



Bölüm 5: Diziler

JAVA ile Nesne Yönelimli Programlama



Diziler

- **Aynı türdeki** verileri düzenli bir şekilde saklar.
- Aynı adla tanımlanan **bir veya daha fazla** veri saklayabilen değişkenlerdir.
- **İndeksleme** ile veriye hızlı erişim sağlar.



Dizi Kullanımı

- Dizi elemanlarına **indeks numaraları** ile erişilir.
- İlk elemanın indeksi **0**'dır.
- Elemanlara erişmek için dizinin adı ve indeksi kullanılır.

- **Örnek:**

`Dizi = [10, 20, 30, 40, 50]`

`İlkEleman = Dizi[0] # İlk eleman 10'dur.`



Avantajları

- Verileri düzenli ve ardışık bir şekilde saklar.
- Elemanlara hızlı erişim sağlar (indeksleme).
- Verileri gruplandırarak düzenlemeye olanak sağlar.



Dezavantajları

- Sabit boyutlu tanımlanır, boyutu sonradan değiştirilemez.
- Dizi içerisinde yer kalmadığında, ekleme veya çıkarma işlemleri ek maliyet gerektirir (dizi elemanlarını kaydırma gibi).



Türleri

- **Tek boyutlu** : Elemanlar yalnızca bir satırda saklanır.
- **İki boyutlu** : Elemanlar satır ve sütunlarda saklanır.
- **Çok boyutlu** : Daha karmaşık düzeyde veri saklama/erişme işlemlerini destekler.



Kullanım Alanları

- Programlama dillerinde yaygın olarak kullanılır.
- Verileri düzenlemek ve işlemek için idealdir.
- Matrisler, resimler ve ses verileri gibi yapıları temsil etmek için kullanılır.



Diziler

- Görsel, beş elemanlı tek boyutlu bir diziyi temsil etmektedir.
- Her bir eleman, bir hücre içerisinde bulunmaktadır.
- Dizinin elemanları bir arada ve düzenli bir şekilde görülmektedir.

indis	0	1	2	3	4
eleman	13	-1	4	2	7



Dizi Tanımlama

- `VeriTipi[] diziAdi = new VeriTipi[diziBoyutu];`
- **VeriTipi:** Dizide saklanacak verilerin türünü belirtir.
- **diziAdi:** Dizinin adını temsil eder.
- **diziBoyutu:** Dizinin kaç eleman içereceğini belirtir.
- **Örnek:**
`int[] sayilar = new int[5];`



Dizi Elemanlarına Erişim

- Dizi elemanlarına **indeks** ile erişilir.
- İndeks **0**'dan başlar.

- **Örnek:**

```
int ilkEleman = sayilar[0]; // İlk elemanı alır
```

```
int ikinciEleman = sayilar[1]; // İkinci elemanı alır
```



Dizi İklendirme

- Dizi tanımlanırken, elemanlarına başlangıç değeri atanabilir.
- İklendirme sırasında dizi boyutunu belirtmeye gerek yoktur.

- **Örnek:**

```
int[] sayılar = {10, 20, 30, 40, 50};
```



Dizi Boyutunu Almak

- `length` özelliği ile dizinin boyutu alınabilir.
- Bu özellik dizinin kaç eleman içerdiğini verir.

- **Örnek:**

```
int diziBoyutu = sayilar.length; // Dizi boyutu: 5
```



Boyut Belirtilmediğinde

- Dizi boyutu belirtilmediğinde, başlatıcılar boyutu belirler.

- Örnek:

```
int[] n = {1, 2, 3, 4, 5}; // 5 değer olduğundan n 5 elemanlı  
                             bir dizi olur.
```



Örnek Kod Parçası

```
public class DiziOrnegi {  
    public static void main(String[] args) {  
        // Bir diziyi tanımlama  
        int[] sayilar = new int[5]; // 5 elemanlı bir tamsayı dizisi  
  
        // Dizinin ilk elemanlarına değer atama  
        sayilar[0] = 10;  
        sayilar[1] = 20;  
        sayilar[2] = 30;  
        sayilar[3] = 40;  
        sayilar[4] = 50;  
    }  
}
```



