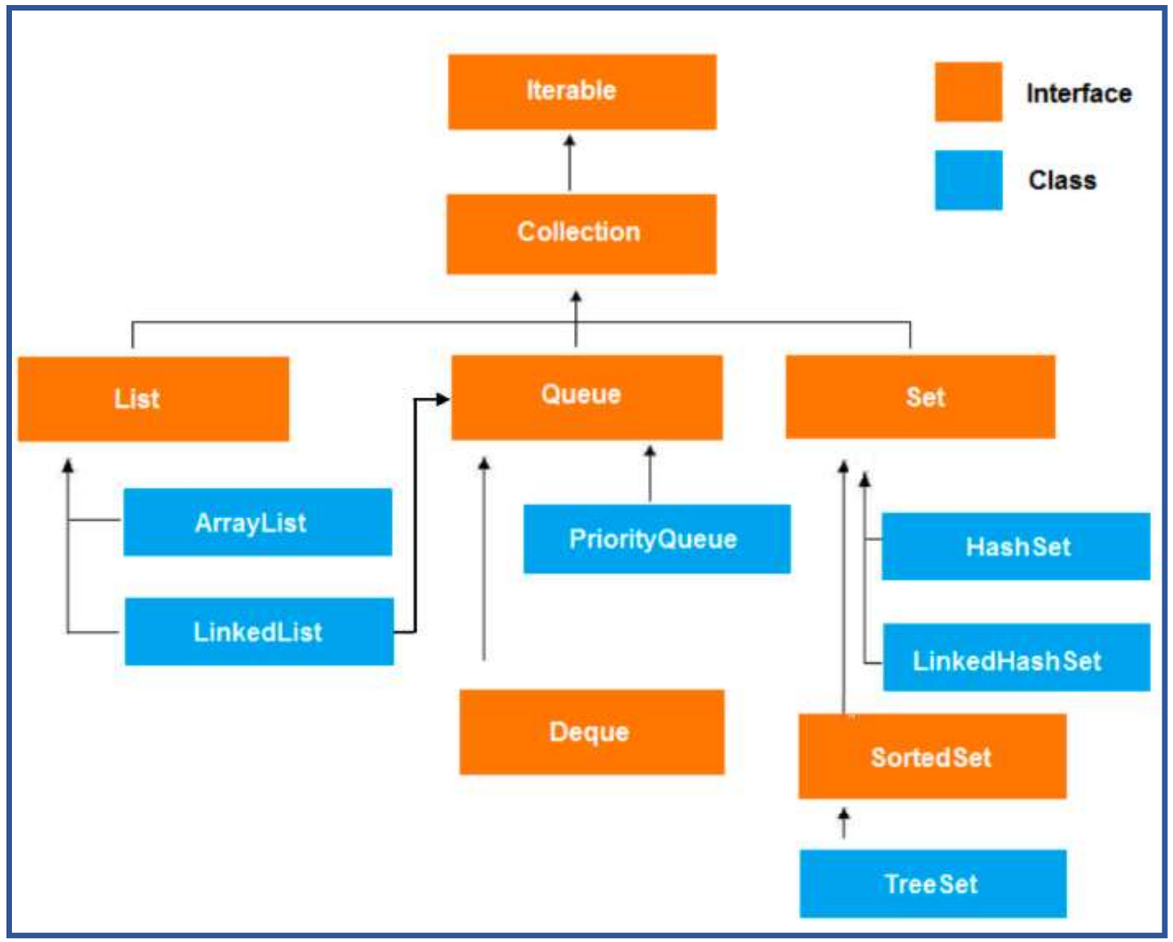
Collections

Collections : nesnelerden oluşan bir topluluğu bir arada tutan yapılardir.



