



---

---

## *Facultad De Estudios Superiores Aragón*

*Materia*  
*Programación Orientada A Objetos*

*Proyecto Final*  
*Sistema de registro y administración de*  
*productos para una tienda de abarrotes*

*Profesor*  
*Miguel Ángel Sánchez Hernández*

*Autor*  
*Ernesto Missael Contreras Emigdio*



# Índice

<b>1. Objetivos del proyecto -----</b>	<b>3</b>
1.1 Descripción de los objetivos	
<b>2. Introducción -----</b>	<b>3</b>
2.1 JavaFX	
2.2 Programación orientada a objetos	
2.3 Los principales componentes de la programación orientada a objetos	
2.4 La interfaz gráfica – GUI	
<b>3. Desarrollo -----</b>	<b>4</b>
3.1 Distribuidor	
3.2 Inventario	
3.3 Producto	
3.4 Diagrama total de la aplicación	
<b>4. Funcionalidad -----</b>	<b>6</b>
4.1 Pantalla inicial	
4.2 Pantalla nuevo	
4.3 Pantalla inventario	
4.4 Pantalla distribuidor	
4.5 Botones	
4.6 Pantallas (Modificar)	
<b>5. Conclusiones -----</b>	<b>9</b>
5.1 Conclusión general	
<b>6. Bibliografía -----</b>	<b>10</b>

# **1. Objetivos del proyecto**

## **1.1 Descripción de los objetivos**

- Registrar y administrar el inventario de productos de una tienda
- Creación de una aplicación para el registro y la administración de los productos, utilizando JavaFX
- Posibilidad de guardar, abrir y editar el inventario de productos a gusto del usuario, por medio de archivos
- Ofrecer una interfaz amigable para el usuario, utilizando JavaFX y SceneBuilder

## **2. Introducción**

Este proyecto se realizó utilizando programación orientada a objetos en JavaFX, utilizando SceneBuilder para facilitar la creación de la interfaz gráfica (GUI), pero primero se tiene que saber que es la programación orientada a objetos y que es GUI.

### **1.1.JavaFX**

Es un lenguaje de programación derivado de Java, el cual nos facilita la implementación desarrollo y despliegue de aplicaciones con interfaz gráfica, incluyendo audio y video, JavaFX resulta ser muy estable a la hora del desarrollo e implementación, de aplicaciones de tipo GUI

### **1.2.programación orientada a objetos**

La programación orientada a objetos es una rama de la programación que tiene características peculiares, ya que ofrece un paradigma más acertado al pensamiento humano la base esta es la abstracción, encapsulamiento, modularidad y jerarquización, por medio de estas se pueden obtener las características esenciales de un proceso, también como la descomposición de un sistema a su vez privatizar y ocultar las características de sí mismo, incluso ordenando jerárquicamente todos los aspectos anteriores para crear un sistema práctico, eficiente y muy funcional.

### **1.3.Los principales componentes de la programación orientada a objetos**

La programación orientada a objetos se compone de la creación de “Clases” las cuales tiene la peculiaridad de describir estructuras que contiene datos y funcionalidades asociadas, los “Objetos” son la representación ejemplar del comportamiento y estructuración que llega a poseer o definir una clase.

Los “Atributos” son datos que ayudan a definir el como llegan a funcionar todos los objetos de una clase, mientras tanto los “Métodos y Mensajes” son los encargados de definir el como se comportara cada objeto, tanto como el llamado y inicialización finalmente la “Herencia y Polimorfismo”, los cuales nos ayudan a compartir las características de una clase y a compartir la funcionalidad de uno o varios objetos.

### **1.4.La interfaz gráfica o GUI**

Interfaz gráfica se refiere a la posibilidad de crear ventanas, botones, menús demás componentes gráficos, básicamente representar la información de un programa mediante estas características, las cuales nos terminan brindando un entorno visual.

