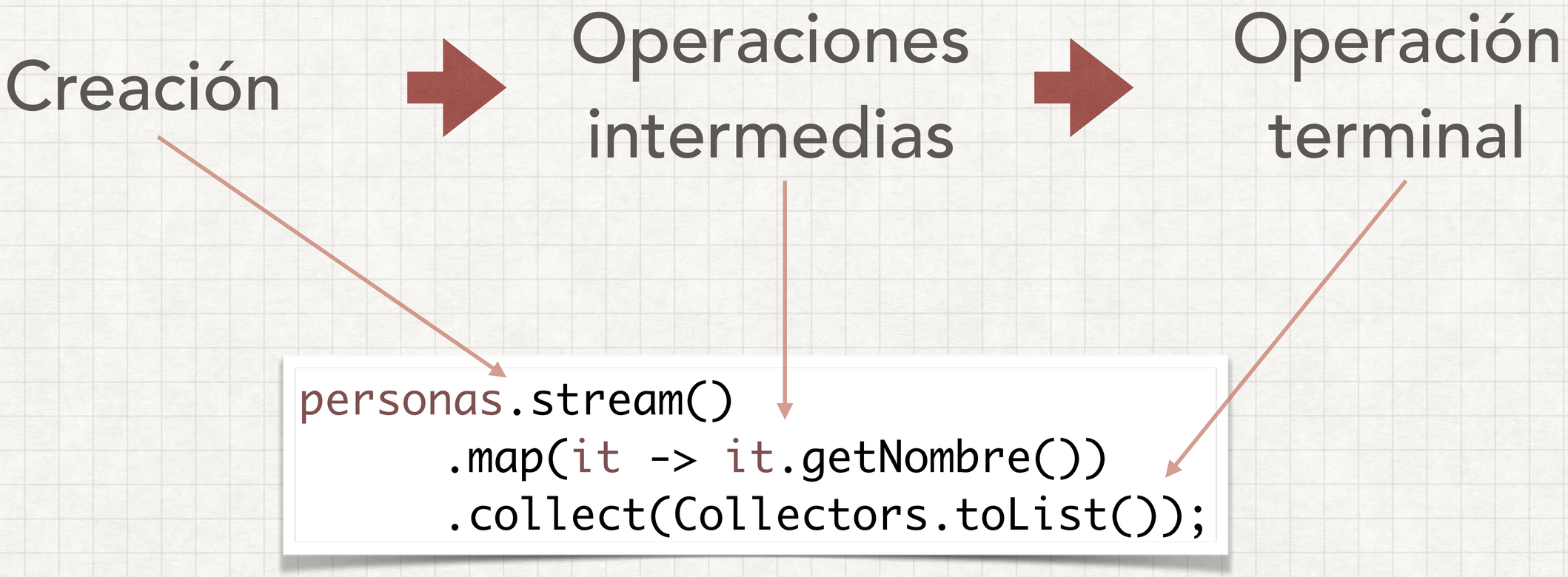
STREAM PIPELINE



Stream pipelines

Creación

Collection stream()

```
List<Persona> personas = Arrays.asList(juan, antonia);
Stream<Persona> s1 = personas.stream();
```

Arrays.asStream()

```
Persona[] personas = {juan, antonia};
Stream<Persona> s2 = Arrays.stream(personas);
```

Utilidades de stream

```
Stream<Persona> s3 = Stream.of(juan, antonia);
```

otras APIs

```
Stream<String> s4 = Files.lines("fichero.txt");
```

Operaciones intermedias principales

- Convertir
- Filtrar
- Ordenar

- Convertir
- Filtrar
- Ordenar

```
Stream<String> nombres = personas.map(it -> it.getNombre());
```

