

Algoritmos y Programas

Reporte del Proyecto "Farmacia"

Mariana Arroyo Muñoz

197344

Profesora Ana Eugenia Díaz

Rosiñol

05 / 12 / 2020



INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Índice

Tabla de contenido

Resumen

1. Introducción	
1.1 Contexto	3
1.2 Identificación del problema	3
1.3 Objetivos	4
1.4 Organización del documento	4
2. Análisis	
2.1 Requisitos funcionales	5
2.2 Restricciones	5
3. Diseño	
3.1 Descripción completa del diseño	6
3.2 Estándares utilizados (UML).....	7
4. Implementación	
4.1 Descripción completa de la implementación	14
4.2 GUI	15
4.3 Manual de usuario	16
5. Pruebas y resultados	
5.1 Descripción de las pruebas aplicadas	23
5.2 Resultados obtenidos	23
5.3 Análisis de los resultados	23
6. Conclusiones	26
Referencias	26
Apéndice (código)	27

1. Introducción

1.1 Contexto

En esta temporada de pandemia, la "Farmacia San Pablo" se ha enfrentado a una demanda grande de medicamentos. Varios clientes piden diferentes tipos de medicinas para curar malestares, para dormir, para la temperatura, gripe etc.

1.2 Identificación del problema

El sistema operativo de la Farmacia San Pablo es ineficiente, ya que no brinda la información necesaria para dar respuesta a los clientes como, por ejemplo:

Como medida preventiva, el director decidió que los medicamentos no estuvieran exhibidos para que los clientes no estuvieran contacto con los productos para no contaminarlos o, por el contrario, contaminarse debido a que otro cliente lo haya tocado. Por lo tanto, cada vez que un cliente pide un producto, los empleados tienen que ir a la bodega para corroborar si está en existencia dicho medicamento.

Por otro lado, el director no ha podido tener un control de las ventas de la Farmacia San Pablo debido a la gran cantidad de pedidos que se hacen, ya sea a domicilio, teléfono o en la sucursal.

El director desea un programa que le permita buscar si los productos están existencia en la bodega sin necesidad de irlo a buscar. Además, desea saber las ventas totales hasta el momento, el mes en el que menos se vendió (para poder promocionar los medicamentos y algunas ofertas), el mes en el que más se vendió, las ventas promedio y el mes que tuvo ventas arriba del promedio.

1.3 Objetivos

Hacer un programa con un software de calidad que permita manejar la "Farmacia San Pablo" de manera efectiva y eficiente, de tal manera que resuelva el problema de la existencia de medicamentos, las ventas totales, mes con mayores ventas, mes con menores ventas, ventas promedio y mes con ventas arriba del promedio.

1.4 Organización del documento

En el presente documento, resolveremos estos problemas mediante una explicación dividida en 5 partes:

- En el **análisis** veremos lo que requiere el software para resolver los problemas y las restricciones necesarias para su buen funcionamiento.
- En el **diseño** se dará una descripción del diseño de cada clase y los estándares utilizados (UML y diagramas de flujo)
- Posteriormente, en la **implementación** se dará una descripción de la razón del diseño del programa y de las interfaces gráficas. Además, tendrá incluido un manual de usuario para que el programa sea de fácil manejo para el empleado y el director.
- Además, viene incluida la parte de **pruebas y resultados** en el que se describirá detalladamente las pruebas aplicadas a lo largo de la construcción del programa, así como los resultados obtenidos y su análisis.
- Por último, expondremos las **conclusiones** de este trabajo.

2. Análisis

2.1 Requisitos funcionales

El programa que se expondrá más adelante está codificado en una plataforma llamada *Eclipse IDE for Java Developers* que es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma. Es por esto que, para la resolución al problema de la Farmacia se ocupará la plataforma antes mencionada.

2.2 Restricciones

La Farmacia San Pablo ubicada en Av. Cuauhtémoc 863, Narvarte Poniente, Benito Juárez, 03020 Ciudad de México, CDMX se encuentra en un terreno relativamente pequeño. Por lo tanto, no contiene todas las medicinas existentes en el mundo, así que no toda medicina o producto que requiera un cliente estará disponible.

Por otro lado, para calcular el mes con mayores ventas, es necesario ingresar las ventas indicando el mes, además de que esta funcionalidad sólo será correcta al final del año (Diciembre) para que el programa tome en cuenta todos los meses (desde Enero hasta Diciembre), de no ser así arrojará que el mes con menores ventas será el mes que aún no ha transcurrido (si solo hemos ingresado ventas del mes de enero y febrero y ejecutamos la funcionalidad de "mes con menores ventas" nos arrojará "marzo" debido a que lo tiene registrado como 0 ventas)

3. Diseño

2.1 Descripción completa del diseño

Se diseñaron 8 clases:

- **Medicinas:** matriz de medicamentos y productos
- **Ventas:** arreglo de ventas en el que se va guardando cada venta realizada.
- **ManejadorArreglo:** clase con varias funcionalidades para un arreglo que se manda a llamar para dar funcionalidad a las ventas y al empleado.
- **ManejadorMatricesGenerico:** clase con varias funcionalidades genéricas para una matriz que se manda a llamar para dar funcionalidad a la matriz de medicinas.
- **Farmacia:** clase para realizar los movimientos y cálculos necesarios para poder responder al problema del director.
- **VistaFarmacia:** clase en la que se da diseño y formato a la ventana que deseamos que aparezca al correr el programa.
- **ControladorFarmacia:** clase que manda a llamar a Farmacia para dar funcionalidad a la VistaFarmacia.
- **EjecutableVistaFarmacia:** clase para trabajar con la farmacia a través de una ventana.

2.2 Estándares utilizados

A continuación, se presentarán los UML utilizados para cada una de las clases, así como los diagramas de flujo.

UML Medicinas

Medicinas
<ul style="list-style-type: none">- nombre: String- tipo: char- código: String- precio: int
<pre>+ Medicinas (String: nombre, char: tipo, String: codigo, int: precio) + getPrecio(): int + getNombre (): String + getTipo(): char + getCodigo(): String + setPrecio (precio: double): void +toString (): String + equals(): Boolean +compareTo: double</pre>

UML Ventas

Ventas

```
- monto: double  
- nombre: String
```

```
+venta (double: monto, String: nombre)  
+getMonto (): double  
+getNombre (): String  
+setMonto (monto:double): void  
+setNombre (nombre:String): void  
+toString (): String  
+equals (): Boolean  
+compareTo: double
```

ITAM Manejador Arreglo

ManejadorArreglo

```
+ sumArreglo (double, int): double  
+ promedioArreglo (double, int) : double  
+ indiceMayor (double, int) : int  
+ indiceMenor (double, int) : int  
+ mayorArreglo (double, int) : int  
+ menorArreglo (double, int): int  
+ cuantosMayorA (double, int, double) : int  
+ cualesMayorA (double, int, double) : int  
+ cuantosMenosA (double, int, double): int  
+ reccorreDerecha (double, int) : int  
+ recorreIzquierda (double, int) : int  
+ recorreIzquierdaApartirPos (double, int, int) : int  
+ invertir (int) : int  
+ esCapicua (double, int) : boolean
```



```

+ eliminaRepetidos (double, int) : int
+ ordena (double, int) : double
+ buscaSeq (double, int, double) : int

```

ZML ManejadorMatricesGenerico

ManejadorMatricesGenerico

```

+ buscaPorRenglon (T, int, int, T) : int
+ creaString (T, int, int) : ArrayList
+ buscaEnMatriz (T, int, int, T) : int
+ indicaMayorColumna (T, int) : int
+ indicaMenorColumna (T, int): int
+ indicaMayorFila (T, int): int
+ indicaMenorFila (T, int): int
+ sonIguales (T, int, int, T) : boolean
+ matrizDiagonalInfMay (T, int, int) : boolean

