

Iterable interface’dir ve --- > interface Collection extends Iterable…..

interface List extends Collection

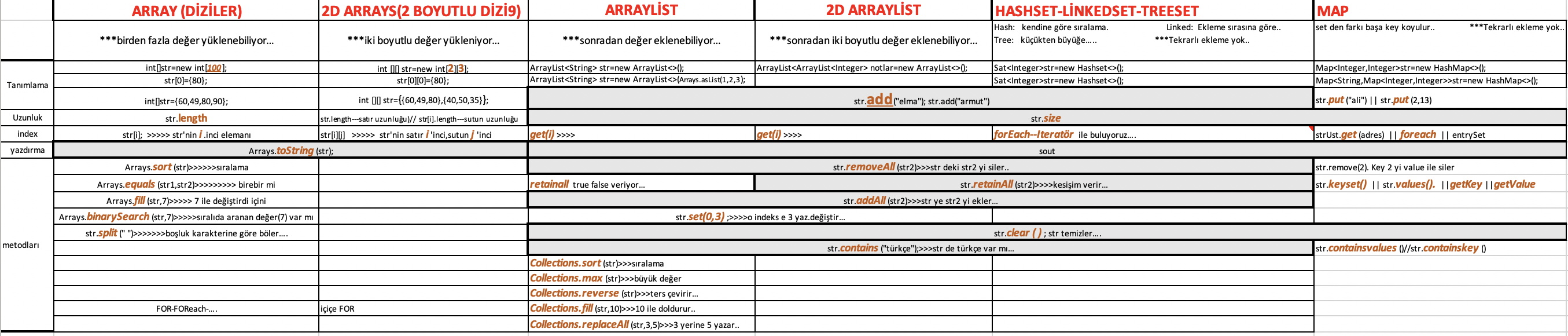
Class LinkedList extends AbstractSequentialList implements List, Deque

interface Queue extends Collection

List<Integer> liste1 = new LinkedList();//methodlari List’ten alir

Queue<Integer> liste2 = new LinkedList();//Queue methodlari kullanilabilir cunku Data turu Queue

Deque<Integer> liste3 = new LinkedList();





**Collections/Sets**

Linked bir rafta fizik kitaplarini arka arkaya dizmek demektir

Data’nin Unique olmasi benzersiz olmasi demek ornegin FakulteNo=GirisYiliUniversiteKoduBolumKoduGirisSiralamasi = Tipki Sql’deki primary key gibi

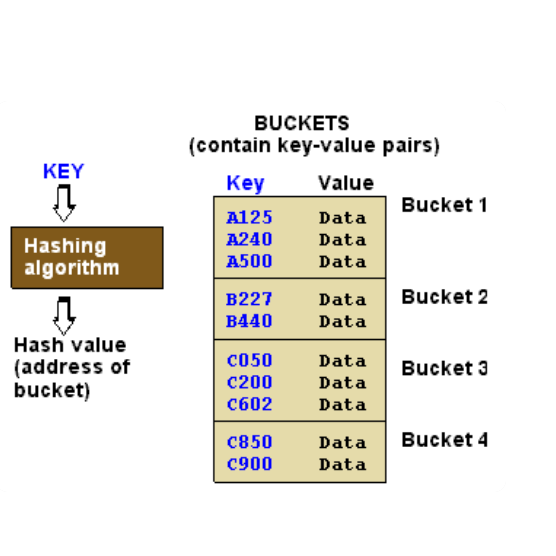
**Bu Hash mekanizmasidir Set’te en cok bu mekanizma kullanilir**

