Java High Order Functions es uno de los conceptos difíciles de entender del nuevo universo de las expresiones Lambda . Una High Order Function es una función que recibe como parámetro otra función o bien devuelve una función . Parece algo cuando menos extraño pero a veces puede ser muy útil, vamos a ver un ejemplo.

```
 

package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;

public class Principal {

public static void main(String[] args) {

List<String> listaNombres = new ArrayList<String>();

listaNombres.add("Pedro");
listaNombres.add("Miguel");
listaNombres.add("Ana");
listaNombres.add("Isabel");
listaNombres.add("MariaPilar");

Predicate<String> filtro3 = (nombre) -> nombre.length() <= 3;

listaNombres.stream().filter(filtro3).forEach(nombre ->
```

```
System.out.println(nombre));

System.out.println("************");

Predicate<String> filtro5 = (nombre) -> nombre.length() <= 5;

listaNombres.stream().filter(filtro5).forEach(nombre ->
System.out.println(nombre));

System.out.println("************");

Predicate<String> filtro7 = (nombre) -> nombre.length() <= 7;

listaNombres.stream().filter(filtro7).forEach(nombre ->
System.out.println(nombre));

}
}
```

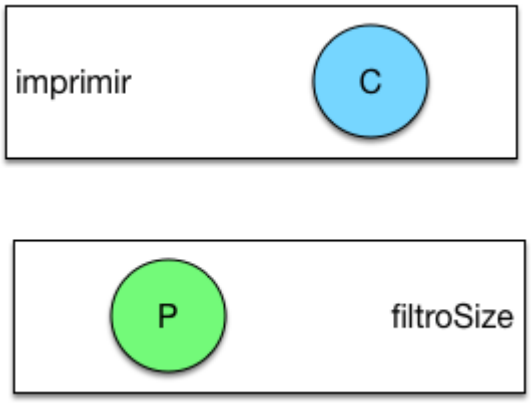
Disponemos de una lista de nombres los cuales imprimimos por pantalla aplicando un Predicado diferente (Functional Interface) . En el primer caso se imprimirán los nombres cuya longitud sea menor o igual a 3 el segundo de 5 y el tercero de 7.

¿Se puede simplificar el código? . La respuesta es sí pero para ello necesitaremos apoyarnos en Java High Order Functions ,vamos a verlo :

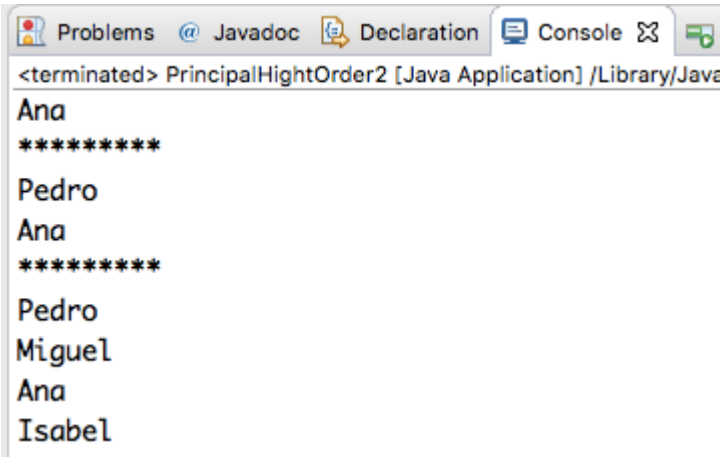
```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;
import java.util.function.Consumer;
import java.util.function.Predicate;

public class PrincipalHightOrder2 {

public static void main(String[] args) {

List<String> listaNombres = new ArrayList<String>();

listaNombres.add("Pedro");
listaNombres.add("Miguel");
```

```
listaNombres.add("Ana");
listaNombres.add("Isabel");
listaNombres.add("MariaPilar");

imprimir (listaNombres,System.out::println,3);
System.out.println("*********");
imprimir (listaNombres,System.out::println,5);
System.out.println("*********");
imprimir (listaNombres,System.out::println,7);

}


public static void imprimir (List<String>listaNombres
,Consumer<String> consumidor, int size) {

listaNombres.stream().filter(filtroSize(size)).forEach(consumidor);
}

public static Predicate<String> filtroSize(final int longitud) {

return texto -> texto.length() <= longitud;
}
}

 
```

En este caso la primera función imprimir recibe como parámetro otra función (Consumer) que será la encargada de imprimir a través de un método de referencia. En segundo lugar la función filtroSize devuelve otra función que es utilizada en el método filter para definir el algoritmo de filtrado en tiempo de ejecución.



El resultado es idéntico solo que en este caso hemos utilizado dos Java High Order Functions que nos han simplificado el código.



Otros artículos relacionados : Java 8 Lambda ,Java Collections y Set