

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the text 'AD'.

AD

# Práctica 1

PERSISTENCIA EN FICHEROS

Álvaro Manuel Navarro Cruz

2º DAM

04/11/2024

# Índice

Introducción (↑).....	3
Enlace Repositorio GitHub.....	3
1.- Clase ArtículoAComprar (↑) .....	4
2.- Clase ListaDeLaCompra (↑) .....	5
3.- Clase Principal (↑).....	7
4.- Pruebas de ejecución (↑).....	8
5.- Valoración Personal.....	12
6.- Bibliografía.....	13

# Introducción

Para esta práctica se pide diseñar y codificar un modelo que permita gestionar listas de la compra adaptadas a los productos disponibles en una tienda.

El modelo deberá constar de 2 clases, a saber, ArtículoAComprar y ListaDeLaCompra, además de una clase Principal que haga uso de los métodos necesarios para leer un archivo de texto, escribir en él, modificarlo eliminando o agregando artículos y finalmente guardar los cambios en el mismo archivo de texto.

Para esta práctica se utilizará un archivo de texto “.txt”.

**Se explicará el funcionamiento del código a modo de comentarios en el mismo, desarrollando en cada función una breve explicación de lo que se realiza.**

Se **agregarán métodos adicionales a los proporcionados** en la clase ListaDeLaCompra con **objetivo** de hacer **más legible la clase Principal**, agregando métodos para cargar la lista, escribir sobre ella, guardarla y mostrarla.

## Enlace Repositorio GitHub

[https://github.com/AlvaroMfco/AD\\_Practica1.git](https://github.com/AlvaroMfco/AD_Practica1.git)

# 1.- Clase ArtículoAComprar

```
1 package es.studium.Practical;
2
3 public class ArtículoAComprar {
4     private String descripcion; //Descripción del artículo
5     private int cantidad; //Cantidad del artículo
6     private String unidad; //Unidad de medida del artículo
7
8     //Inicializar el artículo con su descripción, cantidad y unidad
9     public ArtículoAComprar(String descripcion, int cantidad, String unidad) {
10         this.descripcion = descripcion;
11         this.cantidad = cantidad;
12         this.unidad = unidad;
13     }
14
15     //Método get para obtener descripción
16     public String getDescripcion() {
17         return descripcion;
18     }
19
20     //Método set para establecer la descripción
21     public void setDescripcion(String descripcion) {
22         this.descripcion = descripcion;
23     }
24
25     //Método get para obtener la cantidad
26     public int getCantidad() {
27         return cantidad;
28     }
29
30     //Método set para establecer la cantidad
31     public void setCantidad(int cantidad) {
32         this.cantidad = cantidad;
33     }
34
35     //Método get para obtener la unidad
36     public String getUnidad() {
37         return unidad;
38     }
39
40     //Método set para establecer la unidad
41     public void setUnidad(String unidad) {
42         this.unidad = unidad;
43     }
44 }
45
```

