Universidad Técnica Particular de Loja

Área Técnica
Carrera Computación

Informe Proyecto POO

Nombre: Andres Vallejo Z.

Docente: Ing. Richar Guaya.

Materia: Programación Orientada a Objetos.

Fecha: 19/03/2020

Proyecto Interfaz Gráfica de Usuario.

1) Definir el Problema.

Desarrollar una interfaz grafica de usuario que permita simular una tienda al registrar cierta cantidad de productos, para posteriormente el usuario pueda realizar una factura dependiendo de la cantidad de productos que quiere retirar.

Se debe guardar los productos ingresados dentro una lista en la cual se debe colocar principalmente la cantidad que se encuentra en Stock, y el precio.

2) Análisis del Problema.

Entradas:

nombre, unidad, cantidad, precio

Procesos:

sub_total = (cantidad * precio)

iva = 0.12*sub_total

total = iva + sub_total

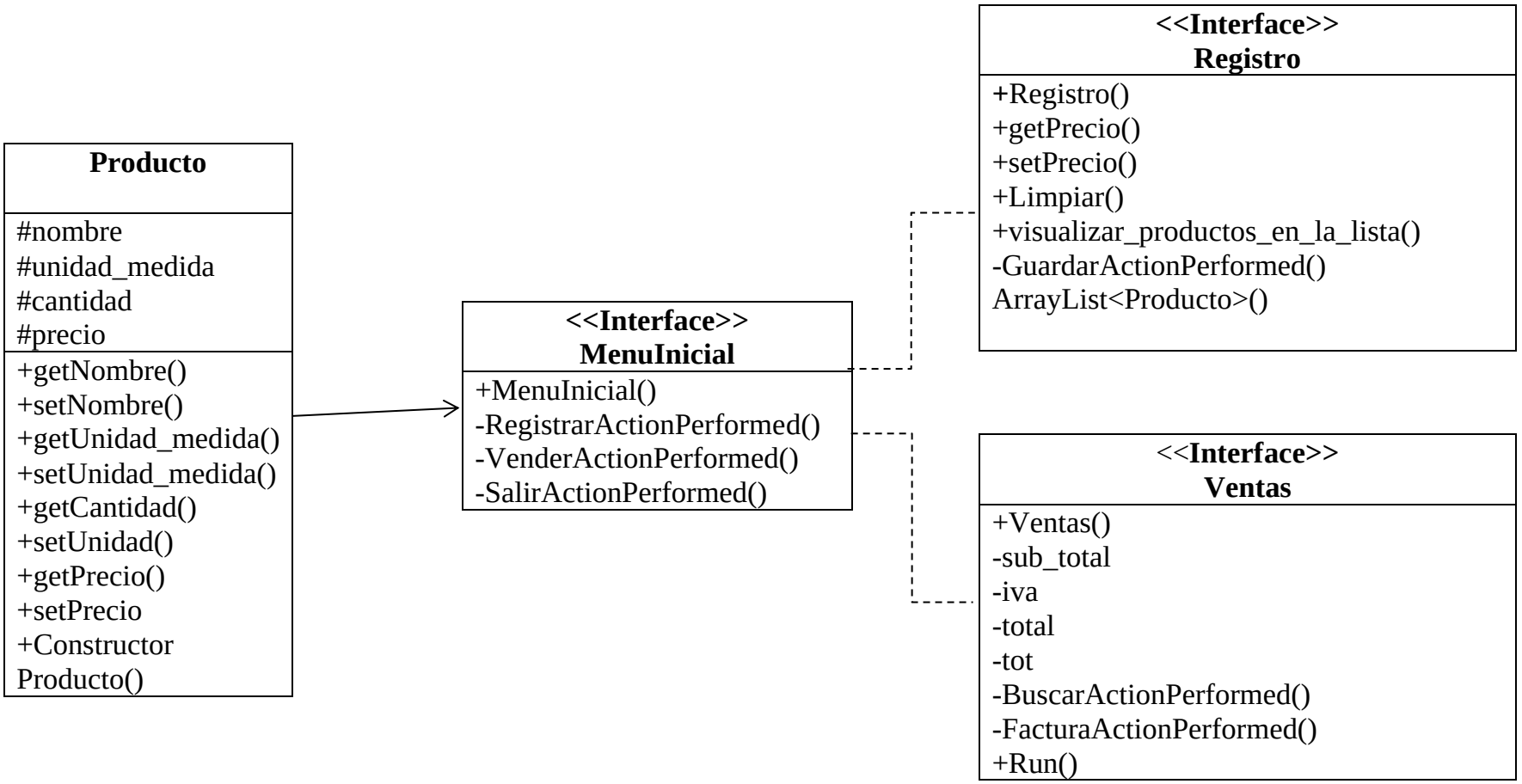
tot +=total

Salidas:

