Actividad

Complete los espacios, es ** que se encuentran en el código, de modo que se imprima el siguiente resultado en pantalla y los atributos y métodos estén lo más encapsulados posible (cumpliendo con el principio de encapsulamiento). **Nota:** no puede modificar otras secciones, ni agregar o eliminar código.

Salida esperada:

2
2
3
Orden 101 creada
5
128 va a retirar producto
Pantalon retirado
4
4

Paquete compras clase Producto

```java
package compras;

public class Producto {

    private final int codigo; // Para que el código sea único y no pueda ser modificado una vez asignado.
    private String nombre; // Ambos atributos son privados porque la clase Producto controla cómo se accede a ellos.
    public String tipo;   // Esto evita que se cambien directamente desde fuera de la clase y permite encapsular el comportamiento de la clase.
    static int totalProductosPedidos;

    public Producto(int codigo, String nombre, String tipo) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.tipo = tipo;
    }

    public void imprimirNombre() {
        System.out.print(nombre);
    }

    public void setCodigo(int codigo) {

    }

    public int getCodigo() { //El tipo de retorno es int, ya que el código es un número entero.
        return codigo;
    }

    public static int getTotalProductosPedidos() { // El método es público para que otros objetos
                                                   //o clases puedan acceder al total de productos pedidos sin necesidad de instanciar un objeto Producto.
        return totalProductosPedidos;
    }
}
```

Paquete compras clase OrdenCompra

```java
package compras;

import gestionHumana.Empleado; //// Se importa la clase Empleado desde el paquete gestionHumana,
//porque necesitamos acceder a los datos del empleado en la orden de compra.
import java.util.ArrayList;

public class OrdenCompra {

    public int codigo;
    private String tipo;
    private Empleado comprador;
    private ArrayList<Producto> productos;

    public OrdenCompra(int codigo, String tipo, Empleado comprador,
            ArrayList<Producto> productos) {
        this.codigo = codigo;
        this.tipo = tipo;
        this.comprador = comprador;
        this.productos = productos;
        Producto.totalProductosPedidos += productos.size();
    }

    public void agregarProducto(Producto producto) {
        if (producto.tipo.equals(tipo)) {
            productos.add(producto);
            Producto.totalProductosPedidos++;
        }
    }

    public void retirarProducto(Empleado empleado, Producto producto) { // Se pone porque el método no devuelve ningún valor.
        if (!empleado.tengoPermiso()) {
            return; // Si no tiene permiso, termina el método sin hacer nada.
        }
        retirarProducto(producto);

    }

    private void retirarProducto(Producto producto) {
        for (int i = 0; i < productos.size(); i++) {
            if (producto.getCodigo() == productos.get(i).getCodigo()) {
                productos.remove(i);
                Producto.totalProductosPedidos--;
                producto.imprimirNombre();
                System.out.println(" retirado");
                break;
            }
        }
    }

    public void descontar() {
        Producto.totalProductosPedidos -= productos.size(); //Se utiliza productos.size() porque el ciclo necesita saber
        //cuántos productos hay en la lista, para recorrerlos correctamente y buscar el producto a retirar.
    }

}
```

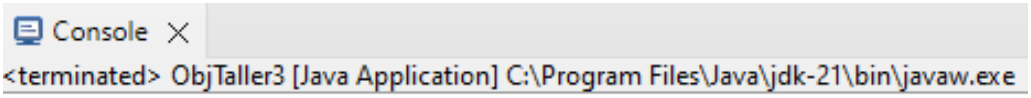
Paquete gestionHumana clase Empleado

```java
package gestionHumana;

public class Empleado {

    public final long cedula; // Atributo público y final.
    //Se usa público porque la cédula no cambiará después de ser asignada, y final indica que es inmutable.
    //Esto asegura que el valor de la cédula no se pueda modificar una vez asignado en el constructor.
    private String nombre; // Se usa `private` para proteger el valor de nombre y no permitir modificaciones directas desde otras clases.
    private String cargo; // Se usa `private` para proteger el valor de nombre y no permitir modificaciones directas desde otras clases.

    public Empleado(long cedula, String nombre, String cargo) {
        this.cedula = cedula;
        this.nombre = nombre;
        this.cargo = cargo;
    }

    public boolean tengoPermiso() { // // Método público que devuelve un booleano indicando si el empleado tiene permiso (es administrador).
        return cargo.contains("Administrador");
    }
}
```

Clase principal Paquete objtaller 3 clase ObjTaller

```java
package objtaller3;

import compras.*; // Importa todas las clases del paquete compras.
import gestionHumana.Empleado;
import java.util.ArrayList; // Importa la clase ArrayList.

public class ObjTaller3 {

    public static void main(String [] args) { // Se completa el tipo de parámetro
        Producto p1 = new Producto(1, "Escoba", "Aseo");
        Producto p2 = new Producto(2, "Camisa", "Ropa");
        Producto p3 = new Producto(3, "Trapera", "Aseo");
        Producto p4 = new Producto(4, "Pantalon", "Ropa");
        Producto p5 = new Producto(5, "Jabon", "Aseo");
        Empleado emp1 = new Empleado(405, "Juan", "Ingeniero");
        ArrayList<Producto> productos1 = new ArrayList<>();
        productos1.add(p1);
        productos1.add(p3);
        OrdenCompra orden1 = new OrdenCompra(101, "Aseo", emp1, productos1); // 101: Este es el código de la primera orden.
        System.out.println(Producto.getTotalProductosPedidos());
        orden1.agregarProducto(p4); // agregarProducto: Este es el método que se debe usar para agregar productos a la orden.
        System.out.println(Producto.getTotalProductosPedidos());
        orden1.agregarProducto(p5);
        System.out.println(Producto.getTotalProductosPedidos());
        System.out.println("Orden " + orden1.codigo + " creada");

        Empleado emp2 = new Empleado(128,"Susana", "Administradora de sucursal");
        ArrayList<Producto> productos2 = new ArrayList<>();
        productos2.add(p2);
        productos2.add(p4);
        OrdenCompra orden2 = new OrdenCompra(202, "Ropa", emp2, productos2);
        System.out.println(Producto.getTotalProductosPedidos());
        System.out.println(emp2.cedula + " va a retirar producto");
        orden2.retirarProducto(emp2, p4);
        System.out.println(Producto.getTotalProductosPedidos());
        orden2.retirarProducto(emp1, p2); // Aquí emp1 retira el producto p2.
        System.out.println(Producto.getTotalProductosPedidos());
    }
}
```

Salida del código después de los cambios




```
2
2
3
Orden 101 creada
5
128 va a retirar producto
Pantalon retirado
4
4
```