Hashcode, herhangi bir nesneden üretilen bir sayıdır. Bu, nesnelerin bir Hashtable'da hızlı bir şekilde alınmasını sağlayan şeydir. Aşağıdaki basit örneği hayal edin: Önünüzdeki masada, her biri 1'den 9'a kadar rakamlarla işaretlenmiş dokuz kutunuz var. onları olabildiğince çabuk bulabilmek için. İhtiyacınız olan şey, her bir nesneyi hangi kutuya koyacağınıza anında karar vermenin bir yoludur. Bir indeks gibi çalışır; Lahanayı bulmaya karar verirsiniz, böylece lahananın hangi kutuda olduğuna bakarsınız, sonra onu almak için doğruca o kutuya gidersiniz. Şimdi indeksle uğraşmak istemediğinizi, hangi kutuda yaşadığını nesneden hemen öğrenmek istediğinizi hayal edin. Örnekte, bunu yapmanın gerçekten basit bir yolunu kullanalım - harf sayısı nesne adına. Yani lahana 7. kutuya, bezelye 3. kutuya, roket 6. kutuya, banjo 5. kutuya vb. Peki ya gergedan? 10 karaktere sahiptir, bu yüzden algoritmamızı biraz değiştireceğiz ve

"yuvarlak saracağız", ="wrap round”

böylece 10 harfli nesneler kutu 1'e, 11 harf kutu 2'ye girecek ve böyle devam edeceğiz. Bu herhangi bir nesneyi kapsamalıdır. Bazen bir kutunun içinde birden fazla nesne olabilir, ancak bir roket arıyorsanız, bir fıstık ve roketi karşılaştırmak, bütün bir lahana, bezelye, banjos ve gergedan yığınını kontrol etmekten çok daha hızlıdır. Bu bir hash kodudur. Hashtable'da saklanabilmesi için bir nesneden sayı almanın bir yolu. Java'da bir karma kod herhangi bir tamsayı olabilir ve her nesne türü kendi kodunu oluşturmaktan sorumludur. Object'in "hashCode" yöntemini arayın.

* Set interface’dir ve obje olusturulamaz….
* Set benzersiz unique datalar icin kullanilir dublicate’e izin vermez….
* Set yukarida aciklanan hashcode mantigiyla calisir…
* HashSet hizlidir…Dublication’a izin vermez…Eski Ali’yi siler yeni Ali’yi ekler..
* LinkedHashSet siralidir ve daha yavastir…
* Treeset bir clas’tir ve siralidir…
* Siz Set’e eleman eklemek istediginizde Java 16 kapasitelik yer olusturur tipki StringBuilder’daki gibi…



Siz ne yaparsaniz yapin Java olusturdugu Bucket’lara cevirir.Ayni bucket’a gelen iki sayi oldugunda Java ya kitaplik methodunu sececek yada Universite no methodunu sececek.

Binlerce kitabiniz varsa hizli olani kullanmamiz gerekiyor yani Hash mekanizmasini

