Universidad Autonoma de Nuevo León.
Facultad de Ciencias Físico Matemáticas.

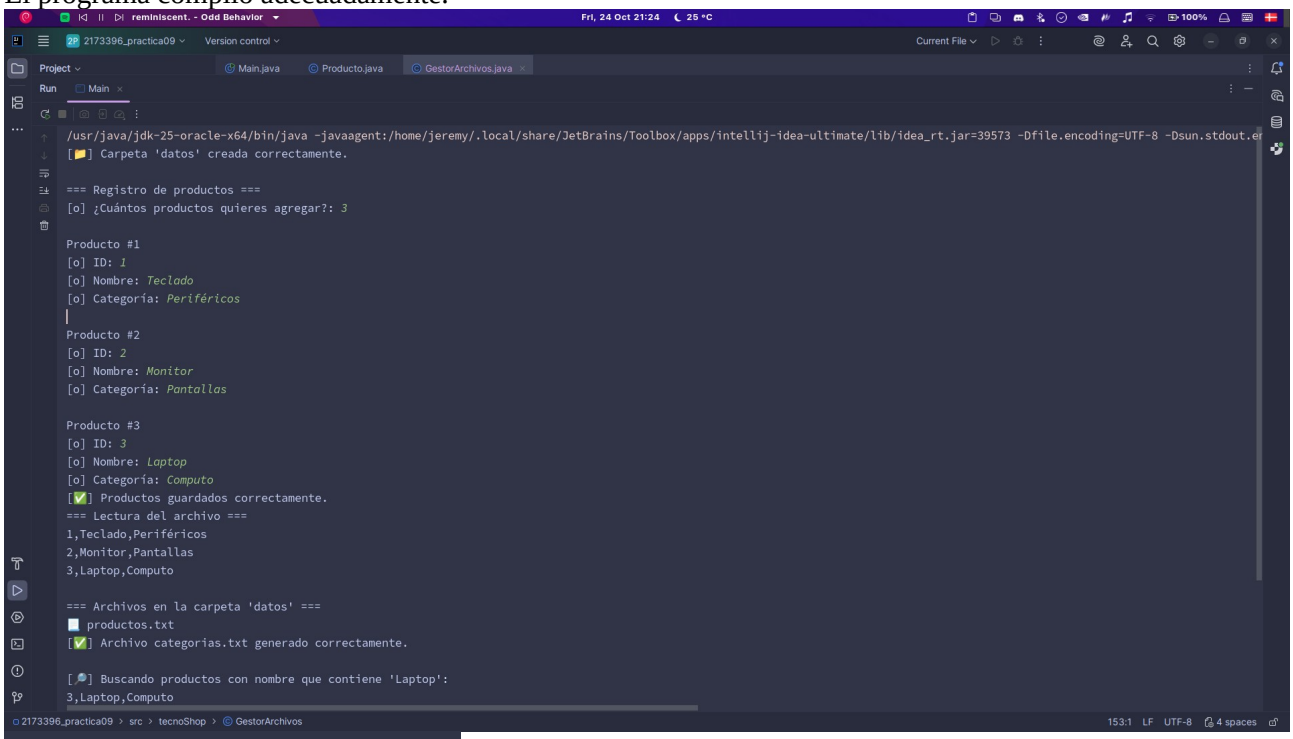
Práctica #09 – Entrada/Salida en Java.

Laboratorio de Programación Orientada a Objetos.

