Class LinkedHashSet

java.util
Class LinkedHashSet

```
java.lang.Object
  └ java.util.AbstractCollection
      └ java.util.AbstractSet
          └ java.util.HashSet
              └ java.util.LinkedHashSet
```

Kılgıladığı arayüzler

Cloneable, Collection, Serializable, Set

Bildirimi

public class **LinkedHashSet**
extends HashSet
implements Set, Cloneable, Serializable

`LinkedHashSet` sınıfı Java Collections Framework 'un bir üyesidir.

Set arayüzünün Hashtable ve LinkedList özeliklerini içeren bir kılgısıdır. Dolayısıyla, bu yapıda döngü sırası öngörülebilir. Bu yapının HashSet yapısından önemli farkı öğelerini çift yönlü bağ ile birbirlerine bağlamasıdır. Bağlı liste olduğu için, döngü sırası öğelerin bağlı listedeki konumlarıdır. Tabii, öğelerin konumu listenin yaratılışında yerleştikleri sıradır.

`LinkedHashSet` sınıfı HashSet sınıfının belirsiz sıralamasını önler, koleksiyonun öğelerine öngörülebilen bir sırada erişimi sağlar. Tabii, bu erişim sırasını, `HashSet` tapısını `Treeset` yapısına dönüştürerek de sağlayabiliriz. Ama, genellikle, `TreeSet` yapısını kurmanın karmaşası (complexity) daha çoktur. Örneğin bir *m* kümesinin `LinkedHashSet` yapısına dönüşmüş bir kopyasını elde etmek için

```
    void foo(Set m) {
    Set copy = new LinkedHashSet(m);
    ...
}
```

metodu yeterlidir. Bu dönüşümde, öğelerin sırası değişmez; yalnızca onlar arasında bağ (link) oluşur. O nedenle, `TreeSet` yapısına dönüştürmeye göre karamaşası daha azdır. Küme olarak, Collections içindeki herhangi bir yapı alınabilir.

Bu yapı Set işlemlerinin hepsine izin verir; null değer içerebilir.

İlgili konular:

`Object.hashCode()`, `Collection`, `Set`, `HashSet`, `TreeSet`, `Hashtable`, Serialized Form

LinkedHashSet sınıfının Kurucuları
LinkedHashSet() Başlangıç sığası 16 ve yükleme çarpanı 0,75 olan boş bir LinkedHashSet yaratır.
LinkedHashSet(Collection c) Verilen koleksiyonu bir LinkedHashSet yapısına dönüştürür.
LinkedHashSet(int initialCapacity) Başlangıç sığası belirtildiği kadar ve yükleme çarpanı 0,75 olan boş bir LinkedHashSet yaratır.
LinkedHashSet(int initialCapacity, float loadFactor) Başlangıç sığası ve yükleme çarpanı belirtildiği kadar olan boş bir LinkedHashSet yaratır.

java.util.HashSet sınıfından kalıtsal gelen metotlar
add, clear, clone, contains, isEmpty, iterator, remove, size

class java.util.AbstractSet sınıfından kalıtsal gelen metotlar
equals, hashCode, removeAll

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

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

java.util.Set arayüzünden gelen metotlar
add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray, toArray

Örnek:

Aşağıdaki program bir `LinkedHashSet` yapısı kuruyor. Bu yapı iki yönlü bağlı bir listedir. Dolayısıyla, erişim sıralıdır ve öğelerin giriş sırasındadır. Yapıya bir öğe eklemek için `add(Object obj)` metodu kullanılır.

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

public class LinkedHashSetTest {

	public static void main(String[] args) {
		LinkedHashSet obj = new LinkedHashSet();

		obj.add("Samsun");
		obj.add("Adana");
		obj.add("Konya");
		obj.add("İzmit");

		Set e = obj;

		for (Iterator i = e.iterator(); i.hasNext();) {
			System.out.println(i.next().toString());
		}
	}
}
	/*
	Çıktı:
	Samsun
	Adana
	Konya
	İzmit
	*/
```

Örnek:

Aşağıdaki program `LinkedHashSet` yapısında iterator ile döngü yapılışını göstermektedir. Yapıda varolan bir öğe tekrar eklenemez. Yapıya ilkel veri tipleri eklenemediği için, onlar önce ilgili sınıflara gömülür (wrapping), sonra yapıya eklenir.

