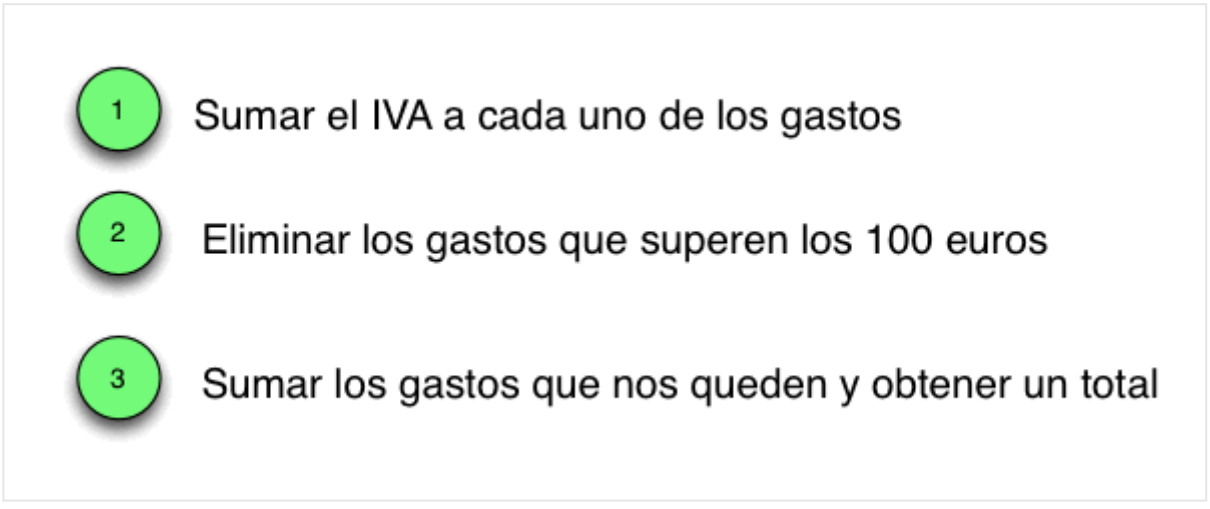
arquitecturajava

Navigation

Programación Funcional, Java 8 Streams

por **Cecilio Álvarez Caules** sobre 28 Agosto, 2014 en **Java Lambda y Streams**, **Java SE**

Cada día la programación funcional está más de moda y Java 8 tiene un buen soporte para ello .Sin embargo muchas veces no tenemos muy claro como aplicarla. Vamos a construir un ejemplo sencillo. Supongamos que tenemos una lista de gastos de viaje diarios y la empresa se hará cargo de todos ellos. Eso sí , si una vez sumado a los gastos el IVA el importe supera los 100 euros la empresa no lo pagará ya que considerará que nos hemos excedido en lo que gastamos en un día. Las operaciones que tendremos que realizar resumiendo son:



Solución Clásica

Bueno vamos a solventar este problema y para ello vamos a construir un Array de Objetos de tipo Gasto y programar la lógica necesaria para que las operaciones nos funcionen correctamente :

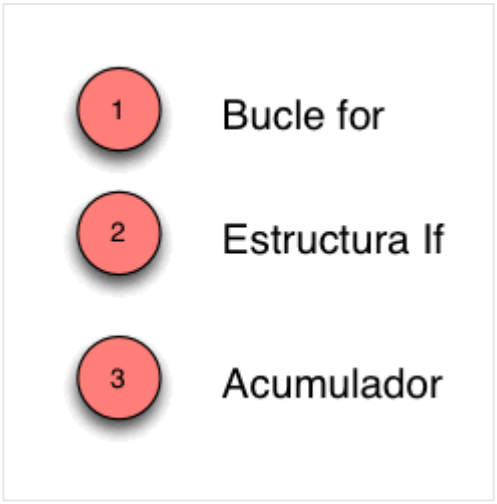
```
1  package com.arquitecturajava.streams;
2
3  import java.util.ArrayList;
4
5  public class Principal {
6
7   public static void main(String[] args) {
8
9   ArrayList<Gasto> lista= new ArrayList<Gasto>();
```

```
10
11    lista.add(new Gasto("A",80));
12    lista.add(new Gasto("B",50));
13    lista.add(new Gasto("C",70));
14    lista.add(new Gasto("D",95));
15
16    double totalPago=0;
17
18    for (Gasto g:lista) {
19
20    if (g.getImporte()*1.21<100) {
21
22    totalPago=totalPago+ g.getImporte()*1.21;
23
24    }
25    }
26
27    System.out.println(totalPago);
28
29    }
30
31  }
```

