INTERFACE AND ABSTRACT

Clase Abstracta

Definición:
Plantilla para otras clases. No se pueden instanciar.
Palabra clave: abstract
Propósito: Definir una estructura común y obligar a implementar métodos.
Características:

- Pueden tener métodos normales y abstractos.
- Las subclases deben implementar los métodos abstractos.

Abstract

```
interface Volador {
    void volar();
}

interface Nadador {
    void nadar();
}

class Pato implements Volador, Nadador {
    public void volar() { System.out.println("Vuelo bajo"); }
    public void nadar() { System.out.println("Nado en el lago"); }
}
```

interface

```
interface Volador {
    void volar();
}

interface Nadador {
    void nadar();
}

class Pato implements Volador, Nadador {
    public void volar() { System.out.println("Vuelo bajo"); }
    public void nadar() { System.out.println("Nado en el lago"); }
}
```

Interfaces

Definición: Contrato que define métodos (sin implementación).
Palabra clave: implements
Propósito: Lograr polimorfismo y herencia múltiple.
Características:

- Solo tienen métodos abstractos (implícitamente).
- Una clase puede implementar varias interfaces.
- Obligan a implementar todos los métodos definidos.

INTERFACE AND ABSTRACT

Polimorfismo

Definición:
Capacidad de un objeto de tomar muchas formas.
Para qué sirve: Permite que un mismo método se comporte de manera diferente según el objeto que lo ejecute.

CUANDO USAR CLASES ABSTRACTAS:

- Cuando necesitas herencia múltiple
- Cuando defines un contrato para clases no relacionadas
- Para desacoplar código (principio de inversión de dependencias)

CUANDO USAR INTERFACES

- Cuando necesitas herencia múltiple
- Cuando defines un contrato para clases no relacionadas
- Para desacoplar código (principio de inversión de dependencias)

Default

- **Propósito**:

Permiten agregar nuevos métodos a una interfaz **sin romper** las clases que ya la implementan.

- **Características**:
  - Tienen una implementación concreta dentro de la interfaz.
  - Se definen con la palabra clave default.
  - Pueden ser sobrescritos (@Override) en las clases que implementan la interfaz.

```
    default void detener() {
        System.out.println("El vehículo se ha detenido.");
    }
}

public class Coche implements Vehiculo {
    @Override
    public void arrancar() {
        System.out.println("Coche arrancado.");
    }
```

Default

- **Propósito**:
- Permiten definir métodos de utilidad asociados a la interfaz.
- **Características**:
  - Se definen con static.
  - No pueden ser sobrescritos por las clases implementadoras.
  - Se invocan directamente desde la interfaz: NombreInterfaz.metodoEstatico()

```
static double sumar(double a, double b) {
    return a + b;
}
```

```
double resultado = Calculadora.sumar(5, 3);
```