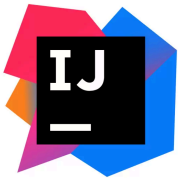

# Ejemplo adaptado para IntelliJ IDEA que muestra cómo leer y escribir en ficheros de texto plano

Este ejemplo incluye dos métodos:

1. `writeTextFile()`: Escribe tres líneas de texto en un archivo llamado "example.txt".
2. `readTextFile()`: Lee el contenido del archivo "**example.txt**" y lo muestra en la consola.

Algunas mejoras y consideraciones en este código:

- Se utiliza try-with-resources para manejar automáticamente el cierre de los recursos (FileWriter, PrintWriter y BufferedReader).
- Se maneja las excepciones de manera más específica, mostrando mensajes de error descriptivos.
- El código está estructurado en métodos separados para mejorar la legibilidad y mantenibilidad.


Este ejemplo creará un archivo "**example.txt**" en el directorio raíz de tu proyecto, escribirá en él y luego leerá su contenido

```java
import java.io.*;

public class FileAccessExample {

    public static void main(String[] args) {
        writeTextFile();
        readTextFile();
    }

    public static void writeTextFile() {
        String fileName = "example.txt";

        try (FileWriter fileWriter = new FileWriter(fileName);
             PrintWriter printWriter = new PrintWriter(fileWriter)) {

            printWriter.println("Esta es la primera línea del fichero.");
            printWriter.println("Esta es la segunda línea del fichero.");
            printWriter.println("Esta es la tercera línea del fichero.");

            System.out.println("Fichero escrito correctamente.");
        } catch (IOException e) {
            System.out.println("Error al escribir en el fichero: " +
e.getMessage());
        }
    }

    public static void readTextFile() {
        String fileName = "example.txt";

        try (BufferedReader bufferedReader = new BufferedReader(new
FileReader(fileName))) {
            String line;
            System.out.println("Contenido del fichero:");
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("Error al leer el fichero: " +
e.getMessage());
        }
    }
}
```

Para ejecutar este ejemplo en IntelliJ IDEA:

1. **Crea un nuevo proyecto Java.**
2. **Crea una nueva clase llamada `FileAccessExample`.**
3. **Copia y pega el código anterior en la clase.**
4. **Ejecuta el método `main`.**

# Ejemplo de Gestor de Notas simple

Programa en Java que permite a los usuarios gestionar notas personales. El programa realiza las siguientes operaciones:

1. **Añadir una nueva nota al fichero "notas.txt".**
2. **Leer y mostrar todas las notas existentes.**
3. **Buscar notas que contengan una palabra clave.**

Cada nota debe tener un título y un contenido, separados por ":".

Este programa crea un gestor de notas simple que utiliza un archivo de texto para almacenar las notas. Permite al usuario añadir nuevas notas, ver todas las notas existentes y buscar notas que contengan una palabra clave específica.

El programa creará un archivo "**notas.txt**" en el directorio raíz de tu proyecto para almacenar las notas. Puedes interactuar con el programa a través de la consola, eligiendo las diferentes opciones del menú.



```java
import java.io.*;
import java.util.Scanner;

public class GestorNotas {
    private static final String ARCHIVO_NOTAS = "notas.txt";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int opcion;

        do {
            System.out.println("\n--- Gestor de Notas ---");
            System.out.println("1. Añadir nota");
            System.out.println("2. Ver todas las notas");
            System.out.println("3. Buscar notas");
            System.out.println("0. Salir");
            System.out.print("Elige una opción: ");
            opcion = scanner.nextInt();
            scanner.nextLine(); // Consumir el salto de línea

            switch (opcion) {
                case 1:
                    anadirNota(scanner);
                    break;
                case 2:
                    verNotas();
                    break;
                case 3:
                    buscarNotas(scanner);
```

```
                    break;
                case 0:
                    System.out.println("¡Hasta luego!");
                    break;
                default:
                    System.out.println("Opción no válida.");
            }
        } while (opcion != 0);

        scanner.close();
    }

    private static void anadirNota(Scanner scanner) {
        System.out.print("Introduce el título de la nota: ");
        String titulo = scanner.nextLine();
        System.out.print("Introduce el contenido de la nota: ");
        String contenido = scanner.nextLine();

        try (PrintWriter writer = new PrintWriter(new
FileWriter(ARCHIVO_NOTAS, true))) {
            writer.println(titulo + ":" + contenido);
            System.out.println("Nota añadida correctamente.");
        } catch (IOException e) {
            System.out.println("Error al añadir la nota: " +
e.getMessage());
        }
    }

    private static void verNotas() {
        try (BufferedReader reader = new BufferedReader(new
FileReader(ARCHIVO_NOTAS))) {
            String linea;
            System.out.println("\n--- Todas las Notas ---");
            while ((linea = reader.readLine()) != null) {
                String[] partes = linea.split(":", 2);
                System.out.println("Título: " + partes[0]);
                System.out.println("Contenido: " + partes[1]);
                System.out.println("-------------------------");
            }
        } catch (IOException e) {
            System.out.println("Error al leer las notas: " +
e.getMessage());
        }
    }

    private static void buscarNotas(Scanner scanner) {
        System.out.print("Introduce la palabra clave a buscar: ");
        String palabraClave = scanner.nextLine().toLowerCase();

        try (BufferedReader reader = new BufferedReader(new
FileReader(ARCHIVO_NOTAS))) {
            String linea;
            boolean encontrada = false;
```

```
            System.out.println("\n--- Resultados de la búsqueda ---");
            while ((linea = reader.readLine()) != null) {
                if (linea.toLowerCase().contains(palabraClave)) {
                    String[] partes = linea.split(":", 2);
                    System.out.println("Título: " + partes[0]);
                    System.out.println("Contenido: " + partes[1]);
                    System.out.println("-------------------------");
                    encontrada = true;
                }
            }
            if (!encontrada) {
                System.out.println("No se encontraron notas con esa palabra
clave.");
            }
        } catch (IOException e) {
            System.out.println("Error al buscar notas: " + e.getMessage());
        }
    }
}
```



