Bölüm 10: Diziler

Bilgisayar ve Programlamaya Giriş

Tanım · İndis · Döngü · Arama · Matris

Sercan KÜLCÜ · Giresun Üniversitesi

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü: https://sercankulcu.github.io/files/computer/slides/Bolum_10_Diziler.html

GİRİŞ

Öğrenme Çıktıları

Oluşturma

Dizi bildirmek ve ilk değer vermek.

Erişim

İndis ve length kullanmak.

İşleme

Diziyi döngülerle gezmek ve dönüştürmek.

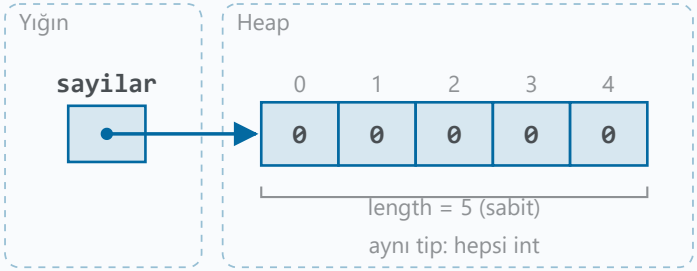
Matris

İki boyutlu dizilerle satır-sütun işlemleri yapmak.

TEMELLER

Dizi Nedir?

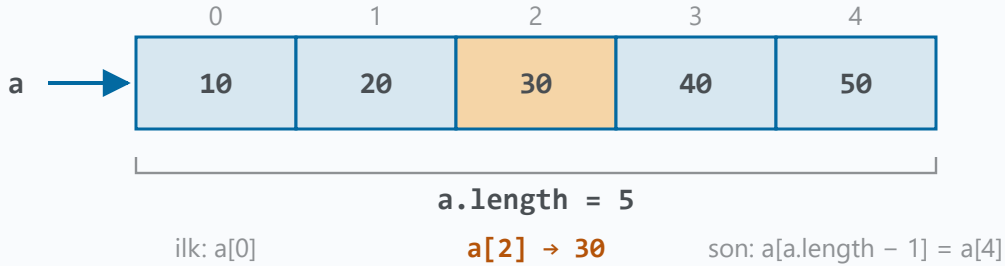
- Aynı tipte, sabit sayıda öğeyi **tek nesnede** ardışık hücrelerde saklar.
- Öğelere sıfır tabanlı **indisle** erişilir: `sayilar[0]` … `sayilar[4]`.
- Uzunluk (`length`) oluşturulduktan sonra değişmez.
- Dizi değişkeni öğeleri değil, heap'teki dizi nesnesinin **referansını** (adresini) tutar.



Değişken → dizi nesnesi (referans oku)

TEMELLER

Dizi ve İndis

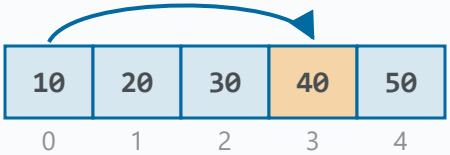


- İlk indis **0**, son indis **length − 1**'dir (5 öğe → 0…4).
- `a[i]` adresi doğrudan hesaplanır (başlangıç + i × hücre boyu); bu yüzden erişim diziyi gezmeden, **sabit sürede** (O(1)) olur.

Avantaj ve Sınır

Avantaj	Sınır
İndisle hızlı erişim	Boyut sabittir
Öğe tipi tutarlıdır	Araya ekleme maliyetlidir
Döngüyle kolay işlenir	Sınır aşımı çalışma zamanı hatasıdır
Bellekte düzenli yapı	Tek bir öğe tipi tutar

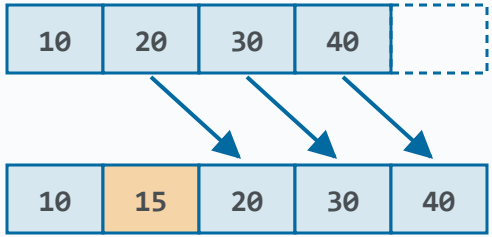
İndisle erişim: doğrudan atlama



adres = başlangıç + 3 × hücre boyu

`a[3] → 40: tek adım, O(1)`

Araya ekleme: 15 → indis 1



sağdaki 3 öğe birer kayar

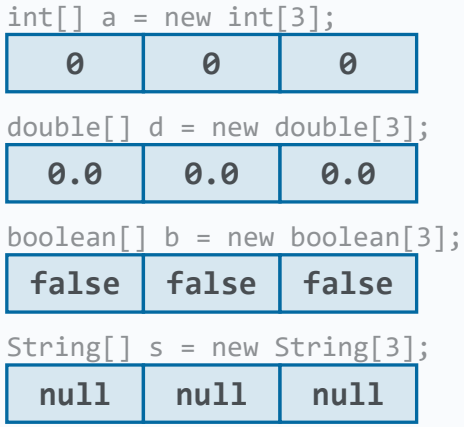
Dizi doluysa: yeni, daha büyük dizi + kopyalama (boyut sabit)

BİLDİRİM

Dizi Oluşturma

```
int[] sayilar = new int[5];
String[] adlar = {"Ada", "Can", "Ece"};
double[] olcumler;
olcumler = new double[10];
```

- `new` ile oluşturulan dizinin hücreleri tipin varsayılan değerini alır (sağda).
- İlklendirici `{…}` kullanılınca uzunluk değer sayısından çıkarılır (adlar.length = 3).



Varsayılan başlangıç değerleri

BİLDİRİM · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: Dizi Oluşturma



Bildirim: yığında yalnızca `sayilar` adlı referans değişkeni açılır; henüz bir dizi yok.

Adım 1 / 6

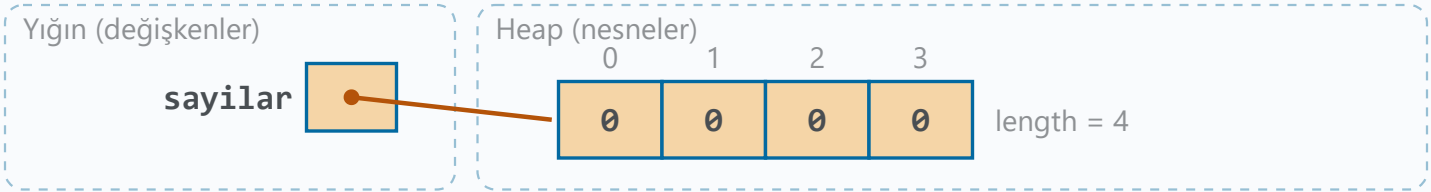
```
▶ int[] sayilar;
2 sayilar = new int[4];
3 sayilar[0] = 7;
4 sayilar[3] = 2;
5 String[] adlar = {"Ada", "Can", "Ece"};
6 System.out.println(sayilar.length + " "
7                    + adlar.length);
```

DEĞİŞKENLER

`sayilar = ? (atanmamış)`

ÇIKTI

Kod İzleme: Dizi Oluşturma



`new int[4]` heap'te 4 hücreli dizi oluşturur, hepsi **0**. Değişkene dizinin **referansı** yazılır (ok).

Adım 2 / 6

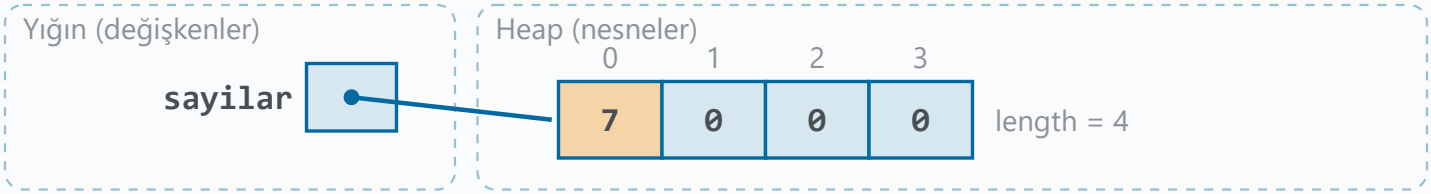
```
1 int[] sayilar;
▶ sayilar = new int[4];
3 sayilar[0] = 7;
4 sayilar[3] = 2;
5 String[] adlar = {"Ada", "Can", "Ece"};
6 System.out.println(sayilar.length + " "
7                    + adlar.length);
```

DEĞİŞKENLER

`sayilar = [0, 0, 0, 0]`

ÇIKTI

Kod İzleme: Dizi Oluşturma



`sayilar[0] = 7` : ok izlenir, **0 numaralı hücreye** 7 yazılır.

Adım 3 / 6

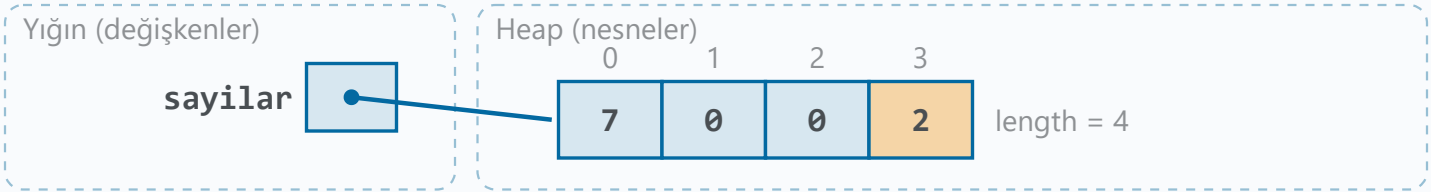
```
1 int[] sayilar;
2 sayilar = new int[4];
▶ sayilar[0] = 7;
4 sayilar[3] = 2;
5 String[] adlar = {"Ada", "Can", "Ece"};
6 System.out.println(sayilar.length + " "
7                    + adlar.length);
```

DEĞİŞKENLER

`sayilar = [7, 0, 0, 0]`

ÇIKTI

Kod İzleme: Dizi Oluşturma



`sayilar[3] = 2` : son hücre (indis **length − 1 = 3**) güncellenir.

Adım 4 / 6

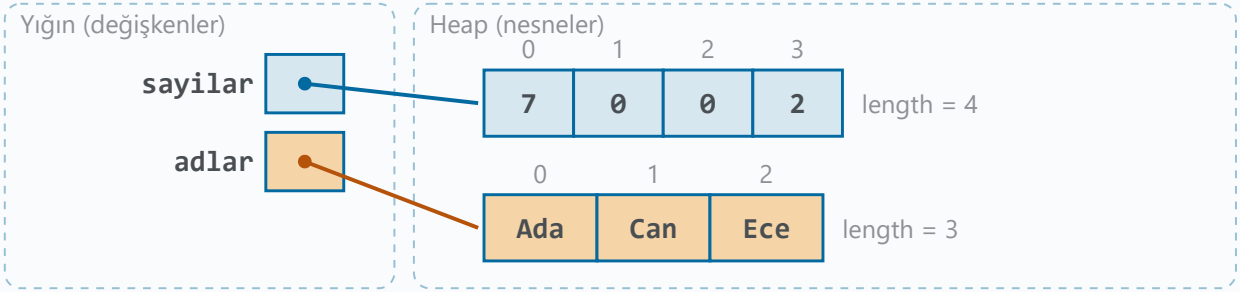
```
1 int[] sayilar;
2 sayilar = new int[4];
3 sayilar[0] = 7;
▶ sayilar[3] = 2;
5 String[] adlar = {"Ada", "Can", "Ece"};
6 System.out.println(sayilar.length + " "
7                    + adlar.length);
```

DEĞİŞKENLER

`sayilar = [7, 0, 0, 2]`

ÇIKTI

Kod İzleme: Dizi Oluşturma



İlklendirici `{…}` : 3 değer var → uzunluk **3** olan dizi kurulur ve `adlar` onu gösterir.

Adım 5 / 6

```
1 int[] sayilar;
2 sayilar = new int[4];
3 sayilar[0] = 7;
4 sayilar[3] = 2;
▶ String[] adlar = {"Ada", "Can", "Ece"};
6 System.out.println(sayilar.length + " "
7                    + adlar.length);
```

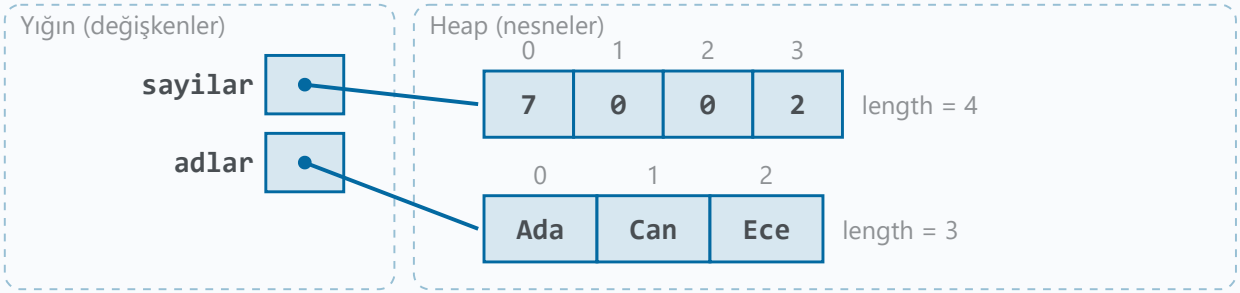
DEĞİŞKENLER

`sayilar = [7, 0, 0, 2]`

`adlar = [Ada, Can, Ece]`

ÇIKTI

Kod İzleme: Dizi Oluşturma



`length` bir alandır (parantez yok): **4** ve **3** yazılır.

Adım 6 / 6

```
1 int[] sayilar;
2 sayilar = new int[4];
3 sayilar[0] = 7;
4 sayilar[3] = 2;
5 String[] adlar = {"Ada", "Can", "Ece"};
▶ System.out.println(sayilar.length + " "
▶                    + adlar.length);
```

DEĞİŞKENLER

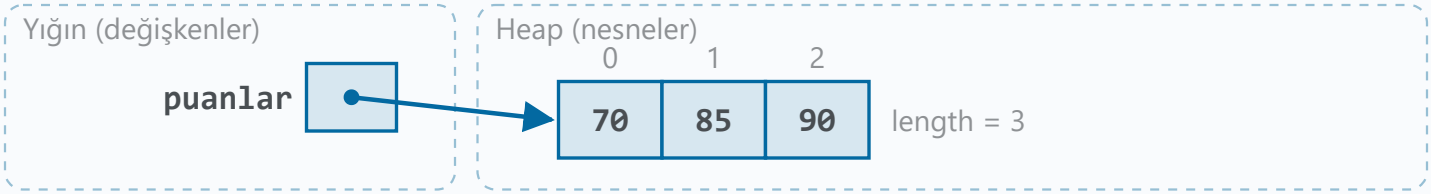
`sayilar = [7, 0, 0, 2]`

`adlar = [Ada, Can, Ece]`

ÇIKTI

```
4 3
```

Öğe Okuma ve Yazma



Üç öğeli dizi oluşur; indisler **0, 1, 2**.

Adım 1 / 5

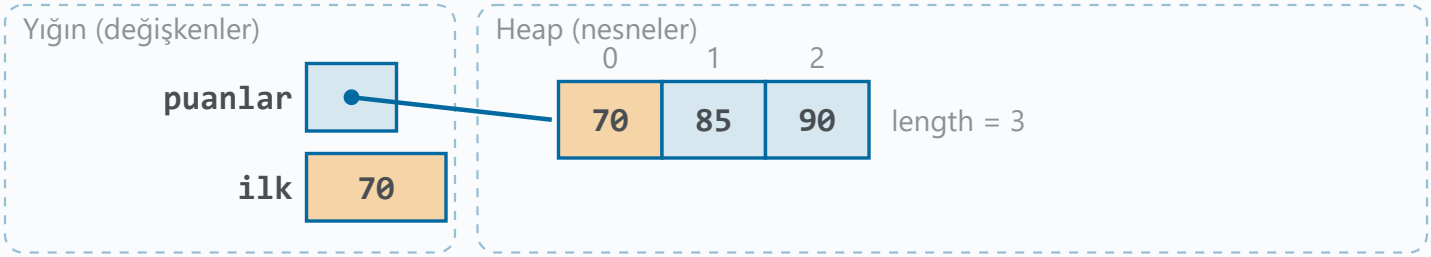
```
▶ int[] puanlar = {70, 85, 90};
2 int ilk = puanlar[0];
3 puanlar[1] = 88;
4 int son = puanlar[puanlar.length - 1];
5 System.out.println(ilk + " " + son);
```

DEĞİŞKENLER

`puanlar = [70, 85, 90]`

ÇIKTI

Öğe Okuma ve Yazma



Okuma: `puanlar[0]` ilk hücredeki değeri verir → `ilk = 70`.

