

JAVA COLLECTIONS FRAMEWORK

[ArrayList](#), [HashMap](#), [HashSet](#), [LinkedList](#)

Adım Adım Örneklerle Collections Öğrenme Rehberi

GİRİŞ

Collections Framework Nedir?

Collections Framework, Java'da birden fazla veriyi bir arada tutmak ve yönetmek için kullanılan hazır yapılar bütünüdür. Bu yapılar sayesinde dizilerden çok daha esnek ve güçlü veri yapıları kullanabiliriz.

Normal Dizi vs Collections:

- **Normal Dizi:** Sabit boyutlu, boyutu değiştirilemez
- **Collections:** Dinamik boyutlu, otomatik büyür/küçülür

Ana Collections Yapıları:

- ArrayList: Dinamik dizi, index ile erişim
- LinkedList: Bağlı liste, hızlı ekleme/silme
- HashSet: Benzersiz elemanlar, tekrar yok
- HashMap: Anahtar-değer çiftleri

1. ARRAYLIST - DİNAMİK DİZİ

1.1 ArrayList Nedir?

ArrayList = Boyutu değişebilen dizi

Normal diziler sabit boyutludur. Örneğin `int[] sayilar = new int[5]` dediğinizde sadece 5 eleman tutabilirsiniz. ArrayList ise ihtiyaca göre otomatik olarak büyür veya küçülür.

Temel Özellikler:

- Index ile erişim var (0'dan başlar)
- Sıralı bir yapıdır (ekleme sırasını korur)
- Tekrar eden elemanlar olabilir
- Hızlı erişim (get metodu çok hızlı)

1.2 ArrayList Temel İşlemler

Kod Örneği:

```
import java.util.ArrayList;

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

        // ArrayList oluşturma
        // <String> = İçinde sadece String tutulabilir
        ArrayList<String> isimler = new ArrayList<>();

        // 1. EKLEME - add()
        isimler.add("Ali");           // Index 0
        isimler.add("Ayşe");         // Index 1
        isimler.add("Mehmet");       // Index 2

        System.out.println(isimler);
        // Çıktı: [Ali, Ayşe, Mehmet]

        // Belirli bir index'e ekleme
        isimler.add(1, "Fatma");
        System.out.println(isimler);
        // Çıktı: [Ali, Fatma, Ayşe, Mehmet]

        // 2. ERİŞİM - get()
        String ilkIsim = isimler.get(0);
        System.out.println("İlk: " + ilkIsim);
        // Çıktı: Ali

        // 3. GÜNCELLEME - set()
        isimler.set(0, "Ahmet");
        System.out.println(isimler);
        // Çıktı: [Ahmet, Fatma, Ayşe, Mehmet]

        // 4. SİLME - remove()
        isimler.remove(1);           // Index ile
```

```

        isimler.remove("Ayşe");        // İsimle

        // 5. BOYUT - size()
        int boyut = isimler.size();
        System.out.println("Boyut: " + boyut);

        // 6. BOŞ MU? - isEmpty()
        boolean bos = isimler.isEmpty();

        // 7. İÇERİYOR MU? - contains()
        boolean varMi = isimler.contains("Ahmet");
        System.out.println("Ahmet var mı? " + varMi);
    }
}

```

1.3 ArrayList ile Döngüler

Kod Örneği:

```

ArrayList<Integer> notlar = new ArrayList<>();
notlar.add(85);
notlar.add(92);
notlar.add(78);

// 1. YÖNTEM: Klasik For Döngüsü
for (int i = 0; i < notlar.size(); i++) {
    System.out.println(notlar.get(i));
}

// 2. YÖNTEM: For-Each Döngüsü (EN YAYGINI)
for (Integer not : notlar) {
    System.out.println(not);
}

// 3. YÖNTEM: forEach + Lambda (Modern Java)
notlar.forEach(not -> System.out.println(not));

```

1.4 Öğrenci Sistemi - Gerçek Hayat Örneği

Bu örnekte bir öğrenci yönetim sistemi oluşturacağız:

```

class Ogrenci {
    private String ad;
    private int numara;
    private double ortalama;

    public Ogrenci(String ad, int numara, double ort) {
        this.ad = ad;
        this.numara = numara;
        this.ortalama = ort;
    }

    public String getAd() { return ad; }
}

