**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**



Programación Orientada a objetos

Proyecto Poo

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Autor:**

David Andrés Pillco Y.

**Loja- Ecuador**

**2020**

Contenido

[Dirección del proyecto en **github:** 3](#_Toc35450979)

[Definición del problema 3](#_Toc35450980)

[Análisis del problema 3](#_Toc35450981)

[Entrada: 3](#_Toc35450982)

[Proceso: 3](#_Toc35450983)

[Salida: 3](#_Toc35450984)

[Diseño 3](#_Toc35450985)

[ Diagrama de clases 4](#_Toc35450986)

[ Pseudocódigo 4](#_Toc35450987)

[Explicación código interfaz 6](#_Toc35450988)

[Explicación Interfaz gráfica: 10](#_Toc35450989)

# Dirección del proyecto en **github:** <https://github.com/davidpillco/proyectopoo>

# Definición del problema

Programa para de una tienda la cual tenga u menú que permita al usuario agregar los productos al stock, ya sea comestible, utensilio, oficina, industriales. Para que luego de ello realizar la venta de productos y al final generar el subtotal, el iva y el total a pagar por los productos ingresados.

# Análisis del problema

## Entrada:

* Ingreso por parte del usuario si desea registrar (jblRegistrar), vender (jblVender) un producto o salir.
* Si desea ingresar un producto, pedir el nombre (nombre), el tipo de producto (tipo\_producto) , cantidad a ingresar (cantidad) y el precio por unidad (precio).
* Luego de ello si desea el usuario, ir a ventas, en el cual primero se busca el producto en el stock, si existe se lo agrega colocando el nombre, la cantidad y el precio.
* Y finalmente luego del ingreso de todos los productos se presenta el subtotal, iva y el precio total.

## Proceso:

* Almacenar todos los productos (nombre, tipo, cantidad, precio) en un objeto, el cual a su vez es enviado a un **arrayList (productos)** el cual pertenece al stock.
* En la parte de ventas busca el producto ingresado en el stock y si es el caso proceder a la venta de cada producto, el cual es almacenado en otro **arraylist (tabla\_facturacion)** para presentarlos en la tabla de facturación.
* Calculo del subtotal, iva y precio total de todos los productos ingresados:
  + Subtotal: Recorre el arraylist y se suma todos los precio (total\_factura)
  + iva = total\_factura \* 0.12
  + precio\_total = total\_factura + iva

## Salida:

* Se presenta todos los productos (tabla\_facturacion), el subtotal (total\_factura), el iva (iva) y el precio total (precio\_total).

# Diseño

## Diagrama de clases

|  |
| --- |
| Venta\_productos |
| nombre  cantidad  precio |
| Venta\_productos()  getNombre()  setNombre()  getCantidad()  setCantidad()  getPrecio()  setPrecio() |

|  |
| --- |
| Producto |
| nombre  tipo\_producto  cantidad  precio |
| Productos()  getNombre()  setNombre()  getTipo\_producto()  setTipos\_producto()  getCantidad()  setCantidad()  getPrecio()  setPrecio() |

## Pseudocódigo

Paquete puntodeventa

Clase **Producto**

1.- Declaración de variables

nombre: cadena

tipo\_producto: cadena

cantidad: decimal

precio: decimal

2.- Método **Producto (**nombre: cadena, tipo\_producto: cadena, cantidad: decimal, precio: decimal**)**

this.nombre = nombre

this.tipo\_prodcuto = tipo\_producto

this.cantidad = cantidad

this.precio = precio

Fin método Producto

3.- Método getNombre()

retorna nombre

Fin método getNombre

4.- Método setNombre(nombre: cadena)

this.nombre = nombre

Fin método setNombre

5.- Método getTipo\_producto()

retorna tipo\_producto

Fin método getTipo\_producto

6.- Método setTipo\_producto (tipo\_producto: cadena)

this. tipo\_producto = tipo\_producto

Fin método setTipo\_producto

7.- Método getCantidad()

