FACULTAD DE INGENIERÍA

Ingeniería de Sistemas e Informática y Software

TRABAJO FINAL – TALLER DE PROGRAMACIÓN (24578)

"Desarrollo e implementación de un sistema de gestión de inventario y ventas para la distribuidora textil Tezza Allure S.A.C"

Autores:

AGUIRRE SALAS, Gianmarco U23315612

CHAHUARES CCOPACATI, Adrian Willman U22331452

INJOQUE GUADAÑA, Fernando U23208728

RAMIREZ ZUÑIGA, Axel Israel U23213913

VADILLO FERNÁNDEZ, Jesus Nazareno U22203604

Ciclo: Marzo 2024

Docente: JIMENEZ DRAGO, Raul Armando

Julio; 2024

Tabla de Contenido

Aspectos Generales

Descripción del Problema

En el contexto de una pequeña empresa de ropa peruana, la gestión de inventarios y ventas es una tarea esencial para asegurar la disponibilidad de productos, la satisfacción del cliente y la sostenibilidad del negocio (Lima, 2020). Muchas *mypes* en Perú aún utilizan métodos manuales o herramientas básicas para administrar sus operaciones (PHC Software Peru, 2021), lo cual puede resultar en errores, ineficiencias y pérdida de oportunidades de venta.

El problema central que enfrenta la distribuidora textil Tezza Allure S.A.C es la necesidad de una solución eficiente y automatizada para el manejo del inventario y las ventas. La falta de un sistema automatizado puede llevar a problemas como la sobreventa de productos, el desabastecimiento, y la dificultad para realizar un seguimiento preciso de las transacciones y el estado del inventario (Drucioc y García, 2021).

La solución propuesta es el desarrollo de un sistema de gestión de inventarios y ventas en Java, diseñado específicamente para las necesidades de una pequeña empresa de ropa. Este sistema debe permitir el control y seguimiento detallado de los materiales/prendas disponibles, facilitar la realización de ventas de manera rápida y precisa, y mantener un historial completo de todas las transacciones. Además, el sistema debe ser fácil de usar y adaptable a las necesidades específicas de la mype.

Objetivos del Sistema

El sistema de gestión de inventarios y ventas desarrollado en Java para Tezza Allure S.A.C tiene los siguientes objetivos:

Automatización del Inventario

- Facilitar el proceso de agregar nuevas prendas/materiales al inventario con validaciones adecuadas para asegurar la integridad de los datos.
- Permitir la búsqueda de prendas en el inventario por ID, asegurando una identificación rápida y precisa de los artículos disponibles.
- Ofrecer una visualización clara y detallada del inventario actual, incluyendo información sobre ID, nombre, precio, y cantidad en stock de cada prenda o material.

Gestión de Ventas

- Facilitar la realización de ventas permitiendo la selección de prendas/materiales y cantidades desde el inventario.
- Generar un recibo detallado para cada venta, mostrando el subtotal por prenda y el total de la venta.
- Actualizar automáticamente el inventario tras cada venta, reduciendo la cantidad de prendas vendidas.

Historial de Ventas

- Mantener un historial detallado de todas las transacciones de ventas, incluyendo información sobre los materiales/prendas vendidas, cantidades, subtotales, y número de recibo.

- Proporcionar una función para imprimir el historial de ventas, permitiendo una revisión y auditoría de las transacciones realizadas.

Manejo de Errores y Validaciones

- Implementar validaciones para asegurar que los datos ingresados sean correctos y coherentes (e.g., IDs positivos, precios mayores a cero, nombres sin números).
- Gestionar excepciones y errores de manera efectiva, proporcionando mensajes de error claros y manteniendo la estabilidad del sistema en caso de entradas inválidas o problemas operacionales.

Interfaz de Usuario Intuitiva

- Ofrecer un menú de opciones fácil de usar que permita a los usuarios navegar entre las diferentes funcionalidades del sistema de manera sencilla y eficiente.
- Asegurar que todas las interacciones del usuario con el sistema sean claras y directas, minimizando la curva de aprendizaje y facilitando el uso diario del sistema.

El código proporcionado implementa estos objetivos mediante una estructura modular en Java, utilizando clases como “Producto”, “Inventario”, “Venta”, y “HistorialVentas” para organizar y manejar los datos y operaciones del sistema. La clase principal “Main” actúa como la interfaz del usuario, proporcionando un menú interactivo para acceder a las diferentes funcionalidades del sistema. Esto permitirá a la distribuidora textil Tezza Allure S.A.C gestionar sus operaciones de manera más eficiente y efectiva, mejorando su capacidad para satisfacer las demandas de sus clientes y mantener un control riguroso sobre su inventario y ventas.

Diseño de la Aplicación

Funcionalidad de la Aplicación

El sistema diseñado para la pequeña empresa de ropa peruana tiene como objetivo principal gestionar eficientemente el inventario de prendas y las ventas realizadas. A continuación, se detallan las funcionalidades principales que ofrece la aplicación:

Gestión del Inventario

El módulo de gestión del inventario permite a los usuarios agregar nuevos productos, buscar productos existentes y visualizar el inventario actualizado en cualquier momento. Esto facilita el control preciso de las existencias de cada prenda, asegurando que siempre haya suficiente stock disponible para las ventas.

Tabla 1

Funcionalidad del módulo de gestión del inventario

Método	Descripción	Implementación
Agregar Producto al Inventario	Los usuarios pueden ingresar manualmente los detalles de un nuevo producto, incluyendo su ID único, nombre, precio unitario y cantidad inicial en stock. Esto asegura que cada prenda esté debidamente registrada en el sistema desde el momento de su ingreso al inventario.	• **Scanner:** Se utiliza para leer los datos del usuario. • **Condicionales:** Validan la entrada del usuario para asegurar que el ID, nombre, precio y cantidad son válidos. • **Objetos:** Se crea un nuevo objeto Producto con los datos ingresados. • **Métodos:** El método *agregarProducto* de la clase Inventario se encarga de agregar el producto al inventario.

