**Universidad Técnica Particular de Loja**



**Proyecto Final Programación Orientada a Objetos**

Santiago David García Jaén

**Docente:**

Richar Stalin Guaya Delgado

**Carrera:**

Ciencias de la Computación

**Área Técnica**

**Loja-Ecuador 2020**

**Punto de Venta:**

**Análisis del Problema.**

Con el uso de datos tipo Objeto y Listas, realizar un punto de venta con los siguientes requerimientos.

El programa debe presentar un menú de opciones:

1. Registrar stock de productos; se debe tener la posibilidad de almacenar los productos en una lista. Los productos pueden ser de varios tipos: Comestibles, Utensilios, Oficina, Industriales
2. Realizar venta; se tiene que ingresar: la cantidad, el producto a comprar y con ello se tiene que verificar si existe en el stock ingresado previamente para poder agregarlo a la compra. Además, al realizar la venta, se tiene que ir presentando el subtotal, el iva generado y el total a pagar.
3. Salir.

**Definición del problema.**

**Entrada**:

Cadena: nombre, peso, tipo\_material, marca, tipo

Entero: cantidad

Real: precio

**Proceso**:

Lista de Objetos Comprados = nombre, detalle, cantidad\_ingresada, precio

Real subTotal = 0;

Real iva;

Real precioTotal;

Para(Comprado sumaCompras: Comprados) Hacer:

subtotal = subtotal + (sumaCompras.obtener.cantidad \* sumarCompras.obtener.cantidad\_ingresada)

Fin Para

iva = subTotal \* 0.12;

precioTotal = subTotal \* 1.12;

**Salida:**

Se presentan los datos del iva, subtotal y total obtenidos de los objetos comprados.

**Diseño**.

1. **Diagrama** **de** **Clases**

**Comprado**

detalle

tipo

establecerDetalle ()

obtenerDetalle()

establecerTipo ()

obtenerTipo ()

ConstructorComprado()

**Industrial**

tipo

establecerTipo()

obtenerTipo()

ConstructorIndustrial()

**Oficina**

marca

establecerMarca()

obtenerMarca()

ConstructorOficina()

**Utensilio**

tipoMaterial

establecertTipoMaterial()

obtenerTipoMaterial ()

ConstructorUtensilio()

**Comestible**

peso\_Neto

establecerPeso\_Neto()

obtenerPeso\_Neto()

ConstructorComestible()

**Producto**

nombre

cantidad

precio

iva

establecerNombre()

obtenerNombre()

establecerCantidad()

obtenerCantidad ()

establecerPrecio()

obtenerPrecio()

establecerIva()

obtenerIva()

**2. Pseudocodigo:**

**Algoritmo Punto de Venta**

**Clase abstracta Producto**

1. Variables: nombre: cadena stock: entero precio, iva, subtotal, total: real
2. Metodo establecer nombre (nombre1: cadena):

nombre = nombre1;

Fin Metodo

1. Metodo obtener nombre: cadena

retornar nombre

Fin Metodo

1. Metodo establecer stock (stock1: entero):

stock = stock1;

Fin Metodo

1. Metodo obtener stock (): entero

retornar stock;

Fin Metodo

1. Metodo establecer precio (precio1: real) :

precio = precio1;

Fin Metodo

1. Metodo obtener precio (): real

retornar precio;

Fin Metodo

1. Metodo establecer iva (iva1: real)

iva = iva1;

Fin Metodo

1. Metodo obtener iva (): real

retornar iva

Fin Metodo

1. Metodo establecer subtotal (subtotal1: real)

subtotal = subtotal1;

Fin Metodo

1. Metodo obtener subtotal (): real

retornar subtotal;

Fin Metodo

1. Metodo establecer total (total1: real)

total = total1;

Fin Metodo

1. Metodo obtener total (): real

retornar total

Fin Metodo

**Clase Comestible extensión de Producto;**

1. Variables: peso\_neto: cadena
2. Metodo establecerPeso\_Neto(peso\_neto1: cadena) :

peso\_neto = peso\_neto1;

