

Algoritmos y Estructuras de Datos I

**PROYECTO INDIVIDUAL – PARTE I**

VERONICA ECHEZARRETA

Contenido

[Introducción 3](#_Toc417411916)

[Problema planteado 3](#_Toc417411917)

[Análisis de alternativas 3](#_Toc417411918)

[Algoritmos 3](#_Toc417411919)

[Selección y justificación de alternativa a implementar 3](#_Toc417411920)

[Conclusiones 3](#_Toc417411921)

[Guía del usuario 3](#_Toc417411922)

# Introducción

Se realizó una simulación de una cadena de supermercados y su funcionamiento, ya sea agregando productos a la empresa, y a su vez, estos mismos productos a cada sucursal, así como eliminando productos de cada sucursal o listando los productos existentes en cada sucursal junto con su stock, etc.

## Problema planteado

Al simular el funcionamiento de una cadena de supermercados, se requieren distintas funcionalidades:

* Incorporar un nuevo producto a una sucursal del supermercado.
* Agregar stock a un producto existente en una sucursal.
* Simular la venta de un producto en una sucursal (reducir el stock de un producto existente). De no haber stock suficiente para la venta en esa sucursal, deberá indicarse la lista de sucursales que tengan el stock suficiente, ordenada por cantidad de producto.
* Eliminar productos que ya no se venden (por no ser comercializados más) en todas las sucursales del supermercado.
* Dado un código de producto, indicar las existencias de este en todas las sucursales, ordenada por sucursal.
* Listar todos los productos registrados, en una sucursal, ordenado por nombre de producto, presentando además su stock.
* Listar todos los productos registrados, ordenados por ciudad, barrio, y nombre de producto, presentando además su stock.

# Análisis de alternativas

### Alternativa 1

La primera alternativa supone el uso de arboles binarios de búsqueda para almacenar los productos en la empresa.

También se utilizan estos mismos para almacenar las sucursales en la empresa, y estas sucursales, a su vez, emplean árboles para almacenar los productos en cada una de ellas.

INSERTAR DIBUJO

### Alternativa 2

La segunda alternativa supone también el uso de arboles binarios de búsqueda para almacenar los productos en la empresa.

Para las sucursales se emplearían nuevamente los mismos, pero en lugar de almacenar los productos en arboles, se almacenarían en listas.

INSERTAR DIBUJO

* Descripción de al menos 2 alternativas de estructuras de datos para representar el problema. Se sugiere utilizar dibujos explicativos
* Análisis del costo de memoria de cada alternativa

### Algoritmos

#### Métodos de Empresa

##### Buscar Producto

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.buscarProducto(Comparable idProducto) : Producto**

COM

TElementoAB<Producto> unProducto ← arbolProductosBase.buscar(idProducto)

SI (unProducto <> nulo) ENTONCES

devolver unProducto.dato

SINO

devolver nulo

FIN SI

FIN

##### Insertar Producto

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.insertarProducto(Producto producto) : void**

COM

TElementoAB<Producto> unProducto ← nuevo TElementoAB(producto.etiqueta, producto)

Producto p ← buscarProducto(producto.etiqueta)

SI (p = nulo) ENTONCES

arbolProductosBase.insertar(unProducto)

FIN SI

FIN

##### Insertar productos desde un archivo

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.insertarProductosArchivo(String nombreArchivo) : void**

COM

String[] línea ← ManejadorArchivosGenerico.leerArchivo(nombreArchivo)

Producto producto

PARA (String data en línea) HACER

String[] datos = data.split("\"");

producto ← nuevo Producto(datos[0], datos[1])

producto.precio(datos[2])

insertarProducto(producto)

FIN PARA

FIN

##### Agregar stock a los productos de la empresa desde un archivo

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.agregarStockEmpresa(String nombreArchivo) : void**

COM

String[] línea ← ManejadorArchivosGenerico.leerArchivo(nombreArchivo)

Producto producto

PARA (String data en línea) HACER

String[] datos = data.split(",");

Comparable idProducto ← datos[0]

int stock ← datos[1]

producto ← buscarProducto(idProducto)

SI (producto <> nulo) ENTONCES

Producto p ← nuevo Producto(idProducto, producto.nombre)