```

ZML Farmacia

Farmacia

```

- nombre: String
- dirección: String
- MAX: int = 50
- producto [] []: Medicinas
- MAXC: int = 3
- MAXF: int =6
- ventas []: double
- TOTALVENTAS: int = 12

```

```
- meses []= String
```

```
+ Farmacia ()  
+ Farmacia (String: nombre, String: direccion, int  
ocupados)  
+ getNombre (): String  
+ getDireccion (): String  
+ getOcupados (): int  
+ setNombre (String): void  
+ setOcupados (int): void  
+ equals (Object): Boolean  
+ compareTo (Farmacia): int  
+ buscaMedicamento (int): int  
+ altaVenta (int, double): Boolean  
+ calculaPromedio (): double  
+ indicaMesMasVentas (): int  
+ indicaMesMenosVentas (): int  
+ indicaCuantosarribaProm (): int  
+ indicaVentasTotales (): double
```

Vista Farmacia

```
VistaFarmacia
```

```
protected info: JTextArea  
protected etVacia, etProducto, etMes, etCosto, etImagen:  
JTextField  
protected btBuscar, btAlta, btVentasTotales, btMesMás,  
btMesMenos, btVentasPromedio, btArribaProm: JButton  
protected miPanel: JPanel
```

```
protected miBorde: Border
```

```
+ VistaFarmacia ()
```

ZML Controlador Vista Farmacia

```
ControladorVistaFarmacia
```

- miFarmacia: Farmacia
- matriz [][]: JTextField
- MAXF: int = 5
- MAXC: int = 2

```
+cargaMatrizMedicamentos(String): String  
+EscuchaBusqueda(): String  
+EscuchaAlta(): String  
+EscuchaVentasTotales(): String  
+EscuchaPromedio(): String  
+EscuchaMayoresVentas(): String  
+EscuchaMenosVentas(): String  
+EscuchaArribaPromedio(): String  
+ControladorFarmacia()
```

