PROGRAMACIÓN ORIENTADA A OBJETOS

La siguiente guía se basa en el siguiente programa que consta de tres clases, las cuales deben estar en el mismo paquete.

```
public class Tienda {


    public static void main(String[] args) {


        Producto manzana = new Producto("Manzana", 0.50);
        Producto leche = new Producto("Leche", 1.20);
        Producto pan = new Producto("Pan", 1.50);

        System.out.println("Â¡Bienvenido a la tienda! Productos
disponibles:");
        System.out.println("- " + manzana.nombre + ": $" + manzana.precio);
        System.out.println("- " + leche.nombre + ": $" + leche.precio);
        System.out.println("- " + pan.nombre + ": $" + pan.precio);
        System.out.println("-----------------------------------------");


        Usuario ana = new Usuario("Ana");
        System.out.println("El usuario " + ana.nombre + " ha entrado a la
tienda.");


        ana.agregarProductoAlCarrito(manzana);
        ana.agregarProductoAlCarrito(leche);
        ana.agregarProductoAlCarrito(pan);

        System.out.println("-----------------------------------------");


        double totalCompra = ana.calcularTotal();
        System.out.println("Calculando el total de la compra para " +
ana.nombre + "...");
        System.out.println("El total de la compra es: $" + totalCompra);
    }
}
```

```
import java.util.ArrayList;

public class Usuario {
    String nombre;
    ArrayList<Producto> carrito;
```

```
    public Usuario(String nombre) {
        this.nombre = nombre;
        this.carrito = new ArrayList<Producto>();
    }


    public void agregarProductoAlCarrito(Producto producto) {
        this.carrito.add(producto);
        System.out.println(producto.nombre + " ha sido agregado al
carrito de " + this.nombre);
    }


    public double calcularTotal() {
        double total = 0.0;
        for (Producto producto : this.carrito) {
            total = total + producto.precio;
        }
        return total;
    }
}
```

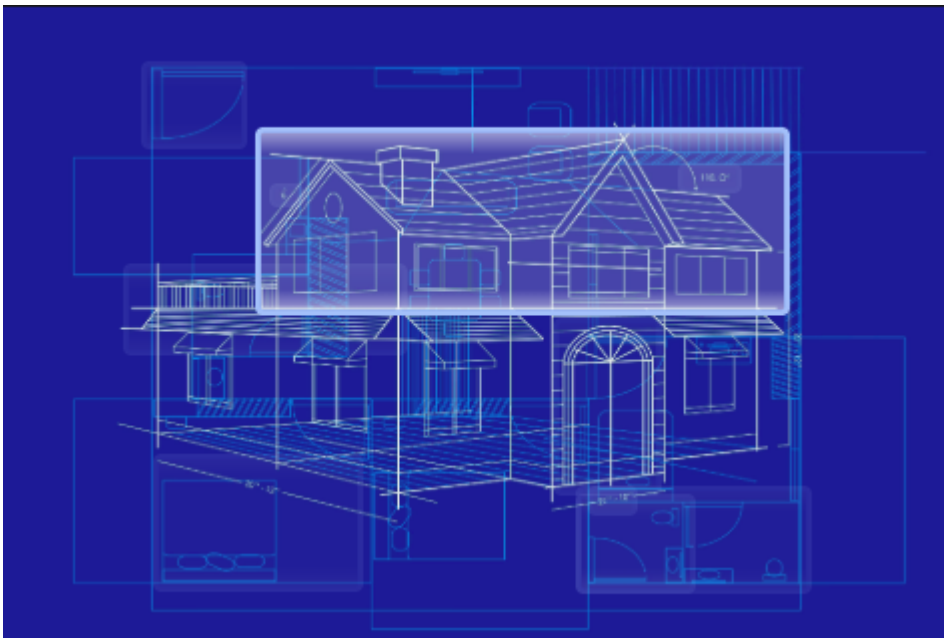
```
public class Producto {
    String nombre;
    double precio;


    public Producto(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }
}
```

¿Qué es el Paradigma POO?

La **Programación Orientada a Objetos (POO)** es una forma de programar que intenta imitar el mundo real. En lugar de escribir una lista larga de instrucciones (como una receta de cocina lineal), organizamos el código en "cosas" o **Objetos** que interactúan entre sí.

El Plano: La Clase (`Class`)

Una **Clase** es la **plantilla** o el **molde**. Define cómo será un objeto, pero no es el objeto en sí mismo. Piensa en el plano de una casa: el plano te dice dónde van las paredes, pero no puedes vivir en el plano.



