



**CICLO: [DAM]**  
**MÓDULO DE [PROGRAMACIÓN]**

# **[Tarea N° 05]**

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

## Contenido

<b>1. Documentos que se adjuntan a este informe.....</b>	<b>3</b>
<b>2. RA05_f) Se han utilizado las herramientas del entorno de desarrollo para crear interfaces gráficos de usuario simples.....</b>	<b>3</b>
PANELS:.....	3
LABEL:.....	5
SEPARADOR:.....	6
TEXT FIELD:.....	8
TEXT AREA:.....	9
BUTTON:.....	11
INTERFAZ GRÁFICA.....	12
<b>3. RA05_a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.....</b>	<b>13</b>
La introducción de información van realizar con los JText del formulario:.....	13
Para cada producto deberemos introducir (código, Nombre, Cantidad, Descripción).....	13
<b>4. RA05_d) Se han utilizado ficheros para almacenar y recuperar información.....</b>	<b>14</b>
Fichero de texto llamado 15456141A.txt, guardado en la misma carpeta del proyecto.....	14
[Insertar] podremos introducir los productos que deseen.....	14
[Buscar] el código del producto y si existe nos mostrará todos los datos del producto, si no existe nos dirá que ese producto no existe.....	16
[Listado/Mostrar] nos mostrará todos los productos dados de alta con todos sus datos.....	18
[Salir] cerrará la aplicación.....	19
<b>5. RA05_h) Se han escrito programas que utilicen interfaces gráficos para la entrada y salida de información.....</b>	<b>19</b>
La programación se hará en un entorno gráfico, usando los controles que el alumno considere oportunos ( <i>botones, textbox, listbox, combobox, etc.....</i> ).....	19
<b>6. RA5_g) Se han programado controladores de eventos.....</b>	<b>20</b>
Se programarán los botones de insertar, buscar, mostrar y salir que son los que le darán la funcionalidad a la aplicación.....	20
Insertar.....	20
Buscar.....	23
Mostrar.....	25
Salir.....	26

## 1. Documentos que se adjuntan a este informe.

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

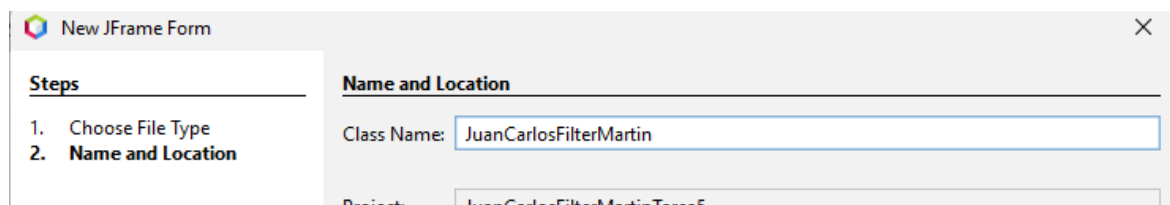
1. Informe de elaboración de la tarea.
2. Proyecto java

## 2. RA05\_f) Se han utilizado las herramientas del entorno de desarrollo para crear interfaces gráficos de usuario simples.

- × Para crear una interfaz primeramente creamos un proyecto
- × Una vez creado:  
→ **Botón derecho sobre el proyecto > new > JFrame form**



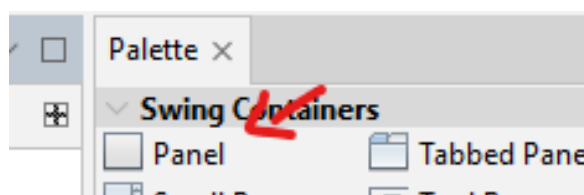
- × Se abrirá el asistente para crear el JFrame Form, en la que le asignamos un nombre



- × Ahora con la interfaz creada vamos a crear visualmente el formulario similar al de la tarea. Añadiendo los **Panel**, **Label**, **Separador**, **Text Field**, **Text Area**, **Button**,

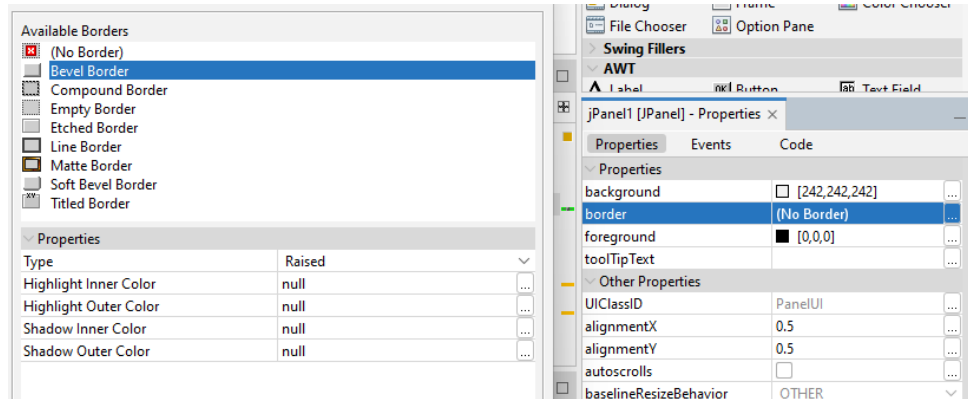
### × PANELS:

→ Para crear 2 paneles tanto a la izquierda como la derecha usamos el contenedor **Panel**.

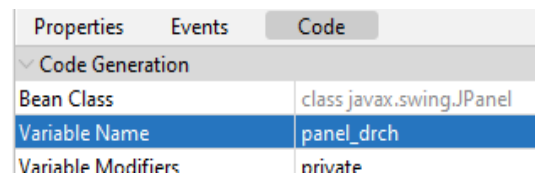
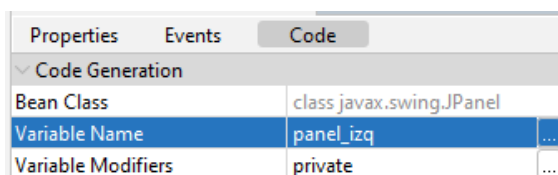


→ Para darle los bordes necesarios en las propiedades podemos cambiarle el tipo de borde

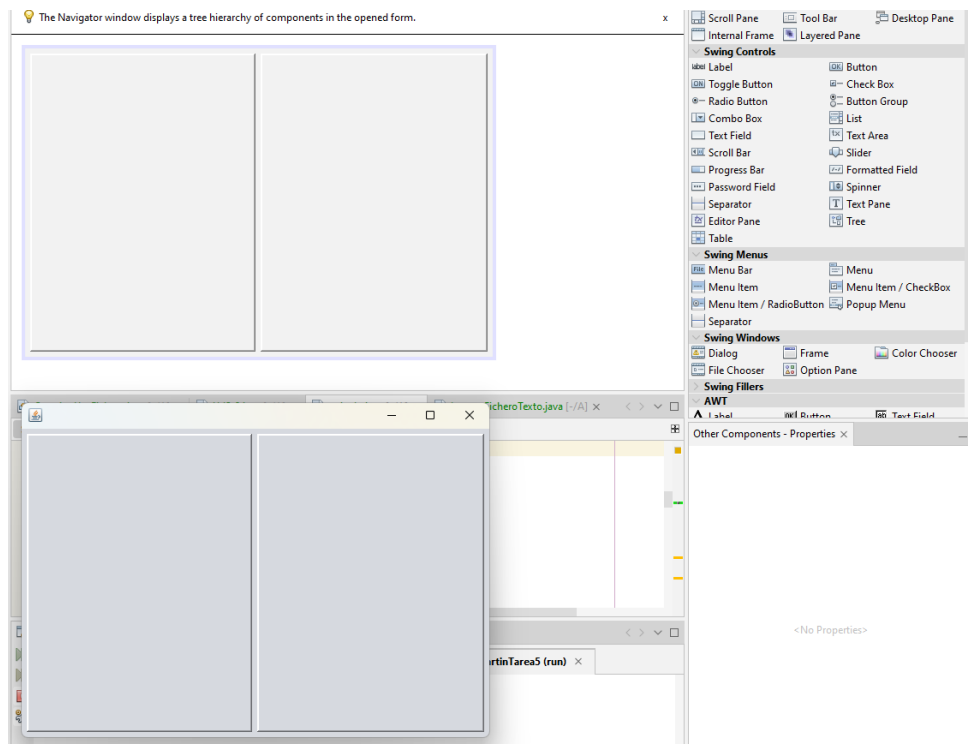
→ Seleccionamos el **border bevel** en ambos paneles



→ Por ultimo podemos cambiar el nombre de las variables a los paneles en la pestaña **Code**:

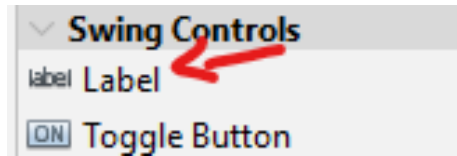


→ **Resultado de los paneles:**

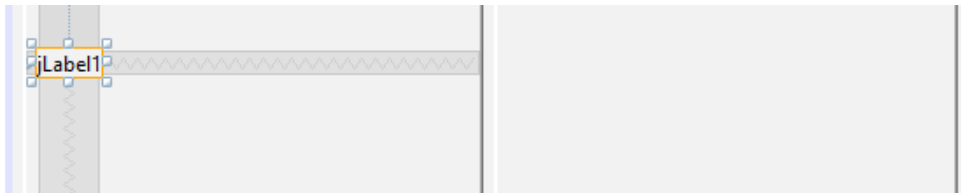


## × LABEL:

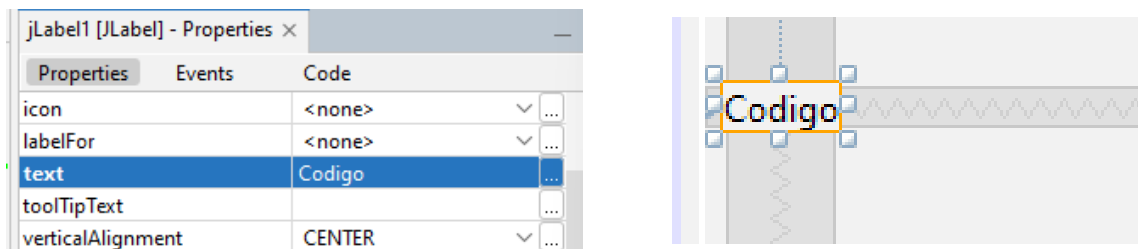
→ Para crear los texto tenemos la opción Label dentro de los controles disponibles.



