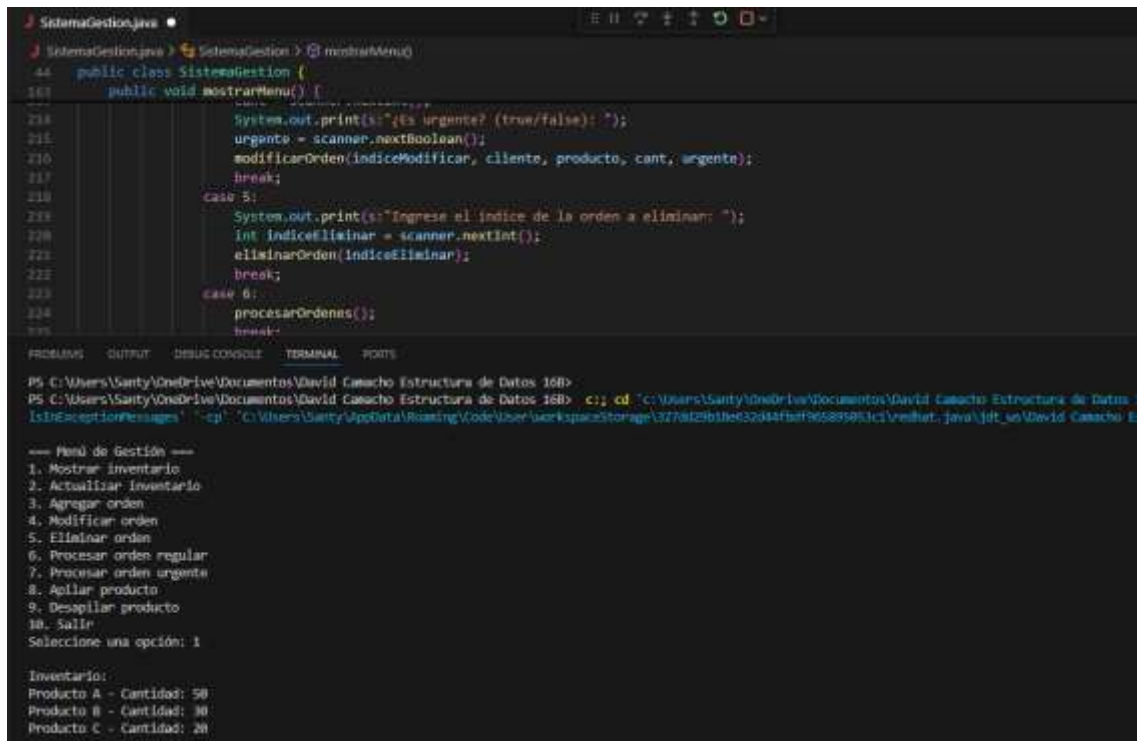


1. Gestión del inventario y ordenes

Al ejecutar el programa se despliega una lista donde se nos muestra diferentes opciones que podemos escoger, al momento de escoger la primera opción = 1, nos muestra el inventario actualizado, existen 3 productos (A, B, C) con diferentes cantidades cada uno.



```
SistemaGestion.java
public class SistemaGestion {
    public void mostrarMenu() {
        System.out.println("¿Es urgente? (true/false): ");
        urgente = scanner.nextBoolean();
        modificarOrden(indiceModificar, cliente, producto, cant, urgente);
        break;
    case 5:
        System.out.println("Ingrese el índice de la orden a eliminar: ");
        int indiceEliminar = scanner.nextInt();
        eliminarOrden(indiceEliminar);
        break;
    case 6:
        procesarOrdenes();
        break;
}

PS C:\Users\Santy\OneDrive\Documentos\David Camacho Estructura de Datos 168>
PS C:\Users\Santy\OneDrive\Documentos\David Camacho Estructura de Datos 168> c:\cd "C:\Users\Santy\OneDrive\Documentos\David Camacho Estructura de Datos 168"
ls!ExceptionMessages' -cp "C:\Users\Santy\AppData\Local\Temp\Code\workspaceStorage\37\ad29b18e632d44f8b1905895861c1\vmehat.java\jdt_ws\David Camacho E

--- Menú de Gestión ---
1. Mostrar inventario
2. Actualizar inventario
3. Agregar orden
4. Modificar orden
5. Eliminar orden
6. Procesar orden regular
7. Procesar orden urgente
8. Agilar producto
9. Desagilar producto
10. Salir
Seleccione una opción: 1

Inventario:
Producto A - Cantidad: 50
Producto B - Cantidad: 30
Producto C - Cantidad: 20
```