nombre, unidad, cantidad, precio, prodNombres[ ], sub_total, iva, total, tot

3) Diseño del programa.

a) Diagrama de Clase


Producto
#nombre
#unidad_medida
#cantidad
#precio
+getNombre()
+setNombre()
+getUnidad_medida()
+setUnidad_medida()
+getCantidad()
+setUnidad()
+getPrecio()
+setPrecio
+Constructor
Producto()
<<Interface>>
MenuInicial
+MenuInicial()
-RegistrarActionPerformed()
-VenderActionPerformed()
-SalirActionPerformed()
<<Interface>>
Registro
+Registro()
+getPrecio()
+setPrecio()
+Limpiar()
+visualizar_productos_en_la_lista()
-GuardarActionPerformed()
ArrayList<Producto>()
<<Interface>>
Ventas
+Ventas()
-sub_total
-iva
-total
-tot
-BuscarActionPerformed()
-FacturaActionPerformed()
+Run()

b) Pseudocódigo

Algoritmo Interfaz

Clase Producto

1) Definir Variables

   nombre: Cadena

   unidad_medida: Cadena

   cantidad: Entero

   precio : Real

2) Metodo obtenerNombre( ):
   return nombre
   Fin metodo
3) Metodo establecerNombres(name: Cadena):
   nombre = name
   Fin metodo
4) Metodo obtenerUnidad_medidad( ):
   return unidad_medida
   Fin metodo
5) Metodo establecerUnidad_mediad(unit: Cadena):
   unidad_medida = unit
   Fin metodo
6) Metodo obtenerCantidad( ):
   return cantidad
   Fin metodo
7) Metodo establecerCantidad(cant: Entero):
   cantidad = cant
   Fin metodo
8) Metodo obtenerPrecio( ):
   return precio
   Fin metodo

9) Metodo establecerPrecio(prec: Entero):

precio = prec

Fin metodo

10) Metodo Constructor

Producto(String nombre, String unidad_medida, int cantidad, double precio):

this.nombre = nombre;

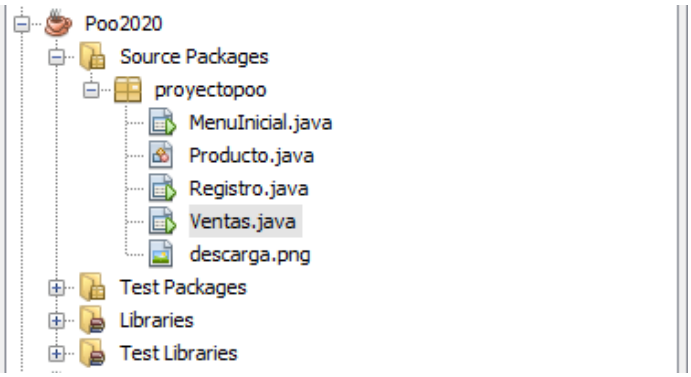
this.unidad_medida = unidad_medida;

this.cantidad = cantidad;

this.precio = precio;

Fin metodo Constructor

c) Explicación del proyecto



Vista General del Proyecto.

```
69      protected String nombre;
70      protected String unidad_medida;
71      protected int cantidad;
72      protected double precio;
```

Las variables nombre, unidad_medida, cantidad y precio dentro de la clase producto, las cuales son declaradas como protegidas para que pueden ser usadas por otras clases o en este caso dentro de los JFrame posteriores.

La variable nombre es una cadena que almacena los nombres de los productos.

La variable unidad_medida es una cadena que almacena la medición de los productos.

La variable cantidad es un entero que almacena el número de stock de los productos.