En el código de ejemplo: Tenemos 3 moldes (clases):

1. `Producto`: Define qué es un producto (tiene nombre y precio).
2. `Usuario`: Define qué es un cliente (tiene nombre y un carrito).
3. `Tienda`: Es la clase principal que pone a funcionar todo.

```
// Esto es el MOLDE. Dice que todo Producto tendrá un nombre y un
//precio.
public class Producto { // ...contenido... }
```

Las Características: Atributos

Los **Atributos** (o campos) son las variables dentro de la clase que guardan la información. Son las características que describen al objeto.

En el código de ejemplo:

- **En la clase `Producto`:**
  - `String nombre;` (¿Cómo se llama?)
  - `double precio;` (¿Cuánto cuesta?)
- **En la clase `Usuario`:**
  - `String nombre;` (¿Quién es?)

- `ArrayList<Producto> carrito;` (¿Qué lleva?)

La Acción: Métodos

Los **Métodos** son las **funciones** dentro de una clase. Representan lo que el objeto **sabe hacer** (comportamiento). Son los verbos de la programación.

En el código de ejemplo: El `Usuario` tiene dos habilidades (métodos) principales:

1. **`agregarProductoAlCarrito(Producto producto)`**:
   - *Acción:* Toma un producto y lo mete en la lista (carrito).
   - *Código:* `this.carrito.add(producto);`
2. **`calcularTotal()`**:
   - *Acción:* Suma los precios de todo lo que hay en el carrito y devuelve el resultado.

```
// Esto es un MÉTODO. Define una acción que el Usuario puede realizar.
public void agregarProductoAlCarrito(Producto producto) {
    this.carrito.add(producto);
    // ...
}
```

Hacerlo Realidad: Instancias (Objetos)

Una **Instancia** es cuando usas el molde (Clase) para crear un objeto real en la memoria. Se usa la palabra clave **new**.

- La **Clase** es "Fruta".
- La **Instancia** es "esa Manzana roja que está en la mesa".

En el código de ejemplo: Esto ocurre dentro del `main` en la clase `Tienda`.

```
// Aquí estamos INSTANCIANDO la clase Producto
// 'manzana' es el objeto real creado a partir del molde Producto
Producto manzana = new Producto("Manzana", 0.50);

// Aquí instanciamos un Usuario
Usuario ana = new Usuario("Ana");
```

Análisis del Código Completo

A continuación, el código de ejemplo comentado para identificar cada parte:

Archivo 1: El Molde del Producto

```
public class Producto { // <--- CLASE
    // ATRIBUTOS (Estado)
    String nombre;
    double precio;

    // CONSTRUCTOR: Método especial que se ejecuta al hacer 'new'
    // Sirve para dar valores iniciales a los atributos.
    public Producto(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }
}
```

Archivo 2: El Molde del Usuario

```
import java.util.ArrayList;

public class Usuario { // <--- CLASE
    // ATRIBUTOS
    String nombre;
    ArrayList<Producto> carrito; // Un atributo complejo (una
lista)

    // CONSTRUCTOR
    public Usuario(String nombre) {
        this.nombre = nombre;
        this.carrito = new ArrayList<Producto>(); //
Inicializa la lista vacía
    }

    // MÉTODO 1: Acción de comprar
    public void agregarProductoAlCarrito(Producto producto) {
        this.carrito.add(producto);
        System.out.println(producto.nombre + " ha sido
agregado al carrito de " + this.nombre);
    }
```

```
    // MÉTODO 2: Acción de calcular cuenta
    public double calcularTotal() {
        double total = 0.0;
        // Ciclo for-each para recorrer el atributo 'carrito'
        for (Producto producto : this.carrito) {
            total = total + producto.precio;
        }
        return total; // Devuelve un valor
    }
}
```

Archivo 3: La Ejecución (Tienda)

```
public class Tienda {
    public static void main(String[] args) {

        // 1. INSTANCIACIÓN (Crear objetos reales)
        Producto manzana = new Producto("Manzana", 0.50);
        Producto leche = new Producto("Leche", 1.20);
        Producto pan = new Producto("Pan", 1.50);

        // Imprimiendo ATRIBUTOS directamente
        System.out.println("Productos disponibles:");
        System.out.println("- " + manzana.nombre + ": $" +
manzana.precio);

        // 2. INSTANCIACIÓN de otro objeto (Usuario)
        Usuario ana = new Usuario("Ana");
        System.out.println("El usuario " + ana.nombre + " ha entrado.");

        // 3. LLAMADA DE MÉTODOS (Interacción entre objetos)
        // El objeto 'ana' interactúa con los objetos producto
        ana.agregarProductoAlCarrito(manzana);
        ana.agregarProductoAlCarrito(leche);
        ana.agregarProductoAlCarrito(pan);

        // 4. OBTENER RESULTADOS
        double totalCompra = ana.calcularTotal(); // Ejecuta el método y
guarda el resultado
        System.out.println("El total de la compra es: $" + totalCompra);
    }
```

```
}
```

Resumen Rápido

Concepto	Definición	Ejemplo en tu código
Clase	El molde o plantilla.	`public class Usuario`
Atributo	Datos o características.	`String nombre;`, `double precio;`
Método	Acciones o funciones.	`calcularTotal()`
Instancia	El objeto creado en memoria.	`ana`, `manzana`, `leche`
Constructor	Configuración inicial.	`public Usuario(String nombre)...`

TALLER:

1. Sistema de Biblioteca
La idea es simular a un miembro de una biblioteca que toma prestado un libro.

Clase 1: Libro

Atributos: titulo (String), autor (String).