Revisando el código nos podemos dar cuenta que la lógica de negocio era bastante sencilla de aplicar . Usamos un bucle for, recorremos el array y con una estructura if que chequea si nos pasamos de 100 euros decidimos si acumulamos el importe o no el programa nos imprimirá

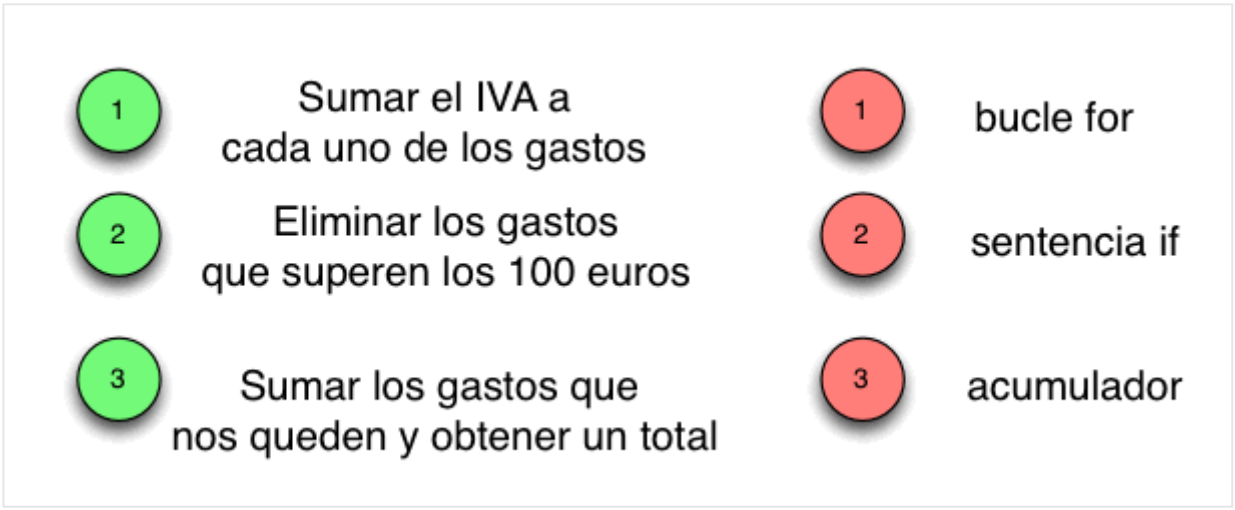
242.0

Hemos realizado las siguientes operaciones :



Programación y Personas

A veces los programadores nos alejamos mucho del pensamiento clásico humano que es mucho más lineal . Es decir es mas del estilo incrementamos a todos el IVA , eliminamos los gastos que pasen de 100 euros ,sumamos el resto. De ahí que aprender a programar nunca haya sido algo “facil” sino mas bien todo lo contrario ya que los conceptos son difíciles de encajar:



La programación funcional nos puede ayudar a acercarnos mas a un pensamiento humano clásico y hacer que los dos enfoques que tenemos encajen mejor.