La variable precio es un double que almacena el precio por unidad de cada producto.

```
73
74      public Producto(String nombre, String unidad_medida, int cantidad, double precio) {
75          this.nombre = nombre;
76          this.unidad_medida = unidad_medida;
77          this.cantidad = cantidad;
78          this.precio = precio;
79      }
80
```

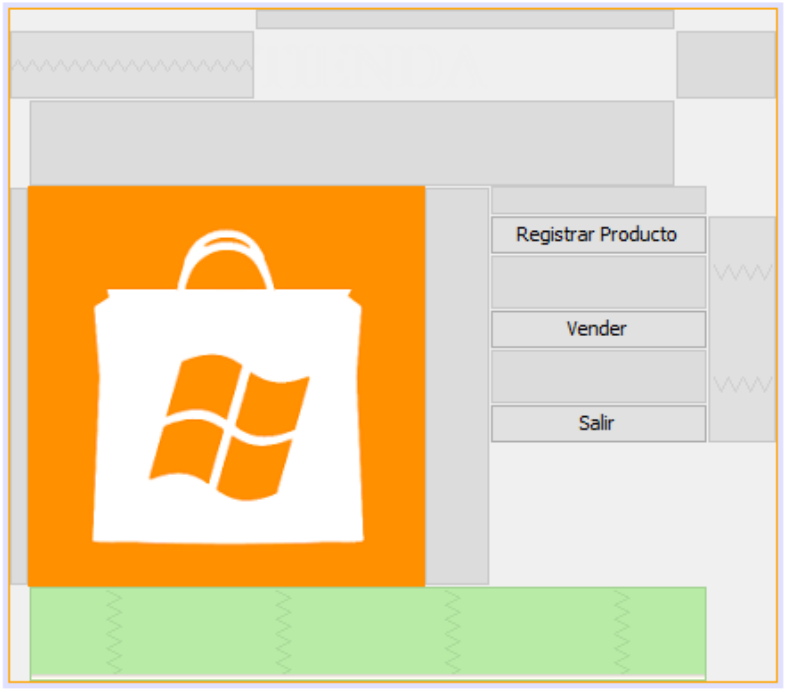
Creación del constructor con el nombre Producto, el cual permite inicializar un objeto el cual es llamado por otra clase.

En este caso se crea un constructor asignando las 4 variables mencionadas anteriormente, también es posible usar el constructor por defecto.

```
13
14      /**
15       * @return the nombre
16       */
17      public String getNombre() {
18          return nombre;
19      }
20
21      /**
22       * @param nombre the nombre to set
23       */
24      public void setNombre(String nombre) {
25          this.nombre = nombre;
26      }
27
28      /**
29       * @return the unidad_medida
30       */
31      public String getUnidad_medida() {
32          return unidad_medida;
33      }
34
```

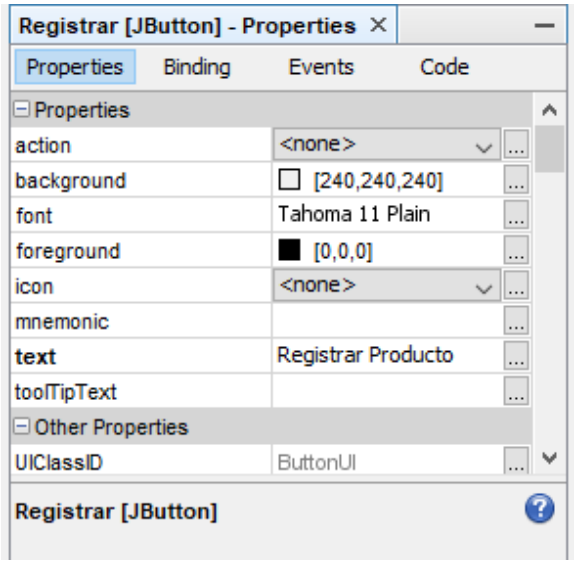
```
37      */
38     public void setUnidad_medida(String unidad_medida) {
39         this.unidad_medida = unidad_medida;
40     }
41
42     /**
43      * @return the cantidad
44      */
45     public int getCantidad() {
46         return cantidad;
47     }
48     /**
49      * @param cantidad the cantidad to set
50      */
51     public void setCantidad(int cantidad) {
52         this.cantidad = cantidad;
53     }
54
55     /**
56      * @return the precio
57      */
58     public double getPrecio() {
59         return precio;
60     }
61
62     /**
63      * @param precio the precio to set
64      */
65     public void setPrecio(double precio) {
66         this.precio = precio;
67     }
```

Métodos generados por el programa NetBeans donde se muestran los métodos obtener y establecer de las variables de la clase Producto.

