

ciclo: [DAM]

MÓDULO DE [PROGRAMACIÓN]

[Tarea Nº 06]

Alumno:

[Juan Carlos Filter Martín]

[15456141A]

**Contenido**

[1. Documentos que se adjuntan a este informe. 3](#__RefHeading___Toc115_2797768776)

[2. Crear entorno gráfico. 3](#__RefHeading___Toc117_2797768776)

[3. RA6\_f) Se han creado clases y métodos genéricos. 4](#__RefHeading___Toc119_2797768776)

[4. RA6\_a) Se han escrito programas que utilicen arrays. 8](#__RefHeading___Toc131_2797768776)

[5. RA6\_i) Se han realizado programas que realicen manipulaciones sobre documentos escritos en diferentes lenguajes de intercambio de datos. 9](#__RefHeading___Toc133_2797768776)

[6. RA6\_d) Se han utilizado utilizado iteradores para recorrer los elementos de las listas. 10](#__RefHeading___Toc135_2797768776)

[7. RA6\_c) Se han utilizado listas para almacenar y procesar información. 11](#__RefHeading___Toc137_2797768776)

[Insertar 12](#__RefHeading___Toc2937_2460476763)

[Insertar en JFrame form 14](#__RefHeading___Toc2939_2460476763)

[Resultado del botón insertar ejecutando el programa 15](#__RefHeading___Toc2941_2460476763)

[8. RA6\_g) Se han utilizado expresiones regulares en la búsqueda de patrones en la búsqueda de patrones en cadenas de texto. 16](#__RefHeading___Toc137_2797768776_Copia_1)

[Buscar 16](#__RefHeading___Toc2943_2460476763)

[Buscar en JFrame form 17](#__RefHeading___Toc2945_2460476763)

[Resultado del botón buscar ejecutando el programa 18](#__RefHeading___Toc2947_2460476763)

[Buscar por Coincidencias 19](#__RefHeading___Toc2943_2460476763_Copia_)

[Buscar las coincidencias en JFrame form 20](#__RefHeading___Toc2945_2460476763_Copia_)

[Resultado del botón buscar por “*coincidencia*”ejecutando el programa 21](#__RefHeading___Toc2947_2460476763_Copia_)

[Borrar 23](#__RefHeading___Toc2949_2460476763)

[Borrar en JFrame form 25](#__RefHeading___Toc2951_2460476763)

[Resultado del botón borrar ejecutando el programa 25](#__RefHeading___Toc2953_2460476763)

[Modificar 27](#__RefHeading___Toc2955_2460476763)

[Modificar en JFrame form 29](#__RefHeading___Toc2957_2460476763)

[Resultado del botón Modificar ejecutando el programa 29](#__RefHeading___Toc2959_2460476763)

[Mostrar 31](#__RefHeading___Toc2955_2460476763_Copia_)

[Mostrar en JFrame form 32](#__RefHeading___Toc2957_2460476763_Copia_)

[Resultado del botón Mostrar ejecutando el programa 32](#__RefHeading___Toc2959_2460476763_Copia_)

[Salir 33](#__RefHeading___Toc2955_2460476763_Copia1)

# Documentos que se adjuntan a este informe.

A continuación se detallan los documentos que componen la presente entrega de la tarea:

1. Informe de elaboración de la tarea.
2. Proyecto java

# Crear entorno gráfico.

Antes de comenzar con los apartados de la tarea, vamos a crear una interfaz gráfica.

Con el **JFRAME** creado procederemos a ello introduciremos los botones, Label, Text Area, etc correspondientes.



Una vez creada la interfaz gráfica seguimos con los siguientes apartados de la tarea.

# RA6\_f) Se han creado clases y métodos genéricos.

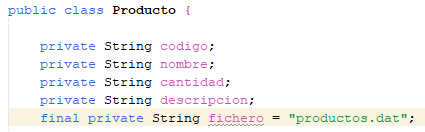
**PRODUCTO**

Vamos a crear una Clase llamada Producto donde se va ir almacenando cada producto con los atributos del programa donde se van a ir almacenando y los métodos getter y setter.



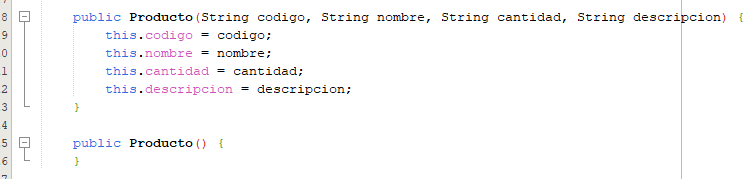
**Atributos**

Esta clase va a contener los atributos de **código, nombre, cantidad y descripción** en la que vamos a almacenar los datos introducidos.



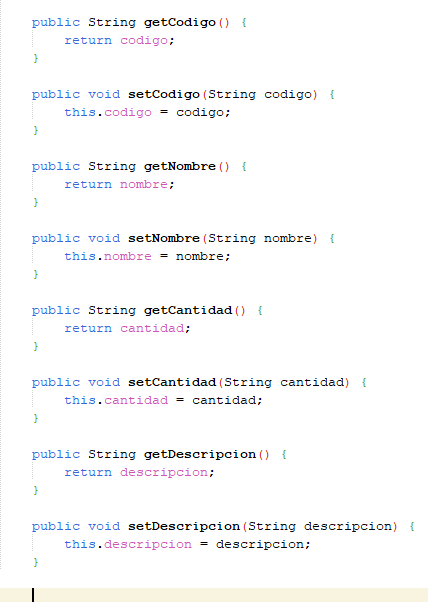
**Constructores**

Con parámetros y sin parámetros.



**Métodos**

Métodos getter y setter de los atributos creados anteriormente.



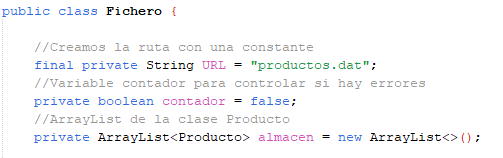
**FICHERO**

También se va a crear una Clase llamada Fichero que va a contener todos los métodos genéricos

**Atributos**

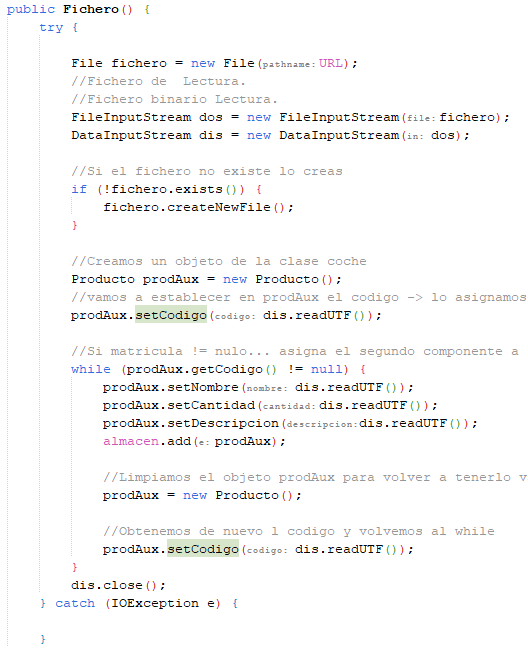
En esta clase en primer lugar podemos ver las variables creadas que se componen de:

