Class HashSet

java.util

Class HashSet

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

Kılgıladığı arayüzler:

Cloneable, Collection, Serializable, Set

Altsınıfları:

JobStateReasons, LinkedHashSet

HashSet sınıfı Java Collections Framework 'un bir üyesidir. *Set* arayüzünü kılgılar. O nedenle tekrarlamalarda (iteration) öğelerin geliş sırası için bir öngörü yapılamaz. Farklı takrarlamalarda, öğeler farklı sıralarda gelebilir. Bu sınıf *null* öğe içerebilir.

Kurucular
HashSet() Başlangıç sığası 16 ve yükleme çarpanı 0.75 olan boş bir `HashSet` kılgılar.
HashSet(Collection c) Belirtilen koleksiyona ait öğeleri içeren bir `HashSet` kılgılar.
HashSet(int initialCapacity) Başlangıç sığası belirtilen sayıda ve yükleme çarpanı 0.75 olan boş bir `HashSet` kılgılar.
HashSet(int initialCapacity, float loadFactor) Başlangıç sığası ve yükleme çarpanı belirtilen sayılarda olan boş bir `HashSet` kılgılar.

Metotlar	
boolean	**add**(Object o) Belirtilen öğeyi ekler. Ancak, belirtilen öğe içeriliyorsa, onu tekrar eklemez (duplikasyon olmaz).
void	**clear**() Bütün öğeleri siler.
Object	**clone**() HashSet kılgısının bir kopyasını yapar.
boolean	**contains**(Object o) Belirtilen öğe küme içinde ise true değerini verir.

boolean	**isEmpty**() Küme boş ise `true` değerini verir.
Iterator	**iterator**() Kümenin öğeleri üzerinde bir tekrarlayıcı (iterator) oluşturur.
boolean	**remove**(Object o) Belirtilen öğeyi kümeden siler.
int	**size**() Kümedeki öğe sayısını verir.

Yukarıdaki metotlara ek olarak, üst sınıflarındaki metotları da kullanabildiği için, kümelerle ilgili hemen her işi yapmayı sağlayan metotlar vardır.

HashSet Kullanımına Örnekler

Örnek:

Aşağıdaki program `Set` sınıfının kullanılışına bir örnektir. Program komut satırından girilen kelimelerle bir *set* oluşturmakta ve sonra onları ekrana yazmaktadır.

```
import java.util.*;

public class HashSet01 {

	public static void main(String[] args) {
		Set set = new HashSet();
		set.add("Matematik");
		set.add("Fizik");
		set.add("Biyoloji");
		System.out.println();
		System.out.println(" Set Öğeleri:");
		System.out.print("\t" + set);
	}
}

		/*
		Çıktı:
		Set Öğeleri:
		   [Biyoloji, Fizik, Matematik]
		*/
```

Örnek:

add() metodu *HashSet*'te var olan bir öğeyi tekrar eklemez; yani *HashSet* içinde dublikasyon olamaz. Aşağıdaki program bunu göstermektedir. *Add("Antalya")* ilk seferinde "Antalya" öğesini HashSet'e ekliyor, ama ikinci kez eklemiyor. *Add()* metodu eklemeyi yaparsa *true*, yapmazsa *false* değeri verir.

```
import java.util.*;

public class HashSet01 {

	public static void main(String[] args) {
		Set hs = new HashSet();
		System.out.println(hs.add("Antalya"));
		System.out.println(hs.add("Antalya"));
```

```
	}
}
		/*
		Çıktı:
		true
		false
		*/
```

Örnek:

Aşağıdaki program bir HashSet yaratıyor, öğe ekliyor, siliyor ve öğe sayısını bildiriyor.

```
import java.util.HashSet;
import java.util.Iterator;

public class HashSet01 {

	public static void main(String[] args) {

		HashSet<String> hs = new HashSet<String>();

		// duplicate element is not permitted
		hs.add("Bursa");
		hs.add("Ankara");
		hs.add("Malatya");
		hs.add("Diyarbakır");
		hs.add("Muğla");
		hs.add("Ankara");

		Iterator it = hs.iterator();

		while (it.hasNext()) {
			String str = (String) it.next();

			System.out.println("Öğe :" + str);
		}

		// HashSet'in sığası (öğe sayısı)

		System.out.println("Sığa :" + hs.size());

		// HashSet' in bir öğesini sil :
		hs.remove("Malatya");
		System.out.println("Sığa :" + hs.size());
		// HashSet' in bütün öğelerini sil :
		hs.clear();
		System.out.println("Sığa :" + hs.size());
	}
}
	/*
	  Çıktı:
	  Öğe :Malatya
	  Öğe :Ankara
	  Öğe :Muğla
	  Öğe :Diyarbakır
	  Sığa :5
	  Sığa :4
	  Sığa :0
	*/
```

Açıklama:
`hs.add()` metodu *HashSet*'e parametrede belirtilen öğeyi ekler. Ancak "*Ankara*" öğesi ikinci kez eklenmek istenince, bu istek yerine gelmez.
`Iterator it = hs.iterator()` deyimi, küme üzerinde bir tekrarlayıcı (iterator) oluşturur.

`size()` metodu HashSet'in sığasını (öğe sayısını) verir.

`next()` metodu, tekrarlayıcının o andaki değerinden sonra başka öğe varsa, onu verir.

`remove()` metodu, parametresinde belirtilen öğesi siler.

`clear()` metodu, HashSet'in bütün öğelerini siler, onu boş küme yapar.

Örnek:

Aşağıdaki program, `HashSet` içindeki duplikasyonların sayısını bulur.

```
import java.util.*;

public class HashSet01 {

	public static void main(String[] args) {

		Set hs = new HashSet();
		hs.add("Berna");
		hs.add("Ahmet");
		hs.add("İpek");
		hs.add("Ceyda");
		hs.add("Tuğçe");
		hs.add("İpek");

		for (int i = 0; i < args.length; i++) {
		 if (!hs.add(args[i]))
		  System.out.println("Duplikasyonlar : " + args[i]);
		}

		System.out.println(hs.size() + " farklı öğe bulundu : " + hs);
	 }
}
	/*
	  Çıktı:
	  5 farklı öğe bulundu : [Ahmet, İpek, Ceyda, Tuğçe, Berna]
	*/
```

Örnek:

```
import java.util.HashSet;
import java.util.Iterator;

public class QueueDemo {

	public static void main(String[] args) {

		HashSet<String> hs = new HashSet<String>();
```

```
            // duplikasyon olamaz
            hs.add("İlknur");
            hs.add("Gözde");
            hs.add("Ceyda");
            hs.add("Ayfer");
            hs.add("Nedim");

            Iterator it = hs.iterator();

            while (it.hasNext()) {
                  String value = (String) it.next();

                  System.out.println("değer :" + value);
            }

            // size() metodu hashSet'in sığasını verir

            System.out.println("Sığa  :" + hs.size());

            // remove() metodu öğe siler :
            hs.remove("d");

            // clear() metodu bütün öğeleri siler
            hs.clear();
      }
}
      /*
      Çıktı:
       değer :Ayfer
       değer :Nedim
       değer :Ceyda
       değer :İlknur
       değer :Gözde
       Sığa  :5
      */
```