Diagrama de Flujo de VentasTotales

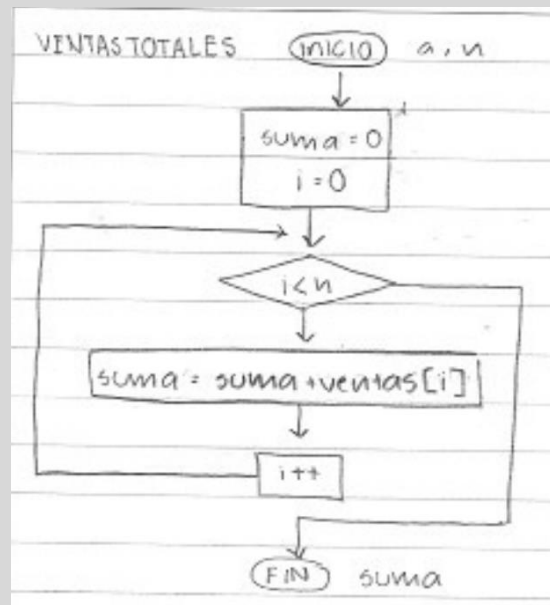


Diagrama de Flujo de Promedio

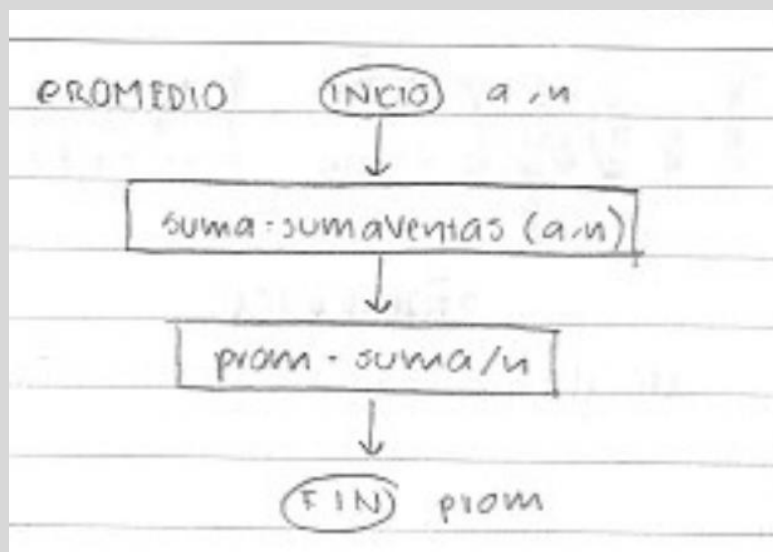


Diagrama de Flujo de AltaVenta

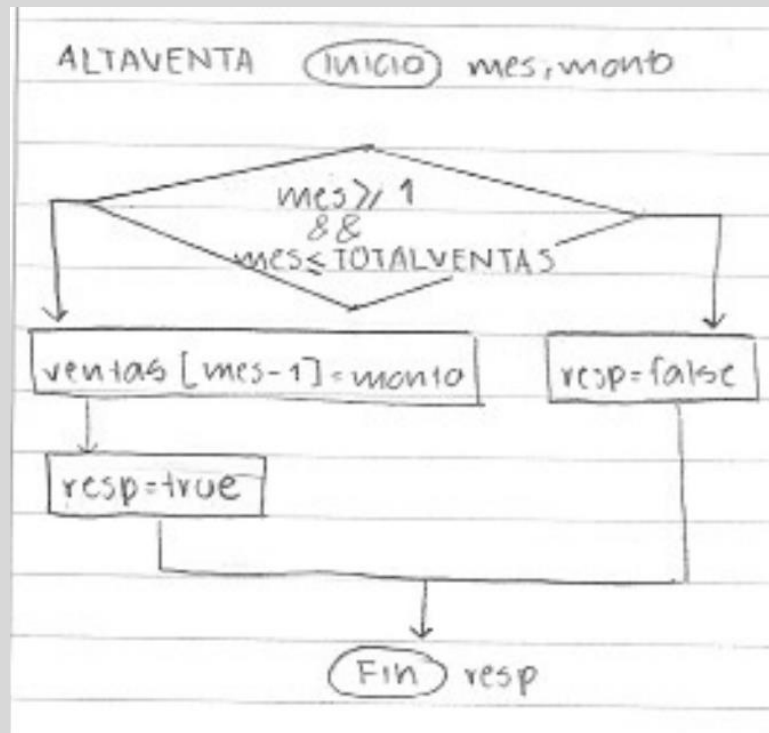


Diagrama de Flujo de IndicaMesMayoresVentas

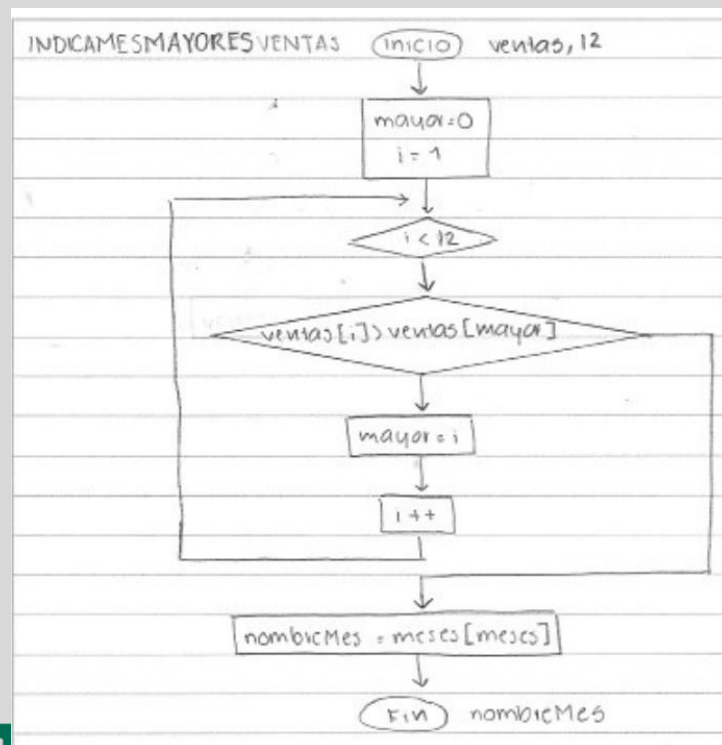
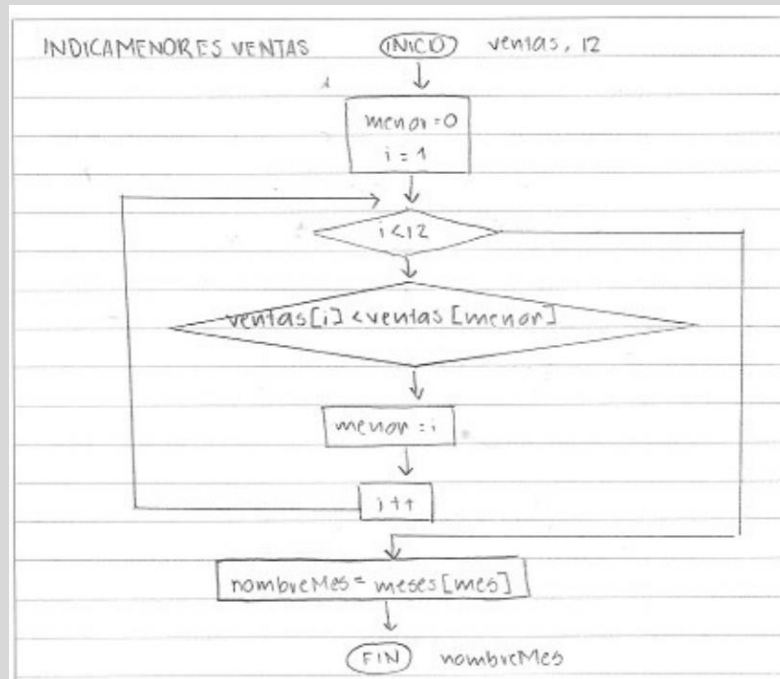


Diagrama de Flujo IndicaMesMayoresVentas



4. Implementación

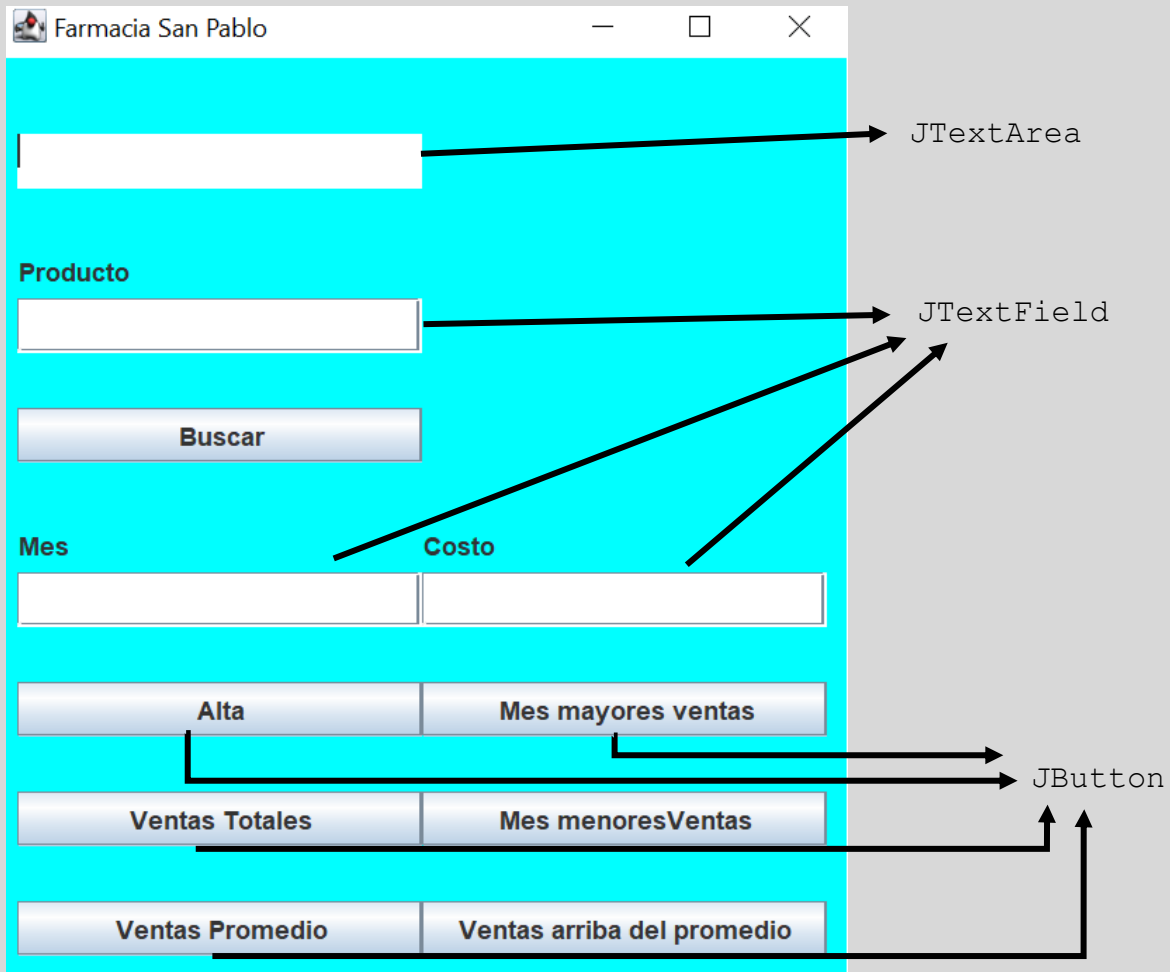
4.1 Descripción completa de la implementación

El código de Eclipse tiene un mismo orden para facilitar el entendimiento de este:

- Atributos
- Constructor
- Get's y set's
- Equals
- compareTo
- toString
- Funcionalidades (altaVenta, VentasTotoales, Promedio, etcétera)

4.2 GUI

Diagramas de interfaces gráficas



Escogí este diseño por diferentes razones:

1. El color al que la gente suele enlazar una farmacia o medicinas es el azul. Es esta la razón por la que decidí que el fondo de la ventana fuera azul. Además, no es un azul marino que transmite frío, es una azul claro y cálida que transmite tranquilidad.
2. El diseño de la ventana está inspirado en una caja registradora, ya que registrará todas las ventas día con día, mes con mes. Es por esto por lo que el JTextArea está

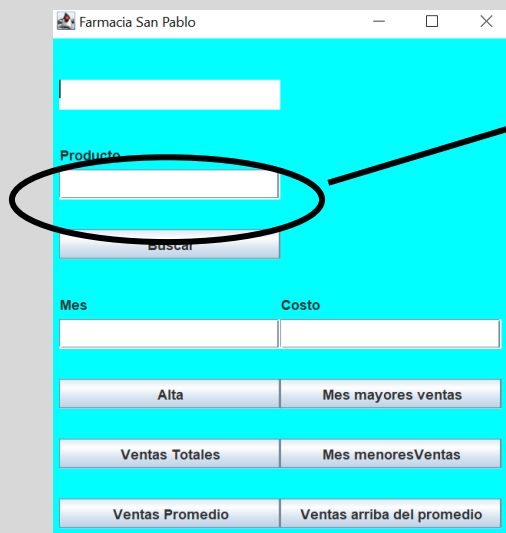
en la parte superior en la que aparecerán todos los mensajes que solicite el empleado o director.