Örnek Kod Parçası

```
// Dizinin elemanlarına erişme
System.out.println("birinci eleman: " + sayilar[0]);
System.out.println("ucuncu eleman: " + sayilar[2]);

// Dizinin tüm elemanlarını döngü içerisinde yazdırma
System.out.print("Elemanlar: ");
for (int i = 0; i < sayilar.length; i++) {
    System.out.print(sayilar[i] + " ");
}
}
```



OrtalamaHesaplama

```
public class OrtalamaHesaplama {  
    public static void main(String[] args) {  
        // Kullanıcıdan dizi boyutunu al  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("boyutu girin: ");  
        int boyut = scanner.nextInt();  
  
        // Kullanıcıdan elemanları al ve dizi oluştur  
        double[] dizi = new double[boyut];  
        for (int i = 0; i < boyut; i++) {  
            System.out.print((i + 1) + ". elemanı girin: ");  
            dizi[i] = scanner.nextDouble();  
        }  
    }  
}
```




OrtalamaHesaplama

```
double toplam = 0;
for (int i = 0; i < boyut; i++) {
    toplam += dizi[i]; // Dizi elemanlarini toplama
}
double ortalama = toplam / boyut; // Ortalama hesaplama
System.out.println("Elemanlar: ");
for (int i = 0; i < boyut; i++) {
    System.out.print(dizi[i] + " ");
}
System.out.println("Ortalama:" + ortalama); // Sonucu yazdir
}
```



OrtancaDegerHesaplama

```
public class OrtancaDegerHesaplama {  
    public static void main(String[] args) {  
        // Kullanıcıdan dizi boyutunu al  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("boyutu girin: ");  
        int boyut = scanner.nextInt();  
        // Kullanıcıdan elemanları al ve dizi oluştur  
        double[] dizi = new double[boyut];  
        for (int i = 0; i < boyut; i++) {  
            System.out.print((i + 1) + ". elemani girin: ");  
            dizi[i] = scanner.nextDouble();  
        }  
        // Diziyi sırala  
        Arrays.sort(dizi);  
    }  
}
```



OrtancaDegerHesaplama

```
if (boyut % 2 == 0) { // Çift boyutlu dizi için ortanca hesabı
    int orta1 = boyut / 2 - 1;
    int orta2 = boyut / 2;
    ortanca = (dizi[orta1] + dizi[orta2]) / 2; // Ortanca hesapla
} else { // Tek boyutlu dizi için ortanca hesabı
    ortanca = dizi[boyut / 2];
} // Sonucu yazdır
System.out.println("Elemanlar (siralı): ");
for (int i = 0; i < boyut; i++) {
    System.out.print(dizi[i] + " ");
}
System.out.println("Ortanca Değer: " + ortanca);
}
```



ModHesaplama

```
public class ModHesaplama {  
    public static void main(String[] args) {  
        // Kullanıcıdan dizi boyutunu al  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("boyutu girin: ");  
        int boyut = scanner.nextInt();  
        // Kullanıcıdan elemanları al ve dizi oluştur  
        int[] dizi = new int[boyut];  
        for (int i = 0; i < boyut; i++) {  
            System.out.print((i + 1) + ". elemani girin: ");  
            dizi[i] = scanner.nextInt();  
        }  
        // Diziyi sırala  
        Arrays.sort(dizi);  
    }  
}
```



ModHesaplama

```
// Mod değeri ve tekrar sayısını bulma
int enCokTekrarEden = dizi[0];    int enCokTekrarSayisi = 1;
int mevcutTekrarEden = dizi[0];  int mevcutTekrarSayisi = 1;

for (int i = 1; i < boyut; i++) {
    if (dizi[i] == dizi[i - 1]) {
        mevcutTekrarSayisi++;
    } else {
        mevcutTekrarSayisi = 1;
        mevcutTekrarEden = dizi[i];
    }
    if (mevcutTekrarSayisi > enCokTekrarSayisi) {
        enCokTekrarSayisi = mevcutTekrarSayisi;
        enCokTekrarEden = mevcutTekrarEden;
    }
}
```



ModHesaplama

```
// Sonucu yazdır
System.out.println("Elemanlar sirali: " + Arrays.toString(dizi));
System.out.print("Mod : " + enCokTekrarEden);
System.out.println("(Tekrar Sayisi: " + enCokTekrarSayisi + ")");
    }
}
```



Çok Boyutlu Diziler

- Dizi içinde birden fazla boyutta veri saklamak için kullanılır.
- Java iki (matrisler) ve daha fazla boyutlu dizileri destekler.



