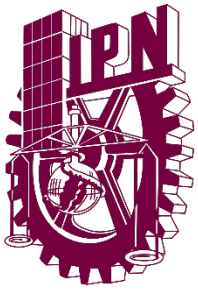


Instituto Politécnico Nacional
Escuela Superior de Computo

Departamento de Ciencias e Ingeniería de la Computación

Paradigmas de Programación

Profesor: Miguel Ángel Rodríguez Castillo

Grupo: 3CV2

Practica: 1

Equipo:

González Oloarte Diego Enrique

Mora Campos Ricardo Uriel

Fecha de entrega: 19/03/2021

Planteamiento del problema

Se busca implementar soluciones a problemas utilizando el paradigma de Programación Funcional.

Los problemas son los siguientes:

1. Conocer la cantidad de elementos en la lista mayores a 10
2. Encontrar en una lista de colores si tenemos el color “red”
3. Programar el cálculo del módulo en base 2, utilizando programación imperativa y declarativa.

Implementación de la solución

```java
public void ejercicio1 (){
    List<Integer> numeros = List.of(18,6,4,15,55,78,12,9,8);

    Long result = numeros.stream().filter(num -> num > 10).count();
    /*--------Stream()-------*/
    //"Toda colección tiene un método stream()
    //que transformará dicha estructura en Stream."
    //Sintaxis:
    //[java]
    //List<String> strings = …;
    //Stream<String> streangs = strings.stream();
    //[/java]
    //
    /*--------Filter()-------*/
    //"Devuelve un Stream que contiene los datos una vez se
    //ejecute el filtro especificado por el Lambda del
    //predicado. Es una operación intermedia."
    //Sintaxis: Stream<T> filter(Predicate<? super T> predicate)
    //
    /*--------Count()-------*/
    //"Returns the count of elements in the stream."
    //Syntax: long count()

    System.out.println("Hay " + result + " números mayores a 10");
}
```

```java
public void ejercicio2 (){
    List<String> colors = List.of("rojo","azul","verde","morado","naranja","amarillo");

    System.out.println("¿El color es rojo?: " + colors.contains("rojo"));
    /*--------Contains()-------*/
    //"The contains() method checks whether a string
    //contains a sequence of characters. Returns true if the
    //characters exist and false if not."
    //Syntax: public boolean contains(CharSequence char)
}
```

```java
public void ejercicio3 (int numero){

    /*--- Programación Imperativa ---*/
    System.out.println("Módulo en Programación Imperativa: ");
    int modulo = numero % 2;
    System.out.println( "El módulo de " + numero + " es " + modulo );


    /*--- Programación Declarativa ---*/
    System.out.println("Módulo en Programación Declarativa: ");
    System.out.println( "El módulo de " + numero + " es " + Math.floorMod(numero, 2) );

}
```

Funcionamiento

Para el ejercicio 1:

Instrucción	numeros	result
Definir e inicializar la lista	18, 6, 4, 15, 55, 78, 12, 9, 8	-
Definir e inicializar result		-
Convertir a Stream		18, 6, 4, 15, 55, 78, 12, 9, 8
Filtrar los numeros tales que sean mayores que 10		18, 15, 55, 78, 12
Contar cuántos elementos hay en result		5

Instrucción	numeros	result
Definir e inicializar la lista	15, 19, 26, 3	-
Definir e inicializar result		-
Convertir a Stream		15, 19, 26, 3
Filtrar los numeros tales que sean mayores que 10		15, 19, 26
Contar cuántos elementos hay en result		3

Para el ejercicio 2:

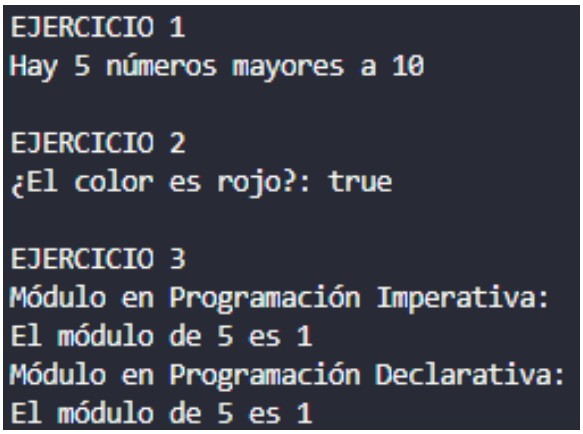
colors	salida
"rojo","azul","verde","morado","naranja","amarillo"	-
	colors.contains("rojo")
	true

colors	salida
"café","azul","verde","morado","naranja","amarillo"	-
	colors.contains("rojo")
	false

Para el ejercicio 3:

Instrucción	numero	modulo	salida
Definir e inicializar modulo	5	1	El módulo de 5 es 1
Imprimir salida usando la función Math.floorMod()	5	1	El módulo de 5 es 1

Instrucción	numero	modulo	salida
Definir e inicializar modulo	512	0	El módulo de 512 es 0
Imprimir salida usando la función Math.floorMod()	512	0	El módulo de 512 es 0



Conclusiones

Esta práctica nos fue muy útil para entender este nuevo paradigma, fue un gran complemento a la teoría que pudimos ver en clase, y creo que además de tener una idea más clara de este paradigma, pudimos tener un acercamiento a Java, utilizando funciones que no conocíamos.

Bibliografía

Escalante, Ó. (2020, 3 agosto). Java 8 – Streams. Oscar Blancarte - Software Architecture. https://www.oscarblancarteblog.com/2017/03/16/java-8-streams-2/#:%7E:text=Los%20Streams%20son%20una%20secuencia,casi%20parecido%20a%20un%20sue%C3%B1o.

Escalante, Ó. (2020, 3 agosto). *Java 8 – Streams*. Oscar Blancarte - Software Architecture. https://www.oscarblancarteblog.com/2017/03/16/java-8-streams-2/#:%7E:text=Los%20Streams%20son%20una%20secuencia,casi%20parecido%20a%20un%20sue%C3%B1o.