



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Ingeniería en computación

Laboratorios de computación salas A y B

Profesor: _____ Ing. Rene Adrian Davila Perez

Asignatura: _____ Programación Orientada a Objetos

Grupo: _____ 1

No. de práctica: _____ Practica 11

Integrante: _____ 424121462

Semestre: _____ 2025-1

Fecha de entrega: _____ 04 de noviembre de 2024

Observaciones: _____

CALIFICACIÓN: _____

Índice

| | |
|------------------|-------------|
| 1. Introducción | (2) |
| <hr/> | |
| 2. Marco Teórico | (3) |
| <hr/> | |
| 3. Desarrollo | (4-5) |
| <hr/> | |
| 4. Resultados | (6-8) |
| <hr/> | |
| 5. Conclusiones | (8) |
| <hr/> | |
| 6. Referencias | (9) |
| <hr/> | |

Introducción:

Este proyecto se enfoca en la implementación de programas en Java para leer y escribir archivos de texto. Los ejercicios proporcionados permiten desarrollar habilidades en la lectura, almacenamiento y escritura de datos de texto en aplicaciones.

Para esta práctica se plantea lo siguiente:

- **Ejercicio 1:** Crear un programa en Java que lea el contenido de un archivo línea por línea y lo muestre en la consola.
- **Ejercicio 2:** Implementar un programa en Java para leer un archivo línea por línea y almacenar cada línea en una variable para su posterior manipulación.
- **Ejercicio 3:** Desarrollar un programa en Java para almacenar el contenido de un archivo de texto línea por línea en un arreglo, permitiendo su acceso y gestión de forma estructurada.
- **Ejercicio 4:** Crear un programa en Java que permita tanto escribir en un archivo de texto cómo leer su contenido, brindando funcionalidad completa de entrada/salida.
- **Ejercicio Tarea:** Crear un programa en Java que lea tres archivos de texto plano y escriba sus contenidos combinados en un archivo de destino diferente.

El ejercicio de Tarea debe cumplir con los siguientes requisitos:

- Estar encapsulado y definido dentro de un paquete.
- Generar un archivo jar y documentación en javadoc.
- Incluir un reporte que contemple un diagrama de clases.
- Desarrollar la capacidad de lectura y escritura de archivos en Java utilizando clases de I/O.

La correcta manipulación de archivos es una habilidad esencial en la programación, ya que permite gestionar grandes volúmenes de datos y realizar operaciones sobre ellos de forma estructurada y adecuada.

Este proyecto nos permitirá fortalecer estas habilidades a su vez que vemos diferentes maneras para poder acceder a los archivos, expandiendo las resoluciones posibles a los problemas planteados, fomentando buenas prácticas en el manejo de datos y garantizando la estabilidad y funcionalidad del código frente a distintas necesidades de entrada/salida.

Marco Teórico:

El manejo de archivos en Java es una capacidad fundamental en la programación, lo cual es crucial para aplicaciones que dependen de la persistencia de datos, la interacción con el usuario, o el análisis de información.

Java implementa operaciones de entrada y salida de archivos a través de: **FileReader**, **BufferedReader**, **FileWriter**, entre otras. Estas clases permiten abrir, leer y escribir archivos, están diseñadas para gestionar recursos de manera controlada, evitando la sobrecarga de memoria. [1]

Lectura de Archivos: La clase `FileReader` permite abrir un archivo de texto y leerlo carácter por carácter, mientras que `BufferedReader` permite la lectura más eficiente de datos, línea por línea, y es más adecuado para leer grandes volúmenes de texto.

Escritura en Archivos: Para escribir en archivos, Java ofrece las clases `FileWriter` y `BufferedWriter`. `FileWriter` permite abrir o crear un archivo para escribir datos, ya sea sobrescribiendo su contenido o agregando información al final.