İki Boyutlu Diziler (Matrisler)

- Satır ve sütunlarla tanımlanan tablo benzeri yapılardır.
- Örneğin; bir satranç tahtası veya bir resim gibi iki boyutlu verileri temsil etmek için kullanışlıdır.

- **Örnek:**

```
int[][] matris = new int[3][3]; // 3x3 boyutunda bir matris
```




İki Boyutlu Dizilerin İlkendirilmesi

- İki boyutlu dizilere başlangıç değerleri atanabilir.
- İlk değer atama sırasında matrisin boyutunu belirtmeye gerek yoktur.

- **Örnek:**

```
int[][] matris = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```



CokBoyutluDiziOrnegi

```
public class CokBoyutluDiziOrnegi {  
    public static void main(String[] args) {  
        // İki boyutlu bir dizi (matris) tanımlama ve başlatma  
        int[][] matris = new int[3][3]; // 3x3 boyutunda bir matris tanımlandı  
  
        // Matrise elemanlarına ilk değerlerini atama  
        for (int satir = 0; satir < 3; satir++) {  
            for (int sutun = 0; sutun < 3; sutun++) {  
                matris[satir][sutun] = satir * 3 + sutun + 1;  
            }  
        }  
    }  
}
```



CokBoyutluDiziOrnegi

```
// Matrisi ekrana yazdırma
System.out.println("Matrisin Icerigi:");
for (int satir = 0; satir < 3; satir++) {
    for (int sutun = 0; sutun < 3; sutun++) {
        System.out.print(matris[satir][sutun] + " ");
    }
    System.out.println(); // Yeni satıra geçme
}
}
```



Üç Boyutlu ve Daha Fazla Boyutlu Diziler

- Java'da üç boyutlu diziler ve daha fazla boyutlu diziler de kullanılabilir.
- Daha fazla boyut, daha karmaşık veri yapılarına izin verir.

- **Örnek:**

```
// 3x3x3 boyutunda üç boyutlu dizi
```

```
int[][][] ucBoyutluDizi = new int[3][3][3];
```



UcBoyutluDiziOrnegi

```
public class UcBoyutluDiziOrnegi {  
    public static void main(String[] args) {  
        // Üç boyutlu bir dizi tanımlama  
        int[][][] ucBoyutluDizi = new int[3][3][3];  
  
        // Üç boyutlu dizinin elemanlarına ilk değer atama  
        for (int x = 0; x < 3; x++) {  
            for (int y = 0; y < 3; y++) {  
                for (int z = 0; z < 3; z++) {  
                    ucBoyutluDizi[x][y][z] = x * 9 + y * 3 + z + 1;  
                }  
            }  
        }  
    }  
}
```



UcBoyutluDiziOrnegi

```
// Üç boyutlu diziyi ekrana yazdırma
System.out.println("Uc Boyutlu Dizinin Icerigi:");
for (int x = 0; x < 3; x++) {
    for (int y = 0; y < 3; y++) {
        for (int z = 0; z < 3; z++) {
            System.out.print(ucBoyutluDizi[x][y][z] + " ");
        }
        System.out.println(); // Yeni satıra geçme (y eksenini)
    }
    System.out.println(); // Yeni satıra geçme (x eksenini)
}
}
```



Doğrusal Arama

- **Örnek:** Bir dizinin içerisinde belirli bir elemanı arama.
- Sıradan bir (lineer) arama algoritması.

```
public boolean lineerArama(int[] dizi, int hedef) {  
    for (int eleman : dizi) {  
        if (eleman == hedef) {  
            return true;  
        }  
    }  
    return false;  
}
```



Doğrusal Arama

- En kötü durumda tüm dizi gezilir.
- Algoritmanın zaman karmaşıklığı $O(n)$ 'dir.
- (n dizinin boyutunu temsil eder)

- **İyi Durum:** Hedef erken bulunur, arama hızlı biter.
- **Kötü Durum:** Hedef dizinin sonunda bulunur veya hiç bulunmaz, arama uzun sürer.



Eşleştirme

- **Örnek:** İki dizi arasında ortak elemanları eşleştirme.
- İç içe iki döngü kullanarak eş eleman arama algoritması.

