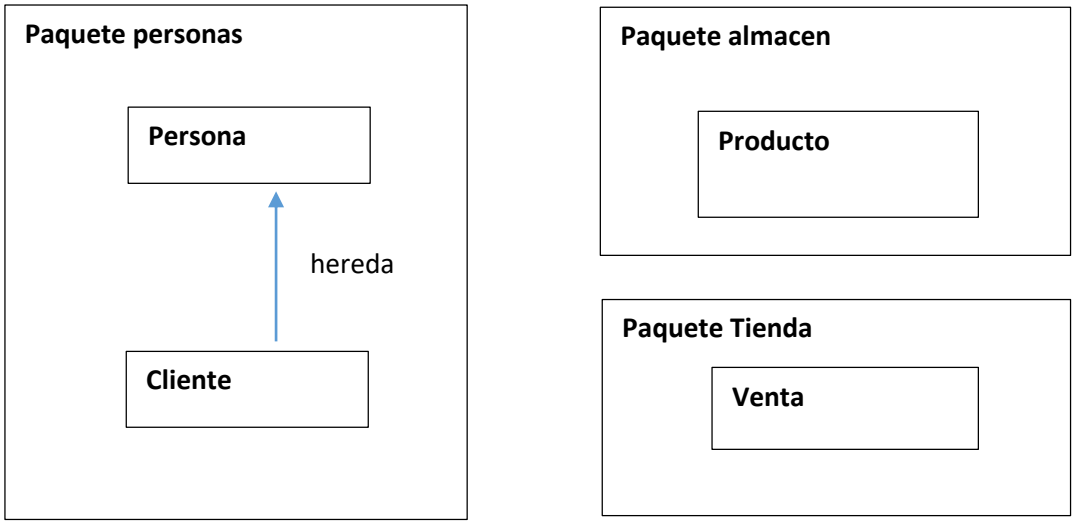
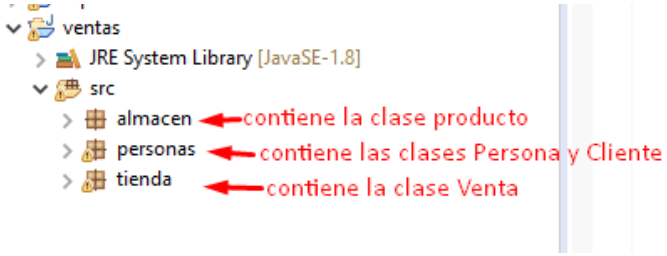
En esta unidad has aprendido nuevos conceptos de POO:

Construcción de objetos con constructores sobrecargados, paquetes, modificadores de acceso de clases, herencia, métodos y atributos estáticos, etc.

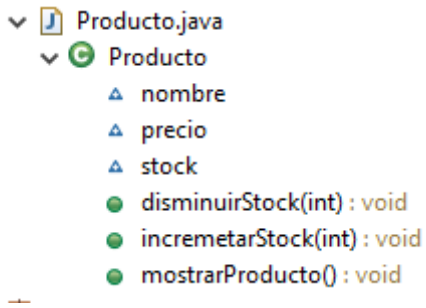
La tarea consiste en implementar el siguiente sistema de Ventas:



Paso 1: Crea los tres paquetes con sus respectivas clases



Paso 2: Implementa la clase Producto



Atributos: (deben ser privados)

- ✓ **nombre:** Un texto con el nombre del producto
- ✓ **precio:** un número decimal con el precio del producto
- ✓ **stock:** un número entero con la cantidad disponible para vender

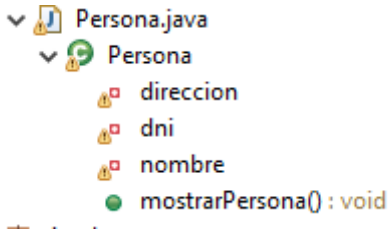
Constructores:

- ✓ **Producto(nombre):** Asigna el nombre al objeto producto. El precio y el stock valen 0.
- ✓ **Producto(nombre, precio):** Asigna el nombre y precio al objeto. El stock vale 0.
- ✓ **Producto(nombre, precio, stock):** Asigna el nombre, precio y stock al objeto.
- ✓ **Producto(Producto p1):** Copia los datos del objeto p1.