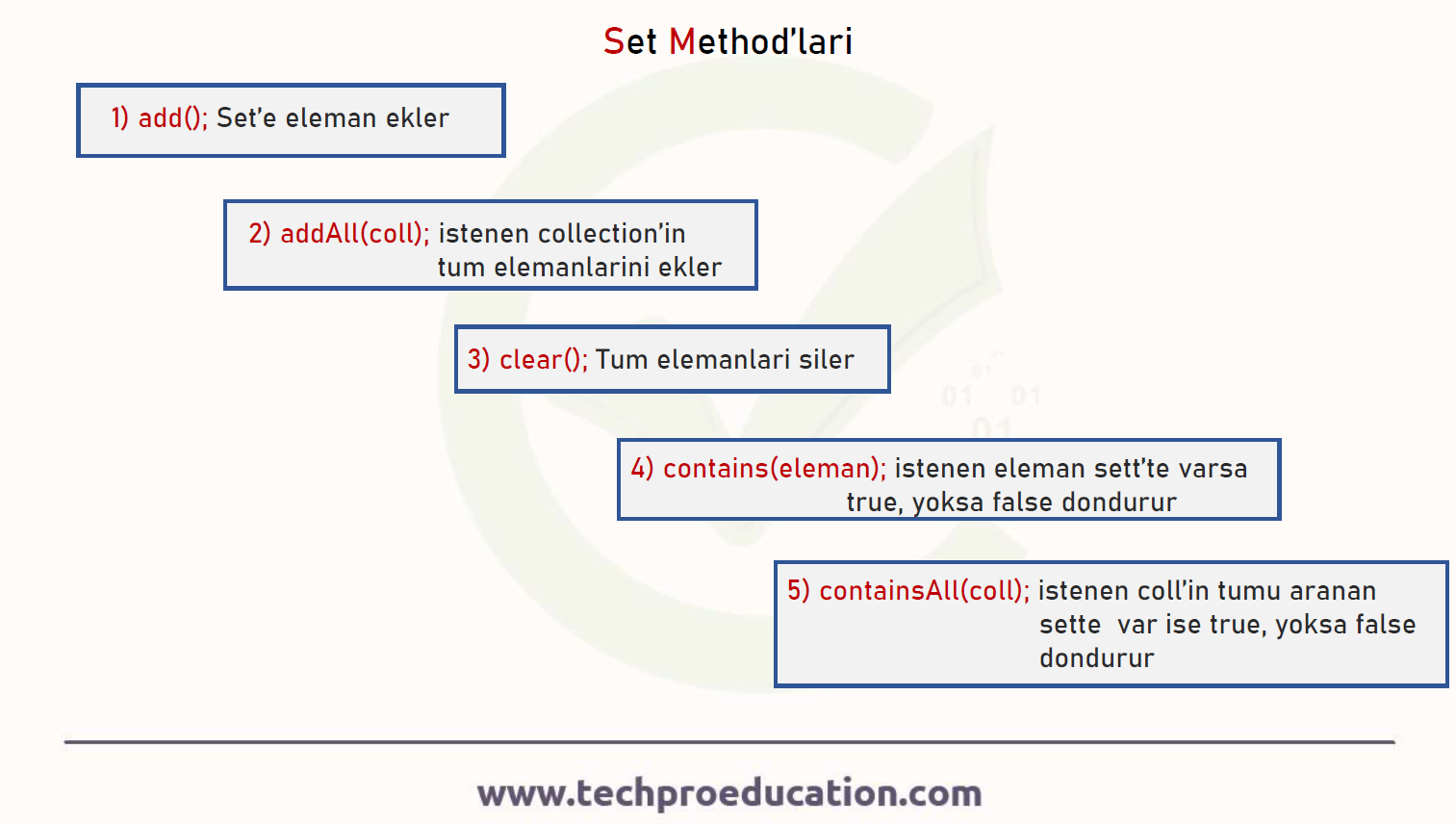
**Ozet: Hash mekanizmasi ile datayi Java heap memory’de yerlestirir, aramak istedigimizde cok hizli buluruz.**