```
public void eslestir(int[] dizi1, int[] dizi2) {  
    for (int eleman1 : dizi1) {  
        for (int eleman2 : dizi2) {  
            if (eleman1 == eleman2) {  
                System.out.println("Eslesen Eleman: " + eleman1);  
            }  
        }  
    }  
}
```



Eşleştirme

- İç içe döngüler, her elemanı her iki dizide karşılaştırır.
- Algoritmanın zaman karmaşıklığı $O(n^2)$ 'dir.
- (n dizinin boyutunu temsil eder).
- $O(n^2)$ algoritmalar büyük veri kümeleri için verimsizdir.



Durağan (static) Diziler

- Yığımda (stack) oluşturulur ve bellekte otomatik *yaşam süresine* sahiptir.
- Belleği yönetmeye gerek yoktur.
- Bu diziler işlevleri sona erdiğinde yok olur.
- Derleme zamanında sabit bir boyuta sahiptirler.

```
String[] plaka = {"", "Adana", "Adıyaman", "Afyon"};  
System.out.println(plaka[1]);
```



Dinamik Diziler

- Dinamik bellek yaşam süresine sahiptir.
- Yığın yerine, heap (serbest bellek) üzerinde depolanır.
- Çalışma zamanında istenilen boyuta sahip olabilirler.
- Ancak, bellek tahsisi ve serbest bırakma işlemleri elle yapılır.

```
int[] dinamikDizi = new int[10]; // Bellek tahsis edilir  
// ...  
dinamikDizi = null; // Bellek serbest bırakılır
```



İndis Sınır Dışı Hatası (IndexOutOfBoundsException)

- Dizinin belirtilen sınırlarının dışına çıkıldığında ortaya çıkar.
- Programın çökmesine veya cevap vermemesine neden olabilir.

```
// Dizi uzunluğunu kontrol etme
if (index >= 0 && index < dizi.length) {
    int deger = dizi[index];
    // indis dışı hatayı önler
} else {
    // indis dışı hatayı ele alma veya hata mesajı gösterme
}
```



Dizi Elemanına Erişme Zaman Karmaşıklığı

- $O(1)$, sabit zaman karmaşıklığıdır, işlem süresi girdi boyundan bağımsızdır.
- Bir dizinin belirli bir elemanına erişmek sabit zaman alır, çünkü dizi elemanlarının bellek adresi hesaplanabilir.

```
int[] dizi = {10, 20, 30, 40, 50};  
// Bir dizinin belirli bir elemanına erişme  
int eleman = dizi[2]; // 30  
// Bu işlem  $O(1)$  karmaşıklığına sahiptir.
```



Dizi Elemanı Arama Zaman Karmaşıklığı

- $O(n)$, lineer zaman karmaşıklığıdır, girdi boyutu ile doğru orantılıdır.
- Her bir elemanı kontrol etmek gerekir.

```
int[] dizi = {10, 20, 30, 40, 50};  
// Bir dizide belirli bir elemanı arama  
int aranan = 30;  
boolean bulundu = false;  
  
for (int eleman : dizi) {  
    if (eleman == aranan) {  
        bulundu = true;  
        break;  
    }  
}  
// Bu işlem  $O(n)$  karmaşıklığına sahiptir.
```



Seçimli Sıralama (Selection Sort)

- Her adımda en küçük eleman bulunup dizinin başına yerleştirilir.

```
int[] dizi = {64, 25, 12, 22, 11};
```

```
for (int i = 0; i < dizi.length - 1; i++) {  
    int minIndeks = i;  
    for (int j = i + 1; j < dizi.length; j++) {  
        if (dizi[j] < dizi[minIndeks]) {  
            minIndeks = j;  
        }  
    }  
    // En küçük eleman bulundu, yer değiştir  
    yerDegistir(dizi[minIndeks], dizi[i])  
}
```




Matris Transpozu

- Orijinal matrisin satırlarını sütunlara, sütunları satırlara dönüştürür.