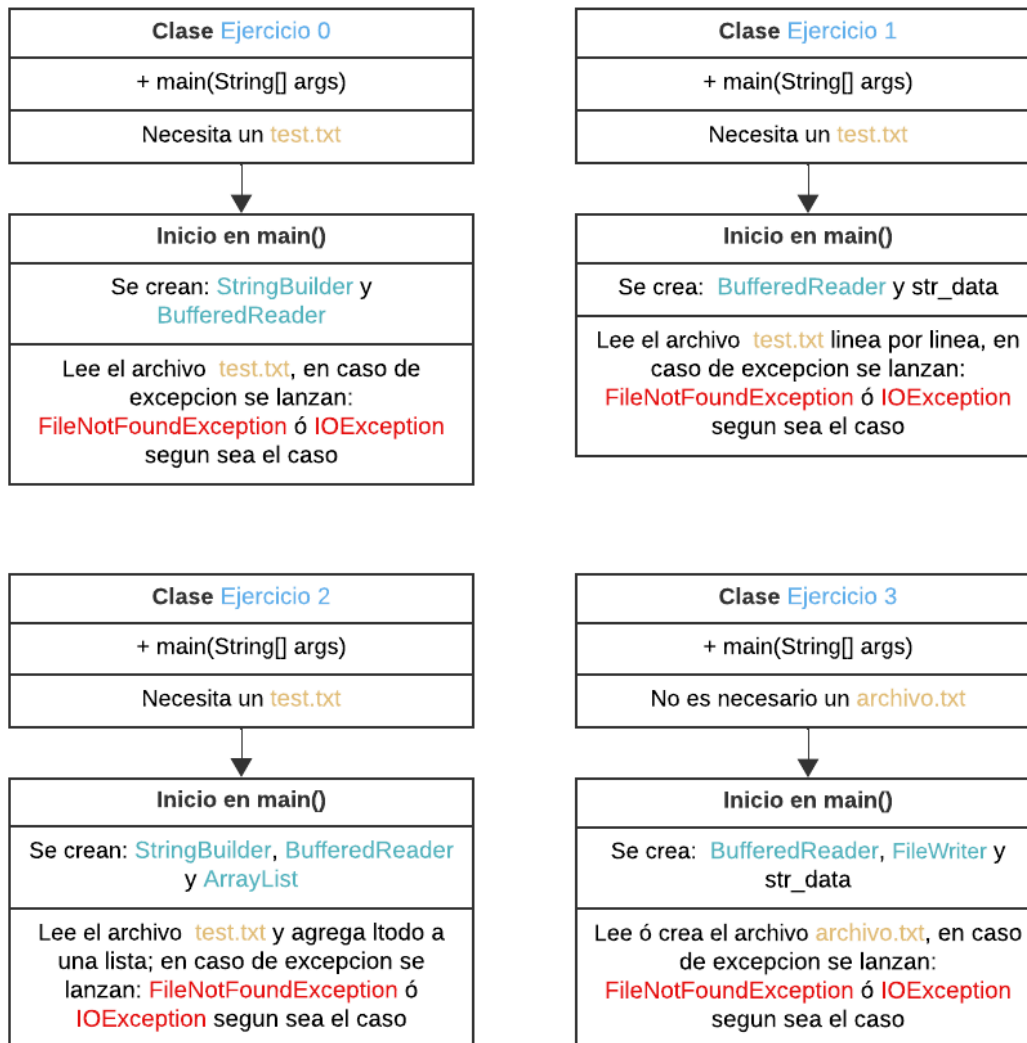
Manejo de Archivos con Arreglos: Al leer un archivo de texto, es común almacenar su contenido en estructuras de datos, como arreglos o listas, para su posterior procesamiento.

Los bloques try-catch permiten capturar y gestionar errores al intentar abrir, leer o escribir en archivos, asegurando así que el programa se mantenga estable. La clase `IOException`, es importante en el manejo de archivos, ya que representa errores comunes de entrada y salida o problemas de acceso. [2]

