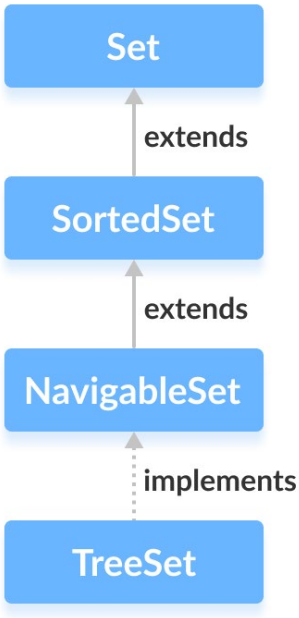
¿ Qué es la clase TreeSet <E> ?

La TreeSetclase del marco de colecciones de Java proporciona la funcionalidad de una estructura de datos de árbol.

Extiende de la interfaz NavigableSet.



TreeSet es una colección que siempre mantienen los elementos en orden ascendente. Si los elementos se insertan en el siguiente orden C, A, B se almacenarán de la siguiente forma A, B, C. A demás en el caso de que se repita uno de los elementos solo mostrará uno de ellos ya que todos los elementos tienen que ser únicos.

Ejemplo, insertamos A, C, B, A y se almacena A, B, C

Crear un TreeSet

Primero tenemos que importar el siguiente paquete java.util.TreeSet.

Posteriormente ya podremos instanciar el objeto:

TreeSet<Integer> nombrecoleccion = new TreeSet<>();

Métodos TreeSet

La clase TreeSet proporciona una serie de métodos para hacer operaciones de conjunto que veremos mas adelante.

Insetar Elementos:

- add(elemento) – Inseta el elemento que se le especifica entre paréntesis
- addAll(nombrecolección) – Inseta todos los elemtendo de la colección que se especifica entre paréntesis

Ejemplo:

```
import java.util.TreeSet;
class Main {
   public static void main(String[] args) {
      TreeSet<Integer> evenNumbers = new TreeSet<>();

      // Metodo add()
      evenNumbers.add(2);
      evenNumbers.add(4);
      evenNumbers.add(6);
      System.out.println("TreeSet: " + evenNumbers);

      TreeSet<Integer> numbers = new TreeSet<>();
      numbers.add(1);

      // Metodo addAll()
      numbers.addAll(evenNumbers);
      System.out.println("New TreeSet: " + numbers);

   }
}
```

Acceder a los elementos:

Para acceder a los elementos de un conjunto de árbol, es necesario utilizar el método iterator() de paquete java.util.Iterator

Para mostrar los elementos de la colección utilizando el paquete java.util.Iterator se hace de la siguiente forma:

```
while(iterate.hasNext()) {

    System.out.print(iterate.next());

    System.out.print(", ");

}
```

Eliminar elementos:

Para eliminar los elementos de la colección usamos los métodos:

- remove(nombreelemento) – Elimina el elemento especificado entre paréntesis
- removeAll(nombrecoleccion) – Eliminar todos los elementos que se especifica entre paréntesis

Ejemplo:

```
// Método remove()
numbers.remove(5);

// Método removeAll()
numbers.removeAll(numbers);
```

Métodos de navegación:

Son una serie de métodos que se utilizaran para navegar sobre la colección. Ejemplo devolver el primer valor o el ultino etc.

- first() – Devuelve el primer elemento del conjunto
- last() – Devuelve el último elemento del conjunto o colección
- higher (elemento) – Devuelve el elemento más bajo entre los elementos que son mayores que los especificados
- lower (elemento) – Devuelve el elemento más grande entre aquellos elementos que son menores que los especificados
- techo (elemento) - devuelve el elemento más bajo entre los elementos que son mayores que el elemento especificado . Si el elemento pasado existe en un conjunto de árbol, devuelve el elementpasado como argumento.
- floor (elemento) – Devuelve el elemento más grande entre aquellos elementos que son menores que los especificados element. Si el elemento pasado existe en un conjunto de árbol, devuelve el elementpasado como argumento.
- pollFirst() – Devuelve y elimina el primer elemento del conjunto.
- pollLast – Devuelve y elimina el último elemento del conjunto
- headSet(elemento, true / false) – Devuelve todos los elementos de un conjunto de árbol antes del elemento especificado (que se pasa como argumento). El segundo argumento booleano es opcional y por defecto es false. Si se pone true el método devuelve todos los elementos antes del elemento especificado, incluido el elemento especificado.
- tailSet (elemento, valor booleano) - devuelve todos los elementos de un conjunto de árbol después del elemento especificado (que se pasa como parámetro), incluido el elemento especificado. Si el booleano es false se devuelve lo mismo que antes pero no se incluye el elemento especificado. Por defecto el valor booleano es true.
- subSet (e1, bv1, e2, bv2) - El subSet()método devuelve todos los elementos entre e1 y e2, incluido e1 .
  El bv1 y el bv2 son parámetros opcionales. El valor predeterminado de bv1 es true, y el valor predeterminado de bv2 es false.
  Si se pasa false se pasa como bv1 , el método devuelve todos los elementos entre e1 y e2 sin incluirlos e1.
  Si se pasa se pasa como bv2 , el método devuelve todos los elementos entre e1 y e2 , incluido e1 .
- retainAll(nombrecolección) – Se utiliza para realizar la intersección entre dos conjuntos
- removeAll(nombrecolección) – Se utiliza para sacar la diferencia de dos conjuntos

- containsAll() – Devuelve un valor booleano que nos indica si el conjunto que se indica entre paréntesis es parte del conjunto.
  EJ: boolean valor = A.containsAll(B);
- clone() – Crea una copia del TreeSet
- contains() - Busca en TreeSet el elemento especificado y devuelve un resultado booleano.
- isEmpty() – Comprueba si el TreeSet está vacído
- size() – Devuelve el tamaño del TreeSet
- clear() – Elimina todos los elementos del TreeSet