



Ikaskuntza Birtual eta Digitalizatuen LHII
CIFP de Aprendizajes Virtuales y Digitalizados

Tarea de Evaluación 01

Miguel Damborenea Terreros
Acceso a datos

Curso: 2024/25

EUSKO JAURLARITZA



GOBIERNO VASCO

HEZKUNTZA SAILA
Lanbide Heziketako Sailburuordetza

DEPARTAMENTO DE EDUCACIÓN
Viceconsejería de Formación Profesional



Ikaskuntza Birtual eta Digitalizatuen LHII
CIFP de Aprendizajes Virtuales y Digitalizados

ÍNDICE

1. AUTOEVALUACIÓN	1
2. BIBLIOGRAFÍA	8



1. AUTOEVALUACIÓN

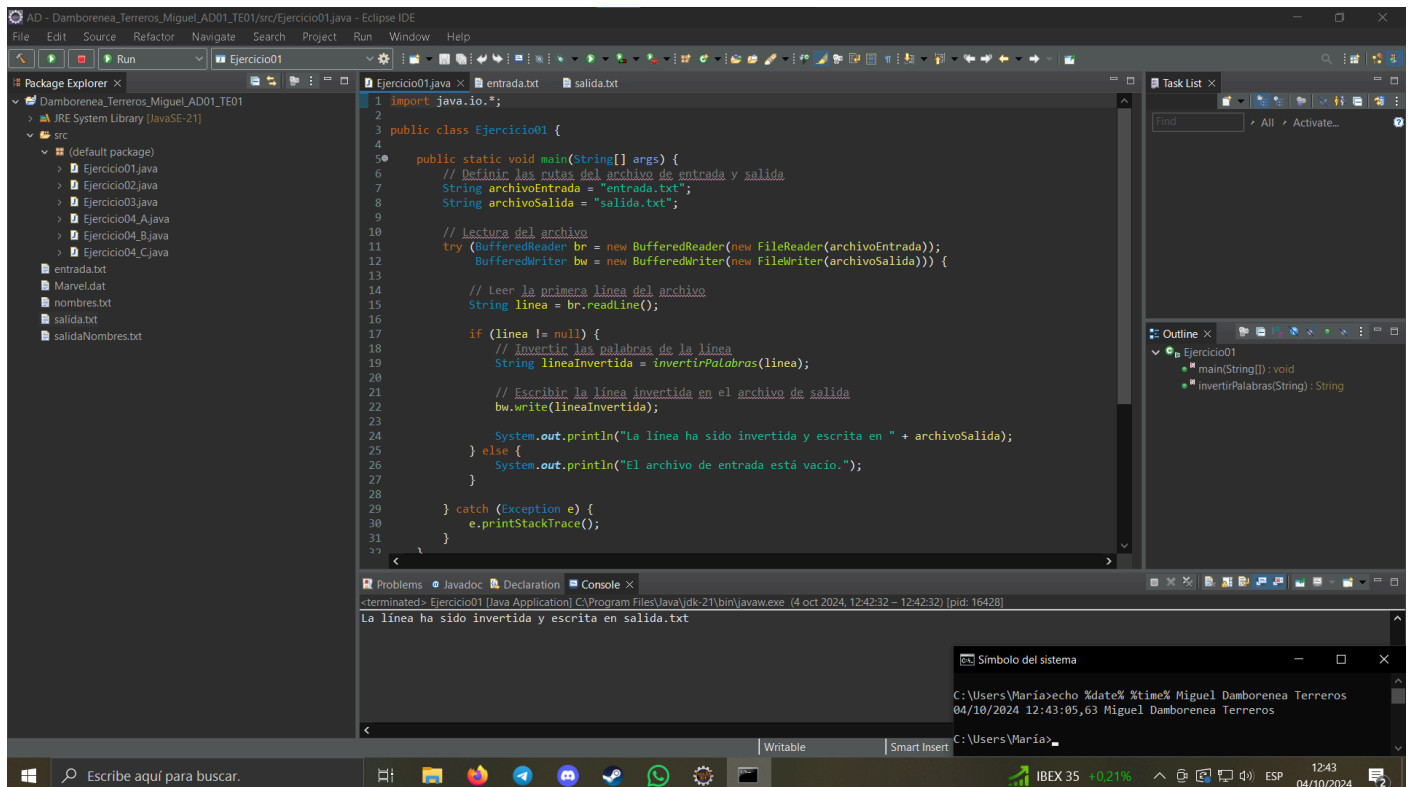
En base a la siguiente autoevaluación, considero que mi nota pasaría a ser un **9**.

