Ingeniería de Sistemas e Informática y Software

24513: Algoritmos y Estructuras de Datos

Solución Informática para Registro y Cálculo de Ventas por Cliente

Versión JDK: 22

Versión Net Beans: 19

Autores

Condori Yucra, Giancarlo - U23269935

Navarro Julián, Fadia - U23212956

Villavicencio Huaman, Miladia Paloma - U23214051

Quispe Horikawa, Renzo Felipe - U23224444

Lima - Perú

2025

ÍNDICE

1. Descripción del problema

Actualmente, la empresa TANVO enfrenta dificultades en el manejo y control de su proceso de ventas. El registro de productos, el control del inventario y el cálculo de importes se realizan de forma manual o con herramientas poco eficientes (como hojas de cálculo), lo que genera errores frecuentes, pérdida de información, demoras en la atención al cliente y una baja capacidad para generar reportes en tiempo real.

Esta situación afecta directamente a la productividad de los trabajadores, la satisfacción del cliente y la toma de decisiones administrativas, ya que no se cuenta con un sistema confiable que centralice y automatice la información relacionada con las ventas.

Ante este problema, se plantea el desarrollo de una aplicación de escritorio en Java, que automatice el registro de productos (código, nombre, precio, cantidad), gestione las compras de los clientes, calcule automáticamente el importe por producto y el total de la compra, y facilite el seguimiento del inventario.

Además, debido al crecimiento del negocio y la necesidad de manejar grandes volúmenes de datos estructurados, se recomienda implementar estructuras de datos como arreglos de objetos para almacenar productos, matrices para representar compras múltiples, listas enlazadas para el historial de ventas, pilas y colas para la gestión de atención o pedidos en orden de llegada, y árboles binarios de búsqueda (ABB) para organizar y buscar productos de manera eficiente.

A medida que la empresa TANVO experimenta un crecimiento sostenido en sus operaciones y volumen de transacciones, se vuelve esencial contar con una arquitectura de software que no solo automatice procesos, sino que también mantenga un alto rendimiento en el manejo de información. En este sentido, el uso adecuado de estructuras de datos permite optimizar el almacenamiento, la búsqueda y la manipulación de la información crítica del sistema. Gracias a lo enseñado en clase y con el uso de otras fuentes de información, hemos podido implementar estructuras como arreglos, listas enlazadas, árboles binarios de búsqueda o colas. Esta estrategia permite reducir tiempos de respuesta, evitar redundancias y minimizar errores humanos, proporcionando una base sólida para la toma de decisiones informadas, la satisfacción del cliente y el control interno del negocio, alineándose con las mejores prácticas en ingeniería de software (Goodrich, Tamassia & Goldwasser, 2014; Knuth, 1997).

2. Objetivos de la solución

2.1 Objetivo General

Desarrollar una aplicación de escritorio en Java que automatice el proceso de ventas por cliente utilizando estructuras de datos dinámicas (colas, pilas, listas, árboles y arreglos), integradas a una interfaz gráfica educativa e interactiva.

2.2 Objetivos específicos

- Automatizar el cálculo del importe por producto y del total de compra.
- Utilizar estructuras de datos para organizar, procesar y mostrar la información:
- Arreglos para el catálogo de productos.
- Listas enlazadas para productos comprados por cliente.
- Pilas para historial de deshacer/últimas operaciones.
- Colas para gestión de clientes en espera.
- Árbol binario para búsqueda y ordenamiento de productos.
- Visualizar en tiempo real el estado de las estructuras utilizadas.
- Crear una interfaz gráfica clara que guíe al usuario durante todo el proceso de venta.

3. Funcionalidad de la aplicación

La presente aplicación de escritorio ha sido desarrollada en Java y tiene como objetivo facilitar el cálculo de ventas de manera rápida y práctica. Está diseñada para registrar

productos de forma momentánea durante el proceso de venta, sin necesidad de mantener un inventario previo. Funciona como una calculadora de ventas que permite sumar productos, calcular importes individuales y determinar el total general de manera dinámica.