Adım 2 / 5

```
1 int[] puanlar = {70, 85, 90};
▶ int ilk = puanlar[0];
3 puanlar[1] = 88;
4 int son = puanlar[puanlar.length - 1];
5 System.out.println(ilk + " " + son);
```

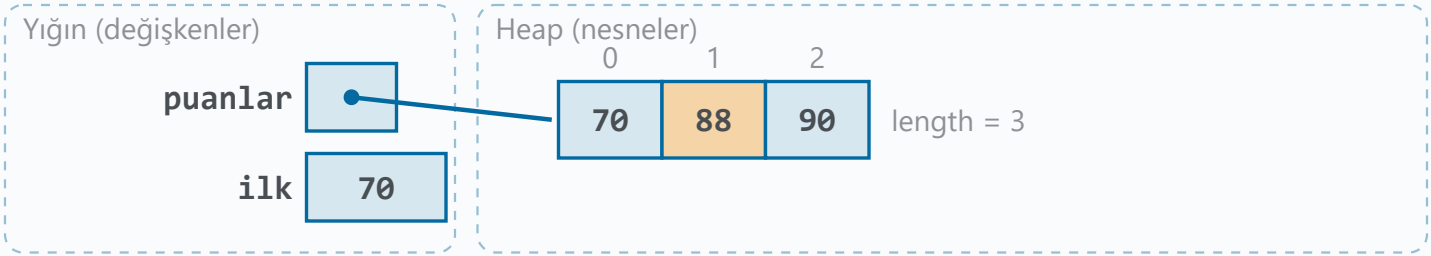
DEĞİŞKENLER

`puanlar = [70, 85, 90]`

`ilk = 70`

ÇIKTI

Öğe Okuma ve Yazma



Yazma: atamanın solundaki `puanlar[1]` hücrenin kendisidir; 85 silinir, **88** yazılır.

Adım 3 / 5

```
1 int[] puanlar = {70, 85, 90};
2 int ilk = puanlar[0];
▶ puanlar[1] = 88;
4 int son = puanlar[puanlar.length - 1];
5 System.out.println(ilk + " " + son);
```

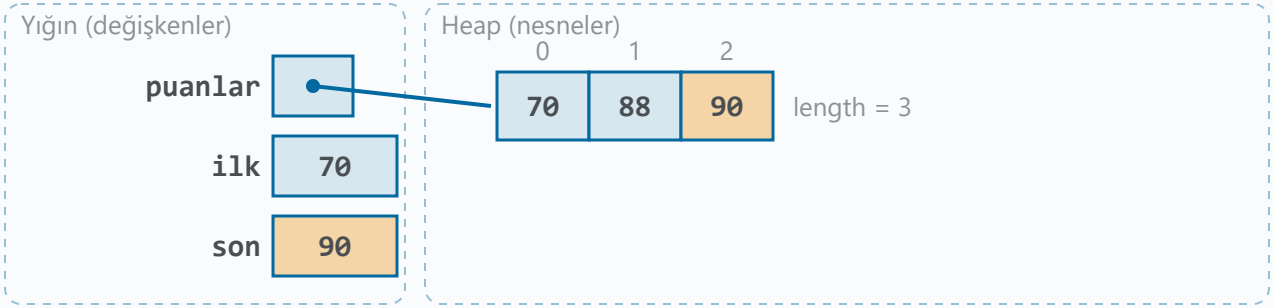
DEĞİŞKENLER

`puanlar = [70, 88, 90]`

`ilk = 70`

ÇIKTI

Öğe Okuma ve Yazma



`puanlar.length - 1` = 3 − 1 = **2** → son öğe **90**. Dizi boyu değişse de kod doğru kalır.

Adım 4 / 5

```
1 int[] puanlar = {70, 85, 90};
2 int ilk = puanlar[0];
3 puanlar[1] = 88;
▶ int son = puanlar[puanlar.length - 1];
5 System.out.println(ilk + " " + son);
```

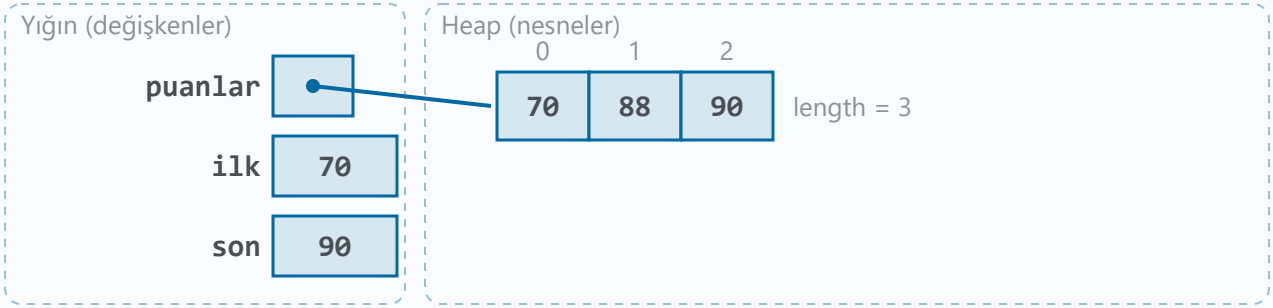
DEĞİŞKENLER

`puanlar = [70, 88, 90]`

`ilk = 70` `son = 90`

ÇIKTI

Öğe Okuma ve Yazma



İki değer aralarında boşlukla yazdırılır.

Adım 5 / 5

```
1 int[] puanlar = {70, 85, 90};
2 int ilk = puanlar[0];
3 puanlar[1] = 88;
4 int son = puanlar[puanlar.length - 1];
▶ System.out.println(ilk + " " + son);
```

DEĞİŞKENLER

`puanlar = [70, 88, 90]`

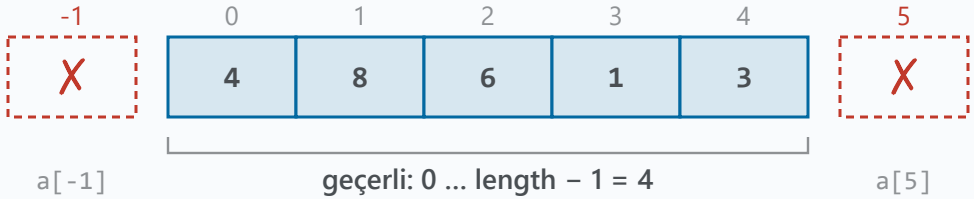
`ilk = 70` `son = 90`

ÇIKTI

```
70 90
```

ERİŞİM

Sınır Dışı Hatası



a.length = 5 ama a[5] YOK → ArrayIndexOutOfBoundsException

- Geçerli indisler **0** ile **length − 1** arasındadır; negatif indis ve `length` 'in kendisi geçersizdir.
- Derleyici bunu yakalamaz; program **çalışırken** `ArrayIndexOutOfBoundsException` ile durur.

Döngü koşulu `i < dizi.length` olmalıdır; `<=` bir tur fazla döner.

Kod İzleme: Bir Tur Fazla (<=)



Uzunluğu **3** olan dizi: geçerli indisler 0, 1, 2.

Adım 1 / 9

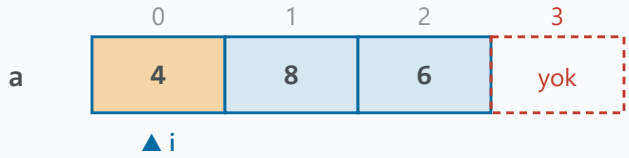
```
▶ int[] a = {4, 8, 6};
2 for (int i = 0; i <= a.length; i++) {
3     System.out.println(a[i]);
4 }
```

DEĞİŞKENLER

a = [4, 8, 6]

ÇIKTI

Kod İzleme: Bir Tur Fazla (<=)



0 <= 3 → **true**, gövdeye girilir.

Adım 2 / 9

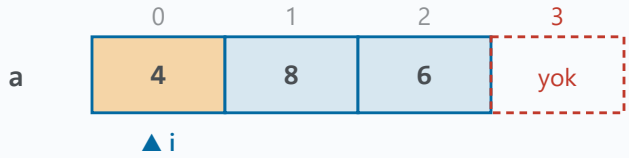
```
1 int[] a = {4, 8, 6};
▶ for (int i = 0; i <= a.length; i++) {
3     System.out.println(a[i]);
4 }
```

DEĞİŞKENLER

a = [4, 8, 6] i = 0

ÇIKTI

Kod İzleme: Bir Tur Fazla (<=)



`a[0]` = 4 yazdırılır.

Adım 3 / 9

```
1 int[] a = {4, 8, 6};
2 for (int i = 0; i <= a.length; i++) {
▶     System.out.println(a[i]);
4 }
```

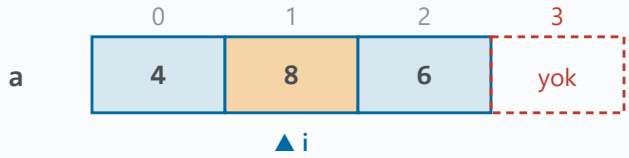
DEĞİŞKENLER

`a = [4, 8, 6]` `i = 0`

ÇIKTI

```
4
```

Kod İzleme: Bir Tur Fazla (<=)



`i++` → **1 <= 3** → **true**, gövdeye girilir.

Adım 4 / 9

```
1 int[] a = {4, 8, 6};
▶ for (int i = 0; i <= a.length; i++) {
3     System.out.println(a[i]);
4 }
```

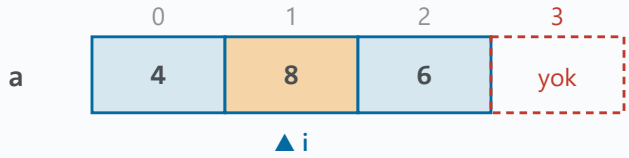
DEĞİŞKENLER

`a = [4, 8, 6]` `i = 1`

ÇIKTI

```
4
```

Kod İzleme: Bir Tur Fazla (<=)



`a[1]` = 8 yazdırılır.

Adım 5 / 9

```
1 int[] a = {4, 8, 6};
2 for (int i = 0; i <= a.length; i++) {
▶     System.out.println(a[i]);
4 }
```

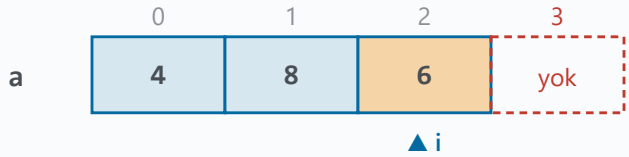
DEĞİŞKENLER

`a = [4, 8, 6]` `i = 1`

ÇIKTI

```
4
8
```

Kod İzleme: Bir Tur Fazla (<=)



`i++` → **2 <= 3** → **true**, gövdeye girilir.

Adım 6 / 9

```
1 int[] a = {4, 8, 6};
▶ for (int i = 0; i <= a.length; i++) {
3     System.out.println(a[i]);
4 }
```

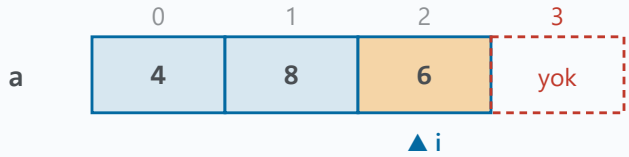
DEĞİŞKENLER

`a = [4, 8, 6]` `i = 2`

ÇIKTI

```
4
8
```

Kod İzleme: Bir Tur Fazla (<=)



`a[2]` = 6 yazdırılır.

Adım 7 / 9

```
1 int[] a = {4, 8, 6};
2 for (int i = 0; i <= a.length; i++) {
▶     System.out.println(a[i]);
4 }
```

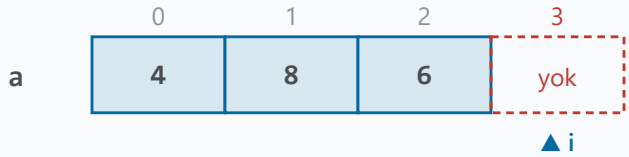
DEĞİŞKENLER

`a = [4, 8, 6]` `i = 2`

ÇIKTI

```
4
8
6
```

Kod İzleme: Bir Tur Fazla (<=)



`i++` → **3 <= 3** → **true** — dikkat: 3 geçersiz indis ama koşul yine de doğru!

Adım 8 / 9

```
1 int[] a = {4, 8, 6};
▶ for (int i = 0; i <= a.length; i++) {
3     System.out.println(a[i]);
4 }
```

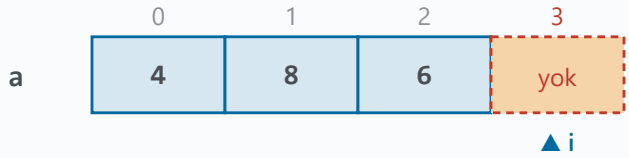
DEĞİŞKENLER

`a = [4, 8, 6]` `i = 3`

ÇIKTI

```
4
8
6
```

Kod İzleme: Bir Tur Fazla (<=)



`a[3]` : böyle bir hücre yok → program **hata ile durur**. Çözüm: koşulu `i < a.length` yapmak.

Adım 9 / 9

```
1 int[] a = {4, 8, 6};
2 for (int i = 0; i <= a.length; i++) {
▶     System.out.println(a[i]);
4 }
```

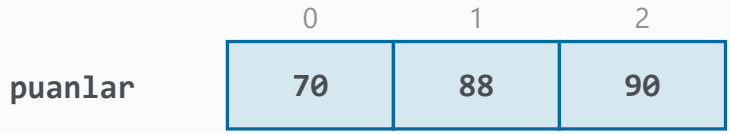
DEĞİŞKENLER

`a = [4, 8, 6]` `i = 3`

ÇIKTI

```
4
8
6
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsExc
eption: Index 3 out of bounds for
length 3
```

Klasik `for` ile Gezinme



Dizi hazır; `i` indis olarak 0'dan başlayacak.

Adım 1 / 8

```
▶ int[] puanlar = {70, 88, 90};
2 for (int i = 0; i < puanlar.length; i++) {
3   System.out.printf("%d: %d%n", i, puanlar[i]);
4 }
```

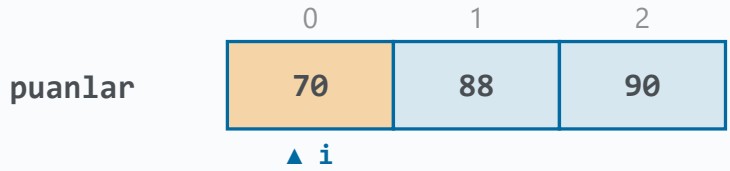
DEĞİŞKENLER

`puanlar = [70, 88, 90]`

ÇIKTI

İndis gerektiğinde (konum yazdırmak, komşu öğeye bakmak, hücreye yazmak) klasik `for` kullanılır.

Klasik `for` ile Gezinme



`int i = 0` ; **0 < 3 → true**

Adım 2 / 8

```
1 int[] puanlar = {70, 88, 90};
▶ for (int i = 0; i < puanlar.length; i++) {
3   System.out.printf("%d: %d%n", i, puanlar[i]);
4 }
```

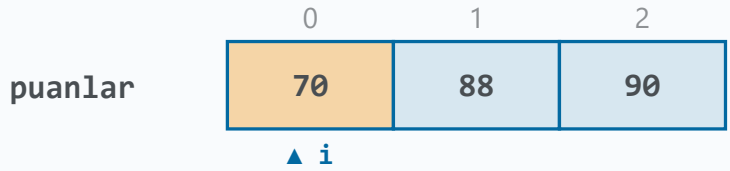
DEĞİŞKENLER

`puanlar = [70, 88, 90]` `i = 0`

ÇIKTI

İndis gerektiğinde (konum yazdırmak, komşu öğeye bakmak, hücreye yazmak) klasik `for` kullanılır.

Klasik `for` ile Gezinme



`puanlar[0]` = 70; `%d: %d` kalıbı **0: 70** üretir.

Adım 3 / 8

```
1 int[] puanlar = {70, 88, 90};
2 for (int i = 0; i < puanlar.length; i++) {
▶   System.out.printf("%d: %d%n", i, puanlar[i]);
4 }
```

DEĞİŞKENLER

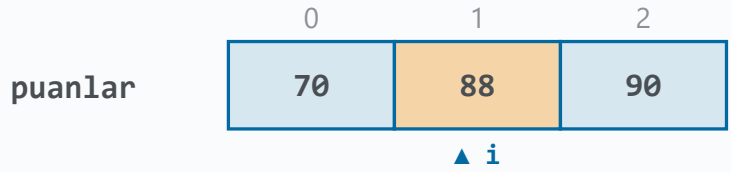
`puanlar = [70, 88, 90]` `i = 0`

ÇIKTI

```
0: 70
```

İndis gerektiğinde (konum yazdırmak, komşu öğeye bakmak, hücreye yazmak) klasik `for` kullanılır.