Alumno: Jeremy Uriel Rossell Segura
Matrícula: 2173396
Materia: Laboratorio de Programación Orientada a Objetos.
Docente: Jorge Alberto Islas Pineda

Compilación

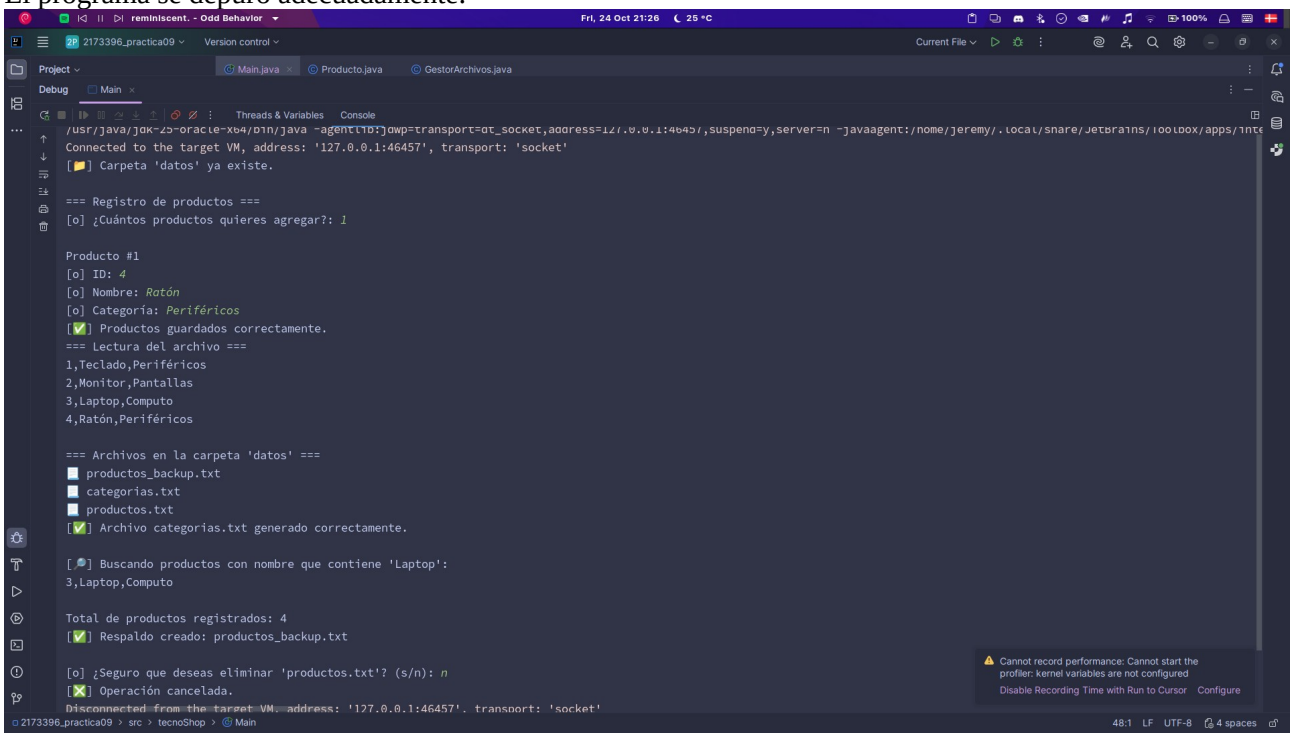
El programa compiló adecuadamente.




```
[🔎] Buscando productos con nombre que contiene 'Laptop':
3,Laptop,Computo

Total de productos registrados: 3
[✅] Respaldo creado: productos_backup.txt

[o] ¿Seguro que deseas eliminar 'productos.txt'? (s/n): n
[❎] Operación cancelada.

Process finished with exit code 0
```