3. Cuando un cliente llega lo primero que dice es el medicamento que va a desear, es por ello que lo segundo (después del JTextArea) que aparecerá es el JTextField en el cual, el empleado pondrá el nombre del producto que le solicita el cliente. Inmediato de esto, se encuentra el botón "Buscar" para que el empleado pueda teclearlo sin necesidad de ir a la bodega a verificar la existencia del producto.
4. Si el producto no está disponible, ahí acaba la ejecución del programa. Si, por el contrario, sí está en existencia el producto, el empleado tendrá que introducir el mes en el que se encuentra de manera numérica e introducirá el costo.
5. Seguidamente se encuentra el botón de "Alta" para guardar la compra del producto
6. Al final, están todos los demás botones que manipulará el director para llevar un mejor control de sus ganancias.

4.3 Manual de usuario

Para poder ejecutar las ventas de manera adecuada se necesita seguir los siguientes pasos:

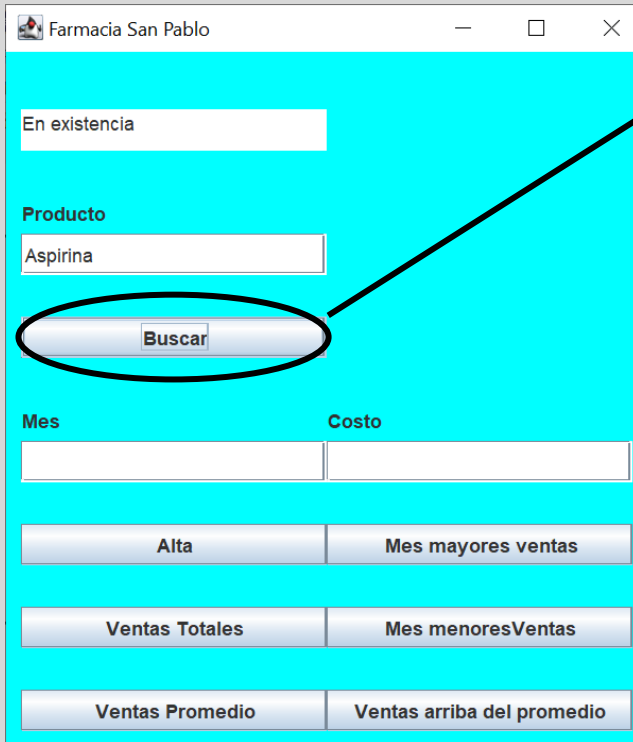
Paso 1:



Teclear el nombre del producto que solicita el cliente

Por ejemplo "Aspirina"

Paso 2:



Farmacia San Pablo

En existencia

Producto

Aspirina

Buscar

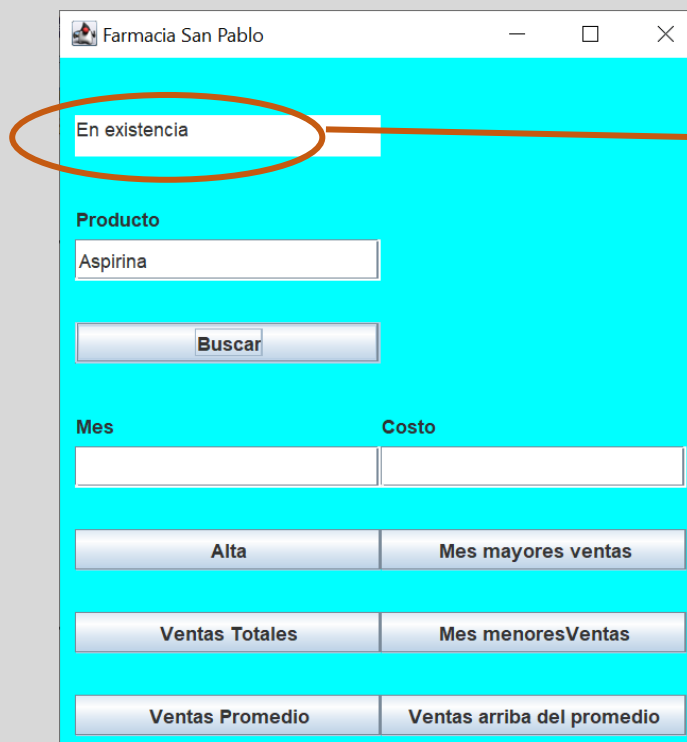
Mes Costo

Alta Mes mayores ventas

Ventas Totales Mes menoresVentas

Ventas Promedio Ventas arriba del promedio

Oprimir el botón "Buscar" para que el programa busque en la base de datos (matriz de Medicamentos) si está en existencia el producto tecleado anteriormente.



Farmacia San Pablo

En existencia

Producto

Aspirina

Buscar

Mes Costo

Alta Mes mayores ventas

Ventas Totales Mes menoresVentas

Ventas Promedio Ventas arriba del promedio

Si el programa encuentra el producto en su base de datos, aparecerá en la parte de arriba "En existencia"

Farmacia San Pablo

No disponible

Producto

Naproxen

Buscar

Mes Costo

Alta Mes mayores ventas

Ventas Totales Mes menoresVentas

Ventas Promedio Ventas arriba del promedio

Si, por el contrario, el programa no encuentra el producto anteriormente tecleado, en la parte superior aparecerá el mensaje

"No disponible"

En este caso, se le brinda una disculpa al cliente, se le pide a otro cliente que pase y se repite el mismo procedimiento del paso 1.

Farmacia San Pablo

En existencia

Producto

Aspirina

Buscar

Mes Costo

12 70

Alta Mes mayores ventas

Ventas Totales Mes menoresVentas

Ventas Promedio Ventas arriba del promedio

Paso 3:

En caso de que se encuentre en existencia el producto deseado, se tendrá que poner el mes en el que se realiza dicha venta.

En este caso 12 debido a que es diciembre.

Además, se deberá ingresar el costo del producto.

En este caso \$70

Paso 4:

Una vez ingresado el producto, el mes de la venta y el costo, se debe oprimir el botón de "Alta" para que la venta se guarde.

En la parte superior de la ventana aparecerá el mensaje:

"Gracias por su compra"

Estos 4 pasos se deberán hacer cada vez que se realice una venta.

Para poder explicar el Paso 5 cabe aclarar que le diremos al programa lo siguiente:

Enero ventas de \$100

Febrero ventas de \$200

Marzo ventas de \$300

Abril ventas de \$400

Mayo ventas de \$500

Junio ventas de \$600

Julio ventas de \$700

Agosto ventas de \$800

Septiembre ventas de \$900

Octubre ventas de \$1,000

Noviembre ventas de \$1,100

Diciembre ventas de \$1,200

Farmacia San Pablo

Mes Mayores Ventas: Diciembre

Producto

Buscar

Mes	Costo
12	1200

Alta

Mes mayores ventas

Ventas Totales

Mes menoresVentas

Ventas Promedio

Ventas arriba del promedio

Paso 5:

Oprimir el botón "Mes mayores ventas" para obtener (al final del año) el mes que obtuvo mayores ventas.

En la parte superior de la pantalla, aparecerá el mensaje:

"Mes Mayores Ventas: " + el mes que haya tenido mayores ventas. En este caso es el mes de Diciembre con ventas de \$1,200

Oprimir el botón "Mes menores ventas" para obtener el mes que obtuvo las menores ventas del año.

En la parte superior de la pantalla aparecerá un mensaje que dice:

"Mes Menores Ventas" + el mes con menores ventas del año. En este caso es Enero con ventas de \$100

Oprimir el botón de "Ventas Totales" para obtener las ventas totales hasta el momento (puede ser en cualquier momento del año).

En la parte superior de la pantalla, aparecerá el mensaje:

"Ventas Totales:" + las ventas totales que ha habido hasta el momento.