Retorna cantidad

Fin método getCantidad

8.- Método setCantidad (cantidad: decimal)

this.cantidad = cantidad

Fin método setCantidad

9.- Método getPrecio()

retorna precio

Fin método getPrecio

10.- Método setPrecio (precio: decimal)

this.precio = precio

Fin método setPrecio

Fin clase **Producto**

Clase **Venta\_productos**

1.- Declaración de variables

nombre: cadena

cantidad: decimal

precio: decimal

2.- Método **Venta\_productos (**nombre: cadena, cantidad: decimal, precio: decimal**)**

this.nombre = nombre

this.cantidad = cantidad

this.precio = precio

Fin método Venta\_productos

3.- Método getNombre()

retorna nombre

Fin método getNombre

4.- Método setNombre(nombre: cadena)

this.nombre = nombre

Fin método setNombre

5.- Método getCantidad()

Retorna cantidad

Fin método getCantidad

6.- Método setCantidad (cantidad: decimal)

this.cantidad = cantidad

Fin método setCantidad

7.- Método getPrecio()

retorna precio

Fin método getPrecio

8.- Método setPrecio (precio: decimal)

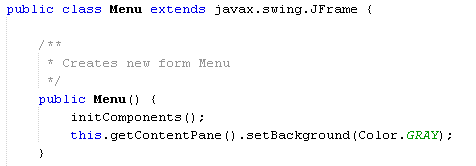
this.precio = precio

Fin método setPrecio

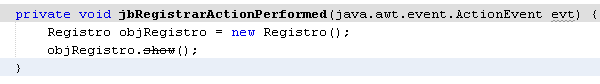
Fin clase **Venta\_productos**

# Explicación código interfaz

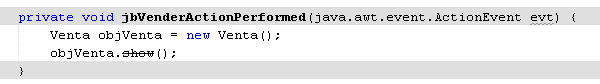
**Menu:**



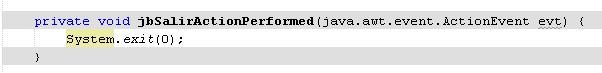
Como se puede ver se emplea la herencia con el JFrame para la interfaz, también el uso del getContentPane para el color.



Método para el botón de Registrar el cual nos lleva a **Registro.**

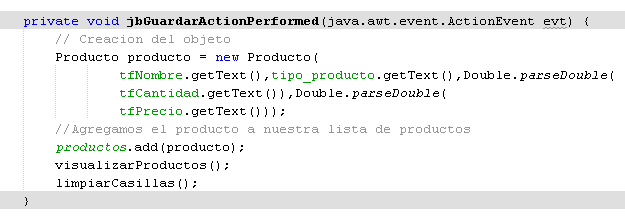


Método para el botón vender el cual nos lleva a Venta.

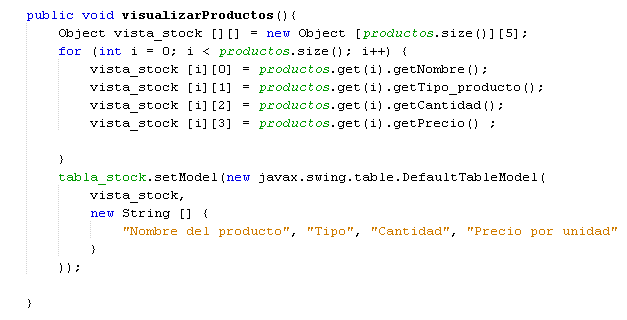


Método para el botón salir el cual cierra todo el proceso

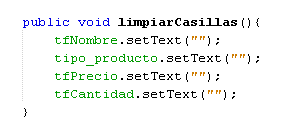
**Registro:**



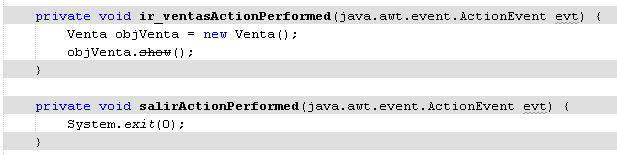
Código para el botón de guardar, aquí se crea el objeto y se guarda el ingreso que hace el usuario y se lo almacena en el arraylist de productos. Luego de ello se ejecuta el método visualizarProductos(), y el de limpiar casillas().