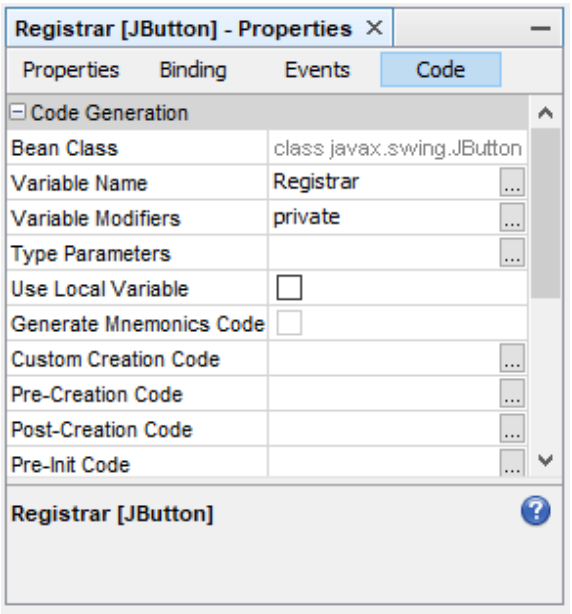


Diseño del menú principal a partir del JFrame MenuInicial.

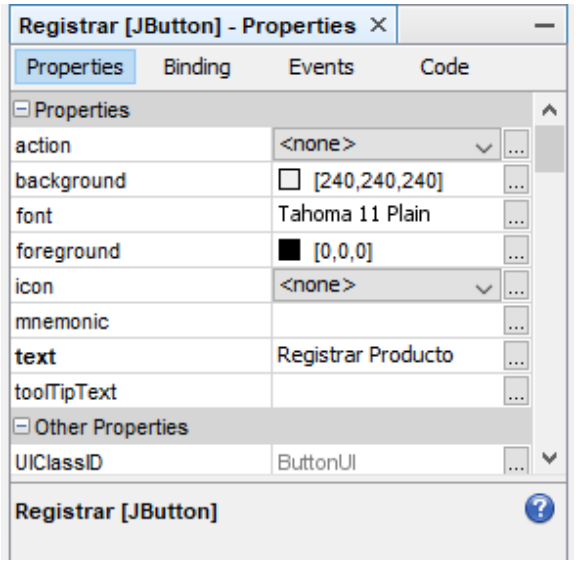
Para ello se utilizaron 3 botones y 2 labels, los botones que se dividen en Registrar Producto, Vender, Salir se les fueron asignados acciones mientras los dos labels restantes se usaron para colocar una imagen dentro de la interfaz y el titulo que se encuentra en la parte superior.



En la pestaña de propiedades se puede editar el tipo de letra y cambiar el color de la fuente en formato color RGB.



En la pestaña de Code se puede cambiar el nombre de la variable para mayor facilidad a la hora de realizar el código de los JFrame.



En la pestaña de Propiedades también se encuentra la opción icon, en la cual se pueden cargar imagen para ser usadas dentro de la interfaz, en el caso se cargo la imagen dentro de un label para colocarla en la interfaz.

```
19      public MenuInicial() {
20          initComponents();
21          this.getContentPane().setBackground(Color.blue);
22
23      }
```

Mediante esta la línea número 21 de código podemos cambiar el color de fondo de la interfaz que en este caso se cambian a un color azul.

```
110
111     private void RegistrarActionPerformed(java.awt.event.ActionEvent evt) {
112       Registro objRegistro = new Registro();
113       objRegistro.show();
114     }
115
116     private void VenderActionPerformed(java.awt.event.ActionEvent evt) {
117        Ventas ven = new Ventas();
118        ven.show();
119     }
120
121     private void SalirActionPerformed(java.awt.event.ActionEvent evt) {
122         System.exit(WIDTH);
123     }
124
```

```
110
111     private void RegistrarActionPerformed(java.awt.event.ActionEvent evt) {
112         Registro objRegistro = new Registro();
113         objRegistro.show();
114     }
```

