# Icon Description automatically generatedTaller tres. Métodos, constructores y this.

## Programación orientada objetos – 2021-1S

### Simón Cuartas Rendón – C.C. 1.037.670.103

Estudiante de estadística

# Parte uno.

La idea de este ejercicio es modificar los códigos dados de tal modo que se tenga la mayor privacidad posible. A continuación se adjunta el código obtenido y al final se observa la figura uno que evidencia el resultado esperado por consola.

## Para la clase **ObjTaller3.**

package objtaller3;  
  
import compras.OrdenCompra;  
import gestionHumana.Empleado;  
import compras.Producto;  
import java.util.ArrayList;  
  
public class ObjTaller3 {  
 public static void main(String[] args) {  
 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);  
 System.*out*.println(Producto.getTotalProductosPedidos());  
 orden1.agregarProducto(p4);  
 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); // Se usa emp1 para no alterar el resultado  
 System.*out*.println(Producto.getTotalProductosPedidos());  
 }  
}

## Para la clase **Empleado.**

package gestionHumana;  
  
public class Empleado {  
 public final long cedula;  
 private String nombre; // Se está usando con el constructor en ObjTaller3.  
 private String cargo;  
  
 public Empleado(long cedula, String nombre, String cargo) {  
 this.cedula = cedula;  
 this.nombre = nombre;  
 this.cargo = cargo;  
 }  
  
 public boolean tengoPermiso() {  
 return cargo.contains("Administrador");  
 }  
}

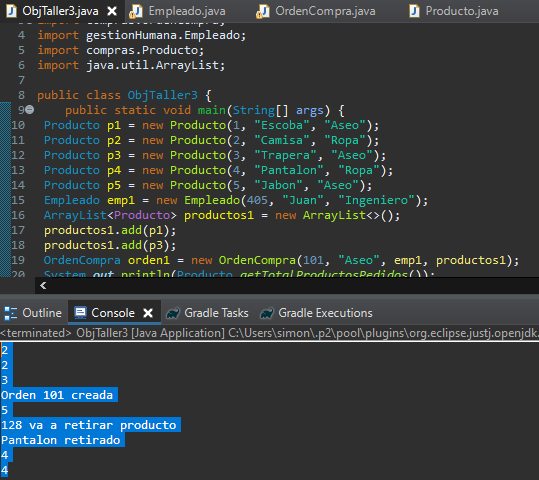
## Para la clase **OrdenCompra.**

package compras;  
  
import gestionHumana.Empleado; // Se define un atributo de tipo Empleado  
import java.util.ArrayList;  
  
public class OrdenCompra {  
 public int codigo; // Se usa con el  
 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();  
 }  
 // La siguiente clase de usa en ObjTalle3  
 public void agregarProducto(Producto producto) {  
 if (producto.tipo.equals(tipo)) {  
 productos.add(producto);  
 Producto.totalProductosPedidos++;  
 }  
 }  
 // Se usa en ObjTalle3 sobre el objeto orden2   
 public void retirarProducto(Empleado empleado, Producto producto) {  
 if (!empleado.tengoPermiso()) {  
 return;  
 }  
 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();  
 }  
}

## Para la clase **Producto.**

package compras;  
  
public class Producto {  
 private final int codigo;  
 private String nombre;  
 String tipo; // Es por defecto o de tipo package  
 static int *totalProductosPedidos*; // Es por defecto o de tipo package  
  
 // El siguiente constructor se usa en ObjTaller3  
 public Producto(int codigo, String nombre, String tipo) {  
 this.codigo = codigo;  
 this.nombre = nombre;  
 this.tipo = tipo;  
 }  
  
 void imprimirNombre() { // Tipo package  
 System.*out*.print(nombre);  
 }  
  
 //private void setCodigo(int codigo) { // No se usa en otra clase  
 //this.codigo = codigo; // Es un setter  
 //}  
  
 /\* El método anterior se borra porque no se puede  
 \* modificar el atributo código en tanto es una constante  
 \* y es definida en el construcutor de esta clase \*/  
  
 public int getCodigo() {  
 return codigo;  
 } // Se retorna un codigo que es de tipo entero (int)  
  
 static public int getTotalProductosPedidos() {  
 return *totalProductosPedidos*;  
 } // Se usa seis veces en ObjTaller3  
}

En la figura uno se observa el resultado en consola de compilar y correr este programa.