Métodos:

- ✓ **incrementarStock(int cantidad):** Simula la llegada de nuevo stock; debes aumentar el stock del producto.
- ✓ **disminuirStock(int cantidad):** Simula la venta del producto; debes disminuir el stock del producto.
- ✓ **Métodos getter y setter** (para poder modificar los atributos)
- ✓ **MostrarProducto() :** Muestra el nombre, el precio y el stock del producto.

Paso 3: Implementa la clase Persona



Atributos: (deben ser privados)

- ✓ **direccion:** Un texto con la dirección de la persona
- ✓ **dni:** Un texto con el dni de la persona
- ✓ **nombre:** Un texto con el nombre de la persona

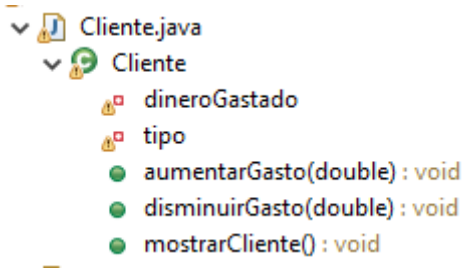
Constructores:

- ✓ **Persona(dni):** Asigna el dni al objeto Persona. La dirección y el nombre de la persona contienen un texto vacío.
- ✓ **Persona(dni, nombre):** Asigna el dni y el nombre al objeto. La dirección contiene un texto vacío.
- ✓ **Persona(dni, nombre, direccion):** Asigna el nombre, dni y dirección al objeto.
- ✓ **Persona(Persona per1):** Copia los datos del objeto per1.

Métodos:

- ✓ **Métodos getter y setter** (para poder modificar los atributos)
- ✓ **MostrarPersona ():** Muestra el dni, nombre y dirección de la Persona.

Paso 4: Implementa la clase Cliente



Atributos: (deben ser privados)

Ten en cuenta que se deben heredar todos los atributos y métodos de la clase padre Persona.

- ✓ **dineroGastado:** Un decimal con el dinero gastado por el cliente al realizar las compras
- ✓ **tipo:** Un texto que puede contener solamente dos opciones; **normal** (un cliente que no ha comprado mucho y no obtiene descuentos) y **vip** (un buen cliente que compra bastante y obtiene descuentos)

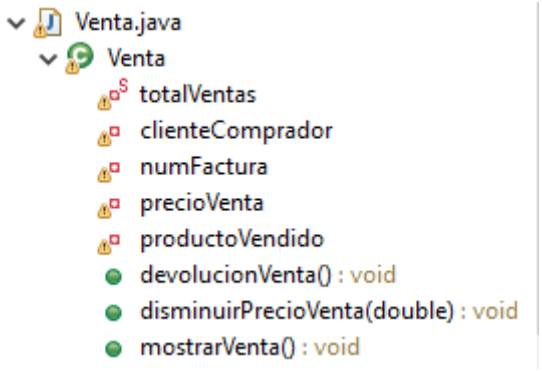
Constructores:

- ✓ **Cliente(dni):** Asigna el dni al objeto Cliente. La dirección y el nombre del cliente contienen un texto vacío. El dinero gastado es cero y el tipo es "normal".
- ✓ **Cliente(dni, nombre):** Asigna el dni y el nombre al objeto. La dirección contiene un texto vacío. El dinero gastado es cero y el tipo es "normal".
- ✓ **Cliente(dni, nombre, direccion):** Asigna el nombre, dni y dirección al objeto. El dinero gastado es cero y el tipo es "normal".
- ✓ **Cliente(dni, nombre, dirección, tipo):** Asigna el nombre, dni, dirección y tipo al objeto. El dinero gastado es cero.
- ✓ **Cliente(dni, nombre, dirección, tipo, dinerogastado):** Asigna todos los atributos al objeto cliente.
- ✓ **Cliente(Cliente c1):** Copia los datos del objeto cliente c1.