		• **Bucles**: Aseguran que la entrada sea correcta (por ejemplo, hasta que el ID sea positivo).
Buscar Producto por ID	Permite buscar rápidamente un producto específico utilizando su identificador único. Esto es útil para verificar la disponibilidad y detalles de cualquier prenda sin necesidad de revisar físicamente el inventario.	• **Métodos:** El método *buscarProducto* de la clase Inventario busca un producto en el array productos. • **Condicionales:** Verifican si el producto existe y, si no, muestran un mensaje de error. • **Objetos:** Se retorna el objeto Producto encontrado.
Visualizar Inventario Completo	Proporciona una lista detallada de todos los productos actualmente en el inventario. Esta función incluye información crucial como ID, nombre, precio y cantidad disponible de cada material/prenda, permitiendo una gestión efectiva de las existencias.	• **Bucles:** Iteran sobre el array productos en la clase Inventario. • **Métodos:** El método *imprimirInventario* imprime los detalles de cada producto. • **Strings:** Se utilizan para construir las líneas de salida que muestran la información del producto.

Gestión de Ventas

El módulo de gestión de ventas facilita el proceso de venta de productos, asegurando un seguimiento preciso de las transacciones y la generación automática de recibos para cada venta realizada.

Tabla 2

Funcionalidad del módulo de gestión de ventas

Método	Descripción	Implementación
Realizar Venta	Los usuarios pueden seleccionar productos del inventario, especificar la cantidad que desean vender y generar automáticamente un recibo detallado. Esta función asegura que cada transacción sea registrada de manera precisa y eficiente.	• **Scanner:** Para leer las selecciones y cantidades de venta del usuario. • **Condicionales:** Aseguran que la cantidad vendida no exceda la cantidad en stock. • **Métodos:** *agregarProducto* y vender de las clases Venta y Producto, respectivamente. • **Bucles:** Permiten al usuario seleccionar múltiples productos para la venta.
Imprimir Recibo de Venta	Después de cada venta exitosa, se genera un recibo que detalla los productos vendidos, las cantidades, los precios unitarios y el total de la venta. Este recibo proporciona una confirmación tangible tanto para el cliente como para el registro interno de la empresa.	• **Métodos:** *imprimirRecibo* en la clase Venta crea y muestra el recibo. • **Strings:** Se utilizan para construir el recibo con todos los detalles de la venta. • **Objetos:** Los productos vendidos se almacenan en el array *productosVendidos*.

Historial de Ventas

El sistema incluye un registro completo de todas las ventas realizadas, proporcionando a la mype un historial detallado y accesible que puede ser consultado en cualquier momento.

Tabla 3

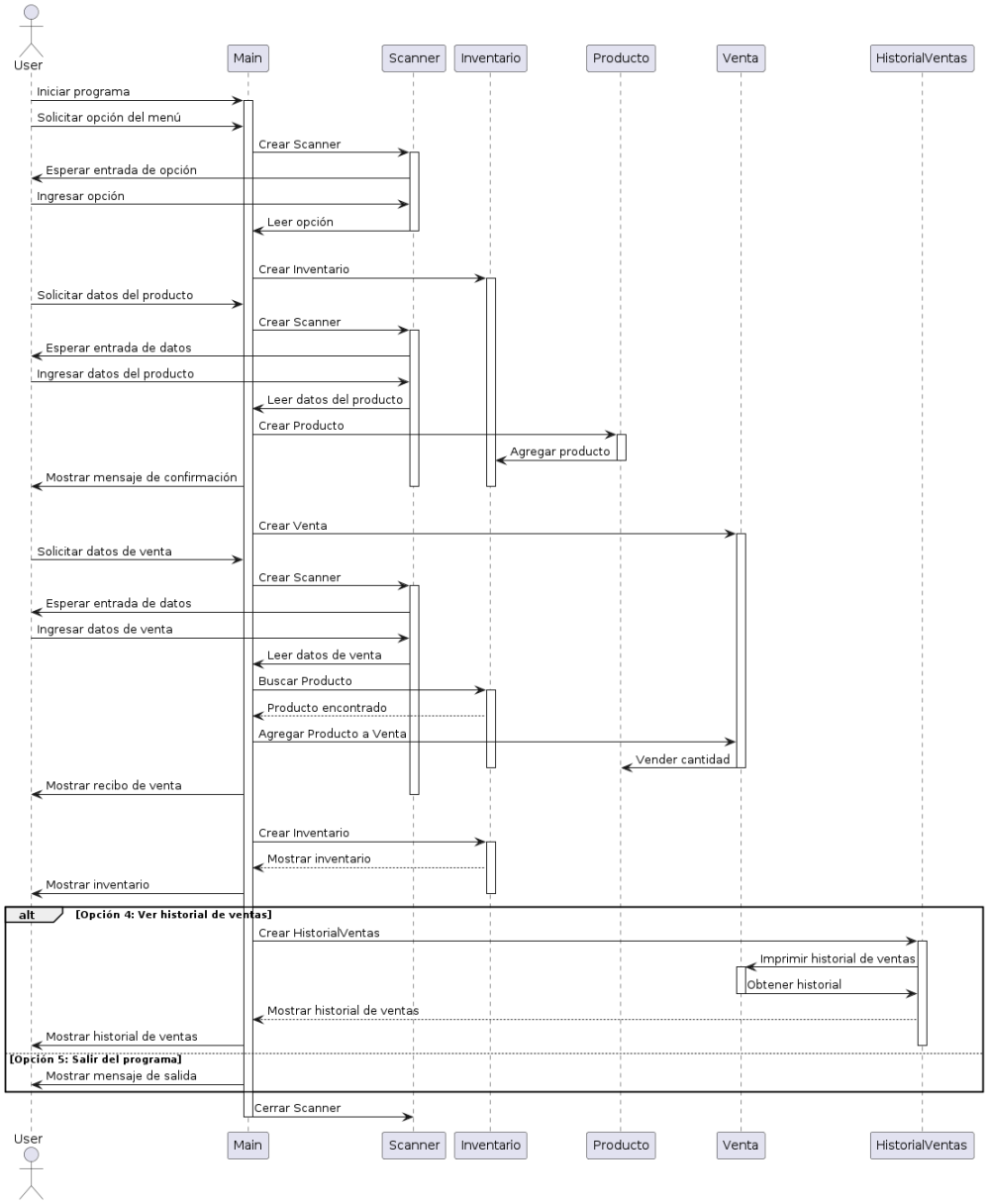
Funcionalidad del módulo de historial de ventas

Método	Descripción	Implementación
Visualizar Historial de Ventas	Permite acceder a un registro cronológico de todas las transacciones de ventas anteriores. Cada entrada en el historial incluye información como el número de venta, los productos vendidos, los subtotales y el total acumulado de ventas hasta la fecha.	• **Métodos:** *imprimirHistorialVentas* en la clase Venta. • **Bucles:** Iteran sobre el historial de ventas para mostrar cada transacción. • **Strings:** Construyen las líneas de salida para cada venta en el historial.
Auditoría de Ventas	Facilita la auditoría interna al proporcionar un seguimiento claro de todas las transacciones realizadas. Esto asegura la transparencia y precisión en la contabilidad de ventas de la empresa.	• **Métodos:** i*mprimirHistorial* en la clase HistorialVentas. • **Objetos:** Los datos de ventas se almacenan en un array bidimensional. • **Condicionales:** Verifican la consistencia y completitud de los datos en el historial.

