



CICLO: [DAM]
MÓDULO DE [PROGRAMACIÓN]

[Tarea N° 06]

Alumno:
[Juan Carlos Filter Martín]
[15456141A]

Contenido

1. Documentos que se adjuntan a este informe.....	3
2. Crear entorno gráfico.....	3
3. RA6_f) Se han creado clases y métodos genéricos.....	4
4. RA6_a) Se han escrito programas que utilicen arrays.....	8
5. RA6_i) Se han realizado programas que realicen manipulaciones sobre documentos escritos en diferentes lenguajes de intercambio de datos.....	9
6. RA6_d) Se han utilizado utilizado iteradores para recorrer los elementos de las listas.....	10
7. RA6_c) Se han utilizado listas para almacenar y procesar información.....	11
Insertar.....	12
Insertar en JFrame form.....	14
Resultado del botón insertar ejecutando el programa.....	15
8. RA6_g) Se han utilizado expresiones regulares en la búsqueda de patrones en la búsqueda de patrones en cadenas de texto.....	16
Buscar.....	16
Buscar en JFrame form.....	17
Resultado del botón buscar ejecutando el programa.....	18
Buscar por Coincidencias.....	19
Buscar las coincidencias en JFrame form.....	20
Resultado del botón buscar por “coincidencia” ejecutando el programa.....	21
Borrar.....	23
Borrar en JFrame form.....	25
Resultado del botón borrar ejecutando el programa.....	25
Modificar.....	27
Modificar en JFrame form.....	29
Resultado del botón Modificar ejecutando el programa.....	29
Mostrar.....	31
Mostrar en JFrame form.....	32
Resultado del botón Mostrar ejecutando el programa.....	32
Salir.....	33

1. 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

2. 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.

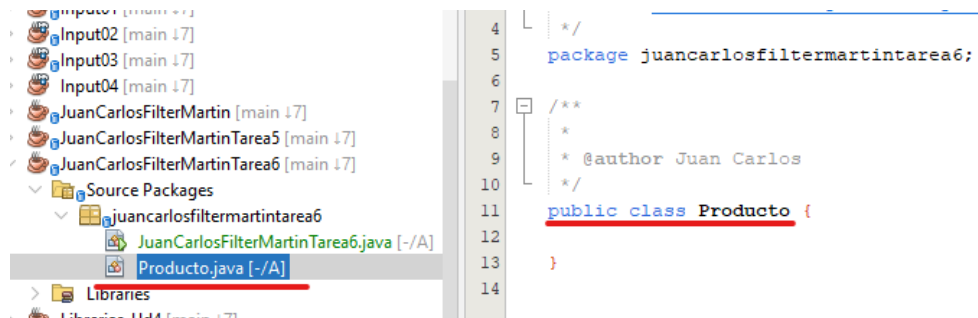
The image shows a Java Swing window titled "Listado de productos". The window is divided into two main sections. The left section contains four text input fields with labels "Codigo", "Nombre", "Cantidad", and "Descripción". Below these fields are four buttons: "INSERTAR", "BUSCAR", "MODIFICAR", and "BORRAR". The right section contains a large text area for displaying the product list, with "MOSTRAR" and "SALIR" buttons at the bottom.

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

3. 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.

```
public class Producto {

    private String codigo;
    private String nombre;
    private String cantidad;
    private String descripcion;
```

Constructores

Con parámetros y sin parámetros.

```
8  L
9  L
10 L
11 L
12 L
13 L
14 L
15 L
16 L
17 L

public Producto(String codigo, String nombre, String cantidad, String descripcion) {
    this.codigo = codigo;
    this.nombre = nombre;
    this.cantidad = cantidad;
    this.descripcion = descripcion;
}

public Producto() {
}
```

Métodos

Métodos getter y setter de los atributos creados anteriormente.

```
public String getCodigo() {  
    return codigo;  
}  
  
public void setCodigo(String codigo) {  
    this.codigo = codigo;  
}  
  
public String getNombre() {  
    return nombre;  
}  
  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}  
  
public String getCantidad() {  
    return cantidad;  
}  
  
public void setCantidad(String cantidad) {  
    this.cantidad = cantidad;  
}  
  
public String getDescripcion() {  
    return descripcion;  
}  
  
public void setDescripcion(String descripcion) {  
    this.descripcion = descripcion;  
}
```

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

```

public class Fichero {

    //Creamos la ruta con una constante
    final private String URL = "productos.dat";
    //Variable contador para controlar si hay errores
    private boolean contador = false;
    //ArrayList de la clase Producto
    private ArrayList<Producto> almacen = new ArrayList<>();
}

```

Constructores

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

```

public Fichero() {
    try {

        File fichero = new File(pathname: URL);
        //Fichero de Lectura.
        //Fichero binario Lectura.
        FileInputStream dos = new FileInputStream(file: fichero);
        DataInputStream dis = new DataInputStream(in: dos);

        //Si el fichero no existe lo creas
        if (!fichero.exists()) {
            fichero.createNewFile();
        }

        //Creamos un objeto de la clase coche
        Producto prodAux = new Producto();
        //vamos a establecer en prodAux el codigo -> lo asignamos
        prodAux.setCodigo(codigo: dis.readUTF());

        //Si matricula != nulo... asigna el segundo componente a
        while (prodAux.getCodigo() != null) {
            prodAux.setNombre(nombre: dis.readUTF());
            prodAux.setCantidad(cantidad: dis.readUTF());
            prodAux.setDescripcion(descripcion: dis.readUTF());
            almacen.add(e: prodAux);

            //Limpiamos el objeto prodAux para volver a tenerlo v
            prodAux = new Producto();

            //Obtenemos de nuevo l codigo y volvemos al while
            prodAux.setCodigo(codigo: dis.readUTF());
        }
        dis.close();
    } catch (IOException e) {
    }
}