Constructor: Para inicializar el título y el autor.

Clase 2: Miembro

Atributos: nombre (String), librosPrestados (una lista de objetos Libro).

Constructor: Para inicializar el nombre del miembro.

Métodos:

prestarLibro(Libro libro): Agrega un libro a la lista del miembro.

devolverLibro(Libro libro): Elimina un libro de la lista.

mostrarLibros(): Muestra los libros que el miembro tiene actualmente.

Clase Principal: Biblioteca

Crea varios objetos Libro.

Crea un objeto Miembro.

Simula que el miembro toma prestados uno o dos libros y luego muestra la lista de libros que tiene.

2. Cuenta Bancaria Sencilla
Simula a un cliente que realiza depósitos y retiros en su cuenta bancaria.

Clase 1: CuentaBancaria

Atributos: numeroDeCuenta (String), saldo (double).

Constructor: Para inicializar el número de cuenta y un saldo inicial.

Métodos:

depositar(double cantidad): Aumenta el saldo.

retirar(double cantidad): Disminuye el saldo (puedes añadir una comprobación simple para que no quede negativo).

consultarSaldo(): Muestra el saldo actual.

Clase 2: Cliente

Atributos: nombre (String), cuenta (un objeto CuentaBancaria).

Constructor: Para inicializar el nombre y asociarle una cuenta.

Clase Principal: Banco

Crea un objeto CuentaBancaria con un saldo inicial.

Crea un objeto Cliente y le asigna esa cuenta.

Simula que el cliente hace un depósito y un retiro, mostrando el saldo después de cada operación.

3. Gestión de Mascotas
Modela a una persona que adopta una mascota y la hace realizar acciones.

Clase 1: Mascota

Atributos: nombre (String), especie (por ejemplo, "Perro" o "Gato"), sonido (String, por ejemplo, "Guau" o "Miau").

Constructor: Para inicializar sus atributos.

Métodos:

hacerSonido(): Imprime el sonido de la mascota.

presentarse(): Imprime un mensaje como "Hola, soy un [especie] y me llamo [nombre]".

Clase 2: Dueño

Atributos: nombre (String), mascota (un objeto Mascota).

Constructor: Para inicializar el nombre del dueño.

Métodos:

adoptarMascota(Mascota nuevaMascota): Asigna una mascota al dueño.

darOrden(String orden): Si la orden es "habla", llama al método hacerSonido() de su mascota.

Clase Principal: Hogar

Crea un objeto Mascota.

Crea un objeto Dueño.

Simula que el dueño adopta a la mascota y luego le pide que haga su sonido.

4. Inscripción a Cursos
Simula a un estudiante que se inscribe en diferentes cursos de una escuela.

Clase 1: Curso

Atributos: nombreMateria (String), codigo (String).

Constructor: Para inicializar los datos del curso.

Clase 2: Estudiante

Atributos: nombre (String), idEstudiante (String), cursosInscritos (una lista de objetos Curso).

Constructor: Para inicializar el nombre y el ID.

Métodos:

inscribirCurso(Curso curso): Agrega un curso a la lista del estudiante.

mostrarCursos(): Imprime la lista de materias en las que está inscrito.

Clase Principal: Escuela

Crea varios objetos Curso (Matemáticas, Historia, etc.).

Crea un objeto Estudiante.

Simula que el estudiante se inscribe en dos o tres cursos y finalmente muestra su horario.

5. Pedido en una Pizzería
Modela un pedido que contiene varias pizzas y calcula el total a pagar.

Clase 1: Pizza

Atributos: sabor (String, ej. "Hawaiana"), tamaño (String, ej. "Mediana"), precio (double).

Constructor: Para definir las características de la pizza.

Clase 2: Pedido

Atributos: nombreCliente (String), pizzas (una lista de objetos Pizza).

Constructor: Para inicializar el nombre del cliente.

Métodos:

agregarPizza(Pizza pizza): Añade una pizza al pedido.

calcularTotal(): Suma los precios de todas las pizzas en la lista y devuelve el total.

Clase Principal: Pizzeria

Crea diferentes objetos Pizza con distintos sabores y precios.

Crea un objeto Pedido para un cliente.

Agrega varias pizzas a ese pedido.

Calcula y muestra el total que el cliente debe pagar.