### 3. Desarrollo

Se utilizará el entorno de programación Eclipse en conjunto de JavaFX y SceneBuilder para la creación de una manera más dinámica de la interfaz gráfica, para comenzar con el desarrollo de esta primeramente se ocupa definir un modelo donde se muestren las clases que se van a utilizar, el modelo que se piensa utilizar contempla 3 clases principales las culés son (Producto, Inventario y Distribuidor).

#### 3.1. Distribuidor

Esta clase se encarga de guardar datos o también llamados atributos de tipo (nombre, empresa, repartidor, correo, rfc y teléfono), cada uno de estos datos poseen su método get y set el cual nos da la oportunidad de modificar y obtener los datos que contengan o se le quieran asignar, la clase "Distribuidor" es la que almacena los distintos inventarios que la clase "Inventario".

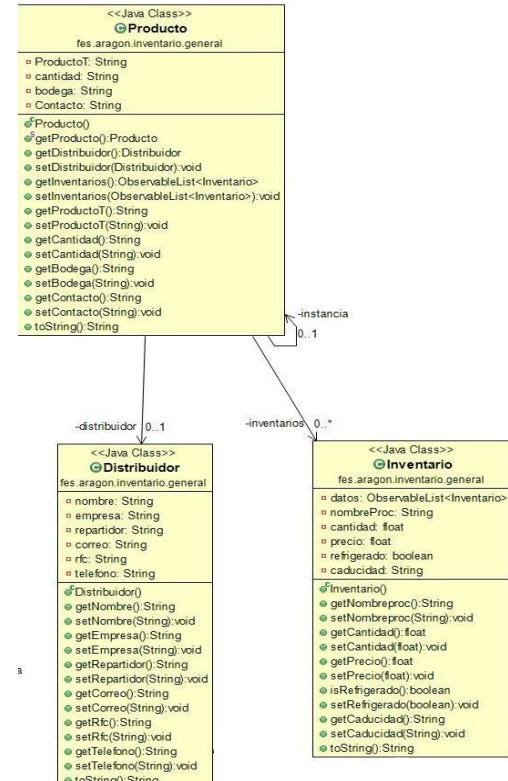
A diferencia que distribuidor no posee la característica de ser un "Array" lo cual significa que no se crearan múltiples de estas mismas, este no tiene por qué serlo, por ejemplo yo quisiera agregar el producto refresco, como se sabe no existe una marca únicamente, pero si un distribuidor o empresa que vente estos productos por sí misma.

Si se quisiera agregar refresco podría ser Fanta, Coca cola, Squirt, etc. Estas marcas pertenecen a FEMSA (El cual es el distribuidor), así que solo basta agregar a FEMSA y sus datos, pero se necesita asignarle la cantidad y marca de los distintos refrescos, de lo cual ahí se utilizara la clase "Inventario" al igual que los refrescos el resto de los productos también pertenecen a corporativos similares, por lo tanto, siempre bastara un solo distribuidor para múltiples inventarios.

#### 3.2. Inventario

Se encarga de guardar los múltiples productos existente dentro de una abarrotera y esta posee los siguientes atributos (nombre, cantidad, precio, refrigerado y caducidad), estos poseen su método get y set el cual nos da la oportunidad de modificar y obtener los datos que contengan, a su vez también su constructor vacío y su método toString.

En este caso la clase en conjunto de su controlador, se convierten en un "Array" el cual permite generar múltiples datos dentro de un campo (Distribuidor), con la finalidad de poseer múltiples productos de un mismo tipo (galletas, refresco, dulces), sin la necesidad de estar creando un campo nuevo para guardar cada vez que la marca cambie, aquí solo bastara ingresar un "Distribuidor" y de ahí designarle un corporativo que posea los productos respetivos a registrar.