En visualizaciónProductos se crea una matriz objeto (vista\_stock) y se crea un for para recorrerlo e ir guardando el ingreso del nombre, el tipo, la cantidad y el precio del producto y con el el tabla\_stock.setModel se le envía vista\_sotck para que se presente en la tabla que tiene la interfaz.

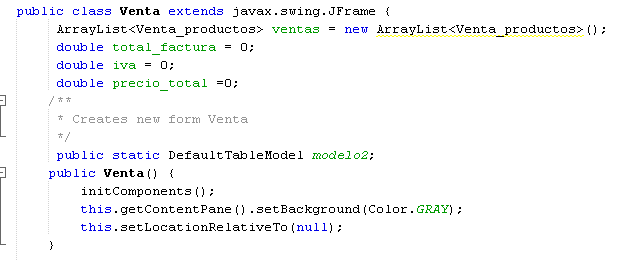


Y el método limpiarCasillas el cual va limpiando donde ingresa la información el usuario para el siguiente ingreso.



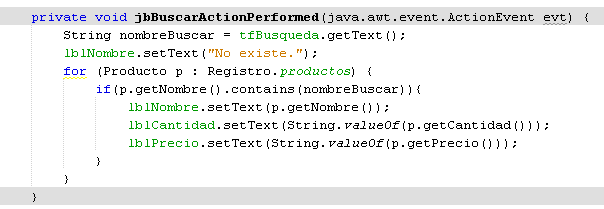
Además un botón para que el usuario vaya directo a la venta de productos y si no es el caso, la posibilidad de salir si solo quiere agregar productos al stock.

**Venta:**

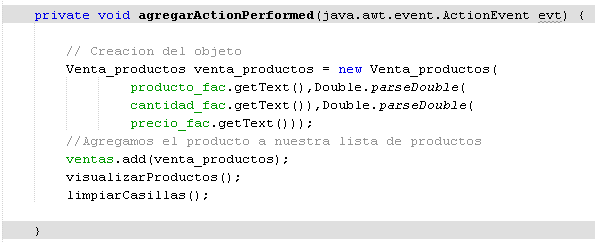


En lo que respecta a venta, se crea otro arraylist que en este caso es ventas y se declaran variables que se emplearan después como es el iva, el total\_factura y el precio total, las cuales se las inicializa.

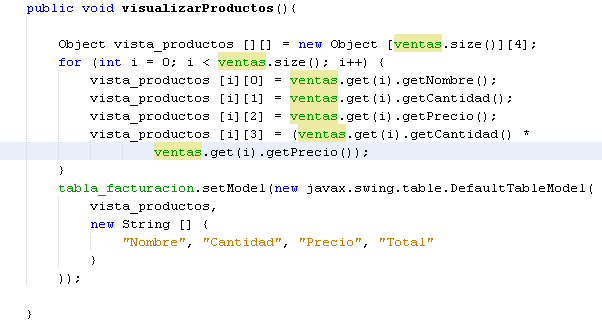
Y después de igual manera se le coloca el color gris para la interfaz del usuario.



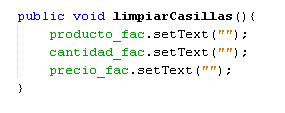
Luego de ello se tiene el botón de **Buscar,** aquí se pide el nombre del producto a buscar y con el for se recorre para presentar si existe el producto junto con la cantidad y su precio.



