Class Hashtable

java.util
Class Hashtable<K,V>

java.lang.Object
  └ java.util.Dictionary<K,V>
    └ **java.util.Hashtable<K,V>**

Kılgıladığı Arayüzler:
Serializable, Cloneable, Map<K,V>

Altsınıfları:
Properties, UIDefaults

Bildirimi:

public class **Hashtable<K,V>**
extends Dictionary<K,V>
implements Map<K,V>, Cloneable, Serializable

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

Bu sınıf anahtarları değerlere göndererek bir hash table yaratır. `Null` olmayan her nesne anahtar ya da değer olarak kullanılabilir.

Bir `Hashtable` yapısına nesneleri depolamak ve onlara erişimi sağlamak için, anahtar olarak kullanılan nesnelerin `hashCode()` metodu ile `equals()` metodunu kılgılıyor olması gerekir.

Bir `Hastable` yapısının performansını etkileyen iki parametre vardır: *başlangıç sığası* ve *yükleme katsayısı*. `Hashtable` içindeki sepet (bucket) sayısı onun sığasıdır (capacity); yani sahip olduğu bileşen sayısıdır. Başlangıç sığası, `Hashtable` yaratılırken belirlenen bileşen sayısıdır. *Yükleme katsayısı* ise, sığasının ne kadarı dolduğunda otomatik olarak sığa artımına geçeceğini belirleyen orandır. Genellikle, yükleme katsayısı 0,75 dir. Bu demektir ki, `Hashtable` bileşenleri %75 oranında dolduğunda, sığa otomatik olarak artar. Tabii, bir `Hashtable` kurulurken, istenirse sığası ve yükleme katsayısı belirlenebilir.

İlgili Konular:
Object.equals(java.lang.Object), Object.hashCode(), rehash(), Collection, Map, HashMap, TreeMap, Serialized Form

---

`Hashtable` **Sınıfının Kurucuları**
Hashtable() Başlangıç sığası 11 ve yükleme katsayısı 0,75 olan boş bir Hashtable yaratır.
Hashtable(int initialCapacity) Başlangıç sığası parametrenin belirlediği sayı ve yükleme katsayısı 0,75 olan boş bir `Hashtable` yaratır.
Hashtable(int initialCapacity, float loadFactor)

Başlangıç sığası ve yükleme katsayısı parametrelerle belirlenen sayılar olan boş bir `Hashtable` yaratır.
Hashtable(Map<? extends K,? extends V> t) Verilen Map ile aynı gönderime sahip bir `Hashtable` yaratır

Method Summary	
void	**clear**() Hiçbir anahtar içermeyecek biçimde Hashtable yapısını temizler.
Object	**clone**() Hashtable nesnesinin boş bir kopyasını yaratır. Anahtarları ve değerleri kopyalamaz.
boolean	**contains**(Object value) Verilen öğeye eşleşen bir ya da birden çok anahtar varsa `true` değeri verir.
boolean	**containsKey**(Object key) Verilen anahtara eşleşen bir gönderim varsa `true` değerini verir.
boolean	**containsValue**(Object value) Verilen öğeye eşleşen bir ya da birden çok anahtar varsa `true` değeri verir.
Enumeration<V>	**elements**() Hashtable içindeki değerleri numaralar.
Set<Map.Entry<K,V>>	**entrySet**() Yapıdaki gönderimleri bir küme olarak verir.
boolean	**equals**(Object o) Verilen nesnenin, parametredeki nesneye eşit olup olmadığını bulur.
V	**get**(Object key) Verilen anahtara eşleşen değeri verir. Eğer verilen anahtara eşleşen öğe yoksa `null` verir.
int	**hashCode**() Map'in hash kodunu verir..
boolean	**isEmpty**() Hashtable 'in boş olup olmadığını denetler.
Enumeration<K>	**keys**() Hashtable içindeki anahtarları sıralar.
Set<K>	**keySet**() Hashtable içindeki anahtarları bir küme olarak verir.
V	**put**(K key, V value) Verilen anahtarrı verilen öğeye eşler.
void	**putAll**(Map<? extends K,? extends V> t) Verilen map'i belirtilen map'e kopyalar.
protected void	**rehash**() Sığayı artırır.
V	**remove**(Object key)

	Verilen anahtarı ve ona eşlenen değeri siler.
int	**size**() Map içindeki anahtar sayısını verir. O sayı, dolan bileşen (bucket) sayısıdır..
String	**toString**() Hashtable içindeki nesnelerin bir string temsilini verir. Bu temsil, değerleri metin olarak yazar ve onları küme simgesi ([]) içinde verir.
Collection<V>	**values**() Map içindeki değerleri bir *Collection* olarak gösterir.

