Class TreeSet

java.util
Class TreeSet

```
java.lang.Object
  └java.util.AbstractCollection
      └java.util.AbstractSet
          └java.util.TreeSet
```

Kılgıladığı arayüzler:
Cloneable, Collection, Serializable, Set, SortedSet

Bildirimi:
public class **TreeSet**
extends AbstractSet
implements SortedSet, Cloneable, Serializable

TreeSet sınıfı, Java Collections Framework 'un bir üyesidir

TreeSet sınıfı kılgıladığı arayüzlerin metotlarını kullanır. Doğal olarak öğelerine artan yönde sıralı erişim olur. Ama istenirse, erişim azalan yöne çevrilebilir. `add`, `remove` ve `contains` metotlarının zaman karmaşası $log(n)$ dir.

İlgili konular:
`Collection`, `Set`, `HashSet`, `Comparable`, `Comparator`, `TreeMap`, `Collections.synchronizedSortedSet(SortedSet)`, `Serialized Form`

Kurucular
`TreeSet``()` Öğelerini doğal sırada tutacak boş bir küme yaratır.
`TreeSet``(Collection c)` Verilen koleksiyonu TreeSet yapısına dönüştürür; dolayısıyla koleksiyon doğal sırasına girmiş olur
`TreeSet``(Comparator c)` Verile Comparator'a göre sıralı öğe tutacak boş bir küme yaratır.
`TreeSet``(SortedSet s)` Verilen sıralı kümedeki öğeleri aynı sırayla tutan bir TreeSet yaratır.

Metotlar	
`boolean`	**`add`**`(Object o)` Verilen öğe kümede yoksa, onu kümeye ekler.
`boolean`	**`addAll`**`(Collection c)`

	Verilen koleksiyonu kümeye ekler. Tabii yapı TreeSet yapısıdır.
void	**clear**() Kümenin bütün öğelerini siler; boş bir küme yaratır.
Object	**clone**() TreeSet nesnesinin bir kopyasını yaratır.
Comparator	**comparator**() Kümeyi sıralayacak comparatoru verir. Ancak kümede doğal sıra varsa, metot null değer verir.
boolean	**contains**(Object o) Aranan öğe kümede ise true değilse false değer verir.
Object	**first**() Sıralı kümenin en küçük öğesini verir.
SortedSet	**headSet**(Object toElement) Kümede toElement den küçük olan öğeleri verir.
boolean	**isEmpty**() Küme boş ise true verir.
Iterator	**iterator**() Kümenin öğelerini sırayla tarayacak iterator'ü verir
Object	**last**() Sıralı kümenin en büyük öğesini verir.
boolean	**remove**(Object o) Verilen öğeyi kümeden siler.
int	**size**() Kümedeki öğe sayısını verir.
SortedSet	**subSet**(Object fromElement, Object toElement) Sıralı kümede fromElement den başlayıp toElement öğesine kadar olan bütün öğeleri verir. Alt uç dahil, üst uç hariçtir.
SortedSet	**tailSet**(Object fromElement) Sıralı kümede fromElement ve sondan sonraki öğeleri (kuyruk) verir.

java.util.AbstractSet sınıfından kalıtımla gelen metotlar
equals, hashCode, removeAll

java.util.AbstractCollection sınıfından kalıtımla gelen metotlar
containsAll, retainAll, toArray, toArray, toString

java.lang.Object sınıfından kalıtımla gelen metotlar
finalize, getClass, notify, notifyAll, wait, wait, wait

java.util.Set arayüzünden kalıtımla gelen metotlar
containsAll, equals, hashCode, removeAll, retainAll, toArray, toArray

Örnek:

Aşağıdaki program Integer tipinden nesneler tutan bir TreeSet yaratıyor. Bu kümeye başka nesne tipinden öğe eklersek, koşma anında `java.lang.ClassCastException` hatası verir.

```
import java.util.*;

public class TreeSetDemo {

	public static void doTreeSetExample() {

		final int MAX = 10;
		System.out.println();

		Set ss = new TreeSet(); // TreeSet ambarını yaratır

		for (int i = 0; i < MAX; i++) {
		 System.out.println("  - Ambara konulan Integer :" + i);
		 ss.add(new Integer(i));
		}
		// tip uyuşmazlığı nedeniyle bu öğe eklenemez
		// ss.add((Object)"Kitap");

		System.out.println();

		Iterator i = ss.iterator(); // iterator yaratıyor
		// ambardaki öğeler yazdırılıyor
		while (i.hasNext()) {
			System.out.print("\t" + i.next());
		}
	}

	public static void main(String[] args) {
		doTreeSetExample();
	}

}
	/*
	Çıktı:
	  - Ambara konulan Integer :0
	  - Ambara konulan Integer :1
	  - Ambara konulan Integer :2
	  - Ambara konulan Integer :3
	  - Ambara konulan Integer :4
	  - Ambara konulan Integer :5
	  - Ambara konulan Integer :6
	  - Ambara konulan Integer :7
	  - Ambara konulan Integer :8
	  - Ambara konulan Integer :9

	 0	1	2	3	4	5	6	7	8	9
	*/



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

```
public class SetDemo {

	public static void doTreeSetExample() {
	}

	public static void main(String[] args) {
		// TreeSet ambarını yaratıyor
		Set treeSet = new TreeSet();
		// Öğeleri ambara yerleştiriyor
		for (int i = 10; i < 15; i++) {
			treeSet.add(new Integer(i));
		}
		// farklı tipten nesne girilirse koşma hatası doğar
		// treeSet.add("a string");
		System.out.print("TreeSet içindeki öğeler : ");

		Iterator i = treeSet.iterator();
		while (i.hasNext()) {
			System.out.print(i.next() + "\t");
		}
	}
}
	/*
	Çıktı:
	 TreeSet içindeki öğeler : 10  11    12    13    14
	*/
```

Örnek:

```
import java.util.*;

public class TreeSetExample{
 public static void main(String[] args) {
  System.out.println("TreeSet Example!\n");
  TreeSet <Integer>tree = new TreeSet<Integer>();
  tree.add(12);
  tree.add(23);
  tree.add(34);
  tree.add(45);
  Iterator iterator;
  iterator = tree.iterator();
  System.out.print("Tree set data: ");
  //Displaying the Tree set data
  while (iterator.hasNext()){
   System.out.print(iterator.next() + " ");
  }
  System.out.println();
  //Check impty or not
  if (tree.isEmpty()){
   System.out.print("Tree Set is empty.");
  }
```

```
  else{
   System.out.println("Tree Set size: " + tree.size());
  }
  //Retrieve first data from tree set
  System.out.println("First data: " + tree.first());
  //Retrieve last data from tree set
  System.out.println("Last data: " + tree.last());
  if (tree.remove(30)){
   System.out.println("Data is removed from tree set");
  }
  else{
   System.out.println("Data doesn't exist!");
  }
  System.out.print("Now the tree set contain: ");
  iterator = tree.iterator();
  //Displaying the Tree set data
  while (iterator.hasNext()){
   System.out.print(iterator.next() + " ");
  }
  System.out.println();
  System.out.println("Now the size of tree set: " + tree.size());
  //Remove all
  tree.clear();
  if (tree.isEmpty()){
   System.out.print("Tree Set is empty.");
  }
  else{
   System.out.println("Tree Set size: " + tree.size());
  }
 }
}
```

Örnek:

Aşağıdaki program `ArrayList` içindeki bütün öğeleri `TreeSet` koleksiyonuna eklemektedir.

`TreeSet`'e eklenen öğeler kendiliğinden sıralanır. Aşağıdaki örnekte, ilk altı sayının adları *`TreeSet`* içine, alfabetik sıraya konularak yerleştirilmiştir.

```
package koleksiyon;

import java.util.*;

public class Koleksiyon {

	public static void main(String[] args) {
		List a = new ArrayList();
		a.add("Bir");
```

```
            a.add("İki");
            a.add("Üç");
            a.add("Dört");
            a.add("Beş");
            a.add("Altı");
            System.out.println("ArrayList = " + a);

            Set ts = new TreeSet();
            ts.addAll(a);
            System.out.println("TreeSet = " + ts);
        }
}
```