En este caso fueron las ventas de todo el año:

$$100+200+300+400+500+600+700+800+900+1000+1100+1200 = 7,800$$

Promedio de ventas: 650.0

Producto

Buscar

Mes Costo

Alta Mes mayores ventas

Ventas Totales Mes menoresVentas

Ventas Promedio Ventas arriba del promedio

Oprimir el botón de "Ventas Promedio" para obtener el promedio de ventas de todo el año. Cabe aclarar que el programa dividirá el total de ventas entre 12)

En la parte superior de la pantalla aparecerá el mensaje:

"Promedio de ventas:" + el promedio de ventas.

En este caso:

$$7800/12 = 650$$

Mes Arriba del Promedio: Julio

Producto

Buscar

Mes Costo

Alta Mes mayores ventas

Ventas Totales Mes menoresVentas

Ventas Promedio Ventas arriba del promedio

Oprimir el botón de "Ventas arriba del promedio" para obtener el mes en el que hubo ventas arriba del promedio.

En la parte superior de la pantalla aparecerá un mensaje que diga:

"Mes Arriba del Promedio:" + el mes que haya tenido ventas arriba del promedio

En este caso, siendo el promedio \$650, el mes arriba del promedio es Julio con ventas de \$700

5. Pruebas y Resultados

5.1 Descripción de las pruebas aplicadas (casos prueba)

Prueba 1:

Intenté buscar un medicamento llamado "Naproxen", el cual, no lo incluí en la matriz de Medicinas que le pedí al controlador que leyera en un archivo de texto. Sin embargo, en vez de aparecer "No disponible", aparecía "En existencia", situación que es falsa. Al buscar el problema, encontré que, en el ManejadorMatricesGenerico, la función que estaba utilizando llamada "BuscaPorRenglon" no estaba funcionando de manera correcta, así que opté por hacer una nueva función llamada "buscarEnMatriz":

```
public static <T extends Comparable<T>> int buscarEnMatriz(T a[][], int f, int c, T valor) {
    int resultado = -1;
    for(int ren=0; ren<=f;ren++) {
        for(int col=0; col <=c; col++) {
            if(a[ren][col].equals(valor)) {
                resultado= 1;
            }
        }
    }
    return resultado;
}
```

Gracias a esta nueva funcionalidad, logré que el botón "Buscar" tuviera la funcionalidad que yo esperaba.

Prueba 2:

Al ingresar el mes y el costo, el alta siempre funcionó de manera correcta

Prueba 3:

Al ingresar las ventas de los 12 meses y oprimir el botón de "Mes mayores ventas", salía la leyenda "Mes Mayores Ventas: 12" refiriéndose con 12 al mes de Diciembre, sin embargo, yo deseaba que apareciera el nombre del mes. Así que en farmacia puse una función de tipo String para que

regresara el mes, tanto de Mes con Menores Ventas como de Mes con Mayores Ventas.

```
//NOMBRE DEL MES CON MÁS VENTAS

public String indicaMesMayoresVentas() {
    int mes;
    String nombreMes;
    mes=ManejadorArreglo.indiceMayor(ventas, TOTALVENTAS);
    nombreMes= meses [mes];
    return nombreMes;
}

//NOMBRE DEL MES CON MENOS VENTAS
public String indicaMesMenoresVentas() {
    int mes;
    String nombreMes;

    mes=ManejadorArreglo.indiceMenor(ventas, TOTALVENTAS);
    nombreMes= meses [mes];
    return nombreMes;
}
```

Prueba 4:

Al querer ejecutar la funcionalidad de los botones, en la ventana no aparecía nada, como si no tuviera funcionalidad. Después de hacer una búsqueda en las funciones realizadas y en el ControladorVistaFarmacia, recordé que al final del mismo se debe de publicar el ControladorFarmacia y mandar a llamar al "super();"

```
public ControladorFarmacia () {
    super ();
    miFarmacia = new Farmacia();
    btBuscar.addActionListener(new EscuchaBusqueda());
    btAlta.addActionListener(new EscuchaAlta());
    btVentasPromedio.addActionListener(new EscuchaPromedio());
    btVentasTotales.addActionListener(new EscuchaVentasTotales());
    btMesMás.addActionListener(new EscuchaMayoresVentas());
    btMesMenos.addActionListener(new EscuchaMenosVentas());
    btArribaProm.addActionListener(new EscuchaArribaPromedio());
}
```


Prueba 5:

Muy parecido fue mi error al de la Prueba 3. Al oprimir el botón de "Ventas Arriba del Promedio" me aparecía la leyenda "Mes Arriba del Promedio: 7" refiriéndose con 7 al mes de Julio. Sin embargo, yo deseaba que apareciera el nombre del mes. Así que realicé una funcionalidad en la clase Farmacia para que regresara un String y no un entero.

```
//NOMBRE DEL MES CON VENTAS ARRIBA DEL PROMEDIO
public String indicaMesArribaProm() {
    int cuantos;
    double promedio;
    String nombreMes;

    promedio = ManejadorArreglo.promedioArreglo(ventas, TOTALVENTAS);
    cuantos = ManejadorArreglo.cuantosMayorA(ventas, TOTALVENTAS, promedio);
    nombreMes = meses [cuantos];
    return nombreMes;
}
```

Prueba 6:

Al ejecutar la VistaFarmacia, deseaba que al inicio se viera el logo de la "Farmacia San Pablo", sin embargo, la imagen era demasiado grande y no se podía ver. Por cuestiones de tiempo, no pude seguir investigando sobre cómo poder ajustar la imagen, es por esto que opté por dejar comentado el intento de la imagen.

```
/*etImagen = new JLabel();
etImagen.setIcon(new ImageIcon("SanPablo.png")); //Se agrega una imagen
etImagen.setHorizontalAlignment(SwingConstants.CENTER);
etImagen.setVerticalAlignment(SwingConstants.CENTER);
miPanel.add(etImagen);
etVacia = new JLabel (" ");
miPanel.add(etVacia);
etVacia = new JLabel (" ");
miPanel.add(etVacia);*/
```

5.2 Resultados obtenidos

Después de un laborioso camino para poder dar la funcionalidad correcta al ControladorVistaFarmacia, los resultados fueron los esperados.

5.3 Análisis de los resultados

Los resultados obtenidos resuelven el problema planteado en la Introducción de este documento mediante un código eficiente y de calidad.

6. Conclusiones

Me sirvió bastante realizar este proyecto debido a que puse en práctica todos los temas vistos a lo largo del semestre. Además, yo tenía la idea errónea de que los genéricos eran como "el salero" en la mesa, pensaba que no tenía mucha funcionalidad; gracias a este proyecto entendí que los genéricos no son el salero, pueden llegar a ser los cubiertos, un elemento muy importante para la correcta funcionalidad del programa. También reafirmé mis conocimientos para usar arreglos y archivos de texto en el código. Por otro lado, entendí de manera más clara el uso de matrices, situación que antes se me complicaba mucho.

Fue un reto querer juntar en un mismo problema arreglos, matrices y genéricos, pero quedo satisfecha con el resultado, reto aceptado y completado.