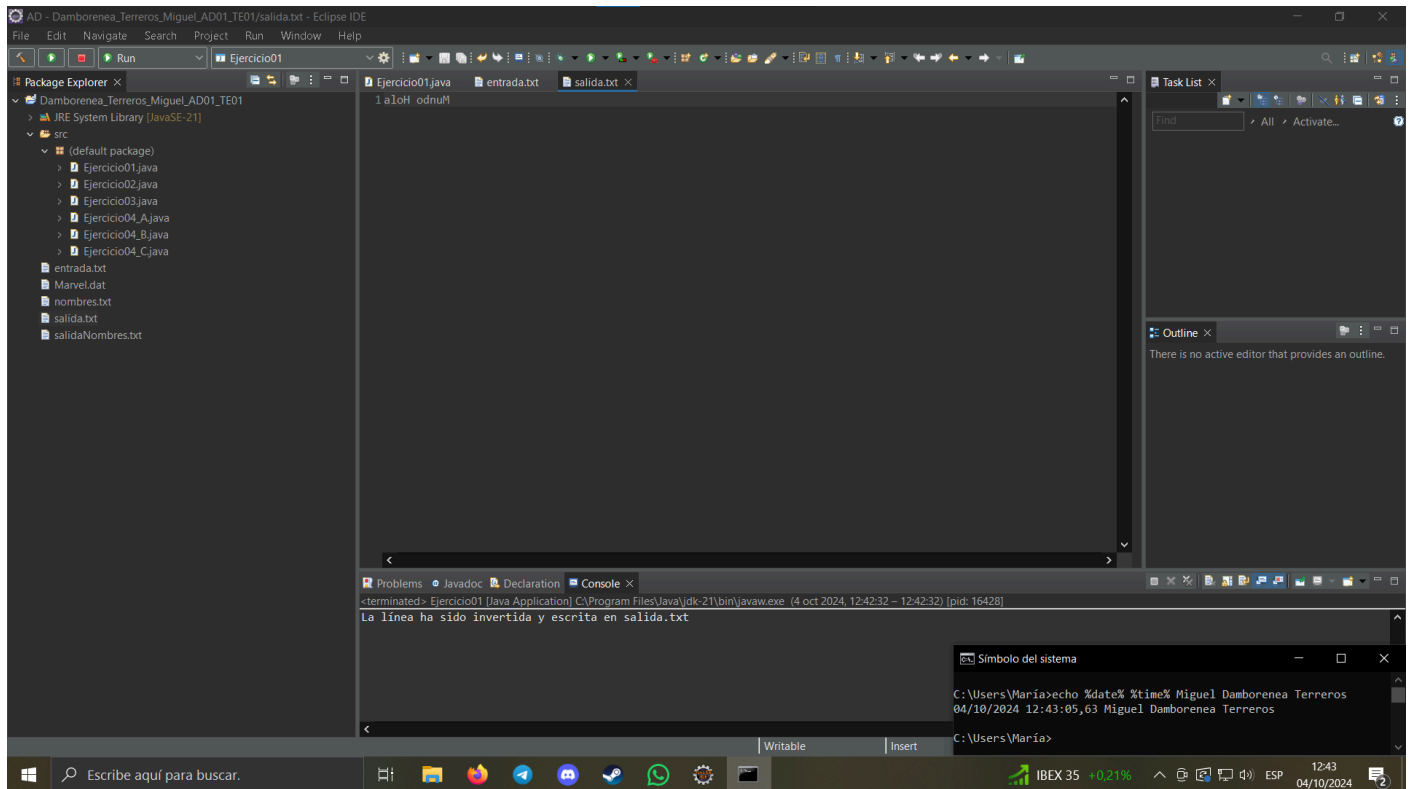
Enlace a github con los ejercicios:

https://github.com/Mdambo/AD_TE01.git

1. Flujos de caracteres: (FileReader, FileWriter) (1,5 puntos)

El programa cumple la función indicada en el ejercicio a la perfección. Está comentado de forma adecuada y no tiene redundancias. ChatGPT ha sido usado para preguntar cómo funciona el try catch con recursos y el Stringbuilder, así como mejoras posibles al código.

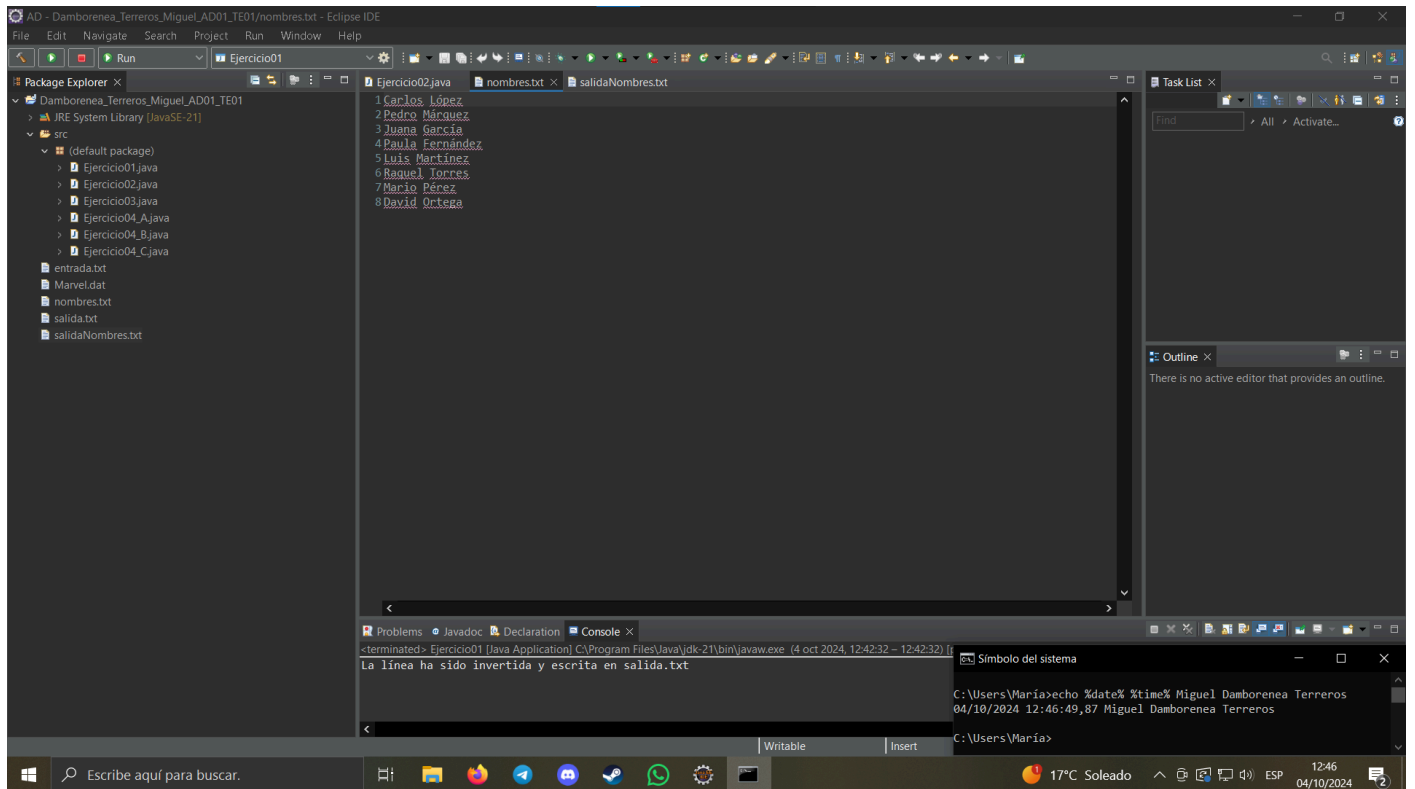




2. Flujos de caracteres: (BufferedReader, BufferedWriter) (1,5 puntos)

Como el anterior, el programa funciona en base a los requisitos del ejercicio y está comentado adecuadamente. ChatGPT ha sido usado para comprobar que el código no es redundante y posibles mejoras.



