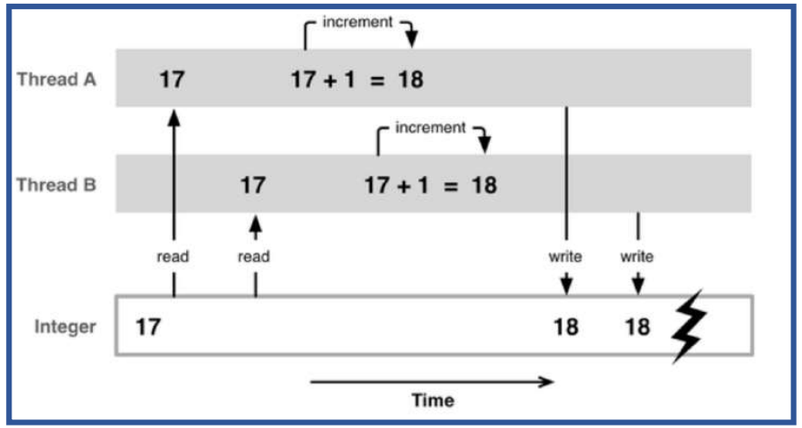
Multi Thread



Multithreading, CPU'nun maksimum kullanımı için bir programın iki veya daha fazla bölümünün aynı anda yürütülmesine izin veren bir Java özelliğidir.

Böyle bir programın her bir parçası bir iş parçacığı (thread) olarak adlandırılır.

Threads iki mekanizma kullanılarak oluşturulabilir:

1) Thread class'ina extend edilerek

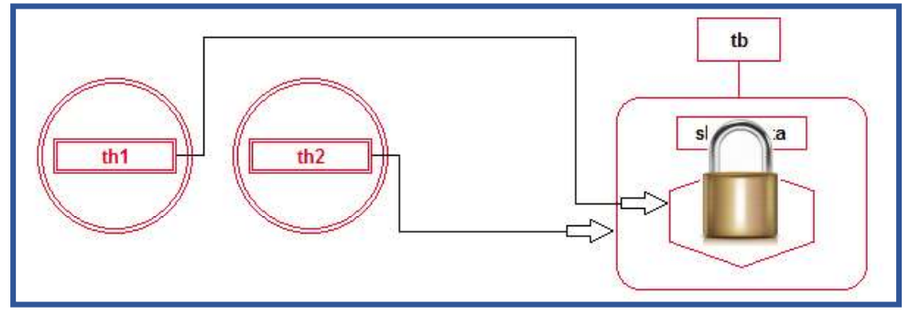
2) Çalıştırılabilir Interface'in implement edilmesi ile

Synchronizing

Multi-thread programlar genellikle birden çok thread'in aynı kaynaklara erişmeye çalıştığı ve sonunda hatalı ve öngörülemeyen sonuçlar ürettiği bir duruma gelebilir.

Bu nedenle, belirli bir zaman araliginda kaynağa yalnızca bir thread'in erişebileceğinden emin olunması gerekir.

Java, senkronize bloklar kullanarak thread oluşturmayi ve threads'in görevlerini senkronize bir sekilde yapmasini sağlar.