Desde el punto de vista algorítmico, la aplicación realiza las siguientes **operaciones principales**:

1. **Cálculo del importe por producto**:
   Al ingresar la cantidad y el precio unitario, la aplicación multiplica ambos valores para obtener el importe de ese producto.
   `importe = cantidad × precio`
2. **Acumulación del total general**:
   Cada vez que se agrega un nuevo producto, el importe calculado se **suma al total general** de la venta.
   `total = total + importe_actual`
3. **Almacenamiento temporal en una tabla (array o lista)**:
   Cada producto registrado se guarda temporalmente en una estructura de datos (por ejemplo, un arreglo o lista), que se muestra en una tabla al usuario. Esto permite visualizar todos los productos agregados hasta ese momento.
4. **Re-inicialización de datos**:
   Cuando el usuario desea iniciar una nueva venta, la aplicación **vacía la tabla y reinicia los valores acumulados**, borrando todos los registros anteriores.

Estas operaciones algorítmicas permiten que el sistema funcione de forma eficiente, brindando una experiencia fluida y sin errores de cálculo para el usuario. A pesar de su simplicidad, la aplicación aplica principios fundamentales de programación como el uso de estructuras de datos (para almacenar los productos de la venta), operaciones aritméticas básicas (multiplicación y suma), y control de flujo (para limpiar o reiniciar los datos).

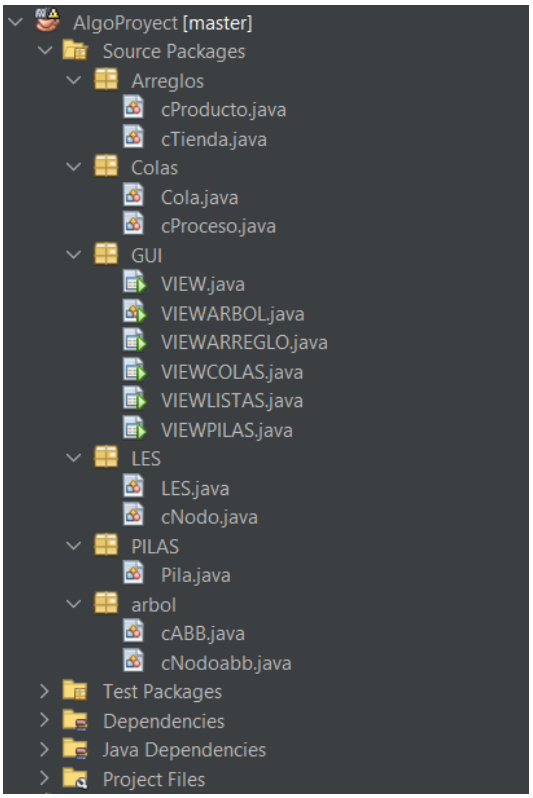
3.1 Funciones principales

- Ingreso manual de nombre, cantidad y precio por producto.
- Cálculo automático del importe por producto (cantidad × precio).
- Agregado del producto a una tabla temporal.
- Suma acumulativa del total general de la venta.
- Eliminación de los registros y reinicio del sistema de venta.

4. Estructura general de la aplicación

La aplicación está organizada en distintos paquetes que agrupan las clases según su funcionalidad dentro del sistema. Esta estructura permite mantener el código ordenado, modular y más fácil de mantener. La organización del proyecto sigue los principios de la programación orientada a objetos, separando la lógica de negocio, las estructuras de datos y la interfaz gráfica.

A continuación, se muestra el gráfico con los paquetes y clases desarrolladas:



4.1 Descripción de los paquetes

- **Arreglos**

  Contiene clases relacionadas con el manejo de productos y tiendas usando arreglos.
  - `cProducto.java`: Representa un producto con sus atributos.
  - `cTienda.java`: Administra una colección de productos.
- **Colas**

  Contiene la estructura de datos tipo cola y su lógica de proceso.
  - `Cola.java`: Implementa una cola.
- **GUI**

  Contiene todas las ventanas de la interfaz gráfica del usuario.
  - `VIEW.java`: Ventana principal del sistema.

- `VIEWARBOL.java`, `VIEWARREGLO.java`, `VIEWCOLAS.java`, `VIEWLISTAS.java`, `VIEWPILAS.java`: Vistas específicas para cada estructura.

