Class HashMap

java.util
Class HashMap
java.lang.Object
  └ java.util.AbstractMap
    └ **java.util.HashMap**

Kılgıladığı Arayüzler:

Cloneable, Map, Serializable

Altsınıfları

LinkedHashMap, PrinterStateReasons

Bildirimi:

public class **HashMap**
 extends AbstractMap
  implements Map, Cloneable, Serializable

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

Hash tablosu için `Map` arayüzünün kılgılanması gerekir. Bu oluşum seçimlik (optional) bütün map işlemlerine, *null* değerlere ve *null* anahtara izin verir. Senkronize olmadığı ve null değer alabilmesi dışındaki, `HashMap` sınıfının `Hashtable`'a denk olduğu söylenebilir. Tabii, bu sınıf gönderimin (map) sırasını garanti edemez. Dolayısıyla, farklı zamanlarda öğelerine erişim sırası değişebilir.

Uygulamada, hash fonksiyonlarının öğeleri sepetlere (bucket) dengeli dağıttığı varsayılırsa, `get` ve `put` işlemlerinin performansı sabit-zaman gerektirir. Kılgılanan `HashMap` sınıfının öğelerine erişim zamanı, sepet sayısı ve yaratılan nesnenin sığası (öğe sayısı) ile doğru orantılıdır. O nedenle, başlangıç sığasının gereksiz yere büyük tutulmaması ve yükleme katsayısının çok küçük tutulmaması önerilir.

`HashMap` kılgısının performansını etkileyen iki şey vardır: *Başlangıç sığası* (initial capacity) ve *yükleme katsayısı* (load factor). Bir `HashMap` kılgısının sığası (capacity) hash tablosundaki sepet (bucket) sayısıdır. Başlangı sığası ise, hash tablosu yaratılırken belirlenen sığadır. `Yükleme katsayısı` (load factor) ise, hash tablosunun sığasının yüzde kaçı dolduğunda sığanın otomatik olarak değişeceğini belirten orandır. Hash tablosuna girilen öğe sayısı yükleme katsayısını aştığında, rehash() metodu otomatik olarak çağrılır; bu metot hash tablosunun sığasını bir kat artırır.

Bir çok uygulamada harcanan zaman zaman ve yer (bellek alanı) arasında iyi bir denge kurmak için yükleme katsayısının 0,75 olarak seçilmesi önerilir. Doğal olarak yükleme katsayısının daha büyük seçilmesi put get metotları gibi bazı `HashMap` işlemlerinin harcadığı zamanı azaltır, ama *rehash()* metodunun devreye girip sığa büyütme ve küçültme işlemlerini yapmasını geciktirir. O nedenle, başlangıç sığasının seçimi önem kazanır. *rehash()* metoduna gerek bırakmayacak ama belleği boşuna harcamayan bir başlangıç sığasının seçimi performansı artıracaktır.

İlgili Konular:

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

HashMap Sınıfının Kurucuları
HashMap()   Başlangıç sığası 16 ve yükleme çarpanı 0.75 olan boş bir *HashMap* kılgılar.
HashMap(int initialCapacity)   Başlangıç sığası belirtilen sayıda ve yükleme çarpanı 0.75 olan boş bir *HashMap* kılgılar.
HashMap(int initialCapacity, float loadFactor)   Başlangıç sığası ve yükleme çarpanı belirtilen sayılarda olan boş bir *HashMap* kılgılar.
HashMap(Map m)   Verilen Map ile belirlenen yeni bir HashMap yaratır.

HashMap Sınıfının Metotları	
void	**clear**() İlgili map yapısındaki bütün öğeleri siler.
Object	**clone**() HashMap nesnesinin bir kopyasını yaratır; ancak anahtarlar ve değerler kopyalanmaz.
boolean	**containsKey**(Object key) Belirtilen anahtar küme içinde ise true değerini verir.
boolean	**containsValue**(Object value) Belirtilen öğeye gönderilen (map edilen) anahtar varsa true değerini verir.
Set	**entrySet**() İlgili koleksiyondaki gönderimleri (map) gösterir.
Object	**get**(Object key) Verilen anahtarın hangi öğeye gönderildiğini (map) belirtir. Eğer anahtarın gönderildiği hiçbir öğe yoksa, metot null değer verir.
boolean	**isEmpty**() İlgili map içinde hiçbir anahtar-değer eşleşmesi yoksa, metot true değerini

	verir.
`Set`	**`keySet`**`()` İlgili map içindeki anahtarları gösterir.
`Object`	**`put`**`(Object key, Object value)` Verilen anahtarı verilen değere göndererek anahtar-değer eşleştirimini yapar.
`void`	**`putAll`**`(Map m)` Verilen gönderimlerin hepsini belirtilen gönderime kopyalar.
`Object`	**`remove`**`(Object key)` Belirtilen anahtara karşılık gelen gönderim varsa, onu siler.
`int`	**`size`**`()` Map içindeki anahtar-değer sayısını verir.
`Collection`	**`values`**`()` Map içindeki değer sayısını verir.