Fin Metodo

1. Metodo obtenerPeso\_Neto(): cadena

retornar peso\_neto;

Fin Metodo

1. Metodo ConstructorComestible(cadena nombre, entero stock, real precio, cadena peso\_neto)

this.nombre = nombre;

this.stock = stock;

this.precio = precio;

this.peso\_neto = peso\_neto1;

Fin Metodo

**Clase Utensilio extensión de Producto;**

1. Variables: tipo\_material: cadena
2. Metodo establecer tipo\_material (tipo\_material1: cadena):

tipo\_material = tipo\_material1;

Fin Metodo

1. Metodo obtener tipo\_material (): cadena

retornar tipo\_material;

Fin Metodo

1. Metodo ConstructorUtensilio(cadena nombre, entero stock, real precio, cadena peso\_neto)

this.nombre = nombre;

this.stock = stock;

this.precio = precio;

this. tipo\_material = tipo\_material1;

Fin Metodo

**Clase Oficina extensión de Producto;**

1. Variables: marca: cadena
2. Metodo establecer marca (marca1: cadena):

marca = marca1;

Fin Metodo

1. Metodo obtener marca (): cadena

retornar marca;

Fin Metodo

1. Metodo ConstructorOficina (cadena nombre, entero stock, real precio, cadena marca)

this.nombre = nombre;

this.stock = stock;

this.precio = precio;

this. marca = marca1;

Fin Metodo

**Clase Industrial extensión de Producto;**

1. Variables: tipo: cadena
2. Metodo establecer tipo (tipo1: cadena):

tipo = tipo1;

Fin Metodo

1. Metodo obtener tipo (): cadena

retornar tipo;

Fin Metodo

1. Metodo ConstructorIndustrial(cadena nombre, entero stock, real precio, cadena tipo)

this.nombre = nombre;

this.stock = stock;

this.precio = precio;

this.tipo = tipo1;

Fin Metodo

**Clase Comprado extensión de Producto;**

1. Variables: tipo, detalle: cadena
2. Metodo establecer tipo (tipo1: cadena):

tipo = tipo1;

Fin Metodo

1. Metodo obtener tipo (): cadena

retornar tipo;

Fin Metodo

1. Metodo establecer detalle (detalle1: cadena):

detalle = detalle1;

Fin Metodo

1. Metodo obtener detalle (): cadena

retornar detalle;

Fin Metodo

1. Metodo ConstructorComprado(cadena tipo, cadena nombre, entero stock, real precio, cadena tipo)

this.tipo = tipo1;

this.nombre = nombre;

this.detalle =detalle1;

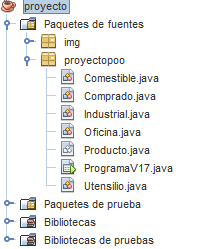
this.stock = stock;

this.precio = precio;

Fin Metodo

**Informe del proyecto**

Para el siguiente proyecto se procedió con la creación de 2 paquetes. El primero llamado “proyectopoo”. El mismo contiene las siguientes clases:

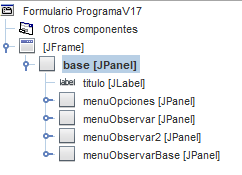


Existen 4 objetos destinados a guardar datos Objetos: Comestible, Utensilio, Oficina, Industrial.

Todas las anteriores son extensión de la clase abstracta, Producto.

Y por último existe un Jframe llamado ProgramaV17, el mismo que ejecutara todo el código.

El Jframe funciona de la siguiente forma:



1. base: JPanel que contiene toda la información del programa y como se indica es una base para el mismo.

2. titulo: Jlabel que muestra el título del programa

3. menuX: Jpanels que contienen diferentes JLabel, JPanel entre otros, para el funcionamiento del programa



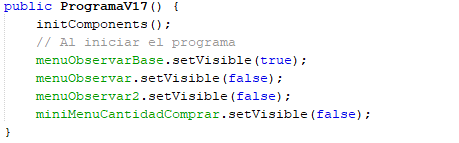
2

1

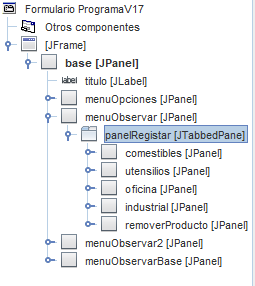
3

3

El programa inicia ocultando los menusXX y mostrando solo uno el “menuObservarBase” el mismo contiene información básica de que ofrece el programa.