Klasik `for` ile Gezinme



`i++` → **1 < 3 → true**

Adım 4 / 8

```
1 int[] puanlar = {70, 88, 90};
▶ for (int i = 0; i < puanlar.length; i++) {
3   System.out.printf("%d: %d%n", i, puanlar[i]);
4 }
```

DEĞİŞKENLER

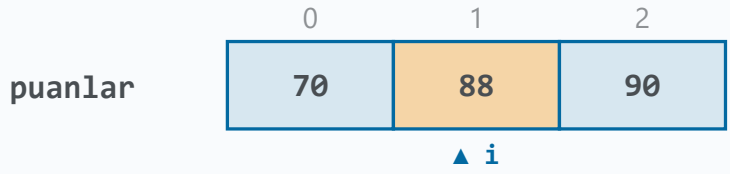
`puanlar = [70, 88, 90]` `i = 1`

ÇIKTI

```
0: 70
```

İndis gerektiğinde (konum yazdırmak, komşu öğeye bakmak, hücreye yazmak) klasik `for` kullanılır.

Klasik `for` ile Gezinme



`puanlar[1]` = 88; `%d: %d` kalıbı **1: 88** üretir.

Adım 5 / 8

```
1 int[] puanlar = {70, 88, 90};
2 for (int i = 0; i < puanlar.length; i++) {
▶   System.out.printf("%d: %d%n", i, puanlar[i]);
4 }
```

DEĞİŞKENLER

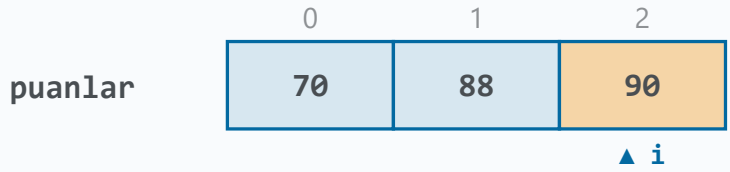
`puanlar = [70, 88, 90]` `i = 1`

ÇIKTI

```
0: 70
1: 88
```

İndis gerektiğinde (konum yazdırmak, komşu öğeye bakmak, hücreye yazmak) klasik `for` kullanılır.

Klasik `for` ile Gezinme



`i++` → **2 < 3 → true**

Adım 6 / 8

```
1 int[] puanlar = {70, 88, 90};
▶ for (int i = 0; i < puanlar.length; i++) {
3   System.out.printf("%d: %d%n", i, puanlar[i]);
4 }
```

DEĞİŞKENLER

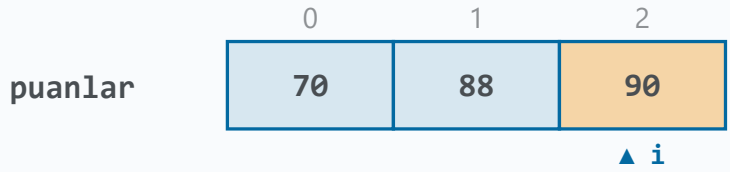
`puanlar = [70, 88, 90]` `i = 2`

ÇIKTI

```
0: 70
1: 88
```

İndis gerektiğinde (konum yazdırmak, komşu öğeye bakmak, hücreye yazmak) klasik `for` kullanılır.

Klasik `for` ile Gezinme



`puanlar[2]` = 90; `%d: %d` kalıbı **2: 90** üretir.

Adım 7 / 8

```
1 int[] puanlar = {70, 88, 90};
2 for (int i = 0; i < puanlar.length; i++) {
▶   System.out.printf("%d: %d%n", i, puanlar[i]);
4 }
```

DEĞİŞKENLER

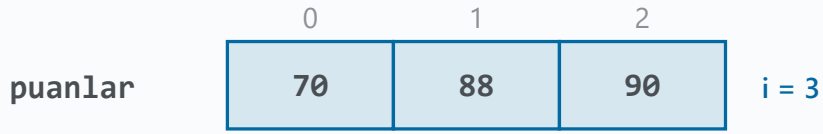
`puanlar = [70, 88, 90]` `i = 2`

ÇIKTI

```
0: 70
1: 88
2: 90
```

İndis gerektiğinde (konum yazdırmak, komşu öğeye bakmak, hücreye yazmak) klasik `for` kullanılır.

Klasik `for` ile Gezinme



`i++` → **3 < 3** → **false** → döngü biter; i = length olduğu için puanlar[i]'ye hiç erişilmez.

Adım 8 / 8

```
1 int[] puanlar = {70, 88, 90};
▶ for (int i = 0; i < puanlar.length; i++) {
3   System.out.printf("%d: %d%n", i, puanlar[i]);
4 }
```

DEĞİŞKENLER

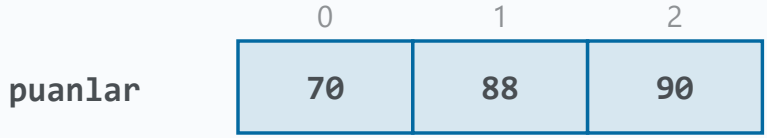
`puanlar = [70, 88, 90]` `i = 3`

ÇIKTI

```
0: 70
1: 88
2: 90
```

İndis gerektiğinde (konum yazdırmak, komşu öğeye bakmak, hücreye yazmak) klasik `for` kullanılır.

Gelişmiş `for` (for-each)



Gelişmiş for dizinin öğelerini sırayla verir; indis yoktur.

Adım 1 / 12

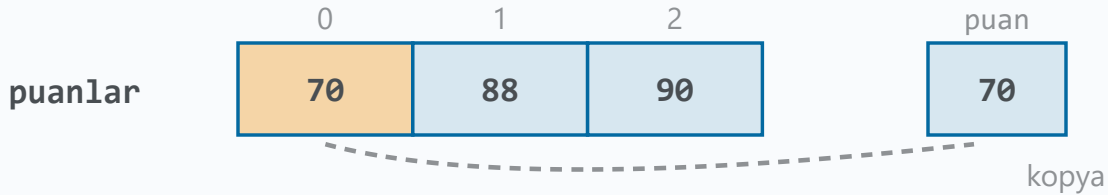
```
▶ int[] puanlar = {70, 88, 90};
2 for (int puan : puanlar) {
3     puan = puan + 5;
4     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

DEĞİŞKENLER

`puanlar = [70, 88, 90]`

ÇIKTI

Gelişmiş `for` (for-each)



1. tur: `puan` 'a `puanlar[0]` 'ın **kopyası** (70) konur.

Adım 2 / 12

```
1 int[] puanlar = {70, 88, 90};
▶ for (int puan : puanlar) {
3     puan = puan + 5;
4     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

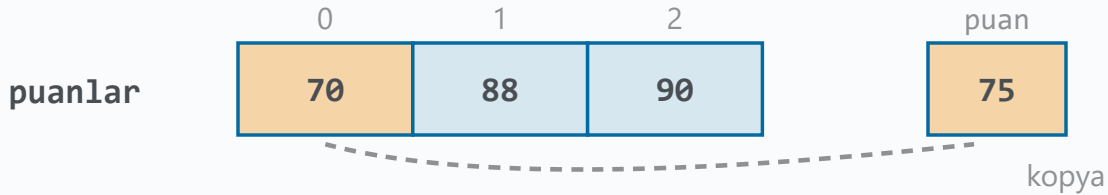
DEĞİŞKENLER

`puanlar = [70, 88, 90]`

`puan = 70`

ÇIKTI

Gelişmiş `for` (for-each)



`puan = 75` olur; dizideki hücre **70** olarak kalır — değişen yalnızca kopya.

Adım 3 / 12

```
1 int[] puanlar = {70, 88, 90};
2 for (int puan : puanlar) {
      puan = puan + 5;
4     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

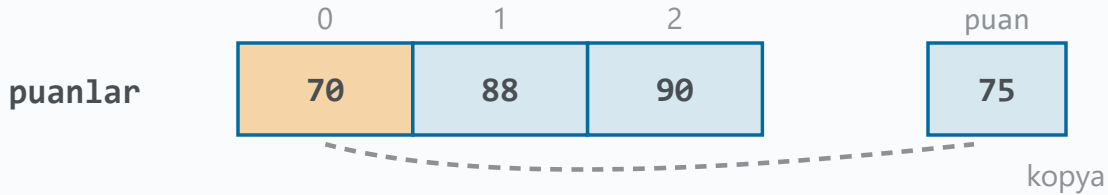
DEĞİŞKENLER

`puanlar = [70, 88, 90]`

`puan = 75`

ÇIKTI

Gelişmiş `for` (for-each)



75 yazdırılır.

Adım 4 / 12

```
1 int[] puanlar = {70, 88, 90};
2 for (int puan : puanlar) {
3     puan = puan + 5;
▶     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

DEĞİŞKENLER

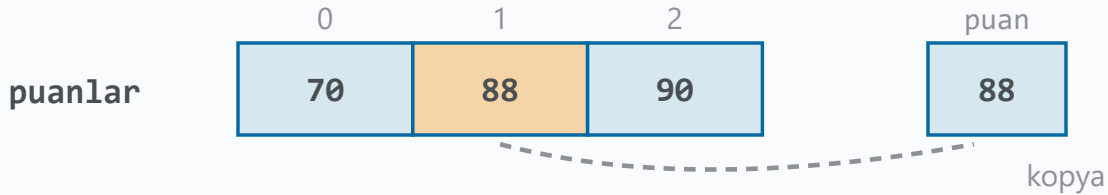
`puanlar = [70, 88, 90]`

`puan = 75`

ÇIKTI

```
75
```

Gelişmiş `for` (for-each)



2. tur: `puan` 'a `puanlar[1]` 'in **kopyası** (88) konur.

Adım 5 / 12

```
1 int[] puanlar = {70, 88, 90};
▶ for (int puan : puanlar) {
3     puan = puan + 5;
4     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

DEĞİŞKENLER

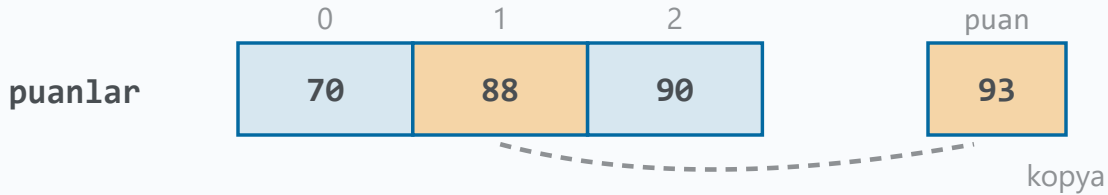
`puanlar = [70, 88, 90]`

`puan = 88`

ÇIKTI

```
75
```

Gelişmiş `for` (for-each)



`puan = 93` olur; dizideki hücre **88** olarak kalır — değişen yalnızca kopya.

Adım 6 / 12

```
1 int[] puanlar = {70, 88, 90};
2 for (int puan : puanlar) {
      puan = puan + 5;
4     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

DEĞİŞKENLER

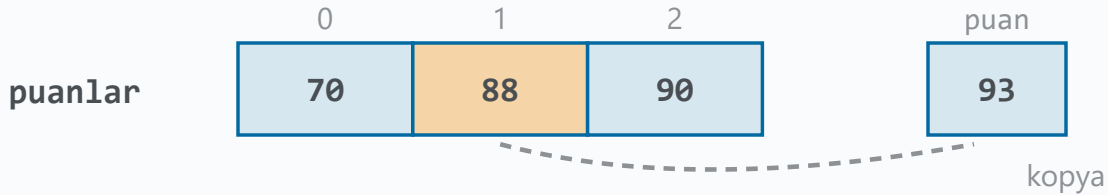
`puanlar = [70, 88, 90]`

`puan = 93`

ÇIKTI

```
75
```

Gelişmiş `for` (for-each)



93 yazdırılır.

Adım 7 / 12

```
1 int[] puanlar = {70, 88, 90};
2 for (int puan : puanlar) {
3     puan = puan + 5;
▶     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

DEĞİŞKENLER

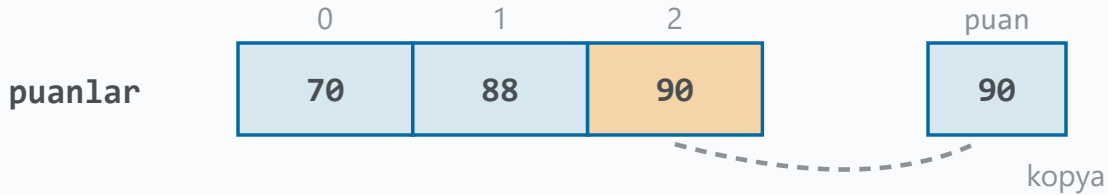
`puanlar = [70, 88, 90]`

`puan = 93`

ÇIKTI

```
75
93
```

Gelişmiş `for` (for-each)



3. tur: `puan` 'a `puanlar[2]` 'nin **kopyası** (90) konur.

Adım 8 / 12

```
1 int[] puanlar = {70, 88, 90};
▶ for (int puan : puanlar) {
3     puan = puan + 5;
4     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

DEĞİŞKENLER

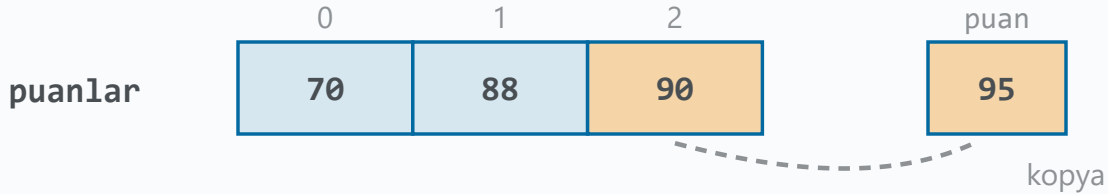
`puanlar = [70, 88, 90]`

`puan = 90`

ÇIKTI

```
75
93
```

Gelişmiş `for` (for-each)



`puan = 95` olur; dizideki hücre **90** olarak kalır — değişen yalnızca kopya.

Adım 9 / 12

```
1 int[] puanlar = {70, 88, 90};
2 for (int puan : puanlar) {
      puan = puan + 5;
4     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

DEĞİŞKENLER

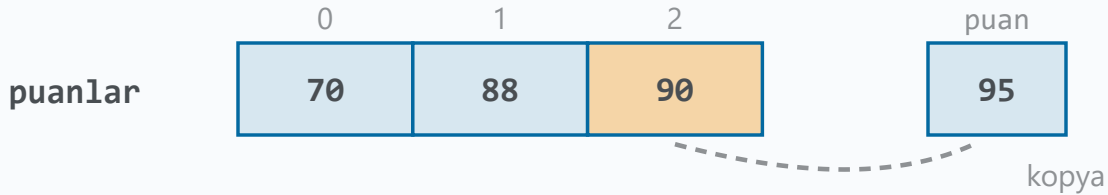
`puanlar = [70, 88, 90]`

`puan = 95`

ÇIKTI

```
75
93
```

Gelişmiş `for` (for-each)



95 yazdırılır.

Adım 10 / 12

```
1 int[] puanlar = {70, 88, 90};
2 for (int puan : puanlar) {
3     puan = puan + 5;
▶     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

DEĞİŞKENLER

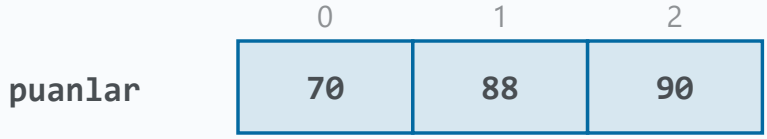
`puanlar = [70, 88, 90]`

`puan = 95`

ÇIKTI

```
75
93
95
```

Gelişmiş `for` (for-each)



Öğe kalmadı → döngü biter; `puan` değişkeni kapsam dışına çıkar.

Adım 11 / 12

```
1 int[] puanlar = {70, 88, 90};
▶ for (int puan : puanlar) {
3     puan = puan + 5;
4     System.out.println(puan);
5 }
6 System.out.println(puanlar[0]);
```

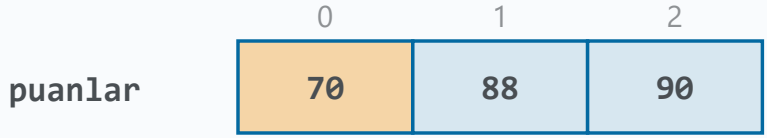
DEĞİŞKENLER

`puanlar = [70, 88, 90]`

ÇIKTI

```
75
93
95
```

Gelişmiş `for` (for-each)



`puanlar[0]` hâlâ **70**: gelişmiş for'da döngü değişkenine atama diziyi değiştirmez.

Adım 12 / 12

```
1 int[] puanlar = {70, 88, 90};
2 for (int puan : puanlar) {
3     puan = puan + 5;
4     System.out.println(puan);
5 }
▶ System.out.println(puanlar[0]);
```

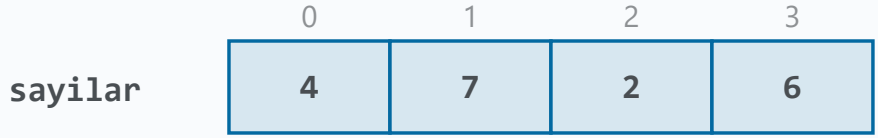
DEĞİŞKENLER

`puanlar = [70, 88, 90]`

ÇIKTI

```
75
93
95
70
```

Toplam ve Ortalama



Dört öğeli dizi.