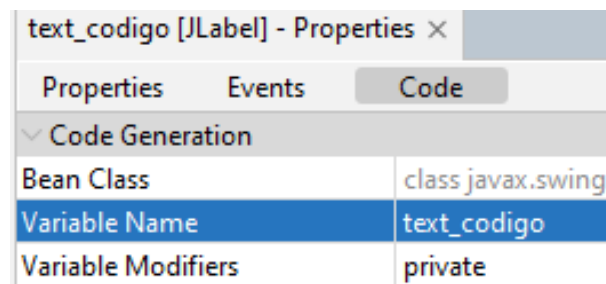
→ Arrastramos el Label donde queramos introducir el texto.



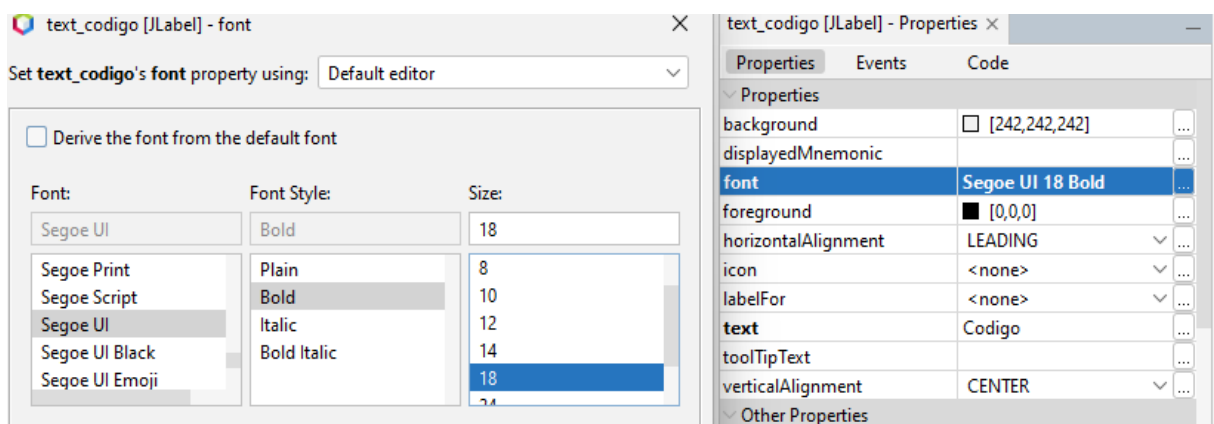
→ Al label le cambiamos el nombre a mostrar desde la pestaña **propiedades**.



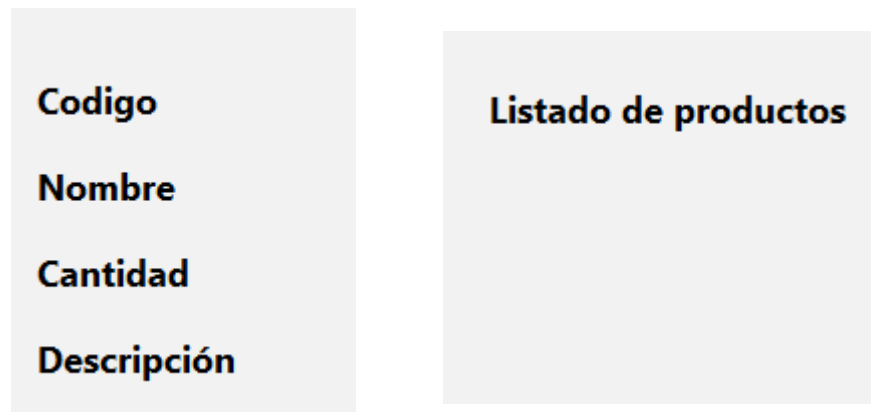
→ Cambiamos también el nombre de la variable que tendrá en nuestro código en la pestaña **Code**.



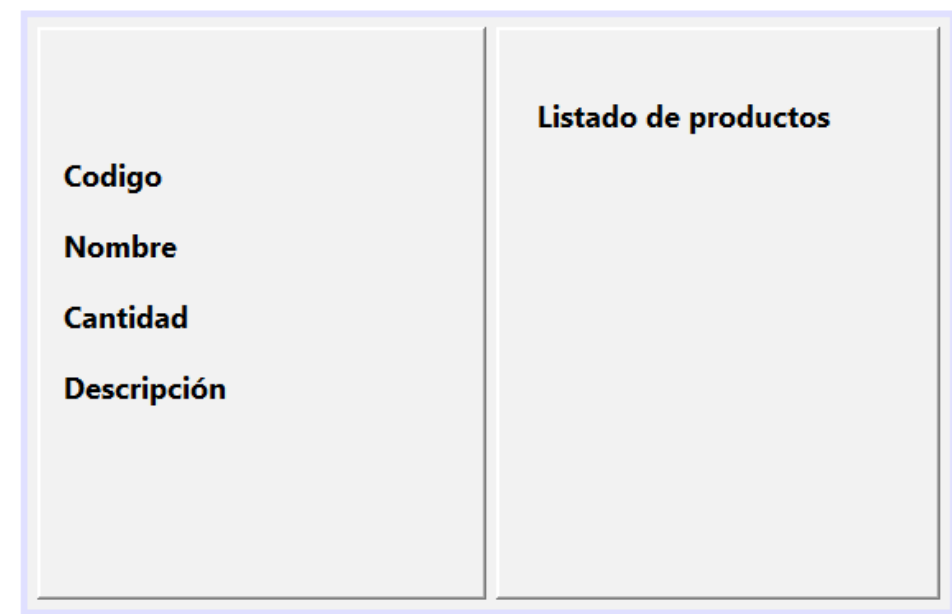
→ También podemos cambiarle el tamaño de texto, negrita, etc desde la pestaña **propiedades > font**



→ Por último tendríamos que crear los demás label como se ha creado el anterior:

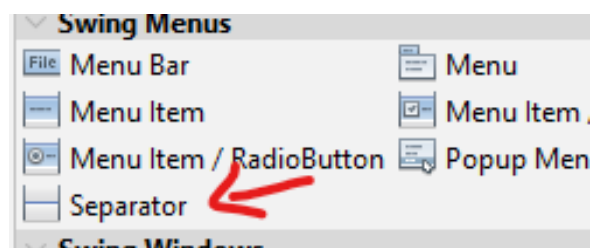


→ **Resultado de los Label:**

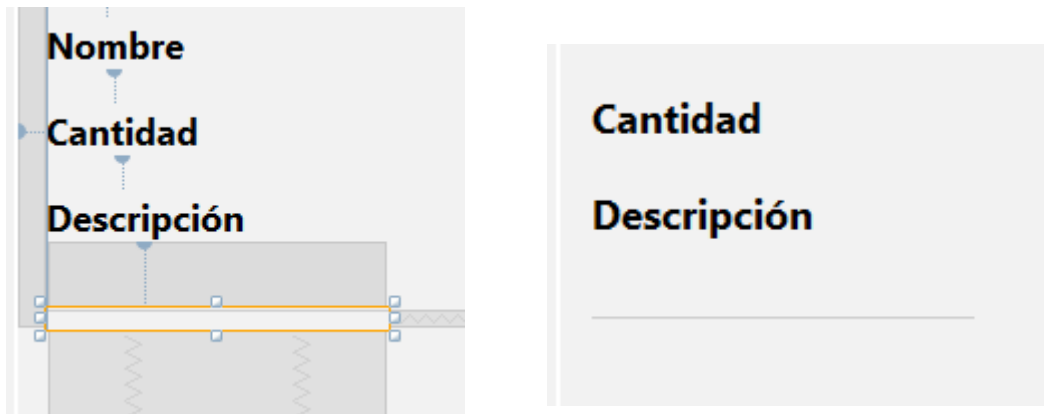


#### × SEPARADOR:

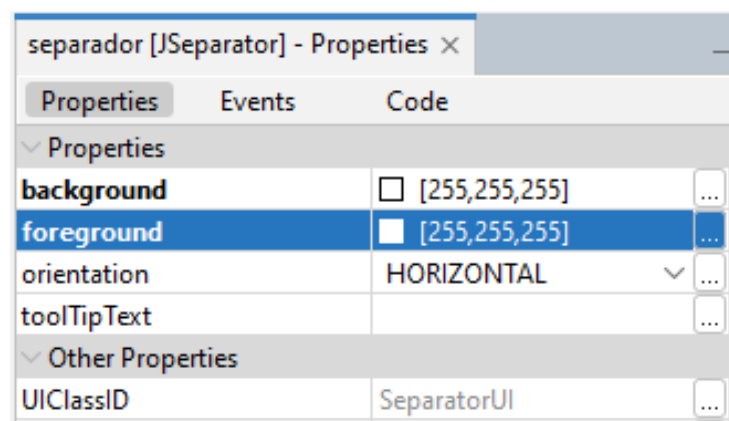
→ Para separar mediante líneas tenemos dentro de **Swing Menus** la opción **separator**