```

Métodos

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

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

```
public String mostrarListado() {
    String texto = "";
    //Desde la posicion 0 hasta el tamaño del arraylist datos recoge la posicion (i)
    for (int i = 0; i < almacen.size(); i++) {
        //Almacena en texto la posicion (i) y cogiendo cada campo con su get.
        texto = texto + "--- PRODUCTO " + (almacen.get(index: i).getNombre()).toUpperCase() +
        texto = texto + "->Código: " + almacen.get(index: i).getCodigo() + System.lineSeparator() +
        texto = texto + "->Nombre: " + almacen.get(index: i).getNombre() + System.lineSeparator() +
        texto = texto + "->Cantidad: " + almacen.get(index: i).getCantidad() + System.lineSeparator() +
        texto = texto + "->Descripción: " + almacen.get(index: i).getDescripcion() + System.lineSeparator() +
    }
    return texto;
}

public void guardar() {
    try {
        File fichero = new File(pathname: URL);
        //Fichero binario Lectura.
        FileOutputStream fos = new FileOutputStream(file: fichero);
        DataOutputStream dos = new DataOutputStream(out: fos);
        //Desde la posicion 0 hasta el tamaño del arraylist datos recoge la posicion (i)
        for (int i = 0; i < almacen.size(); i++) {
            //escribiendo en el fichero cada get
            dos.writeUTF(str:almacen.get(index: i).getCodigo());
            dos.writeUTF(str:almacen.get(index: i).getNombre());
            dos.writeUTF(str:almacen.get(index: i).getCantidad());
            dos.writeUTF(str:almacen.get(index: i).getDescripcion());
        }
        dos.close();
        //Siempre que el contador sea falso significa que tanto borrar producto como modifica:
        //a la hora de pasar por este método no ejecute el JOptionPane de insertar.
        if (!this.contador) {
            Correcto.showMessageDialog(msg:"Se ha insertado correctamente");
        }
    } catch (IOException e) {
        Error.showMessageDialog(msg:"Se ha producido un error al insertar el producto");
    }
}
```

JuanCarlosFilterMartin

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

```
public class JuanCarlosFilterMartin extends javax.swing.JFrame {

    public JuanCarlosFilterMartin() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void codigoTfActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void salirBtActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(status: 0);
    }

    private void insertarBtActionPerformed(java.awt.event.ActionEvent evt) {
        //Objeto fichero sin parámetros (Contiene toda la lectura del fichero)
        // Agregamos Codigo, nombre, cantidad y descripcion del textField a nuestro array y se guardar en e.
        Fichero f = new Fichero();
        f.insertarProducto(codigo: this.codigoTf.getText(), nombre: this.nombreTf.getText(), cantidad: this.cantid,
        limpiarCampos());
    }

    private void mostrarBtActionPerformed(java.awt.event.ActionEvent evt) {
        //Objeto fichero sin parámetros (Contiene toda la lectura del fichero)
        //Mostramos en el listadoTa el texto obtenido en el método listado de la clase fichero*/
        Fichero f = new Fichero();
        this.listadoTextArea.setText(v: f.mostrarListado());
    }

    private void buscarBtActionPerformed(java.awt.event.ActionEvent evt) {
        //Objeto fichero sin parámetros (Contiene toda la lectura del fichero)
        //buscamos la matricula escrita en el TextField con el metodo buscar de la clase Fichero*/
        Fichero f = new Fichero();
        Producto productoEncontrado = f.buscar(codigo: this.codigoTf.getText());
        //Si cocheEncontrado existe
        if (productoEncontrado != null) {
            //Llega al codigo nombre cantidad descripcion y la pista en el TextField con el texto
        }
    }
}
```

4. 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.

```
public class Fichero {

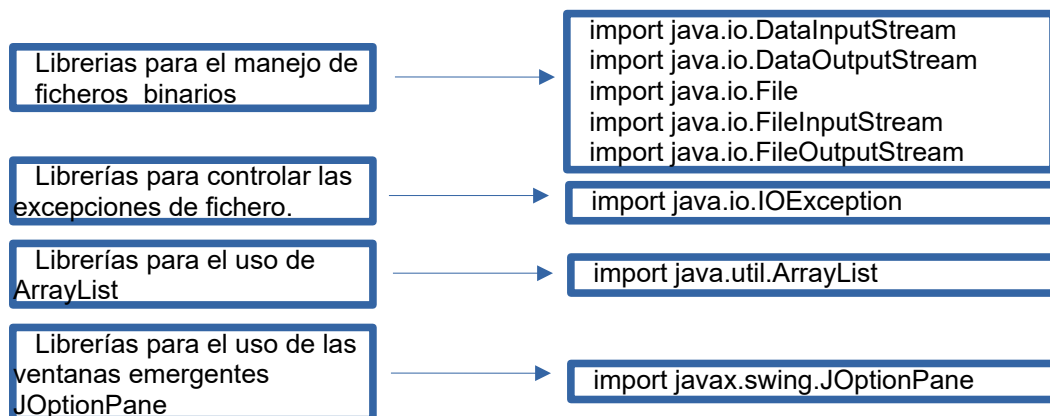
    //Creamos la ruta con una constante
    final private String URL = "productos.dat";
    //Variable contador para controlar si hay errores
    private boolean contador = false;
    //ArrayList de la clase Producto
    private ArrayList<Producto> almacen = new ArrayList<>();
}
```

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

5. 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;
import java.io.IOException;
import java.util.ArrayList;
import javax.swing.JOptionPane;
```



6. 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 getCodigo comparándolo con el código pasado por parámetro si encuentra un código igual entonces retorna la posición donde se encuentra este en el array

```
public Producto buscar(String codigo) {
    //Desde la posición 0 hasta el tamaño del arraylist datos recoge la posición (i)
    for (int i = 0; i < this.almacen.size(); i++) {
        //Comparamos si en (i) getCodigo es igual a código
        if (this.almacen.get(index: i).getCodigo().equals(anObject: codigo)) {
            //retornamos la posición (i)
            return this.almacen.get(index: i);
        }
    }
    //sino pues retornamos null (El producto no existe)
    return null;
}
```

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

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

- 1- Un título con el nombre del producto de la posición (i) + salto de línea
- 2- El código de la posición (i) + salto de línea
- 3- El nombre de la posición (i) + salto de línea
- 4- El cantidad de la posición (i) + salto de línea
- 5- El descripción de la posición (i) + salto de línea

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

```
public String mostrarListado() {
    String texto = "";
    //Desde la posición 0 hasta el tamaño del arraylist datos recoge la posición (i)
    for (int i = 0; i < almacen.size(); i++) {
        //Almacena en texto la posición (i) y cogiendo cada campo con su get.
        texto = texto + "--- PRODUCTO " + (almacen.get(index: i).getNombre()).toUpperCase() + " ---" + System.lineSeparator();
        texto = texto + "->Código: " + almacen.get(index: i).getCodigo() + System.lineSeparator();
        texto = texto + "->Nombre: " + almacen.get(index: i).getNombre() + System.lineSeparator();
        texto = texto + "->Cantidad: " + almacen.get(index: i).getCantidad() + System.lineSeparator();
        texto = texto + "->Descripción: " + almacen.get(index: i).getDescripcion() + System.lineSeparator();
    }
    return texto;
}
```

7. 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 creamos objetos dentro del método de un botón en el JFrame este será llamado.