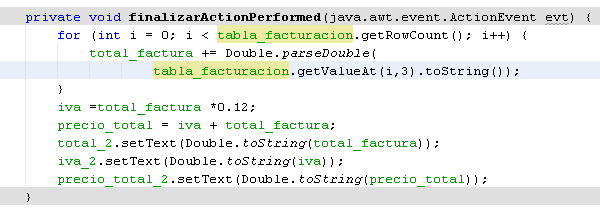
Si el caso que si existe el producto se procede a ingresar el nombre, el precio y la cantidad que desea comprar el usuario y con el botón agregar se lo guarda en venta\_productos y se lo envía a ventas para después ir mostrando con visualizacionProductos.



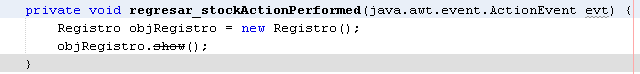
En lo que respecta visualización productos se crea un objeto vista\_productos para guardar todos los productos ingresados y con el for se los recorre y se los presenta con el empleo de tabla\_facturacion.setModel.y se le envia vista\_productos.



De igual forma se utiliza el limpiar casillas para ingreso de producto a vender.



También se implementó el botón finalizar el cual con el for se recorre la tabla con todos los productos y se los suma para obtener el total\_factura que es el subtotal y luego en iva se mulitplica el total de todos los productos por 0.12, y el precio total el cual es la suma del iva más el total\_factura y finalmente se lo envía a total\_”, iva\_2 y precio\_total\_2 los cuales son para que se presenten en la interfaz.



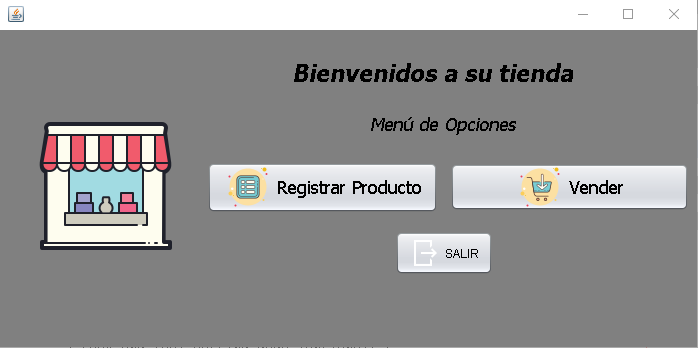
También existe la posibilidad de regresar al registro para el ingreso de mas productos.



Y finalmente un botón salir para finalizar todo el proceso.

# Explicación Interfaz gráfica:

Al inicio del programa se presenta la venta principal



Aquí el usuario puede escoger la opción de ir al registro de producto, es decir agregar productos al stock, vender que nos lleva a la venta de los productos o si simplemente quiere salir también existe la posibilidad.

Si escoge el usuario **Registrar Producto** nos saldrá la siguiente ventana:

En esta ventana de registro producto el usuario tiene que ingresar el nombre del producto, el tipo, la cantidad y el precio del producto a ingresar en la parte izquierda, cada que ingrese se presentará en la tabla de la derecha cada uno de los productos. Cuando haya finalizado puede salir o ir a ventas.

Si es el caso de que escoge ventas saldrá la siguiente ventana:



En la parte superior izquierda en **búsqueda en el stock** el usuario ingresa ahí el producto y le da al botón buscar para ver si existe el producto en el stock, si es el caso procedemos a **Productos a vender** en donde se ingresa el nombre del producto, la cantidad a comprar tomando en cuenta la cantidad que hay en el stock que representó anteriormente y también el precio y se le da el botón de **agregar producto** y se presenta en la tabla facturación que esta en la parte derecha. Una vez ingresado todos los productos se puede regresar al stock o si no se da al botón de **finalizar venta** para que así se presente el subtotal, el iva y el precio total a pagar por todos los productos ingresados.

Y finalmente un mensaje de **Gracias por su compra !!!** 😊