El concepto de  Java Stream file , cada día se usa más ya que Java 8 nos permite simplificar sobremanera la lectura de ficheros. Algo que en Java clásico siempre ha sido bastante complejo . Vamos a ver un ejemplo sencillo de como leer un fichero linea a linea. Para ello partiremos del siguiente fichero.

1,pc,1000

2,mac,2000

3,android,300

4,ios,900

Vamos a construir una clase Java que se denomine Ordenador y almacene los datos que se encuentran en el fichero.

```
package com.arquitecturajava;

public class Ordenador {

	private int numero;
	private String sistema;
	private double precio;
	public int getNumero() {
		return numero;
	}
	public void setNumero(int numero) {
		this.numero = numero;
	}
	public String getSistema() {
		return sistema;
```

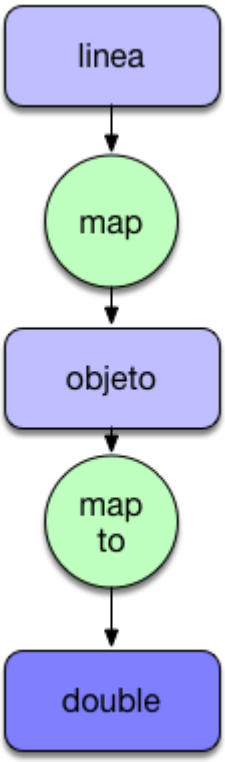
```
	}
	public void setSistema(String sistema) {
		this.sistema = sistema;
	}
	public double getPrecio() {
		return precio;
	}
	public void setPrecio(double precio) {
		this.precio = precio;
	}
	public Ordenador(int numero, String sistema, double precio) {
		super();
		this.numero = numero;
		this.sistema = sistema;
		this.precio = precio;
	}
	public static Ordenador buildFromArray(String[] elementos) {
		return new Ordenador
(Integer.parseInt(elementos[0]),elementos[1],Double.parseDouble(elemen
tos[2]));
	}
}
```

Veamos el programa principal:

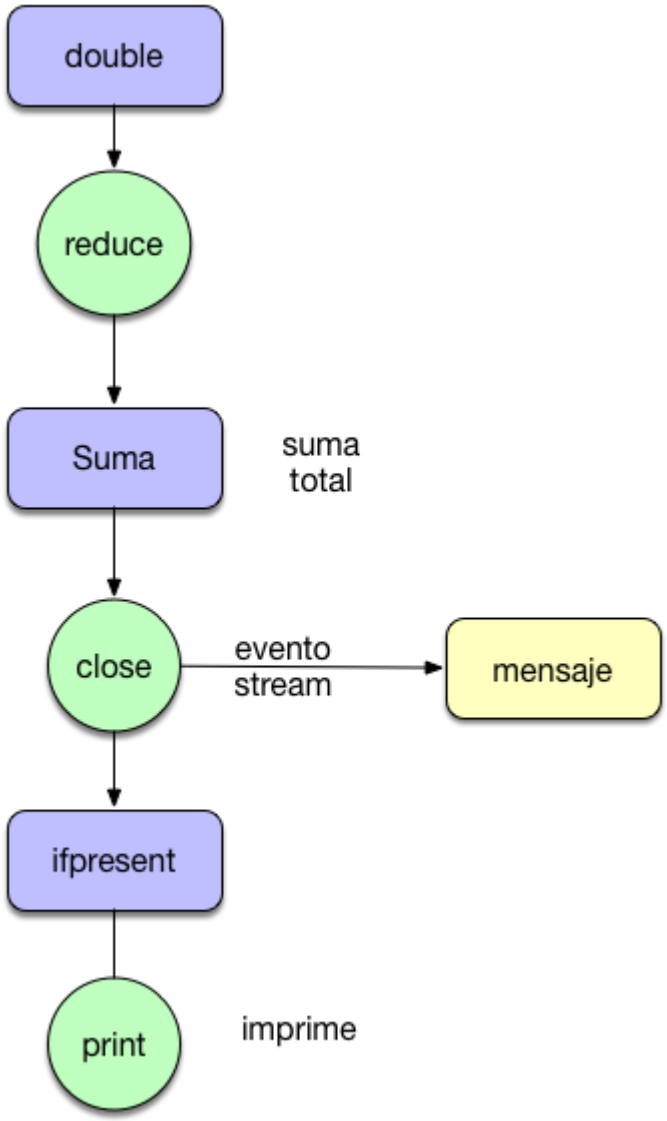
```
package com.arquitecturajava;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
```

```
import java.util.stream.Stream;

public class Principal {

	public static void main(String[] args) {

		String ficheros = "datos.txt";

		try (Stream<String> stream =
Files.lines(Paths.get(ficheros))) {

			stream.map(linea -> linea.split(","))
				.map(Ordenador::buildFromArray)
				.mapToDouble(o -> o.getPrecio())
				.onClose(() ->
System.out.println("termino"))
				.reduce(Double::sum)
				.ifPresent(System.out::println);

		} catch (IOException e) {
			e.printStackTrace();
		}
	}

}
```

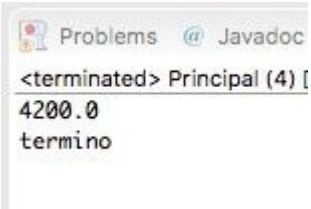
Acabamos de usar el método lines de la clase Files para construir un Stream. Este Stream nos permite leer linea a linea el fichero de texto. Una vez accedemos a cada linea realizamos dos operaciones de mapeo y convertimos cada linea en un objeto y luego en un número.




Hemos esto dos pasos , el siguiente paso es realizar una reducción y sumar todos los elementos.




Al ejecutar el Stream sobre un bloque de try with resources el Stream cerrará los recursos asociados a ficheros invocando al método close. Cuando el método close se invoque asociamos una expresión lambda al momento de su ejecución.



El resultado nos muestra la suma y el mensaje de “terminó”. Acabamos de construir un ejemplo de Java Stream File.

Otros artículos relacionados

1. Java Stream map y estadísticas
2. Java Stream String y Java 8
3. Java Stream forEach y colecciones
4. Java Stream Collectors y su uso
5. Java Stream Oracle