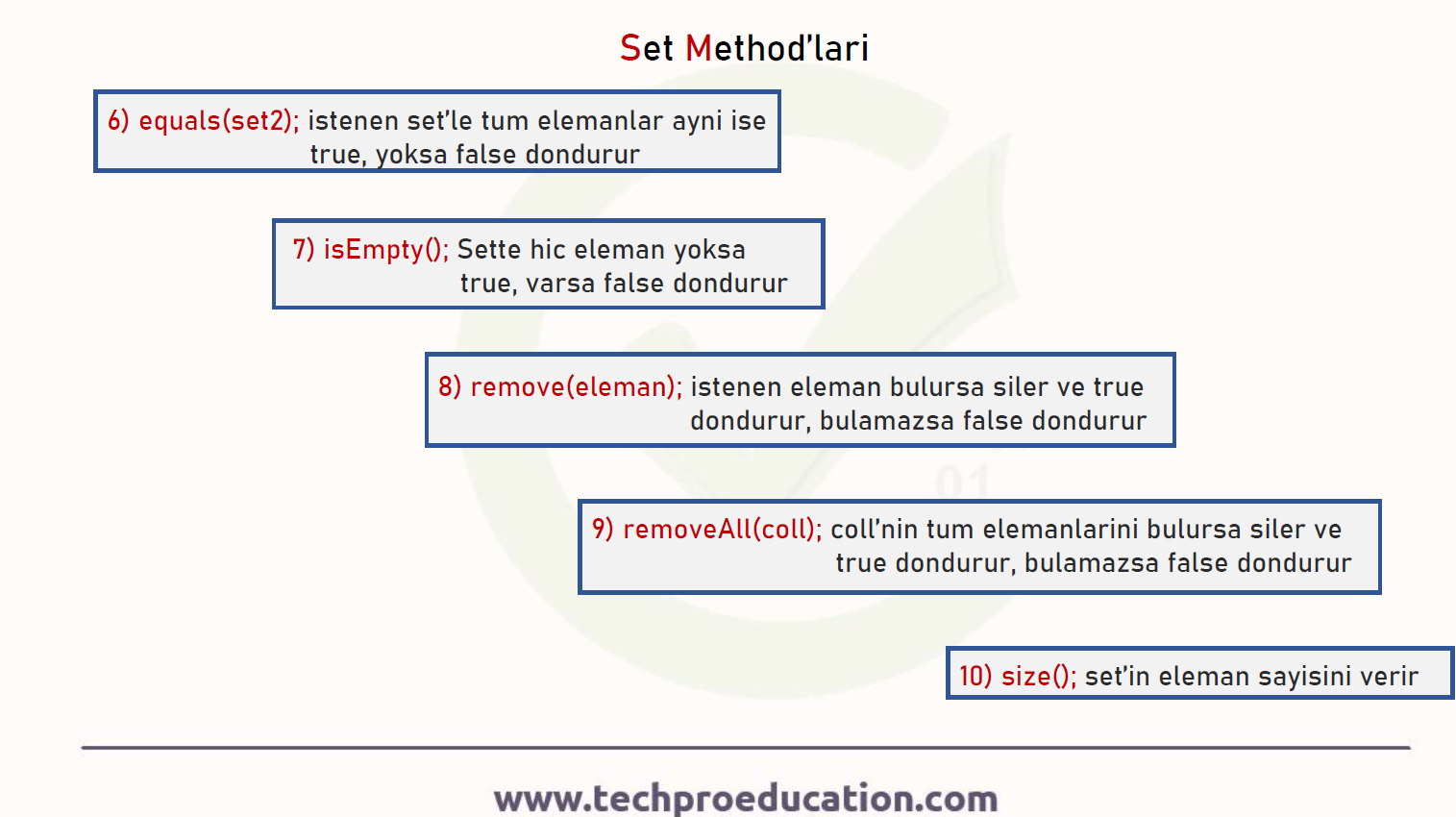
int arr[]= {3,4,6,4,2,4,6,8,6,5,3,1,2,4,6,7};

Burdaki herbir elemani alip Set’e ekledik tekrar eden elemanlar gitti.

Set <Integer> benzersizSet =new HashSet<>();

for (Integer each : arr) {//arr dizisine git Integer tipindeki herbir datayi al   
 benzersizSet.add(each); } //Set dublication’ a izin vermedi

Set'in containsAll() yöntemi, iki kümenin aynı öğeleri içerip içermediğini  
kontrol etmek için kullanılır. Bir kümeyi parametre olarak alır ve bu kümenin tüm öğeleri  
diğer kümede mevcutsa True değerini döndürür.



*//retainAll iki array’in ortak elemanlarını tek kümede toplar*

*//Set'te unique olma sarti vardi queue'da unique olma sarti yok ayni eleman eklenebilir*

*// Deque null eklenebiliyor*

/\*  
 \* Set :   
 \* 1) unique(benzersiz tek) elemanlardan olusur, Set'lerin icinde bir eleman sadece bir kere kullanilabilir.   
 Mukerrer(duplicate) eleman olmaz.  
 \* 2) Sadece bir null degeri alabilir.(treeSet haric).  
 \* Java elementleri unique yapmak icin HASH ALGORITMASI kullanir.   
 \* Java her data icin hashing teknigini kullanarak bir code uretir.Bu code'u almak icin HashCode() methodu kullanilir.   
 \* Hashing, farkli buyuklukteki girdilerden sabit buyuklukte bir cikti olusturma surecine verilen isimdir.  
 \* Universitelerdeki ogrenci numaralari gibi bir ogrenci ismi soruldugunda numarasini bulursaniz onunla ilgili tum datalara ulasilabilir. Tipki Sql’deki primary key gibi  
   