```
public class MatrisTranspoz {  
    public static void main(String[] args) {  
        int[][] matris = {  
            {1, 2, 3},  
            {4, 5, 6}  
        };  
  
        // Orijinal matrisi yazdır  
        System.out.println("Orijinal Matris:");  
        yazdirMatris(matris);  
    }  
}
```



Matris Transpozu

```
int satirSayisi = matris.length;
int sutunSayisi = matris[0].length;

// Transpoz matrisi oluřtur
int[][] transpozMatris = new int[sutunSayisi][satirSayisi];

for (int i = 0; i < satirSayisi; i++) {
    for (int j = 0; j < sutunSayisi; j++) {
        transpozMatris[j][i] = matris[i][j];
    }
}
```



Matris Transpozu

```
// Transpoz matrisi yazdır
System.out.println("Transpoz Matris:");
yazdirMatris(transpozMatris);
}
// Matrisi ekrana yazdırmak için yardımcı fonksiyon
public static void yazdirMatris(int[][] matris) {
    for (int i = 0; i < matris.length; i++) {
        for (int j = 0; j < matris[i].length; j++) {
            System.out.print(matris[i][j] + " ");
        }
        System.out.println();
    }
}
```



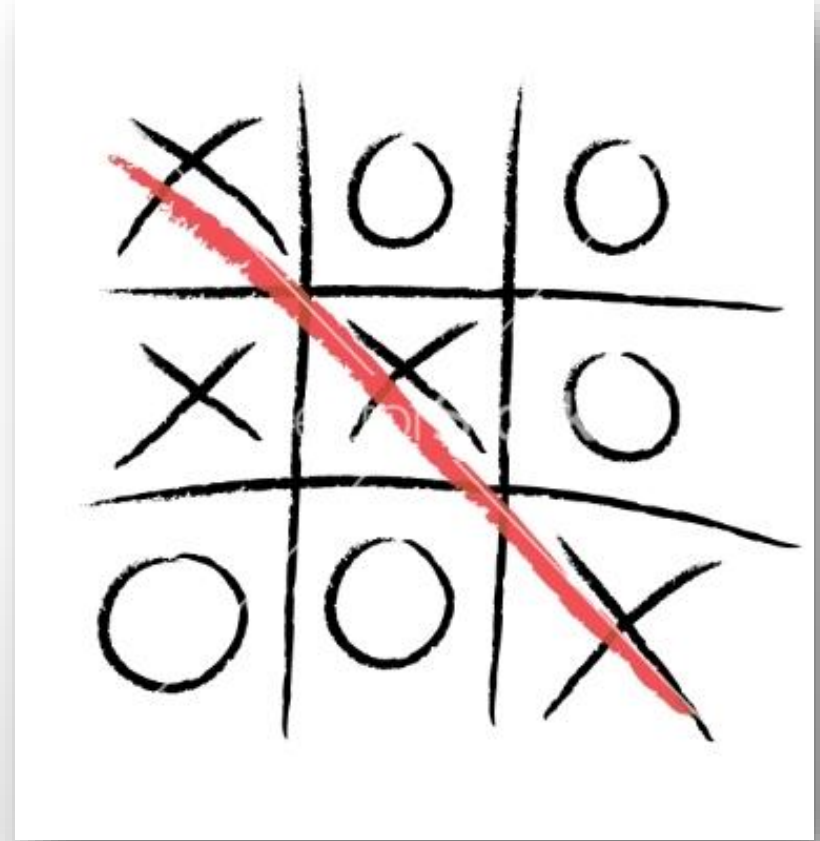
Diziyi Tersine Çevirme

```
public void tersineCevir(int[] dizi) {  
    int baslangic = 0;  
    int bitis = dizi.length - 1;  
  
    while (baslangic < bitis) {  
        // Baslangic ve bitis elemanlarini degistir  
        int gecici = dizi[baslangic];  
        dizi[baslangic] = dizi[bitis];  
        dizi[bitis] = gecici;  
        // Baslangic indeksini arttir, bitis indeksini azalt  
        baslangic++;  
        bitis--;  
    }  
}
```



Tic Tac Toe (X-Oyunu)

- 3x3'lük oyun tahtası kullanılır.
- Oyuncular sırayla hamle yapar.



- 3 tane yan yana ya da çapraz elemanlarını yerleştiren oyunu kazanır.



Tic Tac Toe (X-Oyunu)

