# Map Arayüzü

**java.util**
## Interface Map

**Altarayüzleri:**

SortedMap

**Kılgılayan Sınıflar:**

AbstractMap, Attributes, HashMap, Hashtable, IdentityHashMap, RenderingHints, TreeMap, WeakHashMap

*Map*, Java Collections Framework 'un bir üyesidir.

*Map (gönderim)* anahtarları değerlere eşleştiren bir nesnedir. Örneğin, bir ad listesinde her ada bir sıra numarası vermek bir *Map (gönderim)* işlemidir. Bu işlemde sıra numaralarının her biri bir *anahtar*, her ad bir *değer* olur. Liste karışmasın diye her ada ayrı bir sıra numarası verilir. Sıra numaraları anahtarlar, adlar ise değerlerdir. Sıra numaraları birbirlerinden farklıdır; ancak adlar farklı olmayabilir. Örneğin, aşağıdaki ad listesinde Selda adlı iki kişi var. Onların her birine ayrı sıra numarası verildi. Sıra numaraları (anahtarlar) içinde dublikasyon yoktur. Ama adlar (değerler) içinde dublikasyon olabilir.

| | |
|---|---|
| **1** | **Selda** |
| **2** | Can |
| **3** | Selda |
| **4** | Murat |

Benzer olarak, *Map*, her değere farklı bir anahtar eşler; anahtarlarda duplikasyon olamaz. Değerlerde duplikasyon olabilir. Anahtar sayısı değer sayısından az olamaz; daha çok olabilir. Yaptığı işe göre, *Map* bir fonksiyondur.

Java, üç ayrı Map kılgılar:

HashMap,
TreeMap,
LinkedHashMap

Bunların davranışları ve performansları, sırasıyla, *HashSet, TreeSet* ve *LinkedHashSet* gibidir. Ayrıntı için *Set Arayüzü*'ne bakınız.

Map arayüzü koleksiyonu üç türlü görmemizi sağlar: *anahtarlar kümesi, değerler kümesi* ve *anahtar-değer eşleşmeleri kümesi*. Map içindeki sıralama, koleksiyon üzerinde tanımlanan iteratörün öğeleri bize veriş sırasıdır. Ancak, *TreeMap* yapısında öğelerin sırası belirlidir. *HashMap* yapısında ise sıra belirli değildir.

## Map arayüzünün metotları

`void` **clear**`()`

`Map` içindeki bütün gönderimleri siler.

`boolean` **containsKey**`(Object key)`

`Map` içinde belirtilen anahtar varsa `true` yoksa `false` değeri verir.

`boolean` **containsValue**`(Object value)`

`Map` içinde belirtilen değer varsa `true` yoksa `false` değeri verir.

`Set` **entrySet**`()`

`Map` içindeki öğeleri bir küme olarak verir.

`boolean` **equals**`(Object o)`

Belirtilen öğenin `Map` içinde bir öğeye eşit olup olmadığını bulur.

`Object` **get**`(Object key)`

Belirtilen anahtarın eşleştiği değeri verir.

`int` **hashCode**`()`

`Map` için hash kodu verir.

`boolean` **isEmpty**`()`

anahtar-değer eşleşmesi yoksa `true` verir.

`Set` **keySet**`()`

anahtarları bir küme olarak verir.

`Object` **put**`(Object key, Object value)`

verilen anahtarı verilen nesneye eşler.

`void` **putAll**`(Map t)`

`Map` 'i istenilen `Map`'e kopyalar.

`Object` **remove**`(Object key)`

verilen anahtarla eşleşen nesne varsa onu `Map`'ten siler.

`int` **size**`()`

anahtar-değer eşleşmelerinin sayısını verir.

`Collection` **values**`()`

`map` içindeki değerleri bir koleksiyon olarak verir.

**Örnek:**

Aşağıdaki program `Map` sınıfının kullanılışına bir örnektir. Veriyi depolamak için `HashMap` kullanılmıştır.

```
import java.util.*;
```

```
public class HashSetDemo {

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

**Örnek:**

Aşağıdaki program *Map* sınıfının kullanılışına bir örnektir. Program önce bir *HashMap* yapısı kuruyor sonra o yapıyı bir *TreeMap* yapısına dönüştürüyor. Programın sonunda her iki yapıdaki öğeler yazdırılıyor. *HashMap* yapısına (depo) girilen öğelerin farklı veri tiplerinden nesneler olduğuna dikkat ediniz.

```
import java.util.*;
import java.util.HashMap;

public class MapDemo {

	public HashMap hashMap;
	// veriyi sıralamak için TreeMap yapısı
	private TreeMap treeMap;

	public void startDemo() {

		hashMap = new HashMap();
		hashMap.put("Key1", new Double(3434.34));
		hashMap.put("Key2", new Integer(123));
		hashMap.put("Key3", new String(" Ankara"));
		hashMap.put("Key4", new Boolean(true));

		// Sort the hash map using a tree map
		treeMap = new TreeMap(hashMap);
	}

	public static void main(String[] args) {
		MapDemo md = new MapDemo();
		md.startDemo();
		System.out.print(" HashMap :  ");
		System.out.println(md.hashMap);
		System.out.print(" TreeMap :  ");
		System.out.println(md.treeMap);
	}
}

  /*
```

```
   Sırasız
    HashMap :   {Key2=123, Key1=3434.34, Key4=true, Key3= Ankara}
   Sıralı
    TreeMap :   {Key1=3434.34, Key2=123, Key3= Ankara, Key4=true}
   */
```

## Örnek

Aşağıdaki program, öğrencileri numaralarına eşleyen bir *HashMap* yapısıdır.

