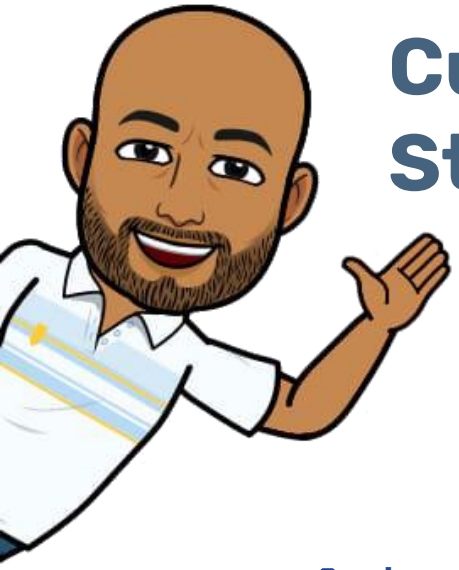


Curso de Java Standard

Ing. Octavio Robleto

octavio.robleto@gmail.com

https://octaviorobleto.com

Java 8

En 2014 Java lanza una versión el cual añade numerosas novedades importantes al lenguaje y que divide el antes y después de esta versión.

Las novedades mas importantes que podemos destacar son el agregado del paquete **stream** y la implementación del paradigma de programación funcional.

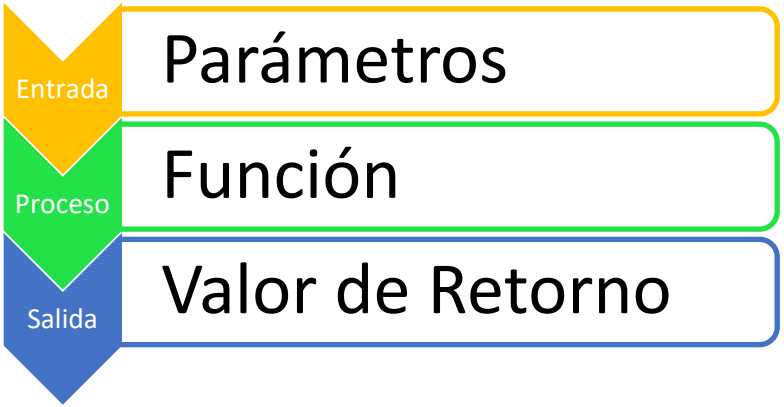
Ya habíamos hablado de los métodos por defecto y estáticos en la interfaces que se agregaron en esta versión y son muy importantes para las otras funcionalidades que vamos a ver en este apartado.

Paradigma Funcional

Recordemos que un paradigma no es mas que una forma de resolver un problema planteado desde un enfoque diferente.

Este paradigma es declarativo, por lo que nos enfocaremos en el **qué** y no en **cómo** se esta resolviendo el problema, es decir, nosotros expresaremos nuestra lógica sin describir controles de flujo; ciclos y/o condicionales.

La programación orientada objetos es imperativa que no es mas que decir el control de flujo (Ciclos y Condicionales) a nuestro software.



Interfaz Funcional

Son aquellas interfaces que poseen si y solo si un solo método abstracto (sin cuerpo).

Puede poseer o no la notación **@FunctionalInterface** que previene que otro desarrollador agregue mas métodos abstractos generando errores en tiempo de compilación.

Adicionalmente puede tener métodos por defecto y estáticos sin restricciones.

```
@FunctionalInterface
public interface interfaz {

    retorno metodo(tipo parametros);

}
```

Lambdas

Para poder implementar el paradigma funcional se agrega el concepto de lambdas que no es mas que un método anónimo, básicamente es un método abstracto que sólo está definido en una interfaz (No necesitan una clase) y está debe ser una interfaz funcional.