TElementoAB<Producto> unProducto ← nuevo TElementoAB(p.etiqueta, p)

arbolProductosEmpresa.insertar(elemProducto)

TElementoAB<Producto> pNombre = nuevo TElementoAB(p.nombre, p);

arbolProductosEmpresaPorNombre.insertar(pNombre);

p.agregarStock(stock)

FIN SI

FIN PARA

FIN

##### Restar stock a un producto

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.** **restarStockEmpresa(Comparable clave, Integer cantidad) : Boolean**

COM

TElementoAB<Producto> elemProducto ← arbolProductosEmpresa.buscar(clave)

SI (elemProducto <> nulo) ENTONCES

int stockFinal ← elemProducto.dato.restarStock(cantidad)

SI (stockFinal <> -1) ENTONCES

devolver true

SINO

devolver false

FIN SI

SINO

devolver false

FIN SI

FIN

##### Eliminar un producto de la empresa

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.eliminarProducto(Comparable clave) : Boolean**

COM

arbolProductosEmpresa.eliminar(clave)

devolver true

FIN

##### Listar productos por nombre

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.listarPorNombre() : void**

COM

SI (NOT arbolProductosEmpresaPorNombre.esVacio()) ENTONCES

listarPorNombreImplementacion(arbolProductosEmpresaPorNombre.raiz)

FIN SI

FIN

**Empresa.listarPorNombreImplementacion (TElementoAB<Producto> producto) : void**

COM

SI(producto.hijoIzq <> nulo) ENTONCES

listarPorNombreImplementacion(producto.hijoIzq)

FIN SI

imprimir(producto.nombre + producto.stock)

SI(producto.hijoDer <> nulo) ENTONCES

listarPorNombreImplementacion(producto.hijoDer)

FIN SI

FIN

##### Buscar una sucursal

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.buscarSucursal(Comparable idSucursal) : Sucursal**

COM

TElementoAB<Sucursal> elemSucursal ← arbolSucursales.buscar(clave)

SI(elemSucursal <> nulo) ENTONCES

Devolver elemSucursal.datos

SINO

Devolver nulo

FIN SI

FIN

##### Buscar una sucursal por departamento

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.buscarSucursalPorDepartamento(Comparable idSucursal) : Sucursal**

COM

TElementoAB<Sucursal> elemSucursal ← arbolSucursalesPorDepartamento.buscar(clave)

SI(elemSucursal <> nulo) ENTONCES

Devolver elemSucursal.datos

SINO

Devolver nulo

FIN SI

FIN

##### Insertar una sucursal

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.insertarSucursal (Sucursal sucursal) : void**

COM

TElementoAB<Sucursal> elemSucursal ← nuevo TElementoAB(sucursal.id, sucursal)

Sucursal s ← buscarSucursal(sucursal.id)

SI (s = nulo) ENTONCES

arbolSucursales.insertar(elemSucursal)

FIN SI

FIN

##### Insertar una sucursal por departamento

Lenguaje natural:

Asdasdasdasd

Precondiciones:

* dssdfsd

Postcondiciones:

* sdfsdf

Pseudocódigo:

**Empresa.insertarSucursalPorDepartamento (Sucursal sucursal) : void**

COM

String stringCompuesto;

stringCompuesto = sucursal.departamento + ", " + sucursal.ciudad + ", "

+ sucursal.codigoPostal + ", " + sucursal.id

TElementoAB<Sucursal> elemSucursal ← nuevo TElementoAB(stringCompuesto ,sucursal)

Sucursal s ← buscarSucursal(sucursal.id)

SI (s = nulo) ENTONCES

arbolSucursalesPorDepartamento.insertar(elemSucursal)

FIN SI

FIN

# Selección y justificación de alternativa a implementar

Breve fundamentación de la elección.

Descripción detallada de la alternativa implementada (dibujos, diagramas, aclaraciones sobre clases e interfaces).

# Conclusiones

Resumen de las características más relevantes del producto. ¿Por qué debería el cliente elegir mi solución?

# Guía del usuario

* Instrucciones (breves) para el uso de la aplicación.
* Consideraciones de instalación (si corresponde) y requerimientos no funcionales para la operación.