

Blog sobre Java EE

[JAVA SE](#)[JAVA SE +](#)[SPRING](#)[JAVA EE](#)[JAVASCRIPT](#)[FRAMEWORKS JS](#)[ARQUITECTURA](#)[MIS LIBROS](#)

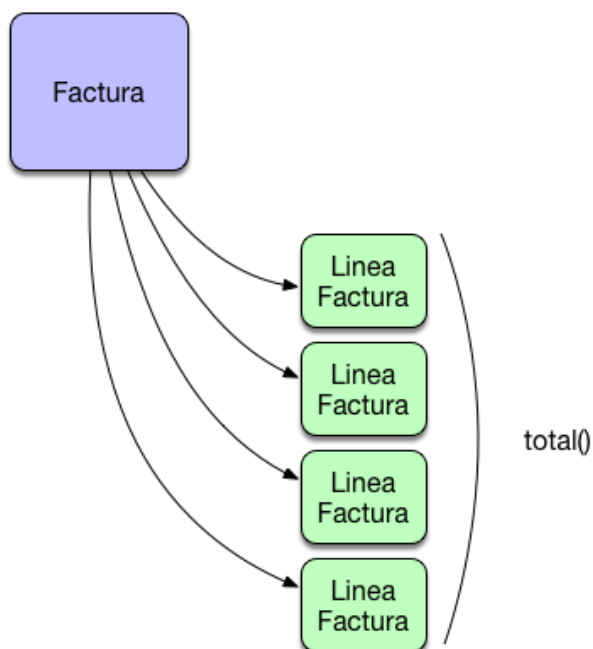
Estás aquí: [Inicio/Sin categoría](#)/Java Stream Sum y Business Objects

Java Stream Sum y Business Objects

por [Cecilio Álvarez Caules](#) — [Deja un comentario](#)



El uso de **Java Stream sum** nos lo encontramos muchas veces en nuestro código de Java para sumar un conjunto de elementos. Sin embargo a veces nos olvidamos que los Streams han venido para ser usados en la plataforma Java en general y se pueden aplicar a muchas situaciones. Vamos a ver un ejemplo en el cual los apliquemos a **objetos de negocio**. Para ello vamos a partir de los conceptos de Factura y Linea de Factura.



Estos conceptos están relacionados y es muy habitual que la clase Factura contenga **un método total que nos calcule el importe total de todas las lineas**. Veamos el código:

```
1 package com.arquitecturajava.ejemplo2;  
2  
3 import java.util.ArrayList;
```

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

```
14 public void addLinea(LineaFactura lf) {
15     lineas.add(lf);
16 }
17 public int getNumero() {
18     return numero;
19 }
20
21 public void setNumero(int numero) {
22     this.numero = numero;
23 }
24
25 public String getConcepto() {
26     return concepto;
27 }
28
29 public void setConcepto(String concepto) {
30     this.concepto = concepto;
31 }
32
33 public List<LineaFactura> getLineas() {
34     return lineas;
35 }
36
37 public void setLineas(List<LineaFactura> lineas) {
38     this.lineas = lineas;
39 }
40
41 public double total() {
42     double total = 0;
43     for (LineaFactura l : lineas) {
44         total += l.getImporte();
45     }
46     return total;
47 }
48
49 }
```

```
1 package com.arquitecturajava.ejemplo1;
2
3 public class LineaFactura {
4
5     private int numero;
6     private String concepto;
7     private double importe;
8
9     public int getNumero() {
10         return numero;
11     }
12     public void setNumero(int numero) {
13         this.numero = numero;
14     }
15     public String getConcepto() {
```

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

```
26 }
27 public LineaFactura(int numero, String concepto, double importe) {
28     super();
29     this.numero = numero;
30     this.concepto = concepto;
31     this.importe = importe;
32 }
33 }
```

Java Stream Sum

El código es correcto. ¿**Ahora bien podemos construir algo más flexible?** En muchas ocasiones cuando realizamos una compra y obtenemos la factura nos gusta mirar **en que conceptos nos hemos gastado más dinero**. Así pues podríamos usar **Java Overload** y sobrecargar el método total de la Factura para calcular el total apoyándonos **en un Predicate**.