Adım 1 / 10

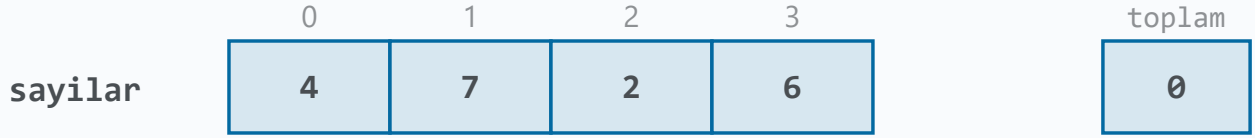
```
▶ int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
3 for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
5 if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

ÇIKTI

Toplam ve Ortalama



Biriktirici `toplam = 0` ile başlar.

Adım 2 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
▶ int toplam = 0;
3 for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
5 if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

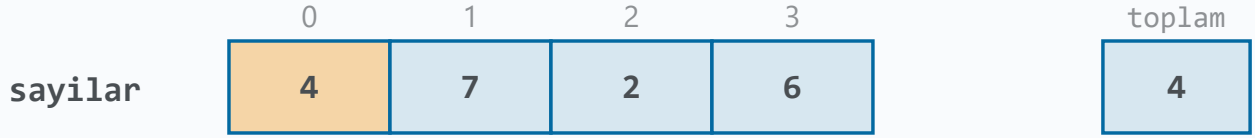
DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

`toplam = 0`

ÇIKTI

Toplam ve Ortalama



`deger = 4` → `toplam += deger` : 0 + 4 = **4**.

Adım 3 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
▶ for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
5 if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

`toplam = 4` `deger = 4`

ÇIKTI

Toplam ve Ortalama



`deger = 7` → `toplam += deger` : 4 + 7 = **11**.

Adım 4 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
▶ for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
5 if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

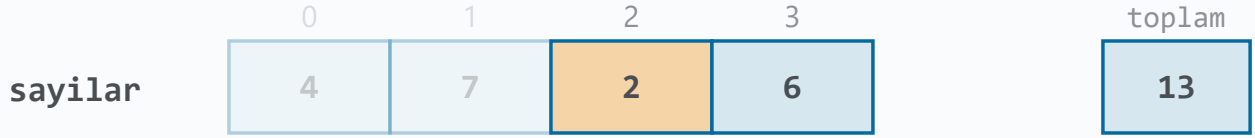
DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

`toplam = 11` `deger = 7`

ÇIKTI

Toplam ve Ortalama



`deger = 2` → `toplam += deger` : 11 + 2 = **13**.

Adım 5 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
▶ for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
5 if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

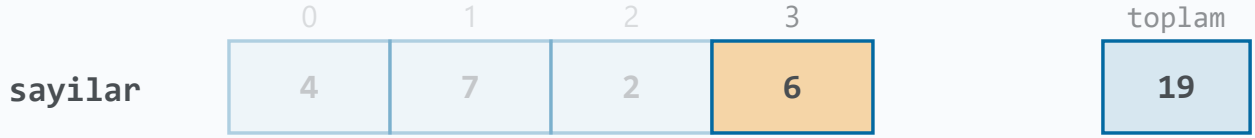
DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

`toplam = 13` `deger = 2`

ÇIKTI

Toplam ve Ortalama



`deger = 6` → `toplam += deger` : 13 + 6 = **19**.

Adım 6 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
▶ for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
5 if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

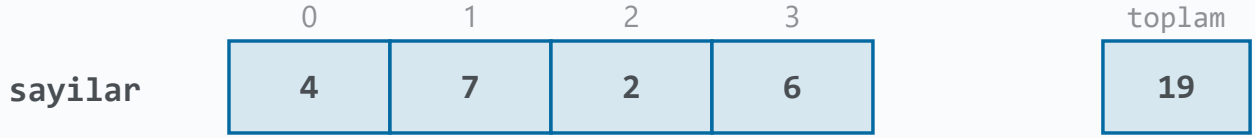
DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

`toplam = 19` `deger = 6`

ÇIKTI

Toplam ve Ortalama



Döngü bitti. `ortalama` 0 ile başlatılır (boş dizi için güvenli değer).

Adım 7 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
3 for (int deger : sayilar) toplam += deger;
▶ double ortalama = 0;
5 if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

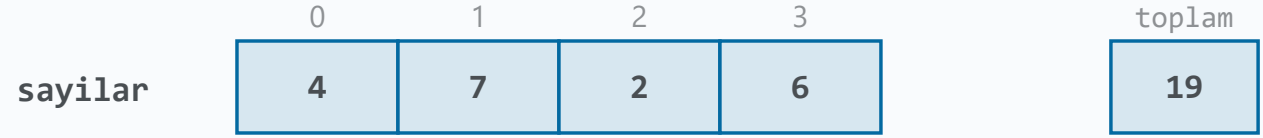
DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

`toplam = 19` `ortalama = 0.0`

ÇIKTI

Toplam ve Ortalama



4 > 0 → **true**: dizi boş değil, bölme yapılabilir.

Adım 8 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
3 for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
▶ if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

`toplam = 19` `ortalama = 0.0`

ÇIKTI

Toplam ve Ortalama



`(double) 19 / 4` = **4.75**. Dönüşüm olmasa tam sayı bölmesi 4 verirdi.

Adım 9 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
3 for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
5 if (sayilar.length > 0)
▶     ortalama = (double) toplam / sayilar.length;
7 System.out.println(toplam + " " + ortalama);
```

DEĞİŞKENLER

`sayilar = [4, 7, 2, 6]`

`toplam = 19` `ortalama = 4.75`

ÇIKTI

Toplam ve Ortalama



Sonuçlar yazdırılır.

Adım 10 / 10

```
1 int[] sayilar = {4, 7, 2, 6};
2 int toplam = 0;
3 for (int deger : sayilar) toplam += deger;
4 double ortalama = 0;
5 if (sayilar.length > 0)
6     ortalama = (double) toplam / sayilar.length;
▶ System.out.println(toplam + " " + ortalama);
```

DEĞİŞKENLER

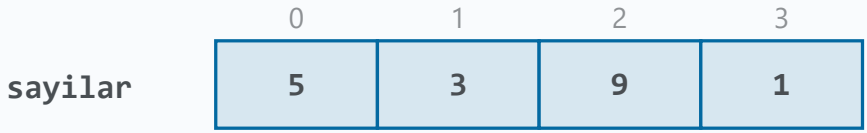
`sayilar = [4, 7, 2, 6]`

`toplam = 19` `ortalama = 4.75`

ÇIKTI

```
19 4.75
```

En Küçük ve En Büyük



Dizi: 5, 3, 9, 1.

Adım 1 / 15

```
▶ int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
4     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

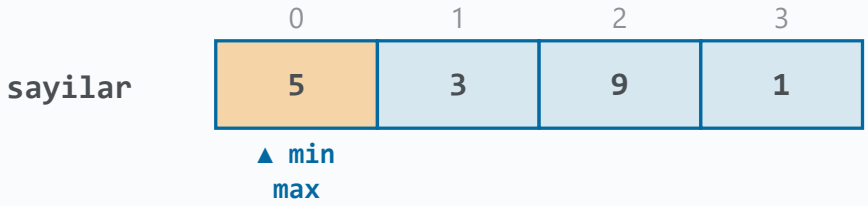
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



İlk öğe hem **min** hem **max** adayıdır (0 ile başlatmak negatif/pozitif dizilerde yanlış sonuç verirdi).

Adım 2 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
▶ int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
4     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

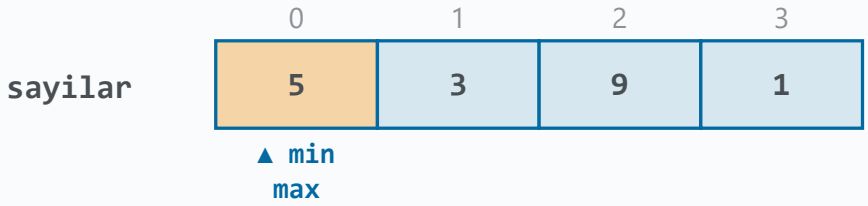
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 5` `max = 5`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



1. tur: `deger = 5`.

Adım 3 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
▶ for (int deger : sayilar) {
4     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

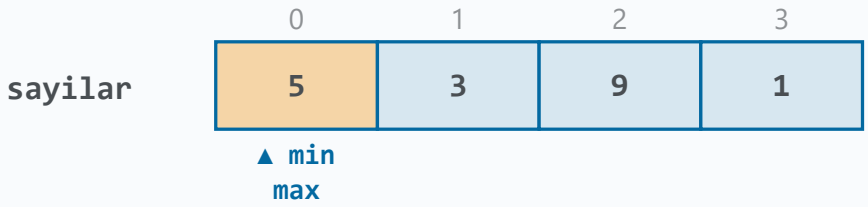
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 5` `max = 5` `deger = 5`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



5 < 5 → **false**; min değişmez.

Adım 4 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
▶     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

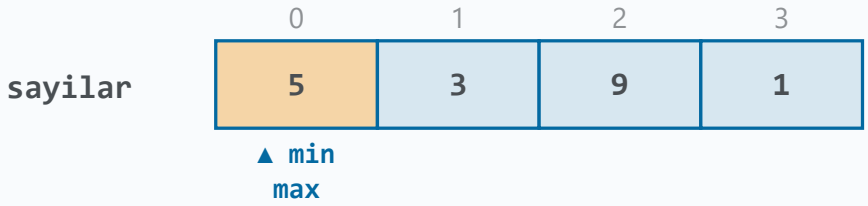
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 5` `max = 5` `deger = 5`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



5 > 5 → **false**; max değişmez.

Adım 5 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
4     if (deger < min) min = deger;
▶     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

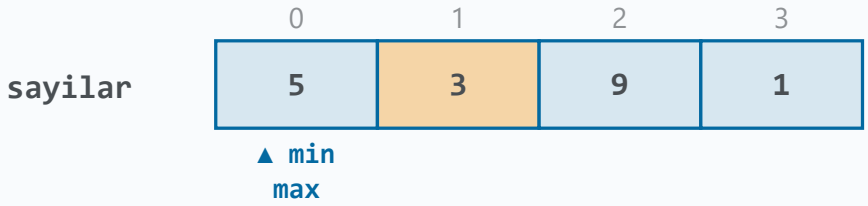
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 5` `max = 5` `deger = 5`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



2. tur: `deger = 3`.

Adım 6 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
▶ for (int deger : sayilar) {
4     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

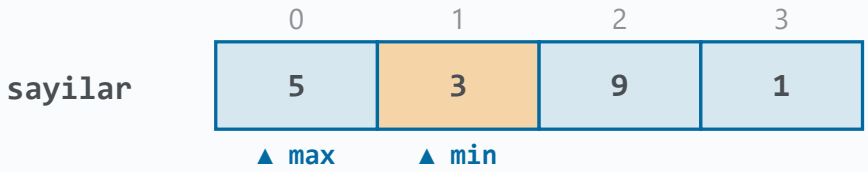
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 5` `max = 5` `deger = 3`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



3 < 5 → **true**: min = 3.

Adım 7 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
▶     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

DEĞİŞKENLER

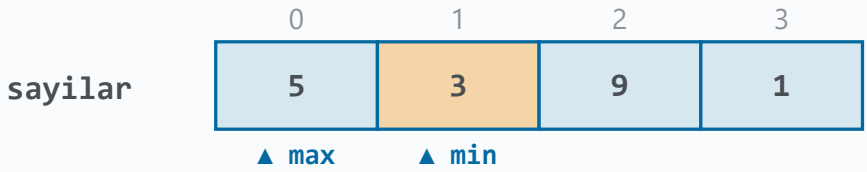
`sayilar = [5, 3, 9, 1]` `min = 3`

`max = 5` `deger = 3`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



3 > 5 → **false**; max değişmez.

Adım 8 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
4     if (deger < min) min = deger;
▶     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

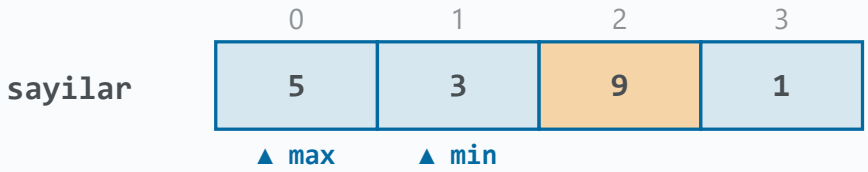
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 3` `max = 5` `deger = 3`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



3. tur: `deger = 9`.

Adım 9 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
▶ for (int deger : sayilar) {
4     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

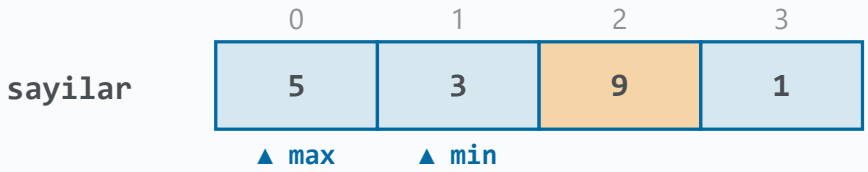
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 3` `max = 5` `deger = 9`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



9 < 3 → **false**; min değişmez.

Adım 10 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
▶     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

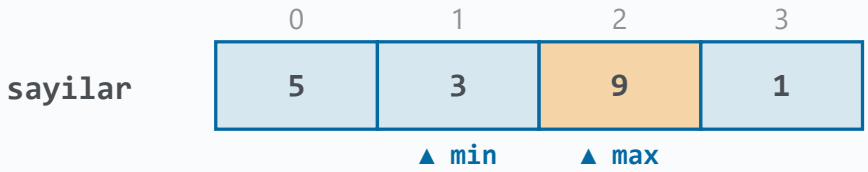
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 3` `max = 5` `deger = 9`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



9 > 5 → **true**: max = 9.

Adım 11 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
4     if (deger < min) min = deger;
▶     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

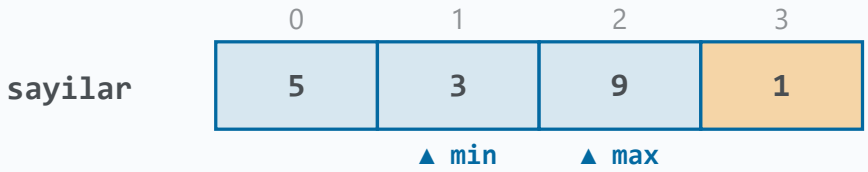
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 3` `max = 9` `deger = 9`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



4. tur: `deger = 1`.

Adım 12 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
▶ for (int deger : sayilar) {
4     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

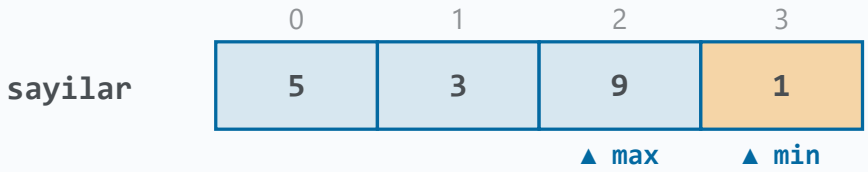
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 3` `max = 9` `deger = 1`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



1 < 3 → **true**: min = 1.

Adım 13 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
▶     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

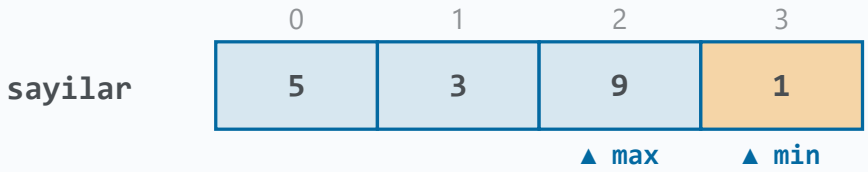
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 1` `max = 9` `deger = 1`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



1 > 9 → **false**; max değişmez.

Adım 14 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
4     if (deger < min) min = deger;
▶     if (deger > max) max = deger;
6 }
7 System.out.println(min + " " + max);
```

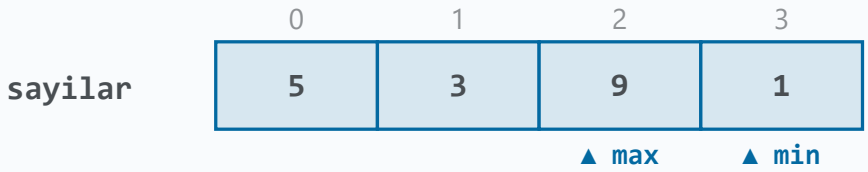
DEĞİŞKENLER

`sayilar = [5, 3, 9, 1]` `min = 1` `max = 9` `deger = 1`

ÇIKTI

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

En Küçük ve En Büyük



Tüm öğelere bakıldı (n = 4 tur) → **1 9**.