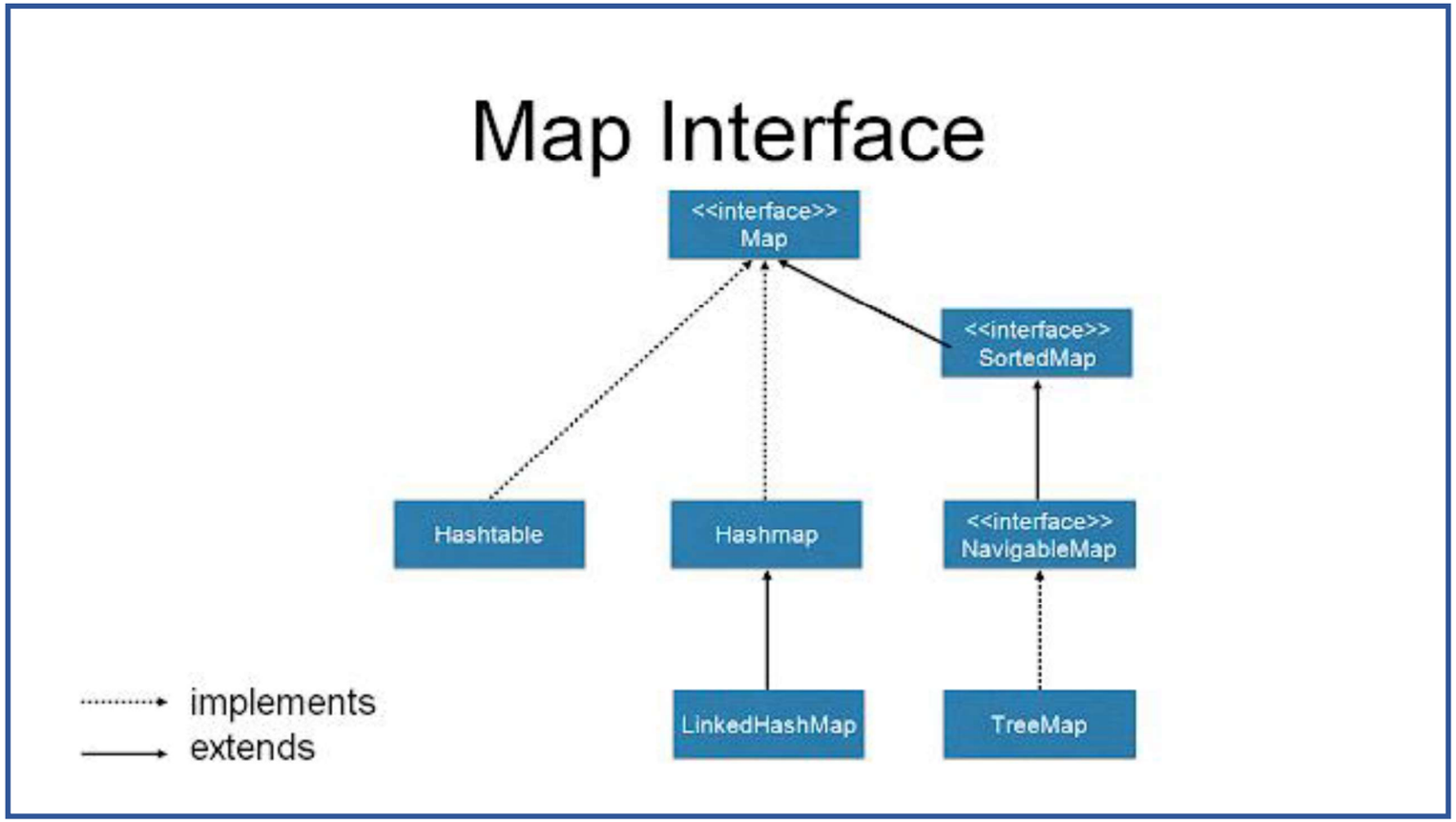


Java'da senkronize bloklar, synchronized keyword ile işaretlenir.

Java'da senkronize edilmiş blok mekanizmasi aynı obje üzerinde tek zaman diliminde tek thread calismasini saglar.. Blokda bir thread calismaya baslayinca, diger tum thread'ler ilk thread'in islemi bitene kadar bekletilir.

Maps

Maps key – value pairs kullanir. (anahtar –deger(ler)). Key'ler unique olmalidir.