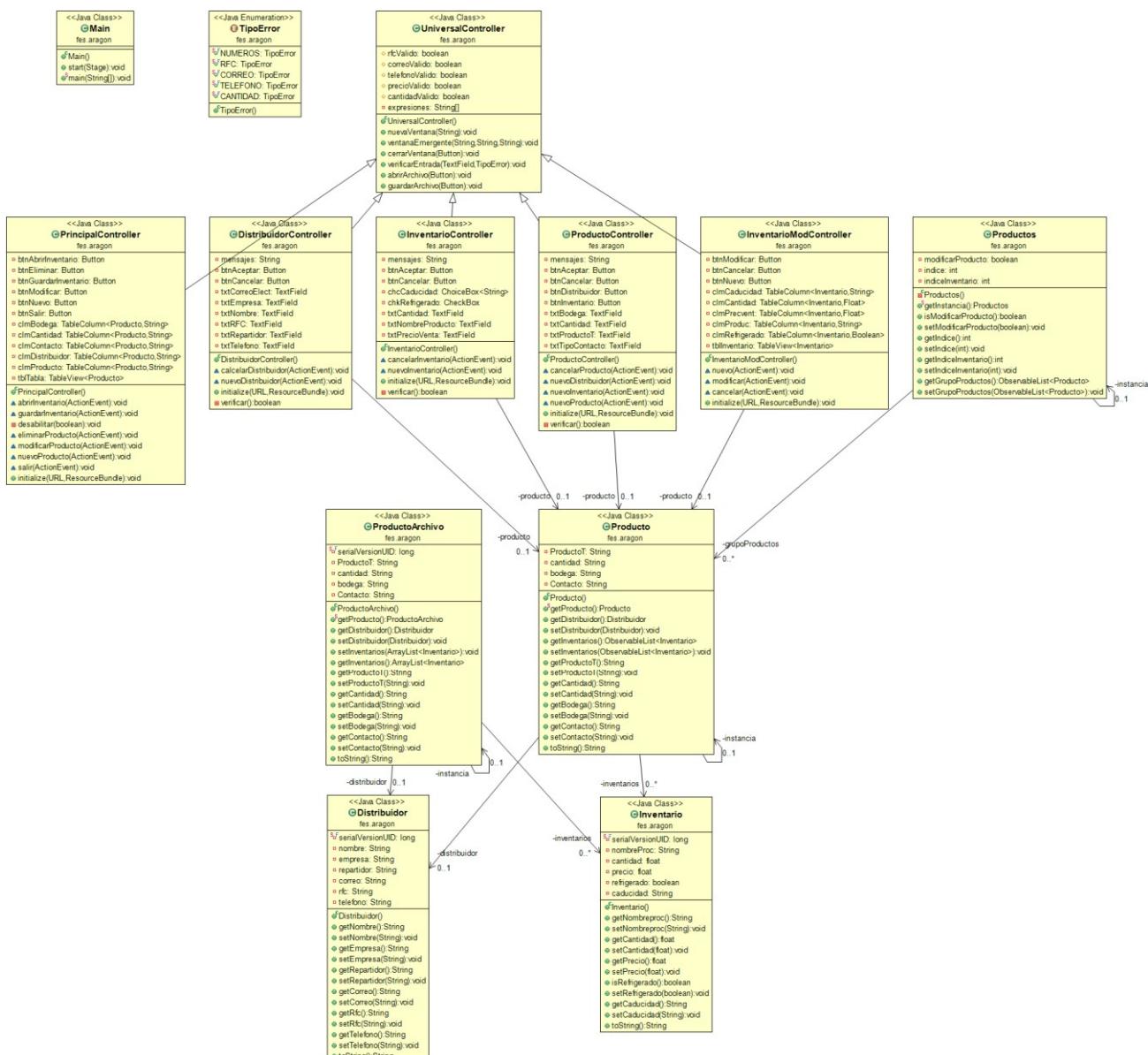


### 3.3. Producto

Se encarga del guardar los datos de la clase “Inventario” y la clase “Distribuidor”, la principal función de “Producto” es ser el campo donde se van a mostrar los datos, guardar y dar el acceso al usuario para editar o agregar más productos, este posee los siguientes atributos (ProductoT, cantidad, bodega, contacto), los cuales solo son llamados a los atributos de las otras dos clases anteriormente explicadas.