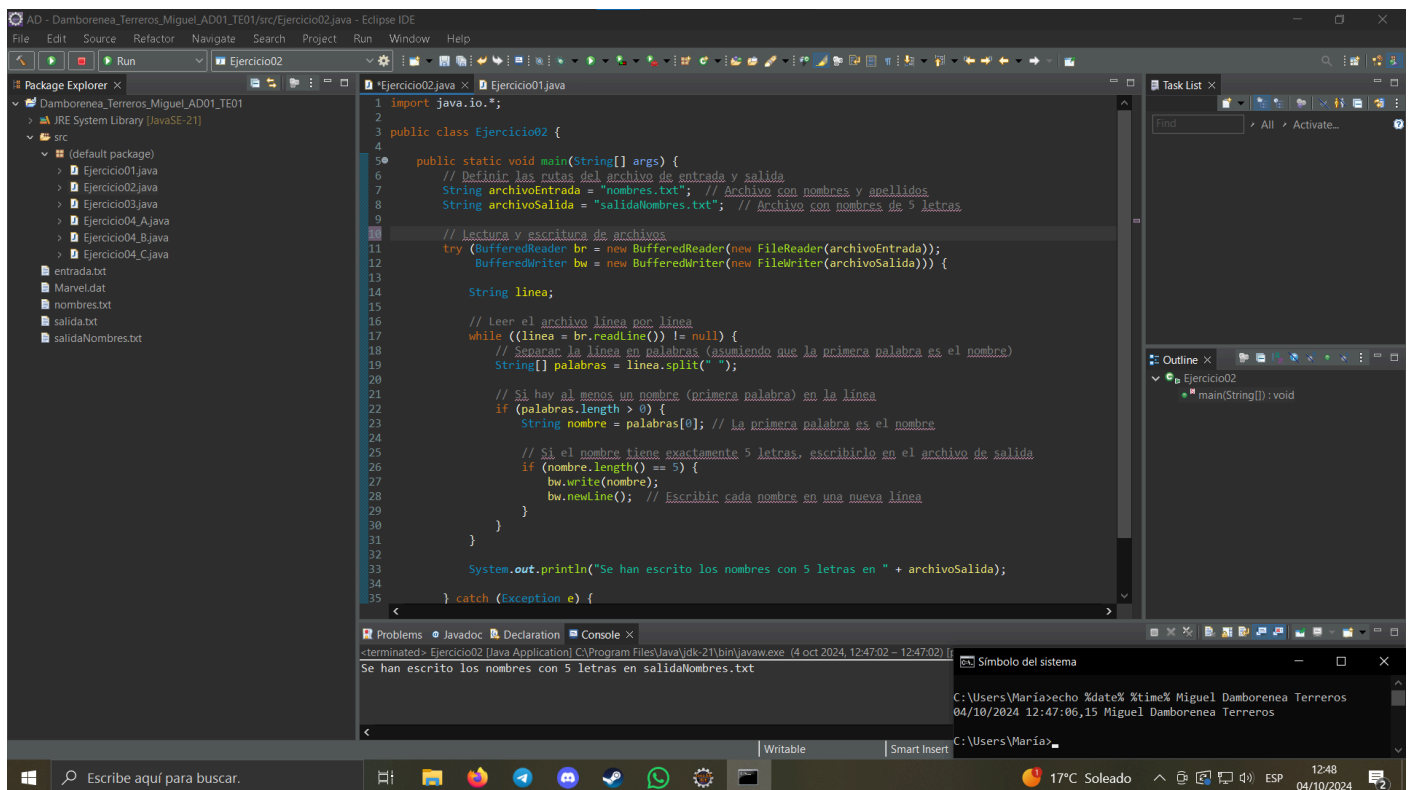


The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure for 'Damborenea_Terrosos_Miguel_AD01_TE01'. The main editor window shows 'Ejercicio01.java' with the following code:

```
1 Carlos López
2 Pedro Márquez
3 Juana García
4 Paula Fernández
5 Luis Martínez
6 Raquel Torres
7 Mario Pérez
8 David Ortega
```

The Console window at the bottom shows the output of the program:

```
<terminated> Ejercicio01 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (4 oct 2024, 12:42:32) [
La línea ha sido invertida y escrita en salida.txt
```