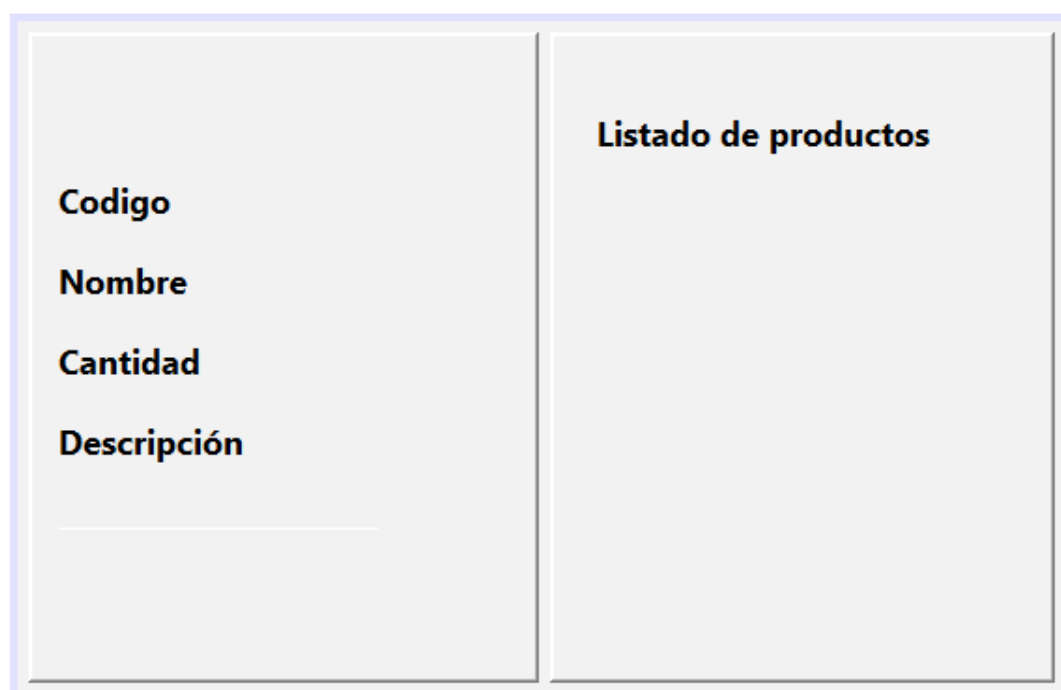
→ Una vez seleccionado podemos arrastrarlo y ponerlo en nuestra interfaz gráfica.



→ Podemos cambiar el color, la orientación, etc. Como puede ser cambiarlo a color blanco.

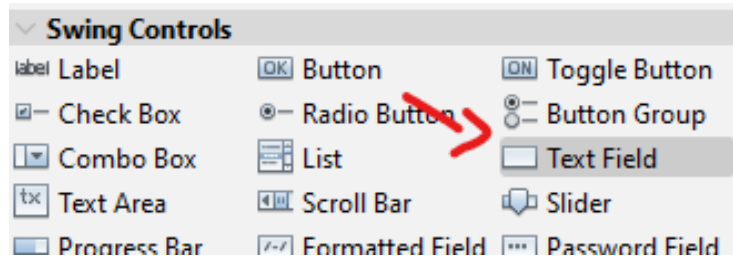


→ Resultado del Separador:



## × TEXT FIELD:

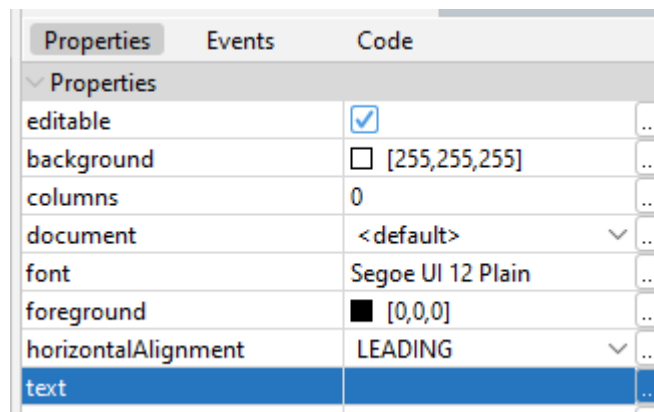
→ Para crear los cuadros pequeños de texto tenemos **Text Field** que se encuentra en **Swing Controls**.



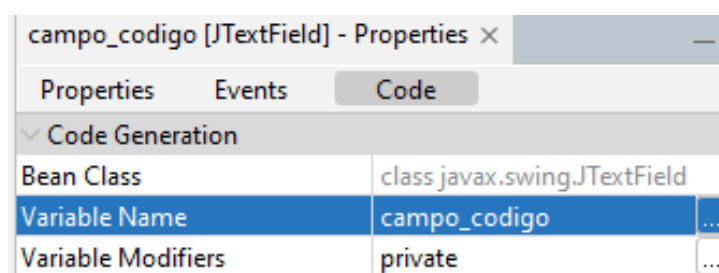
→ Una vez seleccionado podemos mover el cuadro a nuestra interfaz gráfica.



→ Al Text Field para dejarlo sin texto desde la pestaña **propiedades** podemos dejarlo en blanco en el apartado **text**.

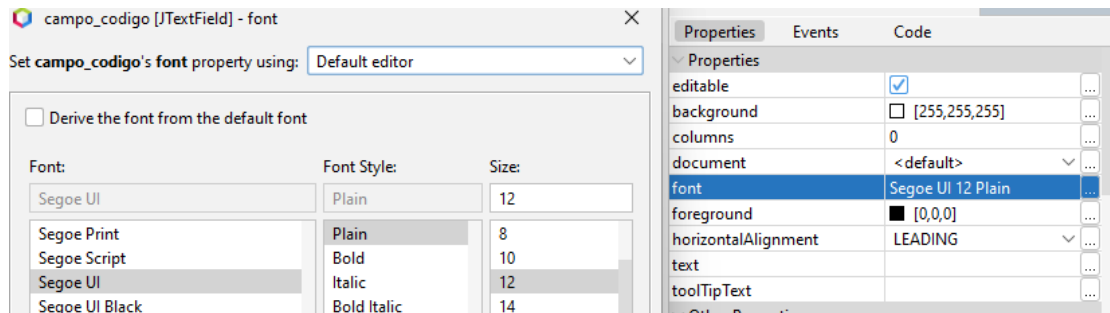


→ Cambiamos también el nombre de la variable que tendrá en nuestro código en la pestaña **Code**.





→ También podemos cambiarle el tipo de texto que vamos a introducir en este cuadro desde la pestaña **propiedades > font**



*Hay muchas propiedades para modificar.*

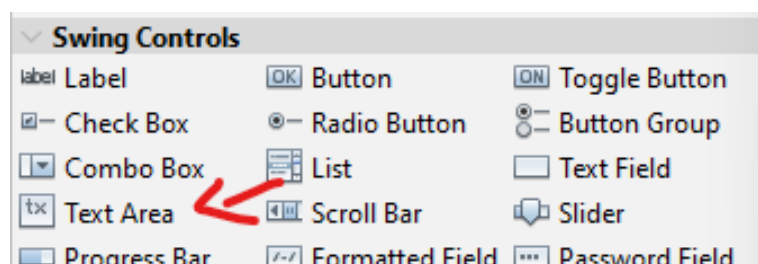
→ Por último tendríamos que crear los demás Text Field como se ha creado el anterior:

→ **Resultado del Text Field:**

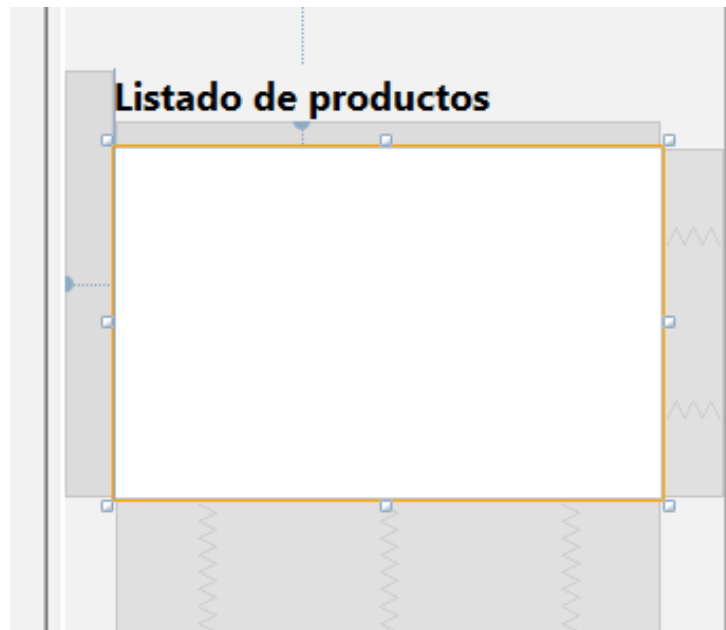
The image shows a light gray rectangular panel containing four text input fields. Each field is preceded by a bold black label: 'Codigo', 'Nombre', 'Cantidad', and 'Descripción'. The fields are empty and have a standard white background with a thin gray border.