Synchronizing & Maps

1) HashMap synchronized degildir. Thread-safe degildir

2) HashTable synchronized'dir. Thread-safe'dir ve thread'ler tarafindan ortak kullanilabilir

3) TreeMap synchronized degildir. Thread-safe degildir

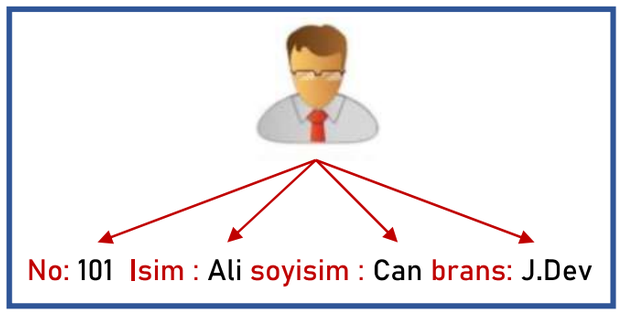
Maps

Reel projelerde kullanilan database yapisina en uygun Java objesidir.

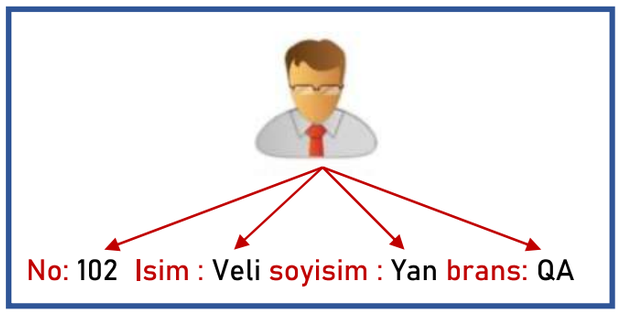
Maps key – value pairs kullanir (anahtar –deger(ler)).

Key'ler unique olmalidir.

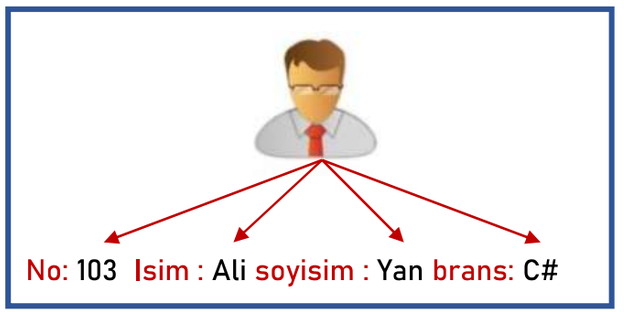
Map ile ayni ozelliklere sahip birden fazla objeyi ve ozelliklerini store edebilirsiniz.



Ogrenci 1



Ogrenci 2



Ogrenci 3

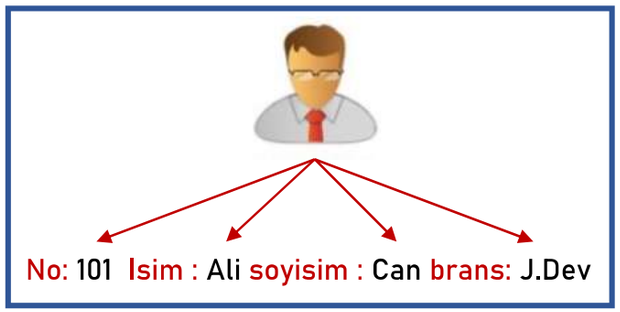
```
public static void main(String[] args) {

    HashMap<Integer,String> objeMap= new HashMap<>();
    objeMap.put(101, "Ali, Can, Java");
    objeMap.put(102, "Veli, Yan, Java");
    objeMap.put(103, "Ali, Yan, C#");

    System.out.println(objeMap);
}
```

```
{101=Ali, Can, Java, 102=Veli, Yan, Java, 103=Ali, Yan, C#}
```

