Java Ordered vs Sortered . Esta es una de las preguntas más típicas cuando programamos con el framework de Colecciones ya que dependiendo del idioma Ordered y Sorted a nivel de traducción pueden llegar a significar lo mismo . Vamos a intentar aclarar las diferentes entre ambas situaciones para ello vamos a construir un sencillo código de Java con un conjunto utilizando la clase HashSet.

```
package com.arquitecturajava;
```

```
import java.util.HashSet;
import java.util.Set;

public class Principoal1 {

	public static void main(String[] args) {
			Set<String> conjunto = new HashSet<>();
			conjunto.add("hola");
			conjunto.add("que");
			conjunto.add("tal");
			conjunto.add("estas");
			conjunto.add("hoy");
			for (String texto: conjunto) {
				System.out.println(texto);
			}
	}

}
```

En principio se trata de un grupo de cadenas sin orden alguno.

Sin orden alguno

Acabamos de construir un HashSet que es una sencilla colección de Java que permite añadir

elementos. Lo curioso es que luego queremos recorrer esta colección y mostrar los resultados. Vamos para ello a usar un bucle forEach.

```
que
hoy
hola
estas
tal
```

El resultado a muchas personas les puede sorprender. Resulta que no existe un orden o una organización aparente a la hora de presentar los datos de un conjunto o Set. No es nada especial sino la cruda de realidad de usar un HashSet que es una implementación del interface Set que define una colección de elementos no repetidos. En este caso cumplimos con que se trata de una colección de elementos que no se repiten ni mas ni menos. Nadie tiene que asegurar ningún tipo de orden a la hora de de recorrerlos.

Java Ordered vs Sortered y List

Vamos a realizar el mismo ejemplo pero usando un List y su implementación ArrayList

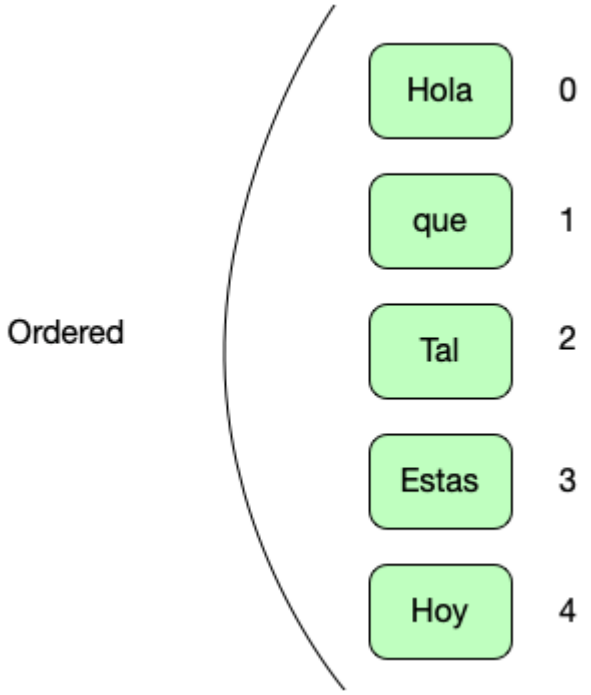
```
package com.arquitecturajava;

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

public class Principoal2 {

        public static void main(String[] args) {
                        List<String> conjunto = new ArrayList<>();
                        conjunto.add("hola");
                        conjunto.add("que");
                        conjunto.add("tal");
```

```
                        conjunto.add("estas");
                        conjunto.add("hoy");
                        for (String texto: conjunto) {
                                System.out.println(texto);
                        }
        }

}
```

El resultado que obtenemos es mucho más razonable:

```
hola
que
tal
estas
hoy
```

¿Nos ha devuelto los datos ordenados? . La realidad es que no , simplemente nos ha devuelto los datos organizados de la misma forma en las que los hemos insertado.



Una lista nos permite tener los datos organizados (“ordered”) y accesibles por posición . De hecho podríamos recorrer el ArrayList accediendo a sus posiciones.

```
package com.arquitecturajava;

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

public class Principoal3 {

        public static void main(String[] args) {
                        List<String> conjunto = new ArrayList<>();
                        conjunto.add("hola");
                        conjunto.add("que");
                        conjunto.add("tal");
                        conjunto.add("estas");
                        conjunto.add("hoy");
                        for (int i=0;i<conjunto.size();i++) {
```

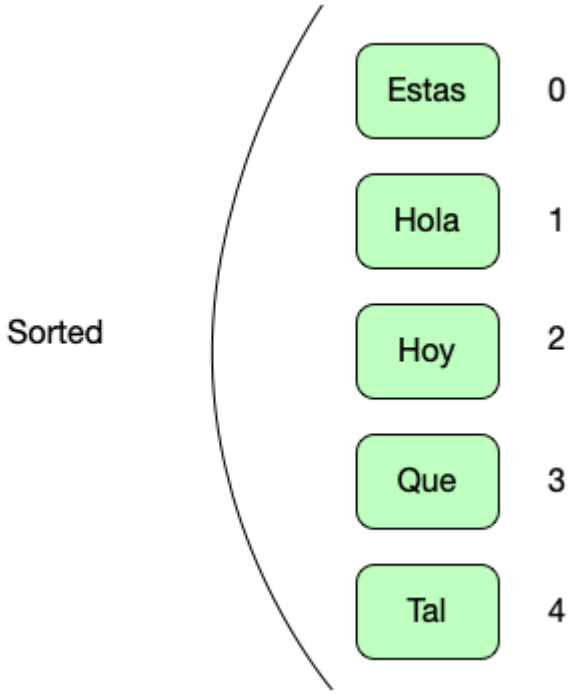
```
                                System.out.println(conjunto.get(i));
                        }
        }

}
```

El resultado es el mismo:

```
hola
que
tal
estas
hoy
```

Java Ordered vs Sortered resultados

Ordered vs Sorted . La diferencia entre ambos conceptos es importante el primero es Organizado y el segundo es lo que nosotros entendemos por ordenado . Es decir hay un “Orden” basado en una propiedad o conjunto de propiedades de la clase que hace que cuando los datos se impriman se siga una regla concreta. Por ejemplo en nuestro caso antes de imprimir los datos podemos realizar una operación de Sort y nos los devolverá ordenados de forma alfabética ya que son cadenas de esto.



Para eso invocaremos el método sort del interface de Collections.

```
package com.arquitecturajava;

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

public class Principoal3 {

        public static void main(String[] args) {
                        List<String> conjunto = new ArrayList<>();
                        conjunto.add("hola");
                        conjunto.add("que");
                        conjunto.add("tal");
                        conjunto.add("estas");
                        conjunto.add("hoy");
```

```
                        Collections.sort(conjunto);
                        for (int i=0;i<conjunto.size();i++) {
                                System.out.println(conjunto.get(i));
                        }
        }

}
```

En este caso si imprimimos el resultado habrán salido ordenadas alfabéticamente:

```
estas
hola
hoy
que
tal
```

No confundamos nunca “ordered” con “sorted” ya que el primero organiza y nos permite acceder a los elementos normalmente por posición y el segundo realiza una ordenación apoyándose en un criterio concreto en este caso alfabetico.

Otros artículos relacionados

- Java Collections Framework y su estructura
- Java ArrayList remove y sus opciones
- Java List Directory en Java 8 con Streams
- Java Map for loop , un enfoque moderno
- TreeSet