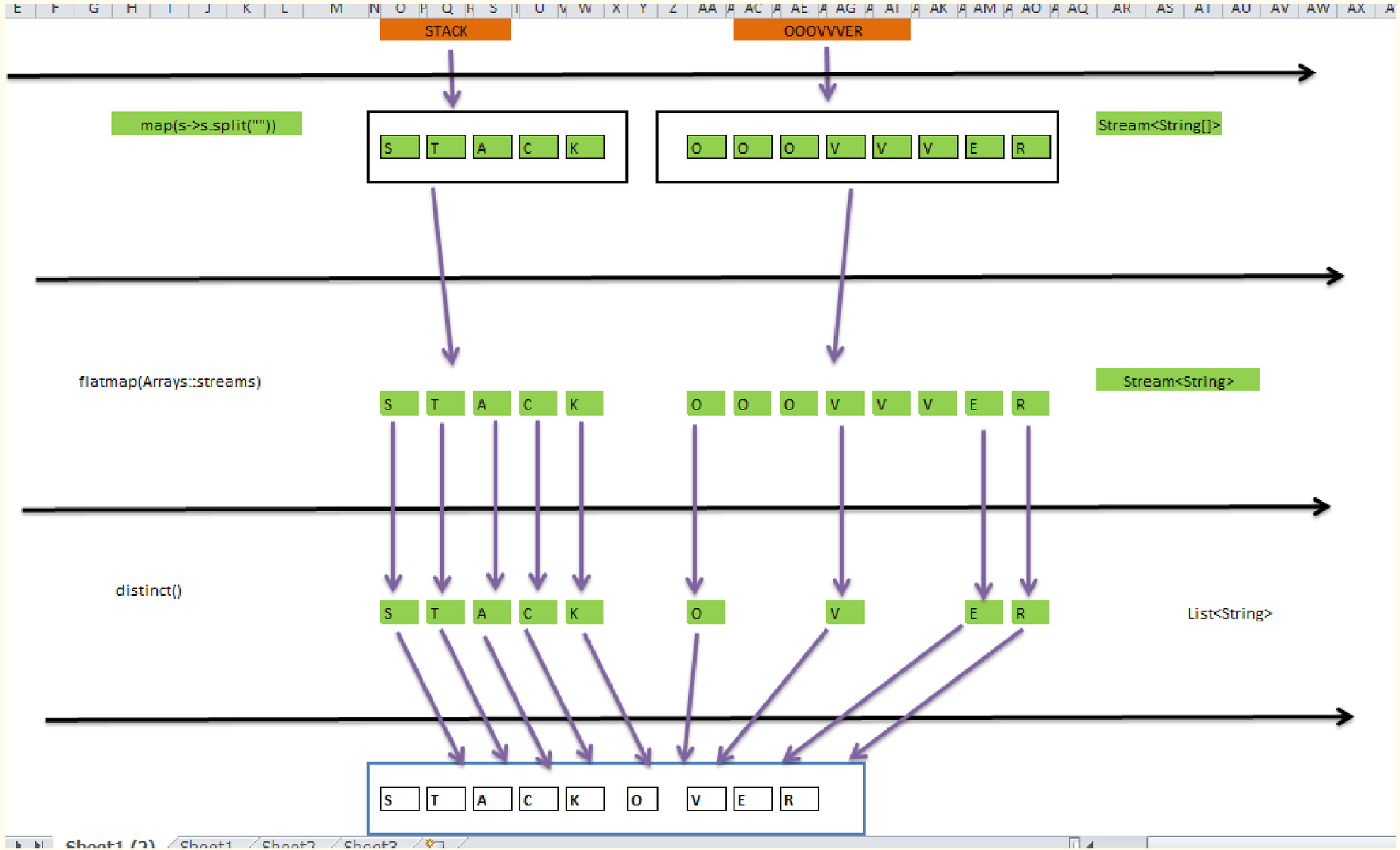

STREAM

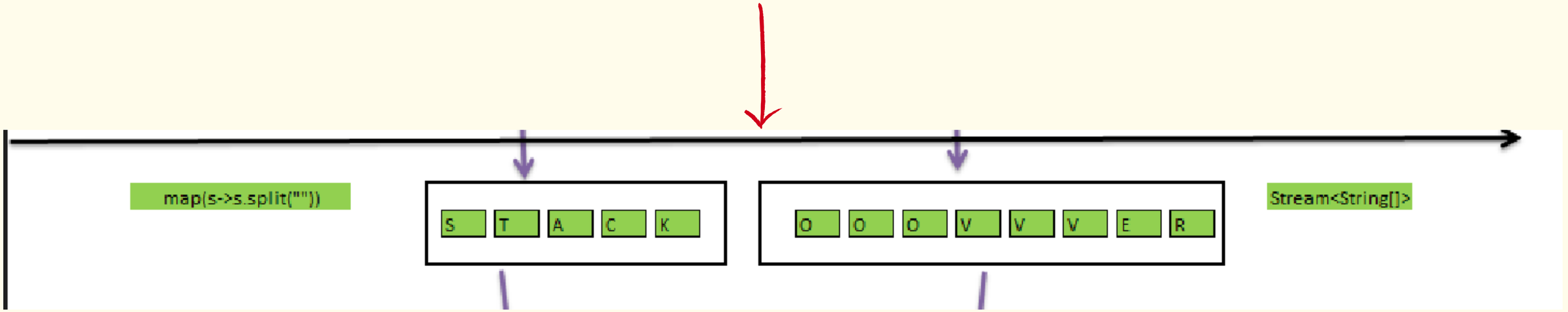
EJEMPLO


STACK
OOOVVVER
map(s->s.split(""))
Stream<String[]>
S T A C K
O O O V V V E R
flatmap(Arrays::streams)
Stream<String>
S T A C K O O O V V V E R
distinct()
S T A C K O V E R
List<String>
S T A C K O V E R

DECLARAMOS UN FLUJO DE CADENAS DE TEXTO QUE UTILIZAREMOS PARA REALIZAR ALGUNAS OPERACIONES