Referencias

<https://stackoverflow.com/questions/4419667/detect-enter-press-in-jtextfield>

Apéndice

Clase Medicinas

```
/*Mariana Arroyo Muñoz
 * 197344
 * 29/11/2020
 *
 * Clase de una matriz de medicinas
 */
public class Medicinas {
    private String nombre;
    private char tipo; //D (dormir), B (bebés), H (higiene), M
(malestares), C (condones), P (primeros auxilios)
    private String codigo; //código de indentificación
    private int precio;

    //Constructor generado
    public Medicinas(String nombre, char tipo, String codigo, int
precio) {
        this.nombre = nombre;
        this.tipo = tipo;
        this.codigo = codigo;
        this.precio = precio;
    }

    //Get's y Set's
    public int getPrecio() {
        return precio;
    }

    public void setPrecio(int precio) {
        this.precio = precio;
    }

    public String getNombre() {
        return nombre;
    }

    public char getTipo() {
        return tipo;
    }

    public String getCodigo() {
        return codigo;
    }
}
```

```

//Equals con Código de Identificación
@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((codigo == null) ? 0 :
codigo.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Medicinas other = (Medicinas) obj;
    if (codigo == null) {
        if (other.codigo != null)
            return false;
    } else if (!codigo.equals(other.codigo))
        return false;
    return true;
}

//compareTo
public int compareTo (Medicinas otro) {
    return codigo.compareTo(otro.getCodigo());
}

//toString
public String toString() {
    StringBuilder sb;
    sb= new StringBuilder();
    sb.append("Medicina \n");
    sb.append("nombre:"+nombre);
    sb.append("Tipo: "+tipo);
    sb.append("Código de Identificación: "+codigo);
    sb.append("Precio : "+precio);
    return sb.toString();
}

}
}

```

Clase Ventas

```
/*Mariana Arroyo Muñoz
 * 197344
 * 04/12/2020
 *
 */
public class Ventas {

    //Atributos
    private double monto;
    private String nombre;

    //Constructor
    public Ventas(double monto, String nombre) {
        this.monto = monto;
        this.nombre = nombre;
    }

    //get's
    public double getMonto() {
        return monto;
    }

    public String getNombre() {
        return nombre;
    }

    //set's
    public void setMonto(double otroMonto) {
        monto=otroMonto;
    }

    public void setNombre(String otroNombre) {
        nombre=otroNombre;
    }

    //equals con Nombre
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((nombre == null) ? 0 :
nombre.hashCode());
    }
}
```

```

        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Ventas other = (Ventas) obj;
        if (nombre == null) {
            if (other.nombre != null)
                return false;
        } else if (!nombre.equals(other.nombre))
            return false;
        return true;
    }

    //compareTo
    public int compareTo(Ventas otraVenta) {
        int x;
        if(otraVenta.getMonto()<monto)
            x=1;
        else if(otraVenta.getMonto()==monto)
            x=0;
        else
            x=-1;
        return x;
    }

    //toString
    public String toString() {
        StringBuilder sb;
        sb= new StringBuilder();
        sb.append("Venta\n");
        sb.append("\nmonto:$"+monto);
        sb.append("nombre:"+nombre);
        return sb.toString();
    }

} //Fin de la clase Venta

```

Clase ManejadorArreglo

```
import java.util.*;

/*Mariana Arroyo Muñoz }
 * 197344
 * 02/10/2020
 * Clase para el manejo general de arreglos
 */
public class ManejadorArreglo {

    //SUMA

    public static double sumArreglo(double a[], int n) {
        double suma;
        int i;

        suma=0;
        for(i=0;i<n;i++)
            suma=suma+a[i];
        return suma;
    }

    //PROMEDIO

    public static double promedioArreglo(double a[], int n) {
        double suma, prom;

        suma=sumArreglo(a,n);
        prom=suma/n;
        return prom;
    }

    //INDICE MAYOR

    public static int indiceMayor(double a[], int n) {
        int mayor, i;

        mayor=0;
        for(i=1;i<n;i++)
            if(a[i]>a[mayor])
                mayor=i;
        return mayor;
    }

    //INDICE MENOR
```

```

public static int indiceMenor(double a[], int n) {
    int menor, i;

    menor=0;
    for(i=1;i<n;i++)
        if(a[i]<a[menor])
            menor=i;
    return menor;
}

//MAYOR A CIERTO PARÁMETRO

public static int mayorArreglo(double a[], int n) {
    int mayor, i;

    mayor=0;
    for(i=1;i<n;i++)
        if(a[i]>a[mayor])
            mayor = i;
    return mayor;
}

//MENOR A CIERTO PARÁMETRO

public static int menorArreglo(double a[], int n) {
    int menor, i;

    menor = 0;
    for(i=1; i<n; i++)
        if(a[i]<a[menor])
            menor = i;
    return menor;
}

//CUANTOS ELEMENTOS MAYORES AL PARÁMETRO

public static int cuantosMayorA(double a[], int n, double p)
{
    int i, cuantos;

    cuantos = 0;
    for (i = 0; i<n; i++)
        if(a[i]>p)
            cuantos ++;
    return cuantos;
}

```



```

    }

    public static ArrayList<Double> cualesMayorA (double a[], int
n, double p){
        int i;
        ArrayList<Double> cuales;

        cuales=new ArrayList<Double>();
        for(i=0; i<n; i++)
            if(a[i]>p)
                return cuales;
        return cuales;
    }

    //CUANTOS ELEMENTO MENORES AL PARÁMETRO

    public static int cuantosMenorA(double a [], int n,double p)
{
    int i, cuantos;

    cuantos=0;
    for(i=0;i<n;i++)
        if(a[i]<p)
            cuantos++;
    return cuantos;
}

    //CORRER LOS ELEMENTOS UN LUGAR A LA DERECHA

    public static void recorreDerecha (double a [], int n) {
        int i;
        for (i=n-1; i>0;i--)
            a[i]=a[i-1];
        a[0]=0;
    }

    //CORRER LOS ELEMENTOS UN LUGAR A LA IZQUIERDA

    public static void recorreIzquierda(double a[], int n) {
        int i;
        for (i=0; i<n-1;i++)
            a[i]=a[i+1];
        a[n-1]=0;
    }
}

```

```

//RECORRER LOS ELEMENTOS A LA IZQUIERDA APARTIR DE UNA
POSICIÓN
public static void recorreIzquierdaApartirPos(double a[], int
n, int pos) {
    int i;
    for(i=pos; i<n-1; i++)
        a[i] = a[i+1];
    a[n-1]=0;
}
//INVERTIR LOS ELEMENTOS

public static void invertir (int a[]) {
    int j=0, k=0, n=0;
    for(int i=0; i<n/2;i++) {
        j=a[i];
        k=a[n-1-i];

        a[i]=k;
        a[n-1-i]=j;
    }
}

//ELEMENTOS CAPICÚOS
public static boolean esCapicua(double a[], int n) {
    int i, mitad;
    boolean resp;

    mitad=n/2;
    i=0;
    while (i<mitad && a[i]==a[n-1-i])
        i++;
    if(i<mitad)
        resp=false;
    else
        resp=true;
    return resp;
}

//ELIMINA REPETIDOS
public static void eliminaRepetidos (double a[], int n) {
    int i, j, k;
    for(i=0;i<n;i++) {//recorrer el arreglo
        j=i+1;
        while (j<n && a[i]!=a[j])//lo busca
            j++;
        if(j<n) {//si esta

```

```

        for(k=j;k<n-1;k++)//recorre izquierda
            a[k]=a[k+1];
        a[n-1]=0;
        n--;//para no revisar los ceros del final
        i--;//por si hay más de 1 repetido contiguo
    }
}

//ORDENA ARREGLO
public static void ordena(double a[], int n) {
    int i, j, menor;
    double aux;

    for (i=0; i<n-1; i++) {
        menor = i;
        for (j=i+1; j<n; j++)
            if (a[j]<a[menor])
                menor=j;
        aux=a[i];
        a[i]=a[menor];
        a[menor]=aux;
    }
}

//BUSCAR UNA SECUENCIA DADO UN VALOR (cuando no tiene orden)
public static int buscaSeq(double a[], int n, double
valor) {
    int i, pos;

    i=0;
    while(i<n && a[i]!=valor)
        i++;
    if(i<n)
        pos=i;
    else
        pos=-1;
    return pos;
}

//Fin del método

//Fin de la clase

