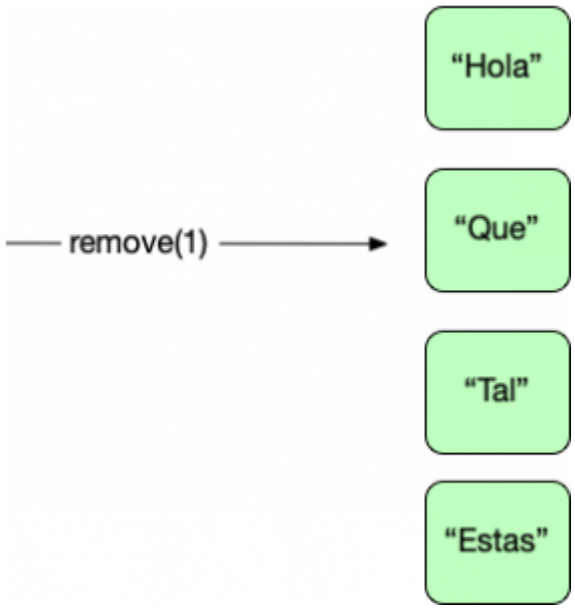
Java ArrayList remove ,es quizás uno de los métodos más habituales del framework de colecciones a la hora de eliminar elementos de una lista en Java . También se trata de uno de los métodos que más opciones soporta y más dudas genera a la hora de eliminar . Vamos a ver esta funcionalidad explicada a detalle partiendo en un primer lugar de una lista de textos y usando el método más sencillo que es el de borrar por posición.



En este caso el método recibe la posición del ArrayList que desea borrar y la borra:

```
package com.arquitecturajava;

import java.util.ArrayList;

public class Principal2 {

  public static void main(String args[]) {

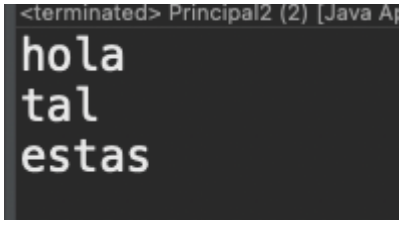
    ArrayList<String> lista = new ArrayList<String>();
    lista.add("hola");
    lista.add("que");
    lista.add("tal");
```

```
    lista.add("estas");
    lista.remove(1);
    for (String texto : lista) {

      System.out.println(texto);
    }
  }

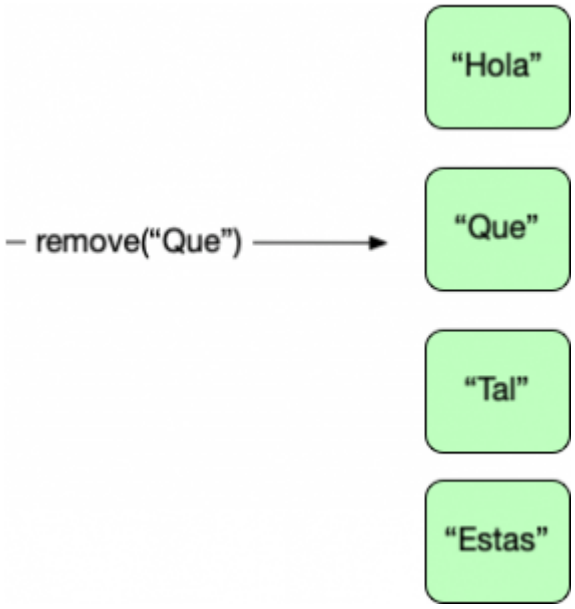
}
```

El resultado sale en la consola:



Eliminando Objetos

Ahora bien en muchos casos los desarrolladores deciden usar otro método. El método que se encarga de borrar un objeto directamente sin que tengas que acceder por posición . En este caso podríamos solicitar un borrado de este estilo:



Veamos el código:

```
package com.arquitecturajava;

import java.util.ArrayList;

public class Principal2 {

  public static void main(String args[]) {

    ArrayList<String> lista = new ArrayList<String>();
    lista.add("hola");
    lista.add("que");
    lista.add("tal");
    lista.add("estas");

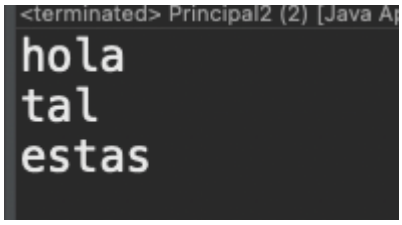
    lista.remove("que");

    for (String texto : lista) {
```

