

75 - Colecciones: HashSet, TreeSet y LinkedHashSet

[Listado completo de tutoriales](#)

La diferencia fundamental entre las clases HashSet, TreeSet, LinkedHashSet con respecto a las listas ArrayList y LinkedList es que no puede haber elementos repetidos en las colecciones que implementan la interfaz Set.

A su vez se han creado estas tres clases que tienen pequeñas diferencias entre una y otras:

- HashSet: El conjunto de datos no se almacena en un orden específico, si bien se garantiza que no hay duplicados.
- TreeSet: Los elementos del conjunto se almacenan de menor a mayor.
- LinkedHashSet: Los elementos del conjunto se encuentran en el orden que se insertan, similar a una lista pero sin dejar ingresar valores repetido.

El siguiente programa muestra la sintaxis para crear objetos de estas clases y los métodos principales que disponen:

Programa:

[Ver video](#)

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

public class PruebaSet {

    public static void main(String[] args) {
        Set<Integer> conjunto1 = new HashSet<Integer>();
        conjunto1.add(20);
        conjunto1.add(10);
        conjunto1.add(1);
        conjunto1.add(5);
        // El valor 20 no se inserta en el conjunto ya que se encuentra repetido
        conjunto1.add(20);
        // La impresión no asegura un orden específico
        for (int elemento : conjunto1)
            System.out.print(elemento + " - ");
        System.out.println();

        Set<Integer> conjunto2 = new TreeSet<Integer>();
        conjunto2.add(20);
        conjunto2.add(10);
        conjunto2.add(1);
        conjunto2.add(5);
        // El valor 20 no se inserta en el conjunto ya que se encuentra repetido
        conjunto2.add(20);
        // Los elementos se muestran de menor a mayor
```

El resultado de ejecutar el programa es similar a:

```

1 import java.util.HashSet;
2 import java.util.LinkedHashSet;
3 import java.util.Set;
4 import java.util.TreeSet;
5
6 public class PruebaSet {
7
8     public static void main(String[] args) {
9         Set<Integer> conjunto1 = new HashSet<Integer>();
10        conjunto1.add(20);
11        conjunto1.add(10);
12        conjunto1.add(1);
13        conjunto1.add(5);
14        // El valor 20 no se inserta en el conjunto ya que se encuentra repetido
15        conjunto1.add(20);
16        // La impresión no asegura un orden específico
17        for (int elemento : conjunto1)
18            System.out.print(elemento + " - ");
19        System.out.println();
20
21        Set<Integer> conjunto2 = new TreeSet<Integer>();
22        conjunto2.add(20);
23        conjunto2.add(10);
24        conjunto2.add(1);
25        conjunto2.add(5);
26        // El valor 20 no se inserta en el conjunto ya que se encuentra repetido
27        conjunto2.add(20);
28        // Los elementos se muestran de menor a mayor
29        for (int elemento : conjunto2)
30            System.out.print(elemento + " - ");
31        System.out.println();
32
33        Set<Integer> conjunto3 = new LinkedHashSet<Integer>();
34        conjunto3.add(20);
35        conjunto3.add(10);
36        conjunto3.add(1);
37        conjunto3.add(5);
38        // El valor 20 no se inserta en el conjunto ya que se encuentra repetido
39        conjunto3.add(20);
40        // Los elementos se muestran en el orden que se insertaron
41        for (int elemento : conjunto3)
42            System.out.print(elemento + " - ");
43        System.out.println();
44    }
45 }
46

```

Problems Javadoc Console

<terminated> PruebaSet [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (11 mar. 2019 8:20:11)

```

1 - 20 - 5 - 10 -
1 - 5 - 10 - 20 -
20 - 10 - 1 - 5 -

```

Métodos más columnes

Los métodos más comunes que tienen estas clases son:

- size: Retorna la cantidad de elementos del conjunto.
- clear: Elimina todos los elementos.
- remove: Elimina el elemento si existe en el conjunto:

```
lista1.remove(20);
```

- isEmpty: Nos informa si la lista está vacía.
- contains: Le pasamos como parámetro el dato a buscar en el conjunto:

```
if (conjunto1.contains(20))
...
```

Más datos podemos conseguir visitando la documentación oficial de las clases [HashSet](#), [TreeSet](#) y [LinkedHashSet](#).

Problema

Generar una lista de 10 valores enteros comprendidos entre 1 y 100. Validar que no se repitan, para esto utilizar la ayuda de una de las colecciones de conjuntos visto en este concepto.

Programa:

[Ver video](#)

```
import java.util.Set;
import java.util.TreeSet;

public class Lista10Valores {
    public static void main(String[] args) {
        Set<Integer> conjunto1 = new TreeSet<Integer>();
        while (conjunto1.size() < 10) {
            int aleatorio = (int) (Math.random() * 100) + 1;
            conjunto1.add(aleatorio);
        }
        System.out.println(conjunto1);
    }
}
```

Dentro de un while mientras el objeto 'conjunto1' tenga menos de 10 elementos, procedemos a generar otro valor aleatorio y lo agregamos al conjunto, como sabemos si el valor ya existe en el conjunto1 luego el método 'add' no lo agrega:

```
while (conjunto1.size() < 10) {
    int aleatorio = (int) (Math.random() * 100) + 1;
    conjunto1.add(aleatorio);
}
```

Podemos recorrer el conjunto para imprimirlo mediante un for o inclusive utilizar el método 'println':

```
System.out.println(conjunto1);
```

[Retornar](#)