 \* Set --> HashSet :   
 \* 1) HashSet, duplicationa(tekrarli eleman) izin vermez.  
 \* Eger bir elemani tekrar HashSet e eklemek isterseniz eski olan silinip, yeni olan uzerine yazilir.  
 \* 2) HashSet'ler Set'ler arasinda en hizli olanidir. bir algoritmaya gore kayit eder. degerlerin giris sirasina dikkat edilmez.  
 (LinkedHashSet haric)  
 \* 3) HashSet "null" object'ini eleman olarak kullanabilir.   
 \* Ancak birden fazla null degerini bir HashSet e eklemek isterseniz sadece bir tane null degeri olur.   
   
 \* LinkedHashset :   
 \* 1) Tekrarli eleman kabul etmezler. Elemanlari ekleme sirasina(insertion order) gore dizerler.   
 \* 2) LinkedHashSet, HashSet'den daha yavas calisir.   
 \* 3) LinkedHashSet "null" object'ini eleman olarak kullanabilir.  
 \* 4) Ekleme ve remove islemlerinde hizlidirlar.  
   
   
 \* TreeSet :   
 \* 1) TreeSet tekrarli eleman kabul etmez, cunku Set'dir.   
 \* 2) Elemanlari natural Order'a(String ise alfabetik, sayi ise kucukten buyuge)gore dizer.   
 \* 3) TreeSet, setlerin en yavasidir. Bu yuzden TreeSet kullanirken dikkatli olunmali...  
 \* 4)AHAN DA TRICK :TreeSet "null" object'ini eleman olarak kullanilamaz.  
 **HashMap Sınıfı:** Karışık eşleme de denilebilir. Eşleme tablosuna eleman ekleme, çıkartma işlemlerinde ve anahtarı verilen elemanı tabloda bulma işlemlerinde oldukça etkilidir.

**LinkedHashMap Sınıfı:** Sıralı eşleme denilebilir. HashMap Sınıfından türemiştir. Bu sınıf elemanları sıralı olarak tutar. Bu sıralama ekleme sırası ya da erişim sırasına göre olabilir.

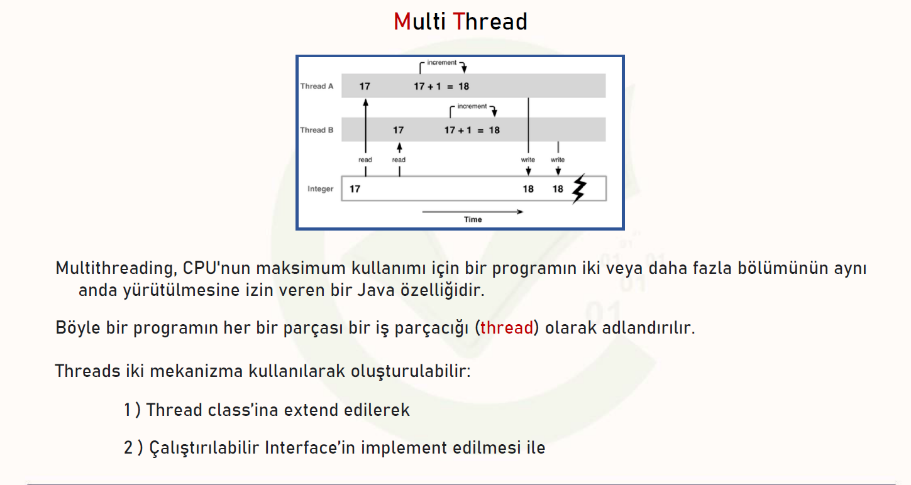
**TreeMap Sınıfı:** Map arabirimini uygulayan sortedMap arabirimini uygular. Eşleme tablosundaki anahtarları sıra ile tutar. Bu sıra tablodaki anahtarların karşılaştırılması ile oluşur. Yani küçükten büyüğe gelir. 1.2.3 , a.b.c vb.