# Parte dos. **Git-Hub.**

A continuación, se responden las *preguntas para pensar* anexas al final de la **parte dos** del taller tres del curso.

1. **¿Por qué cree que es necesario definir el atributo tv a la clase Control?**

**La idea de esta clase es poder realizar el enlazamiento entre un televisor y un control, para lo cual es necesario definir este atributo de la clase TV para poder definir tal enlace.**

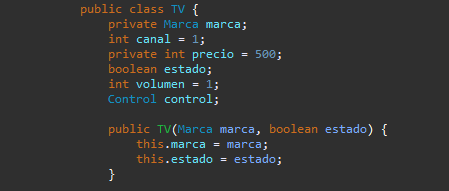
1. **¿Por qué cree que es necesario que la clase Control defina el atributo tv y la clase TV defina el atributo control? Ambas clases tendrían la referencia de los objetos, ¿esto qué facilidad le agrega a la implementación?**

**La idea de un atributo es definir características o información que van a tener las instancias que conforman una clase, por lo que con la definición de estos atributos en cada clase se logra hacer explícita estas características y se facilita el enlace que se quiere hacer entre dos objetos de ambas clases.**

1. **¿Por qué cree que es necesario usar la cláusula this en el constructor de TV?**

**El constructor de la clase TV está recibiendo dos parámetros que son atributos de tal clase, por lo que al usar la cláusula this se desambigua esto: cuando se usa el this.XXXX, lo que está a la derecha del punto hace referencia a un atributo del objeto de la clase TV y cuando aparece el XXXX solo se refiere al parámetro que recibe el constructor.**

1. **¿A quién hace referencia this en un método de referencia o instancia? ¿Cómo pasaste el valor del objeto tipo Control al televisor que pasó como parámetro en el método enlazar?**



**La cláusula this hace referencia a algún objeto de la clase en la que se está definiendo el método y tal objeto será aquel sobre el que se ejecute el método en el programa.**

1. **¿Por qué son útiles los constructores al momento de definir una clase y crear un objeto?**

**Los constructores van a facilitar la definición de los atributos particulares que tenga un objeto al momento de crearse, pues se puede realizar en una sola línea como parámetros del constructor, lo que simplifica el programa.**

1. **¿Qué utilidad tienen los métodos set y get? ¿Considera que este método puede ser de utilidad en la creación de restricciones o validaciones de una clase?**

**El método set es útil en el sentido que permite definir el valor que asumirá el atributo de algún objeto particular de una clase de manera directa, mientras que el get facilitará conocer su valor. Además, con el método set se pueden realizar restricciones o validaciones en su cuerpo; por ejemplo, con el ejercicio realizado sobre los televisores, se puede verificar primero si se va a asignar un valor adecuado al método, de modo que se evita que se asignen valores no deseados a los atributos de los diferentes objetos de una clase.**

1. **¿Qué ocurre con el constructor vacío de la clase TV? ¿Aún es posible crear una instancia de TV sin parámetros?**

**El constructor no es necesario en la medida que al definir un nuevo objeto de la clase TV se les asignarán a sus atributos los valores por defecto asignados a estos cuando fueron definidos, o en caso de no habérsele dado algún valor por omisión, tomarán los asignados por Java en estos casos (como un cero o el valor null). Sin embargo, a la hora de asignar estos parámetros será necesario escribir más líneas de código comparadas con las necesarias si se tuviese el constructor.**