Mediante el llamado de un constructor por defecto más el comando show(); permitimos que mediante el botón registrar se muestre la siguiente ventana de interfaz en este caso la de Registro de productos.

```
115
116     private void VenderActionPerformed(java.awt.event.ActionEvent evt) {
117         Ventas ven = new Ventas();
118         ven.show();
119     }
120
```

Mediante el llamado de un constructor por defecto más el comando show(); permitimos que mediante el botón Vender se muestre la siguiente ventana de interfaz en este caso la de Venta de productos.

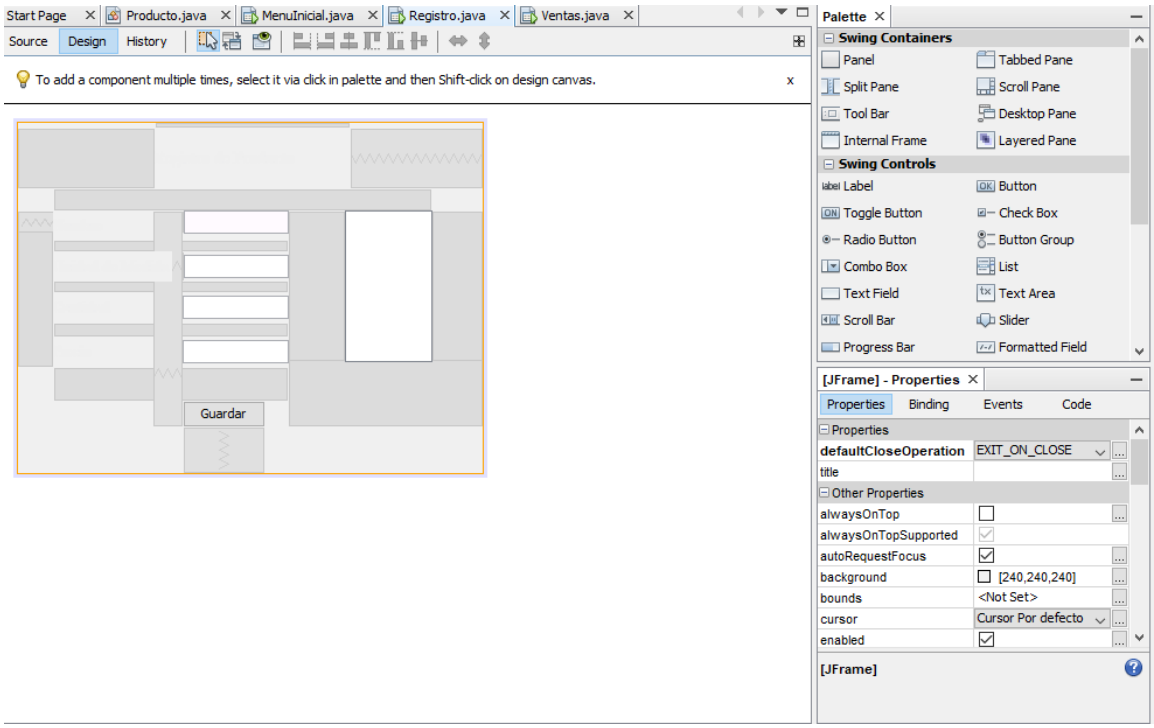
```
120
121     private void SalirActionPerformed(java.awt.event.ActionEvent evt) {
122         System.exit(WIDTH);
123     }
124
```

Para el botón salir usamos la línea System.exit(WIDTH) la cual terminara el proceso cuando presionemos el botón.

```
153         /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
156                 new MenuInicial().setVisible(true);
157             }
158         });
159     }
```

Mediante este metodo hacemos que el menú inicial siga siendo visible después de acceder a las ventanas de Registro y Venta.

Interfaz final de MenuInicial.