* Variable constante para la ruta del fichero **URL**
* Variable boolean para controlar posibles errores
* ArrayList objeto de la clase producto llamada almacen



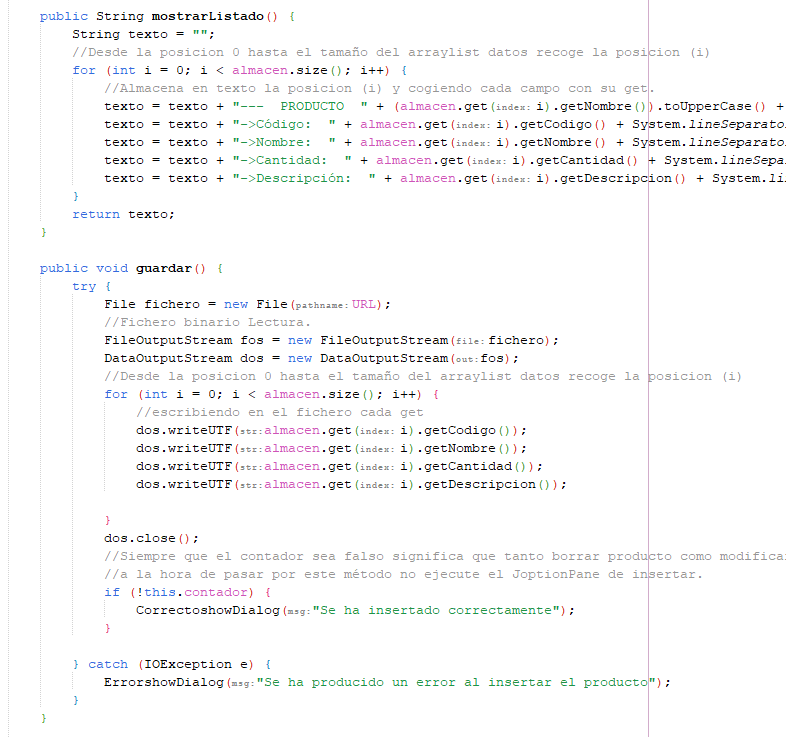
**Constructores**

Se ha creado un constructor sin parámetros con la función para leer el archivo. Entonces cada vez que creemos un objeto de esta clase Fichero se va a leer el archivo



**Métodos**

Los respectivos métodos para poder hacer funcionar el programa siendo estos llamado en el JFrame

Aquí dejo algunos de los métodos de la clase (Más adelante estarán explicado en detalle)

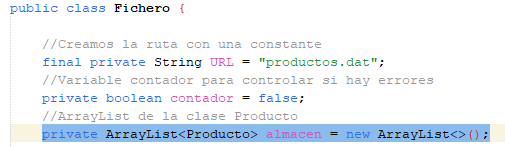
**JuanCarlosFilterMartin**

Y también tenemos que tener una clase JFrame para poder tener nuestra interfaz con sus diferentes métodos.



# RA6\_a) Se han escrito programas que utilicen arrays.

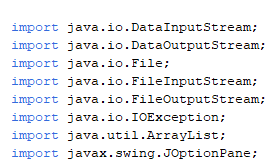
Para manejar los datos se va a utilizar ArrayList de objeto para poder recorrer la clase Producto, poder leer los atributos del mismo, modificar , borrar, añadir los campos al fichero.



Este ArrayList<Producto> se va a llamar almacen y está creado en la clase Fichero.

# RA6\_i) Se han realizado programas que realicen manipulaciones sobre documentos escritos en diferentes lenguajes de intercambio de datos.

En la clase Fichero tenemos las siguientes librerías necesarias que son importadas mediante **import**



import java.io.DataInputStream

import java.io.DataOutputStream

import java.io.File

import java.io.FileInputStream

import java.io.FileOutputStream

Librerias para el manejo de

ficheros binarios

Librerías para controlar las excepciones de fichero.

import java.io.IOException

Librerías para el uso de ArrayList

import java.util.ArrayList

Librerías para el uso de las ventanas emergentes JOptionPane

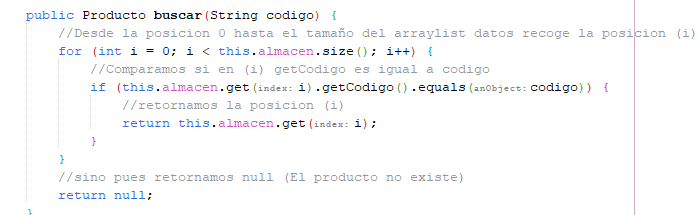
import javax.swing.JOptionPane

# RA6\_d) Se han utilizado utilizado iteradores para recorrer los elementos de las listas.

Se han escrito más métodos usando los ArrayList. En este apartado se muestran el buscar, mostrar y más adelante se explicará todo en detalle

**Métodos de buscar:** Recorre el bucle desde 0 hasta el tamaño del array.

En cada vuelta comprueba el getCogido comparándolo con el codigo pasado por parámetro si encuentra un codigo igual entonces retorna la posición donde se encuentra este en el array



**Método Mostrar:** Recorre el bucle desde 0 hasta el tamaño del array

En cada vuelta pinta en la variable texto lo siguiente concatenandolo:

1- Un titulo con el nombre del producto de la posicion (i) + salto de linea

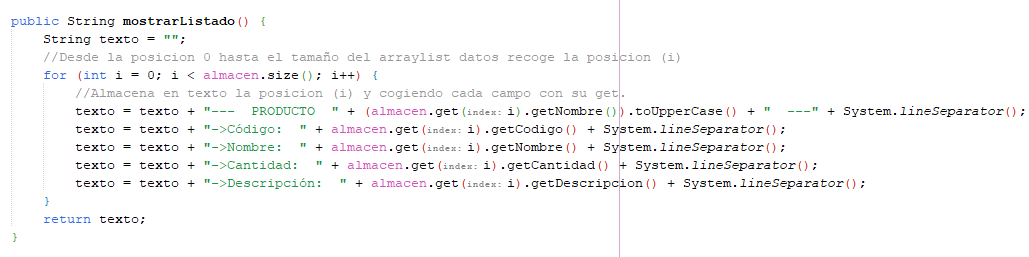
2- El codigo de la posicion (i) + salto de linea