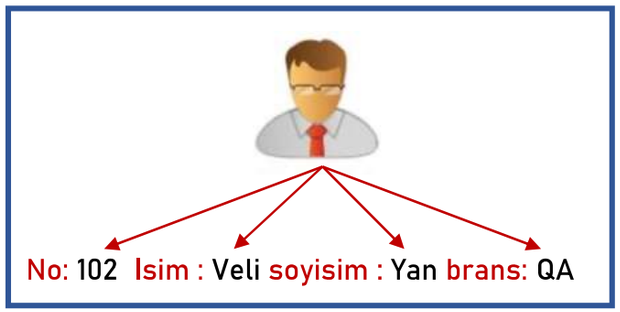
Ogrenci 1 Ogrenci 2 Ogrenci 3

Maps

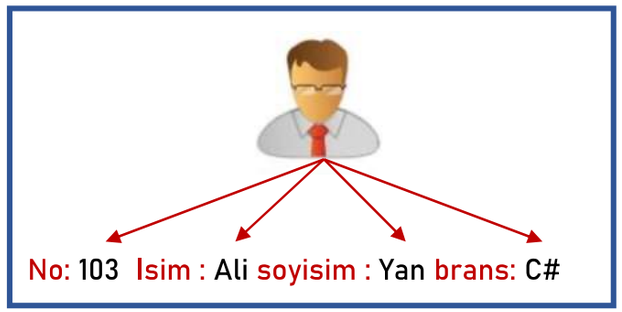
Map ile key ve value bilgilerine ayri ayri ulasabilir, istedigimiz degisiklikleri ayri ayri yapabiliriz



Ogrenci 1



Ogrenci 2



Ogrenci 3

```
System.out.println(objeMap.keySet());
```

→ `[101, 102, 103]`

keyset( ) method'u Set olarak key degerlerini verir.

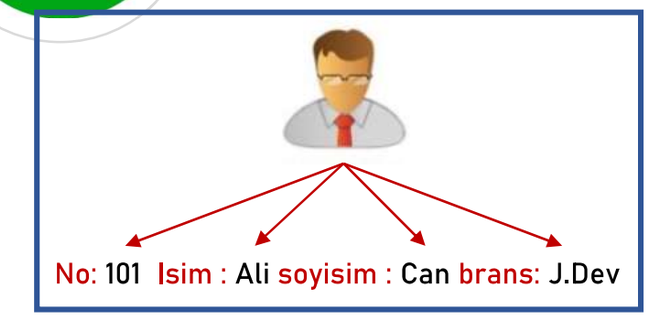
```
System.out.println(objeMap.values());
```

→ `[Ali, Can, Java, Veli, Yan, Java, Ali, Yan, C#]`

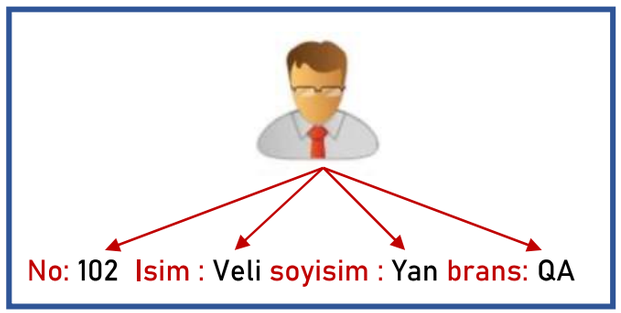
Ogrenci 1 Ogrenci 2 Ogrenci 3

values() method'u Collection olarak "value"lari verir. Collections'dan istedigimiz bir variable'a degerleri ekleyebilir ve kullanabiliriz.

