

PROBLEMA



# EL PROBLEMA

```
List<String> nombres = Arrays.asList("Juan", "Antonia", "Pedro");
```



```
public final class String  
extends Object  
implements Serializable, Comparable<String>, CharSequence
```

```
Collections.sort(nombres);
```



[Antonia, Juan, Pedro]



# EL PROBLEMA

```
Collections.sort(nombres, comparador);
```



```
public interface Comparator<T> {  
    int compare(T o1, T o2);  
}
```


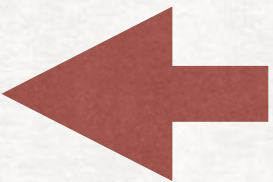
Ordenar por longitud

[Juan, Pedro, Antonia]



# SOLUCIÓN PRE-JAVA8

```
Comparator<String> comparadorLongitud = new Comparator<String>() {  
  
    @Override  
    public int compare(String o1, String o2) {  
        return o1.length() - o2.length();  
    }  
  
};  
  
Collections.sort(nombres, comparadorLongitud);
```





# LAMBDA

```
int compare(String o1, String o2)
```

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

Parámetros

Valor devuelto

f(String, String) -> int



# LAMBDA

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



# LAMBDA

```
Collections.sort(nombres, (o1, o2) -> o1.length() - o2.length());
```



# LAMBDA

```
Collections.sort(nombres, Comparator.comparing(String::length));
```

Nuevos métodos en Comparator



Method reference





# LAMBDA

```
Comparator<String> comparadorLongitud = new Comparator<String>() {  
  
    @Override  
    public int compare(String o1, String o2) {  
        return o1.length() - o2.length();  
    }  
  
};  
  
Collections.sort(nombres, comparadorLongitud);
```



```
Collections.sort(nombres, Comparator.comparing(String::length));
```