5 kelime'ye dikkat

1) Set (Kume)

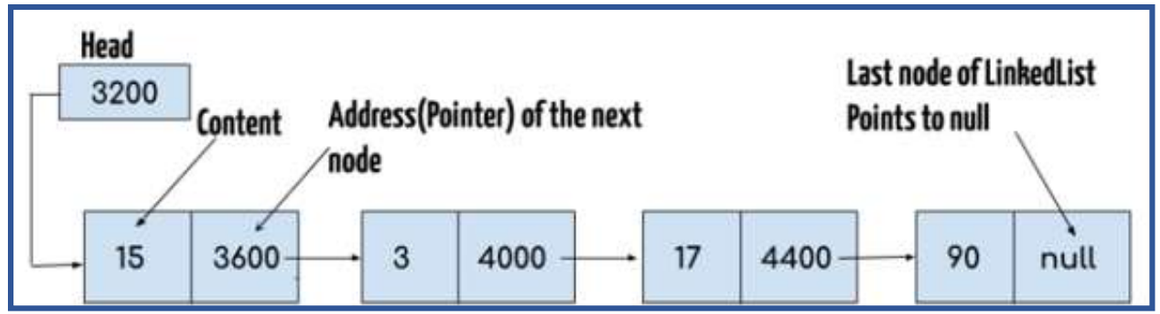
2) Queue (Sira)

3) Linked (Bagli)

4) Tree (DogalSirali)

5) Hash

Collections / Linked List



Linked List (Class)

1) Ilk eleman her zaman head'dir ve head'de data yoktur, sadece address vardir.

2) Son eleman(tail) null'i point eder.

3) Her elemanin icinde data ve address kismi olmak uzere iki kisim vardir.

4) Tum elemanlar pointer'lar / address'ler kullanilarak birbirine baglanir.

5) Her eleman node olarak adlandirilir.

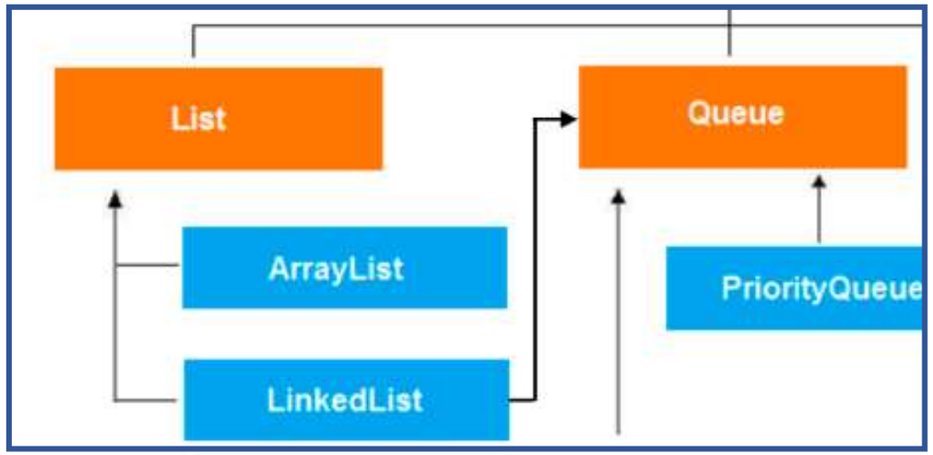
7) Pointer yapisindan dolayi bir elemana ulasmada yavastirlar

Collections / Linked List

Linked List, 2 interface'in child class'idir. Obje olustururken data turu olarak istedigimiz parent interface'i secebilir ve o interface'deki ozellikleri kullanabiliriz.

Data turu olarak LinkedList sectigimizde de ayni method'lari kullanabiliriz, cunku LinkedList concrete bir class'dir ve parent'i olan interface'lerdeki tum method'lari override etmek zorundadir.

*** LinkedList 2 interface'i implement ettigi icin her ikisinin ustun ozelliklerini kullanabilir.




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

    LinkedList<Integer> ll1 = new LinkedList<>();

    ll1.add(10); // LinkedList class'indan add methodu
    ll1.element(); // LinkedList class'indan element methodu
    ll1.remove(1); // LinkedList class'indan remove methodu

    Queue<Integer> ll2 = new LinkedList<>();
    ll2.add(13); // Queue class'indan add methodu
    ll2.element(); // Queue class'indan element methodu

