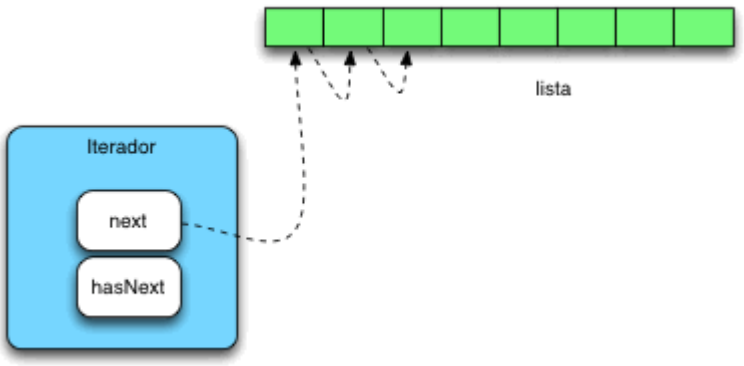
He decidido escribir de vez en cuando artículos mas básicos o esenciales del lenguaje Java para las personas que están empezando a manejar el lenguaje y este será el primero. Una de las situaciones mas habituales en la programación es recorrer una lista de elementos. En Java se puede realizar de varias maneras ,dos de las mas comunes son usando Java Iterator o usando un bucle forEach. Vamos a compararlas y ver algunas de las peculiaridades.

Java Iterator

Definimos una lista de cadenas como la siguiente y la vamos a recorrer a través de un Iterador.

```
ArrayList<String> lista = new ArrayList<String>();
lista.add("Pedro");
lista.add("Olga");
lista.add("Miguel");
lista.add("Antonio");
lista.add("Pedro");
```

Un iterador es un objeto que nos permite recorrer una lista y presentar por pantalla todos sus elementos . Dispone de dos métodos clave para realizar esta operación hasNext() y next().



Vamos a verlo en ejecución:

```
Iterator<String> it = lista.iterator();

while (it.hasNext()) {

System.out.println(it.next());

}
```

Esto nos presentaría la lista de elementos por pantalla :

Pedro
Olga
Miguel
Antonio
Pedro

Java 5 y bucle foreach

A partir de Java 5 existe otra forma de recorrer una lista que es mucho mas cómoda y compacta , el uso de bucles foreach. Un bucle foreach se parece mucho a un bucle for con la diferencia de que no hace falta una variable i de inicialización :

```
for (String nombre : lista) {

 System.out.println(nombre);
}
```

El resultado es claramente superior y mas legible para todo el mundo . ¿Es momento de dejar de usar iteradores para siempre? . La respuesta del público suele ser que sí . Sin embargo a veces los iteradores aportan cosas interesantes que los bucles foreach no pueden abordar.

Borrando elementos

Vamos a borrar todas las personas que se llaman “Pedro” de la lista . La operación parece tan sencilla como hacer lo siguiente

```
for (String nombre : lista) {

if (nombre.equals("Pedro")) {

lista.remove("Pedro");
}
```

```
}
```

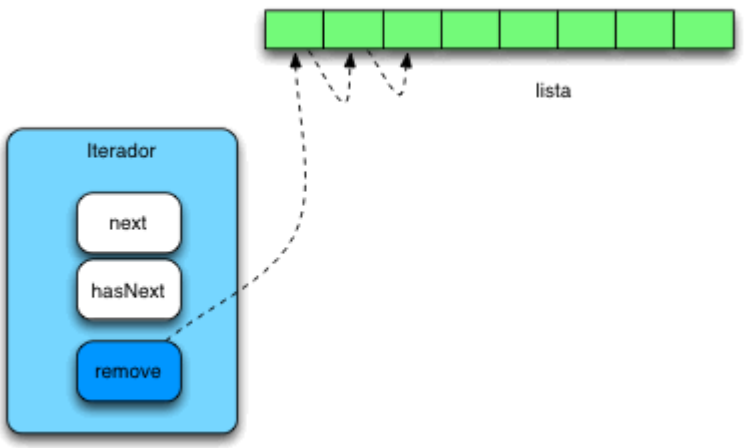
Lamentablemente el código no funciona y lanza una excepción :

```
Exception in thread "main" java.util.ConcurrentModificationException
        at java.util.ArrayList$Itr.checkForComodification(ArrayList.java:819)
        at java.util.ArrayList$Itr.next(ArrayList.java:791)
        at com.arquitecturajava.Principal.main(Principal.java:44)
```

Tenemos un problema ya que la operación que queríamos realizar es sencilla ,pero falla escandalosamente ya que estamos recorriendo y modificando la lista a la vez. Es aquí donde los iteradores pueden venir a rescatarnos.

Java iterator y remove

El interface Iterator dispone de un método adicional que permite eliminar objetos de una lista mientras la recorremos (el método remove) :



Vamos a verlo en ejecución :

```
Iterator<String> it= lista.iterator();

while(it.hasNext()) {

String nombre= it.next();
if (nombre.equals("Pedro")) {

it.remove();
}
}
```

El resultado será :

Olga
Miguel
Antonio

De esta forma podrémos eliminar elementos de una lista de forma segura:)