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 kalıtsal alınan metotlar
`equals, hashCode`

Açıklamalar:

Aşağıdaki programlarda yapılan işlemleri topluca özetlemekte yarar olabilir.

Öntanımlı sıgası (default size) (16) ve öntanımlı yükleme katsayısı (default load factor) 0,75 olan boş bir `HashMap` yaratmak için aşağıdaki kurucu kullanılır:

```
HashMap hm = new HashMap();
```

Eğer başlangıç sıgasının öntanımlı olan 16 dan farklı olması isteniyorsa aşağıdakine benzer bir deyim yazılabilir. Bu deyim, başlangıç sığası 125 olan boş bir `HasMap` yaratır:

```
HashMap HashMap = new HashMap(125);
```

Eğer başlangıç sıgasının öntanımlı olan 16 dan farklı olması ve yükleme katsayısının öntanımlı 0,75 ten farklı olması isteniyorsa aşağıdaki deyim yazılabilir. Bu deyim, başlangıç sığası 125, ve yükleme katsayısı 0,85 olan boş bir `HasMap` yaratır:

```
HashMap(125, 0.85);
```

Varolan bir `Map` 'ten yeni bir `HashMap` yaratılmak isteniyorsa aşağıdaki deyim kullanılır:

```
HashMap HashMap = new HashMap(Map vMap);
```

`HasMap` koleksiyonuna ilkel veri tipleri eklenemez. İlkel veri tipleri eklenecekse, onlar önce ilgili sınıflarına gömülmelidirler. Bunun için, örneğin,

```
hashMap.put("Bir", new Integer(1));
```

deyimi yazılır. Bu deyim, int tipinden 1 tamsayısını (ilkel tiptir) Integer(1) deyimi ile

```
Integer(1);
```

Deyimi int ilkel veri tipindeki 1 tamsayısını Integer sınıfına gömer. Sonra

```
hashMap.put("Bir", new Integer(1));
```

deyimi "Bir" anahtarını Integer(1) değerine eşler (map).

Herhangi bir `Map` koleksiyonunun bütün öğelerini verilen bir `HashMap` yapısına dönüştürmek için

```
void putAll(Map m)
```

metodu kullanılır.

Bir `HashMap` koleksiyonu içinde kaç tane anahtar-değer çifti olduğunu bulmak için `HashMap` sınıfı içindeki

```
size()
```

metodu kullanılır.

Bir `HashMap` koleksiyonunun boş olup olmadığını bulmak için

```
isEmpty()
```

metodu kullanılır. Bu metot, `HashSet` boş ise `true`, değilse `false` değerini alır.

Bir `HashMap` koleksiyonu içinde belirli bir değerin olup olmadığını bulmak için,

```
containsValue(aranan_nesne)
```

metodu kullanılır. Bu metot,aranan değer `HashSet` içinde ise `true`, değilse `false` değerini alır.

Bir `HashMap` koleksiyonu içinde belirli bir anahtarın olup olmadığını bulmak için,

```
containsValue(aranan_anahtar)
```

metodu kullanılır. Bu metot,aranan anahtar `HashSet` içinde ise `true`, değilse `false` değerini alır.

Bir `HashMap` koleksiyonu içinde belirli bir anahtara eşleştirilen değeri bulmak için olup olmadığını bulmak için,

```
get(anahtar)
```

metodu kullanılır. Bu metot, verilen anahtara eşleşen öğeyi verir.

Bir `HashMap` koleksiyonu içinde olan bütün anahtarları bulmak için, Set sınıfı içindeki

```
keySet()
```

metodu kullanılır. Bu metot, verilen anahtara eşleşen öğeyi verir.

Bir `HashMap` koleksiyonu içinde olan bütün anahtarları bulmak için, Set sınıfı içindeki

```
entrySet()
```

metodu kullanılır. Bu metot, verilen anahtara eşleşen öğeyi verir.