3- El nombre de la posicion (i) + salto de linea

4- El cantidad de la posicion (i) + salto de linea

5- El descripcion de la posicion (i) + salto de linea

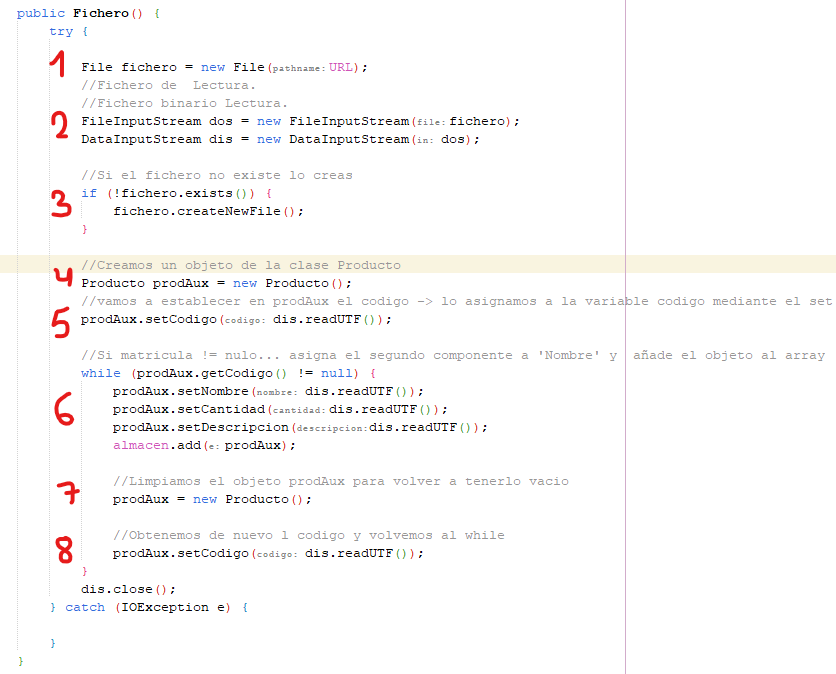
Todo este String texto es enviado con return al llamar a este método



# RA6\_c) Se han utilizado listas para almacenar y procesar información.

Todo el proyecto está documentado pero se va a explicar el funcionamiento del programa desde 0 con la creación de todos sus métodos, etc

Primero se va a crear en el constructor vacío la forma para leer el fichero ya que esté siempre será necesario y cada vez que creemos objetos dentro del método de un botón en el JFrame este será llamado.



**1**. Se prepara la ruta (URL)

**2**. Se abre el fichero binario y se prepara en memoria el fichero binario para leerlo

**3**. Si no existe se crea el fichero

**4.** Se crea un objeto de la clase Producto

**5.** Se lee el fichero con dis.readUTF y este se almacena en la variable codigo de la clase producto mediante el prodAux.setCodigo

**6**. Si el codigo no es nulo entonces se va a asignar los demás componentes a las variables de la clase Producto y este finalmente “las 4 variables” son añadidas al arraylist con el método add

**7**. posteriormente se vuelve a crear un nuevo objeto de producto consiguiendo que este se reinicie.

**8**. Volvemos a leer la siguiente linea del fichero y almacenamos el codigo mediante setCodigo

Esto es repetido hasta que codigo sea nulo (no exista mas codigo en el fichero)

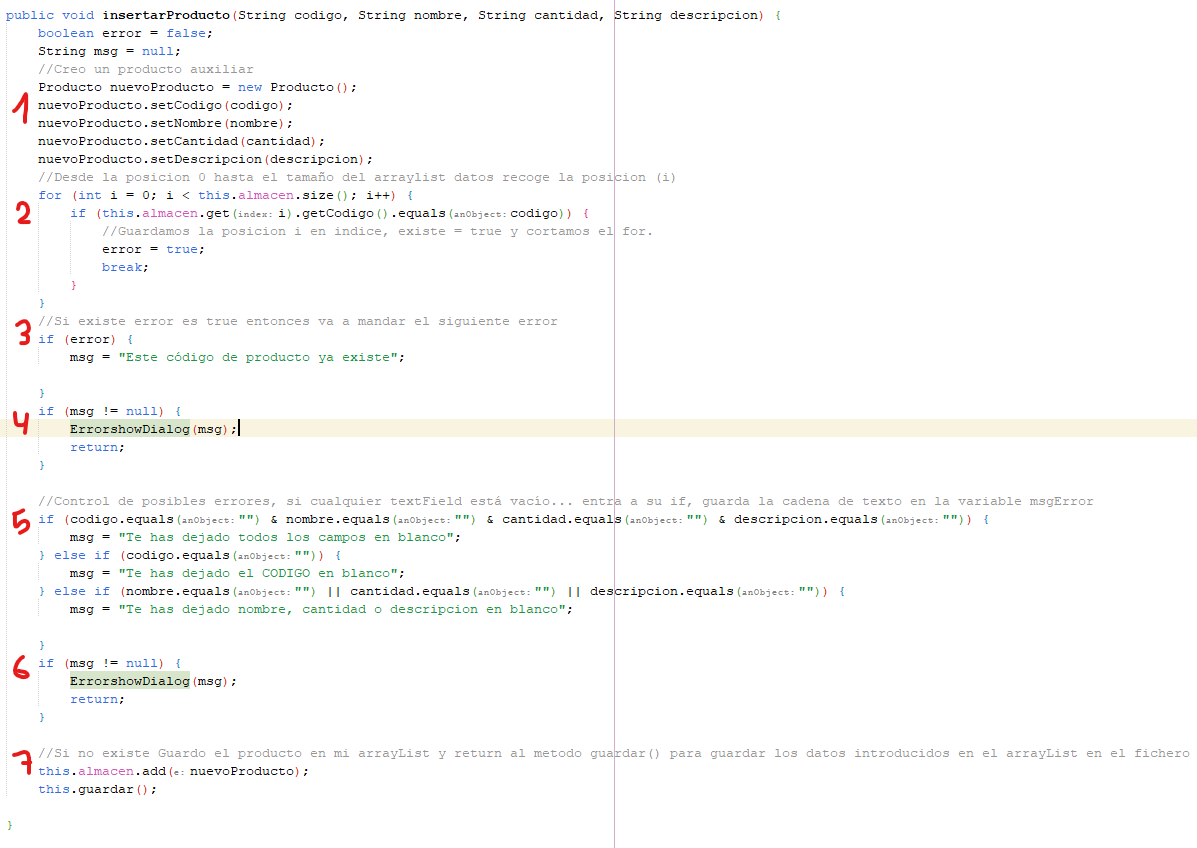
## Insertar

**Para insertar los productos** se van a crear 2 métodos.

Un método para almacenar el contenido al array y el segundo para volcar ese contenido al fichero.

El primer método es el de insertarProducto en el array y esta es su función:

→ public void insertarProducto(codigo, nombre, cantidad, descripcion)

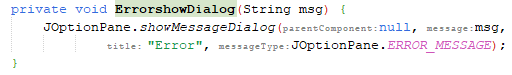


**1**. Se crea un objeto de la clase producto y vamos almacenando en ese objeto con los setter el codigo, nombre, cantidad y descripción recogido por parámetros (estos parámetros al final serán los textField del JFrame)

**2**. Recorre el bucle desde 0 hasta el tamaño del array y va comparando si existe un codigo igual al codigo introducido si lo encuentra la variable error cambia a true.