```

```
        public double getOrtalama() { return ortalama; }
    }

    public class OgrenciSistemi {
        public static void main(String[] args) {
            ArrayList<Ogrenci> ogrenciler = new ArrayList<>();

            // Öğrenci ekle
            ogrenciler.add(new Ogrenci("Ali", 1001, 85.5));
            ogrenciler.add(new Ogrenci("Ayşe", 1002, 92.3));
            ogrenciler.add(new Ogrenci("Mehmet", 1003, 78.9));

            // Başarılı öğrencileri bul (ortalama > 85)
            System.out.println("=== BAŞARILI ÖĞRENCİLER ===");
            for (Ogrenci ogr : ogrenciler) {
                if (ogr.getOrtalama() > 85) {
                    System.out.println(ogr.getAd());
                }
            }
        }
    }
}
```

2. HASHMAP - ANAHTAR-DEĞER ÇİFTLERİ

2.1 HashMap Nedir?

HashMap = Sözlük gibi çalışan yapı (Key-Value)

HashMap, anahtar-değer (key-value) çiftlerini tutar. Tıpkı bir sözlük gibi: kelime (key) → anlamı (value).

Temel Özellikler:

- Key (anahtar) benzersiz olmalı, tekrar edemez
- Value (değer) tekrar edebilir
- Key ile çok hızlı arama yapılır
- Sıralama garantisi yok

2.2 HashMap Temel İşlemler

Kod Örneği:

```
import java.util.HashMap;

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

        // HashMap oluşturma
        // <String, String> = <Key tipi, Value tipi>
        HashMap<String, String> rehber = new HashMap<>();

        // 1. EKLEME - put()
        rehber.put("Ali", "555-1234");
        rehber.put("Ayşe", "555-5678");
        rehber.put("Mehmet", "555-9012");

        System.out.println(rehber);
        // Çıktı: {Ali=555-1234, Ayşe=555-5678, ...}

        // 2. ERİŞİM - get()
        String telefon = rehber.get("Ali");
        System.out.println(telefon); // 555-1234

        // 3. GÜNCELLEME - put() (aynı key ile)
        rehber.put("Ali", "555-9999");
        System.out.println(rehber.get("Ali")); // 555-9999

        // 4. SİLME - remove()
        rehber.remove("Mehmet");

        // 5. VAR MI? - containsKey()
        boolean varMi = rehber.containsKey("Ali");
        System.out.println("Ali var mı? " + varMi);

        // 6. VALUE VAR MI? - containsValue()
        boolean numVarMi = rehber.containsValue("555-5678");
```

```

        // 7. BOYUT - size()
        int boyut = rehber.size();
        System.out.println("Kayıt sayısı: " + boyut);
    }
}

```

2.3 HashMap ile Döngüler

Kod Örneği:

```

import java.util.Map;

HashMap<String, Integer> notlar = new HashMap<>();
notlar.put("Ali", 85);
notlar.put("Ayşe", 92);

// 1. YÖNTEM: keySet() ile
for (String isim : notlar.keySet()) {
    int not = notlar.get(isim);
    System.out.println(isim + ": " + not);
}

// 2. YÖNTEM: entrySet() ile (EN İYİ!)
for (Map.Entry<String, Integer> entry : notlar.entrySet()) {
    String isim = entry.getKey();
    Integer not = entry.getValue();
    System.out.println(isim + " -> " + not);
}

// 3. YÖNTEM: forEach + Lambda
notlar.forEach((isim, not) -> {
    System.out.println(isim + " = " + not);
});

```

2.4 Envanter Sistemi Örneği

```

class Urun {
    private String ad;
    private double fiyat;
    private int stok;

    public Urun(String ad, double fiyat, int stok) {
        this.ad = ad;
        this.fiyat = fiyat;
        this.stok = stok;
    }

    public int getStok() { return stok; }
    public void setStok(int stok) { this.stok = stok; }
}

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

```

```
// Ürün kodu -> Ürün nesnesi
HashMap<String, Urun> envanter = new HashMap<>();

envanter.put("URN001", new Urun("Laptop", 15000, 5));
envanter.put("URN002", new Urun("Mouse", 250, 20));

// Ürün sorgulama
if (envanter.containsKey("URN001")) {
    Urun urun = envanter.get("URN001");
    System.out.println("Ürün bulundu!");
}

// Stok güncelleme
Urun laptop = envanter.get("URN001");
laptop.setStok(laptop.getStok() - 2); // 2 satıldı
}
}
```