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

}
```

El resultado es idéntico ya que borramos el elemento de la lista:



La diferencia es que la lista se recorre y se busca el elemento que cumple la condición para luego borrarle. Los problemas comienzan cuando uno intenta borrar en una lista definida por una clase propia . Por ejemplo una lista de Personas , en este caso hay que recordar que para que un ArrayList pueda eliminar un objeto de la lista simplemente pasándole el objeto como parámetro este debe tener correctamente sobrecargados dos métodos

equals: El método que define cuando dos objetos a nivel de lógica de negocio son iguales

hashCode: El método que nos ayuda a posicionar un objeto dentro de una estructura de datos.

Para que dos objetos sean iguales deben tener el mismo hashcode y el método equals debe devolver true.

Veamoslo en acción con la clase Persona en la cual se define la igualdad por DNI:

```
package com.arquitecturajava;
```

```
import java.util.Objects;

public class Persona {

	private String dni;
	private String nombre;
	public String getDni() {
		return dni;
	}
	public void setDni(String dni) {
		this.dni = dni;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public Persona(String dni, String nombre) {
		super();
		this.dni = dni;
		this.nombre = nombre;
	}
	@Override
	public int hashCode() {
		return Objects.hash(dni);
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
```

```
                if (obj == null)
                        return false;
                if (getClass() != obj.getClass())
                        return false;
                Persona other = (Persona) obj;
                return Objects.equals(dni, other.dni);
        }
}
```

Es momento de crear una lista de Personas y ver si somos capaces de eliminar una de la lista pasando un objeto:

```
package com.arquitecturajava;

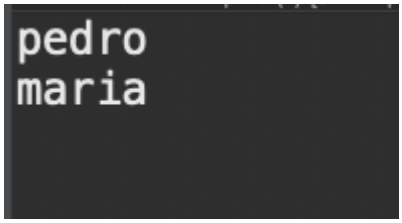
import java.util.ArrayList;

public class Principal3 {

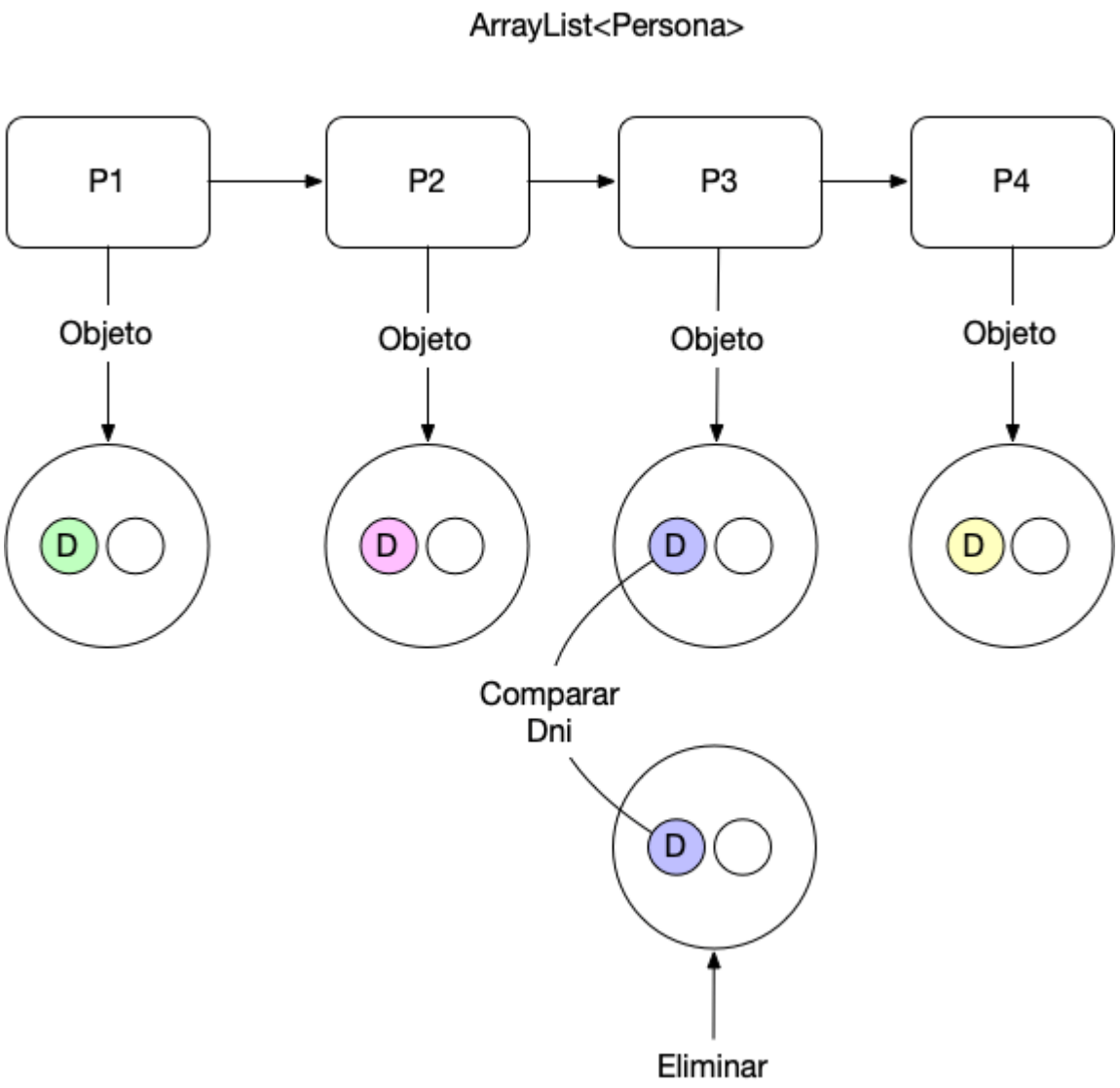
        public static void main(String[] args) {
                Persona p1= new Persona("1","pedro");
                Persona p2= new Persona("2","ana");
                Persona p3= new Persona("3","maria");
                ArrayList<Persona> personas=new ArrayList<Persona>();
                personas.add(p1);
                personas.add(p2);
                personas.add(p3);
                personas.remove(p2);
                for (Persona p: personas) {
                        System.out.println(p.getNombre());
                }
        }
```

