Programación Orientada a Objetos (POO)

1. Paradigma de Programación

Un **paradigma de programación** es una forma de ver y estructurar la solución a problemas mediante el desarrollo de software.
En este caso, la **POO** es un paradigma que se centra en **modelar el mundo real o abstracto** mediante **clases y objetos**, permitiendo representar entidades físicas (personas, autos, productos) o no físicas (conceptos, cualidades, procesos).

---

2. Principios Fundamentales de la POO

2.1 Abstracción

- Permite identificar lo esencial de un objeto del mundo real y trasladarlo a un modelo computacional.
- Ejemplo: De una "Persona", se abstraen atributos como nombre, edad y métodos como hablar() o caminar().

2.2 Encapsulamiento

- Consiste en ocultar los detalles internos de una clase y exponer solo lo necesario a través de métodos públicos.
- Protege los datos y promueve la seguridad y mantenibilidad del código.

2.3 Herencia

- Permite que una clase (subclase) herede atributos y métodos de otra (superclase).
- Facilita la reutilización de código y la creación de jerarquías lógicas.

2.4 Polimorfismo

- Se refiere a la capacidad de un mismo método o función de comportarse de manera distinta según el objeto que lo invoque.
- Ejemplo: un método **calcularÁrea()** puede tener un comportamiento distinto en una clase Círculo y en una clase Rectángulo.

---

3. Componentes Clave en la POO

3.1 Clase

Una **clase** es una plantilla que describe un objeto. Contiene:

- **Nombre:** Identifica la entidad.
- **Atributos:** Características o propiedades (variables).
- **Métodos:** Acciones o comportamientos que la clase puede realizar.

Ejemplo en pseudocódigo:

```
class Persona {
    String nombre;
    int edad;

    void hablar() {
        // acción
    }

    void caminar() {
        // acción
    }
}
```

3.2 Objeto

Un **objeto** es una instancia de una clase, es decir, la materialización de esa plantilla en memoria.

- Permite usar atributos y métodos definidos en la clase.

Ejemplo:

```
Persona p1 = new Persona();
p1.nombre = "Ariel";
p1.hablar();
```

---

4. Ventajas de la POO

- **Reutilización de código:** gracias a la herencia y al polimorfismo.
- **Modularidad:** los programas se organizan en clases y objetos.
- **Mantenibilidad:** facilita el mantenimiento y la actualización del software.
- **Escalabilidad:** se pueden añadir nuevas clases y funcionalidades sin alterar drásticamente el código existente.
- **Naturalidad:** modela el mundo real de forma intuitiva.

---

5. Ejemplo Integrador

Supongamos que queremos modelar un sistema simple de gestión de animales:

```
class Animal {
    String nombre;
    int edad;

    void comer() {
        System.out.println(nombre + " está comiendo.");
    }
```

```
}

class Perro extends Animal {
    void ladrar() {
        System.out.println(nombre + " está ladrando.");
    }
}

class Gato extends Animal {
    void maullar() {
        System.out.println(nombre + " está maullando.");
    }
}

// Instanciación
Perro perro = new Perro();
perro.nombre = "Firulais";
perro.comer();
perro.ladrar();

Gato gato = new Gato();
gato.nombre = "Michi";
gato.comer();
gato.maullar();
```

Este ejemplo ilustra:

- **Herencia** (Perro y Gato heredan de Animal).
- **Abstracción** (se modelan animales con propiedades y comportamientos).
- **Polimorfismo** (el método `comer()` se puede invocar en distintas clases hijas).
- **Encapsulamiento** (los atributos y métodos podrían restringirse con modificadores de acceso).

---

6. Conclusiones

- La **POO** es un paradigma poderoso y ampliamente usado en el desarrollo de software.
- Permite **modelar problemas complejos de manera más cercana al mundo real**, facilitando el diseño y mantenimiento del código.
- Al dominar sus principios —abstracción, encapsulamiento, herencia y polimorfismo— se adquiere una base sólida para el desarrollo de aplicaciones modernas y escalables.