## × TEXT AREA:

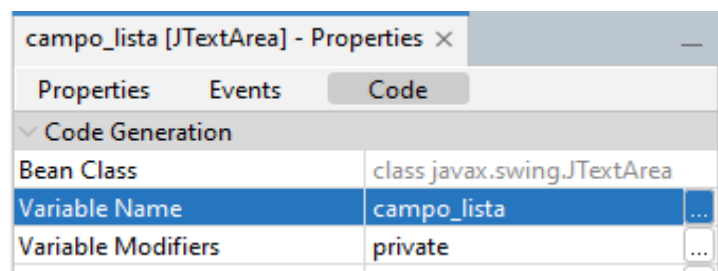
→ Para crear un cuadro de texto y pueda mostrar varias líneas dentro de el tenemos la opción **Text Area** dentro de **Swing Controls**.



→ Una vez seleccionado podemos mover el Text Area a nuestra interfaz gráfica.



→ Cambiamos el nombre de la variable que tendrá en nuestro código en la pestaña **Code**.

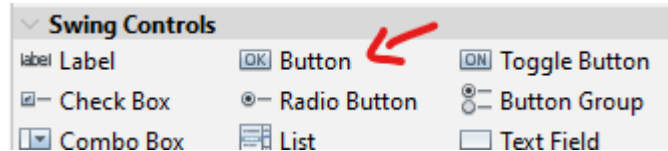


→ **Resultado del Text Area:**

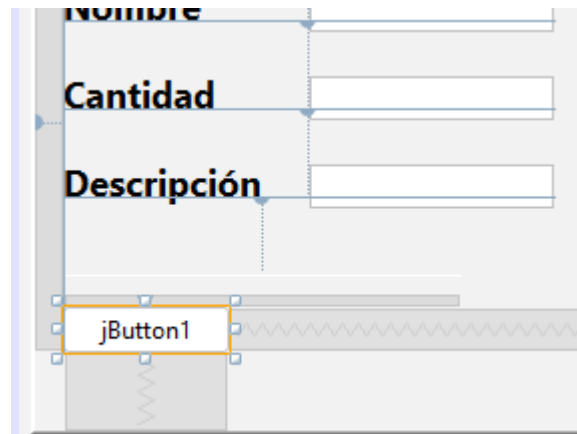
Formulario de Datos		Visualización
<b>Codigo</b>	<input type="text"/>	<div><b>Listado de productos</b></div> <div></div>
<b>Nombre</b>	<input type="text"/>	
<b>Cantidad</b>	<input type="text"/>	
<b>Descripción</b>	<input type="text"/>	

## × **BUTTON:**

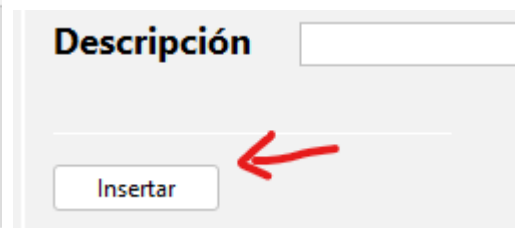
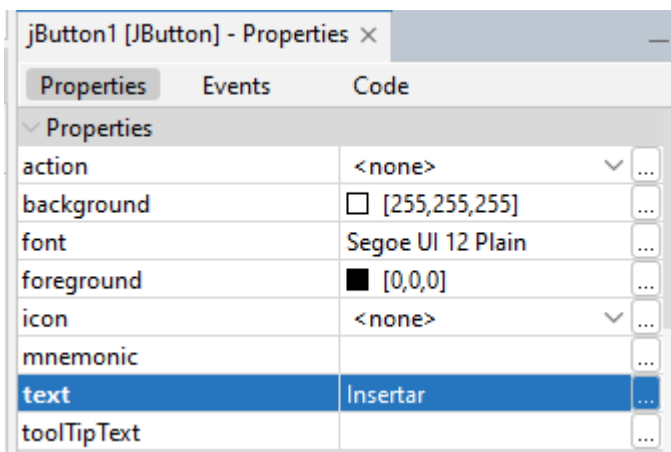
→ Para agregar botones a nuestra interfaz tenemos la opción **Button** dentro de **Swing Controls**.



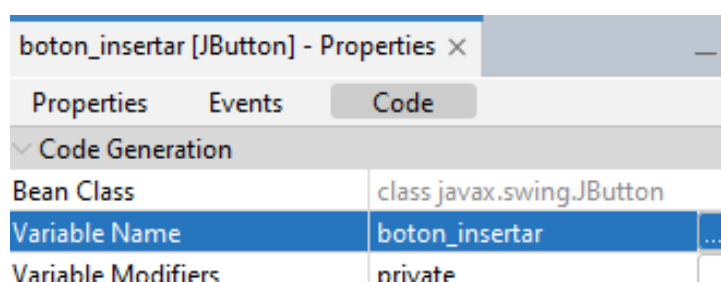
→ Una vez seleccionado podemos mover el botón a nuestra interfaz gráfica.



× → A este botón podemos cambiarle el nombre desde la pestaña propiedades en el apartado text



→ Cambiamos el nombre de la variable que tendrá en nuestro código en la pestaña **Code**.



→ Por último tendríamos que crear los demás botones como se ha creado el anterior:

→ **Resultado del Button:**

The mockup consists of two main panels. The left panel contains four labels with corresponding text input fields: 'Codigo', 'Nombre', 'Cantidad', and 'Descripción'. Below these fields are two buttons labeled 'Insertar' and 'Buscar'. The right panel is titled 'Listado de productos' and contains a large empty rectangular box for displaying a list. Below this box are two buttons labeled 'Mostrar' and 'Salir'.

## × INTERFAZ GRÁFICA

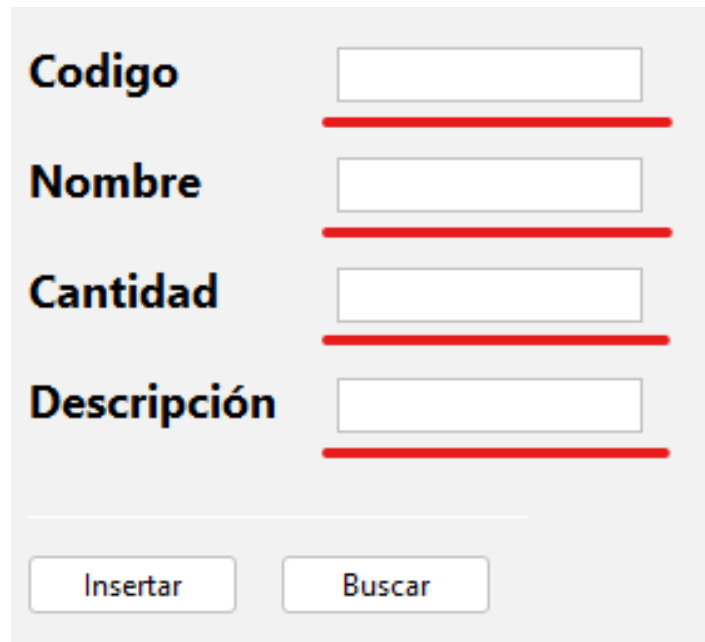
Aquí deajo el ejemplo de interfaz de la tarea a como se ha quedado el resultado final (Solamente el aspecto gráfico):

This is a screenshot of a Windows-style application window titled 'Formulario de Gestión'. It features a standard title bar with minimize, maximize, and close buttons. The window's content is divided into two vertical sections. The left section contains the same input fields and buttons as the mockup. The right section, titled 'Listado de productos', contains a large empty box and 'Mostrar' and 'Salir' buttons. The buttons have a light blue gradient.

This is another screenshot of the same application window, but with a different visual style. The title bar is light blue with standard window controls. The input fields and buttons are now styled with a light gray background and a thin blue border. The 'Mostrar' and 'Salir' buttons in the 'Listado de productos' section also have this gray and blue styling.

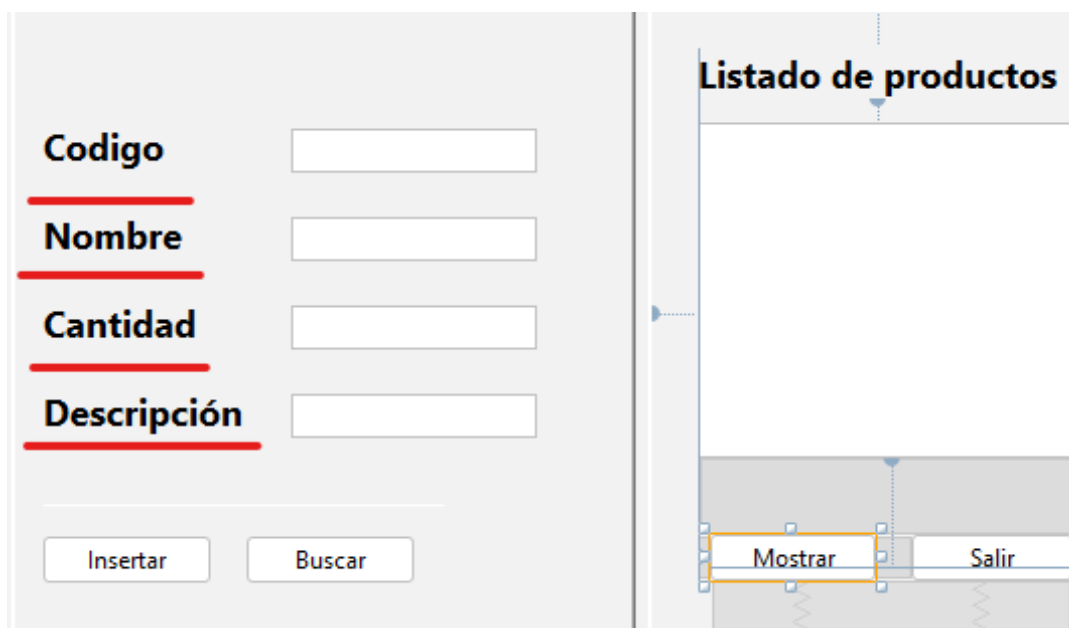
3. RA05\_a) Se ha utilizado la consola para realizar operaciones de entrada y salida de información.