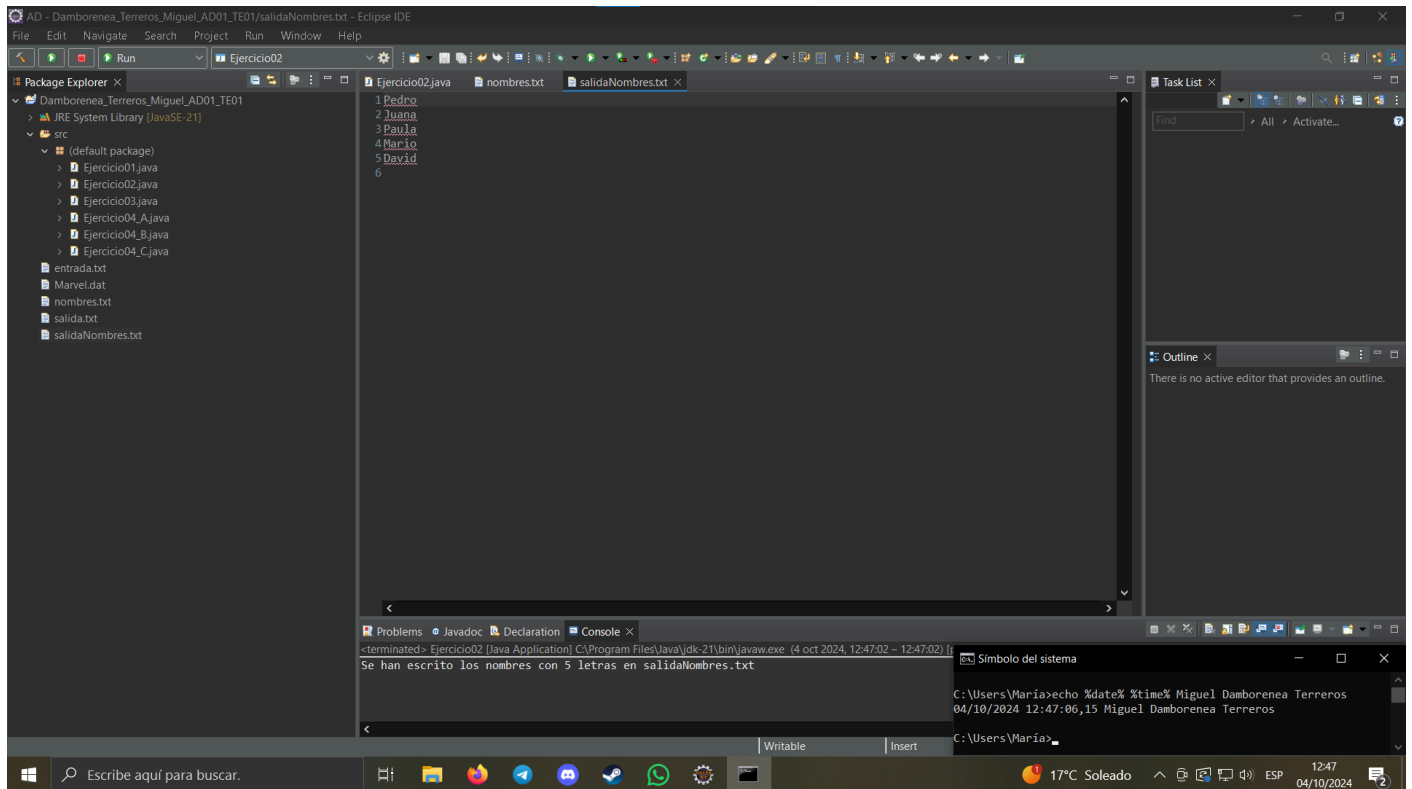
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure for 'Damborenea_Terrosos_Miguel_AD01_TE01'. The main editor window shows 'Ejercicio02.java' with the following code:

```
1 import java.io.*;
2
3 public class Ejercicio02 {
4
5     public static void main(String[] args) {
6         // Definir las rutas del archivo de entrada y salida
7         String archivoEntrada = "nombres.txt"; // Archivo con nombres y apellidos
8         String archivoSalida = "salidaNombres.txt"; // Archivo con nombres de 5 letras
9
10        // Lectura y escritura de archivos
11        try (BufferedReader br = new BufferedReader(new FileReader(archivoEntrada));
12             BufferedWriter bw = new BufferedWriter(new FileWriter(archivoSalida))) {
13
14            String linea;
15
16            // Leer el archivo línea por línea
17            while ((linea = br.readLine()) != null) {
18                // Separar la línea en palabras (asumiendo que la primera palabra es el nombre)
19                String[] palabras = linea.split(" ");
20
21                // Si hay al menos un nombre (primera palabra) en la línea
22                if (palabras.length > 0) {
23                    String nombre = palabras[0]; // La primera palabra es el nombre
24
25                    // Si el nombre tiene exactamente 5 letras, escribirlo en el archivo de salida
26                    if (nombre.length() == 5) {
27                        bw.write(nombre);
28                        bw.newLine(); // Escribir cada nombre en una nueva línea
29                    }
30                }
31            }
32
33            System.out.println("Se han escrito los nombres con 5 letras en " + archivoSalida);
34        } catch (Exception e) {
35
36        }
37    }
38}
```

The Console window at the bottom shows the output of the program:

```
<terminated> Ejercicio02 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (4 oct 2024, 12:47:02) [
Se han escrito los nombres con 5 letras en salidaNombres.txt
```

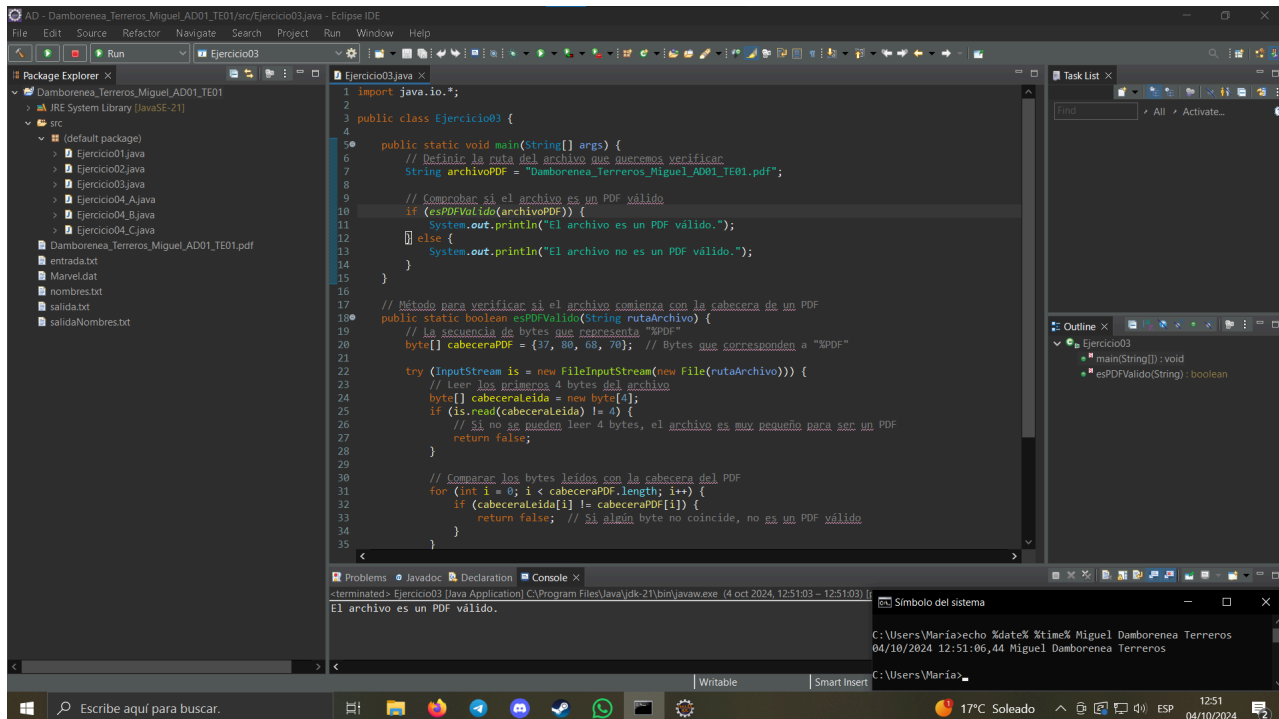




3. (Flujos binarios: InputStream) (1,5 puntos)