- **LES**

  Lista enlazada secuencial para manejar datos de forma dinámica.
  - `LES.java`: Implementación de la lista.
  - `cNodo.java`: Nodo para la lista enlazada.
- **PILAS**

  Contiene la estructura tipo pila.
  - `Pila.java`: Implementación de la pila con sus operaciones.
- **arbol**

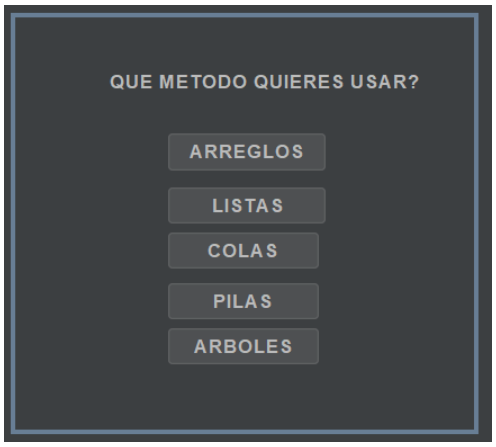
  Manejo de árboles binarios de búsqueda.
  - `cABB.java`: Árbol binario de búsqueda (ABB).
  - `cNodoabb.java`: Nodo del ABB.

5. Formularios del programa

A continuación, se presentan las pantallas (formularios) desarrolladas en la interfaz gráfica de la aplicación. Cada ventana permite al usuario interactuar con una estructura de datos distinta (arreglos, pilas, colas, listas, árboles) o acceder a la vista principal del sistema. Las pantallas fueron diseñadas usando componentes Swing de Java, organizados en paneles dentro de la clase `VIEW.java` y sus subclases.

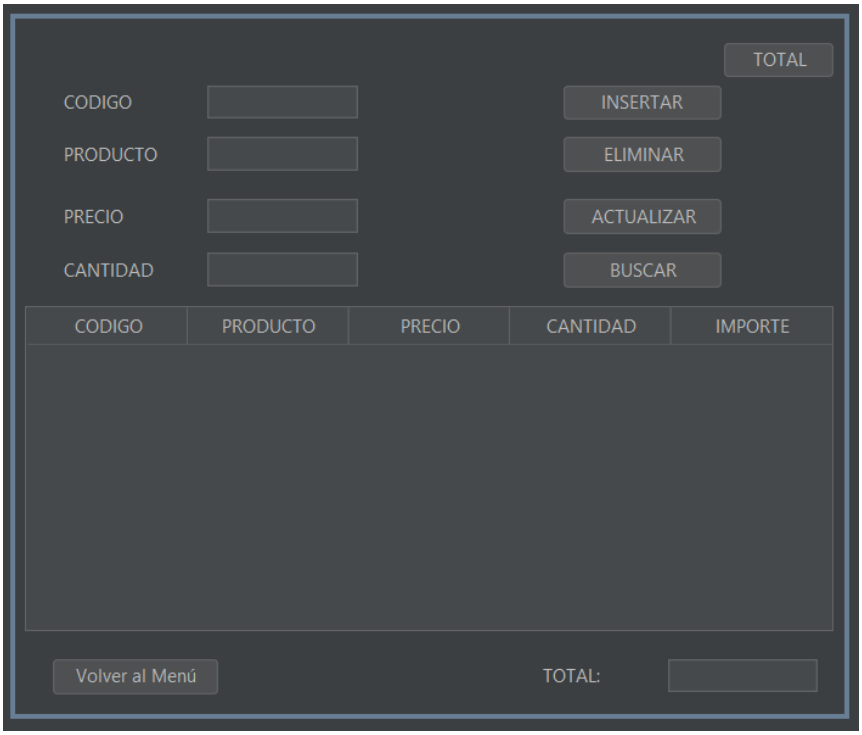
Ventana Principal – VIEW.java

La ventana principal permite al usuario seleccionar qué estructura de datos desea utilizar. Desde aquí se puede acceder a las pantallas específicas para trabajar con arreglos, colas, pilas, listas o árboles. También puede mostrar el total acumulado y reiniciar la aplicación.