    List<Integer> ll3 = new LinkedList<>();
    ll3.add(15); // List class'indan add methodu
    ll3.remove(0); // List class'indan remove methodu
}
```

Linked List Method'lari

1) add( ); LinkedList'in sonuna istenen elemani ekler

2) add(1,"A"); istenen index'e istenen elemani ekler

3) addAll(coll); istenen collection'in tum elemanlarini ekler

4) addAll(2, coll); istenen collection'in tum elemanlarini istenen index'e ekler

Linked List Method'lari

5) addFirst( ); istenen elemani, ilk eleman olarak ekler

6) addLast( ); istenen elemani, son eleman olarak ekler

7) remove( ); ilk elemani siler

8) removeFirst( ); ilk elemani siler (daha hizlidir)

9) remove(index); istenen indexdeki elemani siler ve silinen elemani dondurur

Linked List Method'lari

10) remove(eleman); istenen elemani siler sildi ise true, bulamadi ise false dondurur

11) removeFirstOccurrence("str"); istenen elemanin,ilkini siler

12) removeLast( ); son elemani siler

13) removeAll(list); istenen listedekitum elemanlari siler

Linked List Method'lari

14) contains(eleman); istenen eleman listede var ise true, yoksa false dondurur

15) containsAll(liste); istenen listenin tumu aranan listede var ise true, yoksa false dondurur

16) get(index); istenen indexdeki elemani getirir

Collections / Linked List

Soru : Node'lari "Ali", "Veli", "Can" ve "Ayse" olan bir LinkedList olusturun.

Kullanicidan bir isim alin. Bu isim LinkedList'de varsa silin ve kullaniciya "Bu isim LinkedList'de vardi ve silindi“ diye mesaj verin.

Bu isim LinkedList'de yoksa "Bu isim LinkedList'de yok bu yuzden silinemedi" diye mesaj verin.

```
public static void main(String[] args) {
    LinkedList<String> ll1 = new LinkedList<>();
    ll1.add("Ali");
    ll1.add("Veli");
    ll1.add("Can");
    ll1.add("Ayse");

    Scanner scan = new Scanner(System.in);
    System.out.println("Bir isim giriniz");
    String isim = scan.nextLine();

    System.out.println(ll1);