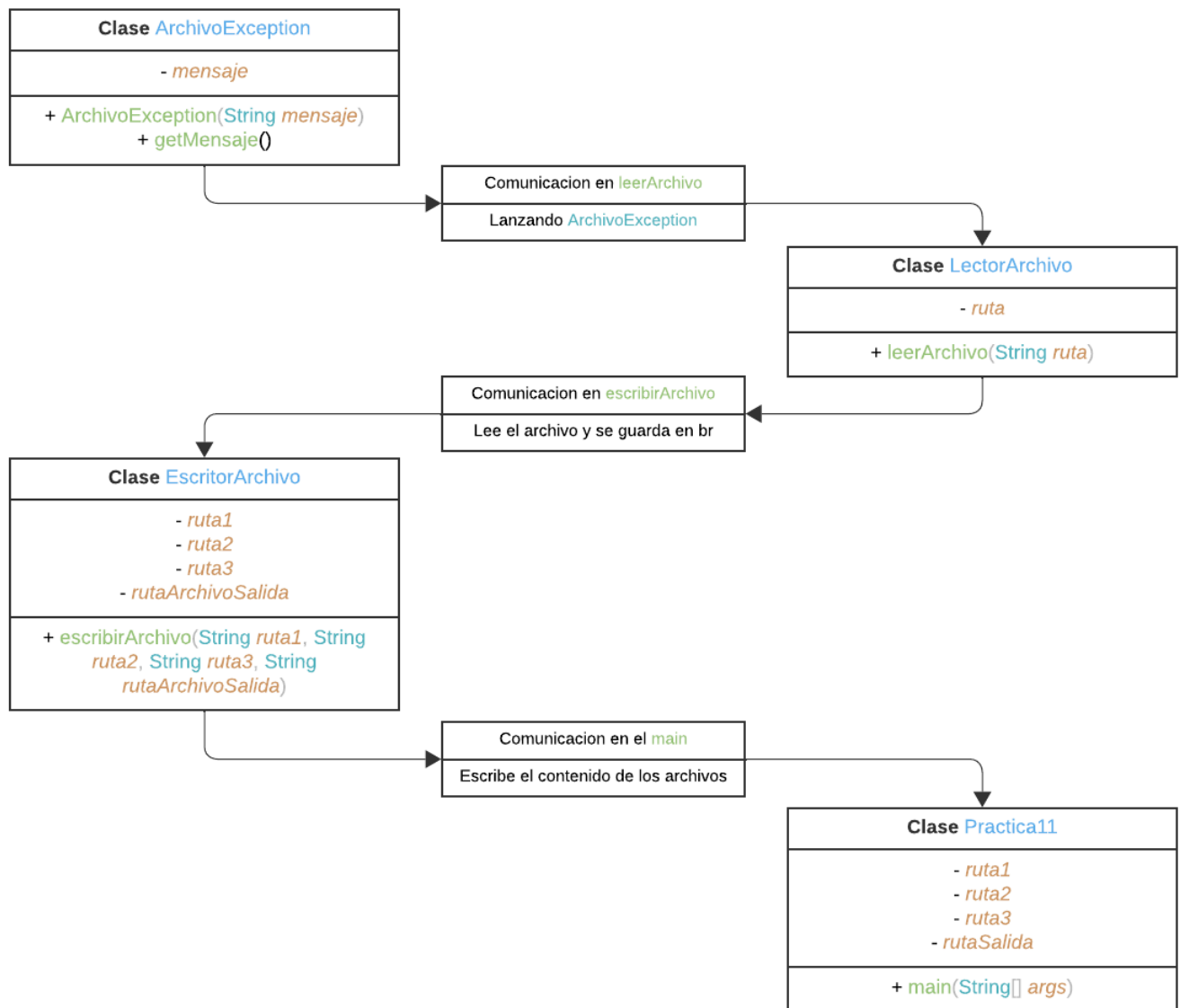
El flujo de datos en Java se refiere al proceso de apertura, manipulación, y cierre de archivos. El correcto cierre de un archivo es crucial para liberar recursos del sistema, evitando problemas de rendimiento o accesos concurrentes a archivos.

La habilidad de leer y escribir en archivos de texto permite que el programa almacene y procese datos, asegurando que la información esté disponible incluso después de que la aplicación se cierre.

Desarrollo:

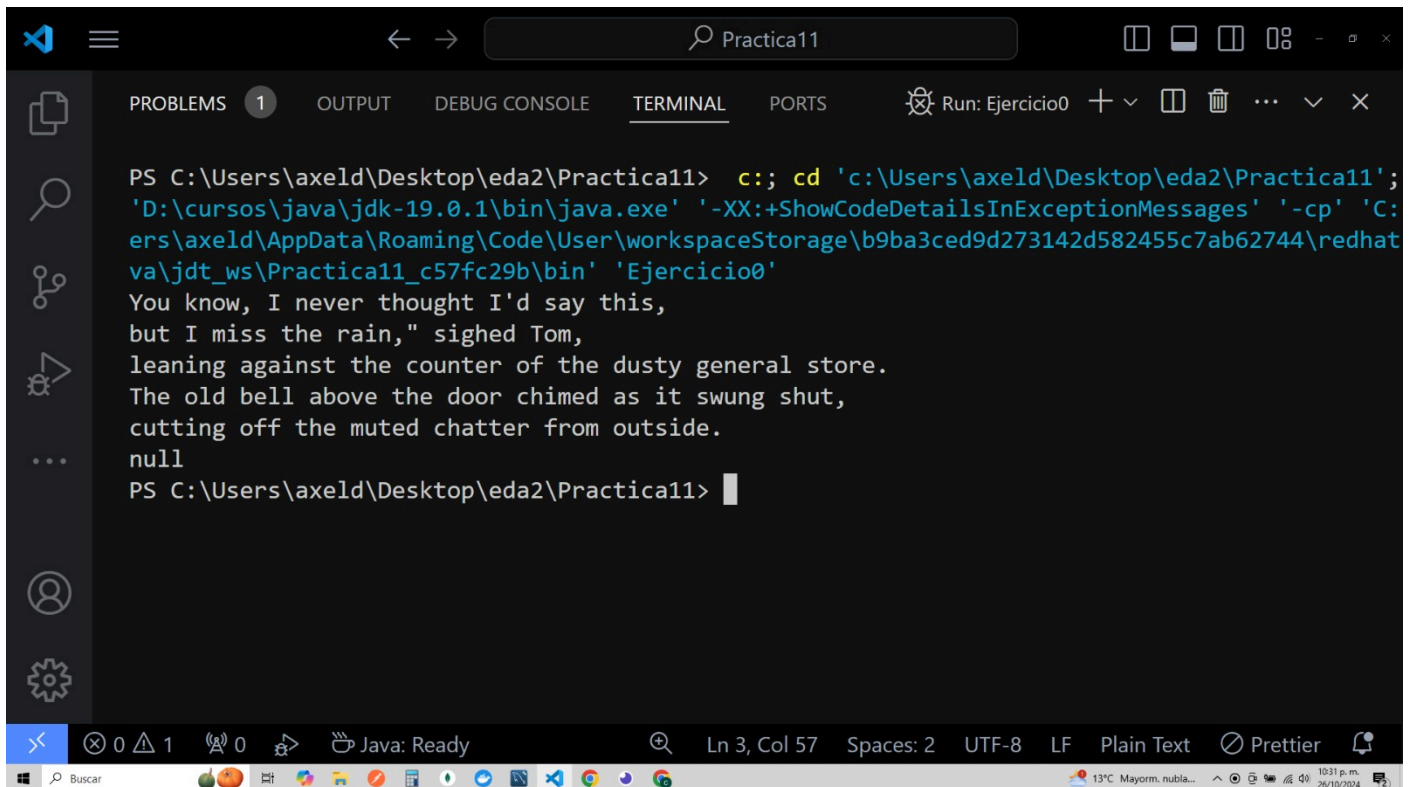


Para los ejercicios 0 al 2, se utilizó un mismo archivo txt para así facilitar el proceso, debido a que si se generan archivos para cada archivo, sería menos practica y ademas el manejar un solo archivo nos facilita el resultado debido a que se espera lo mismo, por lo que al no cambiar el texto, nuestras impresiones deberían ser las mismas, a excepción del ejercicio 2, donde el texto está contenido en una lista.



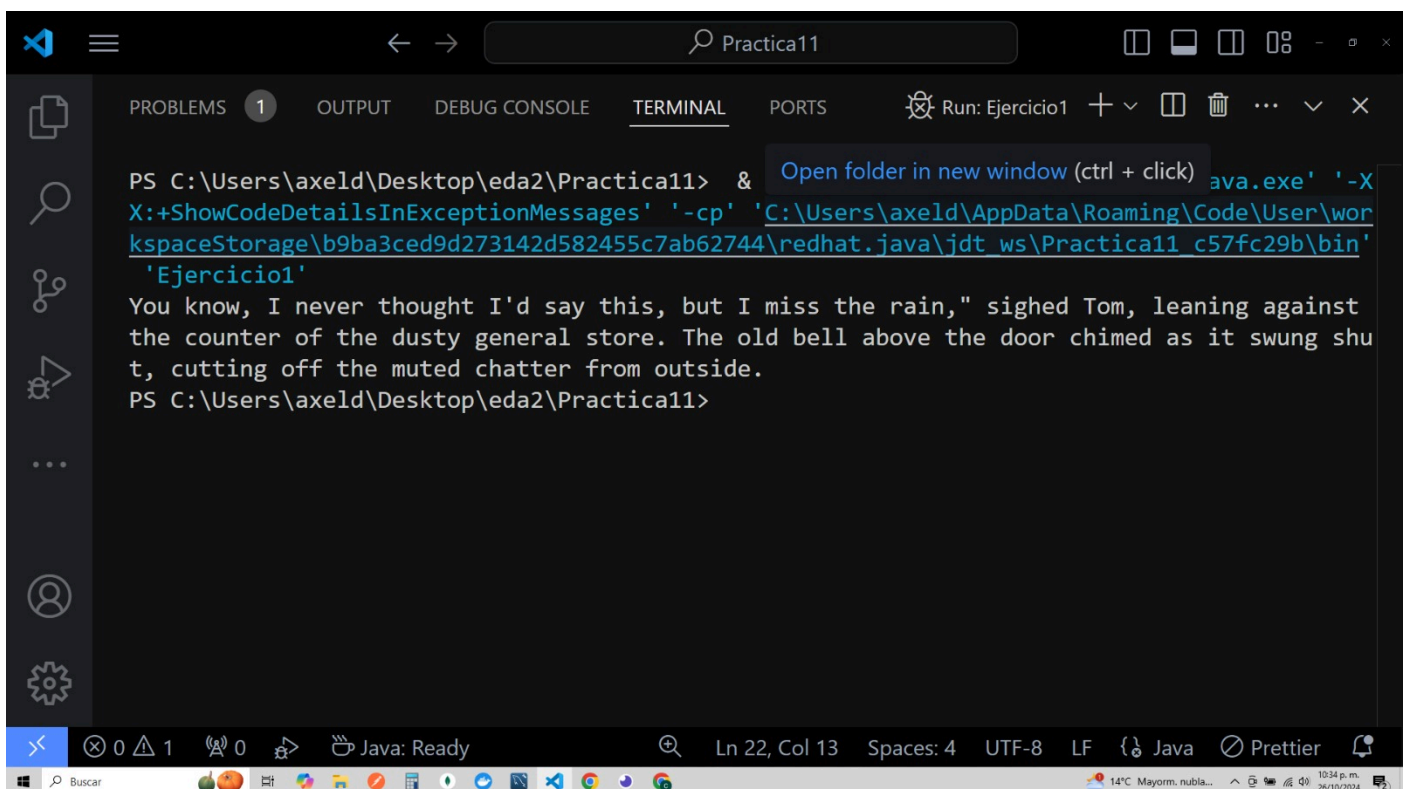
Para la ejecución de este archivo se utilizaron tres archivos diferentes, de los cuales los párrafos se conectaban mediante una historia para así darle continuidad al producto final, se espera que se muestren en pantalla los contenidos de cada uno de los archivos, solo que en uno solo que fue creado con anterioridad.

