Interface SortedSet

java.util
Interface SortedSet

Üst arayüzleri:
Collection, Set
Kılgılayan sınıflar:
TreeSet

Bildirimi:
public interface **SortedSet**
extends Set

SortedSet arayüzü, Java Collections Framework 'un bir üyesidir

Java.util.SortedSet arayüzü *java.util.Set* arayüzünün bir altarayüzüdür ve onun özeliklerini genişletir; yani *Set* arayüzündeki metotlara ek olarak aşağıdaki metotlara sahiptir.

Eğer *java.lang.Comparable* kılgılanmışsa koleksiyondaki öğeler doğal sırasıyla sıralanır. Değilse, oluşturulan *Comparator*'un belirlediği sırayla *iterator* tarafından küçükten büyüğe doğru taranır. Koleksiyona giren bütün öğeler *Comparable* arayüzünü kılgılamış olmalıdır. Dolayısıyla, koleksiyondaki her *(e1,e2)* öğe çifti

```
e1.compareTo(e2)
```

ya da

```
comparator.compare(e1, e2))
```

yöntemleriyle mukayese edilebilir olmalıdır. Koleksiyon içinde olmayan bir öğeye erişim istenirse bazı metotları *NoSuchElementException* hatası atar. Uyumsuz nesne ile karşılaşıldığında `ClassCastException` hatası atılır. *null* öğe kabul etmez ve null öğe ile karşılaşılırsa *NullPointerException* hatası atılır.

TreeSet.descendingIterator() ile `TreeSet` içindeki öğelere erişim büyükten küçüğe doğru yapılabilir.

Collections API içinde bu arayüzü kılgılayan bir tek *TreeSet* sınıfı vardır. Dolayısıyla, *SortedSet* arayüzünü kullanabilmek için *TreeSet* sınıfına başvurulur.

İlgili konular

`Set`, `TreeSet`, `SortedMap`, `Collection`, `Comparable`, `Comparator`, `ClassCastException`

Metotlar	
Comparator	**comparator**() Bu sıralı kümeyle ilişkili comparator'u verir. Eğer küme doğal sırasında ise, null değeri verir.
Object	**first**() Sıralı kümenin en küçük (ilk) öğesini verir.
SortedSet	**headSet**(Object toElement) Sıralı kümede `toElement` den küçük olanları verir.
Object	**last**() Sıralı kümenin en büyük (son) öğesini 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.Set arayüzünden kalıtımla gelen metotlar
add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray, toArray

Örnek:

TreeSet sınıfı *SortedSet*'in bir kılgısıdır (implementation) Dolayısıyla, *TreeSet* yapısına veri depo edilirken, öğeler doğal sıralarında yerleşirler; öğeler çağrılırken o sırada gelirler. Aşağıdaki program ***add()*** metodu ile *TreeSet* yapısına harfleri (öğe) eklerken, onları, doğal sıraları olan alfabetik sırasında ağaca (tree) yerleştirir ve erişimi o sırada yapar.

```
import java.util.Set;

import java.util.TreeSet;

public class TreeSetDemo01 {
	public static void main(String[] args) {
		Set<String> set = new TreeSet<String>();
		set.add("Z");
		set.add("K");
		set.add("F");
		set.add("M");
		set.add("A");
		set.add("X");
		set.add("P");

		for (String item : set) {
			System.out.print(item + " ");
		}
	}
```

```
}
```

```
/*
Çıktı:
A F K M P X Z
*/
```

Örnek:

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

public class SetDemo {

	public static void main(String[] args) {

		SortedSet<String> ss = new TreeSet<String>();

		ss.add("Gökhan");
		ss.add("Sabri");
		ss.add("Merve");
		ss.add("İpek");
		ss.add("Volkan");

		Iterator it = ss.iterator();

		while (it.hasNext()) {
			String value = (String) it.next();

			System.out.println("Değer :" + value);
		}
	}
}
```

```
/*
Çıktı:
 Değer :Gökhan
 Değer :Merve
 Değer :Sabri
 Değer :Volkan
 Değer :İpek
*/
```