Adım 15 / 15

```
1 int[] sayilar = {5, 3, 9, 1};
2 int min = sayilar[0], max = sayilar[0];
3 for (int deger : sayilar) {
4     if (deger < min) min = deger;
5     if (deger > max) max = deger;
6 }
▶ System.out.println(min + " " + max);
```

DEĞİŞKENLER

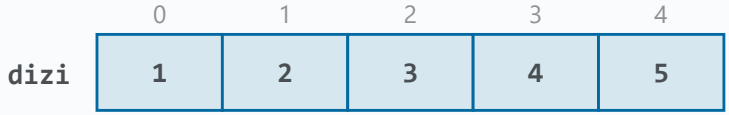
`sayilar = [5, 3, 9, 1]` `min = 1` `max = 9`

ÇIKTI

```
1 9
```

Boş dizide `sayilar[0]` hata verir; gerçek kodda önce `sayilar.length == 0` durumu ele alınmalıdır.

Diziyi Ters Çevirme (iki işaretçi)



Tersine çevrilecek dizi.

Adım 1 / 13

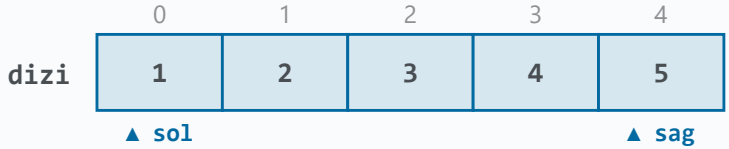
```
▶ int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [1, 2, 3, 4, 5]`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



İki işaretçi: **sol = 0** ve **sag = length − 1 = 4**.

Adım 2 / 13

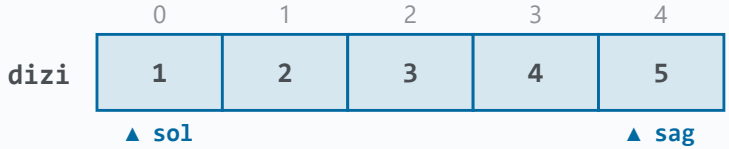
```
1 int[] dizi = {1, 2, 3, 4, 5};
▶ int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [1, 2, 3, 4, 5]` `sol = 0` `sag = 4`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



0 < 4 → **true**, uçlar yer değiştirecek.

Adım 3 / 13

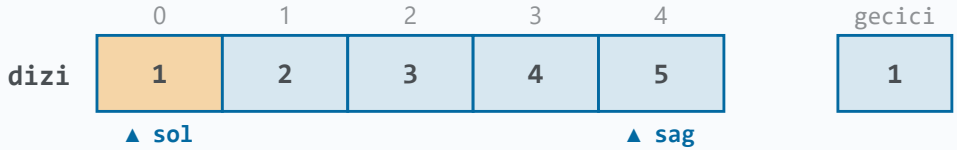
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
▶ while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [1, 2, 3, 4, 5]` `sol = 0` `sag = 4`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



`gecici = dizi[0] = 1` : soldaki değer kaybolmasın diye saklanır.

Adım 4 / 13

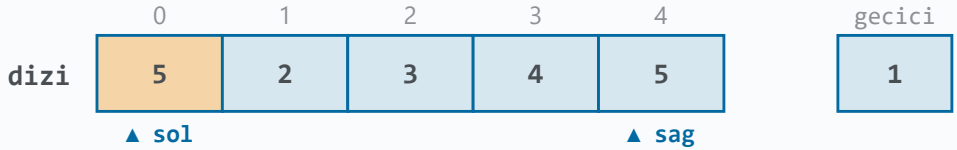
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
▶     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [1, 2, 3, 4, 5]` `sol = 0` `sag = 4` `gecici = 1`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



`dizi[0] = dizi[4]` → sol hücreye **5** yazılır.

Adım 5 / 13

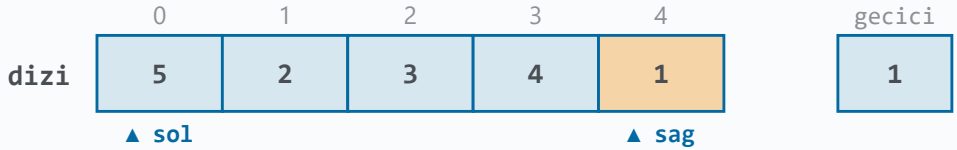
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
4     int gecici = dizi[sol];
▶     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 2, 3, 4, 5]` `sol = 0` `sag = 4` `gecici = 1`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



`dizi[4] = gecici` → sağ hücreye **1**. Takas tamam.

Adım 6 / 13

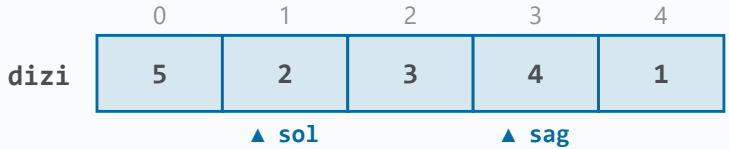
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
▶     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 2, 3, 4, 1]` `sol = 0` `sag = 4` `gecici = 1`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



İşaretçiler merkeze yaklaşır: **sol = 1**, **sag = 3**.

Adım 7 / 13

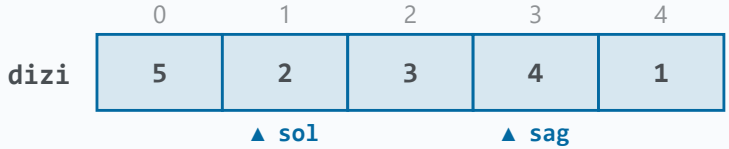
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
▶     sol++;
▶     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 2, 3, 4, 1]` `sol = 1` `sag = 3`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



1 < 3 → **true**, uçlar yer değiştirecek.

Adım 8 / 13

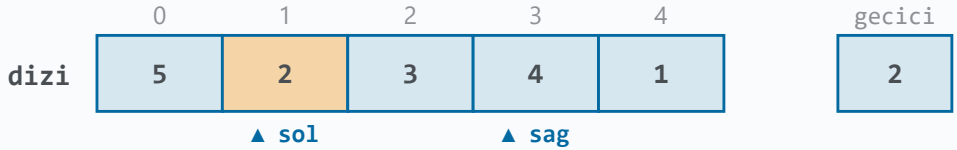
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
▶ while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 2, 3, 4, 1]` `sol = 1` `sag = 3`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



`gecici = dizi[1] = 2` : soldaki değer kaybolmasın diye saklanır.

Adım 9 / 13

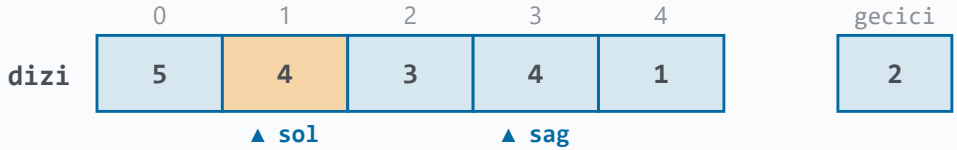
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
▶     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 2, 3, 4, 1]` `sol = 1` `sag = 3` `gecici = 2`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



`dizi[1] = dizi[3]` → sol hücreye **4** yazılır.

Adım 10 / 13

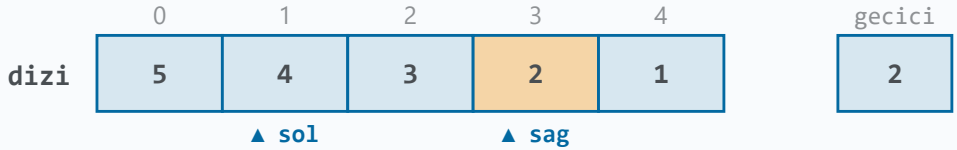
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
4     int gecici = dizi[sol];
▶     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 4, 3, 4, 1]` `sol = 1` `sag = 3` `gecici = 2`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



`dizi[3] = gecici` → sağ hücreye **2**. Takas tamam.

Adım 11 / 13

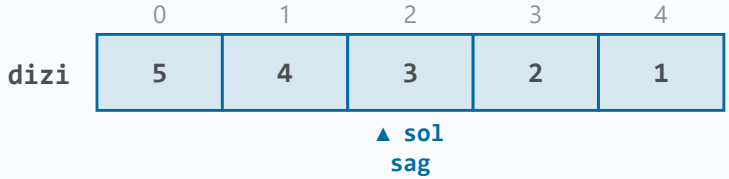
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
▶     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 4, 3, 2, 1]` `sol = 1` `sag = 3` `gecici = 2`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



İşaretçiler merkeze yaklaşır: **sol = 2**, **sag = 2**.

Adım 12 / 13

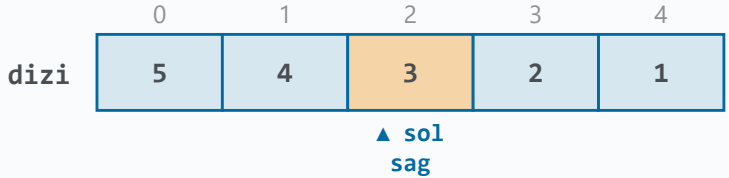
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
3 while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
▶     sol++;
▶     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 4, 3, 2, 1]` `sol = 2` `sag = 2`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Diziyi Ters Çevirme (iki işaretçi)



2 < 2 → **false**: işaretçiler ortada buluştu → dizi ters çevrildi.

Adım 13 / 13

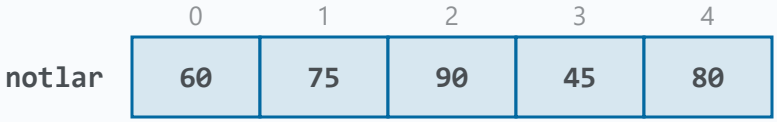
```
1 int[] dizi = {1, 2, 3, 4, 5};
2 int sol = 0, sag = dizi.length - 1;
▶ while (sol < sag) {
4     int gecici = dizi[sol];
5     dizi[sol] = dizi[sag];
6     dizi[sag] = gecici;
7     sol++;
8     sag--;
9 }
```

DEĞİŞKENLER

`dizi = [5, 4, 3, 2, 1]` `sol = 2` `sag = 2`

n/2 takas yapılır; ek dizi gerekmez (yerinde, O(1) ek alan).

Ortalamanın Üstündekileri Sayma



Beş öğrencinin notu. Önce ortalama, sonra ortalamanın üstündekiler: **iki geçiş** gerekir.

Adım 1 / 15

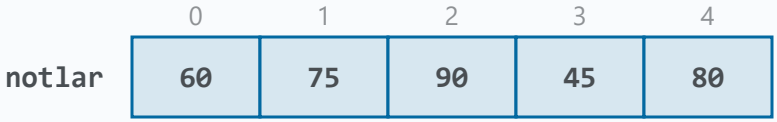
```
▶ int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

ÇIKTI

Ortalamanın Üstündekileri Sayma



1. geçiş: toplam biriktirilir.

Adım 2 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
▶ int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

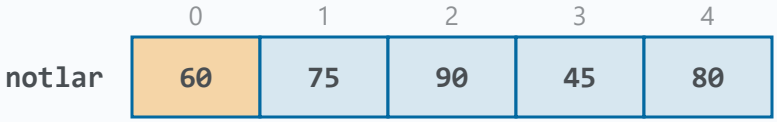
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`toplam = 0`

ÇIKTI

Ortalamanın Üstündekileri Sayma



x = 60 → toplam = **60**.

Adım 3 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
▶ for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

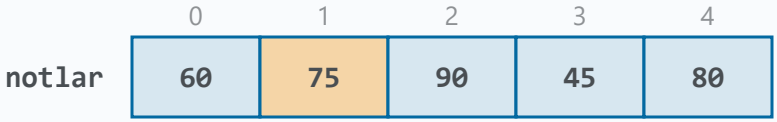
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`toplam = 60` `x = 60`

ÇIKTI

Ortalamanın Üstündekileri Sayma



x = 75 → toplam = **135**.

Adım 4 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
▶ for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

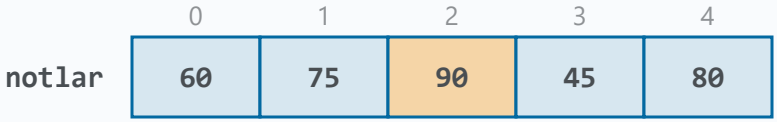
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`toplam = 135` `x = 75`

ÇIKTI

Ortalamanın Üstündekileri Sayma



x = 90 → toplam = **225**.

Adım 5 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
▶ for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

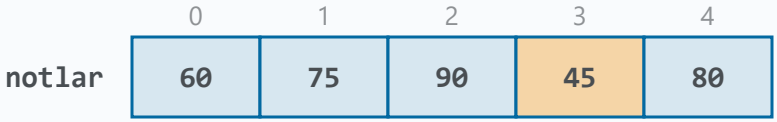
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`toplam = 225` `x = 90`

ÇIKTI

Ortalamanın Üstündekileri Sayma



x = 45 → toplam = **270**.

Adım 6 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
▶ for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

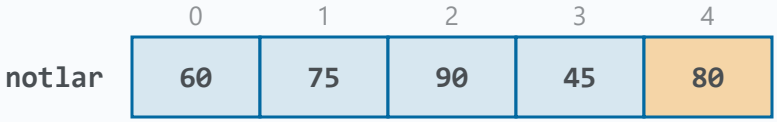
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`toplam = 270` `x = 45`

ÇIKTI

Ortalamanın Üstündekileri Sayma



x = 80 → toplam = **350**.

Adım 7 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
▶ for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

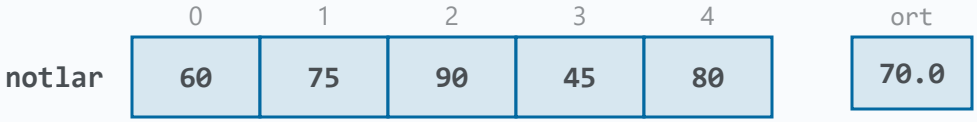
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`toplam = 350` `x = 80`

ÇIKTI

Ortalamanın Üstündekileri Sayma



`(double) 350 / 5` = **70.0**.

Adım 8 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
▶ double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

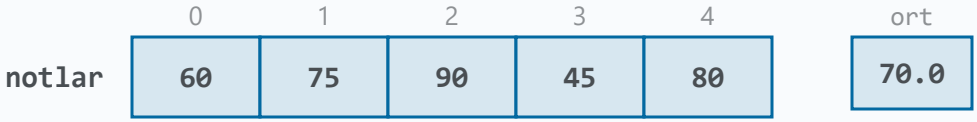
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`toplam = 350` `ort = 70.0`

ÇIKTI

Ortalamanın Üstündekileri Sayma



2. geçiş için sayaç: ustte = 0.

Adım 9 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
▶ int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

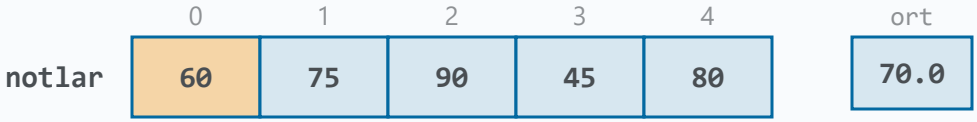
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`ort = 70.0` `ustte = 0`

ÇIKTI

Ortalamanın Üstündekileri Sayma



60 > 70.0 → false.

Adım 10 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
▶ for (int x : notlar)
▶   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

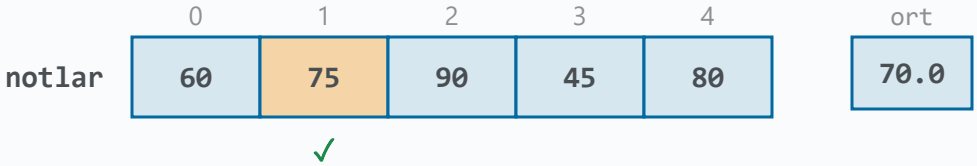
DEĞİŞKENLER

notlar = [60, 75, 90, 45, 80]

ort = 70.0 | ustte = 0 | x = 60

ÇIKTI

Ortalamanın Üstündekileri Sayma



75 > 70.0 → **true**: ustte = 1.

Adım 11 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
▶ for (int x : notlar)
▶   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

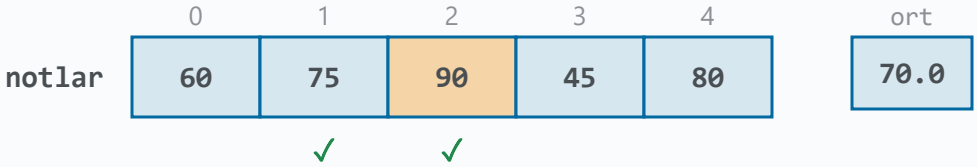
DEĞİŞKENLER

notlar = [60, 75, 90, 45, 80]

ort = 70.0 | ustte = 1 | x = 75

ÇIKTI

Ortalamanın Üstündekileri Sayma



90 > 70.0 → **true**: ustte = 2.

Adım 12 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
▶ for (int x : notlar)
▶   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

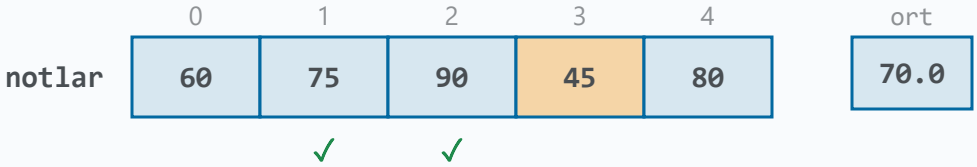
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`ort = 70.0` `ustte = 2` `x = 90`

ÇIKTI

Ortalamanın Üstündekileri Sayma



45 > 70.0 → false.

Adım 13 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
▶ for (int x : notlar)
▶   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

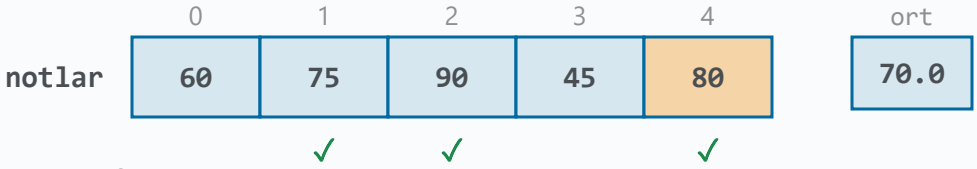
DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`ort = 70.0` `ustte = 2` `x = 45`

ÇIKTI

Ortalamanın Üstündekileri Sayma



80 > 70.0 → **true**: ustte = 3.

Adım 14 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
▶ for (int x : notlar)
▶   if (x > ort) ustte++;
8 System.out.println(ort + " " + ustte);
```