Para usar este programa en IntelliJ IDEA:

1. **Crea un nuevo proyecto Java.**
2. **Crea una nueva clase llamada `GestorNotas`.**
3. **Copia y pega el código anterior en la clase.**
4. **Ejecuta el método `main`.**

## Ejemplo de un programa en Java que permita a los usuarios llevar un registro de sus gastos personales.

El programa debe realizar las siguientes operaciones:

1. **Añadir un nuevo gasto al fichero "gastos.txt".**
2. **Mostrar todos los gastos registrados.**
3. **Calcular y mostrar el total de gastos.**
4. **Mostrar los gastos de una categoría específica.**

Cada gasto debe tener una fecha, una categoría, una descripción y una cantidad, separados por comas.


Este programa crea un registro de gastos personales que utiliza un archivo de texto para almacenar la información. Permite al usuario añadir nuevos gastos, ver todos los gastos registrados, calcular el total de gastos y ver los gastos de una categoría específica.

```
import java.io.*;
import java.util.Scanner;

public class RegistroGastos {
    private static final String ARCHIVO_GASTOS = "gastos.txt";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int opcion;

        do {
            System.out.println("\n--- Registro de Gastos Personales ---");
            System.out.println("1. Añadir gasto");
            System.out.println("2. Ver todos los gastos");
```

```
            System.out.println("3. Calcular total de gastos");
            System.out.println("4. Ver gastos por categoría");
            System.out.println("0. Salir");
            System.out.print("Elige una opción: ");
            opcion = scanner.nextInt();
            scanner.nextLine(); // Consumir el salto de línea

            switch (opcion) {
                case 1:
                    anadirGasto(scanner);
                    break;
                case 2:
                    verGastos();
                    break;
                case 3:
                    calcularTotalGastos();
                    break;
                case 4:
                    verGastosPorCategoria(scanner);
                    break;
                case 0:
                    System.out.println("¡Hasta luego!");
                    break;
                default:
                    System.out.println("Opción no válida.");
            }
        } while (opcion != 0);

        scanner.close();
    }

    private static void anadirGasto(Scanner scanner) {
        System.out.print("Introduce la fecha (DD/MM/YYYY): ");
        String fecha = scanner.nextLine();
        System.out.print("Introduce la categoría: ");
        String categoria = scanner.nextLine();
        System.out.print("Introduce la descripción: ");
        String descripcion = scanner.nextLine();
        System.out.print("Introduce la cantidad: ");
        double cantidad = scanner.nextDouble();

        try (PrintWriter writer = new PrintWriter(new
FileWriter(ARCHIVO_GASTOS, true))) {
            writer.println(fecha + "," + categoria + "," + descripcion + ","
+ cantidad);
            System.out.println("Gasto registrado correctamente.");
        } catch (IOException e) {
            System.out.println("Error al registrar el gasto: " +
e.getMessage());
        }
    }

    private static void verGastos() {
```

```
        try (BufferedReader reader = new BufferedReader(new
FileReader(ARCHIVO_GASTOS))) {
            String linea;
            System.out.println("\n--- Todos los Gastos ---");
            while ((linea = reader.readLine()) != null) {
                String[] partes = linea.split(",");
                System.out.println("Fecha: " + partes[0] + ", Categoría: " +
partes[1] +
                                   ", Descripción: " + partes[2] + ",
cantidad: $" + partes[3]);
            }
        } catch (IOException e) {
            System.out.println("Error al leer los gastos: " +
e.getMessage());
        }
    }

    private static void calcularTotalGastos() {
        double total = 0;
        try (BufferedReader reader = new BufferedReader(new
FileReader(ARCHIVO_GASTOS))) {
            String linea;
            while ((linea = reader.readLine()) != null) {
                String[] partes = linea.split(",");
                total += Double.parseDouble(partes[3]);
            }
            System.out.println("Total de gastos: $" + total);
        } catch (IOException e) {
            System.out.println("Error al calcular el total de gastos: " +
e.getMessage());
        }
    }

    private static void verGastosPorCategoria(Scanner scanner) {
        System.out.print("Introduce la categoría a buscar: ");
        String categoriaBuscada = scanner.nextLine().toLowerCase();

        try (BufferedReader reader = new BufferedReader(new
FileReader(ARCHIVO_GASTOS))) {
            String linea;
            boolean encontrado = false;
            System.out.println("\n--- Gastos de la categoría '" +
categoriaBuscada + "' ---");
            while ((linea = reader.readLine()) != null) {
                String[] partes = linea.split(",");
                if (partes[1].toLowerCase().equals(categoriaBuscada)) {
                    System.out.println("Fecha: " + partes[0] + ",
Descripción: " + partes[2] + ", cantidad: $" + partes[3]);
                    encontrado = true;
                }
            }
            if (!encontrado) {
```

```
                System.out.println("No se encontraron gastos en esta
categoría.");
            }
        } catch (IOException e) {
            System.out.println("Error al buscar gastos por categoría: " +
e.getMessage());
        }
    }
}
```

