Ejemplo borrado con Iterador

El método `Iterator.remove()` es un método que elimina el elemento devuelto por la llamada anterior a `Iterator.next()`. Por ejemplo, el siguiente código rellena una lista de cadenas y luego elimina todas las cadenas vacías.

```
List<String> names = new ArrayList<>();
names.add("name 1");
names.add("name 2");
names.add("");
names.add("name 3");
names.add("");
System.out.println("Tamaño antes: " + names.size());

Iterator<String> it = names.iterator();
while (it.hasNext()) {
  String el = it.next();
  if (el.equals("")) {
    it.remove();
  }
}
System.out.println("Tamaño despues: " + names.size());
```

Salida:

```
Tamaño antes: 5
Tamaño despues: 3
```

Ten en cuenta que el código anterior es la forma segura de eliminar elementos al iterar una colección típica. Si por el contrario, intentas eliminar elementos de una colección como esta:

```
for (String el: names) {
    if (el.equals("")) {
        names.remove(el); // MAL!
    }
}
```

Lanzará una `ConcurrentModificationException` .

El método `remove()` solo puede ser llamado (una vez) después de una llamada `next()` . Si se llama antes de llamar a `next()` o si se llama dos veces después de la llamada `next()` , entonces la llamada `remove()` lanzará una `IllegalStateException` .

La operación de `remove` se describe como una operación *opcional*, es decir, no todos los iteradores lo permitirán. Los ejemplos en los que no se admite incluyen iteradores para colecciones inmutables, vistas de solo lectura de colecciones o colecciones de tamaño fijo. Si se llama a `remove()` cuando el iterador no admite la eliminación, lanzará una `UnsupportedOperationException` .