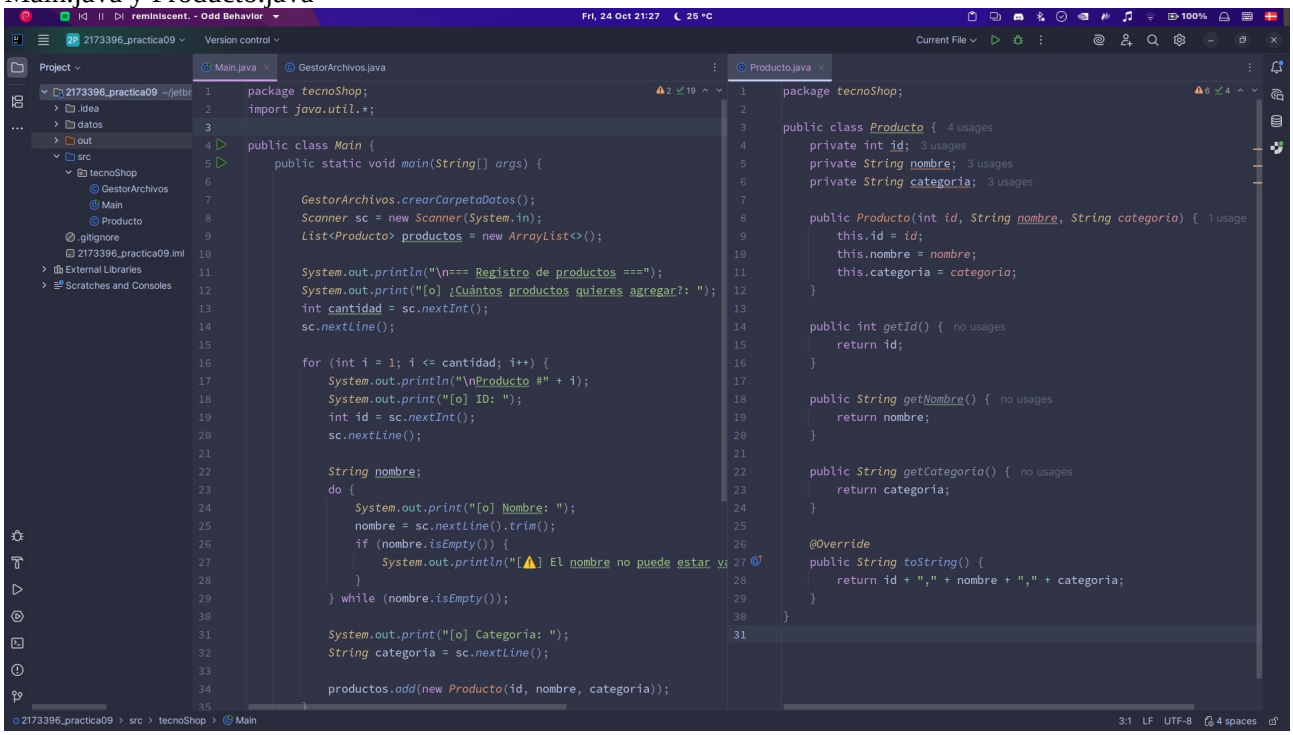
Depuración

El programa se depuró adecuadamente.