```

Clase ManjadorMatricesGenerico

```
import java.util.ArrayList;

/*Mariana Arroyo Muñoz
 * 197344
 * 10/11/2020
 */
public class ManejadorMatricesGenerico {

    public static <T extends Comparable<T>> int buscaPorRenglon(T
a[][], int f,int c, T valor) {
        //busca valor en a en la fila
        int n, m, j;
        boolean encuentre;

        n = a.length;
        m = a[0].length;
        j = 0;
        encuentre = false;
        while (j<m && encuentre == false) {
            if (a[f][j]!=null && a[f][j].equals(valor))
                encuentre = true;
            else
                j++;
        }
        if (!encontre)
            j = -1;
        return j;
    }

    public static <T extends Comparable <T>> ArrayList<T>
creaString(T a[][],int f, int c) {
        //regresa en forma de string los elementos de una matriz
        int i, j;
        ArrayList<T> lista;
        lista= new ArrayList<T>();
        for(i=0;i<f;i++)
            for(j=0;j<c;j++) {
                lista.add((T) a[i][j]);
            }
        return lista;
    }

    public static <T extends Comparable<T>> int busca(T a[][],
int f, int c, T valor) {
```

```

        //busca valor en a
        int i,j,pos;
        pos=0;
        for(i=0;i<f;i++)
            for(j=0;j<c;j++) {
                while(j<c && !a[i][j].equals(valor))
                    i++;
                if(j<c)
                    pos=i;
                else
                    pos=-1;
            }
        return pos;
    }

    public static <T extends Comparable<T>> int buscarEnMatriz(T
a[][], int f, int c, T valor) {
        int resultado = -1;
        for(int ren=0; ren<=f;ren++) {
            for(int col=0;col <=c; col++) {
                if(a[ren][col].equals(valor)) {
                    resultado= 1;
                }
            }
        }
        return resultado;
    }

    public static <T extends Comparable <T>> int
indicaMayorColumna(T a[][], int c) {
        int i, mayor;

        mayor = 0;
        for ( i = 1; i<c; i++)
            if(a[i][c].compareTo(a[mayor][c])>0)
                mayor = i;
        return mayor;
    }

    public static <T extends Comparable <T>> int
indicaMenorColumna(T a[][], int c) {
        int i, menor;

        menor = 0;
        for ( i = 1; i<c; i++)
            if(a[i][c].compareTo(a[menor][c])<0)

```

```

        menor = i;
    }
    return menor;
}

public static <T extends Comparable<T>> int
indicaMayorFila(T a[][], int f) {
    int j, mayor;

    mayor = 0;
    for ( j = 1; j<f; j++)
        if(a[f][j].compareTo(a[f][mayor])>0)
            mayor = j;
    return mayor;
}

public static <T extends Comparable<T>> int
indicaMenorFila(T a[][], int f) {
    int j, menor;

    menor = 0;
    for ( j = 1; j<f; j++)
        if(a[f][j].compareTo(a[f][menor])>0)
            menor = j;
    return menor;
}

public static <T extends Comparable<T>> boolean sonIguales(T
a[][], int f, int c, T b[][]) {
    boolean resp=false;
    int i,j;
    for(i=0;i<f;i++)
        for(j=0;j<c;j++) {
            if(a[i][j].equals(b[i][j]))
                resp=true;
            else
                resp=false;
        }
    return resp;
}

public static <T extends Comparable<T>> boolean
matrizDiagonalInfMay(T a[][], int f, int c) {
    int i,j;
    boolean resp=false;
    for(i=0;i<f;i++)

```

```
        for(j=0;j<c;j++) {  
            while(a[i+1][0].compareTo(a[0][j+1])>0)  
                resp=true;  
        }  
        return resp;  
    }  
}  
} //Fin de la clase
```

Clase Farmacia

```
/*Mariana Arroyo MUñoz
 * 197344
 * 30/11/2020
 * Clase para dar funcionalidad
 */
public class Farmacia {

    private String nombre;
    private String direccion;
    private Medicinas [][] producto; //MATRIZ DE MEDICINAS
    private final int MAXC = 3; //3 columnas de medicamentos
    private final int MAXF = 6; //6 renglones de tipo de
medicamentos
    private double ventas[]; //ARREGLO DE VENTAS;
    private final int TOTALVENTAS = 12;
    private String [] meses= {"Enero", "Febrero", "Marzo",
"Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre",
"Octubre", "Noviembre", "Diciembre"};

    //Constructor vacío
    public Farmacia() {
        ventas = new double[TOTALVENTAS];
    }

    //Constructor Generado
    public Farmacia(String nombre, String direccion, int
ocupados) {
        this();
        this.nombre = nombre;
        this.direccion = direccion;
    }

    //Get's y Set's
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getDireccion() {
```



```

        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    //equals con Nombre
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((nombre == null) ? 0 :
nombre.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Farmacia other = (Farmacia) obj;
        if (nombre == null) {
            if (other.nombre != null)
                return false;
        } else if (!nombre.equals(other.nombre))
            return false;
        return true;
    }

    //compareTo
    public int compareTo(Farmacia otra) {
        return nombre.compareTo(otra.getNombre());
    }

    //-----> FUNCIONES <-----

    //BUSCA MEDICAMENTO (matriz usando genéricos)
    public static <T extends Comparable<T>> int buscaMedicamento
(T producto[][], int f,int c, T medicina) {
        int medicinas;

```

```

        medicinas =
ManejadorMatricesGenerico.buscarEnMatriz(producto, f, c,
medicina);
        return medicinas;
    }

//ALTA VENTA
public boolean altaVenta(int mes, double monto) {
    boolean resp;
    if(mes>=1 && mes<=TOTALVENTAS) {
        ventas[mes-1]=monto;
        resp=true;
    }
    else
        resp=false;
    return resp;///"Gracias por su compra"
}

//PROMEDIO

public double calculaPromedio() {
    double prom;
    prom= ManejadorArreglo.promedioArreglo(ventas,
TOTALVENTAS);
    return prom;
}

//MES CON MÁS VENTAS

public int indicaMesMayoresVentas1() {
    int mes;
    mes=ManejadorArreglo.indiceMayor(ventas, TOTALVENTAS);
    mes++;
    return mes;
}

//NOMBRE DEL MES CON MÁS VENTAS

public String indicaMesMayoresVentas() {
    int mes;
    String nombreMes;
    mes=ManejadorArreglo.indiceMayor(ventas, TOTALVENTAS);
    nombreMes= meses [mes];
    return nombreMes;
}

```

```

//MES CON MENOS VENTAS
public int indicaMesMenoresVentas1() {
    int mes;

    mes=ManejadorArreglo.indiceMenor(ventas, TOTALVENTAS);
    mes++;
    return mes;
}

//NOMBRE DEL MES CON MENOS VENTAS
public String indicaMesMenoresVentas() {
    int mes;
    String nombreMes;

    mes=ManejadorArreglo.indiceMenor(ventas, TOTALVENTAS);
    nombreMes= meses [mes];
    return nombreMes;
}

//MES CON VENTAS ARRIBA DEL PROMEDIO
public int indicaCuantosArribaProm() {
    int cuantos;
    double promedio;

    promedio = ManejadorArreglo.promedioArreglo(ventas,
TOTALVENTAS);
    cuantos = ManejadorArreglo.cuantosMayorA(ventas,
TOTALVENTAS, promedio);
    return cuantos;
}

//NOMBRE DEL MES CON VENTAS ARRIBA DEL PROMEDIO
public String indicaMesArribaProm() {
    int cuantos;
    double promedio;
    String nombreMes;

    promedio = ManejadorArreglo.promedioArreglo(ventas,
TOTALVENTAS);
    cuantos = ManejadorArreglo.cuantosMayorA(ventas,
TOTALVENTAS, promedio);
    nombreMes = meses [cuantos];
    return nombreMes;
}

```

```
//VENTAS TOTALES
public double indicaVentasTotales() {
    double suma;

    suma = ManejadorArreglo.sumArreglo(ventas, TOTALVENTAS);
    return suma;
}
}
```

Clase VistaFarmacia

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;