Resultados:



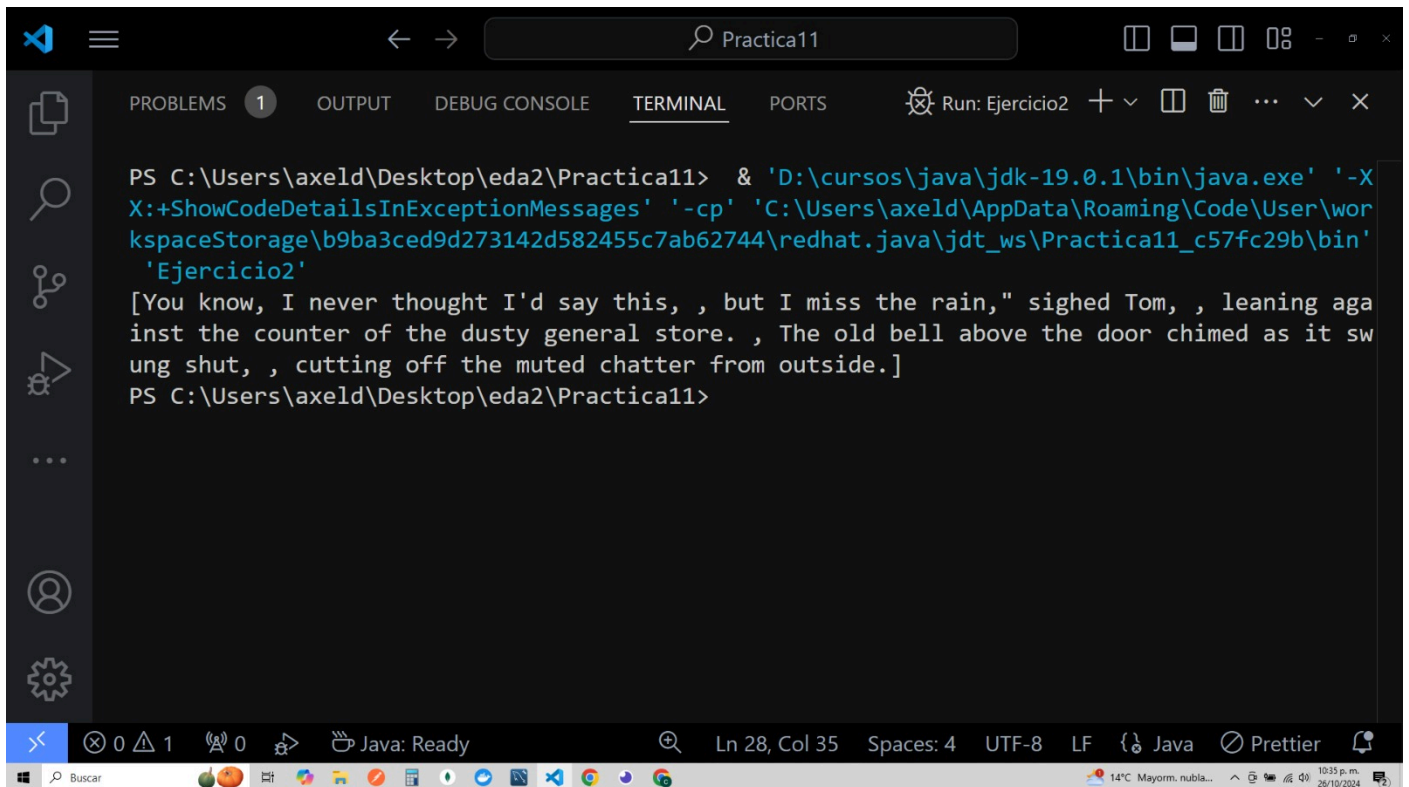
```
PS C:\Users\axeld\Desktop\eda2\Practica11> c:: cd 'c:\Users\axeld\Desktop\eda2\Practica11';  
'D:\cursos\java\jdk-19.0.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:  
ers\axeld\AppData\Roaming\Code\User\workspaceStorage\b9ba3ced9d273142d582455c7ab62744\redhat  
va\jdt_ws\Practica11_c57fc29b\bin' 'Ejercicio0'  
You know, I never thought I'd say this,  
but I miss the rain," sighed Tom,  
leaning against the counter of the dusty general store.  
The old bell above the door chimed as it swung shut,  
cutting off the muted chatter from outside.  
null  
PS C:\Users\axeld\Desktop\eda2\Practica11>
```