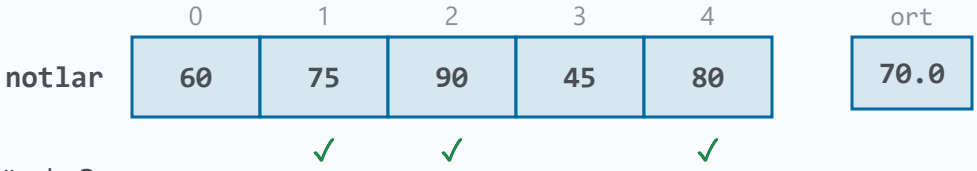
DEĞİŞKENLER

notlar = [60, 75, 90, 45, 80]

ort = 70.0 | ustte = 3 | x = 80

ÇIKTI

Ortalamanın Üstündekileri Sayma



Ortalama 70.0, üstünde **3** not var.

Adım 15 / 15

```
1 int[] notlar = {60, 75, 90, 45, 80};
2 int toplam = 0;
3 for (int x : notlar) toplam += x;
4 double ort = (double) toplam / notlar.length;
5 int ustte = 0;
6 for (int x : notlar)
7   if (x > ort) ustte++;
▶ System.out.println(ort + " " + ustte);
```

DEĞİŞKENLER

`notlar = [60, 75, 90, 45, 80]`

`ort = 70.0` `ustte = 3`

ÇIKTI

```
70.0 3
```

İŞLEMLER · KODU ADIM ADIM ÇALIŞTIR

Dizideki En Büyük İkinci Öğe



İki aday en küçük int değeriyle (Integer.MIN_VALUE) başlar; her öğe bir kez incelenir.

Adım 1 / 7

```
▶ int[] a = {7, 12, 5, 12, 9};
▶ int enBuyuk = Integer.MIN_VALUE;
▶ int ikinci = Integer.MIN_VALUE;
4 for (int x : a) {
5   if (x > enBuyuk) {
6     ikinci = enBuyuk;
7     enBuyuk = x;
8   } else if (x > ikinci && x != enBuyuk) {
9     ikinci = x;
10  }
11 }
12 System.out.println(ikinci);
```

DEĞİŞKENLER

`a = [7, 12, 5, 12, 9]`

`enBuyuk = MIN` `ikinci = MIN`

ÇIKTI

Dizideki En Büyük İkinci Öğe



7 > MIN → yeni en büyük: eski en büyük (MIN) ikinciye iner, enBuyuk = 7.

Adım 2 / 7

```
1  int[] a = {7, 12, 5, 12, 9};
2  int enBuyuk = Integer.MIN_VALUE;
3  int ikinci = Integer.MIN_VALUE;
4  for (int x : a) {
▶    if (x > enBuyuk) {
▶      ikinci = enBuyuk;
▶      enBuyuk = x;
8    } else if (x > ikinci && x != enBuyuk) {
9      ikinci = x;
10   }
11 }
12 System.out.println(ikinci);
```

DEĞİŞKENLER

`a = [7, 12, 5, 12, 9]` `x = 7`
`enBuyuk = 7` `ikinci = MIN`

ÇIKTI

Dizideki En Büyük İkinci Öğe



12 > 7 → yeni en büyük: eski en büyük (7) ikinciye iner, enBuyuk = 12.

Adım 3 / 7

```
1  int[] a = {7, 12, 5, 12, 9};
2  int enBuyuk = Integer.MIN_VALUE;
3  int ikinci = Integer.MIN_VALUE;
4  for (int x : a) {
▶    if (x > enBuyuk) {
▶      ikinci = enBuyuk;
▶      enBuyuk = x;
8    } else if (x > ikinci && x != enBuyuk) {
9      ikinci = x;
10   }
11 }
12 System.out.println(ikinci);
```

DEĞİŞKENLER

`a = [7, 12, 5, 12, 9]` `x = 12` `enBuyuk = 12` `ikinci = 7`

ÇIKTI

Dizideki En Büyük İkinci Öğe



Değişiklik yok: 5 ikinciden (7) büyük değil.

Adım 4 / 7

```
1  int[] a = {7, 12, 5, 12, 9};
2  int enBuyuk = Integer.MIN_VALUE;
3  int ikinci = Integer.MIN_VALUE;
4  for (int x : a) {
5    if (x > enBuyuk) {
6      ikinci = enBuyuk;
7      enBuyuk = x;
▶    } else if (x > ikinci && x != enBuyuk) {
9      ikinci = x;
10   }
11 }
12 System.out.println(ikinci);
```

DEĞİŞKENLER

`a = [7, 12, 5, 12, 9]` `x = 5`

`enBuyuk = 12` `ikinci = 7`

ÇIKTI

Dizideki En Büyük İkinci Öğe



Değişiklik yok: 12 == enBuyuk (tekrar), sayılmaz.

Adım 5 / 7

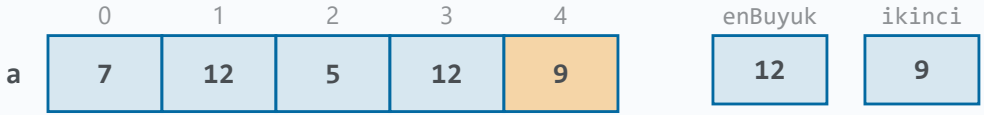
```
1  int[] a = {7, 12, 5, 12, 9};
2  int enBuyuk = Integer.MIN_VALUE;
3  int ikinci = Integer.MIN_VALUE;
4  for (int x : a) {
5    if (x > enBuyuk) {
6      ikinci = enBuyuk;
7      enBuyuk = x;
▶    } else if (x > ikinci && x != enBuyuk) {
9      ikinci = x;
10   }
11 }
12 System.out.println(ikinci);
```

DEĞİŞKENLER

`a = [7, 12, 5, 12, 9]` `x = 12`
`enBuyuk = 12` `ikinci = 7`

ÇIKTI

Dizideki En Büyük İkinci Öğe



9 en büyükten küçük ama **ikinciden büyük** ve 9 != 12 → ikinci = 9.

Adım 6 / 7

```
1  int[] a = {7, 12, 5, 12, 9};
2  int enBuyuk = Integer.MIN_VALUE;
3  int ikinci = Integer.MIN_VALUE;
4  for (int x : a) {
5    if (x > enBuyuk) {
6      ikinci = enBuyuk;
7      enBuyuk = x;
▶    } else if (x > ikinci && x != enBuyuk) {
▶      ikinci = x;
10   }
11 }
12 System.out.println(ikinci);
```

DEĞİŞKENLER

`a = [7, 12, 5, 12, 9]` `x = 9`
`enBuyuk = 12` `ikinci = 9`

ÇIKTI

İŞLEMLER · KODU ADIM ADIM ÇALIŞTIR

Dizideki En Büyük İkinci Öğe



Tek geçişte sonuç: **9** (12 iki kez geçse de ikinci en büyük 9'dur).

Adım 7 / 7

```
1  int[] a = {7, 12, 5, 12, 9};
2  int enBuyuk = Integer.MIN_VALUE;
3  int ikinci = Integer.MIN_VALUE;
4  for (int x : a) {
5    if (x > enBuyuk) {
6      ikinci = enBuyuk;
7      enBuyuk = x;
8    } else if (x > ikinci && x != enBuyuk) {
9      ikinci = x;
10   }
11 }
▶ System.out.println(ikinci);
```

DEĞİŞKENLER

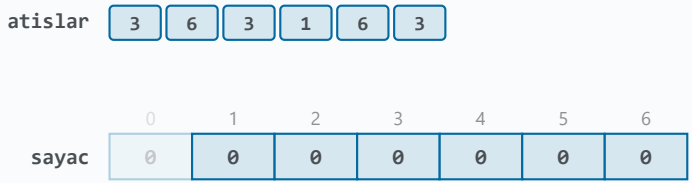
`a = [7, 12, 5, 12, 9]`

`enBuyuk = 12` `ikinci = 9`

ÇIKTI

```
9
```

Frekans Sayma: Zar Histogramı



Atışlar sabit verildi (gerçekte `rastgele.nextInt(6) + 1`). 7 hücreli sayaç: indis = zar değeri (0 kullanılmaz). Bu, **değeri indis olarak kullanma** hilesidir.

Adım 1 / 13

```
▶ int[] atislar = {3, 6, 3, 1, 6, 3};
▶ int[] sayac = new int[7];
3 for (int zar : atislar)
4   sayac[zar]++;
5 for (int y = 1; y <= 6; y++) {
6   System.out.print(y + ": ");
7   for (int k = 0; k < sayac[y]; k++)
8     System.out.print("*");
9   System.out.println();
10 }
```

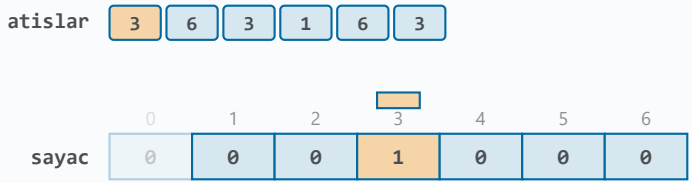
DEĞİŞKENLER

`atislar = [3, 6, 3, 1, 6, 3]`

`sayac = [0, 0, 0, 0, 0, 0, 0]`

ÇIKTI

Frekans Sayma: Zar Histogramı



zar = 3 → **sayac[3]++** → 1. if-else zincirine gerek yok!

Adım 2 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
▶  for (int zar : atislar)
▶    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
6    System.out.print(y + ": ");
7    for (int k = 0; k < sayac[y]; k++)
8      System.out.print("*");
9    System.out.println();
10 }
```

DEĞİŞKENLER

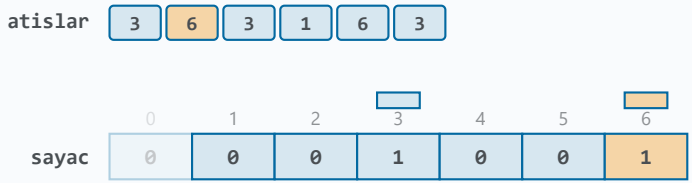
atislar = [3, 6, 3, 1, 6, 3]

sayac = [0, 0, 0, 1, 0, 0, 0]

zar = 3

ÇIKTI

Frekans Sayma: Zar Histogramı



zar = 6 → **sayac[6]++** → 1. if-else zincirine gerek yok!

Adım 3 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
▶  for (int zar : atislar)
▶    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
6    System.out.print(y + ": ");
7    for (int k = 0; k < sayac[y]; k++)
8      System.out.print("*");
9    System.out.println();
10 }
```

DEĞİŞKENLER

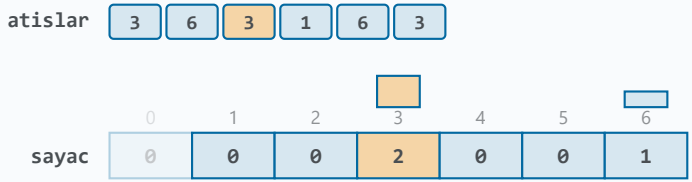
atislar = [3, 6, 3, 1, 6, 3]

sayac = [0, 0, 0, 1, 0, 0, 1]

zar = 6

ÇIKTI

Frekans Sayma: Zar Histogramı



zar = 3 → **sayac[3]++** → 2. if-else zincirine gerek yok!

Adım 4 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
▶  for (int zar : atislar)
▶    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
6    System.out.print(y + ": ");
7    for (int k = 0; k < sayac[y]; k++)
8      System.out.print("*");
9    System.out.println();
10 }
```

DEĞİŞKENLER

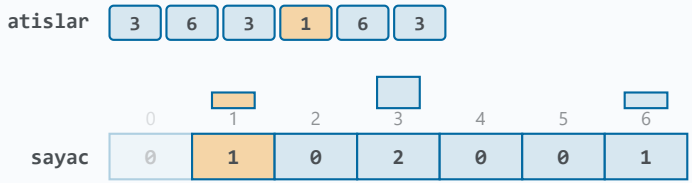
atislar = [3, 6, 3, 1, 6, 3]

sayac = [0, 0, 0, 2, 0, 0, 1]

zar = 3

ÇIKTI

Frekans Sayma: Zar Histogramı



zar = 1 → **sayac[1]++** → 1. if-else zincirine gerek yok!

Adım 5 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
▶  for (int zar : atislar)
▶    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
6    System.out.print(y + ": ");
7    for (int k = 0; k < sayac[y]; k++)
8      System.out.print("*");
9    System.out.println();
10 }
```

DEĞİŞKENLER

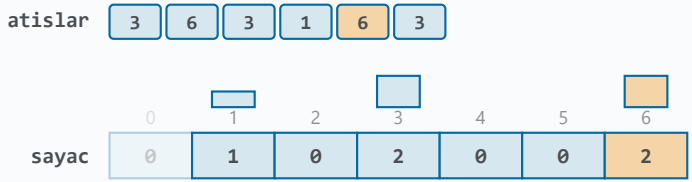
atislar = [3, 6, 3, 1, 6, 3]

sayac = [0, 1, 0, 2, 0, 0, 1]

zar = 1

ÇIKTI

Frekans Sayma: Zar Histogramı



zar = 6 → **sayac[6]++** → 2. if-else zincirine gerek yok!

Adım 6 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
▶  for (int zar : atislar)
▶    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
6    System.out.print(y + ": ");
7    for (int k = 0; k < sayac[y]; k++)
8      System.out.print("*");
9    System.out.println();
10 }
```

DEĞİŞKENLER

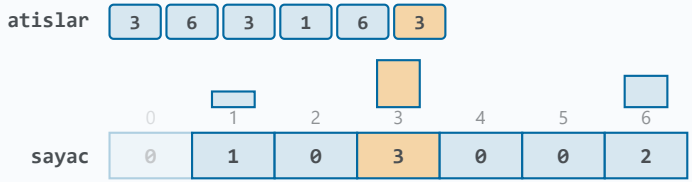
atislar = [3, 6, 3, 1, 6, 3]

sayac = [0, 1, 0, 2, 0, 0, 2]

zar = 6

ÇIKTI

Frekans Sayma: Zar Histogramı



zar = 3 → **sayac[3]++** → 3. if-else zincirine gerek yok!

Adım 7 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
▶  for (int zar : atislar)
▶    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
6    System.out.print(y + ": ");
7    for (int k = 0; k < sayac[y]; k++)
8      System.out.print("*");
9    System.out.println();
10 }
```

DEĞİŞKENLER

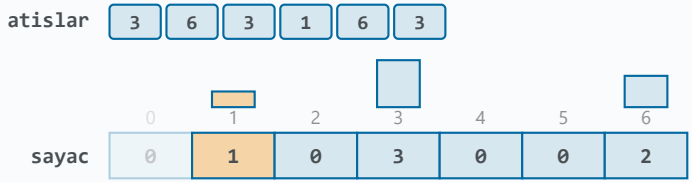
atislar = [3, 6, 3, 1, 6, 3]

sayac = [0, 1, 0, 3, 0, 0, 2]

zar = 3

ÇIKTI

Frekans Sayma: Zar Histogramı



y = 1: sayac[1] = 1 kadar yıldız.