/*Mariana Arroyo Muñoz
 * 197344
 * 04/12/2020
 * Clase para diseñar la ventana de Farmacia (elementos visuales)
 */
public class VistaFarmacia extends JFrame {

    protected JTextArea info;
    protected JLabel etVacia, etProducto, etMes, etCosto,
etImagen;
    protected JTextField ctProducto, ctCosto, ctCantidad, ctMes;
    protected JButton btBuscar, btAlta, btVentasTotales,
btMesMás, btMesMenos, btVentasPromedio, btArribaProm;
    protected JPanel miPanel;
    protected Border miBorde;

    public VistaFarmacia () {
        super ("Farmacia San Pablo");
        miPanel = new JPanel (new GridLayout (16,2));
        miBorde = BorderFactory.createEmptyBorder(10, 10, 10,
10);
        miPanel.setBorder(miBorde);
        miPanel.setBackground(Color.cyan); //se le agrega color
al fondo

        /*etImagen = new JLabel();
        etImagen.setIcon(new ImageIcon("SanPablo.png")); //Se
agrega una imagen
        etImagen.setHorizontalAlignment(SwingConstants.CENTER);
        etImagen.setVerticalAlignment(SwingConstants.CENTER);
        miPanel.add(etImagen);
        etVacia = new JLabel (" ");
        miPanel.add(etVacia);
        etVacia = new JLabel (" ");
        miPanel.add(etVacia);*/

        //Primer Renglón
        etVacia = new JLabel (" ");
        miPanel.add(etVacia);
        etVacia = new JLabel (" ");
```

```

miPanel.add(etVacia);

//Segundo Renglón
info = new JTextArea();
miPanel.add(info);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Tercer renglón
etVacia = new JLabel (" ");
miPanel.add(etVacia);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Cuarto Renglón
etProducto = new JLabel ("Producto");
miPanel.add(etProducto);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Quinto Renglón
ctProducto = new JTextField (20);
miPanel.add(ctProducto);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Sexto Renglón
etVacia = new JLabel (" ");
miPanel.add(etVacia);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Séptimo Renglón
btBuscar = new JButton ("Buscar");
miPanel.add(btBuscar);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Octavo Renglón
etVacia = new JLabel (" ");
miPanel.add(etVacia);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Noveno Renglón
etMes = new JLabel ("Mes");

```

```

miPanel.add(etMes);
etCosto = new JLabel ("Costo");
miPanel.add(etCosto);

//Décimo renglón
ctMes = new JTextField (20);
miPanel.add(ctMes);
ctCosto = new JTextField (20);
miPanel.add(ctCosto);

//Onceavo renglón
etVacia = new JLabel (" ");
miPanel.add(etVacia);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Doceavo renglón
btAlta = new JButton ("Alta");
miPanel.add(btAlta);
btMesMás = new JButton ("Mes mayores ventas");
miPanel.add(btMesMás);

//Treceavo renglón
etVacia = new JLabel (" ");
miPanel.add(etVacia);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

//Catorceavo renglón
btVentasTotales = new JButton ("Ventas Totales");
miPanel.add(btVentasTotales);
btMesMenos = new JButton ("Mes menoresVentas");
miPanel.add(btMesMenos);

etVacia = new JLabel (" ");
miPanel.add(etVacia);
etVacia = new JLabel (" ");
miPanel.add(etVacia);

btVentasPromedio = new JButton ("Ventas Promedio");
miPanel.add(btVentasPromedio);
btArribaProm = new JButton ("Ventas arriba del
promedio");
miPanel.add(btArribaProm);

```

```
    super.setContentPane(miPanel);  
    super.setDefaultCloseOperation(EXIT_ON_CLOSE);  
  
    }//Fin de VistaFarmacia  
  
}//Fin de la clase VistaFarmacia JFrame
```


Clase ControladorFarmacia

```
import java.awt.event.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

import javax.swing.*;

/*Mariana Arroyo Muñoz
 * 197344
 * 04/12/2020
 * Clase para poner funcionalidad a la VistaFarmacia y llamar a la
 Farmacia
 */
public class ControladorFarmacia extends VistaFarmacia {

    private Farmacia miFarmacia;
    private JTextField matriz[][];
    private final int MAXF = 5;
    private final int MAXC = 2;

    public static String [][] cargarMatrizMedicamentos(String
nombreArchivo){

        String [][] matrizMedicamentos = new String [6][3];
        File miArchivo;
        Scanner lectura;
        int i, j, n=5 ,m=2;

        nombreArchivo = nombreArchivo+".txt";
        miArchivo = new File (nombreArchivo);
        try {
            lectura = new Scanner (miArchivo);
            for (i=0; i<=n; i++) {
                for(j=0;j<=m ;j++) {
                    matrizMedicamentos[i][j] =
lectura.next();
                }
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        return matrizMedicamentos;
    }
}
```

```

}

public class EscuchaBusqueda implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {

        String producto;
        int resp;

        producto = ctProducto.getText();
        resp =
miFarmacia.buscaMedicamento\(cargarMatrizMedicamentos\("medicamentos
"\), MAXF, MAXC, producto);
        if (resp == -1)
            info.setText("No disponible");
        else
            info.setText("En existencia");

    }

}

} //Fin de EscuchaBusqueda

public class EscuchaAlta implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        int mes;
        double monto;
        boolean resp;

        monto = Double.parseDouble(ctCosto.getText());
        mes = Integer.parseInt(ctMes.getText());
        resp = miFarmacia.altaVenta(mes, monto);
        if (resp == true)
            info.setText("Gracias por su compra");
        else
            info.setText("Alta NO Exitosa");

    }

}

} //Fin de EscuchaAlta

public class EscuchaVentasTotales implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        double total, resp;

```

```

        total = Double.parseDouble(ctCosto.getText());
        resp = miFarmacia.indicaVentasTotales();
        info.setText("Ventas Totales: "+resp);
    }
}

public class EscuchaPromedio implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        double resp;

        resp = miFarmacia.calculaPromedio();
        info.setText("Promedio de ventas: "+resp);
    }

}

} //Fin de EscuchaPromedio

public class EscuchaMayoresVentas implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        int mes;
        String resp;

        mes = Integer.parseInt(ctMes.getText());
        resp =
String.valueOf(miFarmacia.indicaMesMayoresVentas());
        info.setText("Mes Mayores Ventas: "+resp);
    }

}

public class EscuchaMenosVentas implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        int mes;
        String resp;

        mes = Integer.parseInt(ctMes.getText());
        resp =
String.valueOf(miFarmacia.indicaMesMenoresVentas());
        info.setText("Mes Menores Ventas: "+resp);
    }

}

```

```

        public class EscuchaArribaPromedio implements ActionListener
        {

            @Override
            public void actionPerformed(ActionEvent e) {
                int mes;
                String resp;

                mes = Integer.parseInt(ctMes.getText());
                resp =
String.valueOf(miFarmacia.indicaMesArribaProm());
                info.setText("Mes Arriba del Promedio: "+resp);
            }
        }

        public ControladorFarmacia () {
            super ();
            miFarmacia = new Farmacia();
            btBuscar.addActionListener(new EscuchaBusqueda());
            btAlta.addActionListener(new EscuchaAlta());
            btVentasPromedio.addActionListener(new
EscuchaPromedio());
            btVentasTotales.addActionListener(new
EscuchaVentasTotales());
            btMesMás.addActionListener(new EscuchaMayoresVentas());
            btMesMenos.addActionListener(new EscuchaMenosVentas());
            btArribaProm.addActionListener(new
EscuchaArribaPromedio());

        }

    }//Fin de la clase ControladorFarmacia

```

Clase Ejecutable Farmacia

```
/*Mariana Arroyo Muñoz
 * 197344
 * 04/12/2020
 * Clase para trabajar con el estacionamiento a través de una
ventana
 */
public class EjecutbaleVistaFarmacia {

    public static void main(String[] args) {
        ControladorFarmacia miVentana;

        miVentana = new ControladorFarmacia();
        miVentana.pack();
        miVentana.setVisible(true);
    }

}

} //Fin de la clase EjecutableVistaFarmacia
```