Métodos:

- ✓ **Métodos getter y setter** (para poder modificar los atributos)
- ✓ **aumentarGasto(double cantidad):** Simula la compra de un producto; debes aumentar el dinero gastado.
- ✓ **disminuirGasto(double cantidad):** Simula la devolución de un producto; debes disminuir el dinero gastado.
- ✓ **MostrarCliente() :** Muestra el dni, nombre, dirección y tipo del cliente.

Paso 5: Implementa la clase Venta



Atributos: (deben ser privados)

- ✓ **totalVentas:** Un decimal con el dinero de todas las ventas realizadas (**debe ser una variable estática**)
- ✓ **clienteComprador:** Un objeto de la clase Cliente
- ✓ **productoVendido:** Un objeto de la clase Producto
- ✓ **numFactura:** Un texto con el número de la factura
- ✓ **precioVenta:** Un decimal con el precio logrado con la venta

Constructores:

- ✓ **Venta(numfactura, cliente, producto, precioVenta):** Realiza la venta del producto al cliente. Debes tener en cuenta varias cosas:
  - **Si el producto tiene stock cero (no hay stock)**, debes mostrar un mensaje diciendo que no hay stock disponible. El precio de venta será cero y total ventas no se incrementa.
  - **Si hay stock disponible** pueden ocurrir dos cosas:
    - **Si el cliente es VIP se le hace un descuento de 15% sobre el precioVenta. Éste será el nuevo precio de venta. Debes avisar con un mensaje "has logrado un descuento del 15% por ser cliente VIP". También debes incrementar el total Ventas** (con el nuevo precio Venta) **y el dinero gastado por el cliente** (con el nuevo precio Venta). **El stock del producto debe disminuir en uno.**
    - **Si el cliente es normal debes incrementar el total Ventas y el dinero gastado por el cliente. El stock del producto debe disminuir en uno. Si el cliente llevara gastado 100 euros pasa a ser tipo VIP. Avisa con un mensaje de texto "enhorabuena, ya eres cliente VIP, obtén un descuento del 15% en tu próxima compra."** (cambia su tipo por VIP).

Nota:
Puedes utilizar órdenes como:
this.clienteComprador.aumentarGasto(5);
this.productoVendido.disminuirStock(1);

Métodos:

- ✓ **Métodos getter y setter** (para poder modificar los atributos)
- ✓ **disminuirPrecioVenta(double cantidad):** Simula el descuento sobre el precio de venta de un producto; debes disminuir el dinero gastado por el cliente, el precio de venta y el total ventas.
- ✓ **DevoluciónVenta():** Simula la devolución del producto. Debes disminuir el dinero gastado por el cliente y el total ventas. El precio Venta pasa a ser cero. El stock del producto se incrementa en uno.
- ✓ **MostrarVenta() :** Muestra los datos del cliente, el número de factura, el producto, y el precio de venta.

Paso 5: Prueba del proyecto

Realiza una simulación con 2 clientes, 2 productos y 4 ventas; simula distintos escenarios:

- ✓ Cliente normal
- ✓ Cliente vip
- ✓ Producto con stock
- ✓ Producto sin stock.

Muestra las distintas ventas, el dinero total gastado por cada cliente y el total de dinero logrado por las ventas.

Calificación por la correcta implementación de las clases:

- ✓ **Clase Producto: 1 punto.**
- ✓ **Clase Persona: 1 punto.**
- ✓ **Clase Cliente: 2 puntos.**
- ✓ **Clase Venta: 4 puntos.**
- ✓ **Clase con el programa principal de prueba: 2 puntos.**

Entrega un fichero comprimido con todo el código del proyecto y dale un nombre similar a éste: sanchez_manas_begona_tarea5.1 (si la alumna se llama Begoña Sanchez Mañas)