Adım 8 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
3  for (int zar : atislar)
4    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
▶    System.out.print(y + ": ");
▶    for (int k = 0; k < sayac[y]; k++)
▶      System.out.print("*");
▶    System.out.println();
10 }
```

DEĞİŞKENLER

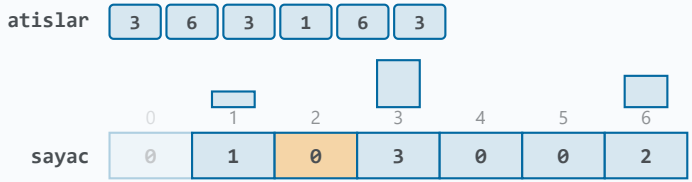
`sayac = [0, 1, 0, 3, 0, 0, 2]`

`y = 1`

ÇIKTI

```
1: *
```

Frekans Sayma: Zar Histogramı



y = 2: sayac[2] = 0 kadar yıldız (hiç yok).

Adım 9 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
3  for (int zar : atislar)
4    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
▶    System.out.print(y + ": ");
▶    for (int k = 0; k < sayac[y]; k++)
▶      System.out.print("*");
▶    System.out.println();
10 }
```

DEĞİŞKENLER

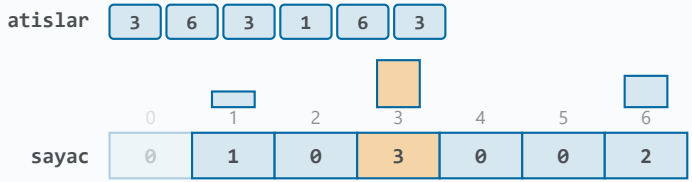
`sayac = [0, 1, 0, 3, 0, 0, 2]`

`y = 2`

ÇIKTI

```
1: *
2:
```

Frekans Sayma: Zar Histogramı



y = 3: sayac[3] = 3 kadar yıldız.

Adım 10 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
3  for (int zar : atislar)
4    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
▶    System.out.print(y + ": ");
▶    for (int k = 0; k < sayac[y]; k++)
▶      System.out.print("*");
▶    System.out.println();
10 }
```

DEĞİŞKENLER

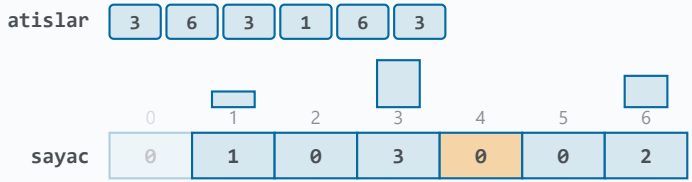
`sayac = [0, 1, 0, 3, 0, 0, 2]`

`y = 3`

ÇIKTI

```
1: *
2:
3: ***
```

Frekans Sayma: Zar Histogramı



y = 4: sayac[4] = 0 kadar yıldız (hiç yok).

Adım 11 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
3  for (int zar : atislar)
4    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
▶    System.out.print(y + ": ");
▶    for (int k = 0; k < sayac[y]; k++)
▶      System.out.print("*");
▶    System.out.println();
10 }
```

DEĞİŞKENLER

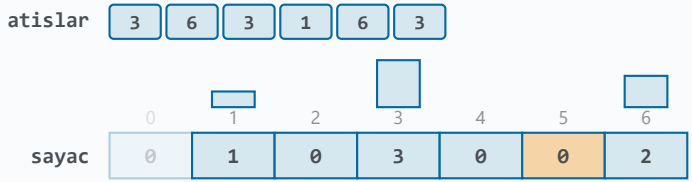
`sayac = [0, 1, 0, 3, 0, 0, 2]`

`y = 4`

ÇIKTI

```
1: *
2:
3: ***
4:
```

Frekans Sayma: Zar Histogramı



y = 5: sayac[5] = 0 kadar yıldız (hiç yok).

Adım 12 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
3  for (int zar : atislar)
4    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
▶    System.out.print(y + ": ");
▶    for (int k = 0; k < sayac[y]; k++)
▶      System.out.print("*");
▶    System.out.println();
10 }
```

DEĞİŞKENLER

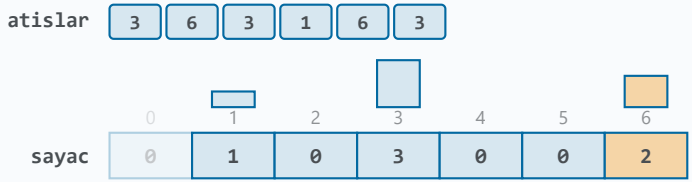
`sayac = [0, 1, 0, 3, 0, 0, 2]`

`y = 5`

ÇIKTI

```
1: *
2:
3: ***
4:
5:
```

Frekans Sayma: Zar Histogramı



y = 6: sayac[6] = 2 kadar yıldız.

Adım 13 / 13

```
1  int[] atislar = {3, 6, 3, 1, 6, 3};
2  int[] sayac = new int[7];
3  for (int zar : atislar)
4    sayac[zar]++;
5  for (int y = 1; y <= 6; y++) {
▶    System.out.print(y + ": ");
▶    for (int k = 0; k < sayac[y]; k++)
▶      System.out.print("*");
▶    System.out.println();
10 }
```

DEĞİŞKENLER

`sayac = [0, 1, 0, 3, 0, 0, 2]`

`y = 6`

ÇIKTI

```
1: *
2:
3: ***
4:
5:
6: **
```

İŞLEMLER · KODU ADIM ADIM ÇALIŞTIR

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Tüm hücreler `false`: başlangıçta 2…20 hepsi asal adayı.

Adım 1 / 9

```
▶ int n = 20;
▶ boolean[] asalDegil = new boolean[n + 1];
3 for (int i = 2; i * i <= n; i++)
4   if (!asalDegil[i])
5     for (int k = i * i; k <= n; k += i)
6       asalDegil[k] = true;
7 for (int i = 2; i <= n; i++)
8   if (!asalDegil[i]) System.out.print(i + " ");
```

DEĞİŞKENLER

n = 20

ÇIKTI

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili



2 × 2 = 4 <= 20 → true.

Adım 2 / 9

```
1 int n = 20;
2 boolean[] asalDegil = new boolean[n + 1];
▶ for (int i = 2; i * i <= n; i++)
4   if (!asalDegil[i])
5     for (int k = i * i; k <= n; k += i)
6       asalDegil[k] = true;
7 for (int i = 2; i <= n; i++)
8   if (!asalDegil[i]) System.out.print(i + " ");
```

DEĞİŞKENLER

n = 20 i = 2

ÇIKTI

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili



2 elenmemiş → asal. Katları 2·2'den başlayarak işaretlenir: k = 4, 6, 8, 10, 12, 14, 16, 18, 20.

Adım 3 / 9

```
1 int n = 20;
2 boolean[] asalDegil = new boolean[n + 1];
3 for (int i = 2; i * i <= n; i++)
▶   if (!asalDegil[i])
▶     for (int k = i * i; k <= n; k += i)
▶       asalDegil[k] = true;
7 for (int i = 2; i <= n; i++)
8   if (!asalDegil[i]) System.out.print(i + " ");
```

DEĞİŞKENLER

`n = 20` `i = 2`

ÇIKTI

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili



3 × 3 = 9 <= 20 → true.

Adım 4 / 9

```
1 int n = 20;
2 boolean[] asalDegil = new boolean[n + 1];
▶ for (int i = 2; i * i <= n; i++)
4   if (!asalDegil[i])
5     for (int k = i * i; k <= n; k += i)
6       asalDegil[k] = true;
7 for (int i = 2; i <= n; i++)
8   if (!asalDegil[i]) System.out.print(i + " ");
```

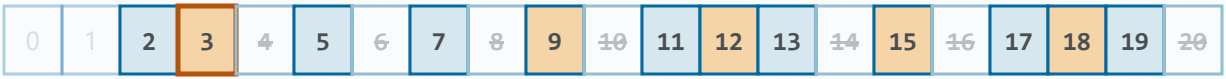
DEĞİŞKENLER

n = 20 i = 3

ÇIKTI

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili



3 elenmemiş → asal. Katları 3·3'ten başlayarak işaretlenir: k = 9, 12, 15, 18.

Adım 5 / 9

```
1 int n = 20;
2 boolean[] asalDegil = new boolean[n + 1];
3 for (int i = 2; i * i <= n; i++)
▶   if (!asalDegil[i])
▶     for (int k = i * i; k <= n; k += i)
▶       asalDegil[k] = true;
7 for (int i = 2; i <= n; i++)
8   if (!asalDegil[i]) System.out.print(i + " ");
```

DEĞİŞKENLER

`n = 20` `i = 3`

ÇIKTI

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili



4 × 4 = 16 <= 20 → true.

Adım 6 / 9

```
1 int n = 20;
2 boolean[] asalDegil = new boolean[n + 1];
▶ for (int i = 2; i * i <= n; i++)
4   if (!asalDegil[i])
5     for (int k = i * i; k <= n; k += i)
6       asalDegil[k] = true;
7 for (int i = 2; i <= n; i++)
8   if (!asalDegil[i]) System.out.print(i + " ");
```

DEĞİŞKENLER

`n = 20` `i = 4`

ÇIKTI

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili



asalDegil[4] = true → 4 zaten elenmiş, katlarına bakmaya gerek yok.

Adım 7 / 9

```
1 int n = 20;
2 boolean[] asalDegil = new boolean[n + 1];
3 for (int i = 2; i * i <= n; i++)
▶   if (!asalDegil[i])
5     for (int k = i * i; k <= n; k += i)
6       asalDegil[k] = true;
7 for (int i = 2; i <= n; i++)
8   if (!asalDegil[i]) System.out.print(i + " ");
```

DEĞİŞKENLER

`n = 20` `i = 4`

ÇIKTI

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili



5 × 5 = 25 <= 20 → **false**: √20'den büyük bölen aramaya gerek yok.

Adım 8 / 9

```
1 int n = 20;
2 boolean[] asalDegil = new boolean[n + 1];
▶ for (int i = 2; i * i <= n; i++)
4   if (!asalDegil[i])
5     for (int k = i * i; k <= n; k += i)
6       asalDegil[k] = true;
7 for (int i = 2; i <= n; i++)
8   if (!asalDegil[i]) System.out.print(i + " ");
```

DEĞİŞKENLER

`n = 20` `i = 5`

ÇIKTI

Eratosthenes Eleği (`boolean` dizi)

asalDegil[i] = true olanlar üstü çizili



Elenmemiş olanlar yazdırılır: 8 asal.

Adım 9 / 9

```
1 int n = 20;
2 boolean[] asalDegil = new boolean[n + 1];
3 for (int i = 2; i * i <= n; i++)
4   if (!asalDegil[i])
5     for (int k = i * i; k <= n; k += i)
6       asalDegil[k] = true;
▶ for (int i = 2; i <= n; i++)
▶   if (!asalDegil[i]) System.out.print(i + " ");
```

DEĞİŞKENLER

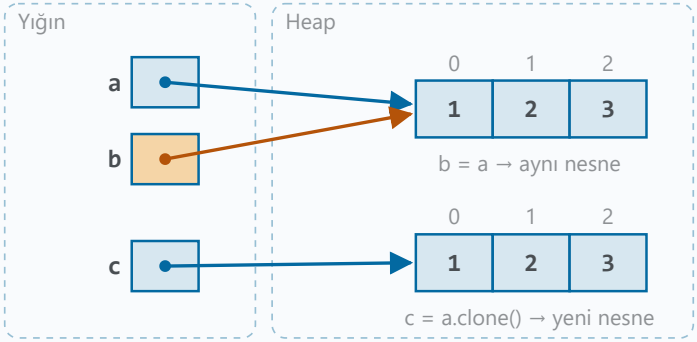
`n = 20`

ÇIKTI

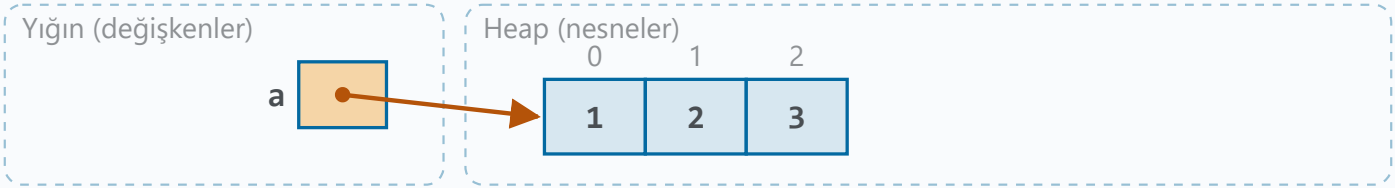
```
2 3 5 7 11 13 17 19
```

Dizi Kopyalama

- `b = a` öğeleri değil **referansı** kopyalar: iki ok aynı nesneyi gösterir.
- Bağımsız kopya için `a.clone()`, `Arrays.copyOf(a, a.length)` veya `System.arraycopy`.
- Nesne dizilerinde bu yöntemler öğe referanslarını kopyalar; nesnelerin kendisi çoğaltılmaz (sığ kopya).



Kod İzleme: Referans Kopyası ve Öğe Kopyası



Heap'te bir dizi; `a` onu gösterir.

Adım 1 / 7

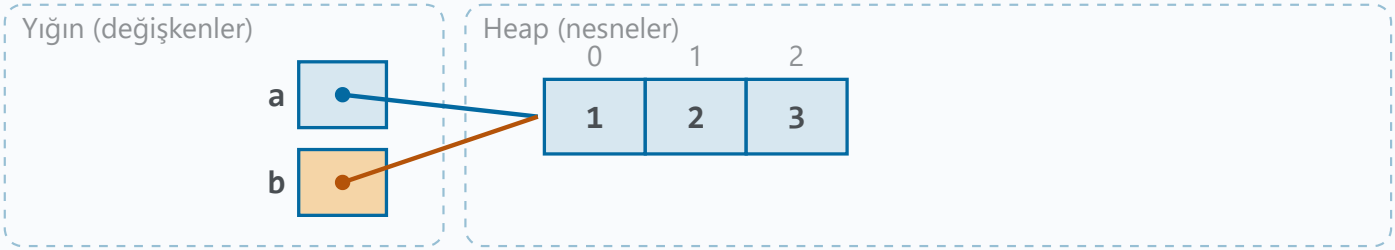
```
▶ int[] a = {1, 2, 3};
2 int[] b = a;
3 b[0] = 99;
4 System.out.println(a[0]);
5 int[] c = a.clone();
6 c[1] = 50;
7 System.out.println(a[1] + " " + c[1]);
```

DEĞİŞKENLER

`a = [1, 2, 3]`

ÇIKTI

Kod İzleme: Referans Kopyası ve Öğe Kopyası



`b = a` : yeni dizi **oluşmaz**; b'ye aynı referans yazılır → iki ok aynı nesnede.

Adım 2 / 7

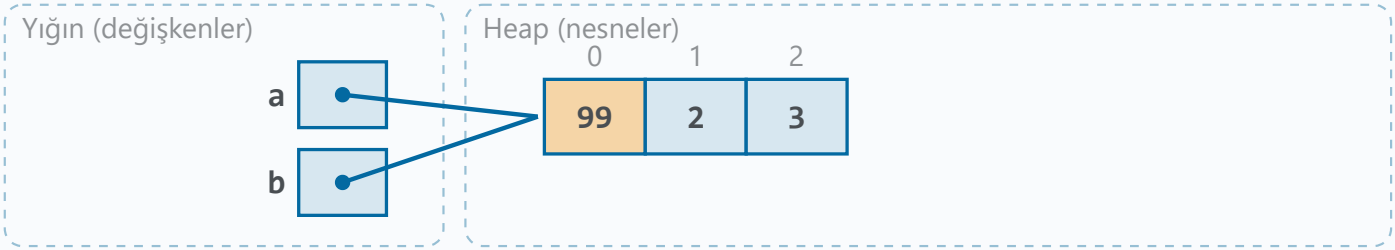
```
1 int[] a = {1, 2, 3};
▶ int[] b = a;
3 b[0] = 99;
4 System.out.println(a[0]);
5 int[] c = a.clone();
6 c[1] = 50;
7 System.out.println(a[1] + " " + c[1]);
```

DEĞİŞKENLER

`a = [1, 2, 3]` `b = [1, 2, 3]`

ÇIKTI

Kod İzleme: Referans Kopyası ve Öğe Kopyası



`b[0] = 99` : b'nin gösterdiği nesne değişti — bu, a'nın da nesnesi!

Adım 3 / 7

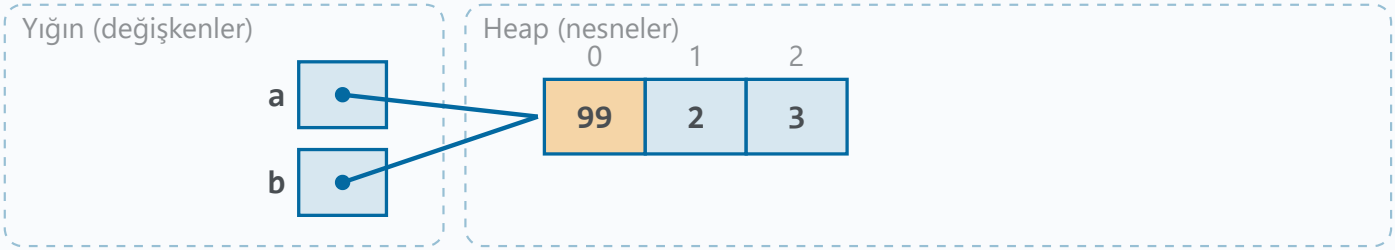
```
1 int[] a = {1, 2, 3};
2 int[] b = a;
▶ b[0] = 99;
4 System.out.println(a[0]);
5 int[] c = a.clone();
6 c[1] = 50;
7 System.out.println(a[1] + " " + c[1]);
```

DEĞİŞKENLER

`a = [99, 2, 3]` `b = [99, 2, 3]`

ÇIKTI

Kod İzleme: Referans Kopyası ve Öğe Kopyası



`a[0]` artık **99**. Referans kopyası bağımsız bir dizi değildir.

Adım 4 / 7