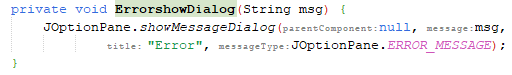
**3**. Si error es true entonces msg = "Este código de producto ya existe"

**4**. Si msg no está vacío se va lanzar el método **ErrorshowDialog** con el texto del msg



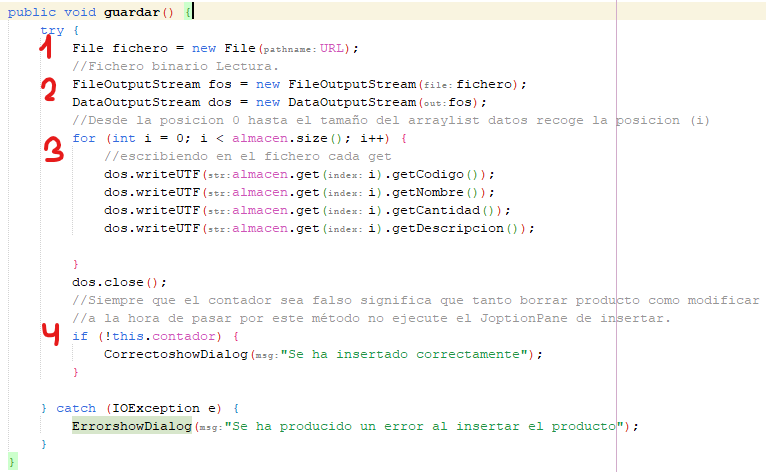
**5**. Se va a comprobar los distintos errores de dejar los campos en blanco con los diferentes if y recibiendo un msg

**6**. Si msg no está vacío se va a lanzar el método **ErrorshowDialog** con el texto del msg



**7**. Por último si todo es correcto el objeto nuevoProducto con las 4 variables introducida mediante sus setter son añadida al ArrayList almacen y entramos en el método guardar para que este ArrayList vuelque los datos al archivo.

→ public void guardar()



1. Se prepara la ruta (URL)

2. Se abre el fichero binario y se prepara en memoria el fichero binario para escribirlo

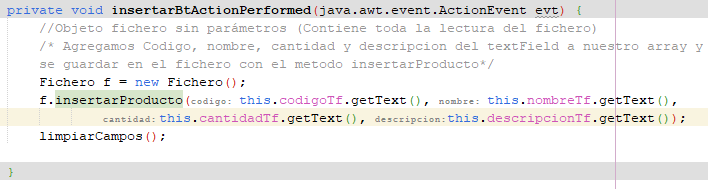
3. Recorre el bucle desde 0 hasta el tamaño del array. En cada vuelta (i) recoge el:

getCodigo, getNombre, getCantidad, getDescipcion de la posición (i) en el array y lo escribe en el archivo mediante dos.writeUTF

4. Cerramos el flujo de escritura y comprobamos que el contador sea verdadero o falso (esto controla modificar y borrar producto ya que si son true no muestre dicho JoptionPane)

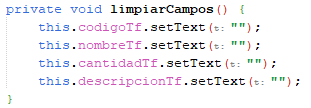
Entonces si es false va a mostrar la siguiente ventana al introducir los datos.

### Insertar en JFrame form

Por ultimo creamos un objeto de la clase Fichero.

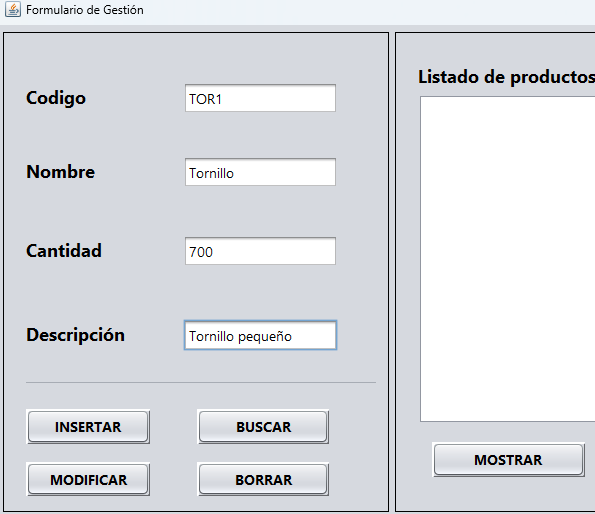
Mediante ese objeto fichero (f) le indicamos el metodo insertarProducto le pasamos los datos codigo, nombre, cantidad y descripcion de los textField obtenido mediante los getText y esto hará la función del método. Por ultimo una vez insertado limpiaremos los campos con el metodo limpiarCampos()

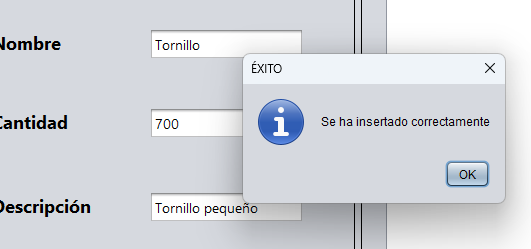
Este método de limpiarCampos se encuentra en el main y de tipo privado ya que solo va a ser utilizado aquí.



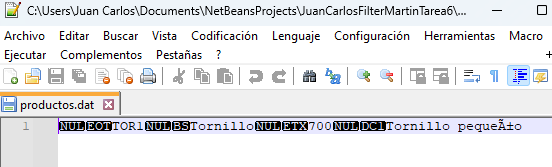
### **Resultado del botón insertar ejecutando el programa**

* **Insertar producto**





* Comprobación en el fichero

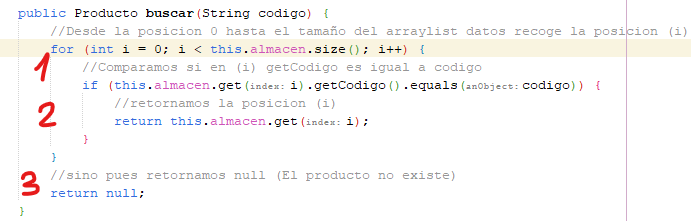


# RA6\_g) Se han utilizado expresiones regulares en la búsqueda de patrones en la búsqueda de patrones en cadenas de texto.

## Buscar

**Para buscar los productos** se van a crear el siguiente método que va a retornar un objeto:

→ public Producto buscar(codigo)



1. Recorre el bucle desde 0 hasta el tamaño del array y va comparando si existe un codigo igual al codigo introducido

2. Si lo encuentra lo coge del array mediante la posición y lo retorna

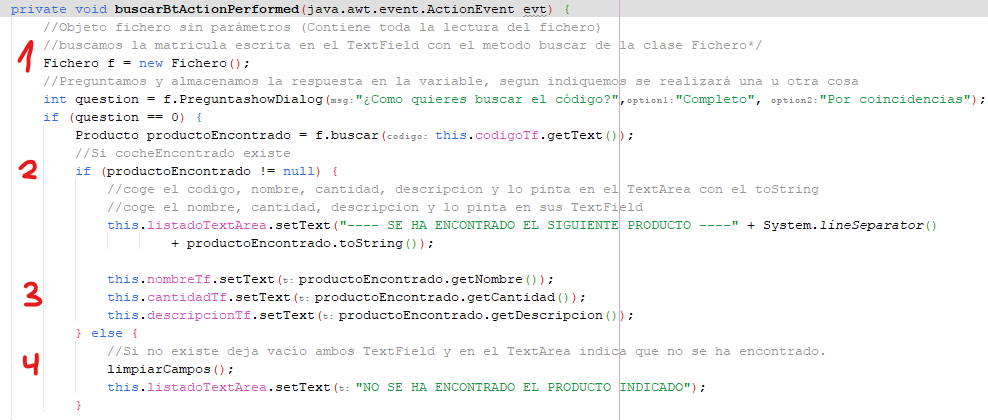
3. Si no lo encuentra entonces retorna null

### Buscar en JFrame form

En el JFrame form creamos un objeto de la clase Fichero. Recordemos que el constructor sin parámetros tiene la función de leer el fichero.



