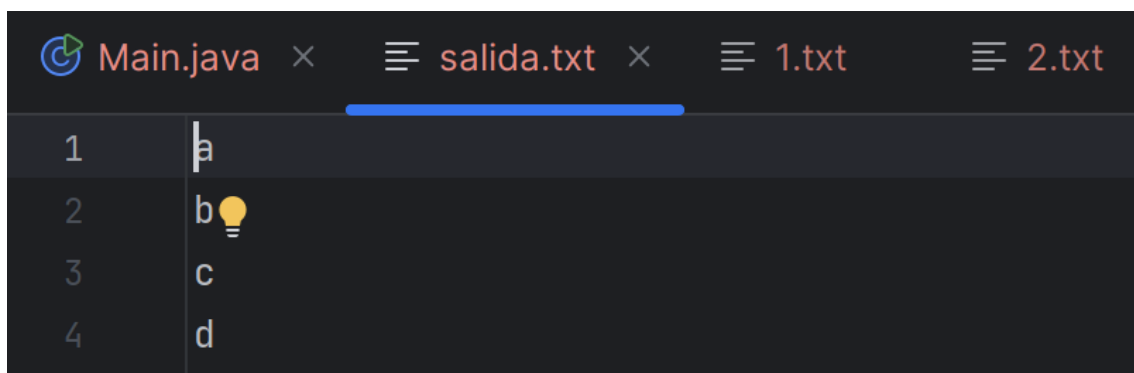
The screenshot shows an IDE window titled '1.txt'. The editor contains two lines of text: 'b' on line 1 and 'd' on line 2. The file is open alongside 'Main.java', 'salida.txt', and '2.txt'.

Line	Content
1	b
2	d



The screenshot shows an IDE window titled '2.txt'. The editor contains two lines of text: 'a' on line 1 and 'c' on line 2. The file is open alongside 'Main.java', 'salida.txt', and '1.txt'.

Line	Content
1	a
2	c



The screenshot shows an IDE window titled 'salida.txt'. The editor contains four lines of text: 'a' on line 1, 'b' on line 2, 'c' on line 3, and 'd' on line 4. The file is open alongside 'Main.java', '1.txt', and '2.txt'. A yellow lightbulb icon is visible next to the text 'b' on line 2.

Line	Content
1	a
2	b
3	c
4	d