La introducción de información van realizar con los JText del formulario:



Formulario de entrada de datos para un producto. Contiene cuatro campos de texto etiquetados como 'Codigo', 'Nombre', 'Cantidad' y 'Descripción'. Cada campo tiene una línea roja horizontal debajo de él. En la parte inferior hay dos botones: 'Insertar' y 'Buscar'.

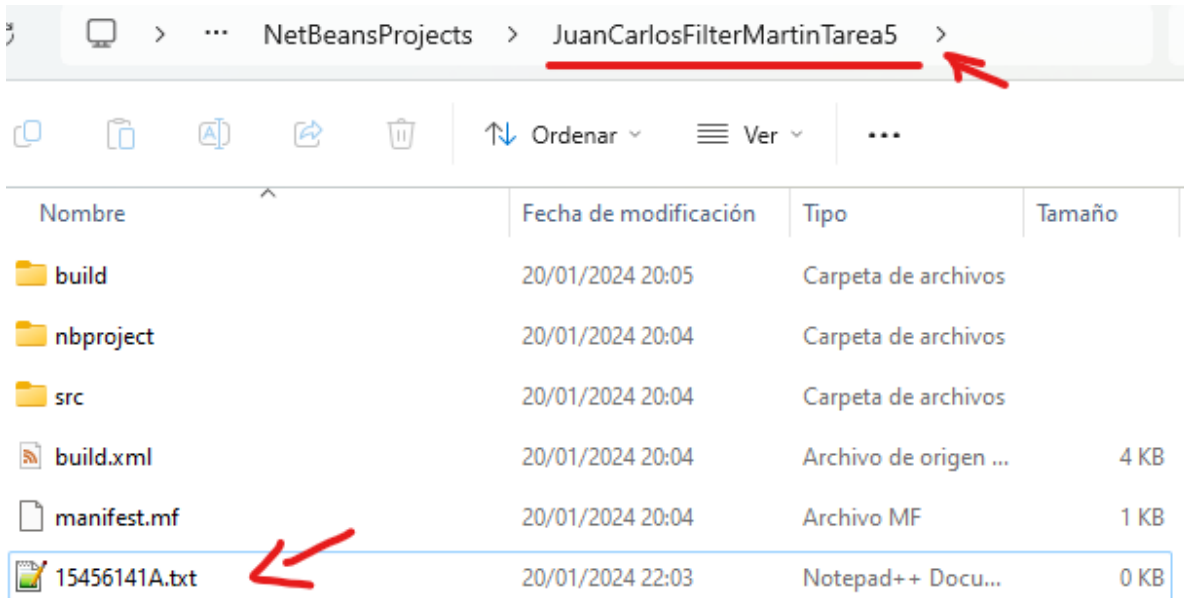
Para cada producto deberemos introducir (código, Nombre, Cantidad, Descripción)



Interfaz de usuario completa. A la izquierda, el formulario de entrada de datos con los campos 'Codigo', 'Nombre', 'Cantidad' y 'Descripción' (cada uno con una línea roja debajo) y los botones 'Insertar' y 'Buscar'. A la derecha, un panel titulado 'Listado de productos' que contiene una lista vacía. En la parte inferior derecha del panel, hay dos botones: 'Mostrar' (destacado con un recuadro naranja) y 'Salir'.

#### 4. RA05\_d) Se han utilizado ficheros para almacenar y recuperar información

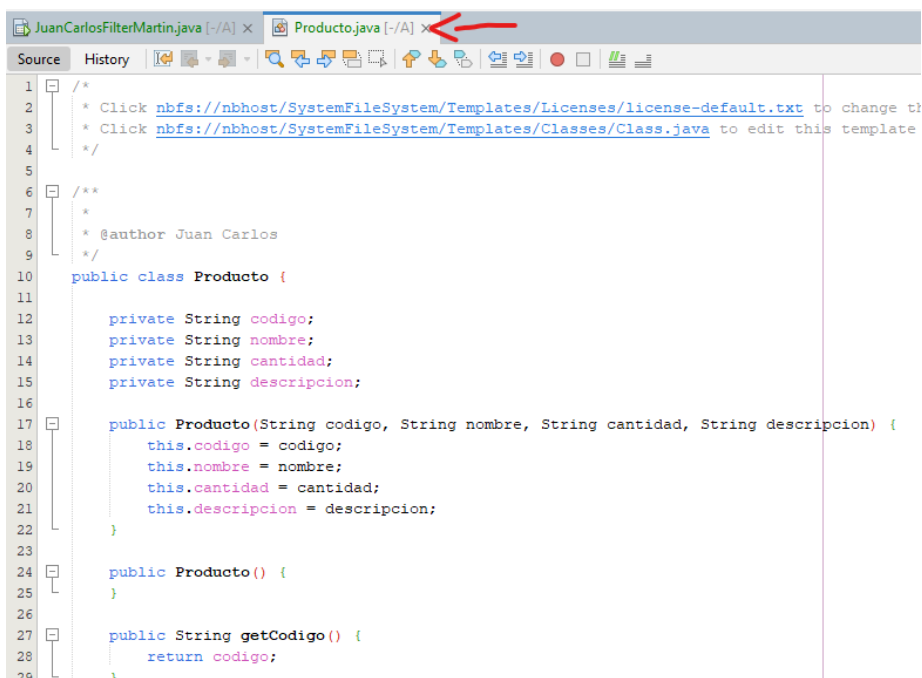
Fichero de texto llamado 15456141A.txt, guardado en la misma carpeta del proyecto.



Nombre	Fecha de modificación	Tipo	Tamaño
build	20/01/2024 20:05	Carpeta de archivos	
nbproject	20/01/2024 20:04	Carpeta de archivos	
src	20/01/2024 20:04	Carpeta de archivos	
build.xml	20/01/2024 20:04	Archivo de origen ...	4 KB
manifest.mf	20/01/2024 20:04	Archivo MF	1 KB
15456141A.txt	20/01/2024 22:03	Notepad++ Docu...	0 KB

[Insertar] podremos introducir los productos que deseemos.

Para ello primero vamos a crear una **clase llamada Producto** donde creamos las variables de los 4 campos del formulario, los constructores y los métodos get y set



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change the
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5
6  /**
7   *
8   * @author Juan Carlos
9   */
10 public class Producto {
11
12     private String codigo;
13     private String nombre;
14     private String cantidad;
15     private String descripcion;
16
17     public Producto(String codigo, String nombre, String cantidad, String descripcion) {
18         this.codigo = codigo;
19         this.nombre = nombre;
20         this.cantidad = cantidad;
21         this.descripcion = descripcion;
22     }
23
24     public Producto() {
25     }
26
27     public String getCodigo() {
28         return codigo;
29     }
```

También **crearemos en esta clase el método ToString** para cuando llamemos a `producto.ToString()` nos escriba todo el contenido en una linea separado por coma.

```
    @Override
    public String toString() {
        return codigo + "," + nombre + "," + cantidad + "," + descripcion;
    }
}
```

En nuestro JFrame si vamos al botón **Insertar** y pulsamos doble click o vamos a la **pestaña Source** podemos ver el método de este botón y seguidamente introduciremos la función de escribir en el.

Una vez introducido los datos mediante el botón insertar mostrará un mensaje y se borrarán los cuadros para poder volver introducir más datos.

Si el programa diera cualquier error saltaría la excepción y mostrará un mensaje de error.

```
private void boton_insertarActionPerformed(java.awt.event.ActionEvent evt) {
    File fichero = new File(pathname: "15456141A.txt");
    //Creamos el objeto de la clase Producto con el constructor para obtener el texto de los campos del formulario
    Producto producto = new Producto(codigo: campo_codigo.getText(), nombre: campo_nombre.getText(), cantidad: campo_cantidad.getText(), descripcion: campo_descripcion.getText());

    try {
        //Si el fichero no existe creamos un nuevo fichero.
        if (!fichero.exists()) {
            fichero.createNewFile();
        }
        //Escribimos en el fichero mediante el método ToString de la clase Producto y saltamos de linea.
        FileWriter fw = new FileWriter(file: fichero, append: true);
        fw.write(str: producto.toString());
        fw.write(str: System.getProperty(key: "line.separator"));

        fw.close();

        JOptionPane.showMessageDialog(parentComponent: null, message: "Se ha insertado correctamente",
            title: "ÉXITO", messageType: JOptionPane.INFORMATION_MESSAGE);
        campo_codigo.setText(t: "");
        campo_nombre.setText(t: "");
        campo_cantidad.setText(t: "");
        campo_descripcion.setText(t: "");

    } catch (IOException ex) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Se ha producido un error",
            title: "ERROR", messageType: JOptionPane.ERROR_MESSAGE);
    }
}
```

**[Buscar] el código del producto y si existe nos mostrará todos los datos del producto, si no existe nos dirá que ese producto no existe.**

### Crear ArrayList

Antes de ir al método del botón buscar vamos a crear un ArrayList de Producto para poder recorrer esa lista más adelante para ello tenemos que crearlo de la siguiente manera:

De la clase ArrayList con el tipo de datos Producto crear un nuevo objeto llamado arrayProducto.

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

    private ArrayList<Producto> arrayProducto = new ArrayList<>();
```