Imágenes de la Solución

Figura 1

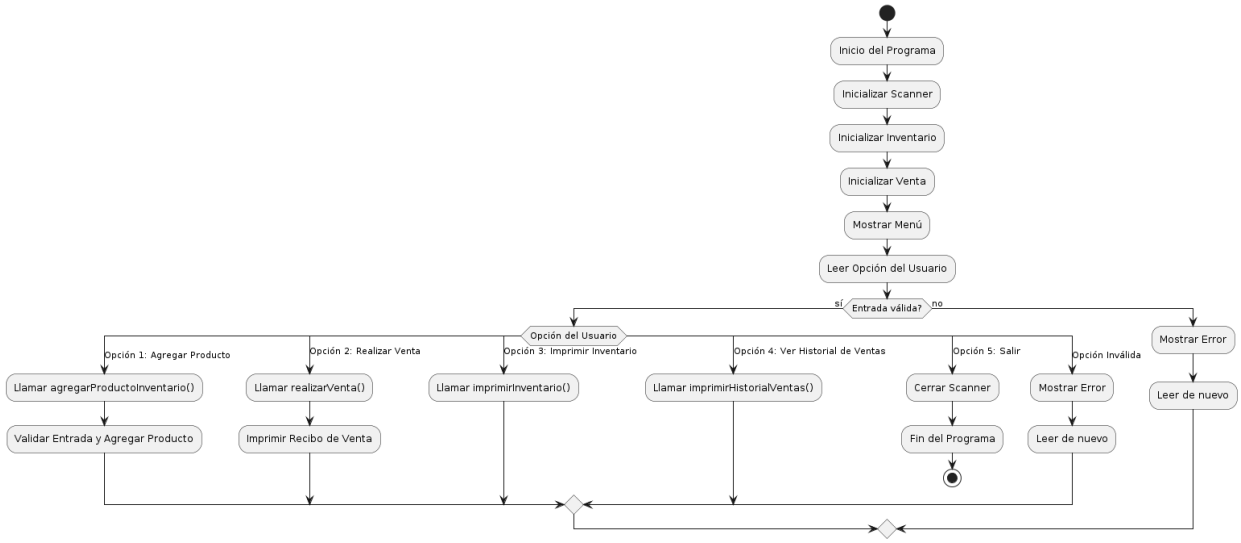
Diagrama de Secuencia de la Solución



Nota. Se utilizó el editor *PlantUML* para generar el diagrama de secuencia, el cual está disponible en su sitio web oficial: https://plantuml.com/es/.

Figura 2

Diagrama de Flujo de la Solución



Nota. Se utilizó el editor PlantUML para generar el diagrama de flujo.

Desarrollo de la Aplicación

Descripción de los Módulos

El sistema de gestión de inventarios y ventas está compuesto por varios módulos, cada uno con responsabilidades específicas. A continuación, se presenta una descripción detallada de las funcionalidades que realiza cada módulo:

Tabla 4

Funcionalidades por módulo

Módulo	Funcionalidades
Main	Controla la interacción del usuario con el sistema mediante un menú principal.
Producto	Representa las prendas del inventario, almacenando información sobre ID, nombre, precio y cantidad.
Inventario	Gestiona la colección de productos, permitiendo agregar, buscar y listar prendas.
Venta	Gestiona las ventas, permitiendo agregar productos a una venta, imprimir recibos y actualizar el inventario.
HistorialVentas	Mantiene un registro de todas las ventas realizadas, permitiendo su consulta y auditoría.

Tabla 5

Opciones del menú principal

Opción del Menú Principal	Descripción de la Funcionalidad
1. Agregar producto al inventario	Permite agregar una nueva prenda al inventario, solicitando ID, nombre, precio y cantidad inicial.
2. Realizar una venta	Permite realizar una venta seleccionando productos del inventario y especificando cantidades. Imprime un recibo al finalizar la venta.
3. Ver inventario	Muestra una lista de todas las prendas actualmente en el inventario.
4. Ver historial de ventas	Muestra el historial de todas las ventas realizadas, con detalles de cada transacción.

5. Salir	Finaliza la ejecución del programa.

Tabla 6

Opciones del menú de venta

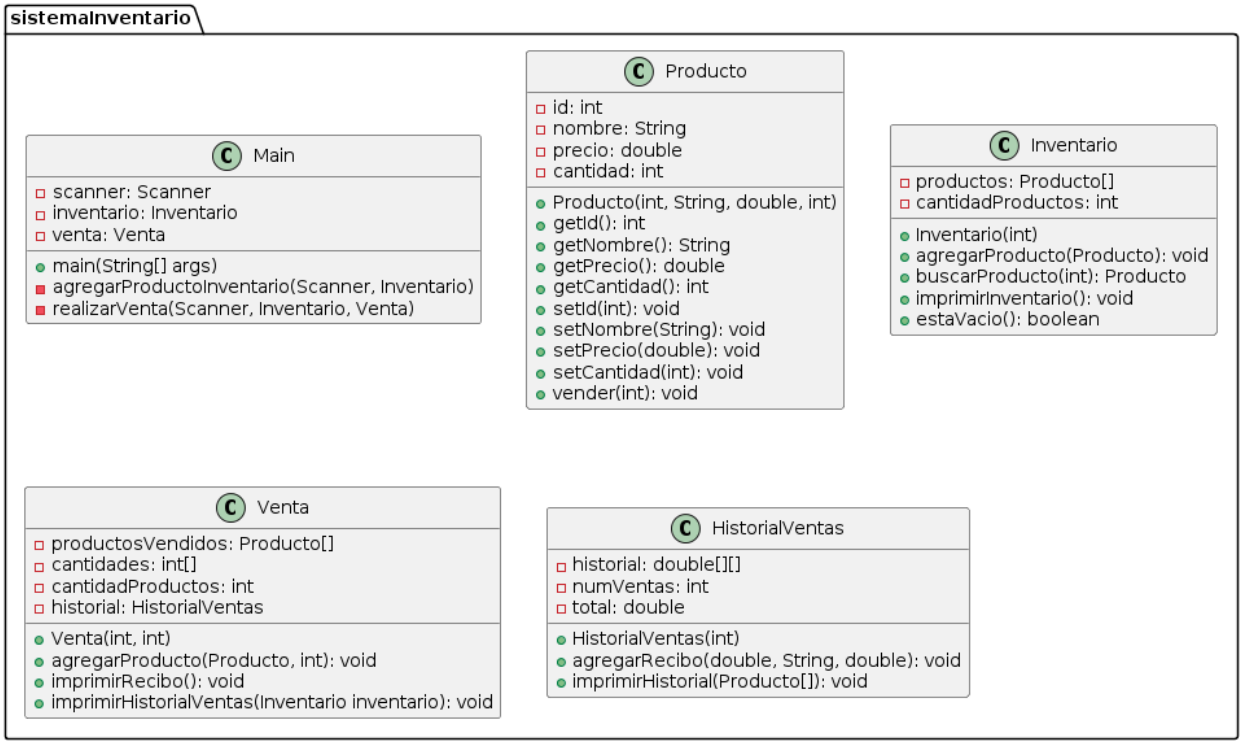
Opción del Menú de Venta	Descripción de la Funcionalidad
Ingresar ID del producto	Permite seleccionar un producto del inventario mediante su ID.
Ingresar cantidad a vender	Especifica la cantidad de la prenda seleccionada que se desea vender.
Ingresar 0 para finalizar	Finaliza la selección de productos y procede a imprimir el recibo de venta.