Bir `HashMap` koleksiyonu içinde anahtarı verilen bir öğeyi silmek için

```
remove(anahtar)
```

metodu kullanılır.

Örnek:

Aşağıdaki program büyük harfleri değer olarak alıyor ve onların her birine ASCII kodunu anahtar olarak eşliyor.

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

public class HashMapDemo {

	public static void main(String[] args) {
		Map<Integer, Character> m = new HashMap<Integer, Character>();

		for (int i = 65; i <= 90; i++) {
			m.put(i, (char) i);
		}

		Set<Integer> kSet = m.keySet();

		Iterator<Integer> sayaç = kSet.iterator();

		while (sayaç.hasNext()) {
			Integer anahtar = sayaç.next();
			// System.out.print(anahtar + "    ");
```

```
                    // System.out.println(m.get(anahtar));
                    System.out.print(anahtar + "-" + m.get(anahtar) + "\t");
                }
        }
}

   /*
   Çıktı:
      68-D    69-E   70-F   71-G   65-A   66-B   67-C   76-L   77-M   78-N   79-O   72-H
      73-I    74-J   75-K   85-U   84-T   87-W   86-V   81-Q   80-P   83-S   82-R   89-Y
      88-X    90-Z
   */
```

Açıklamalar:

`Map` tipinden *m* nesnesi `Integer` ile `Character` tiplerini eşleyen bir `HashMap` nesnesidir.

`m.put(i, (char) i)` metodu ASCII kodu *i* olan harfe onun ASCII kodunu anahtar olarak gönderir.

`Set<Integer> kSet = m.keySet()` deyimi ile tanımlanan kSet değişkeni *m* içindeki anahtarları (key) belirler.

`Iterator<Integer> sayaç = kSet.iterator()` deyimi ile tanımlanan *sayaç* değişkeni tamsayılar üzerinde bir *iterator* (tekrarlayıcı) tanımlar.

while `(sayaç.hasNext())` döngüsü, kontrolün o anda bulunduğu *sayaç* 'dan sonra başka *sayaç* varsa `true` değerini alır ve kontrolü sonraki *sayaç* 'a geçirir.

`System.out.print(anahtar)` deyimi kSet içindeki öğeleri (anahtar) yazar.

`System.out.print(m.get(anahtar))` deyimi m.HashSet koleksiyonu içinde *anahtar* 'ın eşlendiği değeri yazar.

Örnek:

Aşağıdaki bir HashMap örneğidir.

```
import java.util.*;
public class HashMap01 {
public static void main(String[] args) {
      Map m1 = new HashMap();
      m1.put("Bilim", "12");
      m1.put("Sanat", "3");
      m1.put("Edebiyat", "5");
      m1.put("Siyaset", "9");
      System.out.println();
      System.out.println(" Map Oğeleri:");
      System.out.print("\t" + m1);
}
}
     /*
     Çıktı:
      Map Oğeleri:
      {Sanat=3, Siyaset=9, Bilim=12, Edebiyat=5}
     */
```

Örnek:

Aşağıdaki program, öğrenci numaralarına öğrencilerin adlarını eşliyor.

```
import java.util.*;

public class HashMapDemo {

      public static void main(String[] args) {

            HashMap hm = new HashMap();
            hm.put("MERVE ALATLI", new Integer(20895548));
            hm.put("AYGÜN DAMLA", new Integer(20894820));
            hm.put("BÜYÜKKILIÇ AYKUT", new Integer(20893085));
            hm.put("CAN FEHİME", new Integer(20793172));
            hm.put("CANER HALİL İBRAHİM", new Integer(20393385));
            Set set = hm.entrySet();

            Iterator i = set.iterator();

            while (i.hasNext()) {
              Map.Entry me = (Map.Entry) i.next();
              System.out.println(me.getKey() + " : " + me.getValue());
            }

         // AYGÜN DAMLA 'ya yeni numara ver
         hm.put("AYGÜN DAMLA", new Integer(20894828));
         System.out.println("AYGÜN DAMLA : " + hm.get("AYGÜN DAMLA"));