El programa realiza su función adecuadamente y está bien comentado. El ejemplo para mostrar su ejecución es con esta misma autoevaluación. No sabía muy bien cómo abordar este ejercicio, ya que era la primera vez que lo hacía, así que me he apoyado en la IA para preguntar dudas.





```

1 import java.io.*;
2
3 public class Ejercicio03 {
4
5     public static void main(String[] args) {
6         // Definir la ruta del archivo que queremos verificar
7         String archivoPDF = "Damborenea_Terreros_Miguel_AD01_TE01.pdf";
8
9         // Comprobar si el archivo es un PDF válido
10        if (esPDFValido(archivoPDF)) {
11            System.out.println("El archivo es un PDF válido.");
12        } else {
13            System.out.println("El archivo no es un PDF válido.");
14        }
15    }
16
17    // Método para verificar si el archivo comienza con la cabecera de un PDF
18    public static boolean esPDFValido(String rutaArchivo) {
19        // La secuencia de bytes que representa "PDF"
20        byte[] cabeceraPDF = {37, 80, 68, 70}; // Bytes que corresponden a "PDF"
21
22        try (InputStream is = new FileInputStream(new File(rutaArchivo))) {
23            // Leer los primeros 4 bytes del archivo
24            byte[] cabeceraLeida = new byte[4];
25            if (is.read(cabeceraLeida) != 4) {
26                // Si no se pueden leer 4 bytes, el archivo es muy pequeño para ser un PDF
27                return false;
28            }
29
30            // Comparar los bytes leídos con la cabecera del PDF
31            for (int i = 0; i < cabeceraPDF.length; i++) {
32                if (cabeceraLeida[i] != cabeceraPDF[i]) {
33                    return false; // Si algún byte no coincide, no es un PDF válido
34                }
35            }
36        }
37    }
38
39 }

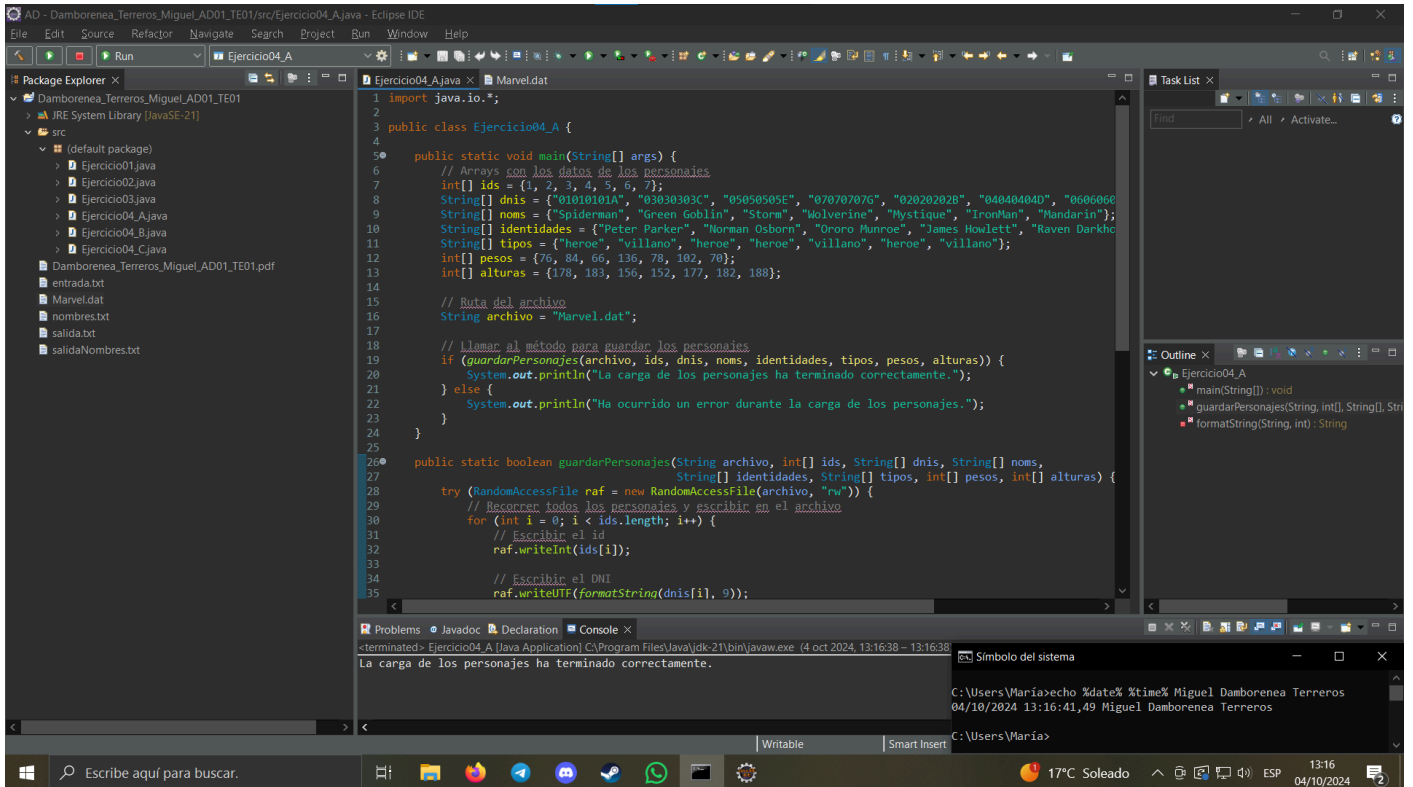
```

Console Output:
El archivo es un PDF válido.

