**1. Clases del Sistema**

* **ClaseProducto**: Representa un producto en el sistema. Cada producto tiene un nombre, un precio, y una cantidad en stock (inventario disponible).
  + **Atributos**:
    - nombre (Cadena): Nombre del producto.
    - precio(doble): Precio del producto.
    - stock(int): Cantidad de productos disponibles en inventario.
  + **Métodos**:
    - Producto (String nombre, double precio, int stock): Constructor que inicializa los atributos con los valores pasados ​​como parámetros.
    - mostrarProducto (): Imprime en consola los detalles del producto (nombre, precio y stock).
* **ClaseCliente**: Representa un cliente del sistema, almacenando su nombre y su cedula(identificación).
  + **Atributos**:
    - nombre (Cadena): Nombre del cliente.
    - cedula (Cadena): Identificación única del cliente.
  + **Métodos**:
    - Cliente (String nombre, String cedula): Constructor para inicializar los atributos.
    - mostrarCliente (): Muestra en consola la información del cliente (nombre y cedula).
* **ClaseVenta**: Representa una transacción de venta, que incluye al cliente que compra, el producto vendido, la cantidad de productos comprados y el total de la venta (calculado como precio \* cantidad).
  + **Atributos**:
    - cliente (Cliente): Cliente que realiza la compra.
    - producto (Producto): Producto que está comprando el cliente.
    - cantidad(int): Cantidad de productos comprados.
    - total(doble): Total a pagar, calculado como precio del producto \* cantidad.
  + **Métodos**:
    - Venta (Cliente.cliente, Producto.producto, int cantidad): Constructor que inicializa la venta con un cliente, un producto y la cantidad deseada, calculando automáticamente el total.
    - mostrarVenta (): Muestra los detalles de la venta, incluyendo el cliente, producto, cantidad y el total.

**2. Clase Principal (SistemaVentas)**

La clase SistemaVentas es el núcleo del sistema y contiene:

* Variables ArrayList para almacenar productos ( productos), clientes ( clientes) y ventas ( ventas).
* Uno Scanner para la entrada de datos.
* Un método main que inicia el sistema y muestra el menú de opciones.

**3. Método main**

El método main se ejecuta al iniciar el sistema y muestra un menú interactivo para que el usuario elija qué acción realizar. Utilice un bucle while que sigue mostrando el menú hasta que el usuario elija la opción de salida.

**Opciones del Menú**:

1. **Ingresar Producto**: Agrega un nuevo producto al sistema.
2. **Listar: Productos** Muestra todos los productos en el inventario.
3. **Ingresar Cliente**: Agrega un nuevo cliente al sistema.
4. **Listar Clientes**: Muestra todos los clientes registrados.
5. **Realizar Venta**: Permite registrar una venta.
6. **Mostrar Ventas Realizadas**: Muestra todas las ventas que se han hecho.
7. **Salir**: Cerrar el sistema.

**Bucle de Menú**:

* Cada opción del menú ejecuta un método específico. Utilizamos un switch para seleccionar la acción correspondiente según la elección del usuario.

**4. Métodos para Gestionar el Sistema**

Cada método en SistemaVentas realiza una tarea específica para mantener el sistema organizado y funcional.

* **Método ingresarProducto ()**:
  + Solicita al usuario el nombre, precio y stock el producto.
  + Crea un nuevo objeto Producto con estos valores.
  + Añade el objeto a la lista productos.
  + Muestra un mensaje confirmando que el producto fue agregado correctamente.
* **Método listarProductos ()**:
  + Recorre la lista de productos y llama a mostrarProducto () cada uno, mostrando su nombre y precio en stock la consola.
* **Método ingresarCliente ()**:
  + Solicita el nombre y cedula del cliente.
  + Crea un nuevo objeto Cliente y lo añade a la lista clientes.
  + Imprima un mensaje confirmando la incorporación del cliente.
* **Método listarClientes ()**:
  + Registre la lista de clientes y llame a mostrarCliente () para mostrar el nombre y cedulade cada cliente en consola.
* **Método realizarVenta ()**:
  + Pide el nombre del cliente y busca el objeto Cliente en la lista clientes.
  + Si no encuentra al cliente, muestra un mensaje de error y termina.
  + Pide el nombreproducto y busca el objeto Producto en la lista productos.
  + Si no encuentra el producto, muestra un mensaje de error y termina.
  + Pide la cantidad a comprar.
  + Verifica que haya suficiente stock. Si no hay suficiente, muestra un mensaje indicando la falta de stock.
  + Si hay suficiente stock:
    - Resta la cantidad vendida del stock producto.
    - Crea un objeto Venta con los datos del cliente, producto y cantidad.
    - Añade la venta a la lista ventas.
    - Muestra un mensaje confirmando que la venta se realizó exitosamente.
* **Método mostrarVentas ()**:
  + Recorre la lista ventas y muestra cada venta realizada, incluyendo el cliente, producto, cantidad y total.

**Ejecución del sistema**

* El sistema permite agregar y gestionar productos y clientes.
* Al realizar una venta, el sistema actualiza el stock del producto.
* Cada venta incluye un cálculo del total basado en la cantidad y precio del producto.
* Finalmente, todas las ventas se pueden listar, proporcionando una vista completa de las transacciones realizadas.

Este sistema está estructurado de tal manera que utiliza **bucles while y for para la navegación y las listas**. Las **condicionales if y if-else** ayudan a verificar y validar datos en cada operación.

**Ejemplo de Ejecución**

Texto

Descripción generada automáticamente