Y realizaremos los siguientes pasos:



1. Se va a crear un objeto de la clase producto que este va a almacenar lo que retorne el método buscar

2. Si obtenemos algo que no sea null significa que si existe ese codigo entonces…

En el textArea va a escribir un título + el objeto productoEncontrado junto al toString para establecer un formato de texto con concatenaciones de los atributos.

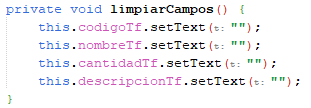
Simplemente primero un titulo para separar cada producto y posteriormente tendremos las lineas a..:

* Código: recogido con el objeto de la clase producto
* Nombre: recogido con el objeto de la clase producto
* Cantidad: recogido con el objeto de la clase producto
* Descripción: recogido con el objeto de la clase producto

3. Y por ultimo rellenamos los campos TextField del nombre, cantidad y descripción con el producto encontrado (El código no ya que el código fue introducido para buscar dicho producto)

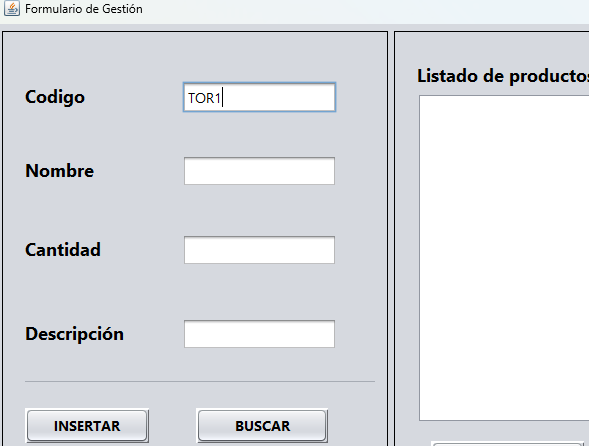
5. Si con el método buscar() de la clase Fichero obtenemos null entonces significa que no existe este codigo de producto y entraría al else que limpiaría el campo codigo con el método limpiarCampos() y mandaría un mensaje en el TextArea de que no se ha encontrado dicho producto.

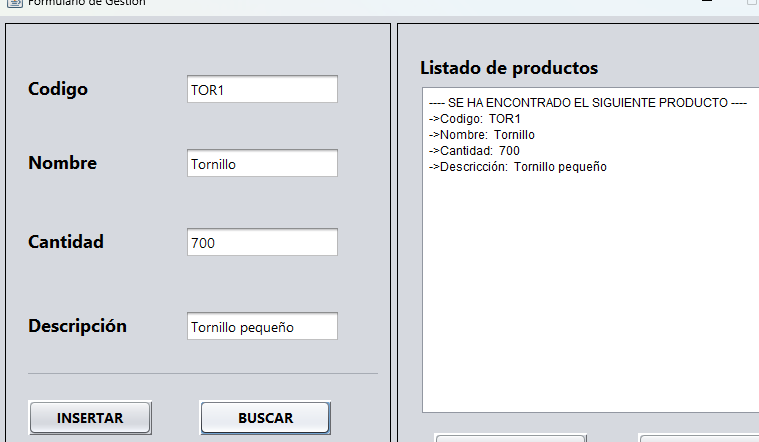
\* Método limpiarCampos()



### **Resultado del botón buscar ejecutando el programa**

* **Buscar producto**

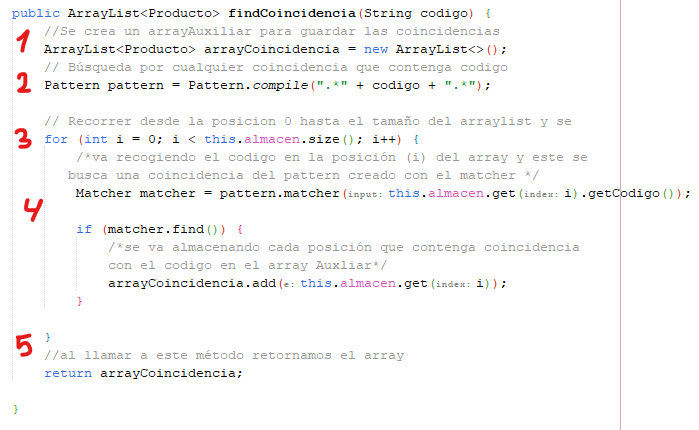
****



## Buscar por Coincidencias

Para buscar productos con el codigo y que coincida con la cadena vamos a crear un nuevo método llamado findCoincidencia siendo este un arraylist de tipo Producto ya que vamos a devolver un arraylist

En este método realizaremos lo siguiente:



1. Se crea un arrayList auxiliar llamado arrayCoincidencia para almacenar en este las coincidencia encontradas más adelante.

2. vamos a recoger el código más las concatenaciones de las expresiones regulares y este mediante las clases Pattern y Martcher comprobaremos las coincidencias.

3. Recorre el bucle desde 0 hasta el tamaño del array

El código obtenido mediante el getCodigo de la posición (i) del array almacén es almacenado y comparado

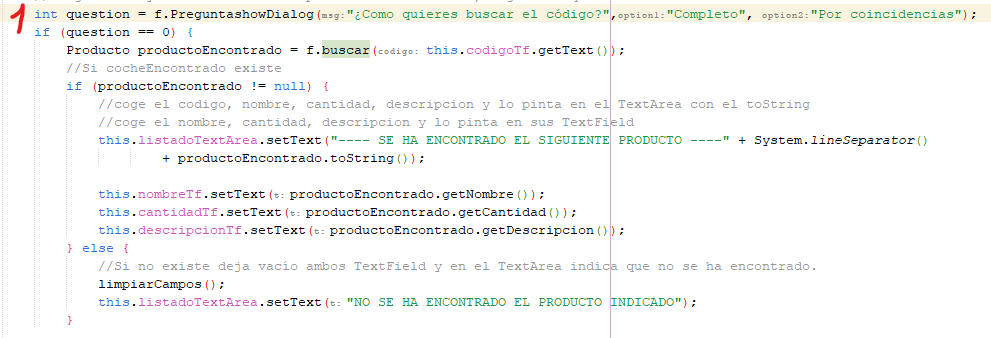
4. Si el resultado de la comparación es true entonces va a añadir ese codigo recogido mediante el array almacén será añadido al arrayCoincidencia y esto será repetido hasta que termine el bucle.

5. Por ultimo al llamar a este método enviaremos el arrayCoincidencia.

### Buscar las coincidencias en JFrame form

En la clase main agregaremos el siguiente código en el botón de buscar.