Diseño de las Clases

El sistema está estructurado en varias clases que representan los diferentes componentes y sus interacciones. En la Figura 3 se presenta un diagrama de clases y la descripción de las funcionalidades de sus métodos.

Figura 3

Diagrama de clases



Nota. Se utilizó el editor PlantUML para generar los diagramas UML (Lenguaje Unificado de Modelado).

Descripción de las Clases y sus Métodos

Paquete: sistemaInventario

Clase Main.

Tabla 7

Atributos y métodos de la clase Main

		Descripción
Atributos	scanner: Scanner	Utilizado para la entrada de datos del usuario.
	inventario: Inventario	Instancia de la clase Inventario para gestionar las prendas.
	venta: Venta	Instancia de la clase Venta para gestionar las ventas.
Métodos	main(String[] args)	Punto de entrada del programa, controla el flujo principal mediante un menú.
	agregarProductoInventario(Scanner, Inventario)	Permite agregar un nuevo producto al inventario.
	realizarVenta(Scanner, Inventario, Venta)	Permite realizar una venta y actualizar el inventario.

Clase Producto.

Tabla 8

Atributos y métodos de la clase Producto

		Descripción
Atributos	id: int	Identificador único de la prenda.
	nombre: String	Nombre de la prenda.
	precio: double	Precio de la prenda.
	cantidad: int	Cantidad disponible en stock.
Métodos	Producto(int, String, double, int)	Constructor de la clase Producto.
	getId(): int	Devuelve el ID de la prenda.
	getNombre(): String	Devuelve el nombre de la prenda.
	getPrecio(): double	Devuelve el precio de la prenda.

	getCantidad(): int	Devuelve la cantidad en stock de la prenda.
	setId(int): void	Establece el ID de la prenda.
	setNombre(String): void	Establece el nombre de la prenda.
	setPrecio(double): void	Establece el precio de la prenda.
	setCantidad(int): void	Establece la cantidad en stock de la prenda.
	vender(int): void	Actualiza la cantidad de la prenda tras una venta.

Clase Inventario.

Tabla 9

Atributos y métodos de la clase Inventario

		Descripción
Atributos	productos: Producto[]	*Array* que almacena los productos del inventario.
	cantidadProductos: int	Número actual de productos en el inventario.
Métodos	Inventario(int)	Constructor de la clase Inventario que inicializa el array de productos.
	agregarProducto(Producto): void	Agrega un nuevo producto al inventario.
	buscarProducto(int): Producto	Busca y devuelve un producto por su ID.
	imprimirInventario(): void	Imprime la lista de productos en el inventario.
	estaVacio(): boolean	Verifica si el inventario está vacío.

Clase Venta.

Tabla 10

Atributos y métodos de la clase Venta

		Descripción
Atributos	productosVendidos: Producto[]	Array que almacena los productos vendidos en una
	cantidades: int[]	Array que almacena las cantidades vendidas de cada producto.

	cantidadProductos: int	Número actual de productos vendidos en la transacción.
	historial: HistorialVentas	Instancia de la clase HistorialVentas para registrar las ventas.
Métodos	Venta(int, int)	Constructor de la clase Venta que inicializa los arrays de productos vendidos y el historial de ventas.
	agregarProducto(Producto, int): void	Agrega un producto a la venta y actualiza el inventario.
	imprimirRecibo(): void	Imprime el recibo de la venta con el total.
	imprimirHistorialVentas(Inventario inventario): void	Imprime el historial de todas las ventas registradas.

Clase HistorialVentas.