3. HASHSET - BENZERSİZ ELEMANLAR

3.1 HashSet Nedir?

HashSet = Benzersiz (unique) elemanlar

Temel Özellikler:

- Tekrar eden eleman YOKTUR
- Çok hızlı arama
- Sıralama garantisi YOK
- Index YOK

3.2 HashSet Temel İşlemler

Kod Örneği:

```
import java.util.HashSet;

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

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

        // 1. EKLEME - add()
        sehirler.add("İstanbul");
        sehirler.add("Ankara");
        sehirler.add("İzmir");

        System.out.println(sehirler);
        // Çıktı: [İzmir, Ankara, İstanbul] (sıra rastgele)

        // TEKRAR EKLEME DENEMESİ
        boolean eklendi = sehirler.add("İstanbul");
        System.out.println("Eklendi mi? " + eklendi);
        // Çıktı: false (EKLENMEDİ!)

        // 2. İÇERİYOR MU? - contains()
        boolean varMi = sehirler.contains("Ankara");
        System.out.println("Ankara var mı? " + varMi);

        // 3. SİLME - remove()
        sehirler.remove("İzmir");

        // 4. BOYUT - size()
        int boyut = sehirler.size();

        // NOT: Index YOK!
        // sehirler.get(0); // HATA!
    }
}
```

3.3 Benzersiz Kelime Sayacı

```

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

        String metin = "java java python java python c++";
        String[] kelimeler = metin.split(" ");

        System.out.println("Toplam: " + kelimeler.length);
        // Çıktı: 6

        // HashSet otomatik tekrarları temizler!
        HashSet<String> benzersiz = new HashSet<>();
        for (String kelime : kelimeler) {
            benzersiz.add(kelime);
        }

        System.out.println("Benzersiz: " + benzersiz.size());
        // Çıktı: 3 (java, python, c++)

        System.out.println(benzersiz);
        // Çıktı: [java, python, c++]
    }
}

```

3.4 Set İşlemleri (Birleşim, Kesişim, Fark)

```

HashSet<String> mat = new HashSet<>();
mat.add("Ali");
mat.add("Ayşe");
mat.add("Mehmet");

HashSet<String> fizik = new HashSet<>();
fizik.add("Ayşe");
fizik.add("Mehmet");
fizik.add("Ahmet");

// 1. BİRLEŞİM (Union) - Tüm öğrenciler
HashSet<String> birlesim = new HashSet<>(mat);
birlesim.addAll(fizik);
System.out.println(birlesim);
// [Ali, Ayşe, Mehmet, Ahmet]

// 2. KESİŞİM (Intersection) - Her iki dersi de alanlar
HashSet<String> kesisim = new HashSet<>(mat);
kesim.retainAll(fizik);
System.out.println(kesisim);
// [Ayşe, Mehmet]

// 3. FARK (Difference) - Sadece matematik alanlar
HashSet<String> fark = new HashSet<>(mat);
fark.removeAll(fizik);
System.out.println(fark);
// [Ali]

```

4. LINKEDLIST - BAĞLI LİSTE

4.1 LinkedList Nedir?

LinkedList = Düğümlerle bağlı liste

ArrayList vs LinkedList:

- Erişim (get): ArrayList hızlı, LinkedList yavaş
- Ekleme/Silme (başta/sonda): LinkedList hızlı, ArrayList yavaş

4.2 LinkedList Özel Metodları

Kod Örneği:

```
import java.util.LinkedList;

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

        LinkedList<String> liste = new LinkedList<>();

        // Sona ekleme
        liste.add("A");
        liste.add("B");
        liste.add("C");
        System.out.println(liste); // [A, B, C]

        // BAŞA EKLEME - addFirst()
        liste.addFirst("BAŞLANGIÇ");
        System.out.println(liste);
        // [BAŞLANGIÇ, A, B, C]

        // SONA EKLEME - addLast()
        liste.addLast("SON");
        System.out.println(liste);
        // [BAŞLANGIÇ, A, B, C, SON]

        // İLK ELEMAN - getFirst()
        String ilk = liste.getFirst();
        System.out.println("İlk: " + ilk);

        // SON ELEMAN - getLast()
        String son = liste.getLast();
        System.out.println("Son: " + son);

        // İLK SİL - removeFirst()
        liste.removeFirst();
        System.out.println(liste); // [A, B, C, SON]

        // SON SİL - removeLast()
        liste.removeLast();
        System.out.println(liste); // [A, B, C]
```

```
}  
}
```