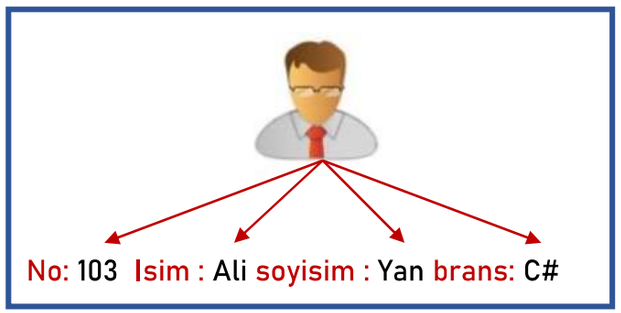
Maps (Nested Maps)



Ogrenci 1



Ogrenci 2



Ogrenci 3

Key **Value**

sinifMap= { 101={Isim : Ali, soyisim : Can, brans: J.Dev }, → **Ogrenci 1**

102={Isim : Veli, soyisim : Yan, brans: QA }, → **Ogrenci 2**

103={Isim : Ali, soyisim : Yan, brans: C# } → **Ogrenci 3**

}

Key **Val.** **Key** **Val.** **Key** **Val.**

{Isim : Ali, soyisim : Can, brans: J.Dev } → **Ogrenci 1**

Maps (Nested Maps)

Map ile bir objeye ait farkli bilgileri kategorilerine gore ayirip depolayabiliriz

Java Bank

Musteri :
Ali Can

Kisisel bilgiler	TL Hesap	Euro Hesap	Usd Hesap	Kredi Karti
Tc No: Isim : Soyisim : Telefon:	Hesap No: Kur : Acilis Tar. : Bakiye:	Hesap No: Kur : Acilis Tar. : Bakiye:	Hesap No: Kur : Acilis Tar. : Bakiye:	Kart No: Limit : Verilis Tar. : Kul.Limit :

Maps (Nested Maps)

Java Bank

Map Bank = { M.No1={Musteri Map1}, M.No2={Musteri Map2}, … }

Musteri :
Ali Can

Musteri Map1 = {KisiselBilgiler={Kis.Bil.Map} , TlHesap={TlHesapMap}…}

TlHesapMap = {HesapNo="12345" , Kur="Tl", AcTar="1.1.2021", Bakiye="0" }

Tc No: Isim : Soyisim : Telefon:	Hesap No: Kur : Acilis Tar. : Bakiye:	Hesap No: Kur : Acilis Tar. : Bakiye:	Hesap No: Kur : Acilis Tar. : Bakiye:	Kart No: Limit : Verilis Tar. : Kul.Limit :
Kisisel bilgiler	TL Hesap	Euro Hesap	Usd Hesap	Kredi Karti

Maps (Nested Maps)

Musteri :
Ali Can

Tc No:
Isim :
Soyisim :
Telefon:

Hesap No:
Kur :
Acilis Tar. :
Bakiye:

Kart No:
Limit :
Verilis Tar. :
Kul.Limit :

Kisisel bilgiler

{Soyisim=Can, TcNo=12345678901, Isim=Ali, Telefon=5553456789}

Hesaplar

{Bakiye=0, Acilis Tarihi=12.12.2021, Kur=TL, HesapNo=1234-1}

{Bakiye=0, Acilis Tarihi=12.12.2021, Kur=Eur, HesapNo=1234-2}

{Bakiye=0, Acilis Tarihi=12.12.2021, Kur=Usd, HesapNo=1234-3}

Kredi Karti

{Kul.Limit=11000, Limit=20000, KartNo=1234567890123456,
Verilis Tarihi=12.12.2021}

{ KrediKarti={ Kul.Limit=11000, Limit=20000, KartNo=1234567890123456, Verilis Tarihi=12.12.2021 },
UsdHesap={ Bakiye=0, Acilis Tarihi=12.12.2021, Kur=Usd, HesapNo=1234-2 },
KisiselBilgiler={ Soyisim=Can, TcNo=12345678901, Isim=Ali, Telefon=5553456789 },
EuroHesap={ Bakiye=0, Acilis Tarihi=12.12.2021, Kur=Eur, HesapNo=1234-2 },
TLHesap={ Bakiye=0, Acilis Tarihi=12.12.2021, Kur=TL, HesapNo=1234-1 } }