4. (Acceso Aleatorio: RandomAccessFile)

a). El programa funciona adecuadamente y está comentado. La IA ha sido usada en un momento de bloqueo porque al cargar los datos al fichero luego no se cumplían las longitudes. Así que se ha añadido el StringBuilder para rellenar con espacios vacíos. También se ha usado para sugerencias de mejora.





```

1 import java.io.*;
2
3 public class Ejercicio04_A {
4
5     public static void main(String[] args) {
6         // Arrays con los datos de los personajes
7         int[] ids = {1, 2, 3, 4, 5, 6, 7};
8         String[] dnis = {"01010101A", "03030303C", "05050505E", "07070707G", "09090909I", "04040404D", "06060606F"};
9         String[] noms = {"Spiderman", "Green Goblin", "Storm", "Wolverine", "Mystique", "IronMan", "Mandarin"};
10        String[] identidades = {"Peter Parker", "Norman Osborn", "Ororo Munroe", "James Howlett", "Raven Darkholme", "Tony Stark", "Zhang Tong"};
11        String[] tipos = {"hero", "villano", "hero", "hero", "villano", "hero", "villano"};
12        int[] pesos = {76, 84, 66, 136, 78, 182, 70};
13        int[] alturas = {178, 183, 156, 152, 177, 182, 188};
14
15        // Ruta del archivo
16        String archivo = "Marvel.dat";
17
18        // Llamar al método para guardar los personajes
19        if (guardarPersonajes(archivo, ids, dnis, noms, identidades, tipos, pesos, alturas)) {
20            System.out.println("La carga de los personajes ha terminado correctamente.");
21        } else {
22            System.out.println("Ha ocurrido un error durante la carga de los personajes.");
23        }
24    }
25
26    public static boolean guardarPersonajes(String archivo, int[] ids, String[] dnis, String[] noms,
27        String[] identidades, String[] tipos, int[] pesos, int[] alturas) {
28        try (RandomAccessFile raf = new RandomAccessFile(archivo, "rw")) {
29            // Recorrer todos los personajes y escribir en el archivo
30            for (int i = 0; i < ids.length; i++) {
31                // Escribir el id
32                raf.writeInt(ids[i]);
33            }
34            // Escribir el DNI
35            raf.writeUTF(formatString(dnis[0], 9));
36        }
37    }
38
39    private static String formatString(String str, int length) {
40        return String.format("%-" + length + "s", str);
41    }
42}

```

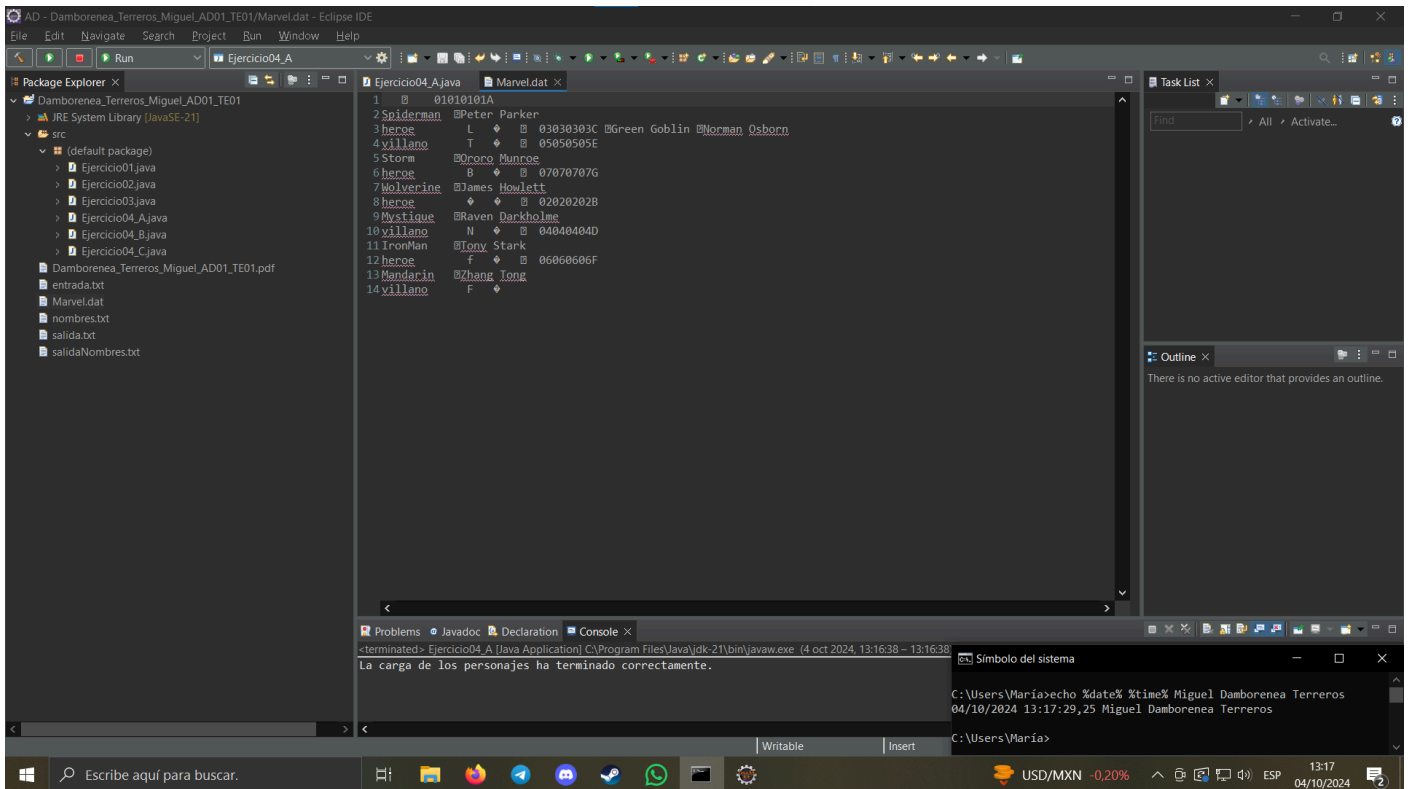
Problems • Javadoc • Declaration • Console ×

<terminated> Ejercicio04_A [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (4 oct 2024, 13:16:38 - 13:16:38)

La carga de los personajes ha terminado correctamente.