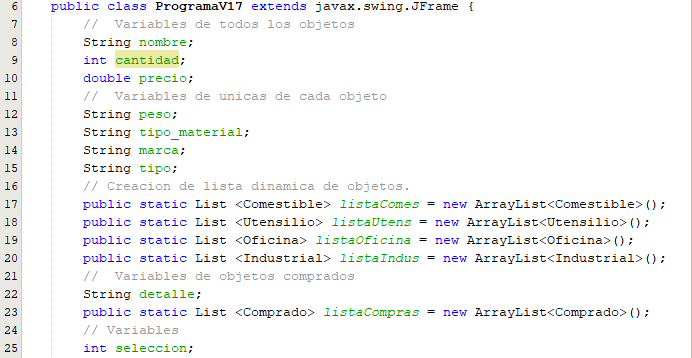


Para ingresar datos se presiona el botón Registrar, el cual presenta el “menuObservar” y contiene un JTabbedPanel, el mismo contiene diferentes JPanels (comestible, utensilio, oficina, industrial) y asu ves contienen diferentes variables de muestra y entrada de datos.

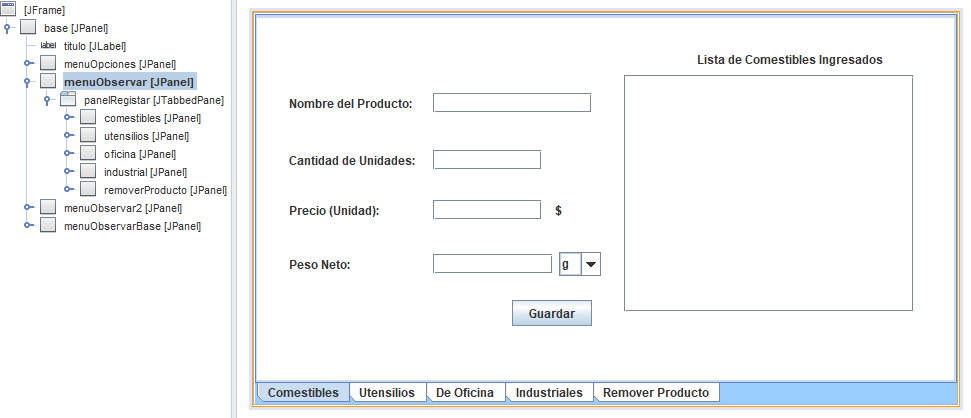


Al inicio de nuestro código se ingresan las variables globales de la clase, las mismas que podremos usar en cada método.

Al igual de listas de objetos que usaremos posteriormente.



Para Ejemplificar la entrada de datos, ya que cada objeto es diferente en algún aspecto, se usará el panel “comestibles”



1

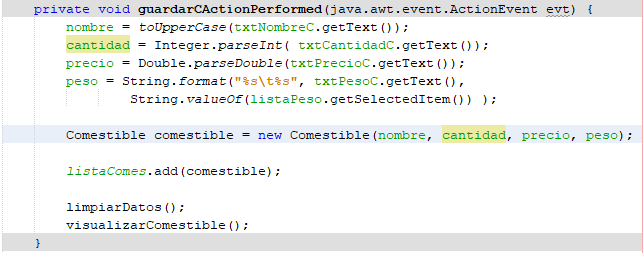
2

3

1. Se ingresa los datos en el campo de texto, por ejemplo “Alimento1” en el campo nombre, el cual tiene el nombre de “txtNombreC”. Se rellenan los demás campos de la misma forma con datos correspondientes sino e mismo generara error. A excepción del campo peso neto que contiene aparte del campo para ingresar la cantidad, una JComboBox “listaPeso” que contiene la información acerca de que valores predeterminados por nosotros.

