PARALLEL STREAM

java.util.Collection

```
default Stream<E> parallelStream()
```

```
/**
 * Returns a possibly parallel {@code Stream} with this
 * collection as its source. ...
*/
```

```
List<Integer> lista = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
Integer suma = lista.stream()
                    .reduce(0, (a,b)-> a+b);
```

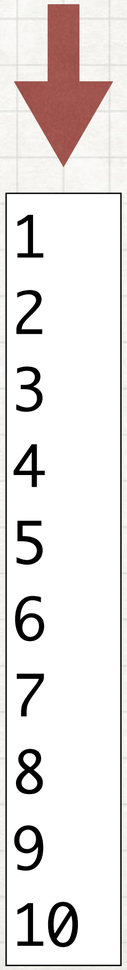
```
suma : 55
```

```
List<Integer> lista = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
Integer suma = lista.parallelStream()
                    .reduce(0, (a,b)-> a+b);
```

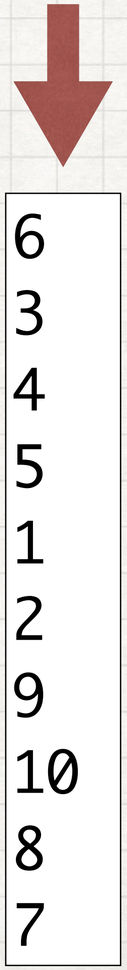
```
suma : 55
```

```
List<Integer> lista = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
Integer suma = lista.parallelStream()
                    .peek(System.out::println)
                    .reduce(0, (a,b)-> a+b);
```

```
lista.stream()
      .peek(System.out::println)
      ...
```




```
lista.parallelStream()
      .peek(System.out::println)
      ...
```



¿Y si no es paralelizable?

Convertimos los elementos a String

```
lista.parallelStream()
        .map(String::valueOf)
        .collect(Collectors.joining(","));
```

Actua subre un stream de String, concatenando el contenido con el separador indicado

```
lista.parallelStream()
        .map(String::valueOf)
        .collect(Collectors.joining(","));
```

1,2,3,4,5,6,7,8,9,10

1,4,5,2,3,9,10,6,7,8

```
Collectors.joining(",")
```

```
characteristics()
```



```
{}
```