Tabla 11

Atributos y métodos de la clase HistorialVentas

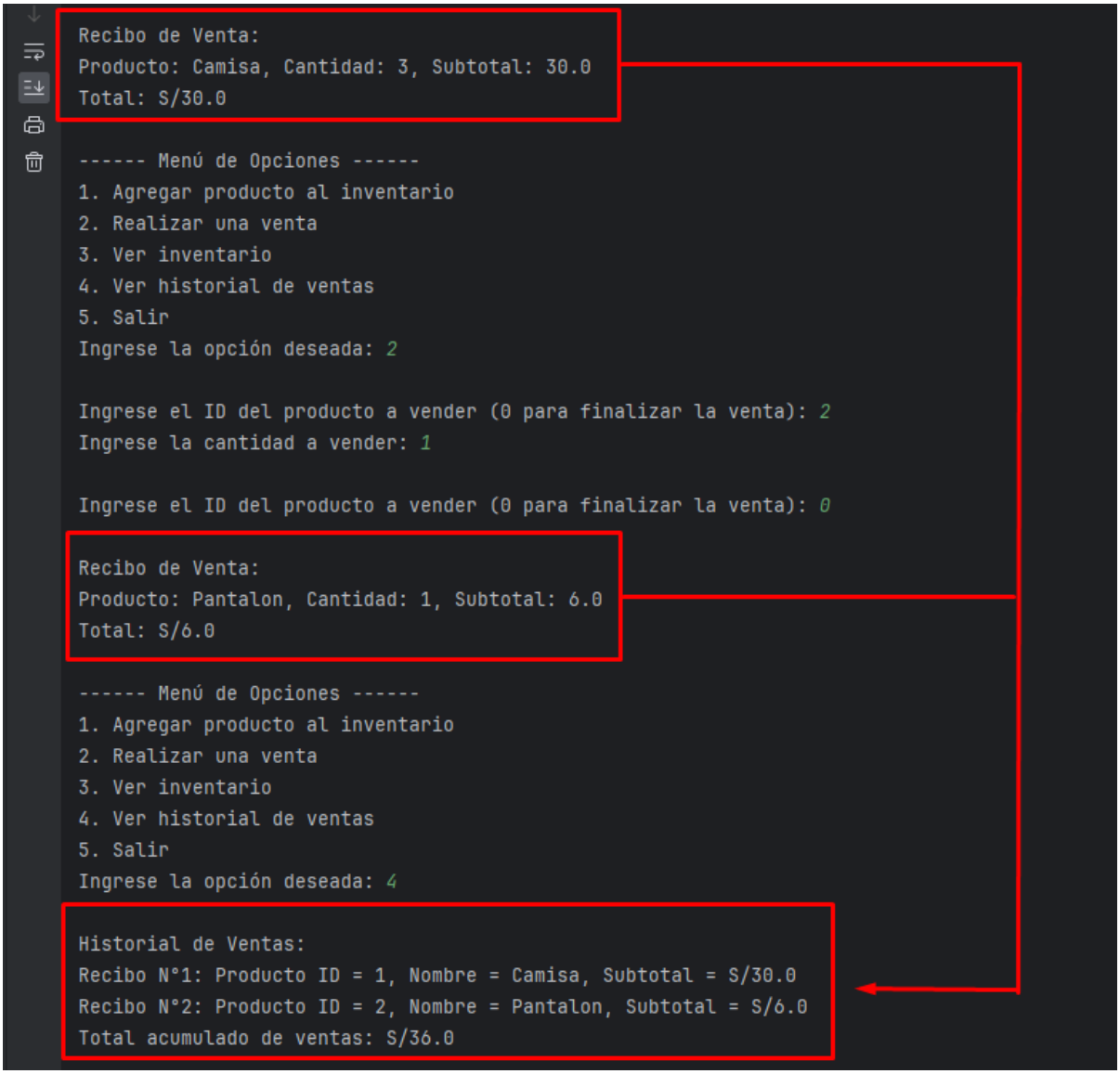
		Descripción
Atributos	historial: double[][]	Array bidimensional que almacena los registros de ventas.
	numVentas: int	Número actual de ventas registradas.
	total: double	Total acumulado de las ventas.
Métodos	HistorialVentas(int)	Constructor de la clase HistorialVentas que inicializa el array de registros de ventas.
	agregarRecibo(double, String, double): void	Agrega un recibo al historial de ventas.
	imprimirHistorial(Producto[]): void	Imprime el historial completo de ventas.

Este diseño modular y bien estructurado permite a Tezza Allure S.A.C gestionar su inventario y ventas de manera eficiente, proporcionando una solución fácil de usar para sus necesidades diarias.

Imágenes de Entrada y Salida

Figura 4

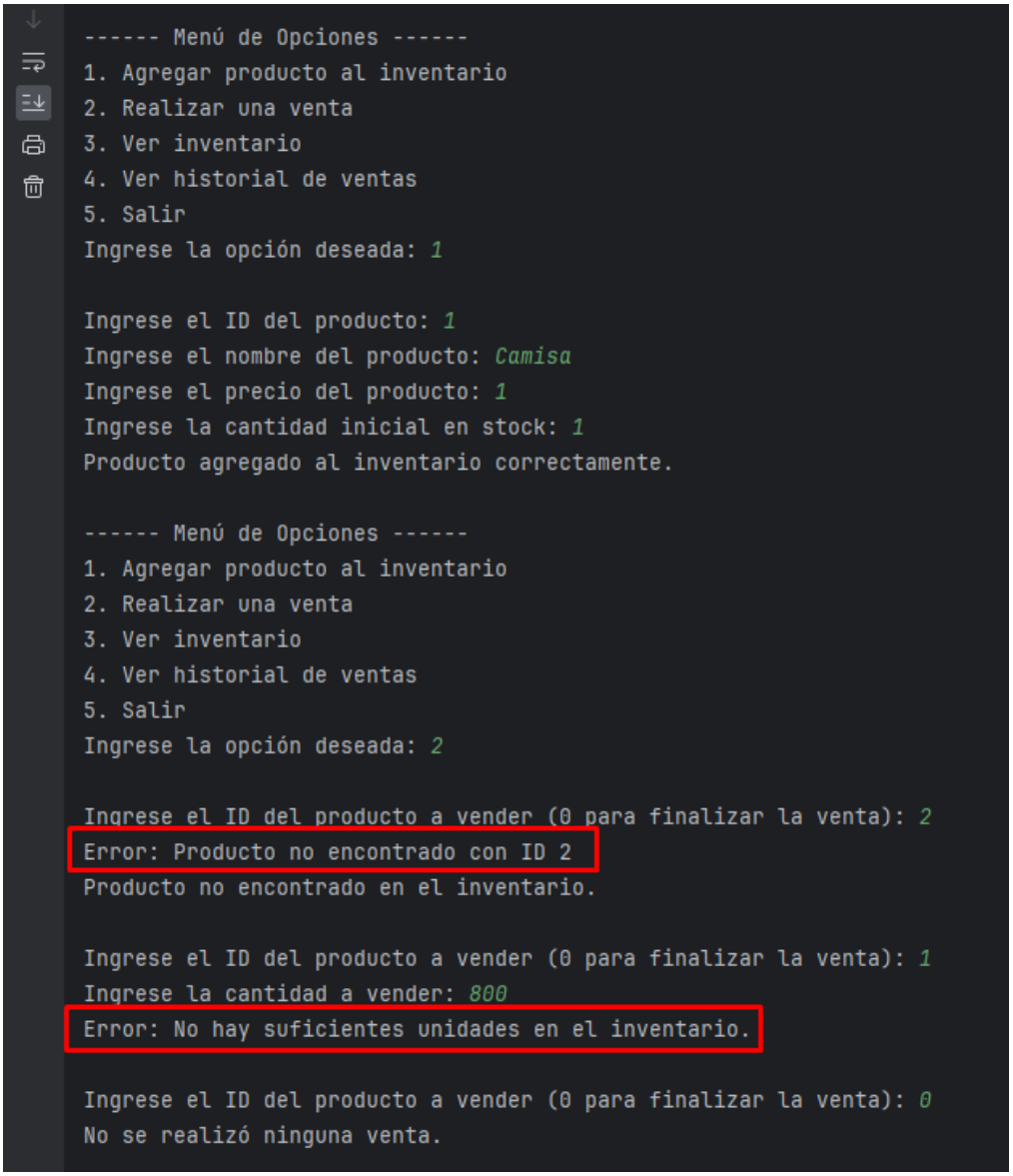
Ejemplo de HistorialVentas para dos o más ventas



Nota. El historial de ventas almacena recibos separados con IDs, productos, cantidades y subtotales distintos, incluso cuando estos se "reinician" por venta.