```
public Fichero() {
    try {
        1 File fichero = new File(pathname: URL);
        //Fichero de Lectura.
        //Fichero binario Lectura.
        2 FileInputStream dos = new FileInputStream(file: fichero);
        DataInputStream dis = new DataInputStream(in: dos);

        //Si el fichero no existe lo creas
        3 if (!fichero.exists()) {
            fichero.createNewFile();
        }

        //Creamos un objeto de la clase Producto
        4 Producto prodAux = new Producto();
        //vamos a establecer en prodAux el codigo -> lo asignamos a la variable codigo mediante el set
        5 prodAux.setCodigo(codigo: dis.readUTF());

        //Si matricula != nulo... asigna el segundo componente a 'Nombre' y añade el objeto al array
        while (prodAux.getCodigo() != null) {
            6 prodAux.setNombre(nombre: dis.readUTF());
            prodAux.setCantidad(cantidad: dis.readUTF());
            prodAux.setDescripcion(descripcion: dis.readUTF());
            almacen.add(e: prodAux);

            7 //Limpiamos el objeto prodAux para volver a tenerlo vacío
            prodAux = new Producto();

            8 //Obtenemos de nuevo 1 codigo y volvemos al while
            prodAux.setCodigo(codigo: dis.readUTF());
        }
        dis.close();
    } catch (IOException e) {
    }
}
```

//Creamos la ruta con una constante

```
final private String URL = "productos.dat";
```

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 código 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 línea del fichero y almacenamos el código mediante setCodigo

Esto es repetido hasta que código sea nulo (no exista más código 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(código, nombre, cantidad, descripción)**

```
public void insertarProducto(String código, String nombre, String cantidad, String descripción) {
    boolean error = false;
    String msg = null;
    //Creo un producto auxiliar
    Producto nuevoProducto = new Producto();
    nuevoProducto.setCodigo(código);
    nuevoProducto.setNombre(nombre);
    nuevoProducto.setCantidad(cantidad);
    nuevoProducto.setDescripción(descripción);
    //Desde la posición 0 hasta el tamaño del arraylist datos recoge la posición (i)
    for (int i = 0; i < this.almacen.size(); i++) {
        if (this.almacen.get(i).getCodigo().equals(código)) {
            //Guardamos la posición i en índice, existe = true y cortamos el for.
            error = true;
            break;
        }
    }
    //Si existe error es true entonces va a mandar el siguiente error
    if (error) {
        msg = "Este código de producto ya existe";
    }
    if (msg != null) {
        ErrorshowDialog(msg);
        return;
    }
    //Control de posibles errores, si cualquier textField está vacío... entra a su if, guarda la cadena de texto en la variable msgError
    if (código.equals("") & nombre.equals("") & cantidad.equals("") & descripción.equals("")) {
        msg = "Te has dejado todos los campos en blanco";
    } else if (código.equals("")) {
        msg = "Te has dejado el CODIGO en blanco";
    } else if (nombre.equals("") || cantidad.equals("") || descripción.equals("")) {
        msg = "Te has dejado nombre, cantidad o descripción en blanco";
    }
    if (msg != null) {
        ErrorshowDialog(msg);
        return;
    }
    //Si no existe Guardo el producto en mi arraylist y return al método guardar() para guardar los datos introducidos en el arraylist en el fichero
    this.almacen.add(nuevoProducto);
    this.guardar();
}
```

1. Se crea un objeto de la clase producto y vamos almacenando en ese objeto con los setter el código, 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 código igual al código 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 a lanzar el método **ErrorshowDialog** con el texto del msg

```
private void ErrorshowDialog(String msg) {  
    JOptionPane.showMessageDialog(parentComponent:null, message:msg,  
        title: "Error", messageType:JOptionPane.ERROR_MESSAGE);  
}
```

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

```
private void ErrorshowDialog(String msg) {  
    JOptionPane.showMessageDialog(parentComponent:null, message:msg,  
        title: "Error", messageType:JOptionPane.ERROR_MESSAGE);  
}
```

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()

```
public void guardar() {  
    try {  
        1 File fichero = new File(pathname:URL);  
        //Fichero binario Lectura.  
        2 FileOutputStream fos = new FileOutputStream(file:fichero);  
        DataOutputStream dos = new DataOutputStream(out:fos);  
        //Desde la posición 0 hasta el tamaño del arraylist datos recoge la posición (i)  
        3 for (int i = 0; i < almacen.size(); i++) {  
            //escribiendo en el fichero cada get  
            dos.writeUTF(str:almacen.get(index: i).getCodigo());  
            dos.writeUTF(str:almacen.get(index: i).getNombre());  
            dos.writeUTF(str:almacen.get(index: i).getCantidad());  
            dos.writeUTF(str:almacen.get(index: i).getDescripcion());  
        }  
        dos.close();  
        //Siempre que el contador sea falso significa que tanto borrar producto como modificar  
        //a la hora de pasar por este método no ejecute el JOptionPane de insertar.  
        4 if (!this.contador) {  
            CorrectoshowDialog(msg:"Se ha insertado correctamente");  
        }  
    } catch (IOException e) {  
        ErrorshowDialog(msg:"Se ha producido un error al insertar el producto");  
    }  
}
```

1. Se prepara la ruta (URL) `//Creamos la ruta con una constante`
`final private String URL = "productos.dat";`
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`, `getDescripcion` 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

```
private void insertarBtActionPerformed(java.awt.event.ActionEvent evt) {  
    //Objeto fichero sin parámetros (Contiene toda la lectura del fichero)  
    /* Agregamos Codigo, nombre, cantidad y descripcion del textField a nuestro array y  
    se guardar en el fichero con el metodo insertarProducto*/  
    Fichero f = new Fichero();  
    f.insertarProducto(codigo: this.codigoTf.getText(), nombre: this.nombreTf.getText(),  
        cantidad: this.cantidadTf.getText(), descripcion: this.descripcionTf.getText());  
    limpiarCampos();  
}
```

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.

