JAVA ÖDEV3

Java 3. Ödevimizin bu bölümünde Comparator arayüzü, HashSet, TreeSet ve HashMap sınıflarının kullanım amacı ve çalışma mantığını detaylı bir şekilde açıklamaya çalışacağım. İlk olarak Comparator arayüzü nedir buna değinelim.

1-Comparator interface:

Comparator arayüzü java collection framework'un bir üyesidir.Bu interface aynı tipteki iki objenin büyüklük,küçüklük ve eşitlik durumunu kontrol etmek için kullanılır ve bunu Collection.sort() methodunu kullanarak yapar. Bu metot, collections içindeki sınıflara ait nesnelerden oluşan koleksiyonları doğal sıraya koyar. Doğal sıralama, örneğin sayılar için küçüklük-büyüklük sırasıdır. Stringler için sözlük sıralamasıdır . Ayrıca, TreeSet ya da TreeMap gibi veri yapılarında da doğal sıralama yapar.

Eğer ki doğal sıralamadan farklı bir sıralama yapılmak isteniyorsa, Collections.sort() metodu kullanılamaz. O durumda, arayüzde olan compare(Object o1, Object o2) metodu kullanılır. Bu metot bir üst sınıfta tanımlanmamışsa, ele alınan sınıfta tanımlanması gerekir. Örneğin;

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import java.util.Comparator;

public class ComparatorDemo implements Comparator<Integer> {
	//Bu metot sıralamayı tesine çeviriyor.
	public int compare(Integer obj1, Integer obj2) {
		return (obj1 > obj2 ? -1 : (obj1 == obj2 ? 0 : 1));
	}
}

class Uygulama {
	public static void main(String[] args) {
		List<Integer> list = new ArrayList<Integer>();
		list.add(5);
		list.add(4);
		list.add(3);
		list.add(7);
		list.add(2);
		list.add(1);
		Collections.sort(list, new ComparatorDemo());
		for (Integer k : list) {
			System.out.print("\t" + k);
		}
	}
}

	/*
	 7	5	4	3	2	1
	*/
```

Burada Comparator arayüzünü algılayan bir sınıf tanımladık. Daha sonra program kendi sıralama metodunu tanımlamaktadır. (Sıralamayı tersine çeviriyor).

2-HashSet nedir:

HashSetler genelde karşılaştırma işlemleri için kullanılır. Hashset içeriğini dizi halinde tutar.Herbir anahtardan birer tane tutar. İki tane anahtar asla eşit olmaz ve iki tane olunca bulunur.

Java Collections yapısının altında yer alan **Set** arayüzüne bağlı hashSet ler yapısı gereği girilen verileri düzensiz bir şekilde **s**ıralamaya tabi tutar, böylelikle yapının performansını arttırır.

HashSet öğeleri hashing mekanizması ile depolar,benzersiz öğeleri içerir,boş değere izin verir,ekleme sırası korunmaz ve arama işlemleri için en iyi yaklaşımdır.
HashSet sınıfın çeşitli methodları vardır.Bunlar:

add(E e)	Zaten mevcut değilse, belirtilen elemanı bu sete eklemek için kullanılır.
clear()	Setten tüm unsurları çıkarmak için kullanılır.
clone()	Bu HashSet örneğinin basit bir kopyasını döndürmek için kullanılır.
contains(Object o)	Bu set, belirtilen öğeyi içeriyorsa, true değerini döndürmek için kullanılır.
isEmpty()	Bu set hiç eleman içermiyorsa, true değerini döndürmek için kullanılır.
iterator()	Bu kümedeki öğeler üzerinde bir yineleyici döndürmek için kullanılır.
remove(Object o)	Varsa belirtilen elemanı bu setten çıkarmak için kullanılır.

Şöyle bir örnekle kapatalım bu başlığı:

```java

import java.util.*;

class Main {
  public static void main(String[] args) {

    HashSet<String> set=new HashSet();
    set.add("Bir");
    set.add("İki");
    set.add("Üç");
    set.add("Dört");
    set.add("Beş");

    Iterator<String> i=set.iterator();
    while(i.hasNext())
    {
      System.out.println(i.next());
    }
  }
}

```

Çıktı:

```
Dört
Bir
İki
Üç
Beş

```

2-TreeSet nedir:

Collection içerisinde **SortedSet** interface yapısının içerisinde yer alan bir kavramdır. Benzersiz özelliklere sahip değerleri korur, bu değerleri hiyerarşik bir sıralama sistemine göre saklar. TreeSet öğeleri SortedSet yapısına bağlıdır, bu yüzden tekrarlanan öğelere izin vermez,verilerin eklenme sırasını korumaz, ama TreeSet yapısındaki doğal sıralamaya göre sıralanır ve daha büyük miktardaki bilgileri depolamak için kullanılır. Bu durum verilerin **daha hızlı** ve **kolay** bir şekilde erişmemize olanak sağlar.

HashSetteki methodların aynısı burada da kullanılıyor. Örneğin :

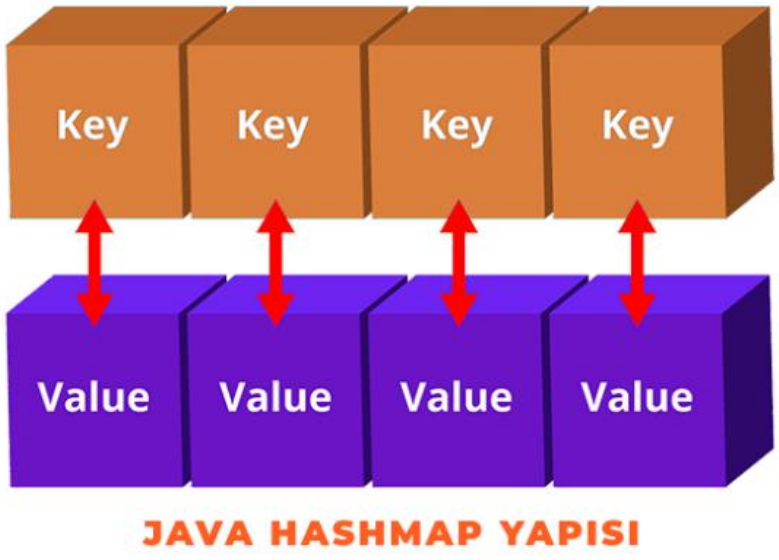
- **first:** Koleksiyon Öğesinin en küçük elemanını verir.
- **Add:** ekleme için kullanılır
- **Remove:** silme için kullanılır
- **last:** en büyük elemanını verir.
- **headSet:** metot içinde verilen öğeden küçük olanları verir.
- **tailSet:** metot içinde verilen öge ve büyük olanları verir.
- **descendingSet:** Küme sıralamasını tersine çevirir

Aşağıdaki örnekte [A,B,C] şeklinde sıralanmış bir çıktı verecektir.

```java
// Importing required utility classes
import java.util.*;

// Main class
class GFG {

    // Main driver method
    public static void main(String[] args)
    {
        // Creating a Set interface with reference to
        // TreeSet
        Set<String> ts1 = new TreeSet<>();

        // Elements are added using add() method
        ts1.add("A");
        ts1.add("B");
        ts1.add("C");

        // Duplicates will not get insert
        ts1.add("C");