Al momento de actualizar un producto debemos hacer uso de la opción #2, vamos a actualizar el Producto A, dándole una mayor cantidad de 80, nos debe mostrar ese cambio.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

10. Salir
Seleccione una opción: 2
Ingrese el nombre del producto: Producto A
Ingrese la nueva cantidad: 80
Inventario actualizado: Producto A - Cantidad: 80

=== Menú de Gestión ===
1. Mostrar inventario
2. Actualizar inventario
3. Agregar orden
4. Modificar orden
5. Eliminar orden
6. Procesar orden regular
7. Procesar orden urgente
8. Apilar producto
9. Desapilar producto
10. Salir
Seleccione una opción: 1

Inventario:
Producto A - Cantidad: 80
Producto B - Cantidad: 30
Producto C - Cantidad: 20
```

Ahora vamos a agregar una nueva orden diciéndole que es de carácter Urgente, usando la opción #3, cumpliendo con todo lo que nos piden

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

10. Salir
Seleccione una opción: 3
Ingrese el nombre del cliente: Santiago
Ingrese el producto solicitado: Producto A
Ingrese la cantidad solicitada: 20
¿Es urgente? (true/false): true
Orden agregada: Cliente: Santiago, Producto: Producto A, Cantidad: 20, Urgente: true
```

Ahora vamos a modificar la orden que acabamos de agregar, con una nueva cantidad y cliente.

```
Seleccione una opción: 4
Ingrese el índice de la orden a modificar: 0
Ingrese el nuevo cliente: David
Ingrese el nuevo producto solicitado: Producto A
Ingrese la nueva cantidad solicitada: 7
¿Es urgente? (true/false): true
Orden modificada: Cliente: David, Producto: Producto A, Cantidad: 7, Urgente: true
```

Y por último vamos a eliminarla, nos da un mensaje de que se eliminó de manera correcta.

```
9. Desapilar producto
10. Salir
Seleccione una opción: 5
Ingrese el índice de la orden a eliminar: 0
Orden eliminada: Cliente: David, Producto: Producto A, Cantidad: 7, Urgente: true
```

2. Uso de cada estructura

- ARREGLO – ARRAY

Uso en el código:

Se utiliza un arreglo (Producto[] inventario) para almacenar los productos del inventario, donde cada producto tiene un nombre y una cantidad. Este arreglo es inicializado con los productos predefinidos.

Por qué se eligió el arreglo:

El arreglo es una estructura de datos adecuada cuando se sabe de antemano el número de elementos o cuando se requieren accesos rápidos a los elementos mediante su índice.

- ARRAYLIST

Uso en el código:

Se usa un ArrayList<Orden> (listaOrdenes) para almacenar las órdenes de los clientes. La lista permite agregar, modificar y eliminar órdenes de manera dinámica.

Por qué se eligió el ArrayList:

El ArrayList es una implementación de lista dinámica, lo que significa que puede crecer según sea necesario, lo que es ideal para manejar un número variable de órdenes.

- ARRAYDEQUE (Pila)

Uso en el código:

Se usa una pila (ArrayDeque<String> pilaAlmacen) para simular el almacenamiento físico de los productos en el almacén, siguiendo el principio LIFO (Last In, First Out).

Por qué se eligió la ArrayDeque:

La ArrayDeque es una estructura de datos que implementa la interfaz de Deque (Double Ended Queue), que permite operar como pila o cola. En este caso, se utiliza para manejar productos en un almacén físico en un orden LIFO.

- LINKEDLIST (Cola)

Uso en el código:

Se utiliza una LinkedList<Orden> (colaOrdenes) para almacenar las órdenes regulares de los clientes en el orden en que se reciben (FIFO: First In, First Out).

Por qué se eligió la LinkedList:

La LinkedList es una estructura de datos que implementa la interfaz de Queue, por lo que es perfecta para almacenar las órdenes en el orden en que deben ser procesadas.

- PRIORITYQUEUE (Cola Priorizada)**Uso en el código:**

Se utiliza una PriorityQueue<Orden> (colaUrgente) para gestionar las órdenes urgentes de los clientes. En este caso, la prioridad está definida por un booleano urgente, donde las órdenes urgentes tienen prioridad.

Por qué se eligió la PriorityQueue:

La PriorityQueue es una cola que organiza los elementos en función de su prioridad. En este caso, las órdenes urgentes deben ser procesadas antes que las órdenes normales.