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

Örnek:

```
import java.util.Hashtable;
import java.util.Enumeration;

public class HashtableDemo {

public static void main(String[] args) {

Hashtable<Integer, String> hTable = new Hashtable<Integer,
String>();

        // Hashtable nesnesine öğe ekleme
        hTable.put(new Integer(474), "Kars");
        hTable.put(new Integer(376), "Balıkesir");
        hTable.put(new Integer(322), "Ankara");
        hTable.put(new Integer(232), "İzmir");
        hTable.put(new Integer(284), "Edirne");
        // Hashtable öğelerini yazdır
        System.out.println(hTable);
        // anahtar ve değerlere erişmek için Hashtable sıralaması
        Enumeration em = hTable.keys();

        while (em.hasMoreElements()) {
              // Hashtable'ın öğelerine erişim
              int key = (Integer) em.nextElement();

              // değere erişim
 String value = (String) hTable.get(key);
 System.out.println("Anahtar :" + key + "  değer :" + value);
        }
 }
}
/*
Çıktı:
 {284=Edirne, 322=Ankara, 376=Balıkesir, 232=İzmir, 474=Kars}
 Anahtar :284  değer :Edirne
```

```
 Anahtar :322  değer :Ankara
 Anahtar :376  değer :Balıkesir
 Anahtar :232  değer :İzmir
 Anahtar :474  değer :Kars
*/
```

```
import java.util.*;

public class HashtableDemo {
      public static void main(String args[]) throws Exception {
            // sığayı 10 ile başlat; dolunca artır
            Hashtable hash = new Hashtable(10, 10);

            for (int i = 0; i <= 100; i++) {
                  Integer integer = new Integer(i);
                  hash.put(integer, "Sayı : " + i);
            }
            System.out.println(hash);
            // değere erişim
            System.out.println(hash.get(new Integer(5)));

            // değere erişim
            System.out.println(hash.get(new Integer(21)));

            System.in.read();

            // bütün değerler
            for (Enumeration e = hash.keys(); e.hasMoreElements();) {
                  System.out.println(hash.get(e.nextElement()));
            }
       }
      }
      /*
      Çıktı:
      {20=Sayı : 20, 41=Sayı : 41, 62=Sayı : 62, 83=Sayı : 83, 40=Sayı : 40,
      61=Sayı : 61, 82=Sayı : 82, 19=Sayı : 19, 60=Sayı : 60, 81=Sayı : 81,
      …
      63=Sayı : 63, 84=Sayı : 84}
      Sayı : 5
      Sayı : 21
      */
```

Örnek:

Aşağıdaki program bir `Hashtable` kuruyor ve onun üzerinde bir iterator tanımlayarak öğelerine erişiyor.

```
import java.util.Enumeration;
import java.util.Iterator;
import java.util.Hashtable;
import java.util.Collection;

public class HashtableDemo {
```

```
	public static void main(String[] args) {

		// Hashtable nesnesi yarat
		Hashtable ht = new Hashtable();

		// Hashtable nesnesine öğe ekle
		ht.put("1", "Medrese");
		ht.put("2", "Darulfünun");
		ht.put("3", "Üniversite");
		Collection c = ht.values();

		System.out.println("Hashtable'in öğeleri :");
		// koleksiyon üzerinde Iterator
		Iterator itr = c.iterator();
		while (itr.hasNext())
			System.out.println(itr.next());
		// koleksiyondan sil
		c.remove("Medrese");

		// Hashtable öğeleri
		System.out.println("Hashtable geri kalan öğeleri :");
		Enumeration e = ht.elements();
		while (e.hasMoreElements())
			System.out.println(e.nextElement());
	}
}

	/*
	Çıktı:
	 Hashtable'in öğeleri :
	 Üniversite
	 Darulfünun
	 Medrese
	 Hashtable geri kalan öğeleri :
	 Üniversite
	 Darulfünun
	*/
```

