



**Ikaskuntza Birtual eta Digitalizatuen LHII**  
CIFP de Aprendizajes Virtuales y Digitalizados

# **AD01**

# **Tarea de Evaluación 1**

**Fco. Javier Fernández Rodrigo**  
**DAM - Acceso a Datos**

**Curso: 2024/25**



## ÍNDICE

1. INTRODUCCIÓN.....	1
2. DIFERENTES APARTADOS DE LA TAREA.....	1
2.1. Ejercicio 1 - Flujos de caracteres: (FileReader, FileWriter) (1,5 puntos).....	1
2.2. Ejercicio 2 - Flujos de caracteres: (BufferedReader, BufferedWriter) (1,5 puntos).....	3
2.3. Ejercicio 3 - (Flujos binarios:InputStream) (1,5 puntos).....	4
3. AUTOEVALUACIÓN.....	4
4. BIBLIOGRAFÍA.....	4

## 1. INTRODUCCIÓN

Para la realización de la tarea evaluativa os podéis apoyar en el código utilizado en las tareas de aprendizaje y también en las herramientas de inteligencia artificial generativa como ChatGPT, Bing o Bard. Como se ha comentado anteriormente, estas herramientas pueden ser muy útiles si se usan con criterio. Es importante que probéis, testeéis y, sobre todo, entendáis bien el código que se entrega ya que en el examen tendréis que ser capaces de realizar ejercicios similares sin su uso.

Solamente se aceptarán entregas en el AULA VIRTUAL y dentro del plazo indicado.

Deberéis realizar una **autoevaluación** siguiendo la rúbrica definida en el apartado "Criterios de calificación". Debéis indicar la nota que creéis que os corresponde en cada ejercicio y explicar por qué, adjuntando pruebas de vuestro programa (pantallazos de la compilación, de la ejecución o del código del programa) que lo demuestren. **Lo podéis realizar en un documento o en un video.** Es importante que indiquéis qué tipo de **interacción habéis tenido con la inteligencia artificial**.

## 2. DIFERENTES APARTADOS DE LA TAREA

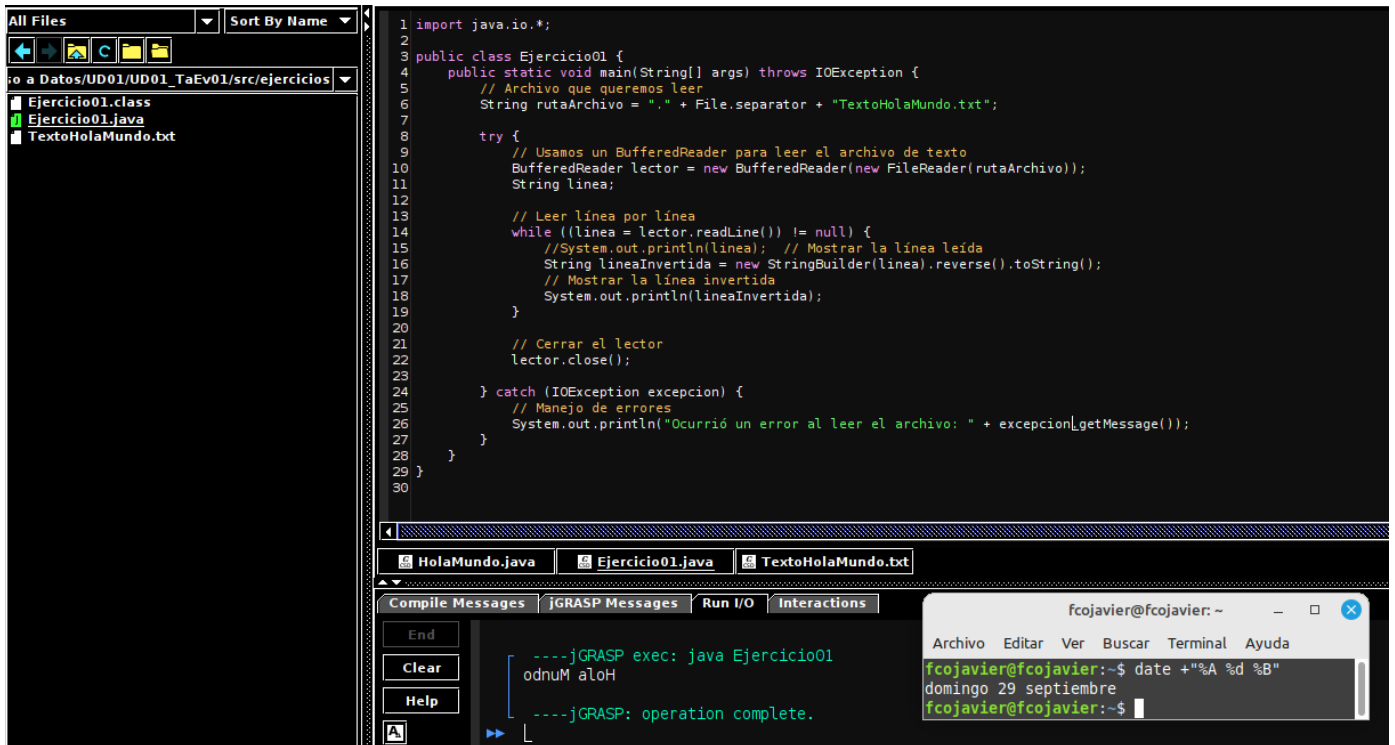
### 2.1. Ejercicio 1 - Flujos de caracteres: (FileReader, FileWriter) (1,5 puntos).