```
private void limpiarCampos() {  
    this.codigoTf.setText("");  
    this.nombreTf.setText("");  
    this.cantidadTf.setText("");  
    this.descripcionTf.setText("");  
}
```

Resultado del botón insertar ejecutando el programa

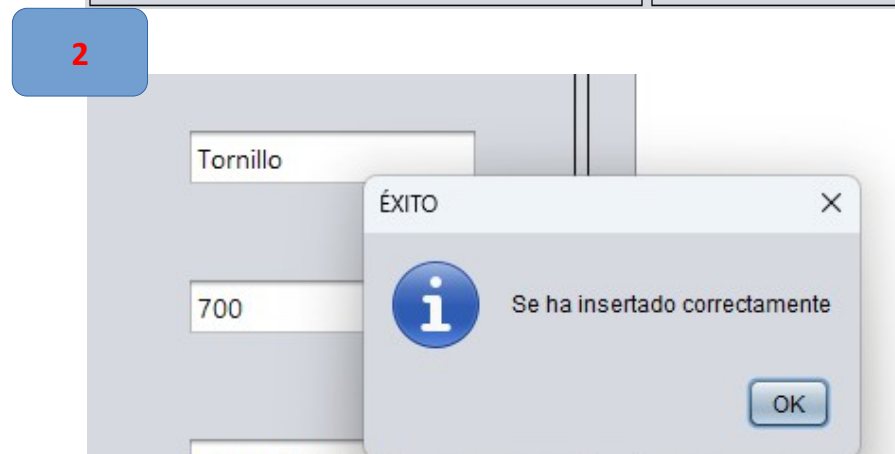
× Insertar producto

1

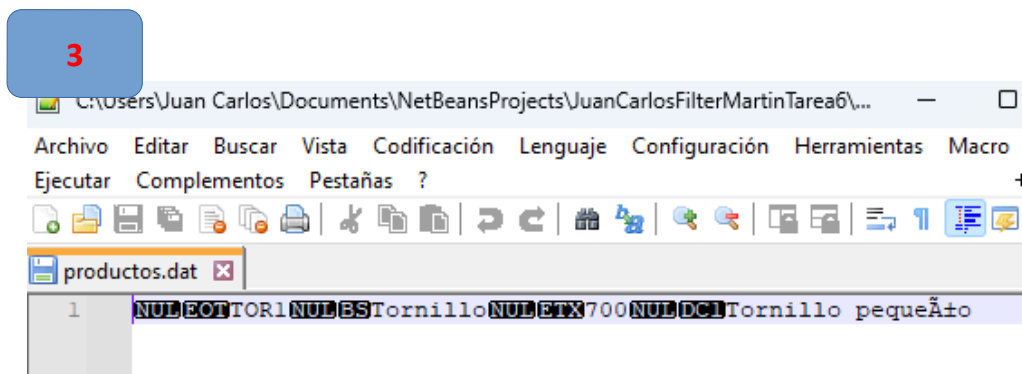
Formulario de Gestión

Código	<input type="text" value="TOR1"/>
Nombre	<input type="text" value="Tornillo"/>
Cantidad	<input type="text" value="700"/>
Descripción	<input type="text" value="Tornillo pequeño"/>

Listado de productos



× Comprobación en el fichero



8. 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)`

```
public Producto buscar(String codigo) {  
    //Desde la posicion 0 hasta el tamaño del arraylist datos recoge la posicion (i)  
    for (int i = 0; i < this.almacen.size(); i++) {  
        //Comparamos si en (i) getCodigo es igual a codigo  
        if (this.almacen.get(index: i).getCodigo().equals(anObject: codigo)) {  
            //retornamos la posicion (i)  
            return this.almacen.get(index: i);  
        }  
    }  
    //sino pues retornamos null (El producto no existe)  
    return null;  
}
```

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

```
Fichero f = new Fichero();
```

Y realizaremos los siguientes pasos:

```
private void buscarActionPerformed(java.awt.event.ActionEvent evt) {  
    //Objeto fichero sin parámetros (Contiene toda la lectura del fichero)  
    //buscamos la matricula escrita en el TextField con el metodo buscar de la clase Fichero*/  
1 Fichero f = new Fichero();  
    //Preguntamos y almacenamos la respuesta en la variable, segun indiquemos se realizará una u otra cosa  
    int question = f.PreguntashowDialog(msg:"¿Como quieres buscar el código?",option1:"Completo", option2:"Por coincidencias");  
    if (question == 0) {  
        Producto productoEncontrado = f.buscar(codigo: this.codigoTf.getText());  
        //Si cocheEncontrado existe  
2 if (productoEncontrado != null) {  
            //coge el codigo, nombre, cantidad, descripcion y lo pinta en el TextArea con el toString  
            //coge el nombre, cantidad, descripcion y lo pinta en sus TextField  
            this.listadoTextArea.setText("---- SE HA ENCONTRADO EL SIGUIENTE PRODUCTO ----" + System.lineSeparator()  
                + productoEncontrado.toString());  
3  
            this.nombreTf.setText(productoEncontrado.getNombre());  
            this.cantidadTf.setText(productoEncontrado.getCantidad());  
            this.descripcionTf.setText(productoEncontrado.getDescripcion());  
4  
        } else {  
            //Si no existe deja vacio ambos TextField y en el TextArea indica que no se ha encontrado.  
            limpiarCampos();  
            this.listadoTextArea.setText("NO SE HA ENCONTRADO EL PRODUCTO INDICADO");  
        }  
    }  
}
```

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()`

```
private void limpiarCampos() {  
    this.codigoTf.setText(t: "");  
    this.nombreTf.setText(t: "");  
    this.cantidadTf.setText(t: "");  
    this.descripcionTf.setText(t: "");  
}
```

Resultado del botón buscar ejecutando el programa

× Buscar producto

1

Formulario de Gestión

Codigo	<input type="text" value="TOR1"/>
Nombre	<input type="text"/>
Cantidad	<input type="text"/>
Descripción	<input type="text"/>

Listado de productos:

2

Formulario de Gestión

Codigo	<input type="text" value="TOR1"/>
Nombre	<input type="text" value="Tornillo"/>
Cantidad	<input type="text" value="700"/>
Descripción	<input type="text" value="Tornillo pequeño"/>

Listado de productos

----- SE HA ENCONTRADO EL SIGUIENTE PRODUCTO -----
->Codigo: TOR1
->Nombre: Tornillo
->Cantidad: 700
->Descripción: Tornillo pequeño

Buscar por Coincidencias

Para buscar productos con el código 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:

```
public ArrayList<Producto> findCoincidencia(String codigo) {  
1 //Se crea un arrayAuxiliar para guardar las coincidencias  
  ArrayList<Producto> arrayCoincidencia = new ArrayList<>();  
2 // Búsqueda por cualquier coincidencia que contenga código  
  Pattern pattern = Pattern.compile(".*" + codigo + ".*");  
  
  // Recorrer desde la posición 0 hasta el tamaño del ArrayList y se  
3 for (int i = 0; i < this.almacen.size(); i++) {  
    /*va recogiendo el código en la posición (i) del array y este se  
    busca una coincidencia del pattern creado con el matcher */  
4    Matcher matcher = pattern.matcher(input: this.almacen.get(index: i).getCodigo());  
  
    if (matcher.find()) {  
      /*se va almacenando cada posición que contenga coincidencia  
      con el código en el array Auxiliar*/  
      arrayCoincidencia.add(e: this.almacen.get(index: i));  
    }  
5 }  
  //al llamar a este método retornamos el array  
  return arrayCoincidencia;  
}
```

1. Se crea un `ArrayList` auxiliar llamado `arrayCoincidencia` para almacenar en este las coincidencias encontradas más adelante.
2. vamos a recoger el código más las concatenaciones de las expresiones regulares y este mediante las clases `Pattern` y `Matcher` 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 código recogido mediante el array almacén será añadido al `arrayCoincidencia` y esto será repetido hasta que termine el bucle.
5. Por último 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 int question = f.PreguntashowDialog(msg:"¿Como quieres buscar el código?",option1:"Completo", option2:"Por coincidencias");
if (question == 0) {
    Producto productoEncontrado = f.buscar(codigo: this.codigoTf.getText());
    //Si cocheEncontrado existe
    if (productoEncontrado != null) {
        //coge el codigo, nombre, cantidad, descripcion y lo pinta en el TextArea con el toString
        //coge el nombre, cantidad, descripcion y lo pinta en sus TextField
        this.listadoTextArea.setText("---- SE HA ENCONTRADO EL SIGUIENTE PRODUCTO ----" + System.lineSeparator()
            + productoEncontrado.toString());

        this.nombreTf.setText(t: productoEncontrado.getNombre());
        this.cantidadTf.setText(t: productoEncontrado.getCantidad());
        this.descripcionTf.setText(t: productoEncontrado.getDescripcion());
    } else {
        //Si no existe deja vacio ambos TextField y en el TextArea indica que no se ha encontrado.
        limpiarCampos();
        this.listadoTextArea.setText(t: "NO SE HA ENCONTRADO EL PRODUCTO INDICADO");
    }
}
```

1. Lo que se va a realizar en primer lugar es preguntar al usuario si quiere que la búsqueda 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.

```
} else {
    //ArrayList de producto que va a almacenar lo que se recoja en el metodo findCoincidencia
1 ArrayList<Producto> arrayCoincidencia = f.findCoincidencia(codigo: this.codigoTf.getText());
    //Se comprueba que el campo textFiel y el array obtenido del método findCoincidencia no esté vacio
2 if (!this.codigoTf.getText().isEmpty()) {
    if (!arrayCoincidencia.isEmpty()) {
        /*Si no está vacío se reestablece el TextArea y se va recorriendo
        el array pintando cada producto con coincidencia mediante el for*/
3 this.listadoTextArea.setText(t: "");
4 this.listadoTextArea.setText(t: " ---- LAS COINCIDENCIAS SON LAS SIGUIENTES ----\n");
        for (int i = 0; i < arrayCoincidencia.size(); i++) {
            this.listadoTextArea.append(t: arrayCoincidencia.get(index: i).toString());
        }
        //Por ultimo si nos e entra en sus respectivos if se mandará un mensaje de error
5 } else {
        this.listadoTextArea.setText("NO HAY COINCIDENCIA PARA EL CÓDIGO: " + this.codigoTf.getText());
    }
} else {
    this.listadoTextArea.setText(t: "INTRODUCE ALMENOS UN CODIGO PARA BUSCAR");
}
}
```

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 código 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 Título
4. En este punto se va a entrar a un bucle for en el que se va a recorrer desde la posición 0 hasta el tamaño del array pintando en cada vuelta cada producto del array
5. Por último 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

The screenshot shows a Java Swing window with a search interface on the left and a product list on the right. A red box with the number '1' highlights the search button. The search interface includes four empty text fields and a button labeled 'BUSCAR'. The product list, titled 'Listado de productos', displays a list of products with their details. The list is as follows:

PRODUCTO	Código	Nombre	Cantidad	Descripción
TUERCA	RU02	Tuerca	69	Tuerca para tornillo pequeño
CLAVO	CL001	Clavo	100	Clavo de cabeza plana
DESTORNILLADOR	RU01	Destornillador	30	Destornillador de estrella
DESTORNILLADOR	03RU	Destornillador	20	Destornillador Plano

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

2

The screenshot shows a search form with fields for 'Codigo', 'Nombre', 'Cant', and 'Descripción'. The 'Codigo' field contains 'RU'. A confirmation dialog box titled 'Ventana de confirmación' is overlaid on the form. The dialog contains a question mark icon and the text '¿Como quieres buscar el código?'. There are two buttons: 'Completo' and 'Por coincidencias'. The 'Por coincidencias' button is highlighted. At the bottom of the form are two buttons: 'INSERTAR' and 'BUSCAR'.