\* Set Methods :   
 \* .add(eleman);-->Set e eleman ekler  
 \* .addAll(collection);-->istenen collection in tum elemanlarini ekler  
 \* .contains(eleman);-->istenen eleman set te varsa true, yoksa false dondurur.  
 \* .containsAll(collection);-->istenen collection in tumu aranan sette var ise true, yoksa false dondurur  
 \* .remove(eleman);-->istenen eleman bulursa siler ve true dondurur, bulamazsa false dondurur   
 \* .removeAll(collection);-->istenen collection in tum elemanlarini bulursa siler ve true dondurur, bulamazsa false dondurur  
 \* .equals(set2);-->istenen set2 ile tum elemanlar ayni ise true, yoksa false dondurur  
 \* .retainAll(collection1);-->collection1 nin elemanlarinin disindaki tum elemanlari siler, silme islemi yapti ise true, yoksa false  
 dondurur(kesisen ortak elemanlari dondurur.)  
 \* .clear();-->sett'teki Tum elemanlari siler

\* .isEmpty();-->Sette hic eleman yoksa true, varsa false dondurur   
 \* .size();-->set in eleman sayisini verir

**Queue (kuyruk) Sınıfı**

Bu sınıf stack sınıfından farklı olarak özel durumlar dışında ilk giren ilk çıkar mantığı ile çalışmaktadır.  
FIFO (first in first out)  
**Bu sınıfa ait metotlar;**  
**add(eleman):** Parametrede verilen elemanı kuyruğa ekler. İşlemin başarısız olması durumunda hata fırlatır.  
**offer(eleman):** Parametrede verilen elemanı kuyruğa ekler. İşlemin başarısız olması durumunda null döner.  
**poll():** Kuyruğun başındaki elemanı kuyruktan çıkartır.  
**peek():** Kuyrukta sıradaki elemana ulaşmak için kullanılır.



**Multi threading Nedir?**

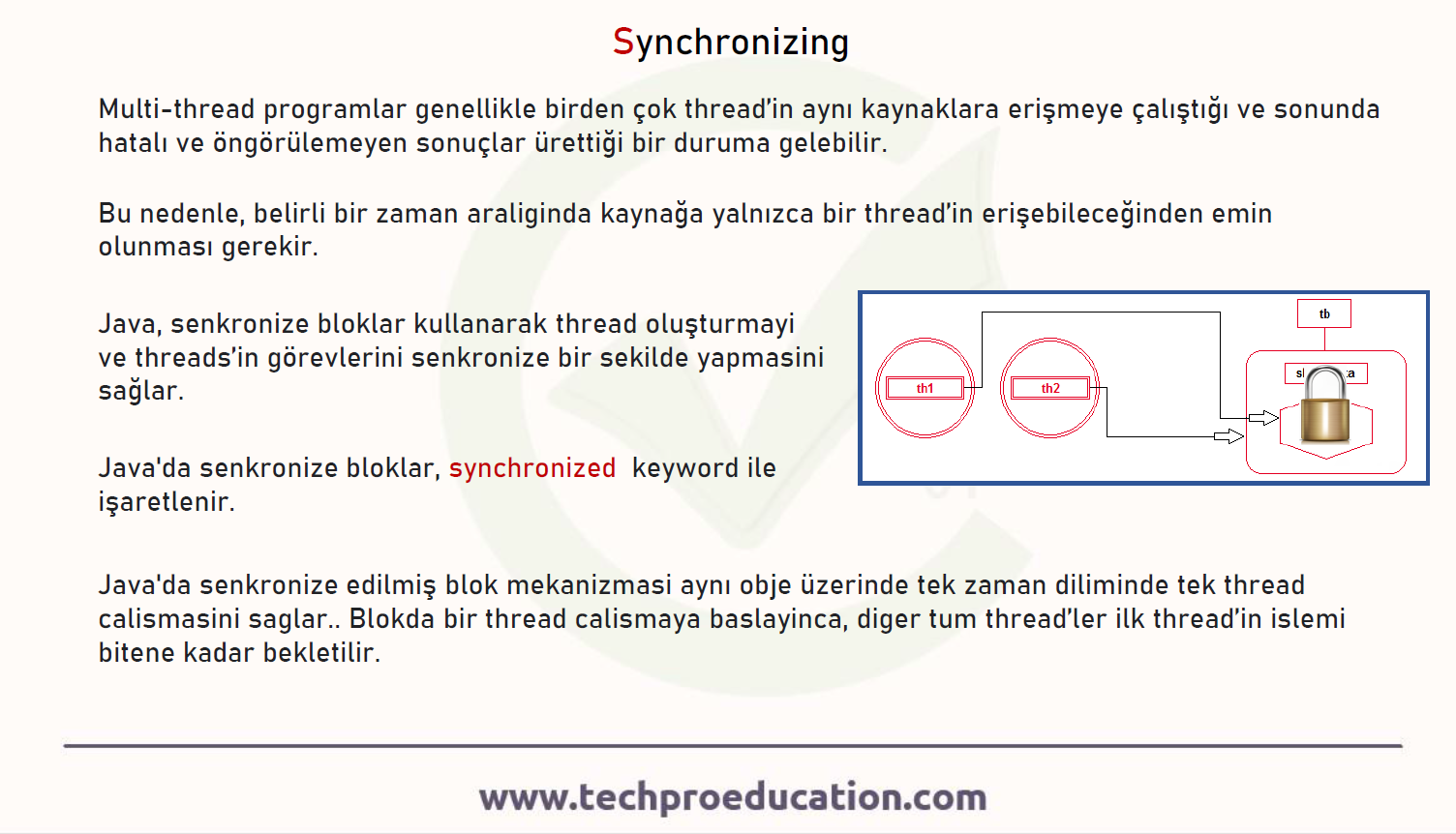
Java’da çoklu thread, maksimum CPU kullanımıyla aynı anda iki veya daha fazla iş parçacığı yürütme işlemidir.