Escribe un programa en Java que lea una línea de texto desde un archivo y devuelva la misma línea, pero con las palabras invertidas individualmente. Es decir, si el archivo contiene "Hola Mundo", el archivo de salida debería contener "aloH odnuM".

```
1 import java.io.*;
2
3 public class Ejercicio01 {
4     public static void main(String[] args) throws IOException {
5         // Archivo que queremos leer
6         String rutaArchivo = "." + File.separator + "TextoHolaMundo.txt";
7
8         try {
9             // Usamos un BufferedReader para leer el archivo de texto
10            BufferedReader lector = new BufferedReader(new FileReader(rutaArchivo));
11            String linea;
12
13            // Leer línea por línea
14            while ((linea = lector.readLine()) != null) {
15                //System.out.println(linea); // Mostrar la línea leída
16                String lineaInvertida = new StringBuilder(linea).reverse().toString();
17                // Mostrar la línea invertida
18                System.out.println(lineaInvertida);
19            }
20
21            // Cerrar el lector
22            lector.close();
23
24        } catch (IOException excepcion) {
25            // Manejo de errores
26            System.out.println("Ocurrió un error al leer el archivo: " + excepcion.getMessage());
27        }
28    }
29 }
30
```



El resultado esperado se imprime correctamente:



The screenshot shows an IDE with the following components:

- File Explorer:** Shows the project structure with files: `Ejercicio01.class`, `Ejercicio01.java`, and `TextoHolaMundo.txt`.
- Code Editor:** Contains the following Java code:
 

```

1 import java.io.*;
2
3 public class Ejercicio01 {
4     public static void main(String[] args) throws IOException {
5         // Archivo que queremos leer
6         String rutaArchivo = "." + File.separator + "TextoHolaMundo.txt";
7
8         try {
9             // Usamos un BufferedReader para leer el archivo de texto
10            BufferedReader lector = new BufferedReader(new FileReader(rutaArchivo));
11            String linea;
12
13            // Leer línea por línea
14            while ((linea = lector.readLine()) != null) {
15                //System.out.println(linea); // Mostrar la línea leída
16                String lineaInvertida = new StringBuilder(linea).reverse().toString();
17                // Mostrar la línea invertida
18                System.out.println(lineaInvertida);
19            }
20
21            // Cerrar el lector
22            lector.close();
23
24        } catch (IOException excepcion) {
25            // Manejo de errores
26            System.out.println("Ocurrió un error al leer el archivo: " + excepcion.getMessage());
27        }
28    }
29 }
30
      
```
- Output Console:** Shows the execution results:
 

```

----jGRASP exec: java Ejercicio01
odnuM aLoH
----jGRASP: operation complete.
      
```
- Terminal:** Shows the command prompt output:
 