        // Elements get stored in default natural
        // Sorting Order(Ascending)
        System.out.println(ts1);
```

3-HashMap nedir:

Verilerinizi depolamak için kullanılan yöntemlerden birisidir. Ancak diğerlerinden farkı key, value çalışma mantığını kullanıyor olmasıdır. Depoladığı her değere bir anahtar işaret eder. Dolayısıyla bir elemanı, ögeyi bulmak kolaydır.



JAVA HASHMAP YAPISI

HashMap özellikleri şunlardır:

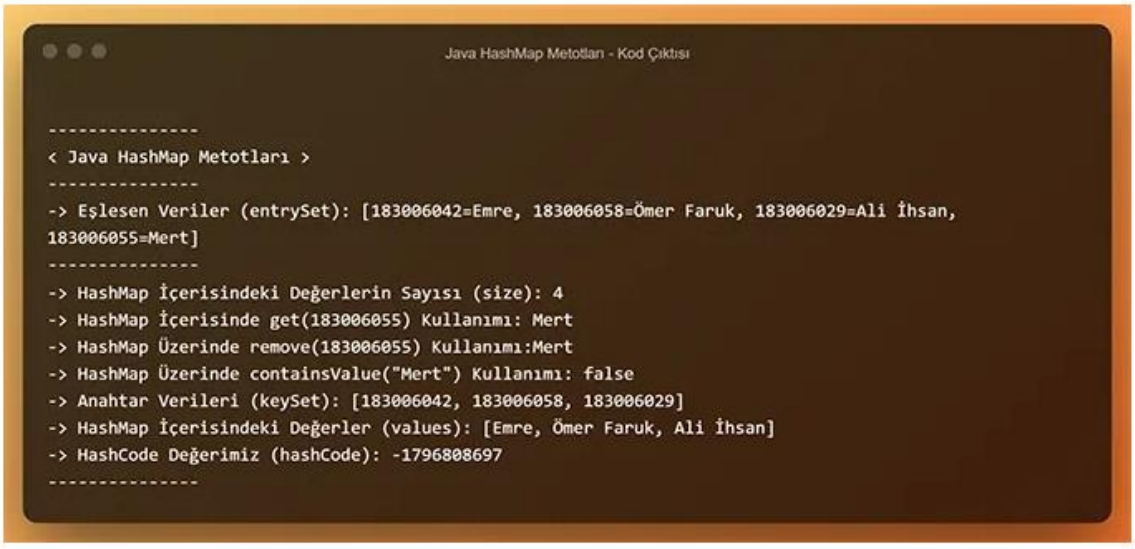
- HashMap temel olarak Map arayüzünün özelliklerini sınıf yapısıdır.
- Yinelenen key verilerinin eklenmesine izin verilmez. Ancak Yinelenen value eklenmesine izin verir.
- HashMap üzerinde eklenen değerlere erişmek için anahtarını bilmemiz gerekir.
- Verilerin sırasına ilişkin bir garanti verilmez. Özellikle oluşan sıranın zaman içerisinde sabit kalacağını garanti etmez.

HashMap methodları şunlardır:

- **Put(**Key,Value**):** Parametre olarak aldığı anahtar değeri yapı içerisinde bulunmuyor ise almış olduğu değer ile anahtar yapısını eşler ve depolar.
- **Remove(**Key**):** Parametre olarak almış olduğu anahtar değeri yapı içerisinde bulunduğu takdirde anahtar değeri ve eşleşen veriyi siler.
- **Get(**Key**):** HashMap içerisinde parametre olarak almış olduğu anahtar değeri bulunuyorsa eşleşmiş olduğu değeri geri döndürür.
- **ContainsKey(**Key**)** & **ContainsValue(**Value**):** HashMap içerisinde belirtilen anahtar veya değer varsa true yoksa false değerini döndüren metotlardır.
- **Clear():** HashMap içerisinde yer alan tüm verileri silmek için kullanılır.
- **EntrySet():** HashMap içerisine eklemiş olduğumuz öğeleri bir küme halinde geriye döndüren metot.
- **KeySet():** HashMap içerisine eklemiş olduğumuz anahtarları küme halinde geriye döndürür.
- **Size():** Anahtar ve değer eşleşmelerinin toplam sayısını geriye döndürür.
- **Values():** HashMap içerisinde yer alan değerleri bir koleksiyon olarak geriye döndürür.

```java
public class Main {

    public static void main(String[] args) {

        ///---> Değişken Tanımlaması <---\\\
        String ayrac = new String(new char[15]).replace("\0", "-");

        ///---> HashMap Non-Generic Tanımlaması <---\\\
        HashMap hMap = new HashMap();

        ///---> HashMap Metotlarını Kullanabilmek İçin Veri Girişi Yapalım <---\\\
        hMap.put(183006029,"Ali İhsan");// HashMap üzerinde veri eşlemek için 'put' metodunu kullanırız.
        hMap.put(183006042,"Emre");
        hMap.put(183006055,"Mert");
        hMap.put(183006058,"Ömer Faruk");

        ///---> HashMap Metotlarını Kullanalım <---\\\
        System.out.println(ayrac+"\n< Java HashMap Metotları >\n"+ayrac);
        System.out.println("-> Eşlesen Veriler (entrySet): "+ hMap.entrySet()+"\n"+ayrac);
        System.out.println("-> HashMap İçerisindeki Değerlerin Sayısı (size): "+ hMap.size());
        System.out.println("-> HashMap İçerisinde get(183006055) Kullanımı: " + hMap.get(183006055));
        System.out.println("-> HashMap Üzerinde remove(183006055) Kullanımı:"+hMap.remove(183006055));
        System.out.println("-> HashMap Üzerinde containsValue(\"Mert\") Kullanımı: " + hMap.containsValue("Mert"));
        System.out.println("-> Anahtar Verileri (keySet): " + hMap.keySet());
        System.out.println("-> HashMap İçerisindeki Değerler (values): " + hMap.values());
        System.out.println("-> HashCode Değerimiz (hashCode): "+ hMap.hashCode()+"\n"+ayrac);

    }
}
```

Kod Çıktısı:



KAYNAKÇA

1-https://hayalindekiyazilim.com/java/java-koleksiyonlar-collections/

2-https://emrecelen.com.tr/java-hashmap-nedir/

3-https://gunceljava.blogspot.com/2016/06/comparator-arayuzu.html

4-https://www.geeksforgeeks.org/treeset-in-java-with-examples/

5-https://www.tasarimkodlama.com/java-programlama/java-hashset-kullanimi/