    if(ll1.remove(isim)) {
        System.out.println("Bu isim LinkedList'de vardi ve silindi");
        System.out.println(ll1);
    }else {
        System.out.println("Bu isim LinkedList'de yok bu yuzden silinemedi");
        System.out.println(ll1);
    }
    scan.close();
}
```

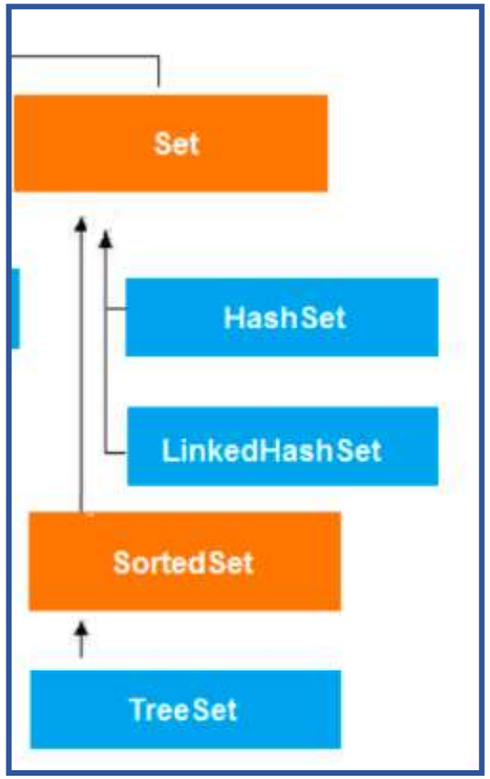
Collections / Sets

Set (interface) , matematikteki kume mantigiyla calisir, her element unique'dir.

Java elementleri unique yapmak icin HASH ALGORITMASI kullanir.

Set, direk kullanilamaz cunku interface'dir ve obje olusturulamaz. 3 Child class'indan bizim icin onemli olan ozellige gore istedigimizi kullanabiliriz.

Collections'in bir ozelligi de farkli data turunden elementleri ekleyebilmenizdir. Bunun icin esitligin sol tarafindaki <> (data turu) kaldirilabilir veya data turu olarak Object yazilabilir. Ancak bu tavsiye edilmez cunku Java'nin cok fazla Casting yapmasi gerekir.



Collections / HashSets

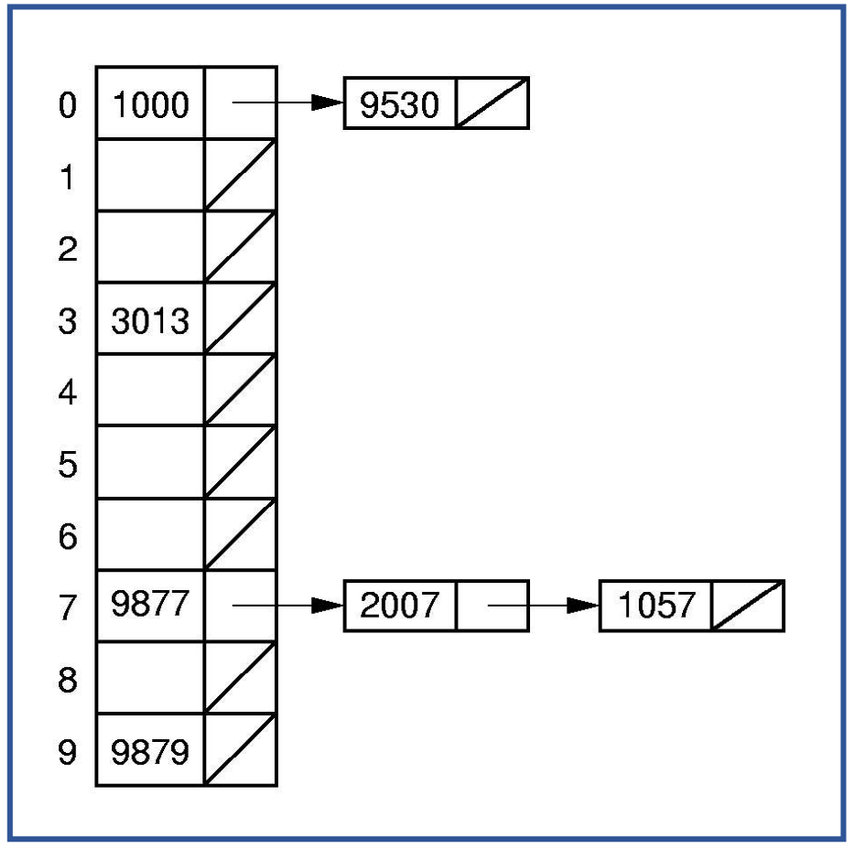
Hashing, farklı büyüklükteki girdilerden sabit büyüklükte bir çıktıya donusturme sürecine verilen isimdir.

Bu işlem, hash fonksiyonları olarak bilinen matematiksel formüllerin kullanımıyla yapılır.

Universitelerdeki ogrenci numaralari gibi bir ogrenci ismi soruldugunda numarasini bulursaniz onunla ilgili tum bilgilere ulasabilirsiniz.

Farklı hash fonksiyonları farklı büyüklüklerde çıktı yaratır fakat her bir hashing algoritması için olası çıktı büyüklüğü her zaman sabittir.

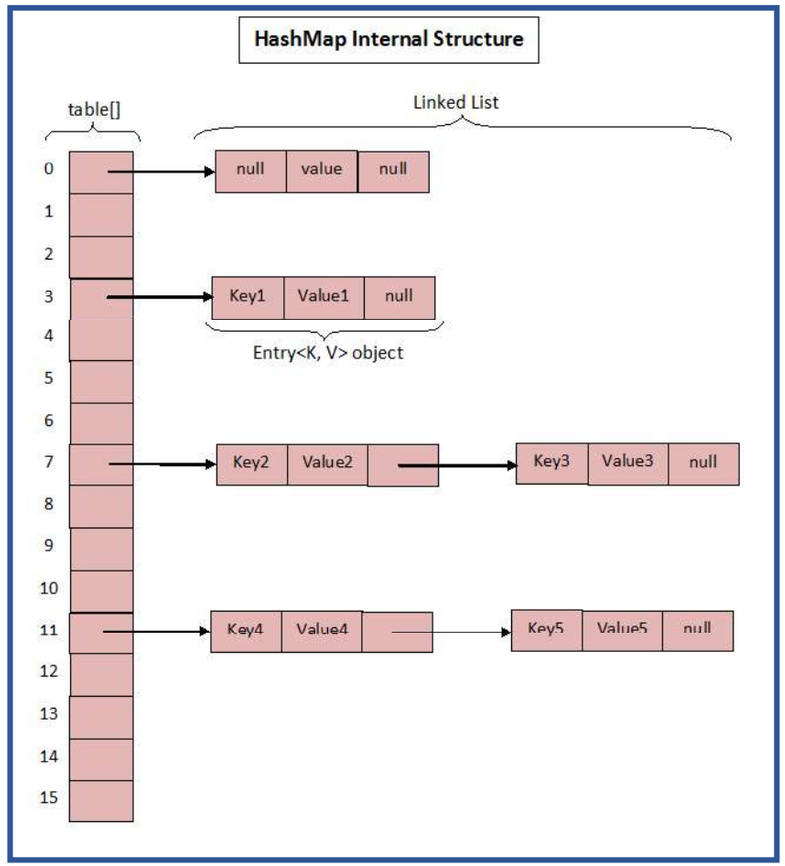
Bir collection'in hash degerini ogrenmek icin hashCode( ) method'u kullanilir.



Collections / HashSets

Hashing Nasil Calisir ?

- Bir HashCollection olusturuldugunda Java 16 bucket olusturur ve elementleri bu bucket'lara yerlestirmeye baslar.
- Olusturulan bucket'larin %75'i doldugunda Java 16 bucket daha olusturur. Buna Load Factor denir.
- Java kullandigimiz key'i kullanarak hash kod uretir. Eger uretilen hash kod daha once uretilen bir hash kod ile ayni ise buna Hash Collision denir
- Hash Collision gerceklestiginde cozum icin 2 yol vardir. A) LinkedList kullanmak B) Formulle belirlenen yeni bir hash kod uretmek



Collections / HashSets

HashSet, elemanlari icin herhangi bir siralama yapmaz. Elemanlari yazdirdiginizda veya cagirdiginizda herhangi bir siralama ile gelebilirler.

HashSet, duplication'a izin vermez. Eger bir elemani tekrar HashSet'e eklemek isterseniz eski olan silinip, yeni olan uzerine yazilir.

HashSet, null degere izin verir. Bununla birlikte birden fazla null degerini bir HashSet'e eklemek isterseniz sadece bir tane null degeri olur.

Index	
0	
1	
-	
-	
-	
11	defabc
12	
13	
14	cdefab
-	
-	
-	
-	
23	bcdefa
-	
-	
-	
38	abcdef
-	
-	

Collections / Sets

Soru: Verilen bir arraydeki tekrarli elemanlari silip, sadece unique degerlerden olusan bir liste haline getiren bir program yaziniz.

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

    int arr[]= {2,5,3,4,2,1,5,4,6,3,2,1,5,4};
    Set<Integer> hs1 = new HashSet<>();

    for (Integer each : arr) {
        hs1.add(each);
    }

    System.out.println(hs1);
}
```

```
[1, 2, 3, 4, 5, 6]
```