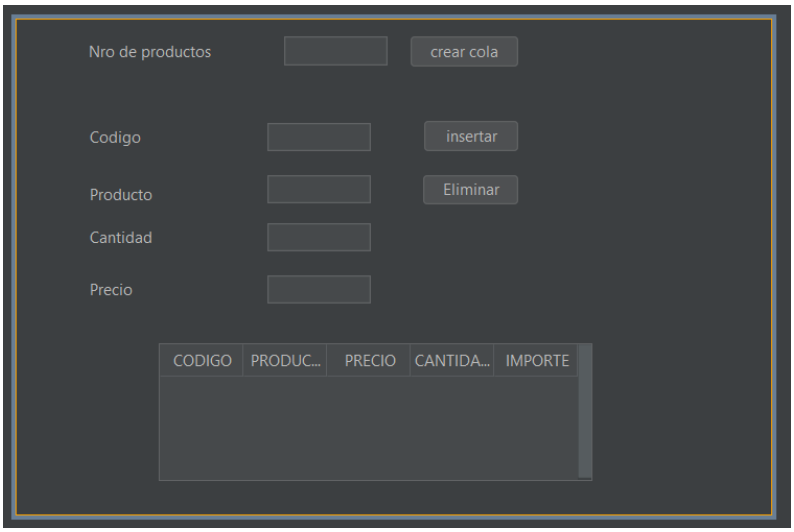
Vista de Arreglos – VIEWARREGLO.java

Permite al usuario ingresar productos utilizando un arreglo como estructura base. Se puede ingresar el nombre, cantidad y precio del producto, y la tabla muestra el importe por cada línea.



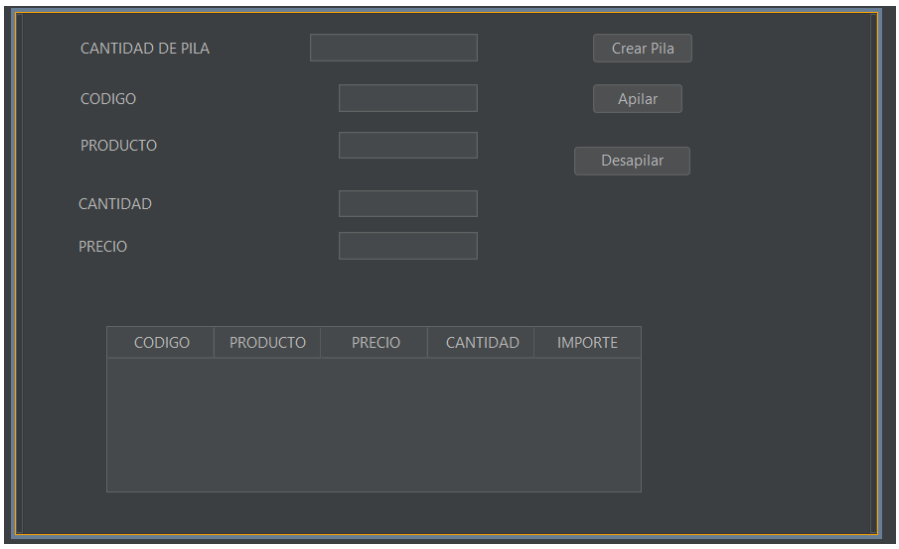
Vista de Colas – VIEWCOLAS.java

Simula el ingreso de productos o procesos en una cola. Los elementos se agregan en orden y se procesan de acuerdo con el principio FIFO (First In, First Out).