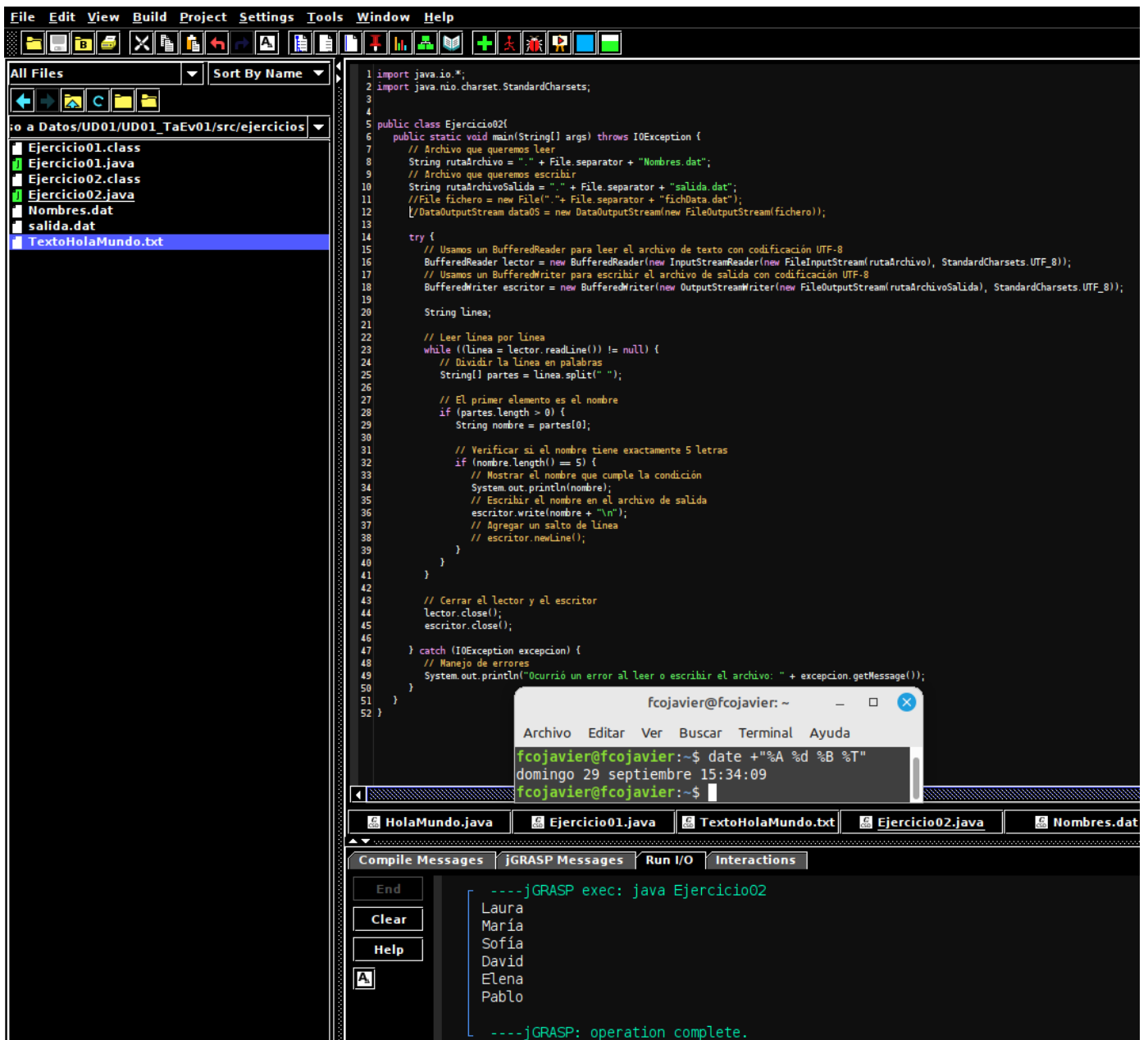
```

fcojavier@fcojavier: ~
Archivo Editar Ver Buscar Terminal Ayuda
fcojavier@fcojavier:~$ date +%A %d %B
domingo 29 septiembre
fcojavier@fcojavier:~$
      
```