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

3

The screenshot shows the search results page. On the left is a form with fields for 'Codigo', 'Nombre', 'Cantidad', and 'Descripción'. The 'Codigo' field contains 'RU'. On the right is a section titled 'Listado de productos'. It contains the text '--- LAS COINCIDENCIAS SON LAS SIGUIENTES ---' followed by three product entries:

- >Codigo: RU02
- >Nombre: Tuerca
- >Cantidad: 69
- >Descripción: Tuerca para tornillo pequeño
- >Codigo: RU01
- >Nombre: Destornillador
- >Cantidad: 30
- >Descripción: Destornillador de estrella
- >Codigo: 03RU
- >Nombre: Destornillador
- >Cantidad: 20
- >Descripción: Destornillador Plano

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)`

```
public void borrarProducto(String codigo) {  
1  boolean existe = false;  
2  int question;  
3  int indice = 0;  
   //Desde la posicion 0 hasta el tamaño del arraylist datos recoge la posicion (i)  
4  for (int i = 0; i < this.almacen.size(); i++) {  
   //Comparamos si en (i) getCodigo es igual a codigo  
   if (this.almacen.get(index: i).getCodigo().equals(anObject: codigo)) {  
3     //Guardamos la posicion i en indice, existe = true y cortamos el for.  
     indice = i;  
     existe = true;  
     break;  
   }  
   }  
   //Si existe va a preguntar si desea borrarlo  
5  if (existe) {  
   question = PreguntarshowDialog(msg: "¿Estás seguro de que deseas borrar este producto?");  
   /*Si es lo confirma entonces elimina del arraylist la posicion que vale indice y  
   guardamos la modificacion con el metodo guardar*/  
6   if (question == 0) {  
    this.contador = true;  
    this.almacen.remove(index: indice);  
    this.guardar();  
    CorrectoshowDialog(msg: "Se ha borrado correctamente");  
   } else {  
    ErrorshowDialog(msg: "Ha ocurrido un error al borrar este producto");  
   }  
   }  
}
```

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()

```
public void guardar() {  
    try {  
        1 File fichero = new File(pathname: URL);  
        //Fichero binario Lectura.  
        2 FileOutputStream fos = new FileOutputStream(file: fichero);  
        DataOutputStream dos = new DataOutputStream(out: fos);  
        //Desde la posicion 0 hasta el tamaño del arraylist datos recoge la posicion (i)  
        3 for (int i = 0; i < almacen.size(); i++) {  
            //escribiendo en el fichero cada get  
            dos.writeUTF(str: almacen.get(index: i).getCodigo());  
            dos.writeUTF(str: almacen.get(index: i).getNombre());  
            dos.writeUTF(str: almacen.get(index: i).getCantidad());  
            dos.writeUTF(str: almacen.get(index: i).getDescripcion());  
        }  
        dos.close();  
        //Siempre que el contador sea falso significa que tanto borrar producto como modificar  
        //a la hora de pasar por este método no ejecute el JoptionPane de insertar.  
        4 if (!this.contador) {  
            CorrectoshowDialog(msg: "Se ha insertado correctamente");  
        }  
    } catch (IOException e) {  
        ErrorshowDialog(msg: "Se ha producido un error al insertar el producto");  
    }  
}
```

//Creamos la ruta con una constante

```
final private String URL = "productos.dat";
```

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, getDescripcion 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

```
private void borrarBtActionPerformed(java.awt.event.ActionEvent evt) {  
    //Objeto fichero sin parámetros (Contiene toda la lectura del fichero)  
    /*eliminamos el producto relacionado con el codigo del TextFiel si  
    existe con el metodo borrarProducto de la clase Fichero*/  
    Fichero f = new Fichero();  
    f.borrarProducto(codigo: this.codigoTf.getText());  
    limpiarCampos();  
}
```

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 código introducido en el textField.

Por último respecto a este botón 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.

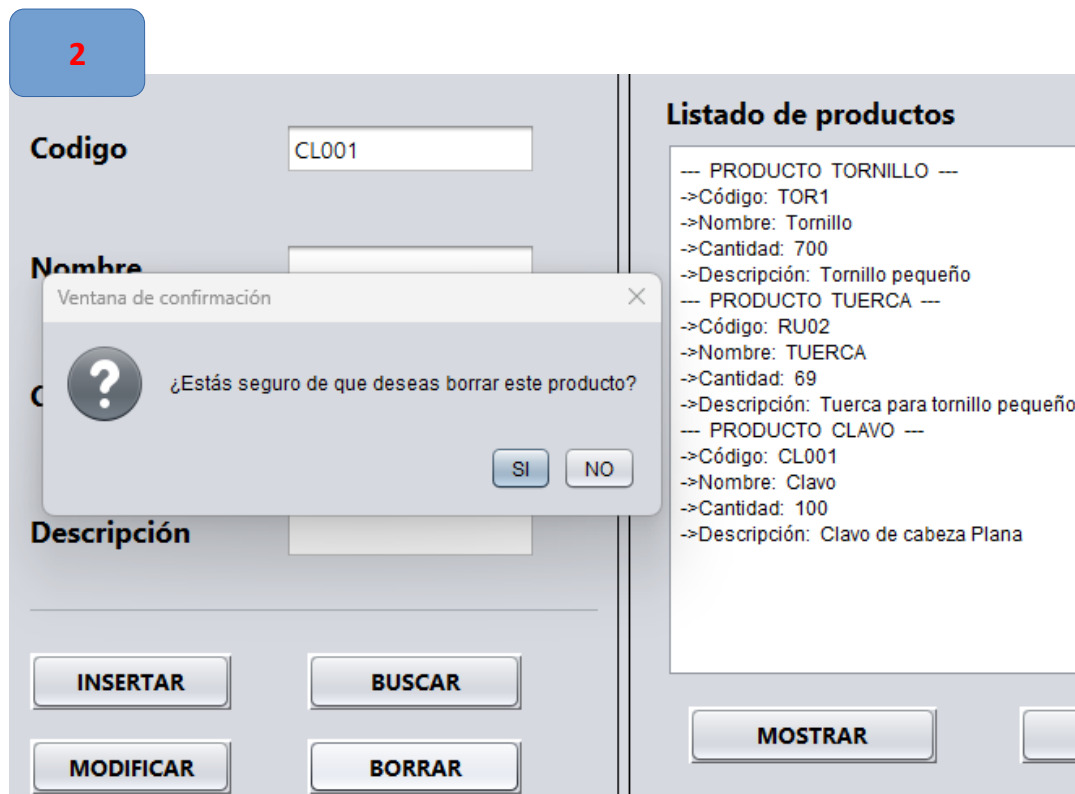
Para ello vamos a introducir el código de este producto en su campo TextField