Estructura

Parámetros:

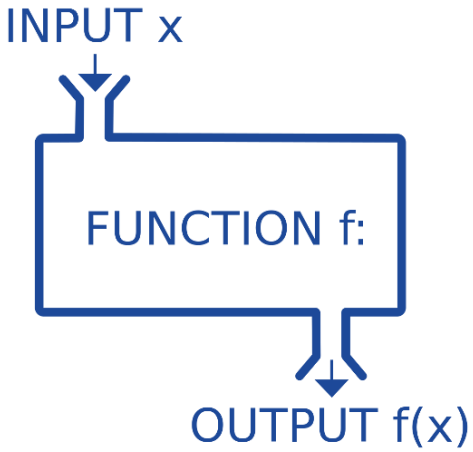
1. Cuando se tiene un solo parámetro no es necesario utilizar los paréntesis.
2. Cuando no se tienen parámetros, o cuando se tienen dos o más, es necesario utilizar paréntesis.
3. Esta implícito el tipo de objeto que va a utilizar por lo que no es necesario declararlo en el parámetro.

El operador lambda:

Operador flecha (->) que separa la declaración de parámetros de la declaración del cuerpo de la función.

Cuerpo:

1. Cuando el cuerpo de la expresión lambda tiene una única línea no es necesario utilizar las llaves y no necesitan especificar la sentencia **return** en el caso de que deban devolver valores.
2. Cuando el cuerpo de la expresión lambda tiene más de una línea se hace necesario utilizar las llaves ...nción deba devolver un valor.



Ejemplo Imperativo

```java
public interface CalculadoraTradicional {

    public static int sumar(int a, int b) {
        return a + b;
    }

    public static int restar(int a, int b) {
        return a - b;
    }

    public static int multiplicar(int a, int b) {
        return a * b;
    }

    public static int dividir(int a, int b) {
        return (b == 0 ? 0 : a / b);
    }
}
```

```java
int a = 4;
int b = 2;
int resultado;

resultado = CalculadoraTradicional.sumar(a, b);
System.out.println("Sumar (" + a + " , " + b + "): " + resultado);
resultado = CalculadoraTradicional.restar(a, b);
System.out.println("Restar (" + a + " , " + b + "): " + resultado);
resultado = CalculadoraTradicional.multiplicar(a, b);
System.out.println("Multiplicar (" + a + " , " + b + "): " + resultado);
resultado = CalculadoraTradicional.dividir(a, b);
System.out.println("Dividir (" + a + " , " + b + "): " + resultado);
```

Ejemplo Declarativo

A simple vista parece lo mismo o hasta mas complicado pero la idea es poder cambiar el comportamiento de los métodos en el momento que necesitemos sin declararlos previamente.

En este ejemplo le cambiamos el comportamiento a nuestro método para hacer las operaciones que hicimos en el ejemplo Imperativo y adicionalmente pudimos darle el comportamiento necesario para que retorne el modulo, también podemos seguir agregando sin problemas.

```java
@FunctionalInterface
public interface CalculadoraFuncional {
    int operar(int a, int b);
}
```

```java
int a = 4;
int b = 2;
int resultado;

CalculadoraFuncional operaracion;

operaracion = (numA, numB) -> numA + numB;
resultado = operaracion.operar(a, b);
System.out.println("Sumar (" + a + " , " + b + "): " + resultado);

operaracion = (numA, numB) -> numA - numB;
resultado = operaracion.operar(a, b);
System.out.println("Restar (" + a + " , " + b + "): " + resultado);

operaracion = (numA, numB) -> numA * numB;
resultado = operaracion.operar(a, b);
System.out.println("Multiplicar (" + a + " , " + b + "): " + resultado);

operaracion = (numA, numB) -> (numB == 0 ? 0 : numA / numB);
resultado = operaracion.operar(a, b);
System.out.println("Dividir (" + a + " , " + b + "): " + resultado);

operaracion = (numA, numB) -> numA % numB;
resultado = operaracion.operar(a, b);
System.out.println("Modulo (" + a + " , " + b + "): " + resultado);
```