Algoritmos y Estructuras de Datos I

CASO DE ESTUDIO I

Facundo Badano

Contenido

Introducción

Se realizó una posible solución en Java en cuanto al proceso de gestión de una cadena de supermercados.

Se busca darle al usuario final una experiencia simple y amena a la hora de utilizar el programa. Se entiende que está pensado para utilizarse en el día a día con alta frecuencia. Se obtuvo mucho conocimiento y se lograron grandes avances, simplificaciones en base al testeo del sistema.

En este caso la razón de poder llegar a este nivel de simplificación (se cree se puede seguir mejorando considerablemente) se debe a probar en sí el sistema y llevarlo a un alto grado de intensidad.

Problema planteado

El problema planteado en este caso refiere a una cadena de Supermercados (GEANT) donde se ofrece un programa capaz de gestionar y controlar el stock de todos los productos y sucursales existentes.

Este programa de gestión necesita como mínimo funcionalidades básicas como pueden ser:

- Incorporar un nuevo producto a la cadena de supermercados.
- Agregar stock a un producto existente.
- Simular la venta de un producto (reducir el stock de un producto existente
- Eliminar productos que ya no se venden (por no ser comercializados más).
- Dado un código de producto, indicar las existencias del mismo en el almacén.
- Listar todos los productos registrados, ordenados por nombre, presentando además su stock.

A la hora de realizar estas funcionalidades se tuvo que dejar de lado la funcionalidad de forma abstracta y tratar de bajarlo a tierra, a tratar de analizarlo como un proceso físico real. De esta forma se puede ver más fácilmente las responsabilidades de cada clase del proyecto. Por ejemplo: Si bien lo que se vende es un producto, ¿quién es el responsable de vender el producto?, ¿desde donde sale la orden?, ¿siempre es el mismo agente quien lo solicita?

Todos estos aspectos se tuvieron en cuenta para llegar a lo que puede ser una posible solución de las innumerables que pueden llegar a haber.

Otro aspecto relevante para tener en cuenta es pensar el proyecto a futuro. Si bien hoy la cadena de supermercados tiene 10 sucursales, hay que tener en cuanta como puede variar ese número. Es prácticamente imposible que de un día para el otro se den de alta 20 sucursales más. Se tiene que tratar de anticipar como pueden ser los cambios que se pueden dar en la cadena de supermercados y que en el programa esté contemplado para asimilarlo de la forma más simple y correcta posible.

Análisis de alternativas

Algoritmos

Pseudocódigo de métodos clase Sucursal:

```
insertarProductos:
Se inserta producto a la sucursal actual


insertarProductos (unProducto : de tipo Producto)
COM
    SI (stockSucursal.esVacio() <> VERDADERO) ENTONCES
        TElementoAB<Producto> elementoExistente <-- stockSucursal.buscar(unProducto);
        SI (elementoExistente = NULO) ENTONCES
            TElementoAB<Producto> elementoAux <-- new TElementoAB(unProducto.etiqueta, unProducto);
            stockSucursal.insertar(elementoAux);
        SINO
            Producto productoActual <-- elementoExistente.Datos;
            productoActual.agregarStock(unProducto.Stock);
        FIN SI
    SI NO
        TElementoAB<Producto> elementoAux <-- new TElementoAB(unProducto.Etiqueta, unProducto);
        stockSucursal.insertar(elementoAux);
    FIN SI
FIN

Tiempo orden de ejecución: O(N^2) Ya que el peor caso sería que se recorra todo el arbol (buscar)
                                  y que luego se recorra todo el arbol nuevamente (insertar).
```

Pseudocódigo de métodos clase Producto:

```
Pseudocódigo de métodos en clase Producto

agregarStock:
Incrementa el stock del producto por el numero recibido.

agregarStock(cantidad : de tipo Entero)
COM
    SI (cantidad > 0 ) ENTONCES
        stockActual += cantidad
    SINO
        Imprimir en pantalla ("Error. La cantidad a incrementar no puede ser negativa.");
    FIN SI
FIN

Tiempo orden de ejecución: O(1)
```

```
restarStock:
Decrementa el stock del producto por el numero recibido.

agregarStock(cantidad : de tipo Entero)
COM
    SI (cantidad > 0 ) ENTONCES
        stockActual += cantidad
    SINO
        Imprimir en pantalla ("Error. La cantidad a incrementar no puede ser negativa.");
    FIN SI
FIN

Tiempo orden de ejecución: O(1)
```

Pseudocódigo de métodos clase Geant:

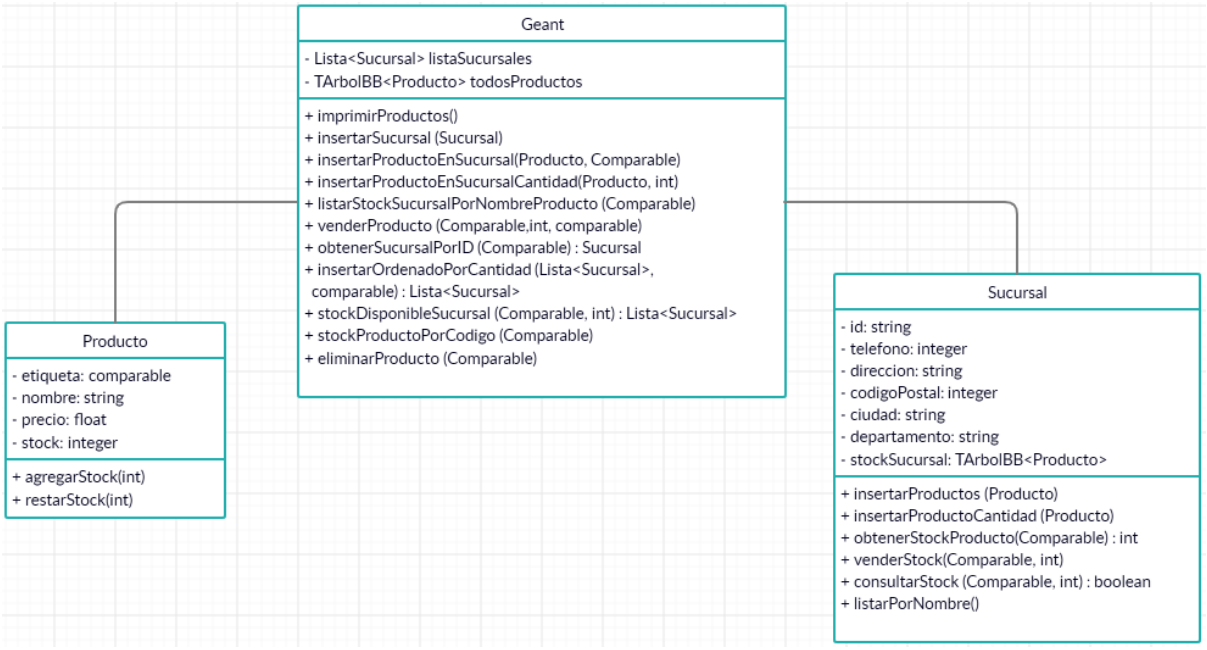
```
insertarSucursal:
Se inserta una nueva sucursal en el listado de sucursales.

insertarSucursal (unaSucursal : de tipo Sucursal)
COM
    Nodo nodoSucursal <-- new Nodo (unaSucursal.id, unaSucursal)
    listaSucursales.insertar(nodoSucursal)
FIN

Orden de tiempo de ejecucion: O(N) se recorre toda la lista para insertar al final
```

Selección y justificación de alternativa a implementar

Se decidió implementar el sistema en base a 3 clases principales. Geant (clase principal), Sucursal y Producto.

Desde Geant es donde se realizan todas las funcionalidades, la misma cuenta con una lista de sucursales. Se decidió utilizar lista ya que al ser pocas sucursales y el cambio en el listado va a ser mínimo se prioriza ante un árbol que cuenta con 2 referencias. También la clase Geant cuenta con un árbol de productos donde se hará la carga inicial.

A su vez la clase Sucursal cuenta con un árbol de productos donde allí se encuentra su stock.



Desde la clase Geant se cruzan los datos del listado de productos general con las sucursales para hacer efectivas las cargas de stock.

Todas las demás consultas se llevan a cabo desde Geant, siendo esta la clase que da la orden a la sucursal que corresponda para que lleve a cabo la acción interna.

Conclusiones

El eslabón más fuerte de la solución brindada es la centralización desde donde se emiten las ordenes, todo se ejecuta desde la clase Geant, siendo este ente el encargado de transmitir a las clases necesarias las ordenes que deben realizar.

Esto permite entre otras cosas, al usuario final tener una mejor experiencia ya que realiza todas las funcionalidades desde la misma sección. Otro punto para destacar es que, si se cambia tanto la forma de trabajo como si se quieren agregar nuevas funcionalidades va a resultar más simple realizarlo.

El programa funciona para todo tipo de negocio de venta con control de stock. Ejemplo suponiendo que Geant decide abrir otro rubro donde tengan un local donde se venden solo artículos de tecnología electrónica, también podrían utilizar la misma solución haciéndole unos pequeños cambios dependiendo de la estructura de los nuevos productos.

Está pensada la solución a largo plazo para poder soportar y asimilar los posibles cambios a futuro.

Guía del usuario

La aplicación es un proyecto NetBeans que no cuenta con interfaz gráfica.

Por lo tanto, solo se debe abrir el proyecto y desde la clase Geant, donde se encuentra el main se pueden probar todas las funcionalidades que se podrán ver reflejadas por consola (por defecto ya vienen cargados ejemplos de todos los métodos implementados).