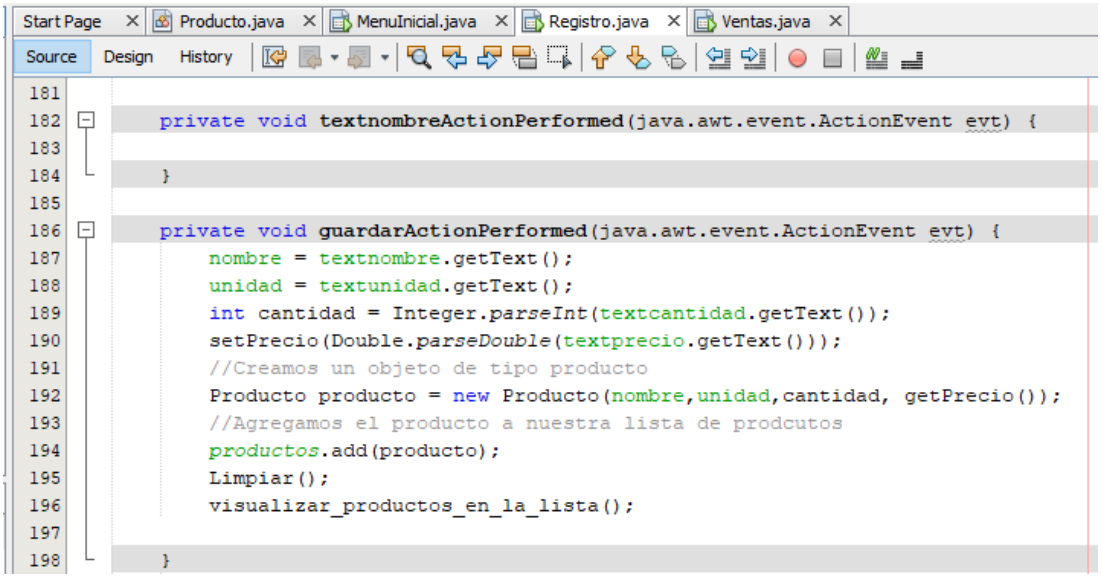


Diseño del JFrame de Registro, la misma conserva similares características al menú inicial como el fondo de la interfaz, fuente de la letra y color de la misma.

Adicionalmente se añadió una lista en la cual se almacenan los productos que se vayan ingresando.

```
static List<Producto> productos = new ArrayList<Producto>();
```

Se crea un ArrayList de tipo producto para de esta manera almacenar los datos de los productos que se ingresaran en el stock, también se crean nuevas variables para luego inicializarlas mediante el constructor de la clase Producto.




```
181
182     private void textnombreActionPerformed(java.awt.event.ActionEvent evt) {
183
184     }
185
186     private void guardarActionPerformed(java.awt.event.ActionEvent evt) {
187         nombre = textnombre.getText();
188         unidad = textunidad.getText();
189         int cantidad = Integer.parseInt(textcantidad.getText());
190         setPrecio(Double.parseDouble(textprecio.getText()));
191         //Creamos un objeto de tipo producto
192         Producto producto = new Producto(nombre,unidad,cantidad, getPrecio());
193         //Agregamos el producto a nuestra lista de prodcutos
194         productos.add(producto);
195         Limpiar();
196         visualizar_productos_en_la_lista();
197
198     }
```

Las acciones que se realizaran al presionar el botón guardar.

```
nombre = textnombre.getText();
unidad = textunidad.getText();
```

Mediante estas líneas de código lo que el usuario coloque dentro la interfaz será guardado dentro de estas variables.

textnombre y textunidad son cuadros de texto vacíos donde el usuario colocara el nombre y la unidad dentro del stock.

```
int cantidad = Integer.parseInt(textcantidad.getText());
setPrecio(Double.parseDouble(textprecio.getText()));
```

En cantidad y precio se realiza el mismo proceso, adicionalmente luego de que el usuario digite los espacios se debe transformar a entero y double respectivamente para que los datos no se queden estancados como cadenas de texto.

```
//Creamos un objeto de tipo producto
Producto producto = new Producto(nombre,unidad,cantidad, getPrecio());
```

Mediante el constructor colocamos todos los datos recopilados dentro de un mismo objeto.

```
//Agregamos el producto a nuestra lista de prodcutos
productos.add(producto);
```

Por consiguiente, adicionamos el producto al ArrayList que creamos anteriormente.

```
public void Limpiar(){
    textnombre.setText("");
    textunidad.setText("");
    textcantidad.setText("");
    textprecio.setText("");
}
```