## 2.2. Ejercicio 2 - Flujos de caracteres: (BufferedReader, BufferedWriter) (1,5 puntos).

Crea un programa en Java que lea un archivo con nombres y apellidos separados por espacios, y escriba en un nuevo archivo solo los nombres que tienen exactamente cinco letras.



```

1 import java.io.*;
2 import java.nio.charset.StandardCharsets;
3
4
5 public class Ejercicio02 {
6     public static void main(String[] args) throws IOException {
7         // Archivo que queremos leer
8         String rutaArchivo = "." + File.separator + "Nombres.dat";
9         // Archivo que queremos escribir
10        String rutaArchivoSalida = "." + File.separator + "salida.dat";
11        //File fichero = new File("." + File.separator + "fichData.dat");
12        //DataOutputStream dataOS = new DataOutputStream(new FileOutputStream(fichero));
13
14        try {
15            // Usamos un BufferedReader para leer el archivo de texto con codificación UTF-8
16            BufferedReader lector = new BufferedReader(new InputStreamReader(new FileInputStream(rutaArchivo), StandardCharsets.UTF_8));
17            // Usamos un BufferedWriter para escribir el archivo de salida con codificación UTF-8
18            BufferedWriter escritor = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(rutaArchivoSalida), StandardCharsets.UTF_8));
19
20            String linea;
21
22            // Leer línea por línea
23            while ((linea = lector.readLine()) != null) {
24                // Dividir la línea en palabras
25                String[] partes = linea.split(" ");
26
27                // El primer elemento es el nombre
28                if (partes.length > 0) {
29                    String nombre = partes[0];
30
31                    // Verificar si el nombre tiene exactamente 5 letras
32                    if (nombre.length() == 5) {
33                        // Mostrar el nombre que cumple la condición
34                        System.out.println(nombre);
35                        // Escribir el nombre en el archivo de salida
36                        escritor.write(nombre + "\n");
37                        // Agregar un salto de línea
38                        // escritor.newLine();
39                    }
40                }
41            }
42
43            // Cerrar el lector y el escritor
44            lector.close();
45            escritor.close();
46
47        } catch (IOException exception) {
48            // Manejo de errores
49            System.out.println("Ocurrió un error al leer o escribir el archivo: " + exception.getMessage());
50        }
51    }
52 }

```

```

fcojavier@fcojavier: ~
Archivo Editar Ver Buscar Terminal Ayuda
fcojavier@fcojavier:~$ date +%A %d %B %T
domingo 29 septiembre 15:34:09
fcojavier@fcojavier:~$

```

```

----jGRASP exec: java Ejercicio02
Laura
María
Sofía
David
Elena
Pablo
----jGRASP: operation complete.

```

Al igual que en el ejercicio anterior, he tenido problemas con el formato del texto, que he terminado solucionando con chatgpt.



### 2.3. Ejercicio 3 - (Flujos binarios:InputStream) (1,5 puntos).

Realiza un programa en Java que lea la cabecera de un fichero PDF y verifique si realmente se trata de un archivo PDF válido. Para que un archivo sea un PDF válido, debe comenzar con la siguiente secuencia de bytes: {37, 80, 68, 70}. Si la cabecera no coincide con esta secuencia, el programa debe informar al usuario de que el archivo no es válido (1 punto).

## 3. AUTOEVALUACIÓN

## 4. BIBLIOGRAFÍA