Figura 5

Manejo de excepciones



Nota. En el primer recuadro (arriba), se realiza la búsqueda del producto en Inventario y el resultado se devuelve a Venta. En el segundo recuadro (abajo), se produce un *IllegalArgumentException* en Venta cuando la cantidad solicitada no está disponible en el stock.

Conclusiones y Recomendaciones

Conclusiones

Durante el desarrollo del proyecto de gestión de inventarios y ventas en Java, se han obtenido varios aprendizajes fundamentales sobre el uso y aplicación de conceptos clave de programación. A continuación, se destacan las conclusiones principales:

1. **Uso de Estructuras de Control (Condicionales y Bucles):** Se aplicaron condicionales para validar datos de entrada como IDs positivos y precios mayores a cero, asegurando la integridad de los datos ingresados. Los bucles, por otro lado, fueron esenciales para iterar sobre arrays, facilitando operaciones como la búsqueda de productos por ID y la impresión del inventario completo de manera eficiente.
2. **Manipulación de Strings:** La manipulación de strings desempeñó un papel importante en la construcción de mensajes de error detallados y en el formateo adecuado de los recibos de ventas y otros mensajes de salida del sistema. Esto permitió una comunicación clara y efectiva con el usuario durante todo el proceso de interacción con el sistema.
3. **Uso de Objetos y Clases:** La creación y gestión de objetos como Producto, Inventario, Venta, y HistorialVentas permitieron una organización eficiente de los datos y las operaciones del sistema. Las clases proporcionaron una estructura modular que facilitó la gestión y mantenimiento del código.
4. **Implementación de Métodos y Funcionalidades:** Cada método implementado en las clases cumplió un rol específico y contribuyó al funcionamiento integral del sistema. Desde la adición de productos al inventario hasta la generación de recibos de ventas y la auditoría del historial de transacciones, cada función se diseñó para interactuar fluidamente con el usuario a través de una interfaz intuitiva y fácil de usar.

Recomendaciones

Basado en las conclusiones obtenidas durante el desarrollo del proyecto, para enriquecer el sistema de gestión de inventarios y ventas, podrían considerarse las siguientes funciones adicionales:

1. **Gestión de Proveedores:** Implementar un módulo que permita registrar y gestionar información de proveedores. Esto incluiría detalles como nombre, contacto, productos suministrados y precios negociados. Facilitaría una mejor coordinación de compras y actualización automática de inventarios al recibir nuevas existencias.
2. **Registro de Clientes y Gestión de Ventas Personalizada:** Integrar un sistema para registrar información básica de clientes, como nombre, dirección y preferencias de compra. Esto podría personalizar la experiencia de venta, permitiendo ofrecer recomendaciones de productos basadas en el historial de compras y facilitando la gestión de devoluciones o cambios.
3. **Seguimiento de Pedidos y Estado de Envíos:** Implementar funcionalidades para rastrear pedidos desde la recepción hasta la entrega final al cliente. Esto incluiría actualizar automáticamente el estado del pedido en el sistema y notificar al cliente sobre el progreso de su entrega.
4. **Generación de Informes de Ventas y Análisis de Inventarios:** Desarrollar herramientas para generar informes periódicos sobre ventas realizadas, ingresos generados y análisis de inventarios. Estos informes podrían incluir gráficos y estadísticas que ayuden a identificar productos más vendidos, estacionalidades y niveles de stock críticos.

Bibliografías

Drucioc, D. y García, A. (2021). Evolución de la automatización y sus consecuencias en el mercado laboral. Universidad de La Laguna. https://riull.ull.es/xmlui/bitstream/handle/915/25682/Evolucion%20de%20la%20automatizacion%20y%20sus%20consecuencias%20en%20el%20mercado%20laboral..pdf?sequence=1&isAllowed=y

Lima, E. (2020). Importancia del control de inventario en las empresas comerciales. Repositorio Universidad Estatal Península de Santa Elena. https://repositorio.upse.edu.ec/bitstream/46000/5512/1/UPSE-TCA-2020-0049.pdf

PHC Software Peru (2021). La digitalización es la única alternativa para las Mypes. https://phcsoftware.pe/business-at-speed/la-digitalizacion-es-la-unica-alternativa-para-las-mypes/

PlantUML (2024). https://plantuml.com/es/

Anexos

Anexo A. Código Fuente de la Aplicación

En la tabla A1 se presenta el código fuente (JAVA) de cada una de las clases de la aplicación.