Mediante este metodo conseguimos que una vez que le usuario digite la información en los espacios asignados, estos vuelvan a colocarse en blanco para que le usuario no tenga que borrar los espacios manualmente.

```
Limpiar();
```

Colocamos el metodo dentro del botón de guardar y después de añadir los productos a la lista.

```
    }
    public void visualizar_productos_en_la_lista(){
        String [] prodNombres = new String[productos.size()];
        int contador =0;
        for(Producto producto : productos){
            prodNombres[contador]= producto.getNombre();
            contador++;
        }
```

Mediante este metodo colocamos los productos para que sean almacenados dentro de una lista, creamos un arreglo de tipo cadena del tamaño de la lista de los productos y mediante un ciclo For los productos serán visibles dentro del listado.

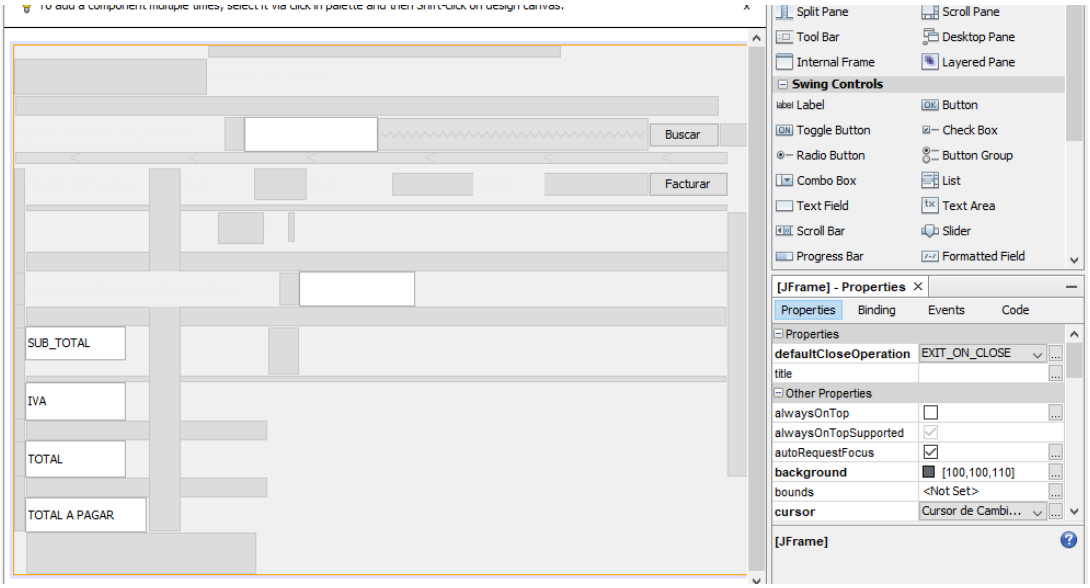
```
        ListProducts.setListData(prodNombres);
    }
```

De esta manera colocamos el nombre del producto dentro del listado que fue nombrado como ListProducts.

```
    visualizar_productos_en_la_lista();
```

Asimismo, colocamos el metodo dentro las acciones realizadas por el botón guardar, se coloca después del metodo limpiar.

Interfaz final del JFrame Registro de Productos.



Diseño del JFrame Ventas.

```
private void BuscarActionPerformed(java.awt.event.ActionEvent evt) {
    String nombre_buscar = txtbusqueda.getText();
    lblnombre.setText("");
    for(Producto p: Registro.productos){
        //Verificamos que el enombre a buscar este contenido en el arreglo
        if(p.getNombre().contains(nombre_buscar)){
            lblnombre.setText(p.getNombre());
            lblunidad.setText(p.getUnidad_medida());
            lblstock.setText(String.valueOf(p.getCantidad()));
            lblprecio.setText(String.valueOf(p.getPrecio()));
        }
    }
```

