

streams

Creación de listas

```
List<String> arrayListDeStrings = new ArrayList<String>();  
arrayListDeStrings.add("Luciano");  
arrayListDeStrings.add("Florecia");  
arrayListDeStrings.add("Victoria");
```

Camino corto con el mismo resultado

```
List<String> arrayListDeStrings = Arrays.asList("Luciano", "Florecia", "Victoria");
```

- Escribir utilizando for
 - anyMatch (List<String>, String inicio)
 - allMatch (List<String>, String inicio)
 - filter (List<String>, String inicio)
 - printAll (List<String>)

Stream anyMatch, allMatch, noneMatch

```
List<String> res = new ArrayList<String>();
for (String s : nombres) {
    if (s.startsWith("1")) {
        res.add(s);
    }
}
```



Streams

A sequence of elements supporting aggregate operations.

```
List<String> arrayListDeStrings = Arrays.asList("Luciano", "Florencia", "Victoria");  
Stream<String> stream = arrayListDeStrings.stream();  
stream.
```

- allMatch(Predicate<? super String> predicate) : boolean - Stream
- anyMatch(Predicate<? super String> predicate) : boolean - Stream
- close() : void - BaseStream
- collect(Collector<? super String,A,R> collector) : R - Stream
- collect(Supplier<R> supplier, BiConsumer<R,? super String> accumulator, BiConsumer<R,? super String> finisher) : R - Stream
- count() : long - Stream
- distinct() : Stream<String> - Stream
- dropWhile(Predicate<? super String> predicate) : Stream<String> - Stream
- equals(Object obj) : boolean - Object
- filter(Predicate<? super String> predicate) : Stream<String> - Stream
- findAny() : Optional<String> - Stream
- findFirst() : Optional<String> - Stream

Stream foreach

```
List<String> arrayListDeStrings = Arrays.asList("Luciano", "Florencia", "Victoria");  
Stream<String> stream = arrayListDeStrings.stream();  
stream.forEach(s -> System.out.println("Hola " + s));
```

Stream anyMatch, allMatch, noneMatch

¿Cómo hago en Java para evaluar si todos los nombres empiezan con L?

Stream anyMatch, allMatch, noneMatch

```
List<String> arrayListDeStrings = Arrays.asList("Luciano", "Florencia", "Victoria");  
Stream<String> stream = arrayListDeStrings.stream();  
System.out.println(stream.anyMatch(s -> s.equals("Luciano")));
```

Stream filter

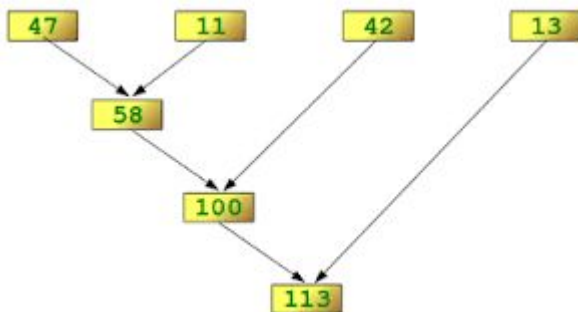
```
return nombres.stream().filter(s -> s.startsWith(inicio)).toList();
```

Stream reduce

```
List<Integer> arrayNumeros = Arrays.asList(47,11,42,13);  
int resultado = arrayNumeros.stream().reduce(0, (acumulado, nuevo) -> acumulado+nuevo);
```

Stream reduce

```
List<Integer> arrayNumeros = Arrays.asList(47,11,42,13);  
int resultado = arrayNumeros.stream().reduce(0, (acumulado, nuevo) -> acumulado+nuevo);
```



Stream reduce

```
List<ValueHolder> arrayVH = Arrays.asList(new ValueHolder(5), new ValueHolder(10), new ValueHolder(15));  
ValueHolder resultado = arrayVH.stream().reduce(new ValueHolder(0),  
    (acumulado, nuevo) -> new ValueHolder(acumulado.getValue() + nuevo.getValue()));
```

map

Obtener la lista de tamaños de nombres

map

```
return this.nombres.stream().map(s -> s.length()).toList();
```

Anidamiento

```
ArrayList<Producto> lista= new ArrayList<Producto>();
```

```
lista.add(new Producto("Milanesa",80));
```

```
lista.add(new Producto("Puré",50));
```

```
lista.add(new Producto("Agua",70));
```

```
lista.add(new Producto("Fideo",95));
```

```
double resultado=lista.stream()
```

```
.mapToDouble(producto->producto.getMonto()*1.21)
```

```
.filter(producto->producto<100)
```

```
.sum();
```

```
System.out.println(resultado);
```


Anidamiento

Pueden leer la interfaz Stream