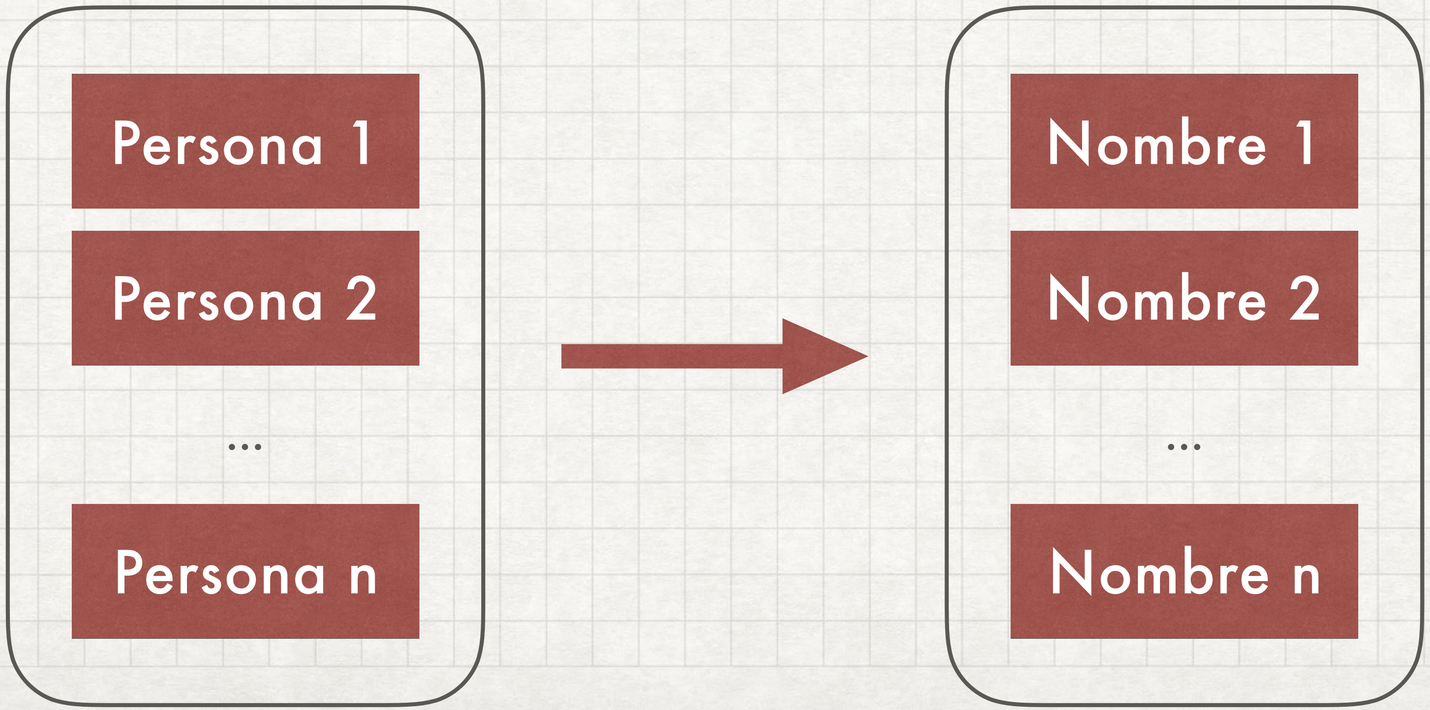
String Persona f : Persona → String

Function<Persona, Nombre>

Function<? super Persona, ? extends String>

- Convertir
- Filtrar
- Ordenar

```
Stream<String> nombres = personas.map(it -> it.getNombre());
```



- Mismo número
- Mismo orden
- Distinto tipo

- Convertir
- Filtrar
- Ordenar

```
Stream<String> nombresA = nombres.filter(it -> it.startsWith("A"));
```