Java y Streams

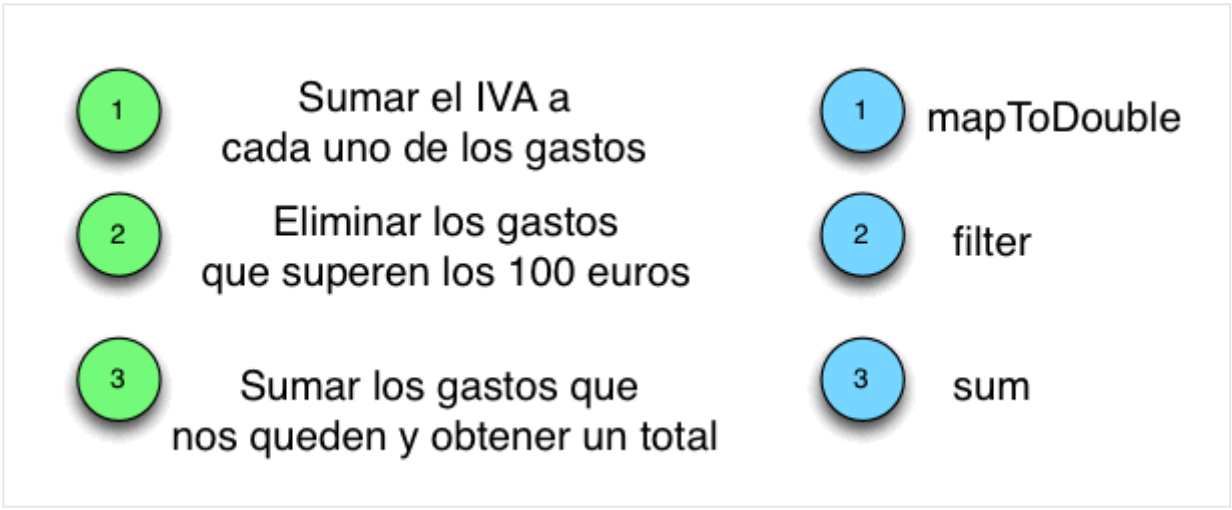
Para poder solventar el problema que tenemos de una forma más amigable vamos a usar el concepto de Java 8 Stream y programación funcional. Un Stream no es ni mas ni menos que un conjunto de funciones que se ejecutan de forma anidada.

```
1  package com.arquitecturajava.streams;
2
3  import java.util.ArrayList;
4
5  public class PrincipalFuncional {
6
7  public static void main(String[] args) {
8
9  ArrayList<Gasto> lista= new ArrayList<Gasto>();
10
11 lista.add(new Gasto("A",80));
12 lista.add(new Gasto("B",50));
13 lista.add(new Gasto("C",70));
14 lista.add(new Gasto("D",95));
15
16 double resultado=lista.stream()
17 .mapToDouble(gasto->gasto.getImporte()*1.21)
18 .filter(gasto->gasto<100)
19 .sum();
20
21 System.out.println(resultado);
22
23 }
24
25 }
26
27  
```

Como podemos ver hemos usado en este nuevo código un stream y lo hemos dividido en tres pasos:

1. Sumamos el IVA a cada uno de nuestros gastos
2. Filtramos aquellos gastos que se pasan del tope
3. Sumamos los que restan

De esta forma habremos resuelto nuestro problema de una forma mucho más funcional y mucho más cercana a como las personas enfocamos.



Poco a poco tenemos que irnos haciendo con este tipo de programación que nos será muy util

Otros Artículos relacionados: Java Lambda ForEach, Introducción a Java Lambda , Java Generics

Related Posts:

Java override HashCode y curiosidades
Java Stream forEach y colecciones
Command Pattern en Java y la gestion de tareas
Java Multiple Inheritance con default methods
Eclipse Pull up , Pull down y refactorings

JavaAPI

‹ JSF, Spring MVC y Java EE 8 JPA Entity Listener ›

22 Responses to *Programación Funcional, Java 8 Streams*

Juan Manuel 29 Agosto, 2014 at 16:51 #

wow !!! Todavía no termino de comprender java 7, cuando me sorprende cada vez mas java 8

Gracias por actualizarnos !! Saludos

RESPONDER

Cecilio Álvarez Caules 29 Agosto, 2014 at 18:58 #

Me alegro te haya sido útil 🙂

RESPONDER

Dulce 21 Octubre, 2016 at 23:44 #

Hola Cecilio, una pregunta este fragmento de código gasto->gasto.getImporte()*1.21 me suena a que es una lambda, pero me causa duda porque no haces uso de interfaz alguna verdad?

RESPONDER

Cecilio Álvarez Caules 23 Octubre, 2016 at 11:41 #

Una expresión lambda puede verse como un microinterface , ya que solo tiene un método

RESPONDER

Mario 13 Diciembre, 2016 at 0:34 #

Hola Cecilio, tengo una duda con esto de los Stream y las expresiones lmbda. Tengo una lista de Strings, a la cual le tengo que sustituir el último carácter del último elemento de la lista, como se podría hacer?? Es que no encuentro la manera usando lista.stream()….. a ver si me puedes echar una mano. Gracias.