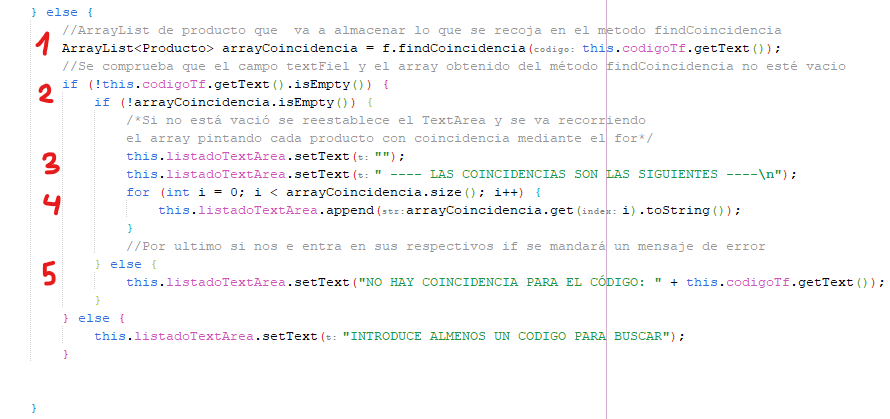
Vamos a explicarlo en dos partes:



1. Lo que se va a realizar en primer lugar es preguntar al usuario si quiere que la busqueda sea completa o por coincidencia recogiendo en caso de que sea 0 o 1 según la opción indicada.

* Opcion1: Completo → 0
* Opcion2: Por coincidencias → 1

Dicho esto si es completa entonces entrará a lo explicado anteriormente sobre el boton buscar, en cambio si la opción seleccionada es la 2º…. se va a realizar lo siguiente entrando en el else del if indicado.



1. Se crea un arrayList de tipo objeto llamado arrayCoincidencia en el se va a almacenar lo que retorne el método findCoincidencia pasandole a este por parámetros el codigo del textField.

2. Si el campo TextField no es vacío hace otra comprobación de que el arrayCoincidencia tampoco esté vacío y si no está vacío …

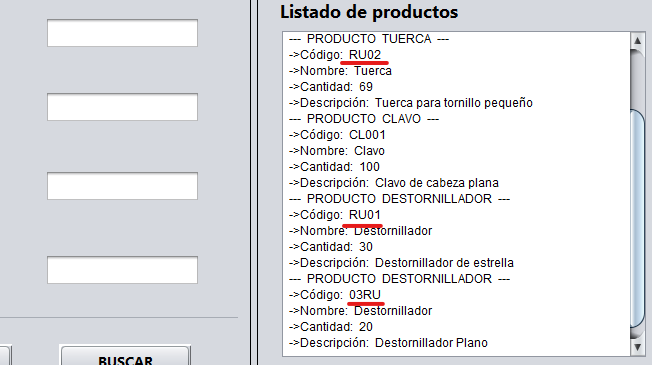
3. Va borrar en el TextArea y va a escribir un Titulo

4. En este punto se va a entrar a un bucle for en el que se va a recorrer desde la posicion 0 hasta el tamaño del array pintando en cada vuelta cada producto del array

5. Por ultimo tener en cuenta que si en el if de comprobar si el arrayCoincidencia o el textField código fuera vacío enviaría un mensaje de error

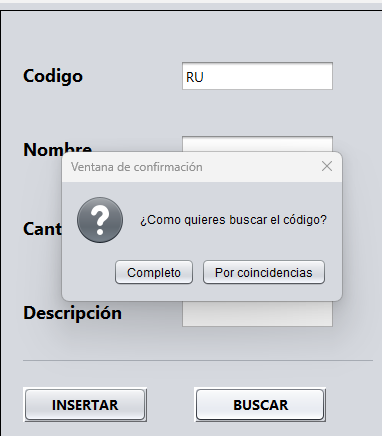
### **Resultado del botón buscar por “*coincidencia*”ejecutando el programa**

Primero vamos a comprobar el listado en el que han agregado varios productos con códigos similares

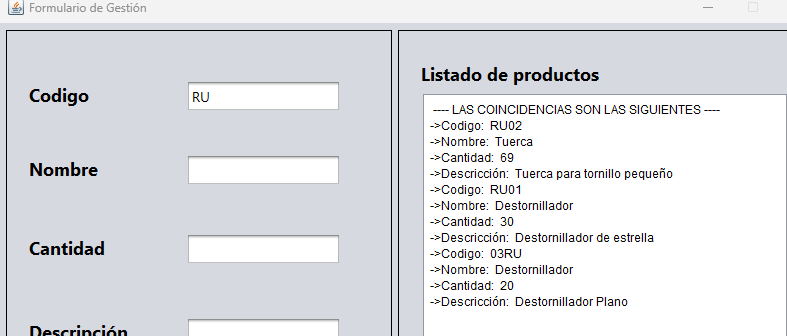


Ahora vamos a introducir en el campo código: **RU**

Se preguntará como quiere realizar la búsqueda y si pulsamos **“Por coincidencias”** enviará 1 en la respuesta obtenida y realizará el código que busca las coincidencias del texto enviado



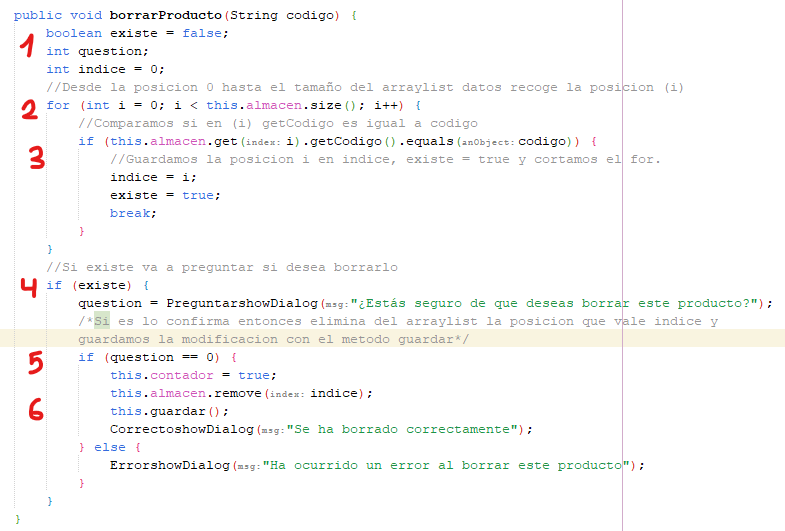
El resultado final es el siguiente mostrando 3 productos que contenía “RU” en el codigo



## Borrar

Para borrar tenemos el siguiente 2 método en la clase fichero. Uno será para borrar el producto en el array y el otro método es el de guardar() que lo necesitaremos para actualizar los cambios en el fichero.