Tabla A1

Código fuente de las clases Main, Producto, Inventario, Venta y HistorialVentas

Clase: Main

```java
package sistemaInventario;
import java.util.Scanner;

// Clase principal del programa
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Creación de un objeto Inventario con capacidad para 20 productos
        Inventario inventario = new Inventario(20);

        // Creación de un objeto Venta con capacidad para 10 productos en la venta actual y 20 en el historial de ventas
        Venta venta = new Venta(10, 20);

        int opcion = 0;
        // Bucle principal del menú de opciones
        do {
            System.out.println("\n------ Menú de Opciones ------");
            System.out.println("1. Agregar producto al inventario");
            System.out.println("2. Realizar una venta");
            System.out.println("3. Ver inventario");
            System.out.println("4. Ver historial de ventas");
            System.out.println("5. Salir");
            System.out.print("Ingrese la opción deseada: ");
            if (scanner.hasNextInt()) {
                opcion = scanner.nextInt();
                scanner.nextLine(); // Consume la nueva línea

                switch (opcion) {
```

```
                        case 1:
                            agregarProductoInventario(scanner,
inventario);
                            break;
                        case 2:
                            realizarVenta(scanner, inventario,
venta);
                            break;
                        case 3:
                            inventario.imprimirInventario();
                            break;
                        case 4:

venta.imprimirHistorialVentas(inventario);
                            break;
                        case 5:
                            System.out.println("Saliendo del
programa...");
                            break;
                        default:
                            System.out.println("Opción inválida.
Por favor ingrese una opción válida.");
                            break;
                    }
                } else {
                    System.out.println("Entrada no válida. Por
favor ingrese un número.");
                    scanner.next();
                }
            } while (opcion != 5);

            scanner.close(); // Cierra el objeto Scanner
        }

        // Método para agregar un producto al inventario
        private static void agregarProductoInventario(Scanner
scanner, Inventario inventario) {
            try {
                int id = -1;
                // Pide y valida el ID del producto
                while (id < 1) {
                    System.out.print("\nIngrese el ID del
producto: ");
                    if (scanner.hasNextInt()) {
                        id = scanner.nextInt();
                        if (id < 1) {
                            System.out.println("El ID debe ser
mayor o igual a 1.");
                        }
                    } else {
                        System.out.println("Entrada no válida.
Por favor ingrese un número.");
                        scanner.next();
```

```
                    }
                }
                scanner.nextLine();

                // Pide y valida el nombre del producto
                String nombre = "";
                while (!nombre.matches("[a-zA-Z\\s]+")) {
                    System.out.print("Ingrese el nombre del
producto: ");
                    nombre = scanner.nextLine();
                    if (!nombre.matches("[a-zA-Z\\s]+")) {
                        System.out.println("El nombre no puede
contener números.");
                    }
                }

                // Pide y valida el precio del producto
                double precio = -1;
                while (precio <= 0) {
                    System.out.print("Ingrese el precio del
producto: ");
                    if (scanner.hasNextDouble()) {
                        precio = scanner.nextDouble();
                        if (precio <= 0) {
                            System.out.println("El precio debe
ser mayor a 0.");
                        }
                    } else {
                        System.out.println("Entrada no válida.
Por favor ingrese un número.");
                        scanner.next();
                    }
                }

                // Pide y valida la cantidad inicial del
producto
                int cantidad = -1;
                while (cantidad < 1) {
                    System.out.print("Ingrese la cantidad
inicial en stock: ");
                    if (scanner.hasNextInt()) {
                        cantidad = scanner.nextInt();
                        if (cantidad < 1) {
                            System.out.println("La cantidad
inicial en stock debe ser mayor o igual a 1.");
                        }
                    } else {
                        System.out.println("Entrada no válida.
Por favor ingrese un número.");
                        scanner.next();
                    }
                }
```

```
                // Crea un objeto Producto y lo agrega al
inventario
                Producto producto = new Producto(id, nombre,
precio, cantidad);
                inventario.agregarProducto(producto);
            } catch (Exception e) {
                System.out.println("Error al agregar producto: "
+ e.getMessage());
                scanner.nextLine();
            }
        }

        // Método para realizar una venta
        private static void realizarVenta(Scanner scanner,
Inventario inventario, Venta venta) {
            try {
                // Verifica si el inventario está vacío
                if (inventario.estaVacio()) {
                    System.out.println("No hay productos en el
inventario para realizar una venta.");
                    return;
                }

                int id;
                Producto producto;
                // Bucle para procesar la venta de productos
                do {
                    System.out.print("\nIngrese el ID del
producto a vender (0 para finalizar la venta): ");
                    if (scanner.hasNextInt()) {
                        id = scanner.nextInt();
                        if (id != 0) {
                            producto =
inventario.buscarProducto(id);
                            if (producto != null) {
                                System.out.print("Ingrese la
cantidad a vender: ");
                                if (scanner.hasNextInt()) {
                                    int cantidad =
scanner.nextInt();

venta.agregarProducto(producto, cantidad);
                                } else {
                                    System.out.println("Entrada
no válida. Por favor ingrese un número.");
                                    scanner.next();
                                }
                            } else {
                                System.out.println("Producto no
encontrado en el inventario.");
                            }
                        }
                    } else {
```

```
                        System.out.println("Entrada no válida.
Por favor ingrese un número.");
                        scanner.next();
                        id = -1; // Mantiene el bucle si la
entrada no es válida
                    }
                } while (id != 0);

                venta.imprimirRecibo(); // Imprime el recibo de
la venta
            } catch (Exception e) {
                System.out.println("Error al realizar venta: " +
e.getMessage());
                scanner.nextLine();
            }
        }
}
```

Producto

```
package sistemaInventario;
// Clase Producto que representa un producto en el
inventario
public class Producto {
    private int id;
    private String nombre;
    private double precio;
    private int cantidad;

    // Constructor de la clase Producto
    public Producto(int id, String nombre, double precio,
int cantidad) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
        this.cantidad = cantidad;
    }

    // Métodos getter y setter para los atributos de la
clase Producto
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

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

```
    public double getPrecio() {
        return precio;
    }

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

    public int getCantidad() {
        return cantidad;
    }

    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }

    // Método para actualizar la cantidad de productos
después de una venta
    public void vender(int cantidadVendida) {
        if (cantidadVendida <= cantidad) {
            cantidad -= cantidadVendida;
        }
    }
}
```

Inventario

```
package sistemaInventario;
// Clase Inventario que representa el inventario de
productos
public class Inventario {
    private Producto[] productos; // Arreglo para almacenar
los productos
    private int cantidadProductos; // Número actual de
productos en el inventario

    // Constructor de la clase Inventario
    public Inventario(int capacidad) {
        productos = new Producto[capacidad];
        cantidadProductos = 0;
    }

    // Método para agregar un nuevo producto al inventario
    public void agregarProducto(Producto producto) {
        // Verifica si el producto ya existe en el
inventario
        if (buscarProducto(producto.getId(), false) != null)
{
            System.out.println("Error: Ya existe un producto
con el ID " + producto.getId() + " en el inventario.");
            return;
        }
```

```
        // Agrega el producto al inventario si hay espacio
disponible
        if (cantidadProductos < productos.length) {
            productos[cantidadProductos++] = producto;
            System.out.println("Producto agregado al
inventario correctamente.");
        } else {
            System.out.println("Error: No se pueden agregar
más productos al inventario.");
        }
    }

    // Método para buscar un producto por su ID
    public Producto buscarProducto(int id) {
        return buscarProducto(id, true);
    }

    // Método privado para buscar un producto por su ID con
control de impresión de errores
    private Producto buscarProducto(int id, boolean
imprimirError) {
        for (int i = 0; i < cantidadProductos; i++) {
            if (productos[i].getId() == id) {
                return productos[i];
            }
        }
        if (imprimirError) {
            System.out.println("Error: Producto no
encontrado con ID " + id);
        }
        return null;
    }