### Crear método leerFichero

Como leer el fichero se va a utilizar más adelante para el botón de listar vamos a crear un método y así evitar duplicar código

```
public void leerFichero(String ruta) {

    //Hace referencia al TextArea para borrar el contenido cada vez que pulsemos en Mostrar
    campo_lista.setText(""); //Elimina lo que se encuentra dentro del TextArea.
    arrayProducto.clear();    //Limpia el array

    File fichero = new File(pathname: ruta);

    try {
        //Si el fichero no existe en este caso lo indicamos mediante un mensaje de error
        if (!fichero.exists()) {
            JOptionPane.showMessageDialog(parentComponent:null, message:"El fichero no existe",
                title: "ERROR", messageType:JOptionPane.ERROR_MESSAGE);
        } else {
            //Si existe lo leemos mediante el FileReader y a su vez guardamos el contenido en memoria del bufered
            FileReader fd = new FileReader(file: fichero);
            BufferedReader bf = new BufferedReader(in: fd);

            //El bufferedReader se posiciona en la primera linea la lee y la devuelve a la variable linea
            String linea = bf.readLine();

            //Mientras linea no sea null ejecuta lo que hay dentro del while
            while (linea != null) {
                //La variable linea que contiene la linea completa la almacenaremos en el array
                //Cada posicion del array va separada del string con el ":"
                String[] producto = linea.split(regex: ":");
                //Se crea un objeto de la clase producto almacenandole el array recién creado e
                //introduciendo en cada posicion correspondiente
                Producto p = new Producto(producto[0], producto[1], producto[2], producto[3]);

                //añadimos el objeto producto al ArrayList arrayProducto y el bufferedReader se
                //posiciona en la siguiente linea y la devuelve a la variable linea
                arrayProducto.add(e: p);
                linea = bf.readLine();
            }
            fd.close();
        }
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"Se ha producido un error",
            title: "ERROR", messageType:JOptionPane.ERROR_MESSAGE);
    }
}
```



## Método Buscar

Una vez creado el ArrayList y el método que nos ayudará a leer el fichero vamos a hacer la función en el botón buscar.

- **Leemos el fichero:**

```
private void boton_buscarActionPerformed(java.awt.event.ActionEvent e) {
    //Leemos el fichero mediante este método
    leerFichero(ruta: "15456141A.txt");
}
```

- **Creamos un bucle para recorrer el ArrayList Producto**

- Primero el bucle for cuenta las líneas del array y las va a recorrer una a una

- Una vez dentro del bucle mediante un if va a comprobar si en el array se encuentra el mismo código escrito en el campo\_codigo (TextField).

- Si se cumple la condición en primer lugar cambia la variable coincide a true (para que no salte el error) y devuelve (envía a los TextFeld) el contenido de la lista en su propia posición.

- Si no entra en el if porque no se cumple la condición entonces como la variable boolean no cambia entrará finalmente en el if con el JOptionPane que contiene el mensaje de error.

```
//Con esta variable contramos si hay error o no y poder entrar en el if con el JOptionPane
boolean coincide = false;

/*se crea un bucle para recorrer el arrayList Producto.
En el array pasa por la primera linea, coge el codigo y compara con el codigo escrito en el campo_codigo del formulario.
Si no coincide no entra en el if, vuelve a comprobar la siguiente linea del array y si se diera el caso de que si
coincide entra en el if y escribe en los campos del formulario correspondiente elCodigo, nombre, cantidad y descripcion
*/
for (int i = 0; i < arrayProducto.size(); i++) {
    if (arrayProducto.get(index: i).getCodigo().equals(anObject: campo_codigo.getText())) {
        coincide = true;
        campo_codigo.setText(t: arrayProducto.get(index: i).getCodigo());
        campo_nombre.setText(t: arrayProducto.get(index: i).getNombre());
        campo_cantidad.setText(t: arrayProducto.get(index: i).getCantidad());
        campo_descripcion.setText(t: arrayProducto.get(index: i).getDescripcion());
        break;
    }
}

if (!coincide) {
    JOptionPane.showMessageDialog(parentComponent:null, message:"Este producto no existe",
        title: "ERROR", messageType:JOptionPane.ERROR_MESSAGE);
}
```

**[Listado/Mostrar] nos mostrará todos los productos dados de alta con todos sus datos.**

El botón mostrar / listado también utiliza leerFichero y el ArrayList arrayProducto.

**leerFichero** contiene lo siguiente (para eliminar todo el contenido del TextArea cada vez que pulsemos el botón de mostrar y muestre el contenido nuevo:

```
public void leerFichero(String ruta) {  
  
    //Hace referencia al TextArea para borrar el contenido cada vez que pulsemos en Mostrar  
    campo_lista.setText(""); //Elimina lo que se encuentra dentro del TextArea.  
    arrayProducto.clear();    //Limpia el array
```

Aparte se ha creado un método para listar los productos

```
public void listarProducto(Producto p) {  
    //Se crea una variable String productoListado para que las lista se muestre mejor visualmente  
    //mediante concatenaciones  
    String productoListado = "| " + p.getCodigo() + " | "  
        + p.getNombre() + " | "  
        + p.getCantidad() + " | "  
        + p.getDescripcion() + " |";  
    //En el texArea borramos dicho recuadro y añadimos el producto listado  
    campo_lista.setText(campo_lista.getText() + productoListado + "\n");  
}
```

### Método Mostrar

Ahora en el **botón Mostrar** mediante un bucle for recorreremos todo el array en lista e imprimimos en el cuadro del text area el contenido.

```
private void boton_mostrarActionPerformed(java.awt.event.ActionEvent evt) {  
    leerFichero(ruta: "15456141A.txt");  
  
    //Bucle for comenzando en la línea 0 para que mientras el arrayProducto contenga más líneas entre dentro  
    //e imprima en listarProducto (hace referencia al campo_lista.setText()) la línea que se encuentra  
    //posicionada el for en ese momento  
    for (int i = 0; i < arrayProducto.size(); i++) {  
        listarProducto(p: arrayProducto.get(index: i));  
    }  
}
```

**[Salir] cerrará la aplicación.**

Para salir de la aplicación se hace mediante `System.exit(0)` dentro del botón salir.

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

5. RA05\_h) Se han escrito programas que utilicen interfaces gráficos para la entrada y salida de información.

La programación se hará en un entorno gráfico, usando los controles que el alumno considere oportunos (*botones, textbox, listbox, combobox, etc.....*)

La interfaz gráfica es la siguiente:

Diagrama de la interfaz gráfica de usuario (GUI) con las siguientes componentes y etiquetas:

- Label:** Etiqueta para el título "Listado de productos".
- TextField:** Campos de entrada para "Codigo", "Nombre", "Cantidad" y "Descripción".
- Button:** Botones "Insertar", "Buscar", "Mostrar" y "Salir".
- TextArea:** Área de texto para el listado de productos.

La interfaz está dividida en dos paneles. El panel izquierdo contiene los campos de entrada y los botones "Insertar" y "Buscar". El panel derecho contiene el "Listado de productos" y los botones "Mostrar" y "Salir".

## 6. RA5\_g) Se han programado controladores de eventos

**Se programarán los botones de insertar, buscar, mostrar y salir que son los que le darán la funcionalidad a la aplicación.**

Para este apartado voy a mostrar las capturas del código y del programa ejecutándose.

### Insertar

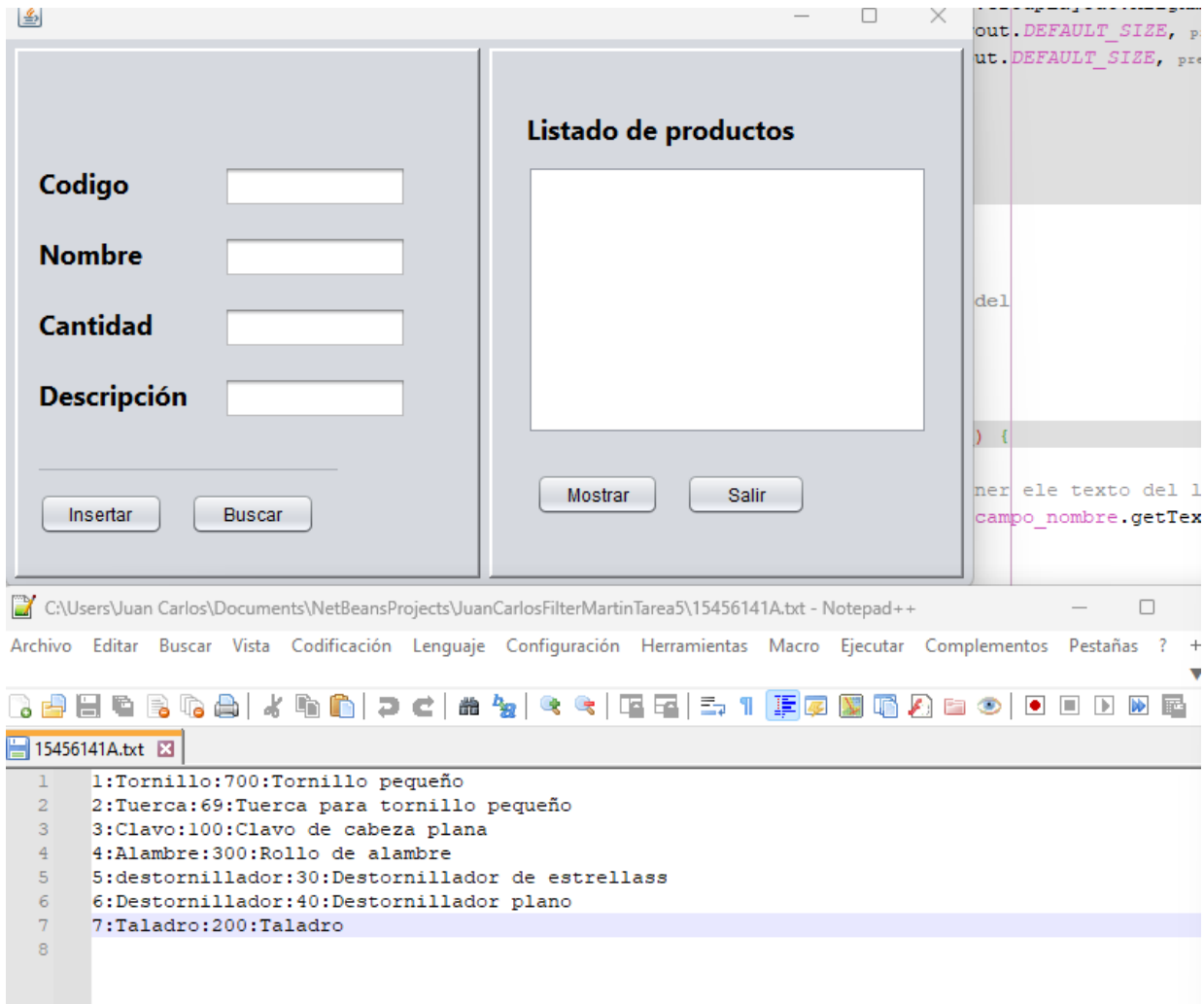
El objetivo es captar la información tecleada por el usuario de la aplicación y guardarla en el fichero.