Set Method'lari

1) add(); Set'e eleman ekler

2) addAll(coll); istenen collection'in tum elemanlarini ekler

3) clear(); Tum elemanlari siler

4) contains(eleman); istenen eleman sett'te varsa true, yoksa false dondurur

5) containsAll(coll); istenen coll'in tumu aranan sette var ise true, yoksa false dondurur

Set Method'lari

6) equals(set2); istenen set'le tum elemanlar ayni ise true, yoksa false dondurur

7) isEmpty(); Sette hic eleman yoksa true, varsa false dondurur

8) remove(eleman); istenen eleman bulursa siler ve true dondurur, bulamazsa false dondurur

9) removeAll(coll); coll'nin tum elemanlarini bulursa siler ve true dondurur, bulamazsa false dondurur

10) size(); set'in eleman sayisini verir

Set Method'lari

11) retainAll(coll); coll'nin elemanlarinin disindaki tum elemanlari siler, silme islemi yapti ise true, yoksa false dondurur

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

    Set<String> lhs1 = new TreeSet<>();
    lhs1.add("Ali");
    lhs1.add("Canan");
    lhs1.add("Veli");
    lhs1.add("Remziye");
    System.out.println(lhs1);//[Ali, Canan, Remziye, Veli]

    Set<String> lhs2 = new TreeSet<>();
    lhs2.add("Ali");
    lhs2.add("Canan");