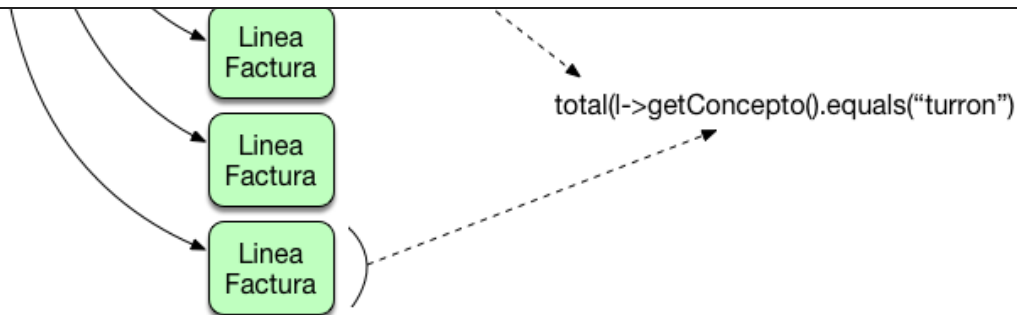
```
1 public double total() {
2
3     double total = 0;
4     for (LineaFactura l : lineas) {
5
6         total += l.getImporte();
7     }
8     return total;
9 }
10
11 public double total(Predicate<LineaFactura> plinea) {
12
13     return lineas.stream().filter(plinea).mapToDouble(l -> l.getImporte
14 }
```

Así ganamos en flexibilidad:

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)

ACEPTAR

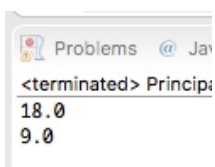


Veamos un programa main con el cual obtener tanto el total como **el total de un producto concreto**.

```

1  package com.arquitecturajava.ejemplo2;
2
3  public class Principal {
4
5      public static void main(String[] args) {
6
7
8          Factura f= new Factura();
9          f.setNumero(1);
10         f.setConcepto("miscompras");
11         f.addLinea(new LineaFactura(1, "turron", 3));
12         f.addLinea(new LineaFactura(1, "turron", 3));
13         f.addLinea(new LineaFactura(1, "turron", 3));
14         f.addLinea(new LineaFactura(1, "jamon", 5));
15         f.addLinea(new LineaFactura(1, "jamon", 4));
16
17         System.out.println(f.total());
18         System.out.println(f.total(l->l.getConcepto().equals("turron")));
19
20     }
21 }
22
23 
```

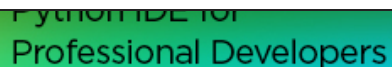
Acabamos de calcular el gasto total y el que hemos hecho en turrón y vemos el resultado en la consola.



Otros artículos relacionados:

1. Java Stream map y estadísticas

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

Archivada en: [Java Lambda y Streams](#), [Sin categoría](#)

Etiquetada con: [Java8Tips](#)

Leave a Reply

Be the First to Comment!

Notify of



Start the discussion

BUSCAR

Buscar en este sitio ...

CUPÓN DESCUENTO NAVIDAD

Aprovecha mi cupón de Navidades para obtener un 50% de descuento [en mis cursos de Java](#) 😊

Cupón:NAVIDAD20

Mis Cursos de Java Gratuitos

[Java Herencia](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)

ACEPTAR

Servlets



POPULAR

- Maven Parent POM y uso de librerías
- Java Interfaces y el concepto de simplicidad
- Java Generic Repository y JPA
- Spring GetMapping ,PostMapping etc
- Spring 5 Hello World

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

CONTACTO

contacto@arquitecturajava.com

LO MAS LEIDO

[¿Qué es Spring Boot?](#)[Usando Java Session en aplicaciones web](#)[Java Constructores this\(\) y super\(\)](#)[Java Iterator vs ForEach](#)[Introducción a Servicios REST](#)[¿Cuales son las certificaciones Java?](#)[¿Qué es Gradle?](#)[Ejemplo de JPA , Introducción \(I\)](#)[Ejemplo de Java Singleton \(Patrones y ClassLoaders\)](#)[Usando el patron factory](#)[REST JSON y Java](#)[Uso de Java Generics \(I\)](#)[Java Override y encapsulación](#)[Mis Libros](#)[Spring REST Client con RestTemplates](#)[Comparando java == vs equals](#)[¿Qué es un Microservicio?](#)[Spring MVC Configuración \(I\)](#)[Java Interfaces y el concepto de simplicidad](#)[Java Stream Sum y Business Objects](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)