Código

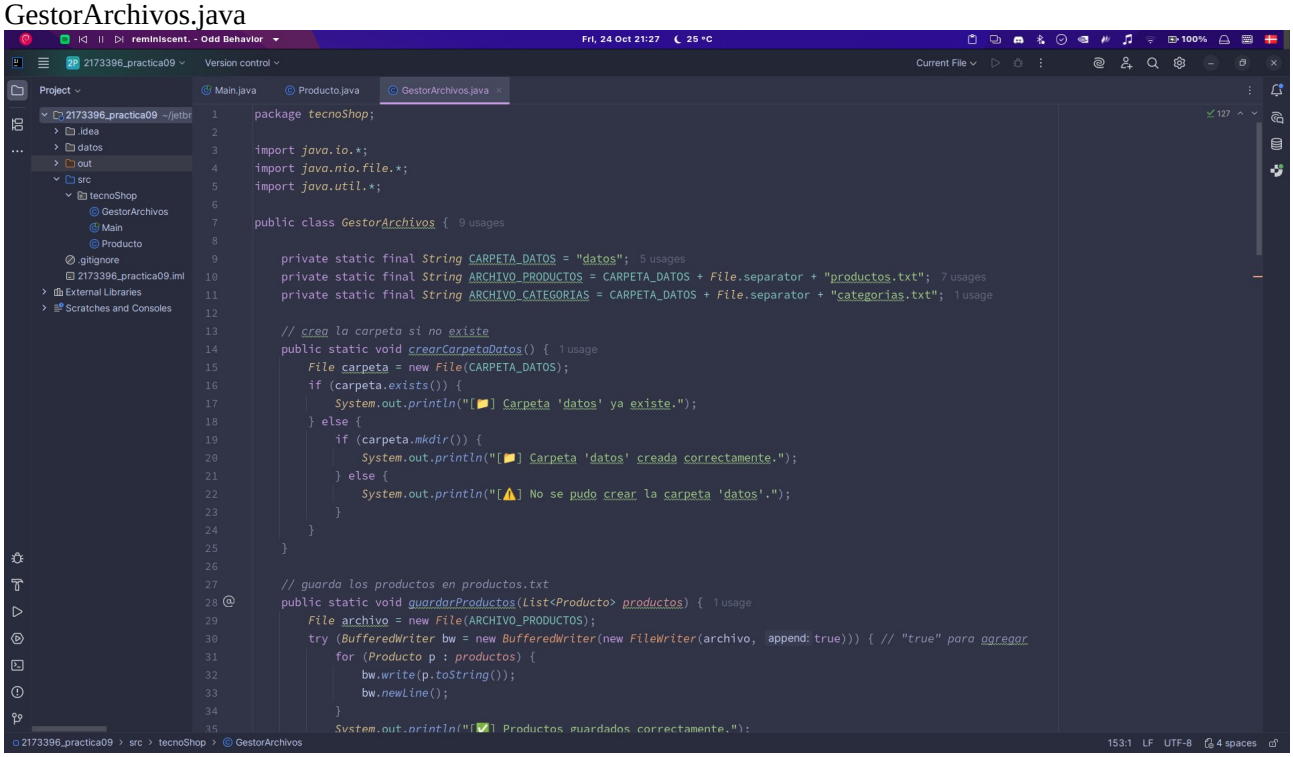
Main.java y Producto.java



```java
package tecnoShop;
import java.util.*;

public class Main {
    public static void main(String[] args) {

        GestorArchivos.crearCarpetaDatos();
        Scanner sc = new Scanner(System.in);
        List<Producto> productos = new ArrayList<>();

        System.out.println("\n=== Registro de productos ===");
        System.out.print("[o] ¿Cuántos productos quieres agregar?: ");
        int cantidad = sc.nextInt();
        sc.nextLine();

        for (int i = 1; i <= cantidad; i++) {
            System.out.println("\nProducto #" + i);
            System.out.print("[o] ID: ");
            int id = sc.nextInt();
            sc.nextLine();

            String nombre;
            do {
                System.out.print("[o] Nombre: ");
                nombre = sc.nextLine().trim();
                if (nombre.isEmpty()) {
                    System.out.println("[⚠] El nombre no puede estar va
                }
            } while (nombre.isEmpty());

            System.out.print("[o] Categoría: ");
            String categoria = sc.nextLine();

            productos.add(new Producto(id, nombre, categoria));
```

```java
package tecnoShop;

public class Producto {
    private int id;
    private String nombre;
    private String categoria;

    public Producto(int id, String nombre, String categoria) {
        this.id = id;
        this.nombre = nombre;
        this.categoria = categoria;
    }

    public int getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

    public String getCategoria() {
        return categoria;
    }

    @Override
    public String toString() {
        return id + "," + nombre + "," + categoria;
    }
}
```