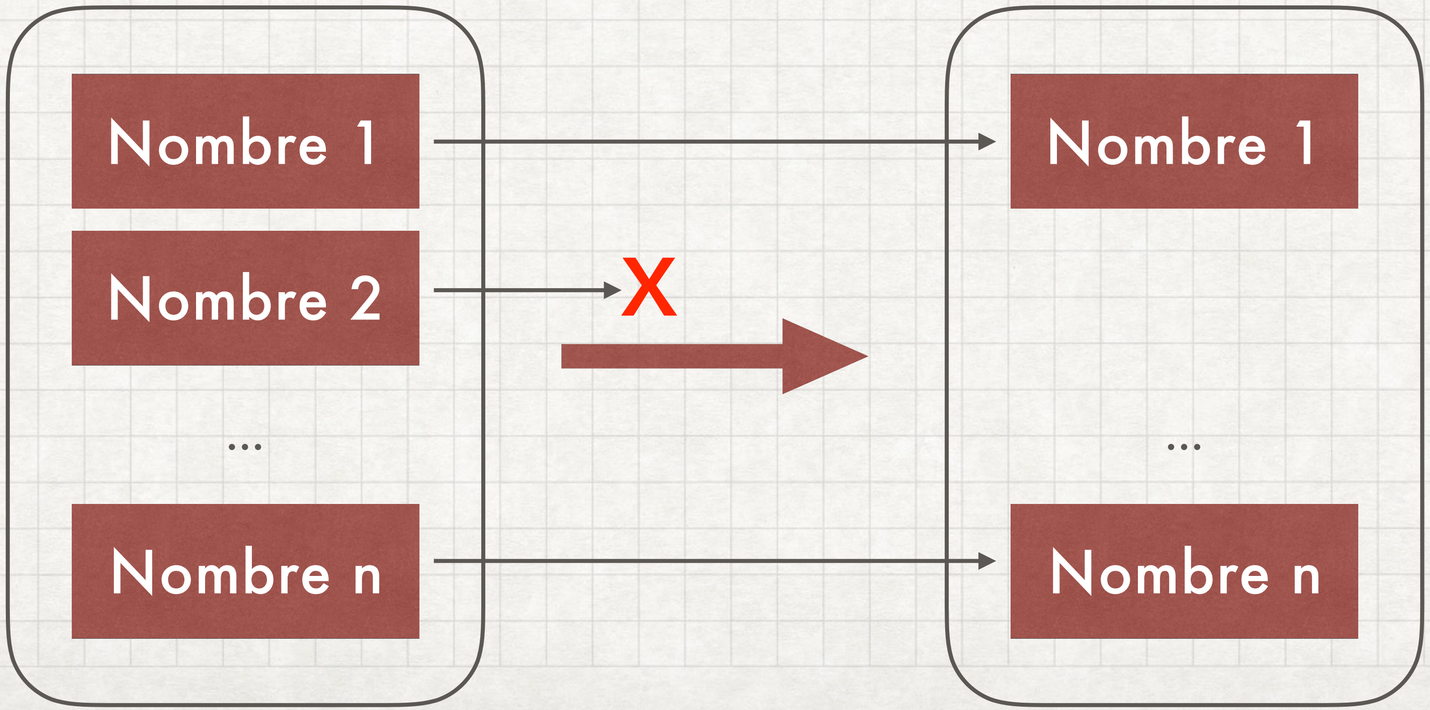
f : String → boolean

Predicate<String>

Predicate<? super String>

- Convertir
- Filtrar
- Ordenar

```
Stream<String> nombresA = nombres.filter(it -> it.startsWith("A"));
```



- Distinto número
- Mismo orden
- Mismo tipo

- Convertir
- Filtrar
- Ordenar

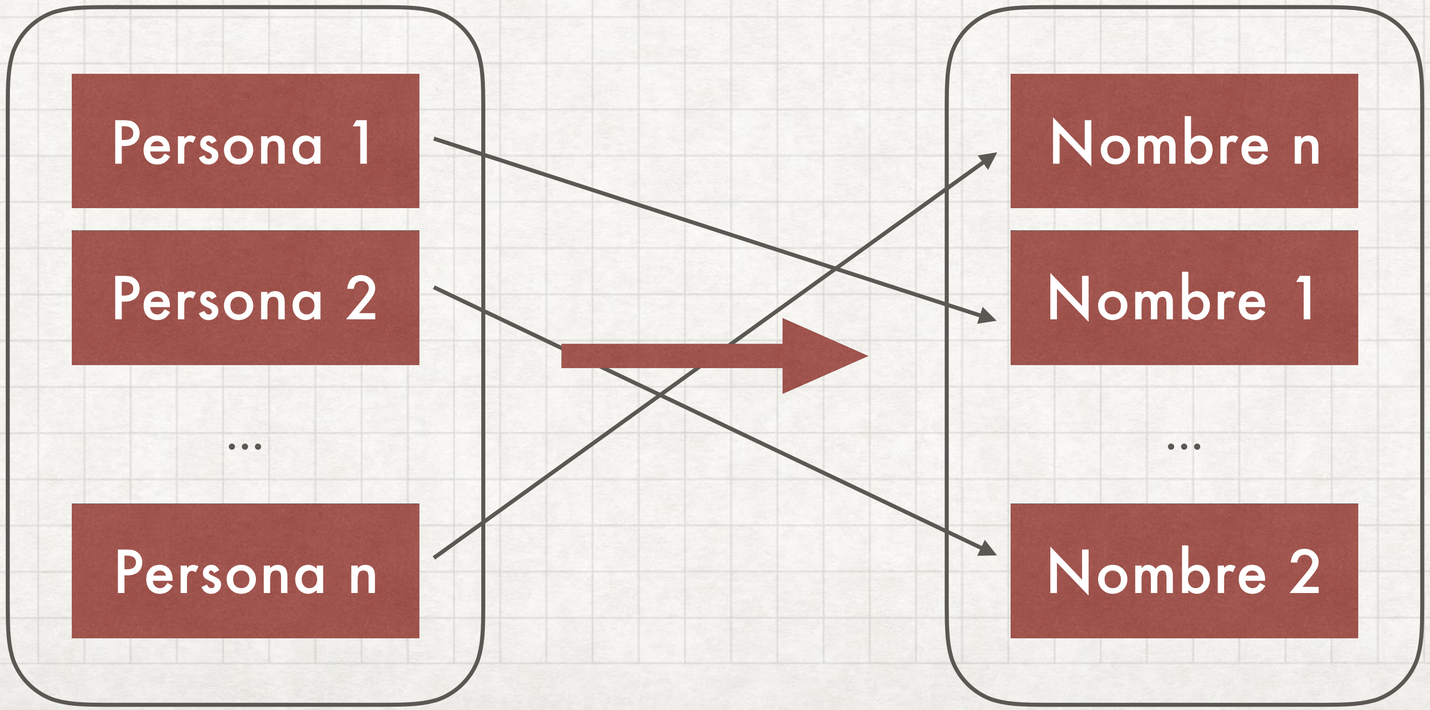
```
nombres.sorted((o1, o2) -> o2.length() - o1.length())
```

