Class LinkedHashMap

java.util

Class LinkedHashMap

```
java.lang.Object
  └java.util.AbstractMap
      └java.util.HashMap
          └java.util.LinkedHashMap
```

Kılgıladığı arayüzler:

Cloneable, Map, Serializable

Bildirimi:

public class **LinkedHashMap**
extends HashMap

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

`Map` arayüzünün `Hashtable` ve `LinkedList` özeliklerini içeren bir kılgısıdır. Dolayısıyla, bu yapıda döngü sırası öngörülebilir. Bu yapının `HashMap` 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.

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

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

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

Bu yapı `Map` işlemlerinin hepsine izin verir; `null` değer içerebilir.

İlgili konular:

`Object.hashCode()`, `Collection`, `Map`, `HashMap`, `TreeMap`, `Hashtable`, Serialized Form

Kurucular

LinkedHashMap()
Başlangıç sığası 16 ve yükleme çarpanı 0,75 olan boş bir `LinkedHashMap` yaratır. Erişim sırası giriş sırasıdır.

LinkedHashMap(int initialCapacity)
Başlangıç sığası belirtildiği kadar ve yükleme çarpanı 0,75 olan boş bir `LinkedHashMap` yaratır. Erişim sırası giriş sırasıdır.

LinkedHashMap(int initialCapacity, float loadFactor)
Başlangıç sığası ve yükleme çarpanı belirtildiği kadar olan boş bir `LinkedHashMap` yaratır. Erişim sırası giriş sırasıdır.

LinkedHashMap(int initialCapacity, float loadFactor, boolean accessOrder)
Başlangıç sığası ve yükleme çarpanı belirtildiği kadar olan boş bir `LinkedHashMap` yaratır. Erişim sırası belirtilen sıradır.

LinkedHashMap(Map m)
Verilen Map için giriş sıralı bir `LinkedHashMap` yaratır.

.

Method Summary

void	**clear**() MAp'in bütün öğelerini siler.
boolean	**containsValue**(Object value) Belirtilen öğeye gönderilen bir ya da daha çok anahtar varsa `true` değerini verir.
Object	**get**(Object key) Verilen anahtara eşlenen öğeyi verir.
protected boolean	**removeEldestEntry**(Map.Entry eldest) En eski girilen öğeyi silecekse `true` değerini verir.

java.util.HashMap sınıfından kalıtsal alınan metotlar
clone, containsKey, entrySet, isEmpty, keySet, put, putAll, remove, size, values

java.util.AbstractMap sınıfından kalıtsal alınan metotlar
equals, hashCode, toString

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

java.util.Map arayüzünden alınan metotlar
equals, hashCode

Örnek:

Aşağıdaki program bir `LinkedHashMap` yapısı kuruyor. Bu yapı iki yönlü bağlı bir listedir. Dolayısıyla, erişim sıralıdır ve öğelerin giriş sırasındadır. Yapıya bir öğe eklemek için `put(Object key, Object value)` metodu kullanılır. Yapının öğelerine erişmek için iterator kullanılmaktadır.

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Set;

class LinkedHashMapTest {
      public static void main(String args[]) {
            // LinkedHashMap nesnesi yarat
            LinkedHashMap lhm = new LinkedHashMap();
            // Öğe ekle
            lhm.put("Zeynep", new Double(34.34));
            lhm.put("Melek",  new Double(53.22));
            lhm.put("Ayhan",  new Double(378.00));
            lhm.put("Deniz",  new Double(49.22));
            lhm.put("Cansu",  new Double(59.08));

            // Girilen öğeler:
            Set set = lhm.entrySet();
            // iterator bildirimi
            Iterator i = set.iterator();
            // Öğeleri göster
            while (i.hasNext()) {
```

```
                Map.Entry me = (Map.Entry) i.next();
                System.out.print(me.getKey() + ": ");
                System.out.println(me.getValue());
            }
            System.out.println();
            // Zeynep'in puanını düzelt
            double puan = ((Double) lhm.get("Zeynep")).doubleValue();
            lhm.put("Zeynep", new Double(puan + 24));
            System.out.println("Zeynep'in puanı: " + lhm.get("Zeynep"));
      }
}
      /*
      Çıktı:
       Zeynep: 34.34
       Melek: 53.22
       Ayhan: 378.0
       Deniz: 49.22
       Cansu: 59.08

       Zeynep'in puanı: 58.34
      */
```

Örnek:

Aşağıdaki program `LinkedHashMap` 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.Set;
import java.util.LinkedHashMap;
import java.util.Iterator;

public class LinkedHashMapTest {

      public static void main(String[] args) {

            // LinkedHashMap nesnesi yarat
            LinkedHashMap lhm = new LinkedHashMap();

            // LinkedHashSet nesnesine öğeler ekle
            lhm.put("İçel", new Integer("33"));
            lhm.put("Edirne", new Integer("22"));
            lhm.put("Bilecik", new Integer("11"));
            lhm.put("İçel", new Integer("33"));

            // Girilen öğeler kümesi
            Set set = lhm.entrySet();
            // iterator bildirimi
            Iterator itr = set.iterator();
            System.out.println(lhm);
            System.out.println("LinkedHashMap 'in öğeleri : ");
            while (itr.hasNext())
                  System.out.println(itr.next());
      }
}
   /*
    {İçel=33, Edirne=22, Bilecik=11}
    LinkedHashMap 'in öğeleri :
```

```
İçel=33
Edirne=22
Bilecik=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
    */
```