    // Método para imprimir el inventario actual
    public void imprimirInventario() {
        if (cantidadProductos == 0) {
            System.out.println("No hay productos en el
inventario");
            return;
        }

        System.out.println("\nInventario Actual:");
        for (int i = 0; i < cantidadProductos; i++) {
            Producto producto = productos[i];
            System.out.println("ID: " + producto.getId() +
", Nombre: " + producto.getNombre() +
                    ", Precio: " + producto.getPrecio() + ",
Cantidad en stock: " + producto.getCantidad());
        }
    }

    // Método para verificar si el inventario está vacío
    public boolean estaVacio() {
```

```
        return cantidadProductos == 0;
    }
}
```

Venta

```
package sistemaInventario;
// Clase Venta que representa una venta de productos
public class Venta {
    private Producto[] productosVendidos; // Arreglo para
almacenar los productos vendidos
    private int[] cantidades; // Arreglo para almacenar las
cantidades vendidas de cada producto
    private int cantidadProductos; // Número actual de
productos en la venta
    private HistorialVentas historial; // Objeto para
manejar el historial de ventas

    // Constructor de la clase Venta
    public Venta(int capacidad, int capacidadHistorial) {
        productosVendidos = new Producto[capacidad];
        cantidades = new int[capacidad];
        cantidadProductos = 0;
        historial = new HistorialVentas(capacidadHistorial);
    }

    // Método para agregar un producto a la venta
    public void agregarProducto(Producto producto, int
cantidad) {
        try {
            // Verifica si hay suficientes unidades en el
inventario
            if (producto.getCantidad() < cantidad) {
                throw new IllegalArgumentException("No hay
suficientes unidades en el inventario.");
            }

            boolean productoExistente = false;
            // Verifica si el producto ya está en la venta
actual
            for (int i = 0; i < cantidadProductos; i++) {
                if (productosVendidos[i].getId() ==
producto.getId()) {
                    cantidades[i] += cantidad;
                    productoExistente = true;
                }
            }

            // Si el producto no está en la venta, lo agrega
            if (!productoExistente) {
                if (cantidadProductos >=
productosVendidos.length) {
                    throw new IllegalStateException("No se
pueden agregar más productos a la venta.");
                }
```

```
                    productosVendidos[cantidadProductos] =
producto;
                    cantidades[cantidadProductos] = cantidad;
                    cantidadProductos++;
                }

                // Actualiza el inventario del producto
                producto.vender(cantidad);
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            } catch (IllegalStateException e) {
                System.out.println("Error: " + e.getMessage());
            } catch (Exception e) {
                System.out.println("Error inesperado: " +
e.getMessage());
            }
        }

        // Método para imprimir el recibo de la venta
        public void imprimirRecibo() {
            if (cantidadProductos == 0) {
                System.out.println("No se realizó ninguna
venta.");
                return;
            }

            double total = 0;
            System.out.println("\nRecibo de Venta:");
            // Itera sobre los productos vendidos y calcula el
subtotal y total
            for (int i = 0; i < cantidadProductos; i++) {
                double subtotal =
productosVendidos[i].getPrecio() * cantidades[i];
                total += subtotal;
                System.out.println("Producto: " +
productosVendidos[i].getNombre() + ", Cantidad: " +
cantidades[i] + ", " +
                        "Subtotal: " + subtotal);
            }
            System.out.println("Total: S/" + total);

            // Agrega el recibo al historial
            for (int i = 0; i < cantidadProductos; i++) {

historial.agregarRecibo(productosVendidos[i].getId(),
productosVendidos[i].getNombre(),
                        productosVendidos[i].getPrecio() *
cantidades[i]);
            }

            cantidadProductos = 0; // Reinicia la venta actual
        }
```

```
    // Método para imprimir el historial de ventas
    public void imprimirHistorialVentas(Inventario
inventario) {
        try {
            historial.imprimirHistorial(productosVendidos,
inventario);
        } catch (Exception e) {
            System.out.println("Error al imprimir el
historial de ventas: " + e.getMessage());
        }
    }
}
```

HistorialVentas

```
package sistemaInventario;
// Clase HistorialVentas que representa el historial de
ventas
public class HistorialVentas {
    private double[][] historial; // Arreglo para almacenar
el historial de ventas (ID, subtotal, número de venta)
    private int numVentas; // Número actual de ventas
registradas
    private double total; // Total acumulado de ventas

    // Constructor de la clase HistorialVentas
    public HistorialVentas(int capacidad) {
        historial = new double[capacidad][3];
        numVentas = 0;
        total = 0;
    }

    // Método para agregar un recibo al historial
    public void agregarRecibo(double idProducto, String
nombreProducto, double subtotal) {
        try {
            if (numVentas >= historial.length) {
                throw new IllegalStateException("No se
pueden registrar más ventas.");
            }

            historial[numVentas][0] = idProducto; //
Almacena el ID del producto
            historial[numVentas][1] = subtotal; // Almacena
el subtotal
            historial[numVentas][2] = numVentas + 1; //
Almacena el número de venta
            total += subtotal; // Actualiza el total
acumulado de ventas
            numVentas++;
        } catch (IllegalStateException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error inesperado: " +
e.getMessage());
```

```
            }
        }

        // Método para imprimir el historial de ventas
        public void imprimirHistorial(Producto[]
productosVendidos, Inventario inventario) {
            try {
                if (numVentas == 0) {
                    System.out.println("No hay ventas
registradas en el historial.");
                    return;
                }

                System.out.println("\nHistorial de Ventas:");
                // Itera sobre el historial y muestra los
detalles de cada venta
                for (int i = 0; i < numVentas; i++) {
                    int idProducto = (int) historial[i][0];
                    double subtotal = historial[i][1];
                    int numeroVenta = (int) historial[i][2]; //
Obtiene el número de venta

                    Producto producto =
inventario.buscarProducto(idProducto);
                    if (producto == null) {
                        throw new
IllegalArgumentException("Nombre del producto no encontrado
para el ID: " + idProducto);
                    }

                    String nombreProducto =
producto.getNombre();

                    System.out.println("Recibo N°" + numeroVenta
+ ": Producto ID = " + idProducto +
                            ", Nombre = " + nombreProducto + ",
Subtotal = S/" + subtotal);
                }
                System.out.println("Total acumulado de ventas:
S/" + total);
            } catch (IllegalArgumentException e) {
                System.out.println("Error: " + e.getMessage());
            } catch (Exception e) {
                System.out.println("Error inesperado: " +
e.getMessage());
            }
        }
}
```