```
public class TicTacToe {  
    public static void main(String[] args) {  
        char[][] tahta = new char[3][3]; // 3x3'lük oyun tahtası  
        char oyuncu = 'X'; // İlk oyuncu X ile başlar  
        boolean oyunDevamEdiyor = true;  
        tahtaDoldur(tahta); // Tahtayı başlangıç durumuyla doldur  
        while (oyunDevamEdiyor) {  
            tahtayiGoster(tahta);  
            hamleYap(tahta, oyuncu);  
            oyunDevamEdiyor = oyunDevamEdiyorMu(tahta, oyuncu);  
            oyuncu = (oyuncu == 'X') ? 'O' : 'X'; // Oyuncu değişimi  
        }  
        tahtayiGoster(tahta);  
    }  
}
```



Tic Tac Toe (X-Oyunu)

```
public void tahtaDoldur(char[][] tahta) {  
    for (int i = 0; i < 3; i++) {  
        for (int j = 0; j < 3; j++) {  
            tahta[i][j] = ' '  
        }  
    }  
}
```



Tic Tac Toe (X-Oyunu)

```
public void tahtayiGoster(char[][] tahta) {  
    System.out.println("-----");  
    for (int i = 0; i < 3; i++) {  
        System.out.print("/ ");  
        for (int j = 0; j < 3; j++) {  
            System.out.print(tahta[i][j] + " | ");  
        }  
        System.out.println("\n-----");  
    }  
}
```




Tic Tac Toe (X-Oyunu)

```
public void hamleYap(char[][] tahta, char oyuncu) {  
    Scanner scanner = new Scanner(System.in);  
    int satir, sutun;  
  
    do {  
        System.out.print("Sıra " + oyuncu + " oyuncusunda. Satır ve sütun seçin (1-3): ");  
        satir = scanner.nextInt() - 1;  
        sutun = scanner.nextInt() - 1;  
    } while (satir < 0 || satir > 2 || sutun < 0 || sutun > 2 || tahta[satir][sutun] != ' ');  
  
    tahta[satir][sutun] = oyuncu;  
}
```



Tic Tac Toe (X-Oyunu)

```
public boolean oyunDevamEdiyorMu(char[][] tahta, char oyuncu) {  
    // Kazanan durumlarını kontrol et  
    if ((tahta[0][0] == oyuncu && tahta[0][1] == oyuncu && tahta[0][2] == oyuncu) ||  
        (tahta[1][0] == oyuncu && tahta[1][1] == oyuncu && tahta[1][2] == oyuncu) ||  
        (tahta[2][0] == oyuncu && tahta[2][1] == oyuncu && tahta[2][2] == oyuncu) ||  
        (tahta[0][0] == oyuncu && tahta[1][0] == oyuncu && tahta[2][0] == oyuncu) ||  
        (tahta[0][1] == oyuncu && tahta[1][1] == oyuncu && tahta[2][1] == oyuncu) ||  
        (tahta[0][2] == oyuncu && tahta[1][2] == oyuncu && tahta[2][2] == oyuncu) ||  
        (tahta[0][0] == oyuncu && tahta[1][1] == oyuncu && tahta[2][2] == oyuncu) ||  
        (tahta[0][2] == oyuncu && tahta[1][1] == oyuncu && tahta[2][0] == oyuncu)) {  
        System.out.println("Tebrikler, " + oyuncu + " oyuncusu kazandı!");  
        return false;  
    }  
}
```



Tic Tac Toe (X-Oyunu)

```
// Berabere durumunu kontrol et
boolean berabere = true;
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        if (tahta[i][j] == ' ') {
            berabere = false;
            break;
        }
    }
    if (!berabere) {
        break;
    }
}
```



Tic Tac Toe (X-Oyunu)

```
    if (berabere) {  
        System.out.println("Oyun berabere bitti.");  
        return false;  
    }  
  
    return true;  
}  
}
```



SON