El programa creará un archivo "**gastos.txt**" en el directorio raíz de tu proyecto para almacenar los gastos. Puedes interactuar con el programa a través de la consola, eligiendo las diferentes opciones del menú.

Este ejercicio refuerza los conceptos de lectura y escritura de archivos, manejo de excepciones, y procesamiento de datos en Java, a la vez que proporciona una aplicación práctica y útil para el usuario.

**Para usar este programa en IntelliJ IDEA:**

1. **Crea un nuevo proyecto Java.**
2. **Crea una nueva clase llamada `RegistroGastos`.**
3. **Copia y pega el código anterior en la clase.**
4. **Ejecuta el método `main`.**


# TAREA Ficheros

# Propuesta Mejorada de Gestor ficheros de Gastos



## Objetivo

Desarrollar un programa en Java que permita a los usuarios gestionar sus gastos personales de manera eficiente y detallada.

## Descripción

Debes crear una aplicación de consola que permita a los usuarios realizar un seguimiento de sus gastos diarios. El programa debe ofrecer varias funcionalidades para agregar, visualizar, editar y analizar los gastos.

## Requisitos Básicos

1. Añadir nuevos gastos con fecha, categoría, descripción y cantidad.
2. Mostrar todos los gastos registrados.
3. Calcular y mostrar el total de gastos.
4. Mostrar los gastos de una categoría específica.

## Requisitos Avanzados

5. Editar un gasto existente.
6. Eliminar un gasto.
7. Buscar gastos por rango de fechas.
8. Exportar todos los gastos a un archivo CSV.
9. Mostrar estadísticas básicas (total, promedio, máximo, mínimo, gastos por categoría).

## Estructura de Datos

Cada gasto debe contener:

- Fecha (utilizar `java.time.LocalDate`)
- Categoría (String)
- Descripción (String)
- Cantidad (double)

## Funcionalidades Detalladas

1. Añadir Gasto: Solicitar al usuario los detalles del gasto y agregarlo a la lista.
2. Mostrar Gastos: Presentar todos los gastos en un formato tabular.
3. Total de Gastos: Calcular y mostrar la suma de todos los gastos.
4. Gastos por Categoría: Filtrar y mostrar gastos de una categoría específica.
5. Editar Gasto: Permitir al usuario modificar los detalles de un gasto existente.
6. Eliminar Gasto: Remover un gasto específico de la lista.
7. Búsqueda por Fecha: Mostrar gastos dentro de un rango de fechas especificado.
8. Exportar a CSV: Generar un archivo CSV con todos los gastos.
9. Estadísticas: Calcular y mostrar estadísticas básicas sobre los gastos.

## Requisitos Técnicos

- Utilizar `ArrayList` para almacenar los gastos.
- Implementar manejo de excepciones para entradas inválidas.
- Usar Streams de Java para operaciones de filtrado y cálculos.
- Emplear `try-with-resources` para el manejo de archivos.

### Interfaz de Usuario

Crear un menú de consola que permita al usuario seleccionar las diferentes opciones del programa.

## Persistencia de Datos

- Los gastos deben guardarse en un archivo de texto ("gastos.txt").
- El programa debe cargar los gastos existentes al iniciar y guardar los cambios al salir.