Para obtener la información acerca de un campo se usa “*nombreVariable.getText()”*

2. Se da clic en el botón guardar respectivo de cada clase en este caso “guardarC” y se usa un método privado vacío. En el mismo establecemos la entrada de datos de nuestras variables globales declaradas con anterioridad.



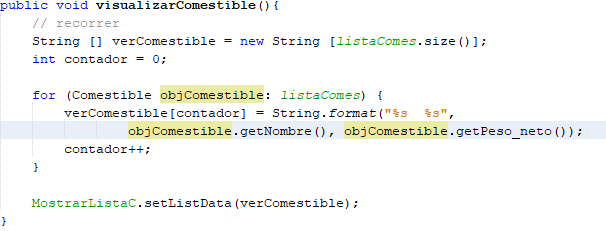
Se crea el objeto en la parte de comestibles, enviando todos los datos obtenidos de los campos de texto al constructor y se añade a la lista respectiva del producto, en este caso comestible.

Se crean dos métodos limpiarDatos() y visualizarComestible().

Los mismos sirven respectivamente para que cuando se de clic al botón guardar, el método limpiarDatos borre la información de los campos de texto y el método visualizarComestible() nos permite visualizar en un la Jlist “MostrarListaC” la información.

En el mismo se crea una lista de tipo cadena del tamaño de la lista de Comestibles, se inicializa un contador y con un For especifico para el objeto, enviamos la información obtenida en formad e cadena, en este caso el aparatado nombre y el apartado peso.

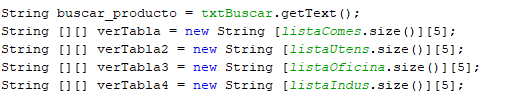
Y finalmente se muestra en “MostrarListaC” ubicada en el panel de comestibles.



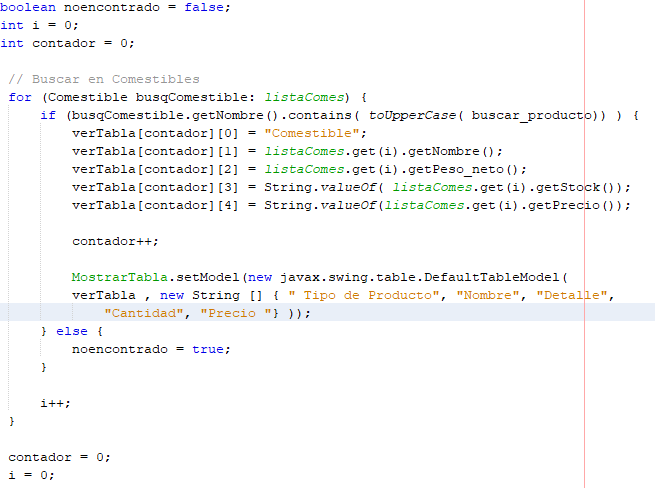
Y para los demás objetos, el programa funcionaria con la misma sintaxis.

Para vender los productos ingresados se presionará el botón vender de la parte derecha, al igual que el botón “Registrar” ocultará los demás paneles de la base y mostrará el panel “menuObservar2”

El botón Buscar obtendrá los datos ingresados en el campo de texto “txtBuscar”. El método de este también creara listas de tipo cadena del tamaño de los diferentes objetos donde ese guardaron los datos.



Para buscar, el método usara lo siguiente:

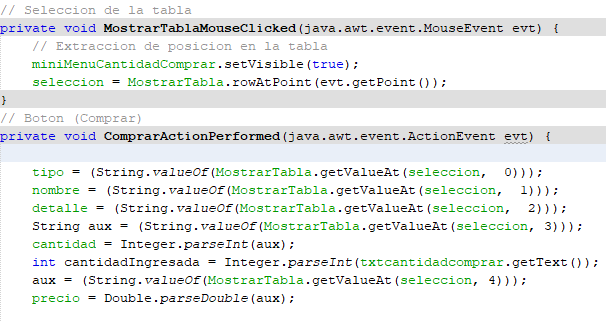


Un boleano para determinar si se encontró el producto, un entero de contador(i)y un segundo(contador). En el for se buscará el producto en las diferentes listas de objetos. Usando un condicional en el cual se evalúa si el objeto contiene la cadena introducida por el usuario, agregará a la lista cadena creada ala inicio, y después se introducirá el valor al JTable.