Örnek:

Aşağıdaki program ülkeleri, telefon kodları ile eşleştiren bir `Hashtable` kuruyor ve başlıca `Hashtable` metotlarını uyguluyor.

```
import java.util.Hashtable;
import java.util.Enumeration;

public class HashtableDemo {

	public static void main(String[] args) {

		Hashtable<Integer, String> hTable = new Hashtable<Integer,
String>();
		// Hashtable<Integer, String> ht = new Hashtable<Integer,
String>();

		// Hashtable nesnesine öğe ekleme
		// Ülkelerin telefon kodları
```

```
            hTable.put(new Integer(1), "ABD");
            hTable.put(new Integer(973), "Bahreyn");
            hTable.put(new Integer(45), "Danimarka");
            hTable.put(new Integer(49), "Almanya");
            hTable.put(new Integer(44), "İngiltere");
            // Hashtable öğelerini yazdır
            System.out.println(hTable);
            System.out.println("contains(\"Kars\")        : "
                    + hTable.contains("Kars"));
            System.out.println("containsValue(\"Ankara\") : "
                    + hTable.containsValue("Kars"));
            System.out.println("containsKey(474)        : "
                    + hTable.containsKey(474));
            System.out.println("elements()              : " +
hTable.elements());
            System.out.println("entrySet()              : " +
hTable.entrySet());
            System.out.println("equals()                : "
                    + "İzmir".equals("İzmir"));
            System.out.println("equals()                : "
                    + "İzmir".equals("Mersin"));
            System.out.println("get(284)                : " +
hTable.get(474));
            System.out.println("hashCode()              : " +
hTable.hashCode());
            System.out.println("isEmpty()               : " +
hTable.isEmpty());
            System.out.println("keys()                  : " +
hTable.keys());
            System.out.println("keySet()                : " +
hTable.keySet());
            System.out.println("size()                  : " +
hTable.size());
            System.out.println("toString()              : " +
hTable.toString());
            System.out.println("values()                : " +
hTable.values());

            Hashtable ht = (Hashtable) hTable.clone();
            System.out.println("println(ht)             : " + ht);
            System.out.println("ht.size()               : " + ht.size());

      }
}
/*
Çıktı:
{49=Almanya, 973=Bahreyn, 45=Danimarka, 1=ABD, 44=İngiltere}
contains("Kars")        : false
containsValue("Ankara") : false
containsKey(474)        : false
elements()              : java.util.Hashtable$Enumerator@42e816
entrySet()  : [49=Almanya, 973=Bahreyn, 45=Danimarka, 1=ABD,  44=İngiltere]
equals()                : true
equals()                : false
get(284)                : null
hashCode()              : 1686308881
isEmpty()               : false
keys()                  : java.util.Hashtable$Enumerator@9304b1
keySet()                : [49, 973, 45, 1, 44]
size()                  : 5
toString()  : {49=Almanya, 973=Bahreyn, 45=Danimarka, 1=ABD, 44=İngiltere}
values()                : [Almanya, Bahreyn, Danimarka, ABD, İngiltere]
println(ht) : {49=Almanya, 973=Bahreyn, 45=Danimarka, 1=ABD, 44=İngiltere}
```

```
ht.size()               : 5
*/
```

Örnek:

Aşağıdaki prgram bir `TreeMap` Map'ini `Hashtable` Map'ine dönüştürüyor.

```
import java.util.*;

public class HashtableDemo {
      public static void main(String[] args) {
            // Bir TreeMap nesnesi yarat:
            TreeMap tm = new TreeMap();
            // Bir Hashtable nesnesi yarat
            Hashtable ht = new Hashtable();

            // TreeMap tm deposuna veri gir
            tm.put("1", "Ankara");
            tm.put("2", "Paris");

            // tm map'inin bütün öğelerini ht mapi'ne kopyala
            ht.putAll(tm);
            // ht map'ini yazdır;
            System.out.println(ht);

            // Her iki map'in ikinci öğelerini yazdır:
            System.out.println("tm içindeki #2 öğe : " + tm.get("2"));
            System.out.println("th içindeki #2 öğe : " + ht.get("2"));
       }
     }
```