GestorArchivos.java



```java
package tecnoShop;

import java.io.*;
import java.nio.file.*;
import java.util.*;

public class GestorArchivos {

    private static final String CARPETA_DATOS = "datos";
    private static final String ARCHIVO_PRODUCTOS = CARPETA_DATOS + File.separator + "productos.txt";
    private static final String ARCHIVO_CATEGORIAS = CARPETA_DATOS + File.separator + "categorias.txt";

    // crea la carpeta si no existe
    public static void crearCarpetaDatos() {
        File carpeta = new File(CARPETA_DATOS);
        if (carpeta.exists()) {
            System.out.println("[📁] Carpeta 'datos' ya existe.");
        } else {
            if (carpeta.mkdir()) {
                System.out.println("[📁] Carpeta 'datos' creada correctamente.");
            } else {
                System.out.println("[⚠] No se pudo crear la carpeta 'datos'.");
            }
        }
    }

    // guarda los productos en productos.txt
    public static void guardarProductos(List<Producto> productos) {
        File archivo = new File(ARCHIVO_PRODUCTOS);
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(archivo, append: true))) { // "true" para agregar
            for (Producto p : productos) {
                bw.write(p.toString());
                bw.newLine();
            }
            System.out.println("[✅] Productos guardados correctamente.");
```

```java
public class GestorArchivos {  9 usages
    public static void guardarProductos(List<Producto> productos) {  1 usage
            }
            System.out.println("[✅] Productos guardados correctamente.");
        } catch (IOException e) {
            System.out.println("[❌] Error al guardar productos: " + e.getMessage());
        }
    }

    // lee e imprime todo el contenido del archivo
    public static void leerProductos() {  1 usage
        System.out.println("=== Lectura del archivo ===");
        try (BufferedReader br = new BufferedReader(new FileReader(ARCHIVO_PRODUCTOS))) {
            String linea;
            while ((linea = br.readLine()) != null) {
                System.out.println(linea);
            }
        } catch (IOException e) {
            System.out.println("[❌] Error al leer archivo: " + e.getMessage());
        }
    }

    // muestra todos los archivos dentro de la carpeta datos
    public static void listarArchivos() {  1 usage
        System.out.println("\n=== Archivos en la carpeta 'datos' ===");
        File carpeta = new File(CARPETA_DATOS);
        File[] archivos = carpeta.listFiles();
        if (archivos != null) {
            for (File f : archivos) {
                System.out.println("📄 " + f.getName());
            }
        } else {
            System.out.println("[!] No hay archivos en la carpeta.");
        }
    }
// busca productos por nombre dentro del archivo
public static void buscarProductoPorNombre(String nombre) {  1 usage
    System.out.println("\n[🔎] Buscando productos con nombre que contiene '" + nombre + "':");
    try (BufferedReader br = new BufferedReader(new FileReader(ARCHIVO_PRODUCTOS))) {
        String linea;
        boolean encontrado = false;
        while ((linea = br.readLine()) != null) {
            if (linea.toLowerCase().contains(nombre.toLowerCase())) {
                System.out.println(linea);
                encontrado = true;
            }
        }
        if (!encontrado) System.out.println("[!] No se encontró ningún producto con ese nombre.");
    } catch (IOException e) {
        System.out.println("[❌] Error al buscar producto: " + e.getMessage());
    }
}

// crea un archivo categorias.txt con categorías únicas
public static void guardarCategoriasUnicas() {  1 usage
    HashSet<String> categorias = new HashSet<>();

    try (BufferedReader br = new BufferedReader(new FileReader(ARCHIVO_PRODUCTOS))) {
        String linea;
        while ((linea = br.readLine()) != null) {
            String[] partes = linea.split( regex: ",");
            if (partes.length == 3) {
                categorias.add(partes[2].trim());
            }
        }
    } catch (IOException e) {
        System.out.println("[❌] Error al leer productos para categorías: " + e.getMessage());
    }
public class GestorArchivos {  9 usages
    public static void guardarCategoriasUnicas() {  1 usage
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(ARCHIVO_CATEGORIAS))) {
            for (String cat : categorias) {
                bw.write(cat);
                bw.newLine();
            }
            System.out.println("[✅] Archivo categorias.txt generado correctamente.");
        } catch (IOException e) {
            System.out.println("[❌] Error al escribir categorías: " + e.getMessage());
        }
    }

    // borra el archivo productos.txt tras confirmación
    public static void borrarArchivo() {  1 usage
        Scanner sc = new Scanner(System.in);
        System.out.print("\n[o] ¿Seguro que deseas eliminar 'productos.txt'? (s/n): ");
        String respuesta = sc.nextLine().trim().toLowerCase();
        if (respuesta.equals("s")) {
            File archivo = new File(ARCHIVO_PRODUCTOS);
            if (archivo.exists() && archivo.delete()) {
                System.out.println("[🗑️] Archivo productos.txt eliminado correctamente.");
            } else {
                System.out.println("[⚠️] No se pudo eliminar el archivo o no existe.");
            }
        } else {
            System.out.println("[❎] Operación cancelada.");
        }
    }

    // cuenta cuántas líneas tiene productos.txt
    public static int contarProductos() {  1 usage
        int contador = 0;
        try (BufferedReader br = new BufferedReader(new FileReader(ARCHIVO_PRODUCTOS))) {
```

```java
    // cuenta cuántas líneas tiene productos.txt
    public static int contarProductos() {  1 usage
        int contador = 0;
        try (BufferedReader br = new BufferedReader(new FileReader(ARCHIVO_PRODUCTOS))) {
            while (br.readLine() != null) contador++;
        } catch (IOException e) {
            System.out.println("[❌] Error al contar productos: " + e.getMessage());
        }
        return contador;
    }

    // realiza una copia de respaldo del archivo productos.txt
    public static void crearRespaldo() {  1 usage
        try {
            Path origen = Paths.get(ARCHIVO_PRODUCTOS);
            Path destino = Paths.get( first: CARPETA_DATOS + File.separator + "productos_backup.txt");
            Files.copy(origen, destino, StandardCopyOption.REPLACE_EXISTING);
            System.out.println("[✅] Respaldo creado: productos_backup.txt");
        } catch (IOException e) {
            System.out.println("[❌] Error al crear respaldo: " + e.getMessage());
        }
    }
}
```