Estos llamados poseen su método get y set el cual nos da la oportunidad de modificar y obtener los datos que contengan, estos métodos permiten acceder modificar y ingresar datos cuando sea necesario, a su vez también implementan herencia para aumentar mas la funcionalidad de las clases y poder mostrar los valores que se quieran en la interfaz inicial.

### 3.4. Diagrama total de la aplicación



## 4. Funcionalidad

### 4.1 Pantalla inicial

En esta se puede observar el programa a la hora de ejecutarse, la ventana posee distintos botones como se puede apreciar, de los cuales (Guardar Inventario, Modificar y Eliminar) están completamente deshabilitados ya que esto ayuda a evitar problemas en la aplicación



### 4.2 Pantalla nuevo

Esta pantalla aparecerá cuando se apriete el botón nuevo de la ventana inicial, el cual al apretarlo este ejecutara un llamado por medio de la acción nuevo producto, el cual ejecutara un conjunto de instancias que llamarán al controlador “Productos” para que este se encargue de recibir la información que se le proporcione en sus entradas de datos, a su vez que también iniciara la llamada de la ventana (Producto), donde se podrán ya ingresar datos

#### 4.3 Pantalla inventario

Esta ventana se desplegará después de presionar el botón “inventario” el cual hará la llamada del método “if”, el cual determinará si se está acezando desde la ventana para agregar o modificar, en este caso al entrar desde el método agregar hará el llamado por medio de “nuevaVentana” al controlador y archivo fxml de inventario, así mostrando la interfaz de inventario.

The screenshot shows a window titled "Inventario". It contains several input fields and a checkbox. On the left, there are labels: "Nombre del producto:", "Cantidad:", "Precio de venta", "Producto refrigerado", "Caducidad", and "Tiempo estimado". To the right of these labels are corresponding input fields or dropdown menus. A checkbox labeled "¿El producto es refrigerado?" is positioned between "Producto refrigerado" and "Caducidad". Below "Caducidad" is a dropdown menu labeled "Selecciona la caducidad". At the bottom right are two buttons: "Aceptar" and "Cancelar".

#### 4.4 Pantalla distribuidor

Esta pantalla al igual que Inventario se utilizarán los mismos métodos y llamados y comparaciones para determinar desde donde se aceza, a diferencia que los datos ingresados cambiaran en donde se guardan.

The screenshot shows a window titled "Distribuidor". It contains six input fields with labels on the left: "Nombre:", "Repartidor:", "Empresa:", "Correo Electronico:", "RFC:", and "Teléfono:". Each label is followed by a corresponding input field. At the bottom right are two buttons: "Aceptar" and "Cancelar".

## 4.5 Botones

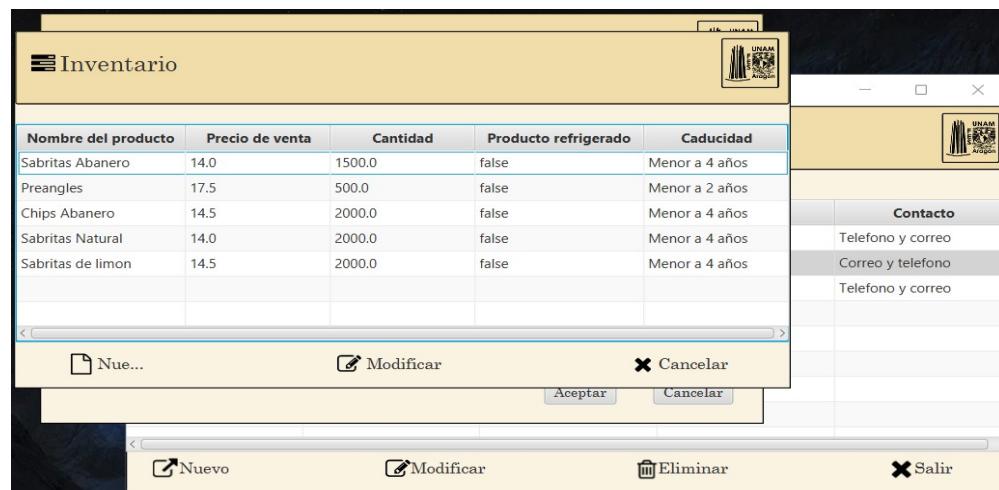
Los botones “Aceptar y Cancelar”, funcionan por medio del método “cerrarVentana” el cual se encarga de cerrar la ventana en la cual se haya presionado el botón “Aceptar o Cancelar”, el botón salir simplemente hace el llamado al método “salir” el cual iniciara la acción de cerrar totalmente la aplicación