Como se muestra en pantalla sale la impresión en pantalla del string por medio de la clase
StringBuilder que contiene el contenido del archivo de texto utilizado.



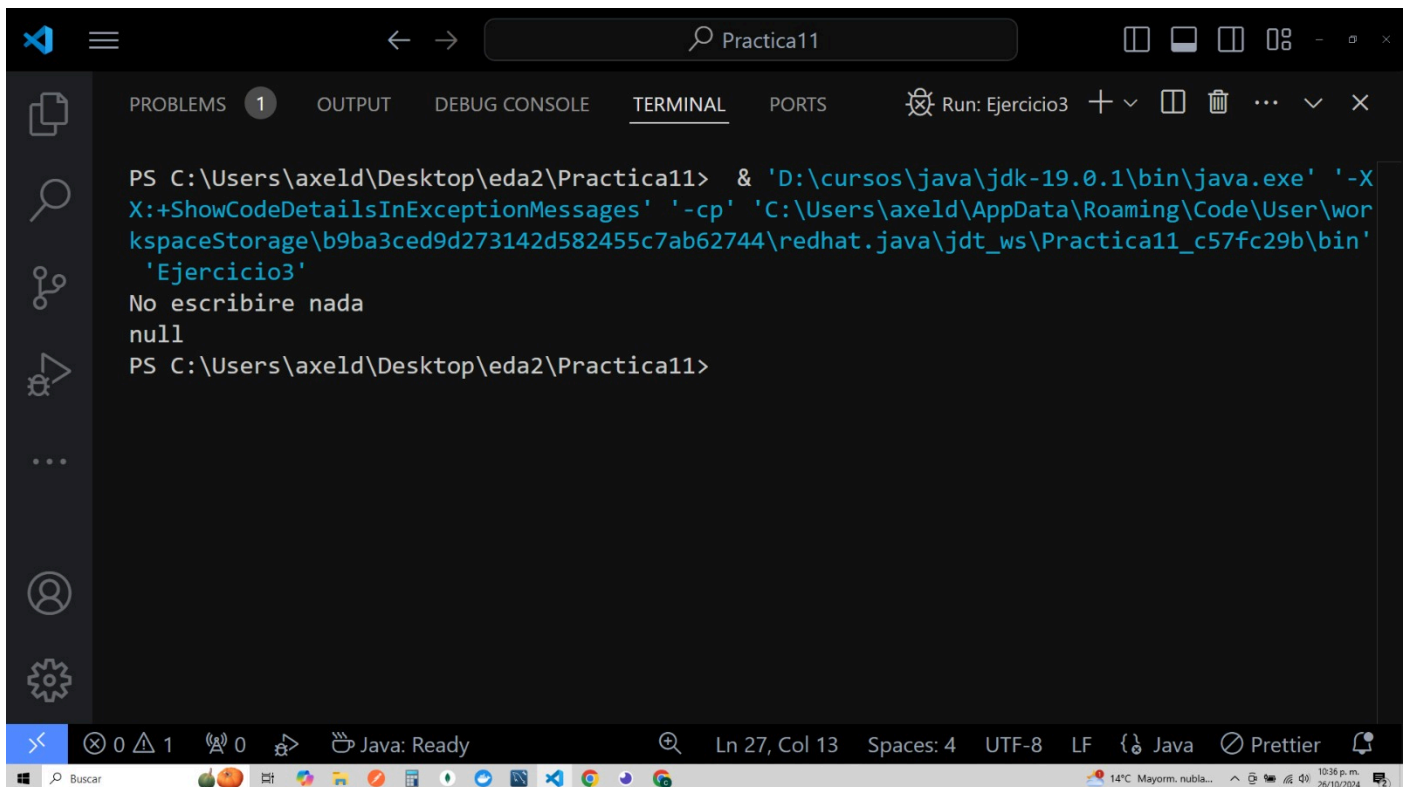
```
PS C:\Users\axeld\Desktop\eda2\Practica11> & Open folder in new window (ctrl + click) java.exe' '-X  
X:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\axeld\AppData\Roaming\Code\User\wor  
kspaceStorage\b9ba3ced9d273142d582455c7ab62744\redhat.java\jdt_ws\Practica11_c57fc29b\bin'  
'Ejercicio1'  
You know, I never thought I'd say this, but I miss the rain," sighed Tom, leaning against  
the counter of the dusty general store. The old bell above the door chimed as it swung shu  
t, cutting off the muted chatter from outside.  
PS C:\Users\axeld\Desktop\eda2\Practica11>
```