The screenshot shows a Java Swing application window. On the left is a form with four text fields labeled 'Codigo', 'Nombre', 'Cantidad', and 'Descripción'. A blue box with the number '1' is positioned over the 'Codigo' field. Below the fields are four buttons: 'INSERTAR', 'BUSCAR', 'MODIFICAR', and 'BORRAR'. On the right is a panel titled 'Listado de productos' containing a text area with the following text:

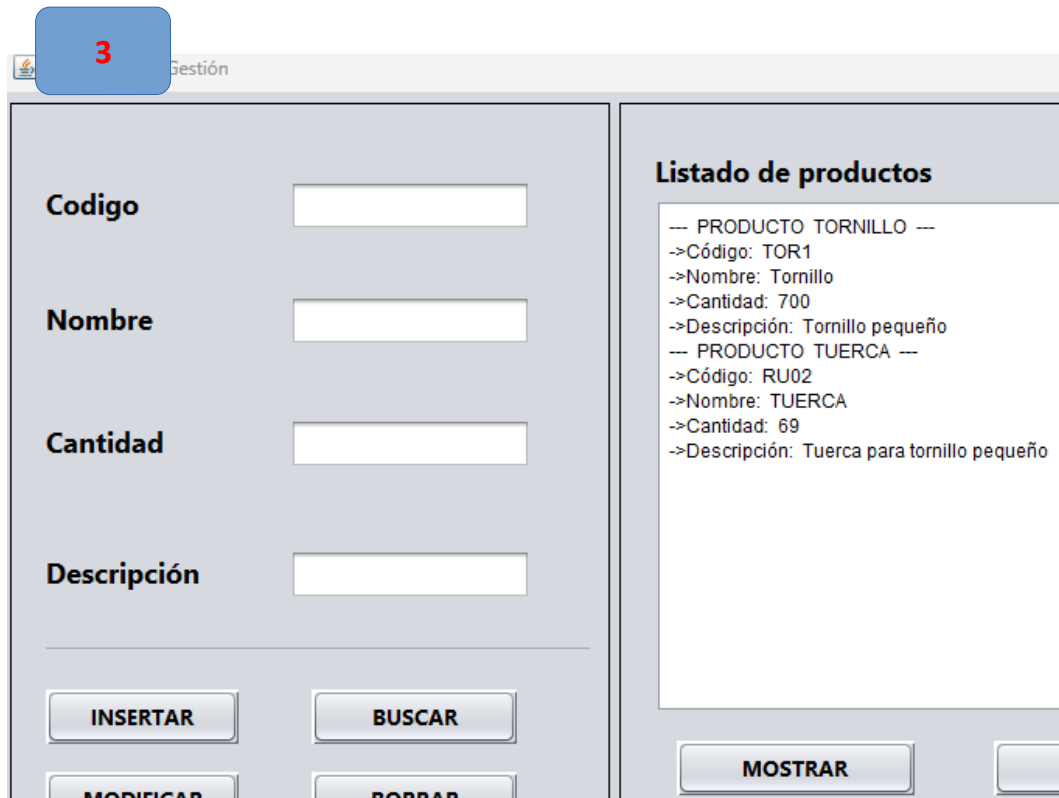
```
--- PRODUCTO TORNILLO ---  
->Código: TOR1  
->Nombre: Tornillo  
->Cantidad: 700  
->Descripción: Tornillo pequeño  
--- PRODUCTO TUERCA ---  
->Código: RU02  
->Nombre: TUERCA  
->Cantidad: 69  
->Descripción: Tuerca para tornillo pequeño  
--- PRODUCTO CLAVO ---  
->Código: CL001  
->Nombre: Clavo  
->Cantidad: 100  
->Descripción: Clavo de cabeza Plana
```

At the bottom right of the 'Listado de productos' panel are two buttons: 'MOSTRAR' and 'SALIR'.

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 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)`

```
public void modificarProducto(String codigo, String nombre, String cantidad, String descripcion) {  
1  boolean modificado = false;  
    int indice = 0;  
  
    //Desde la posicion 0 hasta el tamaño del arraylist datos recoge la posicion (i)  
2  for (int i = 0; i < this.almacen.size(); i++) {  
        //Comparamos si en (i) getCodigo es igual a codigo  
3      if (this.almacen.get(index: i).getCodigo().equals(anObject: codigo)) {  
            //Guardamos la posicion i en indice, existe = true y cortamos el for.  
            indice = i;  
            modificado = true;  
            break;  
        }  
    }  
  
    /*Si modificado = true vamos a obtener el indice y con el setNombre, setCantidad  
    y setDescription vamos a establecer los nuevos datos*/  
4  if (modificado) {  
        this.contador = true;  
        this.almacen.get(index: indice).setNombre(nombre);  
        this.almacen.get(index: indice).setCantidad(cantidad);  
        this.almacen.get(index: indice).setDescription(descripcion);  
5        this.guardar();  
        Correcto.showMessageDialog(msg: "El Producto se ha modificado correctamente");  
    } else {  
        Error.showMessageDialog(msg: "Ha ocurrido un error al intentar modificar el producto");  
    }  
}
```

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 código pasado por parámetro que sea igual al del array con la posición (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()

```
public void guardar() {  
    try {  
1      File fichero = new File(pathname: URL);  
        //Fichero binario Lectura.  
2      FileOutputStream fos = new FileOutputStream(file: fichero);  
        DataOutputStream dos = new DataOutputStream(out: fos);  
        //Desde la posicion 0 hasta el tamaño del arraylist datos recoge la posicion (i)  
3      for (int i = 0; i < almacen.size(); i++) {  
            //escribiendo en el fichero cada get  
            dos.writeUTF(str: almacen.get(index: i).getCodigo());  
            dos.writeUTF(str: almacen.get(index: i).getNombre());  
            dos.writeUTF(str: almacen.get(index: i).getCantidad());  
            dos.writeUTF(str: almacen.get(index: i).getDescripcion());  
        }  
        dos.close();  
        //Siempre que el contador sea falso significa que tanto borrar producto como modificar  
        //a la hora de pasar por este método no ejecute el JOptionPane de insertar.  
4      if (!this.contador) {  
            CorrectoshowDialog(msg: "Se ha insertado correctamente");  
        }  
    } catch (IOException e) {  
        ErrorshowDialog(msg: "Se ha producido un error al insertar el producto");  
    }  
}
```