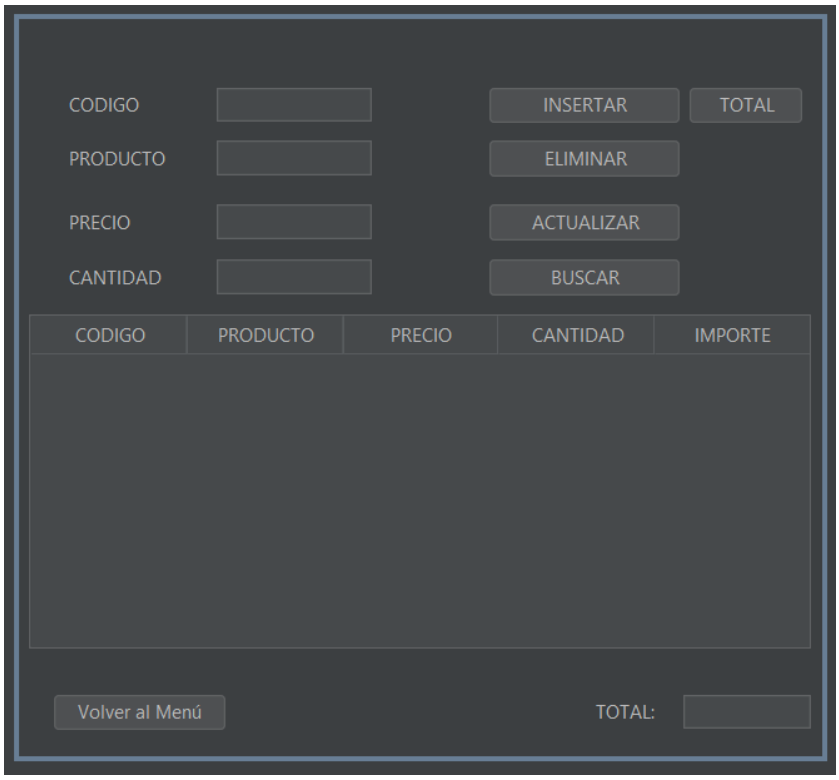
Vista de Pilas – VIEWPILAS.java

Permite ingresar productos o datos a una pila. El último elemento en entrar será el primero en salir (LIFO). Esta vista demuestra el uso de esta estructura de forma visual.



Vista de Listas – VIEWLISTAS.java

Esta ventana representa el uso de listas enlazadas simples para almacenar productos o datos temporalmente. Muestra cómo se conectan los nodos entre sí.



Vista de Árbol Binario – VIEWARBOL.java

Muestra un árbol binario de búsqueda (ABB) donde los productos se ordenan automáticamente por algún criterio (por ejemplo, nombre o precio). Se pueden agregar, eliminar o buscar nodos.

6. Conclusiones y recomendaciones

Conclusiones:

- Se logró automatizar el proceso de ventas en un entorno minorista, reduciendo errores manuales y mejorando la eficiencia operativa.
- La aplicación permite registrar productos, calcular automáticamente importes por producto y generar el total a pagar por cliente, cumpliendo con los requisitos funcionales del negocio.
- La implementación de estructuras de datos (pilas, colas, listas enlazadas, árboles y arreglos) facilitó el procesamiento eficiente y organizado de la información.
- El uso de una interfaz gráfica amigable permitió al usuario una experiencia sencilla e intuitiva, adecuada tanto para fines comerciales como educativos.
- La solución no solo mejora la gestión de ventas, sino que también ofrece un entorno interactivo para que estudiantes comprendan el funcionamiento de las estructuras de datos en aplicaciones reales.
- El sistema está preparado para ser ampliado con funciones adicionales, gracias a su estructura modular y al uso de buenas prácticas en el desarrollo.

Recomendaciones:

1. Implementar persistencia de datos
Actualmente, el sistema guarda los productos solo de forma temporal y no mantiene un inventario real. Sería útil incorporar una base de datos (como MySQL) que permita almacenar la información de productos, ventas y clientes de manera permanente. Esto facilitará consultar el historial de ventas y gestionar el inventario de forma más organizada.

2. Validación de entrada de datos
Para evitar errores que puedan surgir al ingresar productos, cantidades o precios, es importante agregar controles en la interfaz que verifiquen que los datos sean correctos como excepciones. Por ejemplo, impedir que se introduzcan cantidades o precios negativos y asegurarse que los campos obligatorios estén completos. Esto ayudará a que el sistema sea más preciso y fácil de usar.

3. Manejo de excepciones y robustez
Es fundamental que la aplicación pueda manejar de forma adecuada cualquier error o situación inesperada, para que no se bloquee o cierre de forma abrupta. Incorporar un sistema que capture estas excepciones facilitara la experiencia al usuario, haciendo que el programa sea más estable y confiable.

4. Funcionalidad multiusuario y roles
Para un manejo más profesional, sería conveniente incluir un sistema de acceso donde diferentes usuarios puedan ingresar con perfiles distintos, como vendedor o administrador. De esta manera, se podría controlar qué funciones puede usar cada persona, aumentando la seguridad y organización dentro del sistema.

7. Referencias Bibliográficas

Goodrich, M., Tamassia, R., & Goldwasser, M. (2014). *Data Structures and Algorithms in Java* (6th ed.). Wiley.

Knuth, D. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms* (3rd ed.). Addison-Wesley.