#### × Código

```
private void boton_insertarActionPerformed(java.awt.event.ActionEvent evt) {  
    File fichero = new File(pathname: "15456141A.txt");  
    //Creamos el objeto de la clase Producto con el constructor para obtener el texto de los campos del formulario  
    Producto producto = new Producto(codigo: campo_codigo.getText(), nombre: campo_nombre.getText(), cantidad: campo_cantidad.getText(),  
    descripcion: campo_descripcion.getText());  
  
    try {  
        //Si el fichero no existe creamos un nuevo fichero.  
        if (!fichero.exists()) {  
            fichero.createNewFile();  
        }  
        //Escribimos en el fichero mediante el método ToString de la clase Producto y saltamos de línea.  
        FileWriter fw = new FileWriter(file: fichero, append: true);  
        fw.write(str: producto.toString());  
        fw.write(str: System.getProperty(key: "line.separator"));  
  
        fw.close();  
  
        JOptionPane.showMessageDialog(parentComponent: null, message: "Se ha insertado correctamente",  
            title: "ÉXITO", messageType: JOptionPane.INFORMATION_MESSAGE);  
        campo_codigo.setText(t: "");  
        campo_nombre.setText(t: "");  
        campo_cantidad.setText(t: "");  
        campo_descripcion.setText(t: "");  
    } catch (IOException ex) {  
        JOptionPane.showMessageDialog(parentComponent: null, message: "Se ha producido un error",  
            title: "ERROR", messageType: JOptionPane.ERROR_MESSAGE);  
    }  
}
```

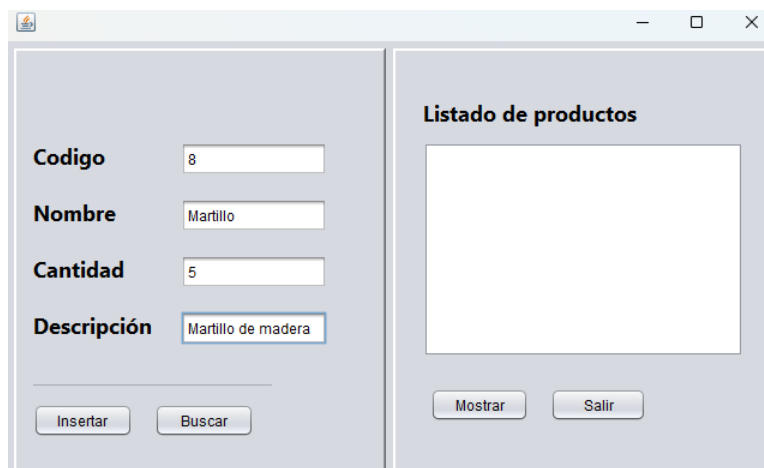
× **Gráfico ejecutable**

- × Este es el programa ejecutado y el fichero de texto con el contenido

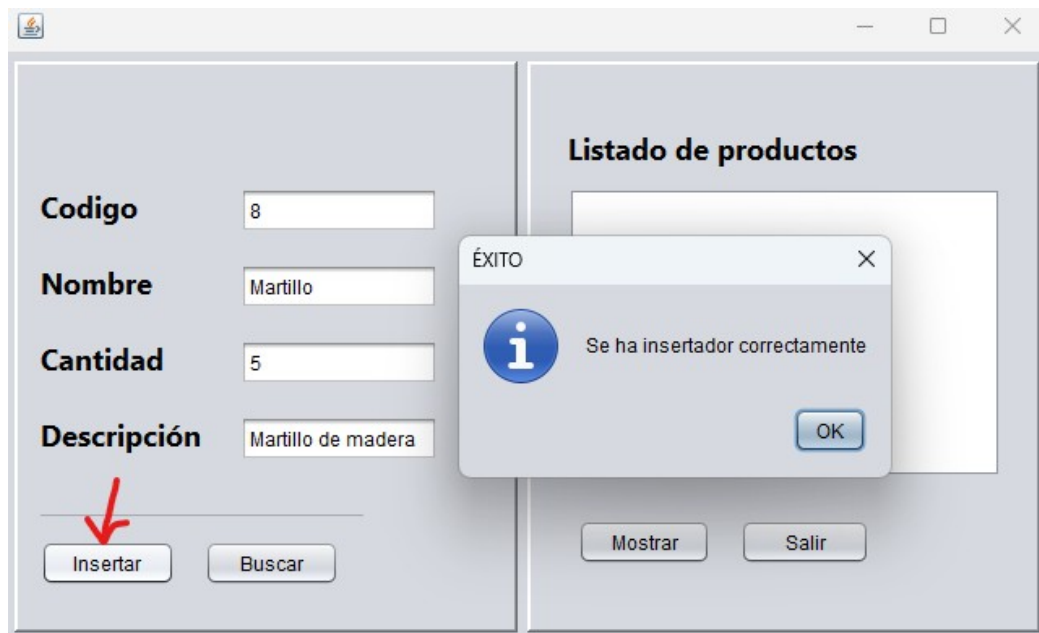


- × Si agregamos mediante el grafico un producto

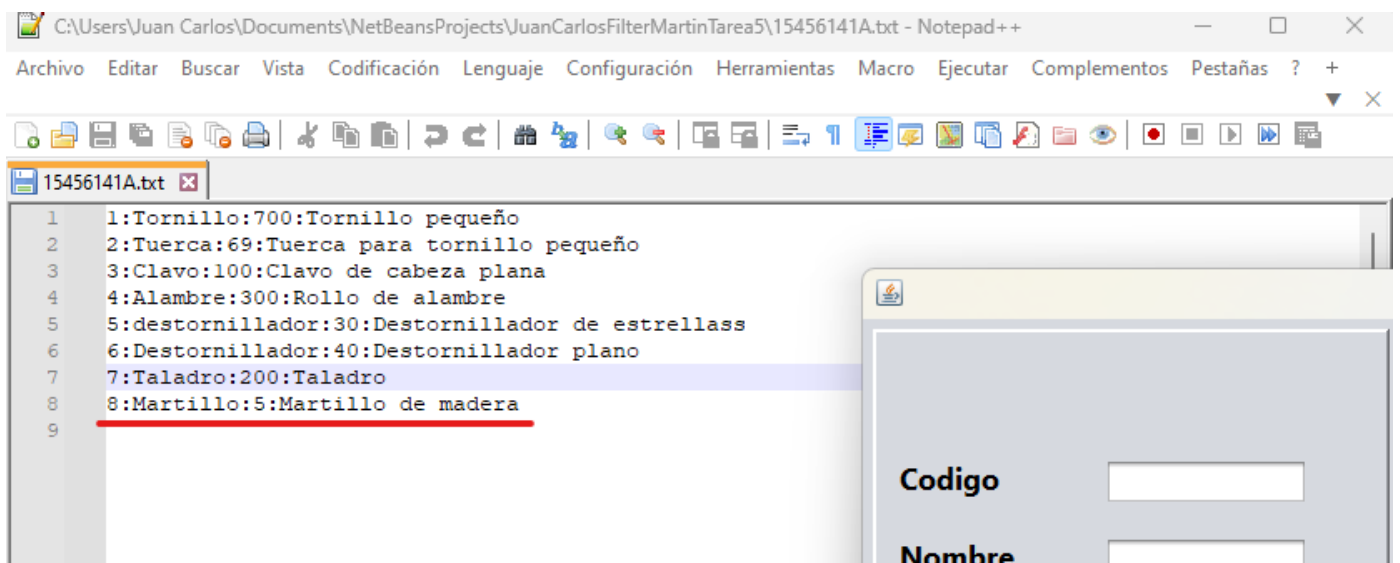
**Código:**8, **Nombre:**Martillo, **Cantidad:** 5 **Descripción:** Martillo de madera



- ✱ Y pulsamos en insertar aparecerá un mensaje indicando que se ha insertado el producto.



- ✱ Por ultimo si vamos al fichero podemos ver como se a agregado este producto



## Buscar

El usuario escribirá un código en el campo correspondiente, al hacer click en buscar, el programa buscará en el fichero el producto con ese código.

Si no existe dará un mensaje de error indicándolo.

Si existe, rellenará el resto de campos con los datos que corresponden al producto con el id tecleado.