## 2.- Clase ListaDeLaCompra

```
1 package es.studium.Practica1;
2
3 import java.io.*;
4 import java.util.ArrayList;
5 import java.util.List;
6
7 public class ListaDeLaCompra {
8
9     private String nombreLista; //Variable para almacenar el nombre de la lista
10    //ArrayList para almacenar la lista de artículos
11    private List<ArticuloAComprar> articulos = new ArrayList<ArticuloAComprar>();
12
13    // Constructor por defecto
14    public ListaDeLaCompra() {
15        nombreLista = "";
16    }
17
18    // Constructor por parámetros (permite inicializar una lista con nombre específico)
19    public ListaDeLaCompra(String nombreLista) {
20        this.nombreLista = nombreLista;
21    }
22
23    //Método get para obtener el nombre de la lista
24    public String getNombreLista() {
25        return nombreLista;
26    }
27
28    //Método set para asignar el nombre a la lista
29    public void setNombreLista(String nombreLista) {
30        this.nombreLista = nombreLista;
31    }
32
33    //Método para agregar artículos a la lista de la compra
34    public void agregarProductoAComprar(ArticuloAComprar articulo) {
35        articulos.add(articulo);
36    }
37
38    //Método para eliminar un artículo de la lista según su descripción
39    public void eliminarArticulo(String descripcionArticulo) {
40        articulos.removeIf(articulo -> articulo.getDescripcion().equals(descripcionArticulo));
41    }
42
43    //Método para obtener la lista de artículos
44    public List<ArticuloAComprar> getArticulos() {
45        return articulos;
46    }
47
48    //Método para guardar la lista de artículos en un archivo ".txt"
49    //El método lanza IOException para ser recogido en el main tras su uso
50    public void guardarLista(String nombreArchivo) throws IOException {
51        BufferedWriter writer = new BufferedWriter(new FileWriter(nombreArchivo));
52        //Guardar el nombre de la lista como primera línea
53        writer.write(nombreLista + "\n");
54
55        //Guardar cada artículo en una línea nueva
56        for (ArticuloAComprar articulo : articulos) {
57            writer.write(articulo.getDescripcion() + ";" + articulo.getCantidad() + ";" +
58                articulo.getUnidad() + "\n");
59        }
60        //Cerrar el writer
61        writer.close();
62    }
63 }
```

```

63
64 //Método para cargar la lista desde un archivo de texto
65 //El método lanza IOException para ser recogido en el main tras su uso
66 public void cargarLista(String nombreArchivo) throws IOException {
67     BufferedReader reader = new BufferedReader(new FileReader(nombreArchivo));
68     //Leer y asignar el nombre a la lista
69     this.nombreLista = reader.readLine();
70
71     //Leer los artículos y añadirlos a la lista
72     String linea;
73     while ((linea = reader.readLine()) != null) {
74         //Almacenar en un Array de Strings cada parte del artículo (descripción, cantidad y unidad)
75         //Como se almacenan en formato descripción;cantidad;unidad, los separamos por ";"
76         String[] datos = linea.split(";");
77         String descripcion = datos[0];
78         int cantidad = Integer.parseInt(datos[1]);
79         String unidad = datos[2];
80         /*Instanciamos el artículo a comprar y lo agregamos
81         utilizando la función agregarProductoAComprar()*/
82         agregarProductoAComprar(new ArtículoAComprar(descripcion, cantidad, unidad));
83     }
84     reader.close();
85 }
86
87
88 // Mostrar el contenido de la lista
89 public void mostrarLista() {
90     System.out.println("\nLista de la compra: " + nombreLista);
91     for (ArtículoAComprar articulo : articulos) {
92         //Añadimos espacios en blanco para hacer más legible la consola
93         System.out.println("    " + articulo.getDescripcion() + " - " + articulo.getCantidad()
94             + " " + articulo.getUnidad());
95     }
96 }
97
98 //Método para agregar un artículo si no existe en la lista
99 public void comprobarExistencias(String descripcion, int cantidad, String unidad) {
100     for (ArtículoAComprar articulo : getArticulos()) {
101         if (articulo.getDescripcion().equals(descripcion)) {
102             return; // Si ya existe, no se agrega y se sale del método
103         }
104     }
105     // Si no se encuentra el artículo, se agrega
106     agregarProductoAComprar(new ArtículoAComprar(descripcion, cantidad, unidad));
107     System.out.println("    " + descripcion + " - " + cantidad + " - " + unidad);
108 }
109 }