→ public void borrarProducto(codigo)



1. Se crean las variables que se necesitarán mas adelante en el método borrarProducto(codigo)

2. Recorre el bucle desde 0 hasta el tamaño del array y va comparando si existe un codigo igual al codigo introducido

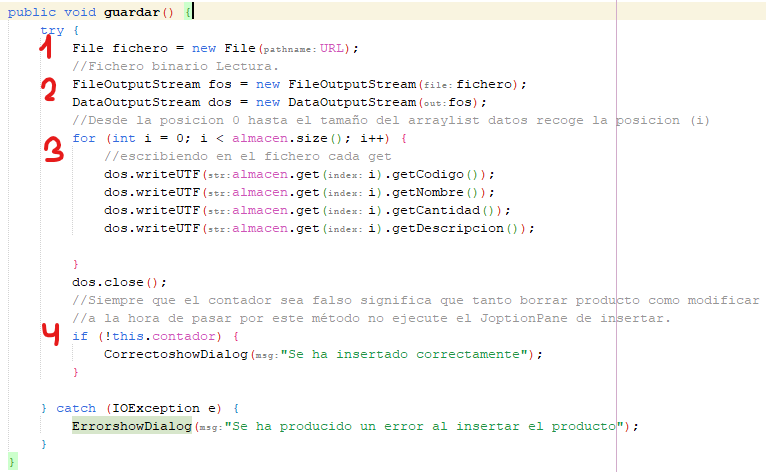
3. Si existe asignamos a indice 1, existe = true y salimos del for

4. Si la variable existe es true entonces se preguntará si se desea borrar el producto

5. Si obtenemos Si en la pregunta entonces:

6. Entra al if , asignamos a la variable contador true ,borramos la posición del objeto en el array, entramos en el método guardar() y mostramos un mensaje que se a borrado correctamente.

→ public void guardar()



1. Se prepara la ruta (URL)

2. Se abre el fichero binario y se prepara en memoria el fichero binario para escribirlo

3. Recorre el bucle desde 0 hasta el tamaño del array. En cada vuelta (i) recoge el:

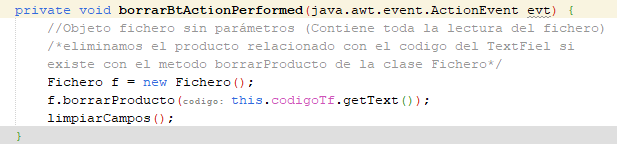
getCodigo, getNombre, getCantidad, getDescipcion de la posición (i) en el array y lo escribe en el archivo mediante dos.writeUTF

4. Cerramos el flujo de escritura y comprobamos que el contador sea verdadero o falso (esto controla modificar y borrar producto ya que si son true no muestre dicho JoptionPane)

Como es true el valor que asignamos a contador ignorará el if y no mostrará el mensaje de insertado correctamente.

Ya terminado con este método queda volver al anterior para ejecutar el mensaje ya sea que se ha borrado correctamente o que ha ocurrido un error con el borrado.

### Borrar en JFrame form



En el JFrame form creamos el objeto de la clase Fichero sin parámetros para que este objeto lea el fichero al declararlo

Con el objeto Fichero f llamamos al método borrarProducto mandándole el codigo introducido en el textField.

Por último respecto a este boton usamos el método de limpiarCampos() para dejar vacío todos los TextField

### **Resultado del botón borrar ejecutando el programa**

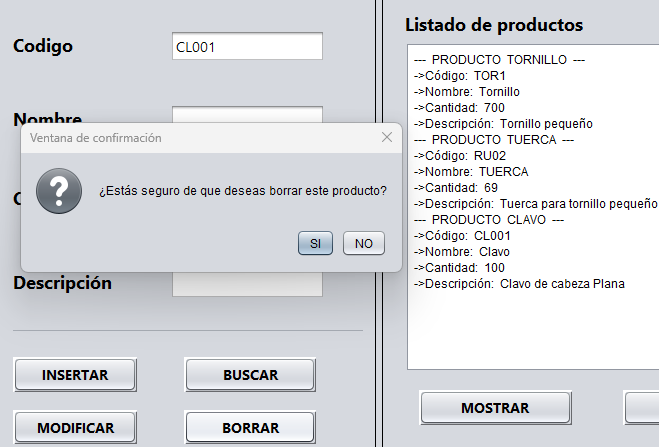
Tenemos varios productos insertados como se puede ver en el listado de productos

Se va a usar borrar para eliminar el producto CLAVO.

Para ello vamos a introducir el código de este producto en su campo TextField

****

Confirmaremos el mensaje emergente y al pulsar Si el producto será borrado.

****

Si volvemos a mostrar la lista de productos podemos ver que ha sido eliminado

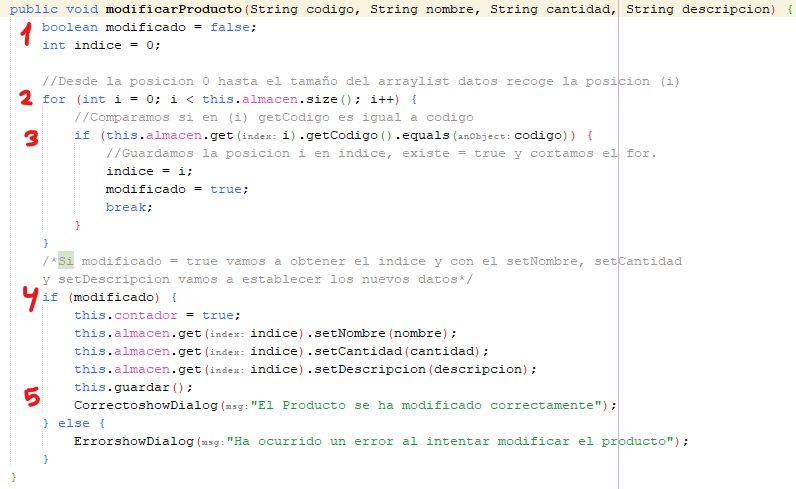
****

## **Modificar**

Para modificar tenemos tenemos 2 métodos igual que en la función de borrado.

Uno será para modificar el producto en el array y el otro método es el de guardar() que lo necesitaremos para actualizar los cambios en el fichero.

→ public void modificarProducto(codigo, nombre, cantidad, descripcion)



1. Creamos las variables que se necesitaran en el método modificarProducto

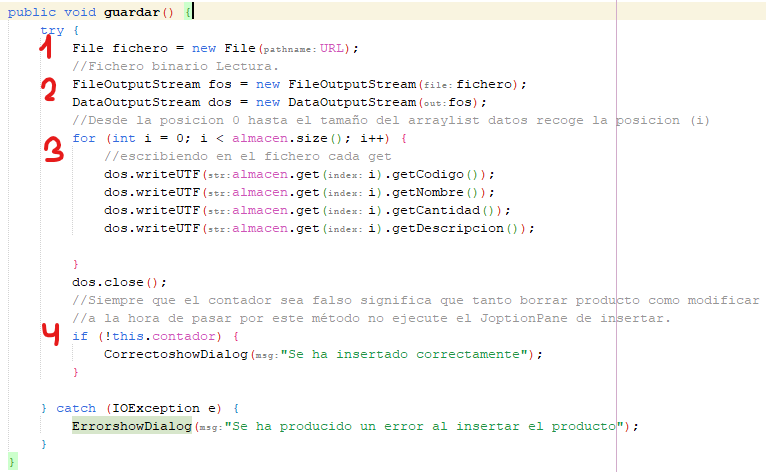
2. Recorre el bucle desde 0 hasta el tamaño del array

3. En esa vuelta del bucle compara el codigo pasado por parámetro que sea igual al del array con la posicion (i). Si encuentra alguno igual entonces la variable indice pasa a tener valor de 1, modificado a true y cortamos el bucle con el break.