Los botones “Guardar y Abrir” estos son utilizados para cargar datos a la aplicación por medio del “guardarArchivo”, perteneciente a la clase “File” la cual entra en funcionamiento al presionar el botón “Guardar” este procederá a crear un archivo con una extensión personalizada, por medio de serialización el cual se podrá acezar posteriormente la aplicación se haya vuelto a ejecutar.



## 4.6 Pantallas (Modificar)

Estas pantallas aparecerán a la hora de haber seleccionado una fila y presionar el botón de modificar, este botón funciona a una manera similar que el de guardar solamente que, no crea un nuevo segmento de memoria donde guardar los datos al contrario, este accede a los datos ya existentes otorgándote la posibilidad de modificar los datos que uno quiera.



## **5. Conclusiones**

### **5.1Conclusion general**

Se logro registrar con éxito los productos y sus distintas variaciones, dentro de la aplicación desarrollada con tecnología JavaFX y su complemento SceneBuilder, logrando darle al usuario una interfaz agradable tanto a la hora de ejecutarse como, a la hora de observarla ya que se utilizaron colores que fueran amigables con la vista del usuario.

Se logro que la aplicación desarrollada en JavaFX funcionara perfectamente incluyendo la posibilidad de guardar los productos para posteriormente modificarlos o en su caso agregar mas o definitivamente eliminar los que no sean necesarios, esto se logro por medio del uso de “Archivos” de java utilizando la clase “File”

Se esperaba lograr aun una manera más dinámica a la hora de visualizar las tablas con las cantidades de los productos, pero el método implementado generaba conflicto, en el momento que se querían guardar datos nuevos, únicamente a la hora de guardarlos, su funcionalidad se comprobó con éxito, aunque aun no se sabe exactamente que provoca ese conflicto, en el código en “InventarioModController” se quedo comentada una parte del método utilizado para cambiar el estilo de la tabla, mientras que en la clase “Inventario” se queda comentada la otra parte del método que genero conflicto, independiente de ese estilo dinámico (Deshabilitado) la aplicación funciona perfectamente

Se buscaba desarrollar una aplicación para el registro administración de productos de una abarrotera, la cual fue lograda con éxito, funcionando en su totalidad, yaqué se cuenta con la opción de (Registrar, Modificar, Eliminar, Guardar, Consultar y Abrir) los datos que se otorgaron, cumpliendo sus funcionalidades en su totalidad, aunque con un 80% de lo esperado en el diseño grafico de esta, por la incompatibilidad que hubo entre los estilos dinámicos y la acción de “Guardar Inventario”

## Bibliografía

Deitel. P, Ditel, H. (2008). *Como programar en java* (7 ed.). (ISBN, Ed.) México: Education, Pearson.

Deitel. P,Ditel, H. (2016). *Como programar en java* (10 ed.). (ISBN, Ed.) México: Education, Pearson

<https://sites.google.com/site/manualpoo/home/interfaz-grafica>

<https://www.java.com/es/download/help/javafx.html>

<https://docs.oracle.com/javase/8/docs/api/index.html>

<https://www.javatpoint.com/javafx-tutorial>

[https://www.etsisi.upm.es/sites/default/files/curso\\_2013\\_14/MASTER/MIW.JEE.POJ.pdf](https://www.etsisi.upm.es/sites/default/files/curso_2013_14/MASTER/MIW.JEE.POJ.pdf)

<https://code.makery.ch/es/library/javafx-tutorial/part4/>

<http://www.w3big.com/es/java/java-regular-expressions.html>