El concepto de Java Collections Remove es muy simple. Este interface soporta una serie de métodos para eliminar elementos de una colección. Los dos métodos más utilizados son :

1. remove (index)
2. remove(Object)

Ambos métodos nos permiten eliminar un elemento de una colección concreta . Vamos a ver un ejemplo de toma de contacto con la clase libro:

```
package com.arquitecturajava;

public class Libro {

	private String titulo;
	private int paginas;
	private String categoria;
	public String getTitulo() {
		return titulo;
	}
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	public int getPaginas() {
		return paginas;
	}
	public void setPaginas(int paginas) {
		this.paginas = paginas;
	}
	public String getCategoria() {
		return categoria;
```

```
	}
	public void setCategoria(String categoria) {
		this.categoria = categoria;
	}
	public Libro(String titulo, int paginas, String categoria) {
		super();
		this.titulo = titulo;
		this.paginas = paginas;
		this.categoria = categoria;
	}
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((categoria == null) ? 0 :
categoria.hashCode());
		result = prime * result + paginas;
		result = prime * result + ((titulo == null) ? 0 :
titulo.hashCode());
		return result;
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Libro other = (Libro) obj;
		if (categoria == null) {
```

```
                        if (other.categoria != null)
                                return false;
                } else if (!categoria.equals(other.categoria))
                        return false;
                if (paginas != other.paginas)
                        return false;
                if (titulo == null) {
                        if (other.titulo != null)
                                return false;
                } else if (!titulo.equals(other.titulo))
                        return false;
                return true;
        }

}
```

Java Collections Remove Clásico

Vamos a crear una lista de libros y eliminar varios:

```
package com.arquitecturajava;

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

public class Principal {
```

```
        public static void main(String[] args) {

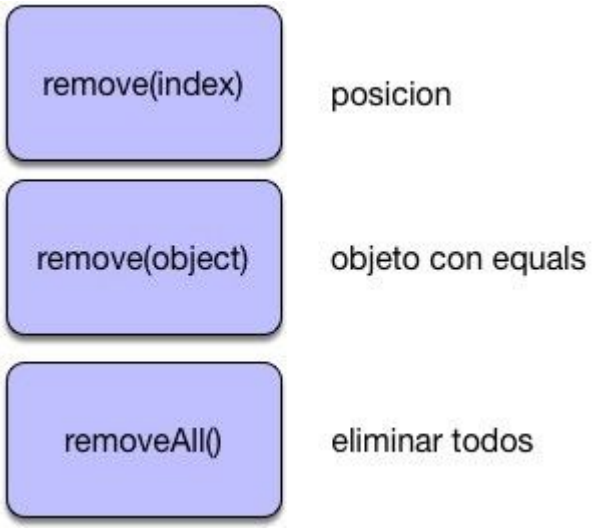
                Libro l1 = new Libro("java", 500, "programacion");
                Libro l2 = new Libro("angular", 300, "web");
                Libro l3 = new Libro("react", 350, "web");

                List&amp;lt;Libro&amp;gt; libros = new ArrayList();
                libros.add(l1);
                libros.add(l2);
                libros.add(l3);

                libros.remove(l1);
                libros.remove(0);
                libros.removeAll(Arrays.asList(l3));

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

Acabamos de eliminar todos los libros . El primero usando remove y pasando un objeto , el segundo eliminando por posición y el tercero eliminando con removeAll, todo es correcto.



.Sin embargo en muchos casos necesitamos eliminar un subconjunto de elementos aplicando una condicion. ¿Cómo podemos hacer esto?. Hasta ahora siempre se ha utilizado el interface Iterator.

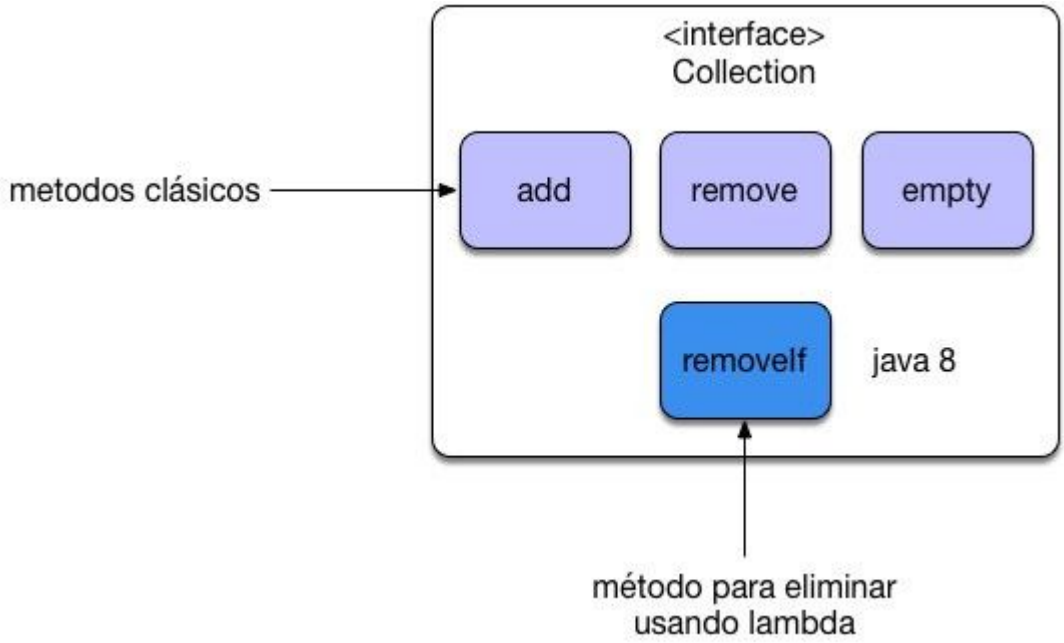
```
Iterator&amp;lt;Libro&amp;gt; it= libros.iterator();

while (it.hasNext()) {

        if (it.next().getCategoria().equals("web")) {
                it.remove();
        }
}
```

Java Collections Remove con Java 8

A mi a nivel personal nunca me ha convencido demasiado, pero era la solución existente. A partir de Java 8 tenemos otra opción ya que podemos utilizar expresiones lambda a través del método removeIf que aparece en el interface de Collection .



Así pues la operación se puede realizar de la siguiente forma.

```
libros.removeIf(l-&amp;gt;l.getCategoria().equals("web"));
```

De esta forma eliminaremos los elementos que se encuentran dentro de la categoría “web” de una forma más elegante.

Otros artículos relacionados:

1. Java 8 Lambda Expressions (I)
2. Java Predicate Interface y sus métodos
3. Java 8 Lambda y forEach (II)
4. Java Collections