4 Si modificado es true (Significa que existe ese código) entonces la variable contador pasa a ser 1 y se establece el nombre, cantidad y en el array con los setter.

5. Posteriormente entramos al método guardar()

→ public void guardar()

1. Se prepara la ruta (URL)

2. Se abre el fichero binario y se prepara en memoria el fichero binario para escribirlo

3. Recorre el bucle desde 0 hasta el tamaño del array. En cada vuelta (i) recoge el:

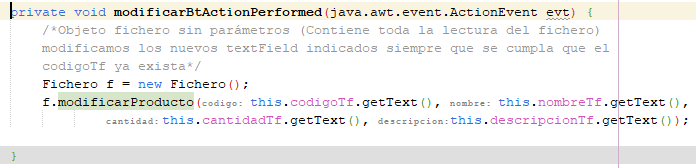
getCodigo, getNombre, getCantidad, getDescipcion de la posición (i) en el array y lo escribe en el archivo mediante dos.writeUTF

4. Cerramos el flujo de escritura y comprobamos que el contador sea verdadero o falso (esto controla modificar y borrar producto ya que si son true no muestre dicho JOptionPane)

Como es true el valor que asignamos a contador ignorará el if y no mostrará el mensaje de insertado correctamente.

Ya terminado con este método queda volver al anterior para ejecutar el mensaje ya sea que se ha modificado correctamente o que ha ocurrido un error al intentar modificar el producto.

### Modificar en JFrame form

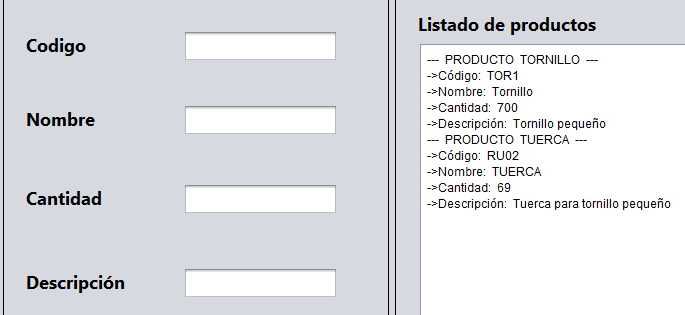


1. Creamos un objeto de la clase Fichero sin parámetros que este tiene la lectura del fichero

2. Y con el objeto de la clase Fichero llamamos al método modificarProducto pesándole por parámetros los distintos TextField

### **Resultado del botón Modificar ejecutando el programa**

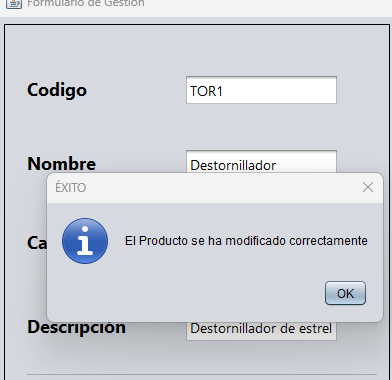
Lo primero mostramos la lista de los posibles productos en el fichero para modificar



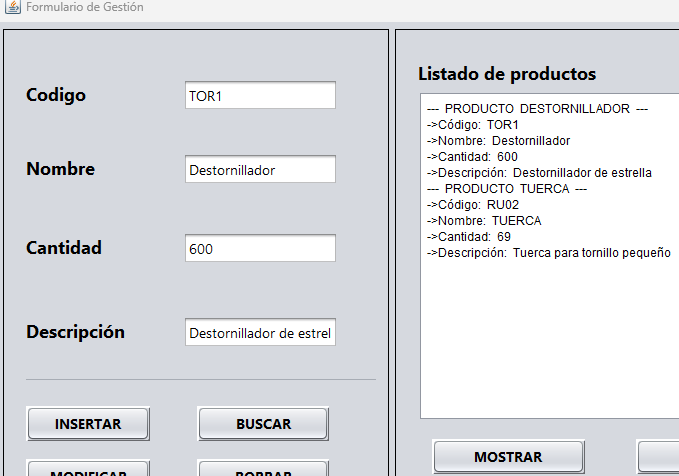
Modificaremos el producto código TOR1 asignándole el nuevo nombre de destornillador.



Si pulsamos en modificar mandará un mensaje en ventana emergente de que ha sido modificado ya que el código es el mismo al que tenía el anterior producto.

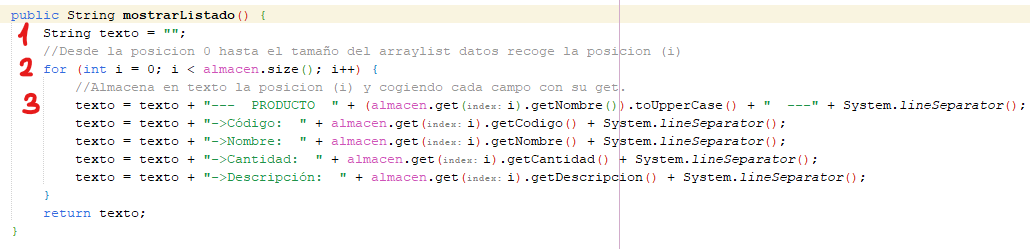


Por ultimo al pulsar en Ok y comprobamos la lista de nuevo podemos ver como ha cambiado el producto con el mismo código a ser de nombre destornillador



## **Mostrar**

Para tener la función de mostrar los productos se va a utilizar el siguiente método, sabiendo que siempre antes de llamar a este método el fichero será leido.



1. Se crea una variable texto en la que se almacenará una cadena de texto concatenada

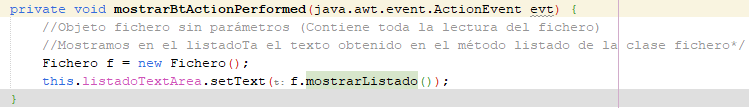
2. Recorre el bucle desde 0 hasta el tamaño del array y en cada vuelta del bucle…

3. Se va a almacenar en primer lugar un titulo del producto recogiendo el nombre con el setNombre seguido de un salto de linea.

Lo mismo para el resto de linea, concatenando siempre a la variable texto con la finalidad de obtener toda la lista en esa variable hasta que se termine de leer el arrayList.

Esto devuelve el texto cuando es llamado

### Mostrar en JFrame form

Con el objeto fichero sin parámetros vamos a leer el fichero.

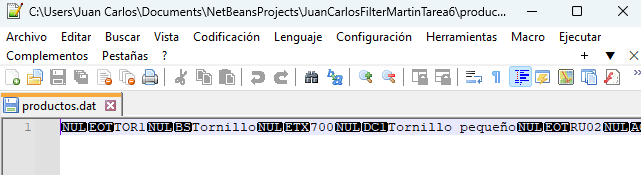
Y finalmente pasamos por parámetros el método mostrarListado en el textArea para que este se muestre.

### **Resultado del botón Mostrar ejecutando el programa**

Pulsamos en el botón mostrar y podemos ver el resultado en el TexArea



Si por último visualizamos el archivo podemos ver los articulos aunque está en formato dat



## **Salir**

Para este botón simplemente escribiremos en el método del botón salir:

System.exit(0);

