TALLER POO JAVA - GUÍA COMPLETA

# **TALLER INTEGRADOR DE PROGRAMACION ORIENTADA A OBJETOS (POO) EN JAVA**

**Título:** Domina la Herencia con Ejercicios Reales y Reflexivos

**Objetivo General:**

Aplicar conceptos fundamentales de la POO en Java mediante el desarrollo de aplicaciones temáticas que integran herencia (simple, jerárquica y multinivel), encapsulamiento, colecciones, sobreescritura, sobrecarga, polimorfismo y menús interactivos.

**Conceptos clave a aplicar en TODOS los ejercicios:**

* Clases y objetos
* Encapsulamiento (private, getters, setters)
* Herencia (simple, jerarquica, multinivel)
* Sobrecarga de metodos (mismo nombre, distintos parametros)
* Sobreescritura (@Override)
* Polimorfismo (colecciones de tipo base y llamadas a metodos polimorficos)
* Colecciones (ArrayList)
* Menú interactivo (Scanner + switch)
* Separación en paquetes (modelo, controlador, main)

# **=== GRUPO 4: SISTEMA DE TIENDA ===**

**Introducción:**

Gestiona productos como alimentos, bebidas y bebidas energéticas con descuentos.

**Pasos:**

1. Clase Producto: nombre, precio, mostrarInformacion().
2. Subclases: Alimento -> Bebida -> BebidaEnergetica.
3. Sobrescribir mostrarInformacion().
4. Sobrecarga: aplicarDescuento(double), aplicarDescuento(int, double).
5. Crear ArrayList<Producto>.
6. Polimorfismo para mostrar productos.
7. Crear menu interactivo.

**Preguntas:**

* + ¿Como manejaste los descuentos de forma flexible?
  + ¿Qué aporta sobrecargar aplicarDescuento?

# **=== PREGUNTAS FINALES PARA TODOS ===**

* + ¿Qué tipo de herencia aplicaste? ¿Cuál fue su propósito?
  + ¿Qué errores encontraste y como los resolviste?
  + ¿Como evalúas el uso de polimorfismo en tu sistema?
  + ¿Podrías extender el sistema fácilmente con una nueva clase?

# **=== ERRORES COMUNES A DETECTAR ===**

| Error | Causa | Solución |

|-----------------------------------|------------------------------------| |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| | Acceso a atributo privado directo | Falta de encapsulamiento | Usar getters y setters |  |  |  | | |
| | Sobrecarga incorrecta | Parámetros duplicados | Cambiar cantidad o tipo |  | | |  |  |
| | @Override falla | Método mal definido o mal llamado | Revisar nombre y firma |  |  | | |  |
| | Constructor sin super() | No se llama al padre | Usar super() en subclases | | |  |  |  |

| Sin polimorfismo | Uso de tipo especifico en lista | Usar clase base como tipo genérico|

# === TEXTO FINAL REFLEXIVO ===

"El verdadero poder de la programación orientada a objetos no está solo en escribir código que funcione, sino en construir estructuras que puedan evolucionar, mantenerse y adaptarse a nuevas necesidades. Cada clase que hereda, cada método que se sobrescribe, y cada colección que abstrae nos ensena a pensar en sistemas como organismos vivos. Reflexiona: ¿Qué tan clara es tu estructura para un nuevo programador? Eso es diseño inteligente, eso es ser desarrollador con propósito."