Cecilio Álvarez Caules 13 Diciembre, 2016 at 7:34 #

mira esto 🙂

Juan Ramón 7 Septiembre, 2014 at 17:25 #

¿Esto no es mas que incorporar funciones de las librerías de guava no? porque con Iterables.filter y luego aplicando una function pienso que seria lo mismo.

Pd: Explica usted de maravilla, 0 paja, imagenes/esquemas grandes y poco texto.

Un saludo y enhorabuena por el blog

RESPONDER

Cecilio Álvarez Caules 7 Septiembre, 2014 at 22:26 #

Es mas complejo y mas flexible ya que es parte del core

RESPONDER

Cecilio Álvarez Caules 8 Septiembre, 2014 at 8:41 #

no conocía las librerías guava, gracias por el aporte

RESPONDER

luis 31 Enero, 2017 at 21:07 #

Sencillo pero eficaz. Saludos!!

RESPONDER

Cecilio Álvarez Caules 1 Febrero, 2017 at 12:15 #

gracias 🙂

RESPONDER

Trackbacks/Pingbacks

Expresiones Lambda y ejecución en paralelo - Arquitectura Java - 9 Septiembre, 2015
[…] Otros Artículos relacionados: Expresiones Lambda ,Lambda ForEach, Streams […]
Java 8 Date Time API - Arquitectura Java - 9 Septiembre, 2015
[…] Otros artículos relacionados: ForEach vs Iterator , Expresiones Lambda ,JavaStreams […]
Java 8 Stream y workflows - Arquitectura Java - 10 Septiembre, 2015
[…] artículos relacionados: Programación funcional y Streams , Java 8 default Methods , Java 8 […]
Novedades de Java 8 Collections y Listas - Arquitectura Java - 29 Octubre, 2015
[…] artículos relacionados: Java Lambda , Java Streams , Java Streams WorkFlows , Oracle Java […]
Utilizando Java 8 Predicate - Arquitectura Java - 25 Mayo, 2016
[…] Otros artículos relacionados: Java Lambda Java Streams […]
Scala, Java y la programación funcional - 2 Agosto, 2016
[…] Programación Funcional en Java: Ejemplo de programación funcional implementado en Java […]
El concepto de Java infinite Stream - Arquitectura Java - 13 Octubre, 2016
[…] artículos relacionados : Java Lambda , Java Streams , Stream y […]

Java Executor Service y Threading - Arquitectura Java - 28 Octubre, 2016
[…] artículos relacionados : Java Streams , Java Lambda Java […]
Java Stream Collectors y su uso - Arquitectura Java - 30 Diciembre, 2016
[…] artículos relacionados : Programación Funcional, Java 8 Streams , Java 8 Lambda y forEach (II) , El concepto de Java infinite […]
El concepto de Java 8 reference method - Arquitectura Java - 7 Enero, 2017
[…] artículos relacionados: Introducción a Lambda , Java Streams , Java Lambda WorkFlows , Oracle Java […]
Java 8 Optional y NullPointerExceptions - Arquitectura Java - 9 Febrero, 2017
[…] artículos relacionados: Java Lambda, Programación Funcional, Java 8 Streams Java Functional […]

Deja un comentario

Name (required)

Email (will not be published) (required)

Website

Submit Comment

Buscar

Search...

Translate:

Seleccionar idioma

Con la tecnología de Google **Traductor de Google**

POPULAR


Command Pattern en Java y la gestion de tareas
26 MAYO, 2017


JPA Composite Key y business objects
28 ABRIL, 2017


El concepto Java Reflection y como utilizarlo
21 MARZO, 2017


Desarrollo Web con React.js , mi nuevo libro
7 ABRIL, 2017


Java LinQ con JinQ y Java Persistence API
17 MAYO, 2017


JPA Database Schema y automatización
31 MAYO, 2017


Java 8 interface static methods y reutilizacion
20 ABRIL, 2017


Java value vs reference y sus curiosidades
17 ABRIL, 2017


Java Boxing y sus curiosidades
24 MARZO, 2017


Java Stream forEach y colecciones
15 JUNIO, 2017

Redes Sociales

Síguenos en Linkedin y Twitter o subscríbete al RSS.

E-mail Submit

Contacto

contacto@arquitecturajava.com