En la imagen anterior se puede apreciar en pantalla sale la impresión en pantalla del string que
contiene el contenido del archivo de texto utilizado.



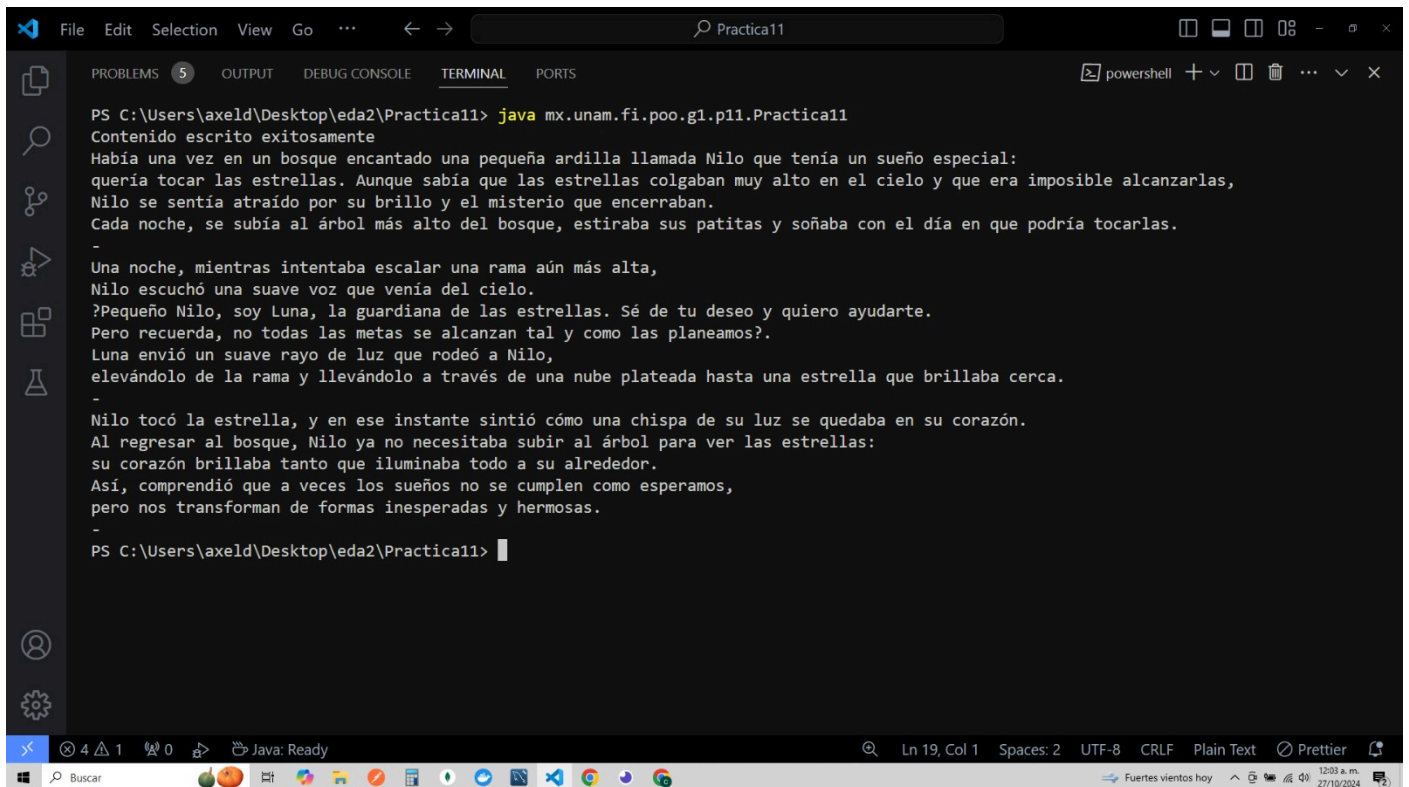
```
PS C:\Users\axeld\Desktop\eda2\Practica11> & 'D:\cursos\java\jdk-19.0.1\bin\java.exe' '-X
X:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\axeld\AppData\Roaming\Code\User\wor
kspaceStorage\b9ba3ced9d273142d582455c7ab62744\redhat.java\jdt_ws\Practica11_c57fc29b\bin'
'Ejercicio2'
[You know, I never thought I'd say this, , but I miss the rain," sighed Tom, , leaning aga
inst the counter of the dusty general store. , The old bell above the door chimed as it sw
ung shut, , cutting off the muted chatter from outside.]
PS C:\Users\axeld\Desktop\eda2\Practica11>
```