Comparator<? super String>

```
nombres.sorted()
```

- Convertir
- Filtrar
- Ordenar

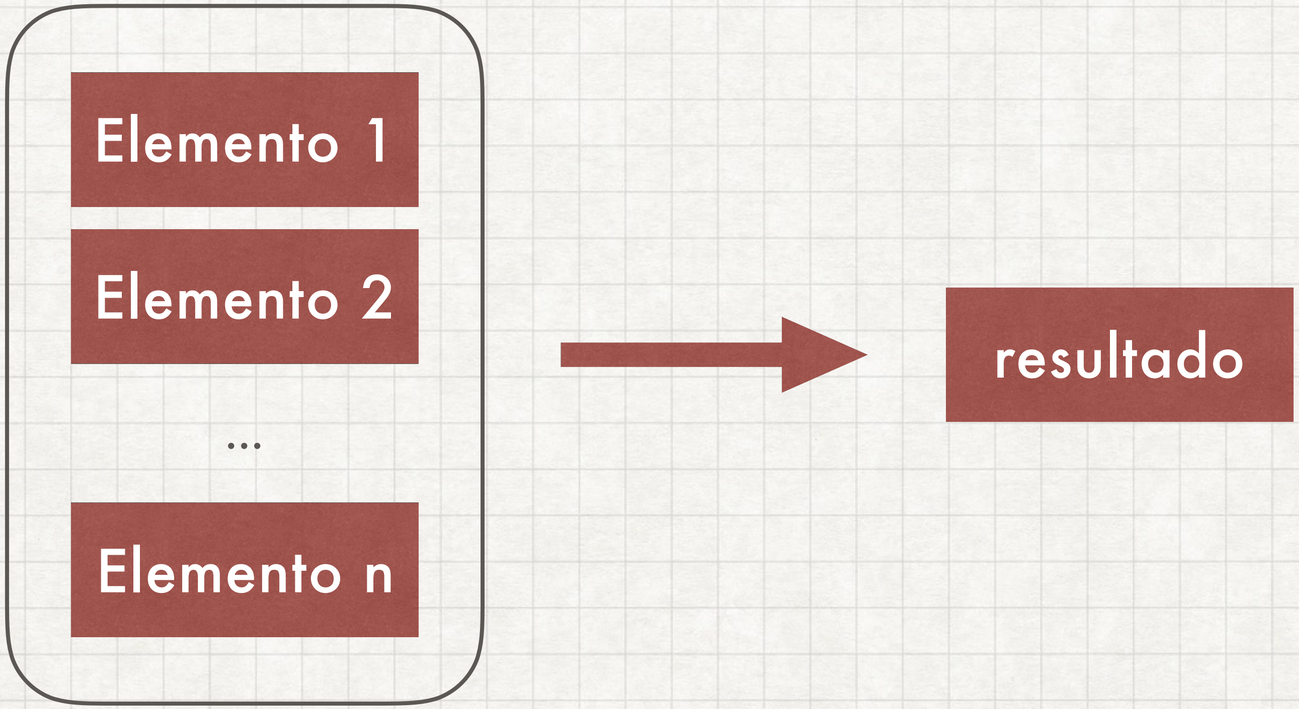
```
nombres.sorted((o1, o2) -> o2.length() - o1.length())
```