```

### 3.- Clase Principal

```

1 package es.studium.Practical;
2
3 import java.io.IOException;
4
5 public class Principal {
6
7     public static void main(String[] args) {
8         ListaDeLaCompra lista = new ListaDeLaCompra("Mercadona");
9
10        // Cargar la lista desde el archivo si existe; si no, se crea una nueva lista
11        try {
12            lista.cargarLista("Mercadona.txt");
13            System.out.println("*** Lista cargada desde el archivo txt ***");
14            lista.mostrarLista();
15        }
16        catch (IOException e) {
17            System.out.println("Error al cargar la lista: " + e.getMessage());
18            System.out.println("Lista creada con éxito: " + lista.getNombreLista());
19        }
20
21        //Agregar artículos solo si no están ya en la lista
22        lista.comprobarExistencias("Paquete de arroz", 3, "Kg");
23        lista.comprobarExistencias("Leche", 6, "L");
24        lista.comprobarExistencias("Pan", 4, "U");
25        lista.comprobarExistencias("Paquete de Nueces", 1, "Kg");
26        lista.comprobarExistencias("CocaCola", 2, "L");
27        lista.comprobarExistencias("Fanta", 2, "L");
28
29        //Eliminar el artículo de la lista
30        lista.eliminarArticulo("Leche");
31
32        //Mostrar la lista después de eliminar el artículo
33        System.out.println("\n*** Lista después de eliminar el artículo ***");
34        lista.mostrarLista();
35
36        //Guardar la lista en un archivo de texto
37        try {
38            lista.guardarLista("Mercadona.txt");
39            System.out.println("\n*** Lista guardada en " + lista.getNombreLista() + ".txt ***");
40        }
41        catch (IOException e) {
42            System.out.println("Error al guardar la lista: " + e.getMessage());
43        }
44    }
45}

```

La clase principal está exenta de métodos innecesarios, habiendo sido creados en la clase ListaDeLaCompra, de manera que queda mucho más legible.

En esta clase se crea en primer lugar una instancia de ListaDeLaCompra, en segundo lugar se procede a cargar la lista desde un archivo de texto mediante `lista.cargarLista("Mercadona.txt")`. Si el archivo existe y se carga correctamente, se muestra por consola un mensaje de feedback y se muestra la lista con `lista.mostrarLista()`;

Si el archivo no se puede cargar (porque no exista), se lanza una excepción con el informe y se nos indica que se ha creado una nueva lista.

Finalmente agregamos productos a la lista si no están ya añadidos, realizamos una eliminación de artículo mediante `lista.eliminarArticulo()` y se vuelve a mostrar la lista actualizada.

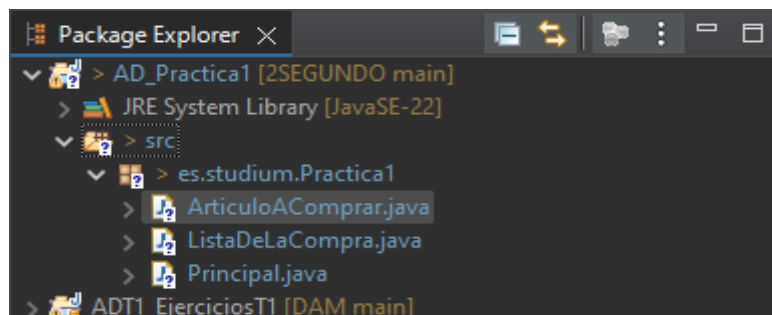
Tras estos pasos, se guardarán los cambios en "Mercadona.txt".

## 4.- Pruebas de ejecución (↑)

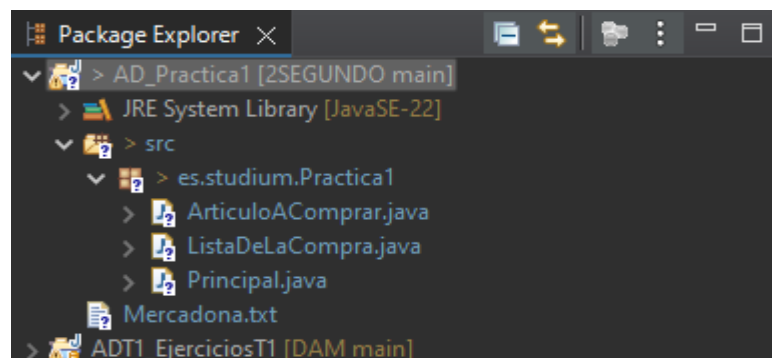
Para realizar las pruebas, primero vamos a comentar la línea para eliminar productos y vamos a comprobar nuestro txt generado:

```
28 //Eliminar el artículo "Leche" de la lista
29 // lista.eliminarArticulo("Leche");
```