C:\Users\Wariajecho %date% %time% Miguel Damborenea Terreros
04/10/2024 13:16:41,49 Miguel Damborenea Terreros

C:\Users\Wariajecho



```

1 01010101A
2 Spiderman Peter Parker
3 hero L 03030303C Green Goblin Norman Osborn
4 villano T 05050505E
5 Storm Ororo Munroe
6 hero B 07070707G
7 Wolverine James Howlett
8 hero 09090909I
9 Mystique Raven Darkholme
10 villano H 04040404D
11 IronMan Tony Stark
12 hero f 06060606F
13 Mandarin Zhang Tong
14 villano F

```

Problems • Javadoc • Declaration • Console ×

<terminated> Ejercicio04_A [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (4 oct 2024, 13:16:38 - 13:16:38)

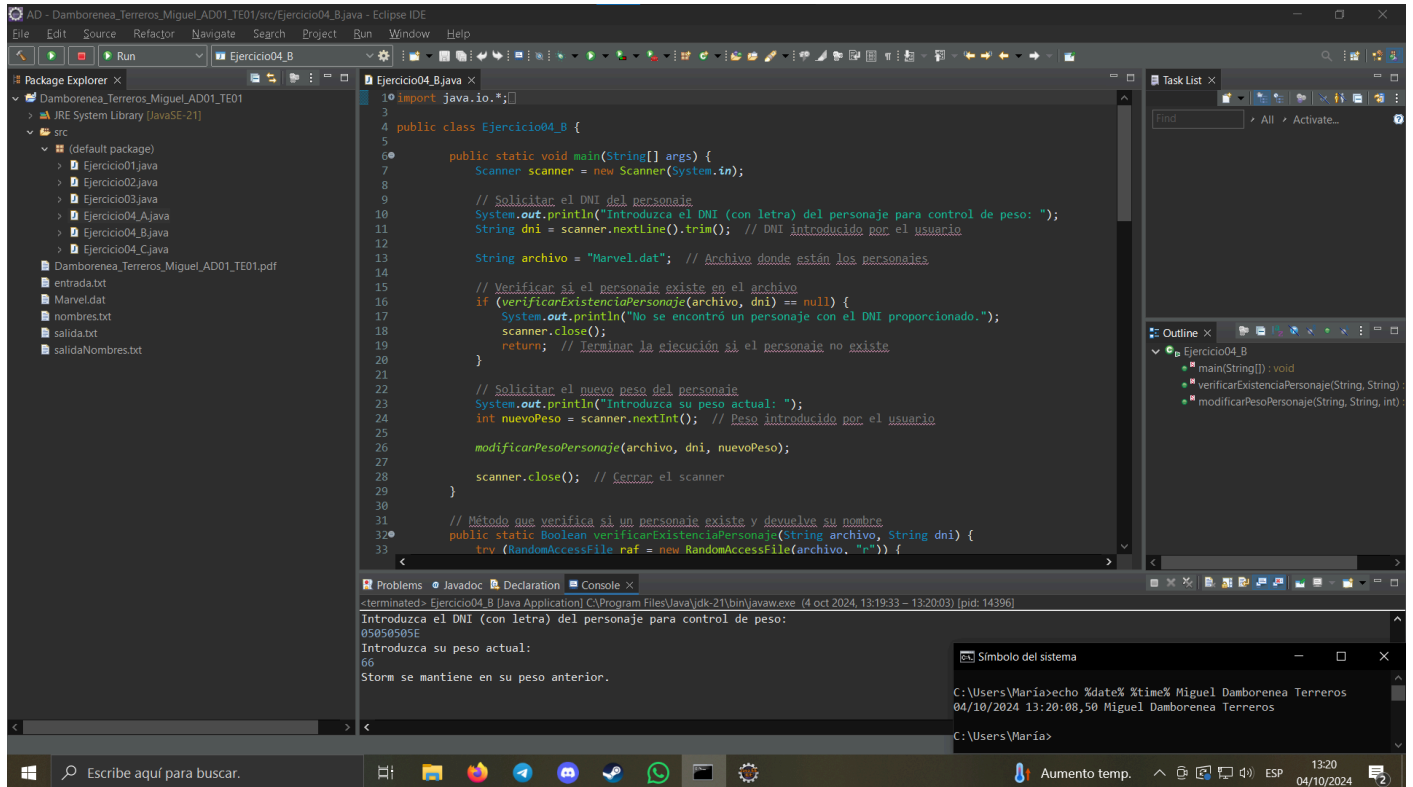
La carga de los personajes ha terminado correctamente.

C:\Users\Wariajecho %date% %time% Miguel Damborenea Terreros
04/10/2024 13:17:29,25 Miguel Damborenea Terreros

C:\Users\Wariajecho



b). El código funciona adecuadamente y está bien comentado. La IA ha sido usada para preguntar dudas del getFilePointer, y sugerencias de mejora.



```

1 import java.io.*;
2
3 public class Ejercicio04_B {
4
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7
8         // Solicitar el DNI del personaje
9         System.out.println("Introduzca el DNI (con letra) del personaje para control de peso: ");
10        String dni = scanner.nextLine().trim(); // DNI introducido por el usuario
11
12        String archivo = "Marvel.dat"; // Archivo donde están los personajes
13
14        // Verificar si el personaje existe en el archivo
15        if (verificarExistenciaPersonaje(archivo, dni) == null) {
16            System.out.println("No se encontró un personaje con el DNI proporcionado.");
17            scanner.close();
18            return; // Terminar la ejecución si el personaje no existe
19        }
20
21        // Solicitar el nuevo peso del personaje
22        System.out.println("Introduzca su peso actual: ");
23        int nuevoPeso = scanner.nextInt(); // Peso introducido por el usuario
24
25        modificarPesoPersonaje(archivo, dni, nuevoPeso);
26
27        scanner.close(); // Cerrar el scanner
28    }
29
30    // Método que verifica si un personaje existe y devuelve su nombre
31    public static Boolean verificarExistenciaPersonaje(String archivo, String dni) {
32        try (RandomAccessFile raf = new RandomAccessFile(archivo, "r")) {
33