//Creamos la ruta con una constante

1. Se prepara la ruta (URL) `final private String URL = "productos.dat";`

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, getDescpcion 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

```
private void modificarBtActionPerformed(java.awt.event.ActionEvent evt) {  
    /*Objeto fichero sin parámetros (Contiene toda la lectura del fichero)  
    modificamos los nuevos textField indicados siempre que se cumpla que el  
    codigoTf ya exista*/  
    Fichero f = new Fichero();  
    f.modificarProducto(codigo: this.codigoTf.getText(), nombre: this.nombreTf.getText(),  
        cantidad: this.cantidadTf.getText(), descripcion: this.descripcionTf.getText());  
}
```

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

1

Codigo	<input type="text"/>
Nombre	<input type="text"/>
Cantidad	<input type="text"/>
Descripción	<input type="text"/>

Listado de productos
--- PRODUCTO TORNILLO ---
->Código: TOR1
->Nombre: Tornillo
->Cantidad: 700
->Descripción: Tornillo pequeño
--- PRODUCTO TUERCA ---
->Código: RU02
->Nombre: TUERCA
->Cantidad: 69
->Descripción: Tuerca para tornillo pequeño

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

2

Código	<input type="text" value="TOR1"/>
Nombre	<input type="text" value="Destornillador"/>
Cantidad	<input type="text" value="600"/>
Descripción	<input type="text" value="Destornillador de estrel"/>

Listado de productos
--- PRODUCTO TORNILLO ---
->Código: TOR1
->Nombre: Tornillo
->Cantidad: 700
->Descripción: Tornillo pequeño
--- PRODUCTO TUERCA ---
->Código: RU02
->Nombre: TUERCA
->Cantidad: 69
->Descripción: Tuerca para tornillo pequeño


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.

3

Formulario de Gestion

Código	<input type="text" value="TOR1"/>
Nombre	<input type="text" value="Destornillador"/>
Cantidad	<input type="text" value="600"/>
Descripción	<input type="text" value="Destornillador de estrel"/>

ÉXITO

 El Producto se ha modificado correctamente

OK

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

4

Formulario de Gestión

Codigo	<input type="text" value="TOR1"/>
Nombre	<input type="text" value="Destornillador"/>
Cantidad	<input type="text" value="600"/>
Descripción	<input type="text" value="Destornillador de estrel"/>

Listado de productos
--- PRODUCTO DESTORNILLADOR ---
->Código: TOR1
->Nombre: Destornillador
->Cantidad: 600
->Descripción: Destornillador de estrella
--- PRODUCTO TUERCA ---
->Código: RU02
->Nombre: TUERCA
->Cantidad: 69
->Descripción: Tuerca para tornillo pequeño

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 leído.

```
public String mostrarListado() {  
1  String texto = "";  
  //Desde la posicion 0 hasta el tamaño del arraylist datos recoge la posicion (i)  
2  for (int i = 0; i < almacen.size(); i++) {  
3    //Almacena en texto la posicion (i) y cogiendo cada campo con su get.  
    texto = texto + "---- PRODUCTO " + (almacen.get(index: i).getNombre()).toUpperCase() + " ----" + System.lineSeparator();  
    texto = texto + "->Código: " + almacen.get(index: i).getCodigo() + System.lineSeparator();  
    texto = texto + "->Nombre: " + almacen.get(index: i).getNombre() + System.lineSeparator();  
    texto = texto + "->Cantidad: " + almacen.get(index: i).getCantidad() + System.lineSeparator();  
    texto = texto + "->Descripción: " + almacen.get(index: i).getDescripcion() + System.lineSeparator();  
  }  
  return texto;  
}
```

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 título del producto recogiendo el nombre con el setNombre seguido de un salto de línea.

Lo mismo para el resto de línea, 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

```
private void mostrarBtActionPerformed(java.awt.event.ActionEvent evt) {  
    //Objeto fichero sin parámetros (Contiene toda la lectura del fichero)  
    //Mostramos en el listadoTa el texto obtenido en el método listado de la clase fichero*/  
    Fichero f = new Fichero();  
    this.listadoTextArea.setText( f.mostrarListado());  
}
```

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

1

Codigo

Nombre

Cantidad

Descripción

INSERTAR BUSCAR

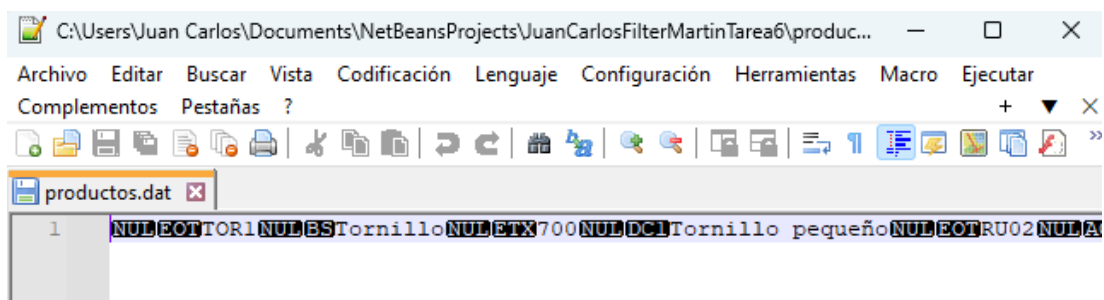
MODIFICAR BORRAR

Listado de productos

--- PRODUCTO TORNILLO ---
->Código: TOR1
->Nombre: Tornillo
->Cantidad: 700
->Descripción: Tornillo pequeño
--- PRODUCTO TUERCA ---
->Código: RU02
->Nombre: TUERCA
->Cantidad: 69
->Descripción: Tuerca para tornillo pequeño
--- PRODUCTO CLAVO ---
->Código: CL001
->Nombre: Clavo
->Cantidad: 100
->Descripción: Clavo de cabeza plana
--- PRODUCTO ALAMBRE ---
->Código: AL45
->Nombre: Alambre
->Cantidad: 300
->Descripción: Rollo de alambre

MOSTRAR SALIR

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);

```
private void salirBtActionPerformed(java.awt.event.ActionEvent evt) {  
    System.exit(status: 0);  
}
```