       }
}
     /*
     Çıktı:
       MERVE ALATLI : 20895548
       BÜYÜKKILIÇ AYKUT : 20893085
       CANER HALİL İBRAHİM : 20393385
       AYGÜN DAMLA : 20894820
       CAN FEHİME : 20793172
       AYGÜN DAMLA : 2089482
     */
```

Açıklamalar:

`HasMap` tipinden *hm* nesnesi değerlere anahtarlar eşleyecek bir gönderimdir (map). `Integer` ile `Character` tiplerini eşleyen bir `HashMap` nesnesidir.

`m.put(i, (char) i)` metodu ASCII kodu *i* olan harfe onun ASCII kodunu anahtar olarak gönderir.

`Set<Integer> kSet = m.keySet()` deyimi ile tanımlanan kSet değişkeni *m* içindeki anahtarları (key) belirler.

`Iterator<Integer> sayaç = kSet.iterator()` deyimi ile tanımlanan *sayaç* değişkeni tamsayılar üzerinde bir *iterator* (tekrarlayıcı) tanımlar.

while `(sayaç.hasNext())` döngüsü, kontrolün o anda bulunduğu *sayaç* 'dan sonra başka *sayaç* varsa `true` değerini alır ve kontrolü sonraki *sayaç* 'a geçirir.

`System.out.print(anahtar)` deyimi kSet içindeki öğeleri (anahtar) yazar.

`System.out.print(m.get(anahtar))` deyimi m.HashSet koleksiyonu içinde *anahtar* 'ın eşlendiği değeri yazar.

```
import java.util.HashMap;
import java.util.Iterator;

public class HashMapDemo {

	public static void main(String args[]) {

		// Öntanımlı sıgası (16) ile boş bir HashMap yaratıyor
		HashMap hashMap = new HashMap();
		// HashMap'e öğe ekleme
		hashMap.put("Elma", new Integer(1));
		hashMap.put("Armut", new Integer(2));
		hashMap.put("Portakal", new Integer(3));

		System.out.println(hashMap.entrySet());

		if (hashMap.containsValue(new Integer(1))) {
		  System.out.println("HashMap 1 değerini içeriyor");
		} else {
		  System.out.println("HashMap 1 değerini içermiyor");
		}

		if (hashMap.containsKey("Elma")) {
		  System.out.println("HashMap 'Elma' anahtarını içeriyor");
		} else {
		  System.out.println("HashMap 'Elma' anahtarını içermiyor");
		}

		Integer n = (Integer) hashMap.get("Portakal");
		System.out.println("Portakal anahtarına eşlenen değer:" + n);

		System.out.println("HashMap'in anahtarları :");
		Iterator iterator = hashMap.keySet().iterator();

		while (iterator.hasNext()) {
			System.out.println(iterator.next());
		}
		System.out.println("HashMap'in bütün değerleri");
		iterator = hashMap.entrySet().iterator();

		while (iterator.hasNext()) {
			System.out.println(iterator.next());
		}

		System.out.println(hashMap.remove("Armut")
				+ " öğesi HashMap'ten silindi.");

	}

}
	/*
	Çıktı:
	  [Elma=1, Armut=2, Portakal=3]
```

```
        HashMap 1 değerini içeriyor
        HashMap 'Elma' anahtarını içeriyor
        Portakal anahtarına eşlenen değer: 3
        HashMap'in anahtarları :
        Elma
        Armut
        Portakal
        HashMap'in bütün değerleri
        Elma=1
        Armut=2
        Portakal=3
        2 öğesi HashMap'ten silindi.
    */
```

Örnek:

```
import java.util.*;

class HashMapDemo {
      public static void main(String args[]) {
            // Create a hash map
            HashMap hm = new HashMap();
            // Put elements to the map
            hm.put("Deniz", new Double(83.34));
            hm.put("Melek", new Double(67.22));
            hm.put("Berna", new Double(78.00));
            hm.put("Cansu", new Double(99.22));
            hm.put("Gökhan", new Double(84.08));
            // Get a set of the entries
            Set set = hm.entrySet();
            // Get an iterator
            Iterator i = set.iterator();
            // Display elements
            while (i.hasNext()) {
                  Map.Entry me = (Map.Entry) i.next();
                  System.out.print(me.getKey() + " : ");
                  System.out.println(me.getValue());
            }
            System.out.println();
            // Berna'nın notunu düzelt
            double puan = ((Double) hm.get("Berna")).doubleValue();
            hm.put("Berna", new Double(puan + 10.07));
            System.out.println("Berna'nın yeni puanı: " + hm.get("Berna"));
      }
}
  /*
  Çıktı:
    Deniz : 83.34
    Cansu : 99.22
    Gökhan : 84.08
    Berna : 78.0
    Melek : 67.22

    Berna'nın yeni puanı: 88.07
  */
```