Concepto Interfaces.

🧩 Interfaces en Java

Cuando hablamos de **interfaces**, es común confundir el término con “interfaz gráfica de usuario”. Sin embargo, en Java, una **interfaz** es un **concepto de la Programación Orientada a Objetos (POO)** totalmente diferente.

Una **interfaz** es una **clase completamente abstracta** que define un **contrato** que las clases concretas deben cumplir. Dicho contrato especifica **qué puede hacer una clase**, pero **no cómo lo hace**.

Las interfaces muchas veces son definidas como un tipo de contrato entre las clases concretas que la implementen, ya que la clase que lo haga se encuentra obligada a definir los métodos abstractos que la componen.

🔷 Concepto general

Las interfaces permiten declarar métodos y constantes que serán implementados por otras clases. Las clases que implementen una interfaz están **obligadas a definir** el comportamiento de todos sus métodos abstractos.

En otras palabras, una interfaz **establece un conjunto de reglas** que una clase concreta debe seguir.

⚙️ Características principales

- Se definen con la palabra clave `interface`.
- Sus métodos son **públicos y abstractos** por defecto (no es necesario especificarlo).
- Sus variables son **constantes implícitas** (`public static final`).
- No pueden ser instanciadas directamente.
- Una clase puede **implementar múltiples interfaces**, lo que permite **simular la herencia múltiple**.
- Una interfaz puede **heredar (extends)** de una o más interfaces, pero **no de clases concretas**.
- Pueden ser **públicas** o tener **acceso por paquete** (por defecto).
- Los tipos de interfaz pueden ser usados **polimórficamente**.

🧠 Desde Java 8 y 9

Las versiones modernas de Java ampliaron las capacidades de las interfaces:

- **Métodos** `default`: permiten incluir una implementación por defecto dentro de la interfaz.
- **Métodos** `static`: pueden tener implementación y ser invocados directamente desde la interfaz.
- **Métodos** `private` (Java 9+): pueden usarse dentro de la interfaz para reutilizar código interno.

Ejemplo:

```
interface Operaciones {
    int sumar(int a, int b); // Método abstracto

    default void mostrarResultado() {
        System.out.println("Mostrando resultado...");
    }

    static void info() {
        System.out.println("Método estático dentro de la interfaz");
```

```
    }
}
```

🧬 Herencia y clases abstractas

- Una **interfaz puede heredar de otra interfaz**, pero **no está obligada** a implementar sus métodos.
- Si una **clase abstracta** implementa una interfaz, **no está obligada** a definir sus métodos; la responsabilidad recae en la clase concreta que la extienda.

🚫 Importante recordar

- Las interfaces no se pueden instanciar.
- No pueden contener variables de instancia.
- Sus métodos no pueden ser `final`, `synchronized` ni `native`.

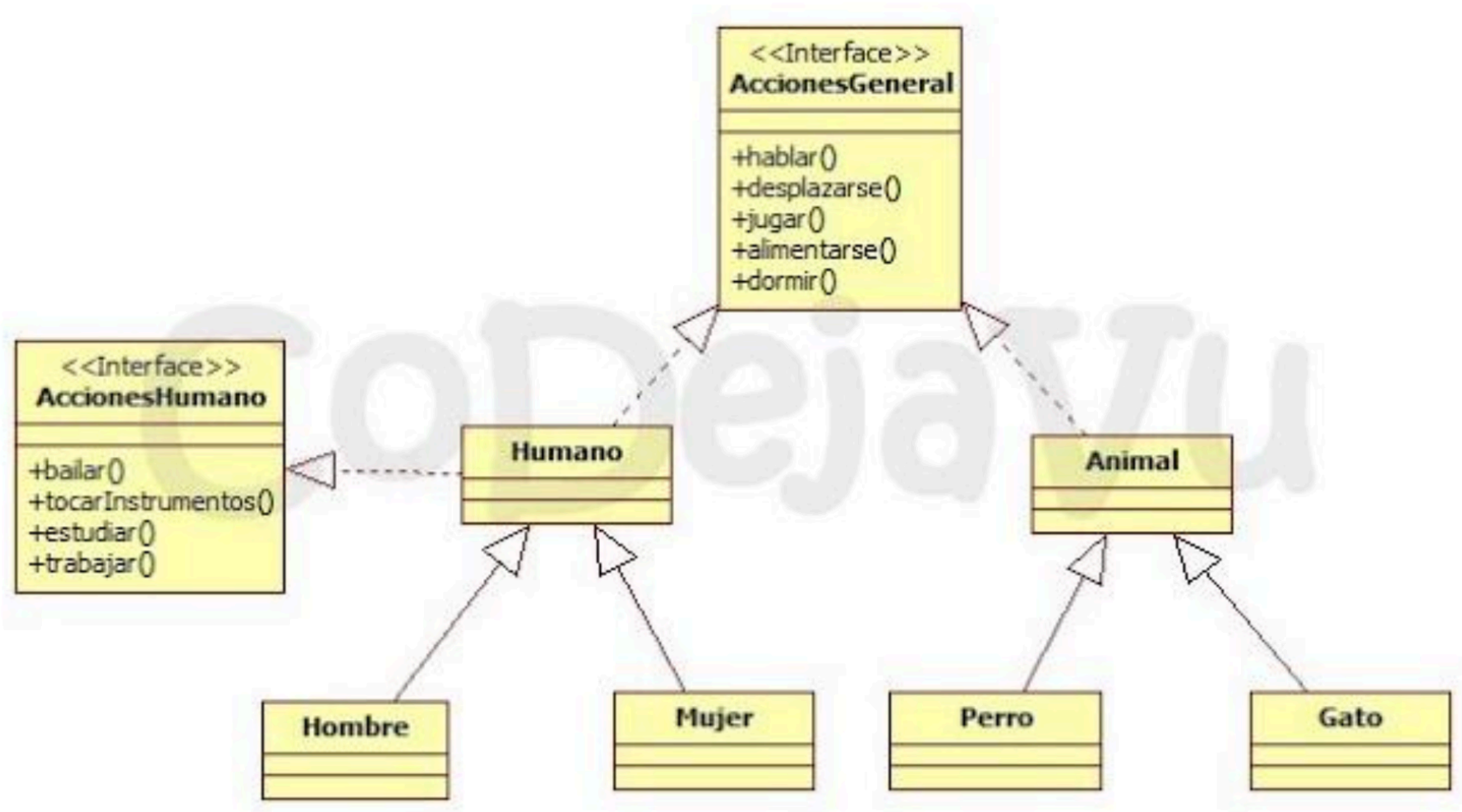
💡 En resumen

> Una interfaz en Java define un conjunto de comportamientos que las clases deben cumplir. Representa un contrato que describe qué hace una clase, pero no cómo lo hace. Permite lograr polimorfismo, desacoplamiento y herencia múltiple de comportamiento en Java.

Su uso está muy ligado al concepto de herencia, ya que permite definir comportamientos comunes que pueden ser compartidos por diferentes clases. A diferencia de las clases abstractas, las interfaces no comparten atributos de instancia, sino que establecen un contrato de métodos que las clases deben implementar. De esta forma, solo las clases que implementen la interfaz podrán acceder y definir el comportamiento de sus métodos.

⚠️ Ejemplo.

Veamos el siguiente diagrama de clases donde se puede evidenciar el uso de 2 interfaces con propósito diferente pero comportamiento útil para las clases que lo implementan.



Construya un proyecto en Java y cree el código correspondiente según el diagrama anterior.

Instructor: Cristian David Henao Hoyos