Elaboro	Documento	Versión	Descripción	Fecha
Luis Sánchez Martínez	Práctica Programación Funcional	1.0	Creación de documento	25/03/2024

Práctica Programación Funcional.

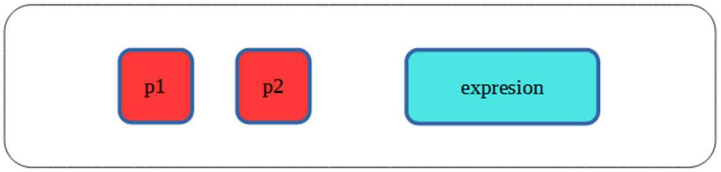
Tiempo estimado: 1 hora.

Uso de:

- Java 17
- IntelliJ
- Programación Funcional

Programación Funcional en Java

La programación funcional es un paradigma de programación basado en funciones matemáticas, los lenguajes de programación funcional son aquellos lenguajes donde las variables no tienen estado.



Las variables que no tienen estado son aquellas donde no hay cambios a lo largo del tiempo, no pueden cambiarse los valores a lo largo de la ejecución.

En estos lenguajes los programas se estructuran como expresiones que se evalúan como funciones. Ejemplo de estos lenguajes son lisp, scheme, clojure, etc.

En este caso Java es un lenguaje híbrido que va a componerse del lenguaje imperativo que es el que conocemos y el lenguaje de la programación funcional.

En la programación funcional las instrucciones cíclicas: *for*, *while*, *do while* no existe, todo se procesa usando recursividad y funciones alto orden. Pero en Java vamos a tener las dos formas.

En Java por medio de las expresiones lambda vamos a poder hacer uso de la inferencia de tipos o poder prescindir de declarar los tipos de datos y dejar ese trabajo al compilador por medio de la inferencia de tipos.

Ejercicio 1. Conoce cuantos números mayores a 10 hay en la lista.

```
List<Integer> numeros = List.of(11, 8, 9, 15, 39, 1, 4, 83);
```

Resolviendo por Programación declarativa:

```java
// Declarativo
List<Integer> numeros = List.of(11, 8, 9, 15, 39, 1, 4, 83);

int contador = 0;

for (int numero: numeros) {
    if(numero > 10){
        contador ++;
    }
}

System.out.println("Forma declarativa: "+contador);
```

Resolviendo por Programación imperativa:

```java
// Imperativo
List<Integer> numeros = List.of(11, 8, 9, 15, 39, 1, 4, 83);

Long resultado = numeros.stream().filter(num -> num >
10).count();
System.out.println("Forma imperativa: "+resultado);
```

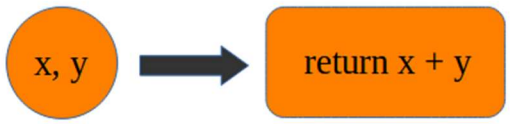
Con la programación funcional las líneas de código se reducen y nuestro código se muestra más legible.

La implementación imperativa es funcional debido a que delega el flujo a las funciones *filter* y *count*.

Expresión lambda

```
parámetro -> cuerpo de la expresión lambda
```

La sintaxis de las expresiones lambda son:



Una expresión lambda representa el método abstracto de una interfaz funcional.

Una interfaz funcional es aquella que solo tiene un método abstracto, puede tener cualquier cantidad de métodos default o estáticos, pero solamente puede tener un método abstracto. Este método puede ser representado por una expresión lambda.

Ejercicio 2. Declaración de una interfaz funcional.

```java
interface operacion {

    // el metodo abstracto
    public double suma(double x, double y);
}
```

Interfaces funcionales

Una interfaz funcional es aquella que solo tiene un método abstracto, podemos utilizar métodos default, métodos estáticos y métodos heredados de la clase *Object* y declararlos como métodos abstractos.

Si la interface que estamos declarando contiene la Anotación *@FunctionalInterface* y no cumple con los criterios para que sea una interfaz funcional nos dará un error de compilación, esto ayuda y es una buena práctica para desarrollar correctamente.

Una interface con un solo método abstracto como lo hemos comentado sigue siendo una interfaz funcional, aunque no tenga la Anotación *@FunctionalInterface*.

Ejercicio 3. Declaración de una interfaz funcional con *@FunctionalInterface*.

```java
// Anotacion para declarar la interface
@FunctionalInterface
interface operacion {

    // el metodo abstracto
    public double suma(double x, double y);
}
```

Métodos referenciados

Las expresiones lambda sirven para reemplazar los métodos anónimos en el caso de las interfaces funcionales. Algunas veces una expresión lambda no hace otra cosa más que llamar a un método que ya existe, en estos casos podría ser más claro referirnos al método que ya existe por su nombre.

Los métodos referenciados nos permiten referirnos a los métodos que ya existen mediante su nombre.

Ejercicio 4. Uso de métodos referenciados.

```java
public static void main(String[] args) {

    // utilizaremos de la interface Consumer el unico metodo denominado accept
    // el cual recibe un parametro y no devuelve nada
    Consumer<String> consumidor = x -> System.out.println(x);
    consumidor.accept("Bienvenido en metodo accept");

    procesar(x -> System.out.println(x), "Bienvenido en metodo procesar");

    // Pasamos como referencia una funcion que se encuentra en nuestro propio programa
    procesar(MetodoReferenciado::saludar, "Hola");
}
```

Referencias

Vázquez González, Marcos. (2021, 30 de enero). Introducción a la Programación Funcional en Java. Codemind. Java. [en línea]. https://blog.codmind.com/que-es-la-programacion-funcional-en-java/