Una **interfaz funcional** en Java es una interfaz que contiene **exactamente un método abstracto**.

Esta característica las hace ideales para ser implementadas mediante **expresiones lambda**, lo que permite un código más conciso y expresivo. Son la base de la programación funcional introducida en Java 8.

## **Características Principales**

* **Un Único Método Abstracto (SAM):** La regla fundamental es que solo pueden declarar **un** método abstracto. Si una interfaz tiene más de uno o ninguno, no es una interfaz funcional.
* **Anotación @FunctionalInterface:** Aunque no es obligatoria, es una buena práctica anotar la interfaz con @FunctionalInterface. Esta anotación hace que el **compilador verifique** que la interfaz cumple con la regla del método abstracto único, evitando errores.
* **Métodos default y static:** Una interfaz funcional puede contener cualquier cantidad de métodos default o static, ya que estos tienen una implementación y no son abstractos.

## **Ejemplos**

### **Interfaz Funcional Propia**

Aquí se define una interfaz funcional simple y se usa con una expresión lambda.

// 1. Definición de la interfaz funcional

@FunctionalInterface

interface OperacionMatematica {

int calcular(int a, int b); // Único método abstracto

}

public class Calculadora {

public static void main(String[] args) {

// 2. Implementación usando una expresión lambda

OperacionMatematica suma = (a, b) -> a + b;

OperacionMatematica resta = (a, b) -> a - b;

// 3. Uso

System.out.println("Suma: " + suma.calcular(10, 5)); // Salida: Suma: 15

System.out.println("Resta: " + resta.calcular(10, 5)); // Salida: Resta: 5

}

}

### **Interfaces Funcionales Nativas de Java**

Java proporciona un conjunto de interfaces funcionales de uso común en el paquete java.util.function.

* Predicate<T>: Recibe un argumento y devuelve un booleano.  
  Predicate<String> esLargo = s -> s.length() > 10;  
  System.out.println(esLargo.test("HolaMundoJava")); // true
* Function<T, R>: Recibe un argumento de tipo T y devuelve un resultado de tipo R.  
  Function<String, Integer> longitud = s -> s.length();  
  System.out.println(longitud.apply("Java")); // 4
* Consumer<T>: Recibe un argumento y no devuelve nada (realiza una acción).  
  Consumer<String> imprimir = s -> System.out.println(s);  
  imprimir.accept("¡Hola, mundo!"); // Imprime "¡Hola, mundo!"
* Supplier<T>: No recibe argumentos, pero devuelve un valor.  
  Supplier<Double> numeroAleatorio = () -> Math.random();System.out.println(numeroAleatorio.get()); // Imprime un número aleatorio