### × Código

```
private void boton_buscarActionPerformed(java.awt.event.ActionEvent evt) {  
    //Leemos el fichero mediante este método  
    leerFichero(ruta: "I5456141A.txt");  
  
    //Con esta variable contramos si hay error o no y poder entrar en el if con el JOptionPane  
    boolean coincide = false;  
  
    /*se crea un bucle para recorrer el arrayList Producto.  
    En el array pasa por la primera linea, coge el codigo y compara con el codigo escrito en el campo_codigo del formulario.  
    Si no coincide no entra en el if, vuelve a comprobar la siguiente linea del array y si se diera el caso de que si  
    coincide entra en el if y escribe en los campos del formulario correspondiente elCodigo, nombre, cantidad y descripcion  
    */  
    for (int i = 0; i < arrayProducto.size(); i++) {  
        if (arrayProducto.get(index: i).getCodigo().equals(anObject: campo_codigo.getText())) {  
            coincide = true;  
            campo_codigo.setText(t: arrayProducto.get(index: i).getCodigo());  
            campo_nombre.setText(t: arrayProducto.get(index: i).getNombre());  
            campo_cantidad.setText(t: arrayProducto.get(index: i).getCantidad());  
            campo_descripcion.setText(t: arrayProducto.get(index: i).getDescripcion());  
            break;  
        }  
    }  
    if (!coincide) {  
        JOptionPane.showMessageDialog(parentComponent:null, message:"Este producto no existe",  
            title: "ERROR", messageType:JOptionPane.ERROR_MESSAGE);  
    }  
}
```

### × Gráfico ejecutable

- × Con el programa ejecutado si escribimos un código que se encuentre en el fichero como puede ser el 7, rellenará los campos de nombre, cantidad y descripción de ese producto.

The screenshot shows a Java Swing window titled "Listado de productos". On the left, there is a form with four labels and text input fields: "Codigo" (containing "7"), "Nombre" (containing "Taladro"), "Cantidad" (containing "200"), and "Descripción" (containing "Taladro"). Below these fields are two buttons: "Insertar" and "Buscar". The "Buscar" button is highlighted with a blue border. On the right side of the window, there is a large empty rectangular area, likely a list box or text area for displaying search results. At the bottom right, there are two buttons: "Mostrar" and "Salir".

- ✱ Si vamos al fichero podemos comprobar que el código 7 se corresponde a ese producto.

C:\Users\Juan Carlos\Documents\NetBeansProjects\JuanCarlosFilterMartinTarea5\15456141A.txt -

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro

15456141A.txt

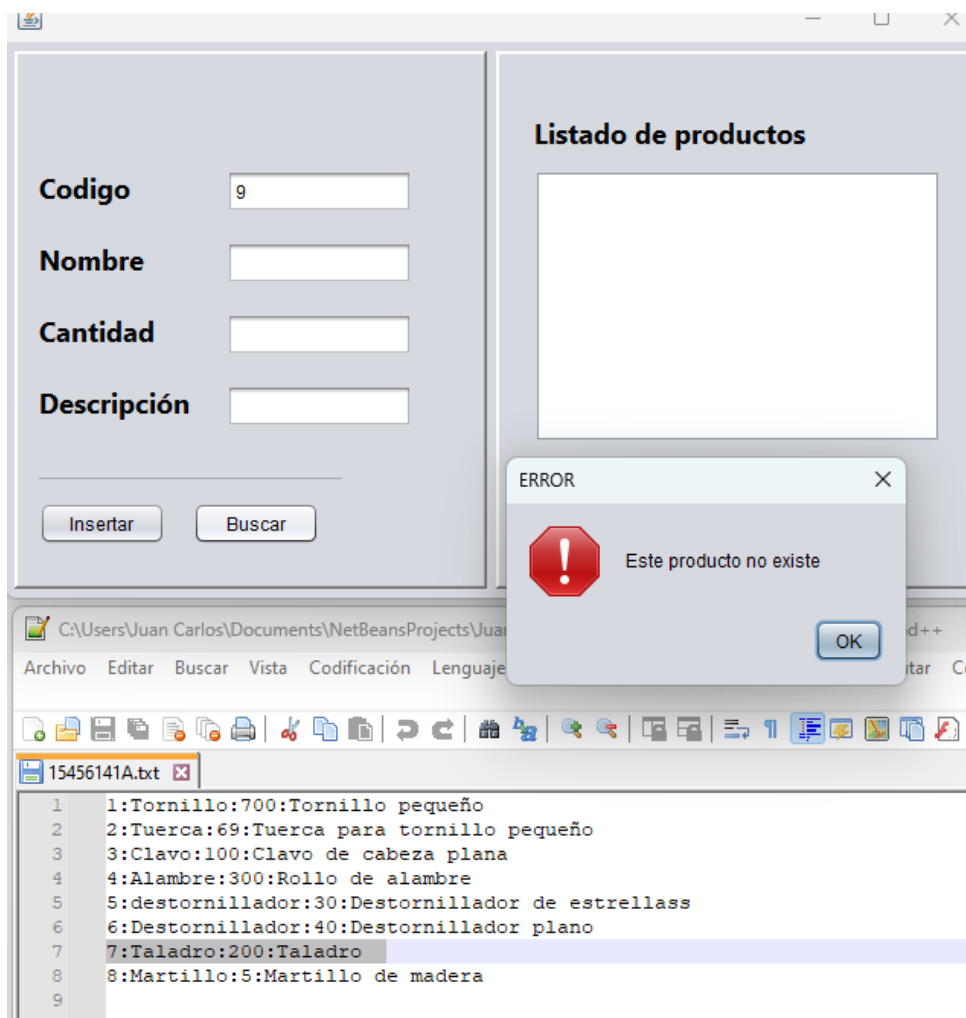
```

1 1:Tornillo:700:Tornillo pequeño
2 2:Tuerca:69:Tuerca para tornillo pequeño
3 3:Clavo:100:Clavo de cabeza plana
4 4:Alambre:300:Rollo de alambre
5 5:destornillador:30:Destornillador de estrellass
6 6:Destornillador:40:Destornillador plano
7 7:Taladro:200:Taladro
8 8:Martillo:5:Martillo de madera
9

```

- ✱ Si en cambio ponemos un código que no se corresponde con ningún producto, obtendremos un mensaje de error.

El producto con el código en este caso el 9 no existe





## Mostrar

Muestra dentro del cuadro de la derecha una lista con todos los datos registros en el fichero.

### × Código

```
private void boton_mostrarActionPerformed(java.awt.event.ActionEvent evt) {  
    leerFichero(ruta: "15456141A.txt");  
  
    //Bucle for comendando en la linea 0 para que mientras el arrayProducto contenga más linea entre dentro  
    //e imprima en listarProducto (hace referencia al campo_lista.setText() la linea que se encuentra  
    //posicionada el for en ese momento  
    for (int i = 0; i < arrayProducto.size(); i++) {  
        listarProducto(p: arrayProducto.get(index: i));  
    }  
}
```

### × Gráfico ejecutable

- × Desde el ejecutable si pulsamos el botón mostrar podemos ver el resultado del método que hace referencia a este botón.

**Codigo**

**Nombre**

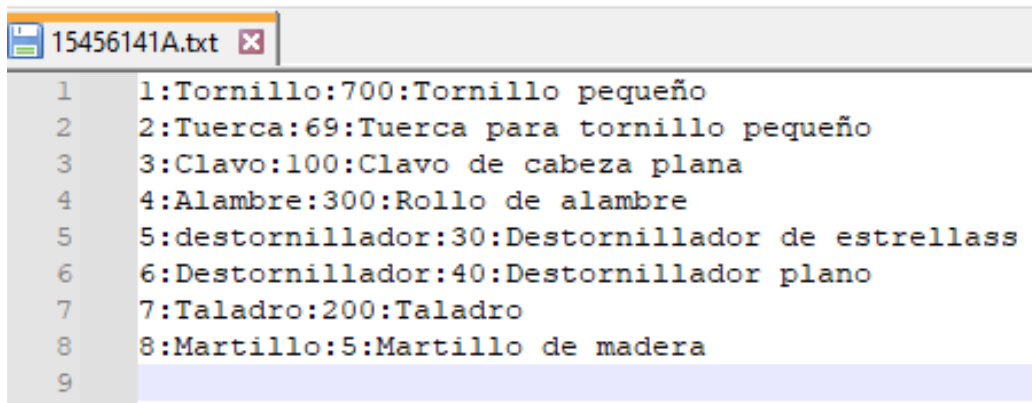
**Cantidad**

**Descripción**

**Listado de productos**

| 1 | Tornillo | 700 | Tornillo pequeño  
| 2 | Tuerca | 69 | Tuerca para tornillo pequeño  
| 3 | Clavo | 100 | Clavo de cabeza plana  
| 4 | Alambre | 300 | Rollo de alambre  
| 5 | destornillador | 30 | Destornillador de estre  
| 6 | Destornillador | 40 | Destornillador plano  
| 7 | Taladro | 200 | Taladro  
| 8 | Martillo | 5 | Martillo de madera

- ✖ Si vamos al fichero se puede comprobar que los datos mostrados son los mismos que se encuentran en el fichero.



```
1 1:Tornillo:700:Tornillo pequeño
2 2:Tuerca:69:Tuerca para tornillo pequeño
3 3:Clavo:100:Clavo de cabeza plana
4 4:Alambre:300:Rollo de alambre
5 5:destornillador:30:Destornillador de estrellass
6 6:Destornillador:40:Destornillador plano
7 7:Taladro:200:Taladro
8 8:Martillo:5:Martillo de madera
9
```

## Salir

Cierra la aplicación.

### ✖ Código

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

### ✖ Gráfico ejecutable

- ✖ Procedemos a ejecutar la aplicación



- ✖ Si pulsamos en el botón de Salir se cerrará la aplicación.

- ✖ El fichero donde se guardarán los datos se llama **15456141A.txt** y está guardado en la misma carpeta donde se ejecuta la aplicación.