Preguntas

1. ¿Qué diferencia hay entre File y Buffered Writer?
File representa una ruta o archivo en el sistema pero no escribe ni lee directamente, mientras que BufferedWriter escribe texto de manera eficiente en un archivo, usando un búfer para mejorar el rendimiento.

2. ¿Qué sucede si ejecutas el programa dos veces? ¿Se sobreescriben los datos o se agregan?
Los datos se agregan al final porque se usa new FileWriter(archivo, true) (modo append).

3. ¿Qué pasa si cambias new FileWriter(archivo, true) por new FileWriter(archivo)?
El archivo se sobrescribe completamente en cada ejecución, eliminando los registros anteriores.

4. ¿Por qué se usa try-with-resources en este programa?
Para cerrar automáticamente los recursos (FileWriter, BufferedReader, etc.) y evitar memory-leaks (fugas de memoria) o bloqueos de archivo.

5. ¿Qué función cumple newLine() dentro del Buffered Writer?
Inserta un salto de línea apropiado según el sistema operativo, separando cada producto en una línea distinta.

6. ¿Cómo podrías modificar el programa para leer solo productos de una categoría especifica?
Leeyendo línea por línea y usar:
```java
if (linea.contains("CategoríaBuscada")) {
   System.out.println(linea);
}
```

O dividir cada línea con split(",") y comparar
```java
partes[2].trim().equalsIgnoreCase("CategoríaBuscada").
```

7. ¿Qué ocurriría si intentas leer un archivo que no existe?

Se lanza una excepción FileNotFoundException, que en mi programa se maneja dentro del catch mostrando un mensaje de error.

8. ¿Cómo podrías crear un método contarProductos () que cuente cuántas líneas tiene el archivo?
Leer el archivo con un BufferedReader y aumentar un contador en cada línea (ya implementado).

9. ¿Cuál es la ventaja de usar BufferedReader en lugar de FileReader directamente?
BufferedReader es más eficiente porque lee datos en bloques grandes en memoria, reduciendo el acceso directo al disco.

10. ¿Cómo podrías hacer un respaldo del archivo productos.txt dentro de la misma carpeta?
Usando:
```java
Files.copy(origen, destino, StandardCopyOption.REPLACE_EXISTING);
```
Como en crearRespaldo(), copiando productos.txt a productos_backup.txt.