En el botón guardar se realiza la acción que permita mostrar nuevamente los productos al comprador cuando este decida qué productos desea llevar.

```
290
291     private void facturaActionPerformed(java.awt.event.ActionEvent evt) {
292         // TODO add your handling code here:
293         cantidad = Integer.parseInt(text.getText());
294         precio = Double.parseDouble(lblprecio.getText());
295         deposito = Integer.parseInt(lblstock.getText());
296         labelsub.setText("0");
297 
298         if(cantidad <= deposito){
299             sub_total = (cantidad * precio);
300             deposito-=cantidad;
301 
302         }else{
303             reporte = "Stock insuficiente";
304             labelmen.setText(String.valueOf(reporte));
305         }
306 
307         iva = sub_total*(0.12);
308         total = sub_total+iva;
309         tot +=total;
310         labelsub.setText(String.valueOf(sub_total));
311         labeliva.setText(String.valueOf(iva));
312         labeltotal.setText(String.valueOf(total));
313         labeltot.setText(String.valueOf(tot));
314 
```

Acciones realizadas una vez que se acciones el botón facturar.

```
// TODO add your handling code here:
cantidad = Integer.parseInt(text.getText());
precio = Double.parseDouble(lblprecio.getText());
deposito = Integer.parseInt(lblstock.getText());
labelsub.setText("0");
```

Se debe recoger información del usuario acerca de cuantos productos quiere desea llevar.

Dentro de las variables precio y deposito guardamos una vez que se muestra al comprador los productos que busca, el precio que fue guardado anteriormente dentro del registro del stock, de igual manera a la cantidad de productos que se encuentran almacenados, también se hace una limpieza de la cantidad de productos que el usuario decidió llevar.

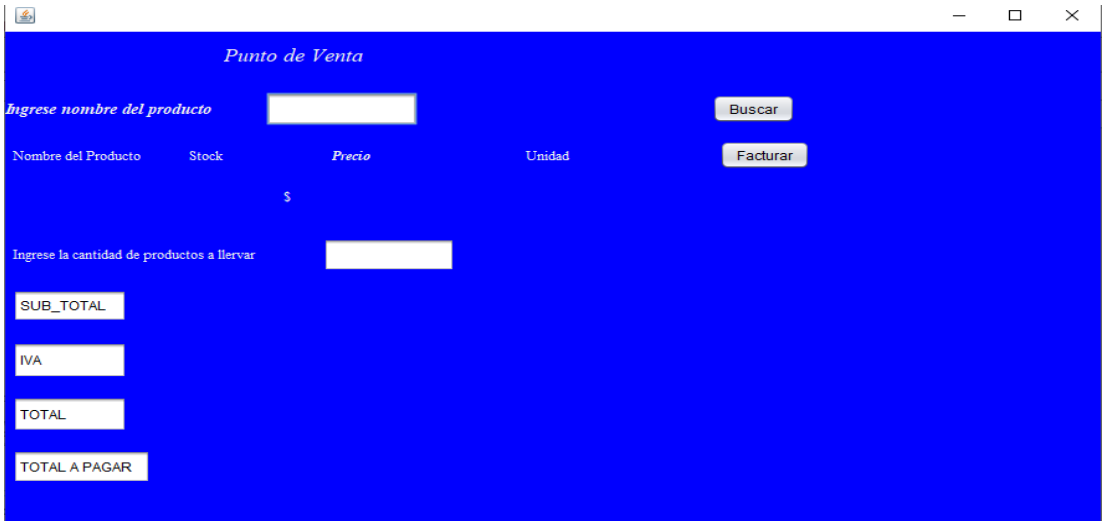
```
if(cantidad <= deposito){
    sub_total = (cantidad * precio);
    deposito-=cantidad;

}else{
    reporte = "Stock insuficiente";
    labelmen.setText(String.valueOf(reporte));
}
```

Se realiza un condicional ya que en caso de que el comprador solicite mas productos que los que se encuentran en el stock aparezca un mensaje que mencione que el stock es insuficiente.

```
iva = sub_total*(0.12);
total = sub_total+iva;
tot +=total;
labelsub.setText(String.valueOf(sub_total));
labeliva.setText(String.valueOf(iva));
labeltotal.setText(String.valueOf(total));
labeltot.setText(String.valueOf(tot));
```

Se realiza el calculo de subtotal, iva y total de cada producto, adicionalmente se realiza un calculo del total a pagar de todos los productos al comprador.



Interfaz final del JFrame Ventas.

Enlace del video.

https://youtu.be/88otQMOsZ5E