

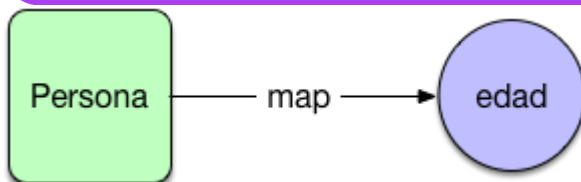
Tabla de Contenidos



- [Java Stream Map](#)
- [Java Stream Map vs MapInt y MapDouble](#)
- [Java Stream map statistics](#)
- [Otros artículos relacionados:](#)

El uso de Java Stream map es una de las operaciones más comunes cuando trabajamos con un flujo de Streams . El método map nos permite realizar una transformación rápida de los datos y muy directa sobre el flujo original.

CURSO JAVA 8 GRATIS APUNTATE!!

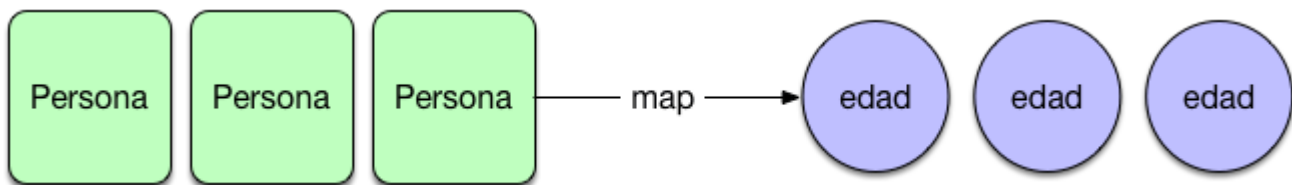


Vamos a ver algunos ejemplos que nos ayuden a clarificar como se utiliza Java Stream map. Para ello nos vamos a apoyar en una lista de Personas .

```
Persona p1 = new Persona("pedro", 20, "perez");
Persona p2 = new Persona("juan", 25, "perez");
Persona p3 = new Persona("ana", 30, "perez");
List < Persona > lista = new ArrayList < Persona > ();
lista.add(p1);
lista.add(p2);
lista.add(p3);
```

Java Stream Map

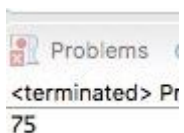
La primera operación que vamos a realizar es convertir nuestra lista de personas a una lista de números enteros.



Realizada esa operación podemos usar el método `reduce` para sumar todas las edades e imprimirlas en la consola.

```
lista
.stream()
.map(Persona::getEdad)
.reduce((a, b) -> a + b)
.ifPresent(System.out::println);
```

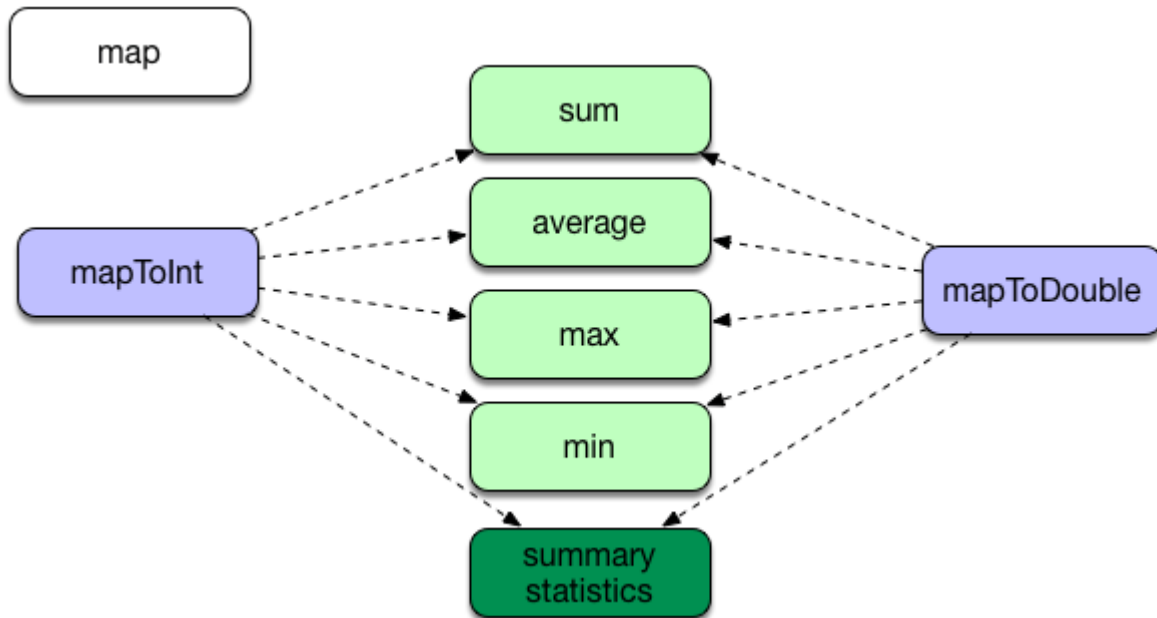
El resultado es :



Sin embargo esta aunque este una solución válida no es la mejor solución

Java Stream Map vs MapInt y MapDouble

El método `map` viene con dos métodos adicionales orientados a trabajar con datos numéricos. Estos métodos son `mapToInt` y `mapToDouble`. Si cambiamos nuestro método de `map` a `mapToInt` o `mapToDouble` se nos abrirá la posibilidad de acceder a métodos adicionales muy orientados a estadísticas.

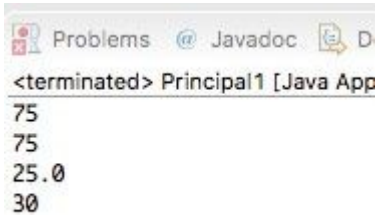


Veamos su código:

Podemos ver el resultado por la consola.

```
int total = lista.stream().mapToInt(Persona::getEdad).sum();
System.out.println(total);
lista.stream().mapToDouble(Persona::getEdad).average().ifPresent(System.out::println);
lista.stream().mapToInt(Persona::getEdad).max().ifPresent(System.out::println);
lista.stream().mapToInt(Persona::getEdad).min();
```

**TODOS LOS CURSOS
PROFESIONALES
25\$/MES
APUNTATE!!**



```
<terminated> Principal1 [Java App]
75
75
25.0
30
```

Es una forma más cómoda de trabajar con valores numéricos ya que las transformaciones directamente nos lo pasan a tipos fundamentales.

Java Stream map statistics

Hay situaciones en las que podemos querer acceder de forma directa a todas las estadísticas. Los streams numéricos soportan el método de `summaryStatistics` que nos permite acceder directamente a todos los valores.

```
DoubleSummaryStatistics estadisticas = lista.stream()
    .mapToDouble(Persona::getEdad)
    .summaryStatistics();
System.out.println(estadisticas.getAverage());
```

CURSO SPRING BOOT
GRATIS
APUNTATE!!

Otros artículos relacionados:

1. [Java Stream Filter y Predicates](#)
2. [Java Stream String y Java 8](#)

3. [Java Stream Collectors y su uso](#)
4. [Oracle IntStream](#)
5. [Java Lambda reduce y wrappers](#)
6. [Java List Directory en Java 8 con Streams](#)
7. [El concepto de Java constructor reference](#)