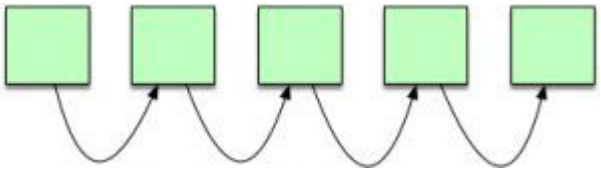
El concepto de Java Iterable es un concepto clásico en el mundo Java y existe desde la versión de Java 1.5 . Un Iterable es un interface que hace referencia a una colección de elementos que se puede recorrer, ni más ni menos.



Así pues el interface solo necesita que implementemos un método para poder funcionar de forma correcta, este método es iterator(). Vamos a ver un ejemplo con una lista de cadenas y como se usa el bucle “forEach” de java para recorrer una lista de elementos que implementan el interface Iterable. Recordemos que un ArrayList implementa este interface.

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

public class Principal {

        public static void main(String[] args) {

                List&amp;amp;lt;String&amp;amp;gt; lista= new
ArrayList&amp;amp;lt;String&amp;amp;gt;();
                lista.add("hola");
                lista.add("que");
                lista.add("tal");
                lista.add("estas");

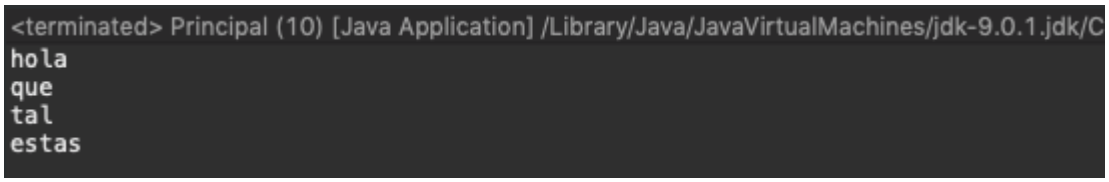
                for (String s:lista) {
```

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

        }

}
```

Veamos el resultado en la consola:



Esto es muy sencillo de entender . Lo que a veces es un poco más complicado es construir una clase que implemente correctamente el interface y podamos usarla con un bucle forEach de Java . En nuestro caso vamos a construir una clase que genere los valores de una tabla de multiplicar por la consola.

```
package com.arquitecturajava;

import java.util.Iterator;

public class ListaMultiplicar implements
Iterable&amp;amp;lt;Integer&amp;amp;gt; {

        private int posicion;
        private int size;
        private int numero;
```

```
	public ListaMultiplicar(int numero,int size) {

		this.size = size;
		this.numero = numero;

	}

	@Override
	public Iterator&amp;amp;lt;Integer&amp;amp;gt; iterator() {

		int size= this.size;

		Iterator&amp;amp;lt;Integer&amp;amp;gt; i= new
Iterator&amp;amp;lt;Integer&amp;amp;gt;() {

			@Override
			public boolean hasNext() {

				if (posicion&amp;amp;lt;size) {

					return true;
				}else {
					return false;
				}

			}

			@Override
			public Integer next() {

				return numero*posicion++;
```

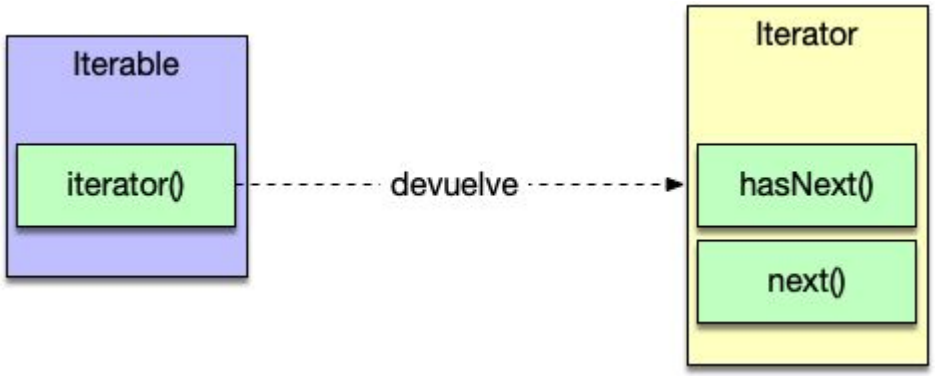
```
                        }

                };
                return i;
        }

}
```

Java Iterable e Iterator

En este caso hemos construido una lista que implementa el interface Iterator a través de una clase anónima.