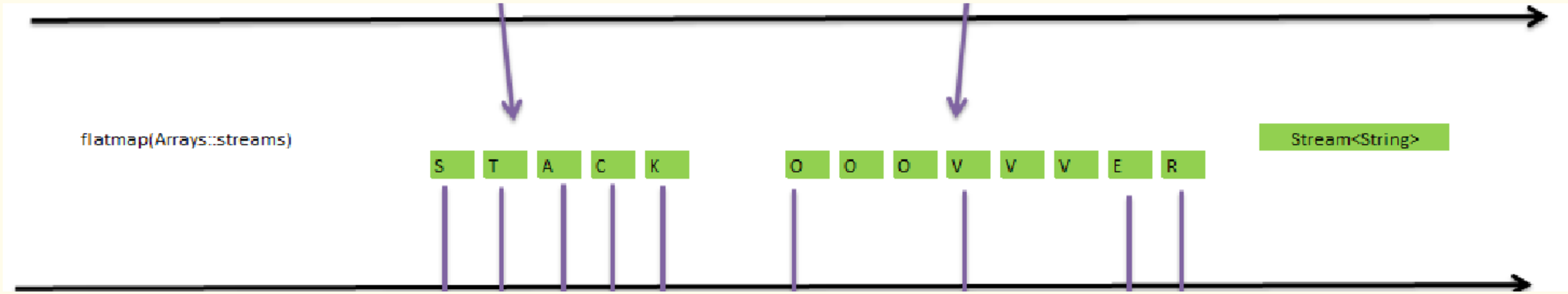
```
Stream<String> palabras = Stream.of("STACK"," OOOVVVER");
```

.map nos ayuda a que "**STACK**" se dividirá en **["S", "T", "A", "C", "K"],** y la palabra "**OOOVVVER**" se dividirá en **["", "O", "O", "O", "V", "V", "V", "E", "R"].**

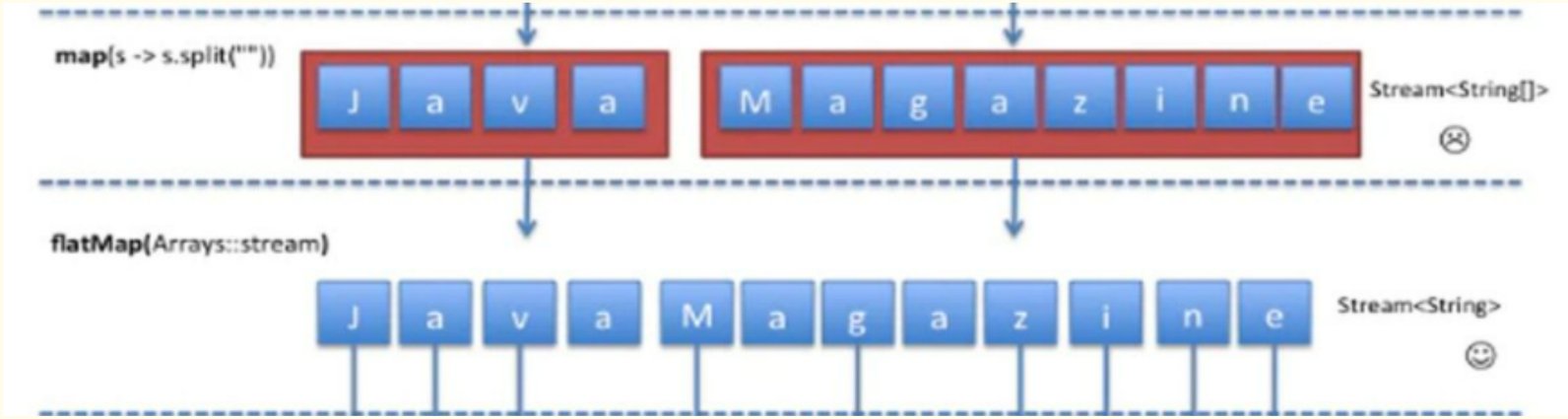



```
List<String> palabrasRefactorizadas = palabras
        .map(p -> p.split(""))
```