Maps

Soru 1) Verilen bir String'deki harfleri ve harflerin kacar kez kullanildigini return eden bir method yaziniz

Ornek : Input : Hellooo output : H=1, e=1, l=2, o=3

```
public static void main(String[] args) {
    System.out.println(harfSayisiBul("Hellooo"));// {e=1, H=1, l=2, o=3}

}
public static HashMap<String, Integer> harfSayisiBul(String str) {

    HashMap<String, Integer> map = new HashMap<>();
    String arr[] = str.split("");
    System.out.println(Arrays.toString(arr));

    for (String w : arr) {

        if (!map.containsKey(w)) {

            map.put(w, 1);
        } else {
            map.put(w, map.get(w) + 1);
        }
    }
    return map;
}
```

Maps

Soru 2) Verilen map'te istenen programlama dilini bilen kisileri list olarak donduren bir method yaziniz.

map➔ { 101=Ali, Can, java, 102=Veli, Yan, java, 103=Ali, Yan, C#}

Istenen dil ➔ java

Sonuc ➔ [Ali, Veli]

```
    Map<Integer,String> map1 = new HashMap<>();
    map1.put(101, "Ali, Can, java");
    map1.put(102, "Veli, Yan, java");
    map1.put(103, "Ali, Yan, C#");

    String istenenDil="JAVA";

    List<String> isimList = javaBilenler(map1,istenenDil);
    System.out.println(isimList);
}

private static List<String> javaBilenler(Map<Integer, String> map1, String istenenDil) {

    List<String> isimListesi=new ArrayList<>();

    for (String each : map1.values()) {

        String arr[] = each.split(", ");

        if(arr[2].equalsIgnoreCase(istenenDil)) {
            isimListesi.add(arr[0]);
        }
    }
    return isimListesi;
}
```

Önceki Dersten Aklımızda Kalanlar

1- Map : Map ayni ozelliklere sahip objelerin kendilerini ve ozelliklerini store edebilecegimiz bir interface'dir.
2- Map'de her element 2 bilgiye sahiptir
key : unique olmak zorundadir, tum elementler key'lere gore store edileceginden benzersiz olmasi gerekir.
value : elementin tum ozelliklerini tutabilecegimiz bolumdur, basit ve unique olmak zorunda degildir, tekrarli ve kompleks olabilir. Nested data turleri bile kullanilabilir. Ancak value ne kadar kompleks olursa, sonrasinda bilgilere ulasmak icin o kadar fazla ugrasmamiz gerekir.
3- Key ve value istenen data turunde olabilir.
4- Map'de siralama olmak zorunda degildir.
5- Map icindeki bilgilere saglikli ulasabilmek icin, value olarak girilen degerlerler ayni icerik ve ayni bicimlerde olmalidir. Bilgilere ulasmak maniplasyon ile mumkun olacagindan, bicim ve bilgi sayisi farkli olmamalidir.

Maps

1) containsKey(key); istenen key degeri Map'de varsa true, yoksa false doner .

2) containsValue(value); istenen key degeri Map'de varsa true, yoksa false doner .

3) entrySet(); Map'deki entry'leri bir Set olarak verir.

Entry : Map'de her bir elemani olusturan key-value ikilisidir

4) equals(map); Map'deki tum elemanlari karsilastirir. Hepsi ayni ise true farkli olan varsa false dondurur

Maps

5) get(key); istenen key degeri Map'de varsa o key'e ait value'yu, map'de yoksa null doner.

6) getOrDefault(key,defaultDeger); istenen key degeri Map'de varsa o key'e ait value'yu, key map'te yoksa default degeri doner.