```

Console output:

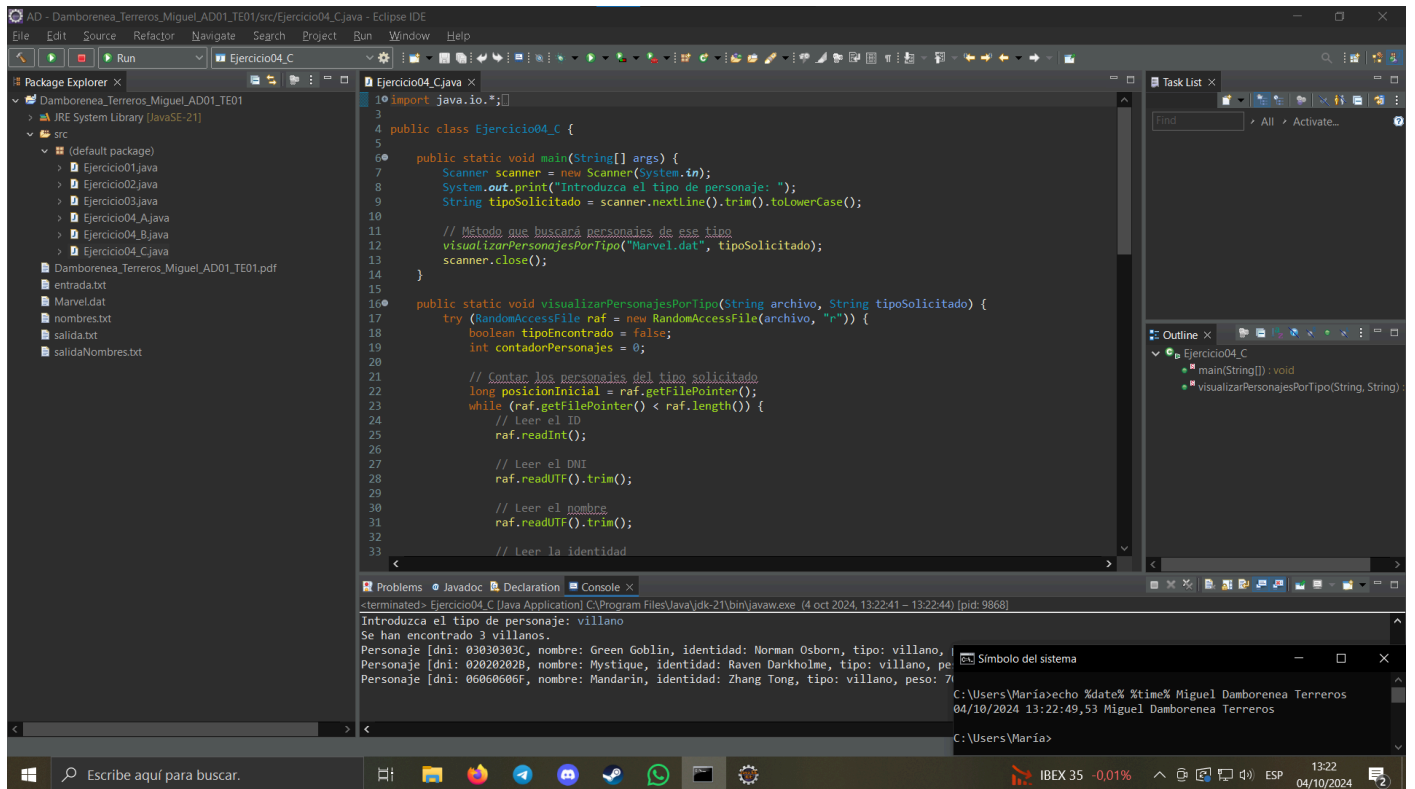
```

<terminated> Ejercicio04_B [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (4 oct 2024, 13:19:33 - 13:20:03) [pid: 14396]
Introduzca el DNI (con letra) del personaje para control de peso:
05050505E
Introduzca su peso actual:
66
Storm se mantiene en su peso anterior.

```

c). El código funciona adecuadamente, cumple todos los requisitos del ejercicio y está bien comentado, aunque quizás hay alguna redundancia en el código. La IA ha sido usada como en los anteriores ejercicios para sugerencias de mejora.





```

1 import java.io.*;
2
3
4 public class Ejercicio04_C {
5
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8         System.out.print("Introduzca el tipo de personaje: ");
9         String tipoSolicitado = scanner.nextLine().trim().toLowerCase();
10
11         // Método que buscará personajes de ese tipo
12         visualizarPersonajesPorTipo("Marvel.dat", tipoSolicitado);
13         scanner.close();
14     }
15
16     public static void visualizarPersonajesPorTipo(String archivo, String tipoSolicitado) {
17         try (RandomAccessFile raf = new RandomAccessFile(archivo, "r")) {
18             boolean tipoEncontrado = false;
19             int contadorPersonajes = 0;
20
21             // Contar los personajes del tipo solicitado
22             long posicionInicial = raf.getFilePointer();
23             while (raf.getFilePointer() < raf.length()) {
24                 // Leer el ID
25                 raf.readInt();
26
27                 // Leer el DNI
28                 raf.readUTF().trim();
29
30                 // Leer el nombre
31                 raf.readUTF().trim();
32
33                 // Leer la identidad

```

Console Output:

```

<terminated> Ejercicio04_C [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (4 oct 2024, 13:22:41 - 13:22:44) [pid: 9868]
Introduzca el tipo de personaje: villano
Se han encontrado 3 villanos.
Personaje [dni: 03030303C, nombre: Green Goblin, identidad: Norman Osborn, tipo: villano, peso: 70]
Personaje [dni: 02020202B, nombre: Mystique, identidad: Raven Darkholme, tipo: villano, peso: 70]
Personaje [dni: 06060606F, nombre: Mandarin, identidad: Zhang Tong, tipo: villano, peso: 70]

```

Uso de la IA: La IA ha sido usada principalmente para resolver bloqueos y aportar sugerencias de mejora al código. En total, el porcentaje aproximado de uso para este trabajo rondaría alrededor del 40%.

2. BIBLIOGRAFÍA

- Apuntes de la asignatura.
- ChatGPT.