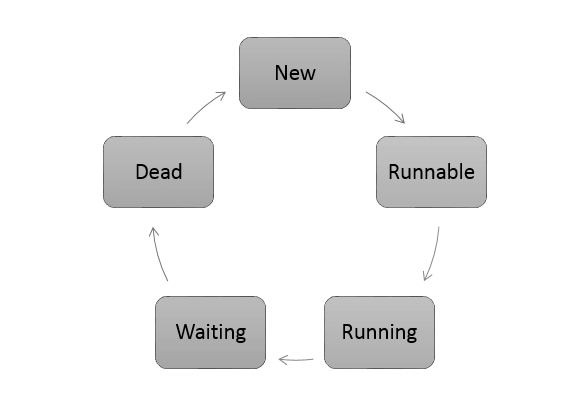
Çok iş parçacıklı uygulamalar, iki veya daha fazla iş parçacığının aynı anda çalıştığı yerlerdir; Bu nedenle, Java’da eşzamanlılık olarak da bilinir. Bu çoklu görev, birden fazla işlem CPU, bellek vb. ortak kaynakları paylaştığında yapılır.

Çoklu thread avantajları:

Kullanıcılar engellenmez, çünkü iş parçacıkları bağımsızdır ve zaman zaman birden fazla işlem yapabiliriz

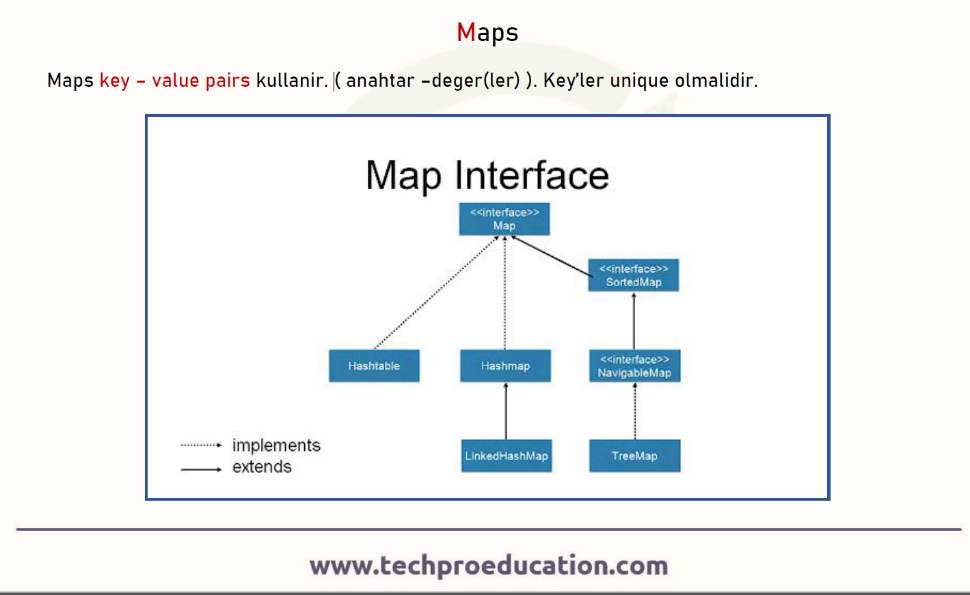
* Bu threadler bağımsız olduğu için, bir iş parçacığı bir istisna ile karşılaşırsa, diğer iş parçacıkları etkilenmez.