```
import java.util.LinkedHashSet;
import java.util.Iterator;

public class LinkedHashSetTest {

	public static void main(String[] args) {

		// LinkedHashSet nesnesi yarat
		LinkedHashSet lhashSet = new LinkedHashSet();

		// LinkedHashSet nesnesine öğeler ekle
		lhashSet.add(new Integer("33"));
		lhashSet.add(new Integer("22"));
		lhashSet.add(new Integer("11"));
		lhashSet.add(new Integer("33"));
```

```
            // Iterator bildirimi
            Iterator itr = lhashSet.iterator();

            System.out.println("LinkedHashSet 'in öğeleri : ");
            while (itr.hasNext())
                  System.out.println(itr.next());
      }
}
   /*
    LinkedHashSet 'in öğeleri :
    33
    22
    11
   */
```

Örnek:

Aşağıdaki program *Java Collections Framework* içindeki farklı yapıları kurmaktadır.

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.TreeMap;
import java.util.TreeSet;

public class LinkedHashSetTest {
      public static void main(String[] args) {
            List list = new LinkedList();
            list.add("Zeynep");
            list.add("Çağlar");
            list.add("Volkan");
            list.add("Berna");
            System.out.println("List : " + list);
            // print(list);

            Set s = new HashSet();
            s.add("Papatya");
            s.add("Lale");
            s.add("Sümbül");
            s.add("Kardelen");
            System.out.println("Set  : " + s);

            SortedSet ss = new TreeSet();
            ss.add("Jale");
            ss.add("Semra");
            ss.add("Melek");
            ss.add("Semra");
            System.out.println("SortedSet     : " + ss);
```

```
        LinkedHashSet sss = new LinkedHashSet();
        sss.add("Irmak");
        sss.add("Göl");
        sss.add("Deniz");
        sss.add("Nehir");
        System.out.println("LinkedHashSet : " + sss);

        Map m1 = new HashMap();
        m1.put("Motzart", "Müzisyen");
        m1.put("Kant", "Filozof");
        m1.put("Poincare", "Matematikçi");
        m1.put("Shakespeare", "Yazar");
        System.out.println("keySet() : " + m1.keySet());
        System.out.println("values() : " + m1.values());

        SortedMap sm = new TreeMap();
        sm.put("Motzart", "Müzisyen");
        sm.put("Kant", "Filozof");
        sm.put("Poincare", "Matematikçi");
        sm.put("Shakespeare", "Yazar");
        System.out.print("keySet() : ");
        print(sm.keySet());
        System.out.print("values() : ");
        print(sm.values());

        LinkedHashMap lm = new LinkedHashMap();
        lm.put("Motzart", "Müzisyen");
        lm.put("Kant", "Filozof");
        lm.put("Poincare", "Matematikçi");
        lm.put("Shakespeare", "Yazar");
        System.out.print("keySet() : ");
        print(lm.keySet());
        System.out.print("values() : ");
        print(lm.values());
    }

    static void print(Collection coll) {
        Iterator iter = coll.iterator();
        while (iter.hasNext()) {
            String elem = (String) iter.next();
            System.out.print(elem + " ");
        }
        System.out.println();
    }

}
/*
Çıktı:
 List : [Zeynep, Çağlar, Volkan, Berna]
 Set  : [Kardelen, Papatya, Sümbül, Lale]
 SortedSet     : [Jale, Melek, Semra]
 LinkedHashSet : [Irmak, Göl, Deniz, Nehir]
 keySet() : [Kant, Poincare, Shakespeare, Motzart]
 values() : [Filozof, Matematikçi, Yazar, Müzisyen]
 keySet() : Kant Motzart Poincare Shakespeare
 values() : Filozof Müzisyen Matematikçi Yazar
 keySet() : Motzart Kant Poincare Shakespeare
 values() : Müzisyen Filozof Matematikçi Yazar
*/
```