Nuestro explorador de paquetes se verá de esta manera antes de ejecutar el programa:



Una vez ejecutado, refrescamos el proyecto y aparecerá el txt creado:



Por consola, obtendremos el siguiente resultado al ejecutarlo por primera vez:

```
Problems Javadoc Declaration Console
<terminated> Principal [Java Application] C:\Users\alvar\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-1706\jre\bin\javaw.exe (1 nov 2024)
Error al cargar la lista: Mercadona.txt (El sistema no puede encontrar el archivo especificado)
Lista creada con éxito: Mercadona
Paquete de arroz - 3 - Kg
Leche - 6 - L
Pan - 4 - U
Paquete de Nueces - 1 - Kg
CocaCola - 2 - L
Fanta - 2 - L

*** Lista después de eliminar el artículo ***

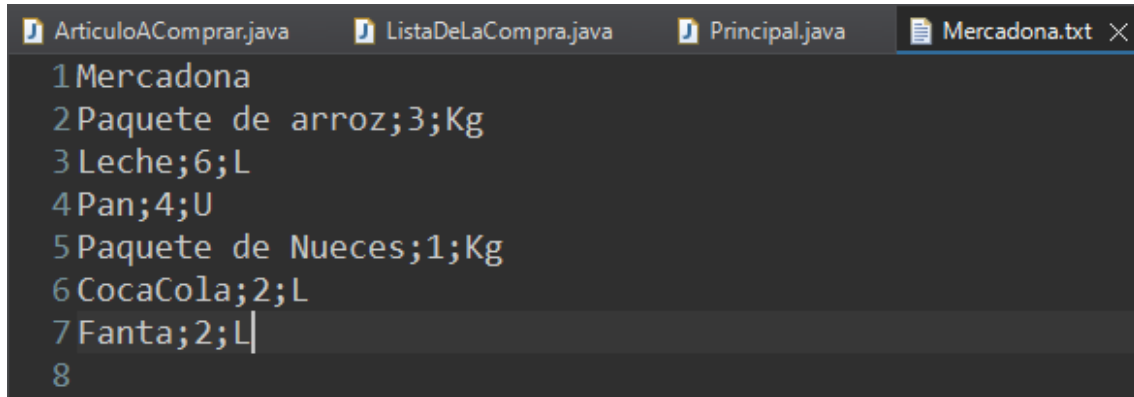
Lista de la compra: Mercadona
Paquete de arroz - 3 Kg
Leche - 6 L
Pan - 4 U
Paquete de Nueces - 1 Kg
CocaCola - 2 L
Fanta - 2 L

*** Lista guardada en Mercadona.txt ***
```



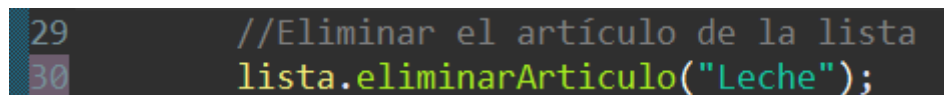
El primer mensaje indicará que no se ha podido encontrar el archivo txt, puesto que aún no está creado.

Si accedemos al archivo txt que se ha generado, veremos los productos añadidos:



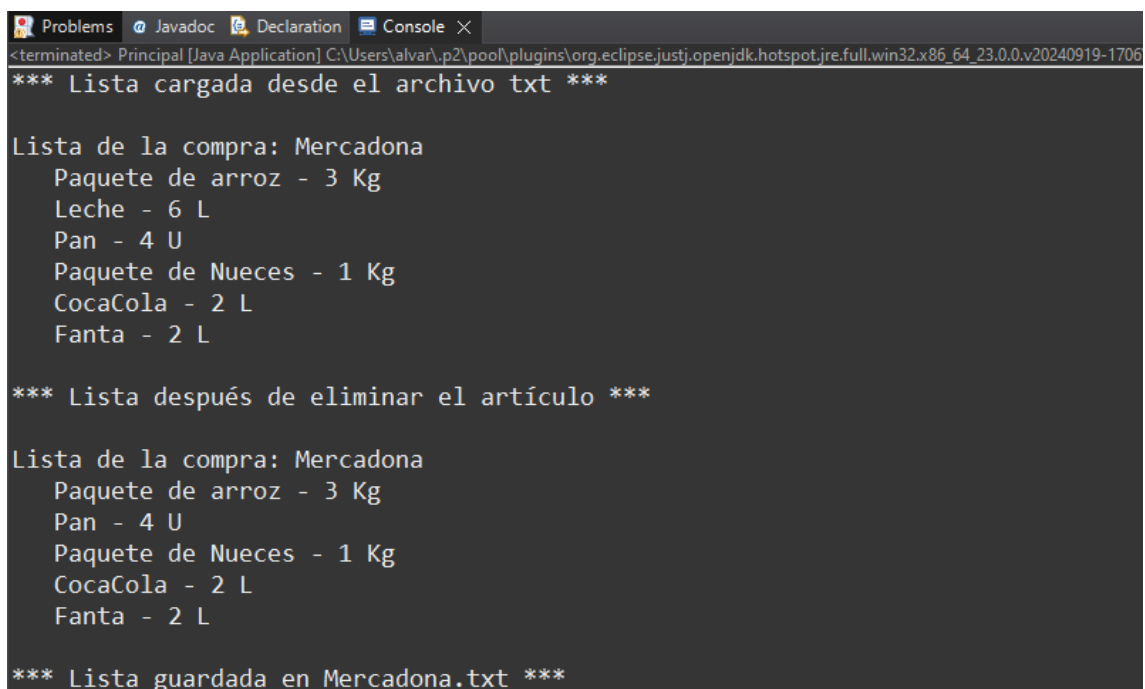
```
ArticuloAComprar.java  ListaDeLaCompra.java  Principal.java  Mercadona.txt X
1 Mercadona
2 Paquete de arroz;3;Kg
3 Leche;6;L
4 Pan;4;U
5 Paquete de Nueces;1;Kg
6 CocaCola;2;L
7 Fanta;2;L
8
```

Ahora vamos a des comentar la línea para eliminar artículos y vamos a borrar el artículo "Leche":



```
29 //Eliminar el artículo de la lista
30 lista.eliminarArticulo("Leche");
```

Ejecutamos y saldrá lo siguiente por consola:



```
Problems  Javadoc  Declaration  Console X
<terminated> Principal [Java Application] C:\Users\alvar\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-1706
*** Lista cargada desde el archivo txt ***

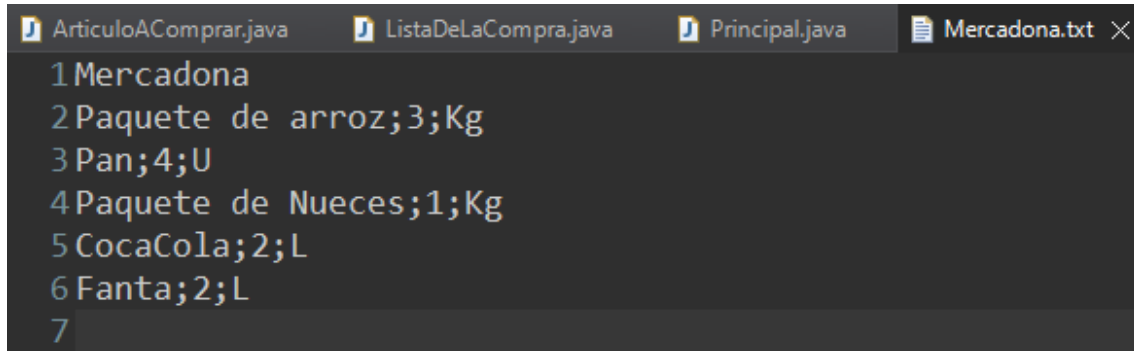
Lista de la compra: Mercadona
Paquete de arroz - 3 Kg
Leche - 6 L
Pan - 4 U
Paquete de Nueces - 1 Kg
CocaCola - 2 L
Fanta - 2 L

*** Lista después de eliminar el artículo ***

Lista de la compra: Mercadona
Paquete de arroz - 3 Kg
Pan - 4 U
Paquete de Nueces - 1 Kg
CocaCola - 2 L
Fanta - 2 L

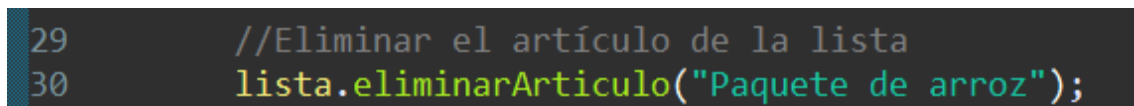
*** Lista guardada en Mercadona.txt ***
```