El mismo for solo funciona para un objeto, por lo que se debe realizar varias para ser buscado en los cuatro objetos (Comestible, Utensilio, Oficina, Tenedor.), pero diferenciando sus variables.

El botón Comprar requiere de un método de clic, es decir, al seleccionar un producto de la tabla se obtendrá la ubicación del producto y sus datos. Al igual que se enviarán dichos datos obtenidos y transformados a cadena a una variable de tipo cadena.

Algunos de estos datos no pueden ser transformados directamente a cadena, por lo que se optó por usar un auxiliar par la resolución del problema.

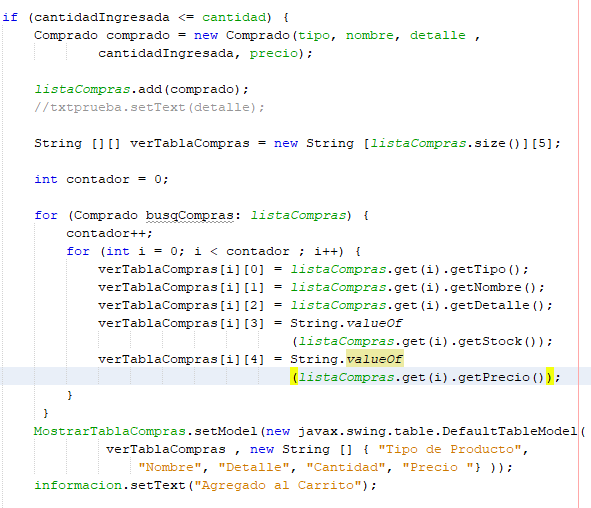


Se usará un condicional con el fin de determinar si la cantidad de unidades del producto que el usuario desea adquirir está en stock.

Si la condición es verdadera, se creará el Objeto “Comprado” con los datos obtenidos de las variables que a su vez extrajeron la información de la tabla.

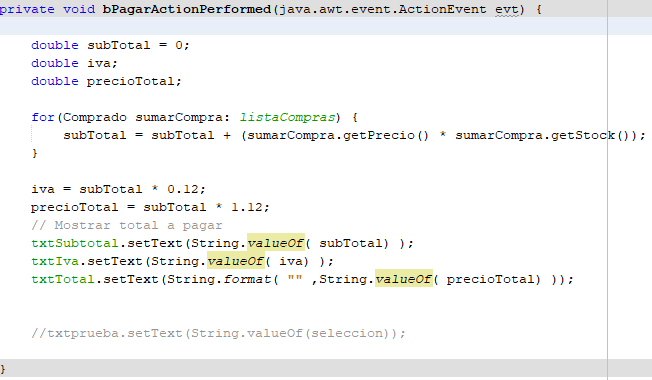
Y usando un For para el producto comprado, se lo extraerá en una nueva lista de tipo cadena que llenará una siguiente tabla, la misma ubicada en el panel “Toral a Pagar”.

Finalmente se mostrará una información “Agregado al Carrito” y en caso de que el condicional sea falso, mostrara “No existe stock del producto.”



Finalmente en el JPanel ”Total a pagar”, se mostrara la lista de objetos de “listaCompras”,

Y con el botón Pagar se efectuara el calculo mediante la siguiente Forma:



Se usan variables de tipo real para almacenar los datos.

En un For especial para lista de objetos “listaCompras”, se determina el valor del subtotal, que nos mas sino la suma de los precios de todos los objetos requeridos por por su precio, esto para todo los clientes, es por ello necesario el uso del For.

Para Finalizar el programa se establecen las cantidades obtenidas en las variables de tipo Jlabel.