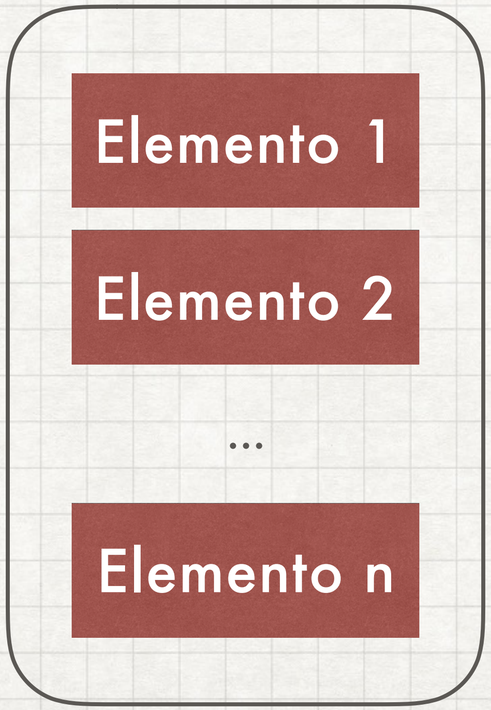
- Mismo número
- Distinto orden
- Mismo tipo

Operación terminal

reduce



reduce



"ingredientes"

Elemento identidad

f : a, b → c

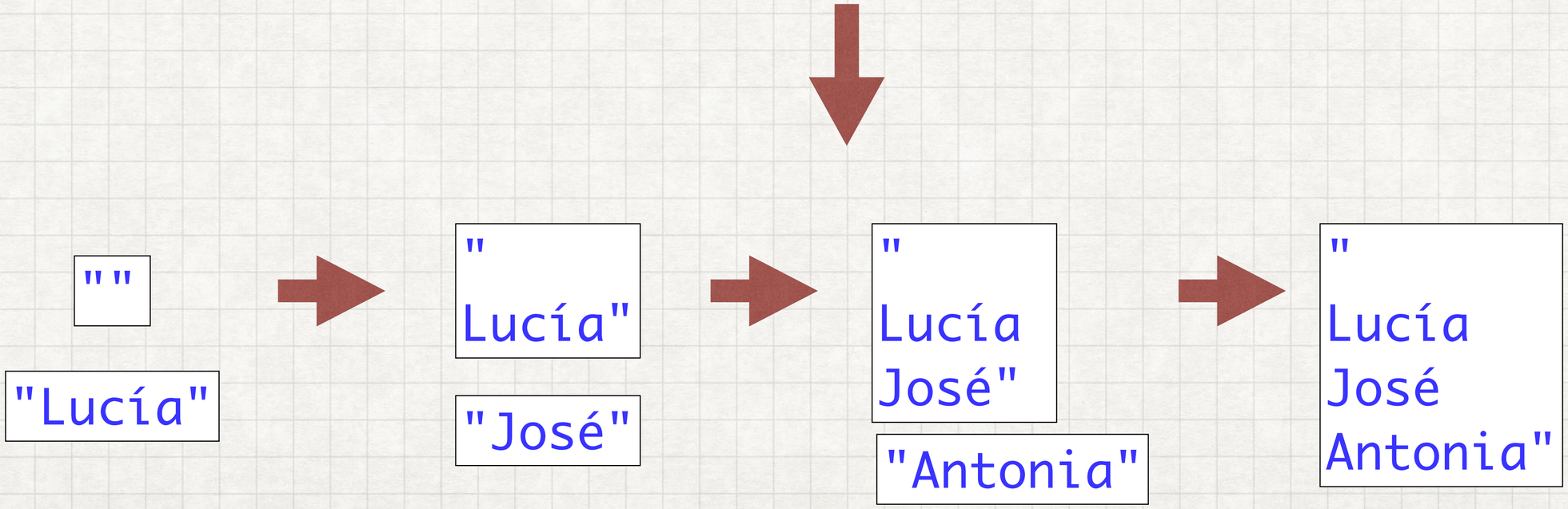
ejemplo:
listado de elementos

```
""
```

```
(a, b) -> a + "\n" + b
```

reduce

```
String todos =  Stream.of("Lucía", "José", "Antonia")
                      .reduce("", (a, b) -> a + "\n" + b);
```



stream pipeline

Creación

```
String todos =  listaPersonas.stream()
                .map(it -> it.getNombre())
                .sorted( (o1, o2) -> o1.length() - o2.length())
                .map( it -> "'" + it + "'")
                .reduce("", (a, b) -> a + "\n" + b);
```

Operaciones intermedias

Operación terminal