Si comprobamos el archivo Mercadona.txt, debería haber desaparecido el producto "Leche":



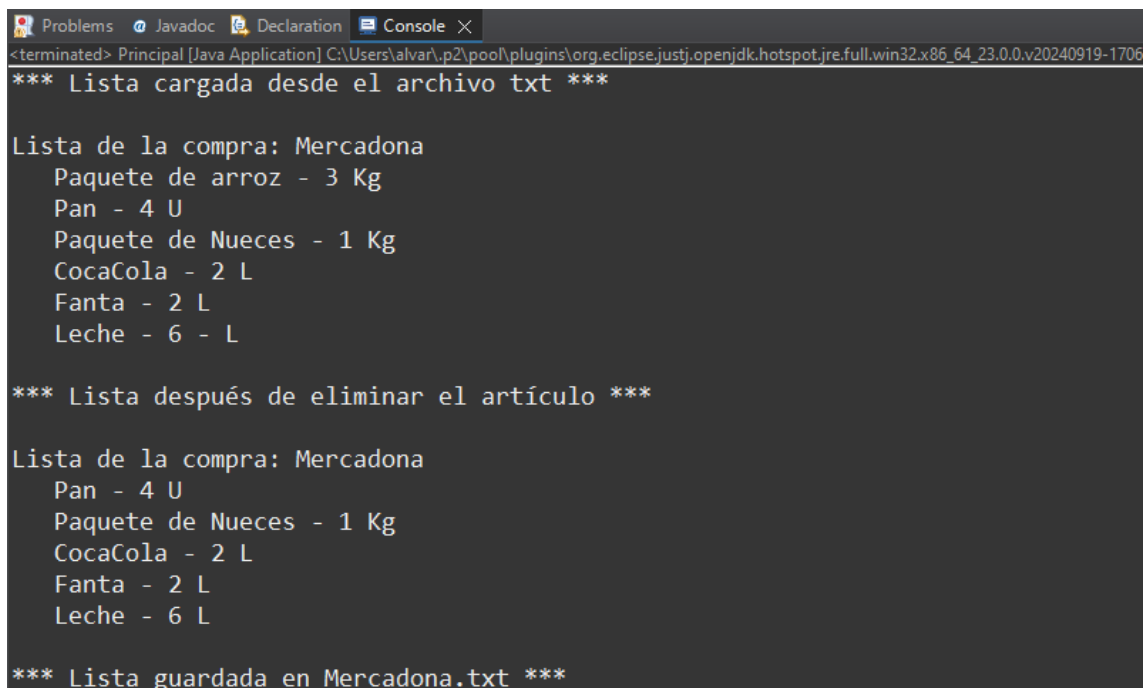
```
ArticuloAComprar.java  ListaDeLaCompra.java  Principal.java  Mercadona.txt X
1Mercadona
2Paquete de arroz;3;Kg
3Pan;4;U
4Paquete de Nueces;1;Kg
5CocaCola;2;L
6Fanta;2;L
7
```

Vamos a seguir eliminando productos, por ejemplo, "Paquete de arroz":



```
29      //Eliminar el artículo de la lista
30      lista.eliminarArticulo("Paquete de arroz");
```