**



### Java’da iş parçacığı yaşam döngüsü

1. **New:**Bu aşamada, konu “Thread sınıfı” sınıfı kullanılarak yaratılır. Program threadi başlayana kadar bu durumda kalır. Ayrıca doğuştan thread olarak da bilinir.
2. **Runnable:** Bu bölümde, iş parçacığının örneği bir start yöntemiyle çağrılır. Thread kontrolü, çalışmayı bitirmesi için programlayıcıya verilir. İş parçacığının çalıştırılıp çalıştırılmayacağı zamanlayıcıya bağlıdır.
3. **Running:** İş parçacığı yürütmeye başladığında, durum “Running-Çalışan” durumuna değiştirilir. Zamanlayıcı, iş parçacığı havuzundan bir iş parçacığı seçer ve uygulamada çalışmaya başlar.
4. **Waiting:** Bir iş parçacığının beklemesi gereken durum budur. Uygulamada birden fazla iş parçacığı bulunduğundan, iş parçacığı arasında eşitlemeye ihtiyaç vardır. Bu nedenle, bir iş parçacığı diğer iş parçacığı işlenene kadar beklemek zorundadır. Dolayısıyla bu duruma bekleme durumu denir.
5. **Dead:** Bu, iş parçacığının sonlandırıldığı durumdur. İş parçacığı çalışıyor durumda ve işleme tamamlanır tamamlanmaz “dead state” durumuna geçer.

*MAP GIRIS*

*Maplerde key value pairs kullanılır.ornegin 101 nolu arabayi getiriyoruz*

*List’te -Set’te- Array’de olmayan ozellik budur….*

*Tipki sql’deki ID gibi..*

*KEY CHAR DA OLABILIR AKSI IDDIA EDILEMEZ…*

**Map arabirimine ait metotlar;size():** Eşleme tablosundaki eşleme sayısını döner.  
**values():** Eşleme tablosundaki elemanlardan bir koleksiyon nesnesi döner.  
**clear():** Map’teki tüm eşleşmeleri kaldırır.  
**isEmpty():** Eşleşme tablosu boş ise yani eşleşme tablosunda eleman yoksa true döner.  
**entrySet():** Eşleşme tablosunda bulunan elemanlar için bir set nesnesi-küme döner.  
**keySet():** tabloda bulunan anahtarlar için set nesnesi döner.  
**put(anahtar,eleman):** Parametrede verilen anahtar elemanı birbirleri ile eşleştirip, eşleme tablosuna yazar.  
**putAll(eslemeTablosu X):** Parametrede verilen eş. tab. X in tüm elemanlarını ilgili eşleme tablosuna ekler.  
**get(anahtar):** Parametrede verilen anahtar ile eşlesen elemanı döner.  
**containsKey(anahtar):** parametrede verilen anahtar ile eşlesen bir eleman varsa true döner.  
**containsValue(eleman):** parametrede verilen anahtar ile eşlesen bir eleman varsa true döner.  
**remove(anahtar):** parametrede verilen değeri eşleme tablosundan çıkarır.