4.3 Queue (Kuyruk) Kullanımı

```
import java.util.Queue;  
import java.util.LinkedList;  
  
public class KuyrukOrnegi {  
    public static void main(String[] args) {  
  
        Queue<String> kuyruk = new LinkedList<>();  
  
        // KUYRUĞA EKLEME - offer()  
        kuyruk.offer("Ali");  
        kuyruk.offer("Ayşe");  
        kuyruk.offer("Mehmet");  
        System.out.println(kuyruk);  
        // [Ali, Ayşe, Mehmet]  
  
        // İLK KİŞİYİ GÖRME (silmeden) - peek()  
        String sıra = kuyruk.peek();  
        System.out.println("Sırada: " + sıra);  
        // Ali (kuyruk değişmedi)  
  
        // İŞLEM YAPMA (ilk kişiyi al ve sil) - poll()  
        String islenen = kuyruk.poll();  
        System.out.println("İşlendi: " + islenen);  
        System.out.println("Kalan: " + kuyruk);  
        // Ali işlendi, [Ayşe, Mehmet] kaldı  
    }  
}
```

5. COLLECTIONS SINIFI

5.1 Collections Yardımcı Metodları

Kod Örneği:

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

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

        ArrayList<Integer> sayilar = new ArrayList<>();
        sayilar.add(5);
        sayilar.add(2);
        sayilar.add(8);
        sayilar.add(1);

        System.out.println("Orijinal: " + sayilar);

        // 1. SIRALAMA - sort()
        Collections.sort(sayilar);
        System.out.println("Sıralı: " + sayilar);
        // [1, 2, 5, 8]

        // 2. TERS ÇEVİRME - reverse()
        Collections.reverse(sayilar);
        System.out.println("Ters: " + sayilar);
        // [8, 5, 2, 1]

        // 3. KARIŞTIRMA - shuffle()
        Collections.shuffle(sayilar);
        System.out.println("Karışık: " + sayilar);
        // [2, 8, 1, 5] (rastgele)

        // 4. MAKSİMUM - max()
        int max = Collections.max(sayilar);
        System.out.println("Max: " + max);

        // 5. MİNİMUM - min()
        int min = Collections.min(sayilar);
        System.out.println("Min: " + min);

        // 6. FREKANS - frequency()
        ArrayList<String> isimler = new ArrayList<>();
        isimler.add("Ali");
        isimler.add("Ayşe");
        isimler.add("Ali");

        int aliSayisi = Collections.frequency(isimler, "Ali");
        System.out.println("Ali kaç kere: " + aliSayisi);
```

```
    // 2  
    }  
}
```

6. KARŞILAŞTIRMA VE SEÇİM REHBERİ

6.1 Hangi Collection Ne Zaman Kullanılır?

ARRAYLIST

- Sık sık index ile erişim yapıyorsanız
- Tekrar eden elemanlara ihtiyacınız varsa

Örnek: Öğrenci listesi, not listesi

LINKEDLIST

- Çok sık ekleme/silme yapıyorsanız
- Kuyruk oluşturacaksınız

Örnek: Banka kuyruğu, işlem sırası

HASHSET

- Benzersiz elemanlar tutacaksınız
- Hızlı arama yapacaksınız

Örnek: Benzersiz kelimeler, kullanıcı ID'leri

HASHMAP

- Anahtar-değer çiftleri tutacaksınız
- Key ile hızlı arama yapacaksınız

Örnek: Telefon rehberi, envanter sistemi

SONUÇ

Bu dokümanda Java Collections Framework'ün en yaygın kullanılan dört yapısını detaylı olarak inceledik:

- **ArrayList:** Dinamik dizi, index ile hızlı erişim
- **HashMap:** Anahtar-değer çiftleri, hızlı arama
- **HashSet:** Benzersiz elemanlar, tekrar yok
- **LinkedList:** Bağlı liste, hızlı ekleme/silme

Her yapının kendine özgü güçlü ve zayıf yönleri vardır. Doğru collection seçimi, programınızın performansını önemli ölçüde etkiler.