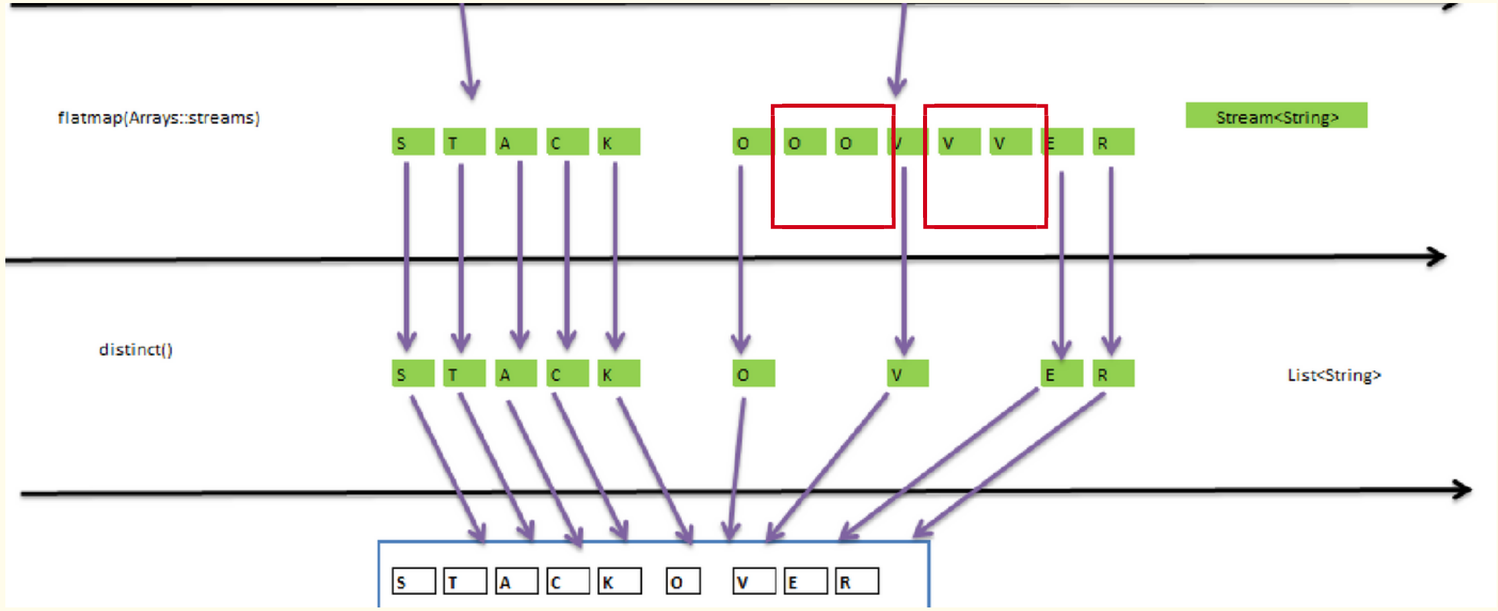
.fatMap nos ayuda a convertir ["S", "T", "A", "C", "K"], en uno solo se convinan y se convierte en uno solo



.flatMap(Arrays::stream)



.distinct() nos ayuda a quitar los caracteres que estan duplicados



POR ULTIMO SE UNEN LOS CARACTERES Y SE ALMACENAN EN PALABRARREFACTORIZADAS

```
.collect(Collectors.toList());
```

imprimimos el resultado aplicando un forEach

```java
Stream<String> palabras = Stream.of("STACK"," OOOVVVER");

List<String> palabrasRefactorizadas = palabras
        .map(p -> p.split(""))
        .flatMap(Arrays::stream)
        .distinct()
        .collect(Collectors.toList());

palabrasRefactorizadas.forEach(resul -> System.out.print(resul));
```