```
import java.util.*;

public class HashMapDemo {

      public static void main(String[] args) {

            HashMap hm = new HashMap();
            hm.put("ALATLI MERVE", new Integer(20895548));
            hm.put("AYGÜN DAMLA ", new Integer(20094828));
            hm.put("BÜYÜKKILIÇ AYKUT", new Integer(20893085));
            hm.put("CAN FEHİME  ", new Integer(20793172));
            hm.put("CANER HALİL ", new Integer(20393385));
            hm.put("ÇELİK FATİH ", new Integer(20893682));
            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'nın numarasını düzelt
            hm.put("AYGÜN DAMLA", new Integer(20894828));
            System.out.println("AYGÜN DAMLA 'nın numarası : "
                        + hm.get("AYGÜN DAMLA"));
      }
}
     /*
     Çıktı:
     BÜYÜKKILIÇ AYKUT : 20893085
     CANER HALİL  : 20393385
     ALATLI MERVE : 20895548
     ÇELİK FATİH  : 20893682
     CAN FEHİME   : 20793172
     AYGÜN DAMLA  : 20094828
     AYGÜN DAMLA 'nın numarası : 20894828
     */
```

## Örnek:

Aşağıdaki program bazı *Map* fonksiyonlarının kullanılışını göstermektedir.

```
import java.util.*;

public class MapDemo {

	public static void main(String[] args) {
		Map map = new HashMap();
		map.put("Karaçay", "ist264");
		map.put("Umut", "ist378");
		map.put("Özlem", "ist254");
		map.put("Erdem", "ist360");

		System.out.println("Map        :" + map);
		System.out.println("hashCode() :" + map.hashCode());
		System.out.println("entrySet() :" + map.entrySet());
		System.out.println("values()   :" + map.values());
		System.out.println("keySet()   :" + map.keySet());
	}
}

   /*
    Map        :{Erdem=ist360, Karaçay=ist264, Özlem=ist254, Umut=ist378}
    hashCode() :-1075459124
    entrySet() :[Erdem=ist360, Karaçay=ist264, Özlem=ist254, Umut=ist378]
    values()   :[ist360, ist264, ist254, ist378]
    keySet()   :[Erdem, Karaçay, Özlem, Umut]
   */
```

**Örnek :**

Aşağıdaki program alfabenin büyük harflerine ASCII kodlarını 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> ks = m.keySet();

		Iterator<Integer> i = ks.iterator();

		while (i.hasNext()) {
			Integer key = i.next();
			System.out.print(key + "    ");
			System.out.println(m.get(key));
		}
	}
}

     /*
```

```
Çıktı:
68    D
69    E
70    F
71    G
65    A
66    B
67    C
…
*/
```

Bu örnekte bir HashMap nesnesi kılgılanıyor. `Map<Integer, Character>` ifadesi `Map` için anahtarları `Integer` ve değerleri `Character` olarak tanımlıyor. Put() metodu ile HashMap koleksiyonuna ascii kodları eşlenmiş harfleri ekliyoruz. Get() metodu ile koleksiyonu yazdırıyoruz. Bu koleksiyonda sıralamanın rasgele olduğuna dikkat ediniz. Çünkü *Map*'e eklenen *anahtar-değer* ikilileri ekleniş sırasını korumayabilir.

**Örnek:**

Aşağıdaki program *Map* 'i başka bir *Map*' e sonra da bir *TreeMap* yapısına dönüştürüyor. *TreeMap* yapısında değerlerin sıralı olduğuna dikkat ediniz.

```
import java.util.*;

public class MapDemo {
      Map<String, String> map1, map2;

      public static void main(String[] args) {
            MapDemo md = new MapDemo();
            md.map1 = new HashMap<String, String>();
            md.map1.put("ALATLI", "20895548");
            md.map1.put("DAMLA", "20894828");
            md.map1.put("AYKUT", "20893085");
            md.map1.put("CAN", "20793172");
            System.out.println();
            System.out.println("Map Öğeleri:");
            System.out.println("\t" + md.map1);

            System.out.println("putAll() metodu ile ile aktarma :");
            md.map2 = new HashMap<String, String>();
            md.map2.putAll(md.map1);
            System.out.println("\t" + md.map2);

            System.out.println("TreeMap yapısına dönüştürme :");
            TreeMap treeMap = new TreeMap(md.map2);
            System.out.println("\t" + treeMap);
      }
}

  /*
  Map Öğeleri:
      {DAMLA=20894828, AYKUT=20893085, ALATLI=20895548, CAN=20793172}
  putAll() metodu ile ile aktarma :
      {DAMLA=20894828, AYKUT=20893085, ALATLI=20895548, CAN=20793172}
  TreeMap yapısına dönüştürme :
      {ALATLI=20895548, AYKUT=20893085, CAN=20793172, DAMLA=20894828}
  */
```