Este clase anónima es la que implementa los dos métodos del Iterator next() y hashNext() de tal forma que podamos recorrerla de forma sencilla utilizando una estructura forEach.

```
package com.arquitecturajava;

public class Principal {

        public static void main(String[] args) {

                ListaMultiplicar lista2= new ListaMultiplicar(5,10);
```

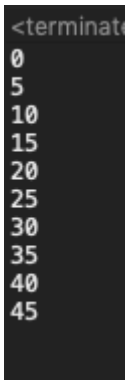
```
                for (int i: lista2) {

                        System.out.println(i);
                }

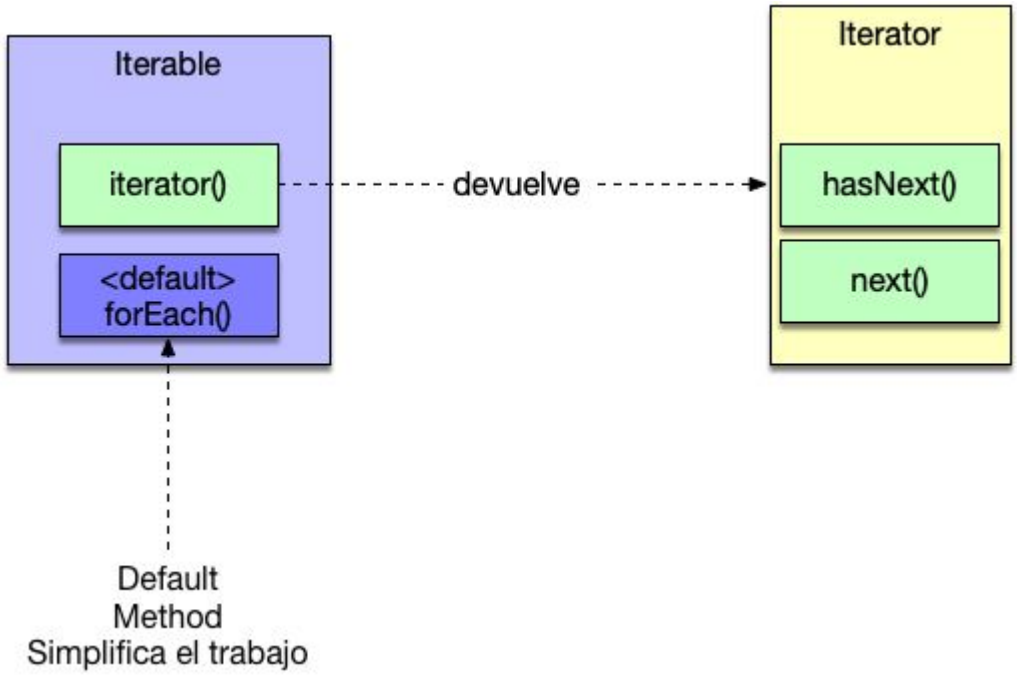
        }

}
```

El resultado lo podemos ver en la consola y nos imprimirá los valores de la tabla de multiplicar del 5.



El interface Iterable es como vemos relativamente sencillo de implementar , simplemente hay que tener en cuenta que necesitamos una clase que implemente iterator para que este funcione. Ahora bien existen algunas otras ventajas que a veces pasan desapercibidas a nivel del API. Si nuestra clase implementa Iterable de forma automática incluirá los métodos por defecto (default methods) que este interface aporta.



En nuestro caso el método mas conocido es el método forEach que nos permite realizar lo que se denomina un internal Iterable a nivel de Java 8.

```
package com.arquitecturajava;

public class Principal {

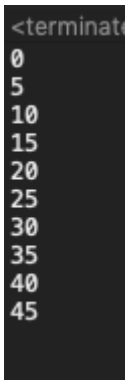
	public static void main(String[] args) {

		ListaMultiplicar lista2= new ListaMultiplicar(5,10);

		lista2.forEach(System.out::println);
	}

}
```

En este caso todo es mas sencillo invocamos al método forEach y usamos un reference Method. El resultado será idéntico.



Otros artículos relacionados:

1. Java Iterator vs ForEach
2. Java Collections Remove con Java 8
3. Java varargs y colecciones
4. Java Collections