Como se muestra en pantalla sale la impresión en pantalla del string por medio de la clase `StringBuilder` y utilizando el `System.lineSeparator`.



```
PS C:\Users\axeld\Desktop\eda2\Practica11> & 'D:\cursos\java\jdk-19.0.1\bin\java.exe' '-X
X:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\axeld\AppData\Roaming\Code\User\wor
kspaceStorage\b9ba3ced9d273142d582455c7ab62744\redhat.java\jdt_ws\Practica11_c57fc29b\bin'
'Ejercicio3'
No escribiré nada
null
PS C:\Users\axeld\Desktop\eda2\Practica11>
```

En la imagen anterior se muestra en pantalla la frase y lo escrito en un nuevo archivo de texto, todo gracias a `FileWriter.write`.



```
PS C:\Users\axeld\Desktop\eda2\Practica11> java mx.unam.fi.poo.g1.p11.Practica11
Contenido escrito exitosamente
Había una vez en un bosque encantado una pequeña ardilla llamada Nilo que tenía un sueño especial:
quería tocar las estrellas. Aunque sabía que las estrellas colgaban muy alto en el cielo y que era imposible alcanzarlas,
Nilo se sentía atraído por su brillo y el misterio que encerraban.
Cada noche, se subía al árbol más alto del bosque, estiraba sus patitas y soñaba con el día en que podría tocarlas.
-
Una noche, mientras intentaba escalar una rama aún más alta,
Nilo escuchó una suave voz que venía del cielo.
?Pequeño Nilo, soy Luna, la guardiana de las estrellas. Sé de tu deseo y quiero ayudarte.
Pero recuerda, no todas las metas se alcanzan tal y como las planeamos?.
Luna envió un suave rayo de luz que rodeó a Nilo,
elevándolo de la rama y llevándolo a través de una nube plateada hasta una estrella que brillaba cerca.
-
Nilo tocó la estrella, y en ese instante sintió cómo una chispa de su luz se quedaba en su corazón.
Al regresar al bosque, Nilo ya no necesitaba subir al árbol para ver las estrellas:
su corazón brillaba tanto que iluminaba todo a su alrededor.
Así, comprendió que a veces los sueños no se cumplen como esperamos,
pero nos transforman de formas inesperadas y hermosas.
-
PS C:\Users\axeld\Desktop\eda2\Practica11>
```

En la imagen anterior se muestra la lectura de un archivo que contiene el contenido de 3 archivos, estos contenidos separados por un “-” para así mostrarlo de manera más clara.

Conclusiones:

La utilización de clases como `FileReader`, `BufferedReader`, `FileWriter`, etc. demuestra cómo la comprensión de las operaciones de entrada y salida es crucial para el desarrollo de aplicaciones eficientes. Además, el uso de estructuras de datos para almacenar información extraída de archivos resalta la importancia de gestionar correctamente los datos de la aplicación correspondiente.

Asimismo, el manejo adecuado de excepciones, como `IOException`, es fundamental para garantizar la estabilidad del programa. La práctica de cerrar correctamente los archivos tras su uso no solo previene problemas de rendimiento, sino que también refleja una buena gestión de recursos disponibles, la cual no causará una sobrecarga en el desarrollo de nuestras aplicaciones, volviéndolas más rápidas, lo cual en el mundo del desarrollo se está bien premiado.

En conclusión, la aplicación de estos conceptos permite resolver problemas reales en la programación, mejorando así la capacidad del programador para crear soluciones más eficientes.

Referencias:

[1] GeeksforGeeks, "File Handling in Java." [En línea]. Disponible: <https://www.geeksforgeeks.org/file-handling-java-using-filewriter-filereader/>. [Accedido: Nov. 01, 2024].

[2] ChatGPT. [En línea]. Disponible: <https://chatgpt.com/>. [Accedido: Nov. 01, 2024]. Pregunta:
“como funciona y para que sirve FileReader, BufferedReader, FileWriter en Java”