

CURSO Java Herencia

GRATIS

APUNTATE!!

Cuanto más usamos los streams más usamos los diferentes Java Stream Collectors para transformar estos. Vamos a ver cuales son las diferentes opciones que Java soporta para transformar un stream a otro tipo de estructura. Para ello nos vamos a crear la clase Libro y usarla en diferentes ejemplos.

```
package com.arquitecturajava;

public class Libro {

    private String titulo;
    private int paginas;

    public Libro(String titulo, int paginas) {
        super();
        this.titulo = titulo;
        this.paginas = paginas;
    }
    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
    public int getPaginas() {
```

```

        return paginas;
    }
    public void setPaginas(int paginas) {
        this.paginas = paginas;
    }
}

```

```
package com.arquitecturajava;
```

```

import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Optional;
import java.util.stream.Collectors;
import java.util.stream.Stream;

```

```

public class Principal {

    public static void main(String[] args) {

        Libro l1= new Libro ("El señor de los anillos",1000);
        Libro l2= new Libro ("La fundacion",500);
        Libro l3= new Libro ("El caliz de fuego",600);

    }
}

```

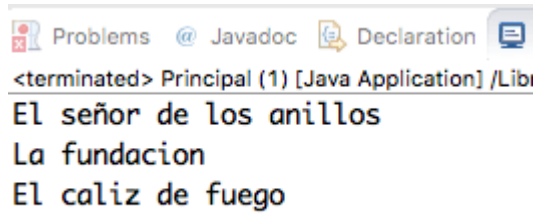
Acabamos de instanciar tres objetos ,vamos a añadir todos a un stream y recorrerlo utilizando el método `forEach`.

```

Stream<Libro> st = Stream.of(l1, l2, l3);
st.forEach((l) -> System.out.println(l.getTitulo()));

```

El resultado se imprimirá por la consola:



```
<terminated> Principal (1) [Java Application] /Libr
El señor de los anillos
La fundacion
El caliz de fuego
```

En muchas ocasiones no queremos simplemente imprimir un resultado sino que lo que queremos es convertir el Stream a un tipo de dato para su posterior gestión .



Vamos a ver algunas opciones, una de las más sencillas es convertirlo a un array con el método `toArray()` :

```
Libro[] arrayLibro= st.toArray(Libro[]::new);

for(int i=0;i<arrayLibro.length;i++) {

    System.out.println(arrayLibro[i].getPaginas());
}
```

Este código obtendrá un array de objetos a partir del Stream y lo imprimirá por la consola:

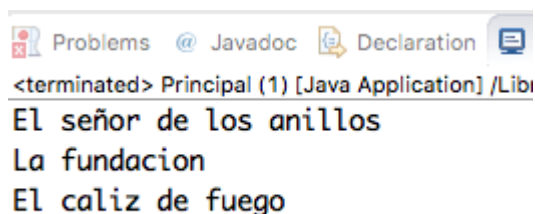
```
1000  
500  
600
```

Java Stream Collectors

De igual forma que podemos convertir el Stream a un array, podemos usar los Java Stream Collectors y convertir nuestro stream a una Lista o Conjunto , utilizando la clase Collectors y su método `toList()` o `toSet()`.

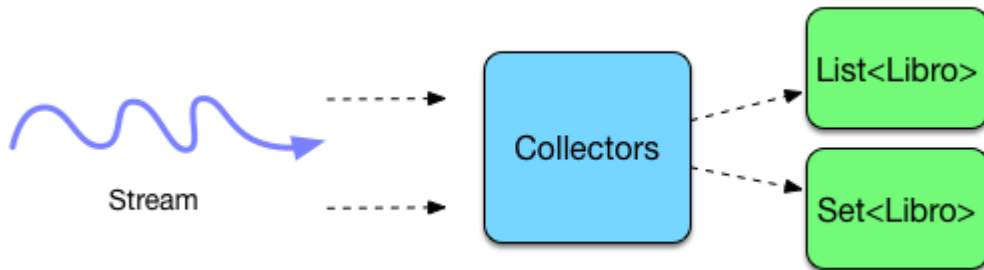
```
List<Libro> lista= st.collect(Collectors.toList());  
  
for (Libro l:lista) {  
  
    System.out.println(l.getTitulo());  
}  
  
Set<Libro> lista = st.collect(Collectors.toSet());  
  
for (Libro l : lista) {  
  
    System.out.println(l.getTitulo());  
}
```

El resultado es idéntico:



```
<terminated> Principal (1) [Java Application] /Libro  
El señor de los anillos  
La fundacion  
El caliz de fuego
```

Acabamos de usar dos de los métodos fundamentales de Collectors para generar Sets y List a partir de un Stream.



Los Java Stream collectors nos permiten realizar operaciones más complejas con poco código . Por ejemplo podemos obligar a que todos los títulos de los Libros se impriman en una única línea usando el método `joining` de la clase `Collectors`.

```
String resultado = st.map((l) ->
l.getTitulo()).collect(Collectors.joining(","));
System.out.println(resultado);
```

La consola mostrará:

El señor de los anillos,La fundacion,El caliz de fuego

Podemos complicarlo más y sumar todas las páginas de los libros utilizando el método `reducing` de `Collectors`.

```
Optiona<Integer> resultado3 = st.map((l) ->
l.getPaginas()).collect(Collectors.reducing(Integer::sum));
```

```
System.out.println(resultado3.get());
```

Vemos en consola el sumatorio de todas las páginas de todos los libros:

2100

Hay muchas opciones pero la clase `Collectors` es una de las indispensables a la hora de trabajar con Streams y transformarlos.

CURSO Java Herencia
GRATIS
APUNTATE!!

Otros artículos relacionados :

- [Programación Funcional, Java 8 Streams](#)
- [Java 8 Lambda y forEach \(II\)](#)
- [El concepto de Java infinite Stream](#)