```
}
```

El resultado en la consola es claro :



Hemos sobrecargado correctamente los métodos equals y hashcode. A continuación se muestra un diagrama orientativo de como esta operación funciona:



Java ArrayList remove y colecciones

Otra de las opciones posibles es pasar una colección de Personas al método remove . Esa colección debe implementar el interface Collection y lo más habitual es que se trate de otro ArrayList del cual queramos eliminar elementos.

```
package com.arquitecturajava;

import java.util.ArrayList;
```

```
public class Principal3 {

	public static void main(String[] args) {
		Persona p1= new Persona("1","pedro");
		Persona p2= new Persona("2","ana");
		Persona p3= new Persona("3","maria");
		ArrayList<Persona> personas=new ArrayList<Persona>();
		personas.add(p1);
		personas.add(p2);
		personas.add(p3);
		ArrayList<Persona> otras= new ArrayList<Persona>();
		otras.add(new Persona("1","pedro"));
		otras.add(new Persona("2","ana"));
		personas.removeAll(otras);
		for (Persona p: personas) {
			System.out.println(p.getNombre());
		}
	}

}
```

Remove y Java 8 (Premium)

[ihc-hide-content ihc_mb_type=”show” ihc_mb_who=”4″ ihc_mb_template=”1″]

Una de las opciones más avanzadas o modernas cuando tenemos la necesidad de eliminar elementos de un ArrayList es usar el metodo removeIf que recibe una expresión lambda de tipo Consumer y es capaz de afinar mucho mas la forma en la que nosotros eliminamos elementos de una lista.

```
package com.arquitecturajava;
```

```
import java.util.ArrayList;

public class Principal5 {

	public static void main(String[] args) {
		Persona p1= new Persona("1","pedro");
		Persona p2= new Persona("2","ana");
		Persona p3= new Persona("3","maria");
		ArrayList<Persona> personas=new ArrayList<Persona>();
		personas.add(p1);
		personas.add(p2);
		personas.add(p3);
		personas.removeIf(p->p.getDni().equals("2"));
		for (Persona p: personas) {
			System.out.println(p.getNombre());
		}
	}

}
```

Otros artículos relacionados

- Java 8 StringJoiner y manejo de cadenas
- Entity to DTO y Java 8
- Java 8 Functional Interfaces y sus tipos
- Java Stream String y Java 8
- Java 8