Por consola, aparecerá lo siguiente:



```
Problems  Javadoc  Declaration  Console X
<terminated> Principal [Java Application] C:\Users\alvar\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-1706
*** Lista cargada desde el archivo txt ***

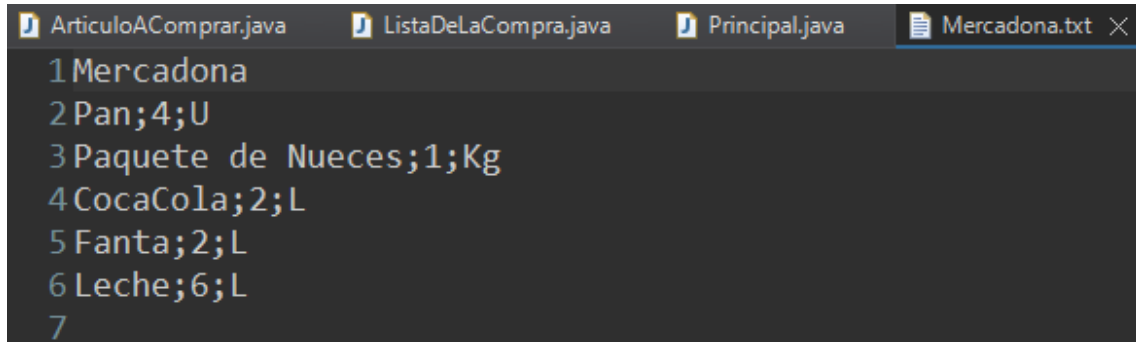
Lista de la compra: Mercadona
Paquete de arroz - 3 Kg
Pan - 4 U
Paquete de Nueces - 1 Kg
CocaCola - 2 L
Fanta - 2 L
Leche - 6 - L

*** Lista después de eliminar el artículo ***

Lista de la compra: Mercadona
Pan - 4 U
Paquete de Nueces - 1 Kg
CocaCola - 2 L
Fanta - 2 L
Leche - 6 L

*** Lista guardada en Mercadona.txt ***
```

Si comprobamos el archivo Mercadona.txt, habrá desaparecido el artículo “Paquete de arroz”:



```
1 Mercadona
2 Pan;4;U
3 Paquete de Nueces;1;Kg
4 CocaCola;2;L
5 Fanta;2;L
6 Leche;6;L
7
```

Como podemos comprobar, ha desaparecido “Paquete de arroz” pero ha vuelto a aparecer “Leche”.

Esto se debe a que, en el Main, agregamos productos siempre que no existan en el txt, y previamente habíamos eliminado “Leche”, por lo que el programa no ha detectado dicho elemento y lo ha introducido de nuevo en la lista.

## 5.- Valoración Personal

Me ha parecido muy interesante realizar esta práctica, ya que he podido afianzar los conocimientos adquiridos en la asignatura Acceso a Datos y reforzar los vistos en la asignatura Entornos de Desarrollo del primer año.

Trabajar con programación orientada a objetos me resulta muy atractivo y me motiva a buscar métodos que mejoren el flujo del código y acabe por convertirse en un trabajo con sentido lógico partiendo de una clase vacía.

Siempre es interesante aprender a realizar de manera más óptima las tareas que se nos enseñan, y con ello lograr ser cada vez mejor programador.

## 6.- Bibliografía

Grupo Studium ([↑](#))

1. *María José Martínez Navas, Acceso a Datos (Práctica Tema 1)*. Publicado en **Grupo Studium**.

Recuperado:

[https://campustudium.com/pluginfile.php/15072/mod\\_resource/content/6/Tema%201%20-%20Persistencia%20en%20ficheros%20-%20Pr%C3%A1ctica\\_v2.pdf](https://campustudium.com/pluginfile.php/15072/mod_resource/content/6/Tema%201%20-%20Persistencia%20en%20ficheros%20-%20Pr%C3%A1ctica_v2.pdf)

Último acceso (01/11/2024).