COLLECTION

java.util.Collection

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

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

```java
default Spliterator<E> spliterator()
```

```java
nombres.spliterator().forEachRemaining(System.out::println);
```

```java
Spliterator<T> trySplit();

int characteristics()
```

```
default boolean removeIf(Predicate<? super E> filter)
```

java.util.List

```
default void replaceAll(UnaryOperator<E> operator)

default void sort(Comparator<? super E> c)
```

java.util.Map

```
default V getOrDefault(Object key, V defaultValue)
```

```
default void forEach(BiConsumer<? super K, ? super V> action)
```

java.util.Map

```
default V replace(K key, V value)

default boolean replace(K key, V oldValue, V newValue)

default V putIfAbsent(K key, V value)

default boolean remove(Object key, Object value)
```

java.util.Map

```java
default void replaceAll(
    BiFunction<? super K, ? super V, ? extends V> function)
```

function(key, value) -> newValue

keyA	valueA

keyA	function(keyA, valueA)

java.util.Map

```
default V compute(K key,
     BiFunction<? super K, ? super V, ? extends V> remappingFunction)

default V computeIfPresent(K key,
     BiFunction<? super K, ? super V, ? extends V> remappingFunction)

    default V computeIfAbsent(K key,
                Function<? super K, ? extends V> mappingFunction)
```

java.util.Map

```
default V merge(K key, V value,
     BiFunction<? super V, ? super V, ? extends V> remappingFunction)
```

```
Map<String, Integer> contadores = ...

contadores.merge("clave", 1, (a,b) -> a+b);
```