

TIPOS DE LOS LAMBDA

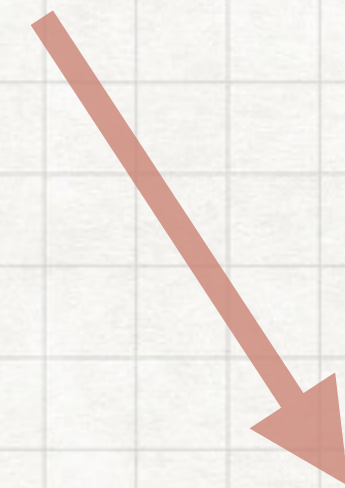

```
Comparator<String> comparadorLongitud =  
    (o1, o2) -> o1.length() - o2.length();
```

(o1, o2) -> o1.length() - o2.length() ?

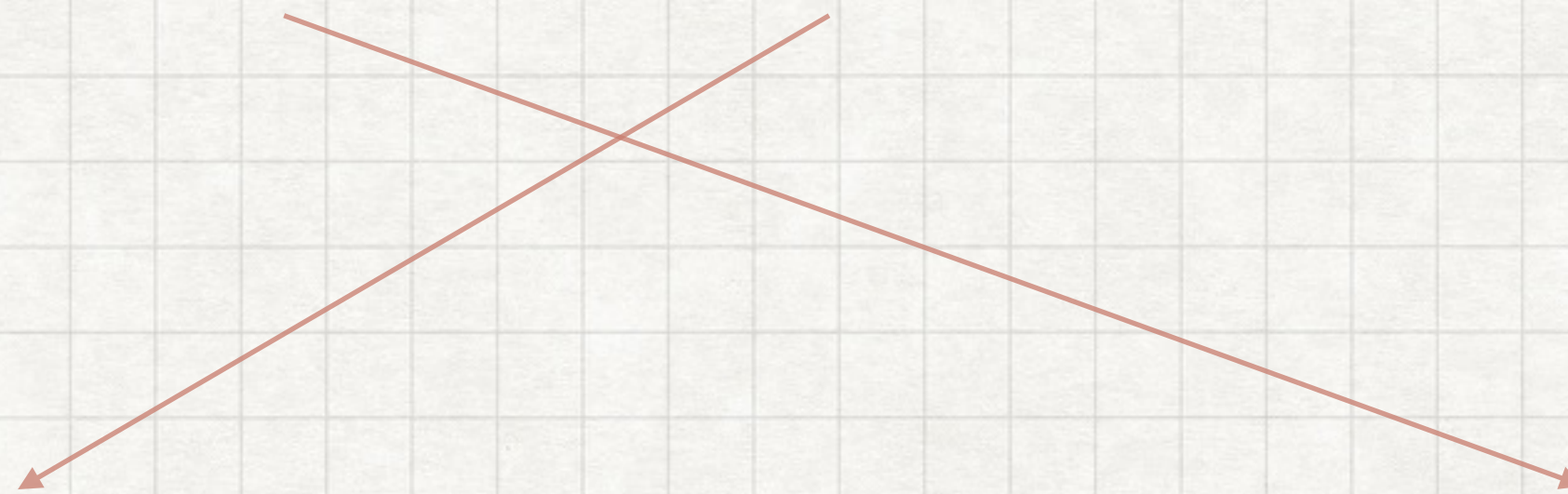
¿Como han encajado los lambdas en Java?

$(o1, o2) \rightarrow o1.length() - o2.length()$

lambda expression



functional interface instance



es decir: un objeto de un determinado tipo


```
Comparator<String> comparadorLongitud =  
    (o1, o2) -> o1.length() - o2.length();
```

lambda expresssion

```
@FunctionalInterface  
public interface Comparator<T>
```

*implementación del
único método abstract*


```
@FunctionalInterface  
public interface Comparator<T>
```

int compare(T **o1**, T **o2**);

(**o1**, **o2**) -> **o1.length() - o2.length()**




```
Comparator<String> comparadorLongitud =  
(o1, o2) -> o1.length() - o2.length();
```

A diagram consisting of a rounded rectangle box containing the lambda expression `(o1, o2) -> o1.length() - o2.length();`. An arrow originates from the right side of this box and points to the `compare` method signature inside a large circle.

```
class XX  
    implements Comparator<String>
```

```
int compare(T o1, T o2) {  
    ...  
}
```

A large circle containing the implementation of the `compare` method. An arrow from the lambda expression box points to the `compare` method signature. Another arrow from the top of the circle points back to the `comparadorLongitud` variable in the first code block.