7) putAll(map); verilen map'deki tum elemanlari bizim map'imize ekler, tekrarlanan eleman varsa uzerine yazar

8) compute(key, (key,value)->yeniDeger); verilen map'deki istenen key degerine sahip elemanin value'sunu gunceller key map'te yoksa ekler

Maps

9) ComputeIfPresent(key, (key,value)->yeniDeger); istenen key degeri Map'de varsa o key'e ait value'yu gunceller, map'de yoksa birsey yapmaz

10) ComputeIfAbsent(key, k ->yeniDeger); istenen key degeri map'de yoksa o key'i ve value'yu ekler, map'de varsa birsey yapmaz

11) putIfAbsent(key, value); verilen key map'de yoksa ekler.

12) size(); map'teki entry sayisini verir

Maps

Soru 3) Verilen bir listedeki tekrar etmeyen elmanlari veren bir method yaziniz

Ornek : Input : Hellooo output : [H, e]

```
public static List<String> getNonRepeatedChars(String str){

    List<String> list = new ArrayList<>();
    HashMap<String, Integer> map = new HashMap<>();
    String arr[] = str.split("");

    for(String w : arr) {
        map.computeIfPresent(w, (key, value)-> value+1);
        map.computeIfAbsent(w, k->1);
    }
    System.out.println(map);

    for(Entry<String, Integer> w : map.entrySet()) {

        if(w.getValue()==1) {
            list.add(w.getKey());
        }
    }
    return list;
}
```

Maps

Soru 4) Bir csv file'i okuyup icerigi map'e ceviren bir method yaziniz.

csv ➔ isim,Ali Id,101 Adres, Ankara

map➔ { isim=Ali, Id=101, Adres=Ankara}

```
        String dosyaYolu="C:\\Users\\lenovo\\Desktop\\NewCsv.csv";
        Map<String,String> map1 = csvdenMapYap(dosyaYolu);
        System.out.println("main method map olarak " + map1);
    }

    private static Map<String, String> csvdenMapYap(String dosyaYolu) {
        List<String> satirListesi = new ArrayList<>();
        Map<String,String> mapCsv=new HashMap<>();

        try {
            BufferedReader br1= new BufferedReader(new FileReader(dosyaYolu));
            String line=br1.readLine();

            while(line != null){
                satirListesi.add(line);
                line=br1.readLine();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("Liste olarak : " + satirListesi);
        for (String each : satirListesi) {
            String[] str= each.split(",");
            mapCsv.put(str[0], str[1]);
        }
        return mapCsv;
    }
```

HashMaps VS HashTable

- HashMap, key olarak sadece 1 tane null, value olarak ise istedigimiz kadar null'a izin verir, HashTable ise null'in kullanilmasina izin vermez

	Synchronized	Thread Safe	Null Keys And Null Values	Performance	Extends	Legacy
HashMap	No	No	Only one null key and multiple null values	Fast	AbstractMap	No
HashTable	Yes	Yes	No	Slow	Dictionary	Yes

Maps / TreeMap

- TreeMap, elemanlari natural order'a gore siralar. Siralama icin key'i dikkate alir
- HashMap thread-safe ve synchronized degildir
- Yavastir

1) ceilingEntry(key); key map'te varsa entry'yi dondurur, key map'de yoksa olmasi gereken yerden sonraki ilk entry'yi dondurur . En buyuk keyden daha buyuk deger girilirse null doner

2) descendingKeySet(); key'leri descending order'la dondurur

3) firstEntry(); ilk Entry'i dondurur

Maps / TreeMap

4) floorEntry(key); girdigimiz key map'te yoksa, key'i girdigimiz sayidan kucuk olan en yakin Entry'yi dondurur

5) headMap(key); girdigimiz key exclusive olmak uzere onceki Entry'leri bir map olarak verir

6) headMap(key,true); girdigimiz key inclusive olmak uzere onceki Entry'leri bir map olarak verir

7) tailMap(key); girdigimiz key inclusive olmak uzere sonraki Entry'leri bir map olarak verir

8) tailMap(key,false); girdigimiz key exclusive olmak uzere sonraki Entry'leri bir map olarak verir