    System.out.println(lhs1.retainAll(lhs2)); // true
    System.out.println(lhs1); // [Ali, Canan]

}
```

Collections / LinkedHashSets

1) Tekrarli eleman kabul etmezler

2) Elemanlari ekleme sirasina(insertion order) gore dizerler.

3) Ekleme ve remove islemlerinde hizlidirlar.

4) LinkedHashSet, HashSet'den yavastir.

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

    Set<String> lhs1 = new LinkedHashSet<>();
    lhs1.add("Ali");
    lhs1.add("Canan");
    lhs1.add("Veli");
    lhs1.add("Remziye");
    System.out.println(lhs1);//[Ali, Canan, Veli, Remziye]

}
```

Collections / Sets

Soru 1 : Bir TreeSet ve HashSet'e random 100 sayi ekleyin, islem surelerini kiyaslayin

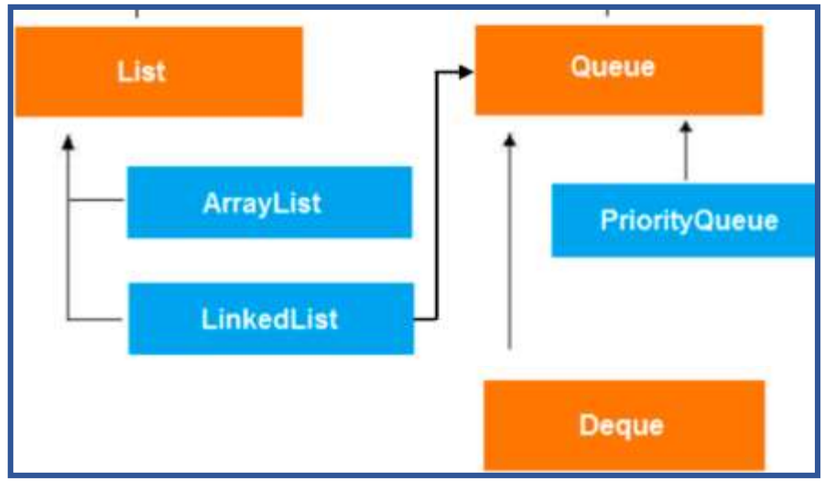
Soru 2 : Ilk soruya 3.bir islem ekleyelim, set'i Hashset olarak olusturup elemanlari ekleyelim ve sonra TreeSet'e cevirip yazdiralim

Long time=System.currentTimeMilis() method'unu kullanin

Collections / Queue

Queue interface'dir dolayisiyla constructor'i yoktur.

Queue olusturmak icin child class'I olan LinkedList veya PriorityQue kullanilabilir.



1) PriorityQueue constructor'i kullanarak Queue uretirseniz,

Java kendisi bir "priority"(oncelik) kurali uretir ve urettigi bu kurala gore elemanlari dizer.

Istersek biz kendi"priority"(oncelik) kuralimizi uretip elemanlari bu kurala gor dizebiliriz.

2) LinkedList constructor'i kullanarak Queue uretirseniz elemanlari insertion sirasina gore ekler

NOT : Queue icin ayirici ozellik : Elemanlar en sona eklenir ve en bastan silinir.

Bu sisteme FIFO(First In First Out) denir. Eczaneler, Yemekhaneler bu sistemi kullanir..

Collections / Queue Method'lari

1) peek( ); ilk elemani silmeden bize retrun eder.

2) poll( ); ilk elemani queue'dan siler ve bize return eder

3) offer( ); eleman eklemek icin kullanilir

NOT : remove( ) ve poll( ) ilk elemani siler ve return eder. Ama collection'da eleman yoksa remove() methodu Exception atar poll( ) methodu Exception atmaz null return eder.

Collections / Deque

Deque

Double Ended Queue

Queue'larda FIFO gecerli, Deque'lerde hem FIFO hem de LIFO(Last In First Out) gecerlidir.

Deque bir interface'dir dolayisiyla constructor'i yoktur. LinkedList constructor'I kullanilarak deque olusturulabilir.

Deques do not accept Null as an element.

Deque'de ilk ve son eleman onemli oldugu icin ilk ve son elemana ozel bircok method vardir.

getFirst( ) - getLast( )
peekFirst( ) - peekLast( )
pollFirst( ) - pollLast( )