```
1 int[] a = {1, 2, 3};
2 int[] b = a;
3 b[0] = 99;
▶ System.out.println(a[0]);
5 int[] c = a.clone();
6 c[1] = 50;
7 System.out.println(a[1] + " " + c[1]);
```

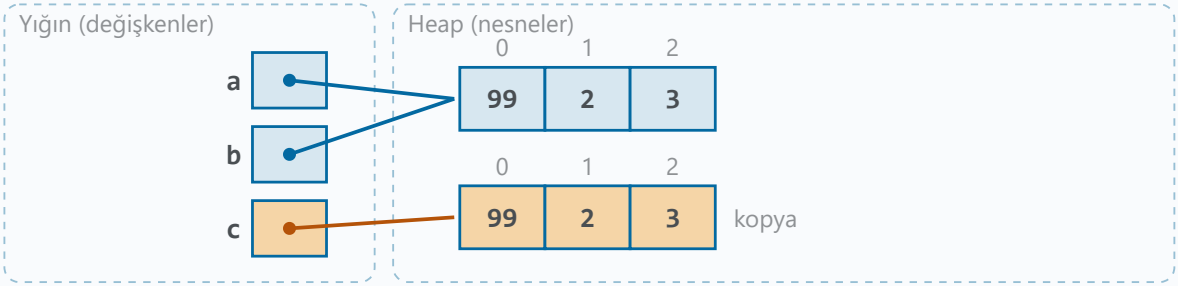
DEĞİŞKENLER

`a = [99, 2, 3]` `b = [99, 2, 3]`

ÇIKTI

```
99
```

Kod İzleme: Referans Kopyası ve Öğe Kopyası



`a.clone()` heap'te **yeni** bir dizi açar ve öğeleri tek tek kopyalar.

Adım 5 / 7

```
1 int[] a = {1, 2, 3};
2 int[] b = a;
3 b[0] = 99;
4 System.out.println(a[0]);
▶ int[] c = a.clone();
6 c[1] = 50;
7 System.out.println(a[1] + " " + c[1]);
```

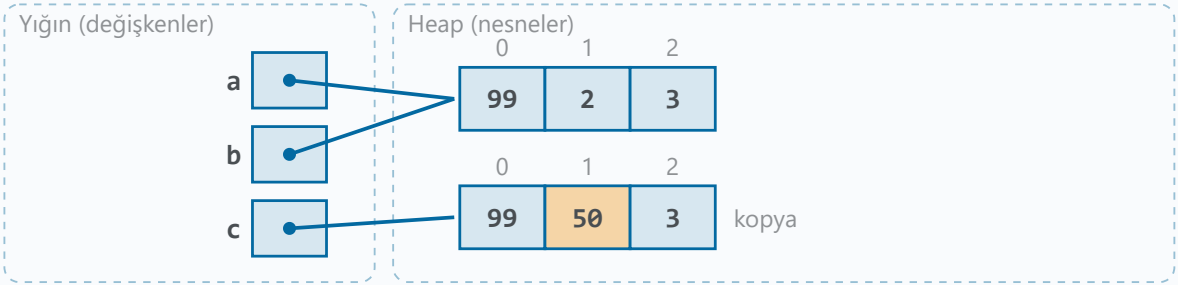
DEĞİŞKENLER

`a = [99, 2, 3]` `b = [99, 2, 3]` `c = [99, 2, 3]`

ÇIKTI

```
99
```

Kod İzleme: Referans Kopyası ve Öğe Kopyası



`c[1] = 50` yalnızca kopyayı değiştirir.

Adım 6 / 7

```
1 int[] a = {1, 2, 3};
2 int[] b = a;
3 b[0] = 99;
4 System.out.println(a[0]);
5 int[] c = a.clone();
▶ c[1] = 50;
7 System.out.println(a[1] + " " + c[1]);
```

DEĞİŞKENLER

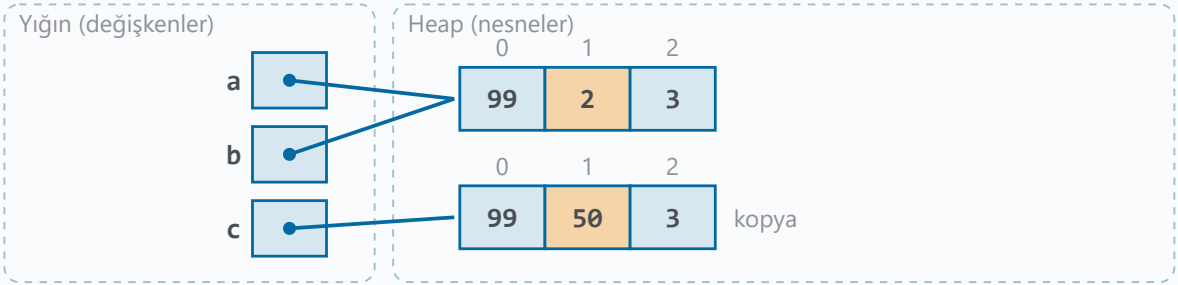
`a = [99, 2, 3]` `b = [99, 2, 3]`

`c = [99, 50, 3]`

ÇIKTI

```
99
```

Kod İzleme: Referans Kopyası ve Öğe Kopyası



`a[1]` = 2 (değişmedi), `c[1]` = 50.

Adım 7 / 7

```
1 int[] a = {1, 2, 3};
2 int[] b = a;
3 b[0] = 99;
4 System.out.println(a[0]);
5 int[] c = a.clone();
6 c[1] = 50;
▶ System.out.println(a[1] + " " + c[1]);
```

DEĞİŞKENLER

`a = [99, 2, 3]` `b = [99, 2, 3]` `c = [99, 50, 3]`

ÇIKTI

```
99
2 50
```

İŞLEMLER

Arrays Yardımcıları

```
import java.util.Arrays;
int[] a = {5, 2, 9}, r = {10, 20, 30, 40, 50}, p = {1, 2, 3}, q = {1, 2, 3}, f = new int[4];
int[][] m = {{1, 2}, {3, 4}};
```

Çağrı	Sonuç
`Arrays.toString(a)`	`[5, 2, 9]`
`Arrays.deepToString(m)`	`[[1, 2], [3, 4]]`
`Arrays.toString(m)`	`[[I@…` (adres)
`Arrays.fill(f, 7)` → f	`[7, 7, 7, 7]`
`p == q` / `p.equals(q)`	`false` / `false`
`Arrays.equals(p, q)`	`true`

Kopya (Arrays.)	Sonuç
`copyOf(a, 5)`	`[5, 2, 9, 0, 0]`
`copyOf(a, 2)`	`[5, 2]`
`copyOfRange(r, 1, 4)`	`[20, 30, 40]`

- `==` ve `equals` referansı karşılaştırır; içerik için `Arrays.equals` (2B için `deepEquals`).
- `copyOf` uzunluk büyükse sıfırla doldurur, küçükse keser; `copyOfRange`'te bitiş indisi dahil değildir.

Doğrusal Arama: `ara(d, 6)`

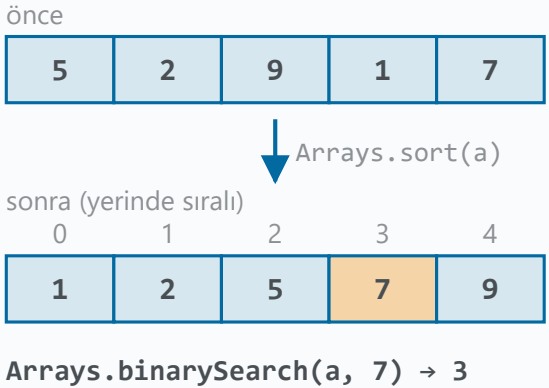
```
static int ara(int[] dizi, int hedef) {
    for (int i = 0; i < dizi.length; i++)
        if (dizi[i] == hedef) return i;
    return -1;
}
// main içinde:
int[] d = {8, 3, 6, 2};
System.out.println(ara(d, 6));
```

i	dizi[i]	== 6 ?
0	8	false
1	3	false
2	6	**true** → `return 2`

- Çıktı: **2**. İlk eşleşmede döngü erken biter.
- `dizi` parametresi `d` ile **aynı diziyi** gösterir (referans aktarılır); kopya oluşmaz.
- Hedef yoksa n karşılaştırmadan sonra −1 döner → **O(n)**.

Sıralama ve İkili Arama

- `Arrays.sort(a)` diziyi küçükten büyüğe **yerinde** sıralar (özgün sıra kaybolur).
- `Arrays.binarySearch(a, x)` yalnızca **sıralı** dizide doğru çalışır; bulursa indisi döndürür; bulamazsa `-(eklenme noktası) - 1` gibi negatif bir değer döndürür.
- Her adımda aralığı yarıya indirdiği için doğrusal aramadan çok daha az karşılaştırma yapar.
- İkisi için de `import java.util.Arrays;` gerekir.

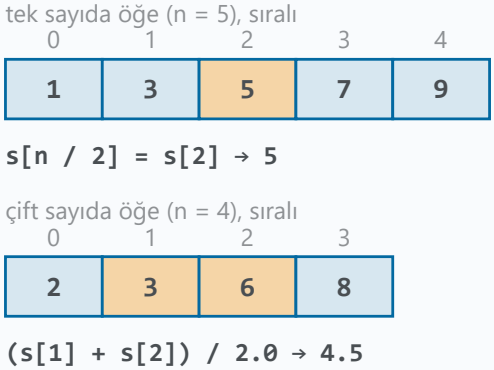


`Arrays.binarySearch(a, 7) → 3`

Ortanca (Medyan)

- **Ortanca**: sıralanmış dizinin **ortasındaki** değer; öğe sayısı çiftse ortadaki ikisinin ortalaması.
- Uç değerden etkilenmez: {1, 2, 3, 4, 100} için ortalama 22.0, ortanca 3.
- `Arrays.sort` diziyi yerinde değiştirir; bu yüzden **kopya** sıralanır.

```
int[] a = {7, 3, 9, 1, 5};
int[] s = Arrays.copyOf(a, a.length);
Arrays.sort(s);    // a bozulmaz
int n = s.length;
double m = (n % 2 == 1) ? s[n / 2]
    : (s[n/2 - 1] + s[n/2]) / 2.0;
System.out.println(m);   // 5.0
```



Kabarcık Sıralaması (Bubble Sort)

```java
int[] a = {5, 1, 4, 2};
int n = a.length;
for (int gecis = 0; gecis < n - 1; gecis++) {
  for (int j = 0; j < n - 1 - gecis; j++) {
    if (a[j] > a[j + 1]) {
      int gecici = a[j];
      a[j] = a[j + 1];
      a[j + 1] = gecici;
    }
  }
}
```

geçiş	geçiş sonu dizi	karş.	takas
1	[1, 4, 2, 5]	3	3
2	[1, 2, 4, 5]	2	1
3	[1, 2, 4, 5]	1	0

- Her geçişte kalanların en büyüğü sona yerleşir; iç döngü her geçişte bir kısalır.
- Toplam 3 + 2 + 1 = 6 karşılaştırma → **$O(n^2)$**.
- 3. geçişte hiç takas yok: dizi 2. geçişten sonra sıralıydı.

Eklemeli Sıralama (Insertion Sort)

```java
int[] a = {4, 3, 5, 1};
for (int i = 1; i < a.length; i++) {
  int anahtar = a[i];
  int j = i - 1;
  while (j >= 0 && a[j] > anahtar) {
    a[j + 1] = a[j];
    j--;
  }
  a[j + 1] = anahtar;
}
```

i	anahtar	sağa kayanlar	tur sonu dizi
1	3	4	[3, 4, 5, 1]
2	5	—	[3, 4, 5, 1]
3	1	5, 4, 3	[1, 3, 4, 5]

- Sol bölüm hep sıralıdır; anahtardan büyük öğeler birer sağa kayar, anahtar açılan yere konur.
- Neredeyse sıralı dizide çok hızlı; en kötü durumda (ters sıralı) **$O(n^2)$**.

ARAMA

İkili Arama: `hedef = 23`

```
int[] a = {2, 5, 8, 12, 16, 23, 38, 56};
int hedef = 23, lo = 0, hi = a.length - 1;
int sonuc = -1;
while (lo <= hi) {
  int mid = (lo + hi) / 2;
  if (a[mid] == hedef) { sonuc = mid; break; }
  if (a[mid] < hedef) lo = mid + 1;
  else hi = mid - 1;
}
System.out.println(sonuc);
```

tur	lo	hi	mid	a[mid]	karar
1	0	7	3	12	12 < 23 → lo = 4
2	4	7	5	23	bulundu: sonuc = 5

- Çıktı: **5**. Dizi sıralı olmalıdır.
- 2 turda bulundu; doğrusal arama 6 karşılaştırma yapardı → **O(log n)**.

Başarısız Arama: `Arrays.binarySearch(a, 5)` → -3

```
static int ara(int[] a, int x) {
  int lo = 0, hi = a.length - 1;
  while (lo <= hi) {
    int mid = (lo + hi) / 2;
    if (a[mid] == x) return mid;
    if (a[mid] < x) lo = mid + 1; else hi = mid - 1;
  }
  return -(lo + 1);
}
System.out.println(ara(new int[]{2, 4, 6, 8}, 5));
```

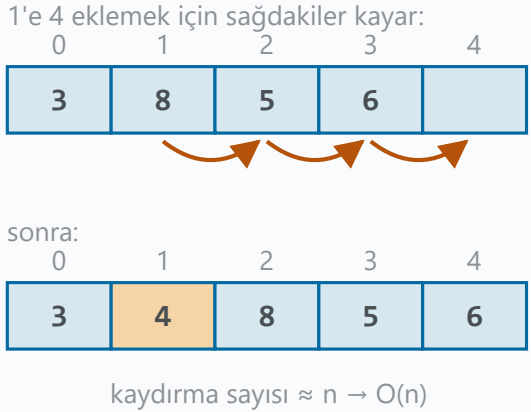
tur	lo	hi	mid	a[mid]	karar
1	0	3	1	4	4 < 5 → lo = 2
2	2	3	2	6	6 > 5 → hi = 1
—	2	1			lo > hi: döngü biter

- Sonuç `-(lo + 1)` = **-3**; negatif değer "yok" demektir.
- Ekleme noktası = -(-3) - 1 = 2: 5 dizide 2 numaralı konuma girerdi. `Arrays.binarySearch` da aynı değeri döndürür.

ANALİZ

Temel Karmaşıklıklar

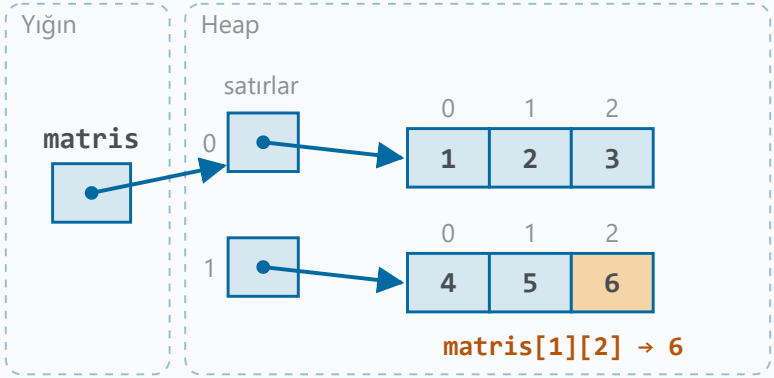
İşlem	Zaman
İndisle erişim	O(1)
Doğrusal arama	O(n)
Sona kadar gezinme	O(n)
Ortaya ekleme / silme	O(n)
İki iç içe tam gezinme	$O(n^2)$



kaydırma sayısı ≈ n → O(n)

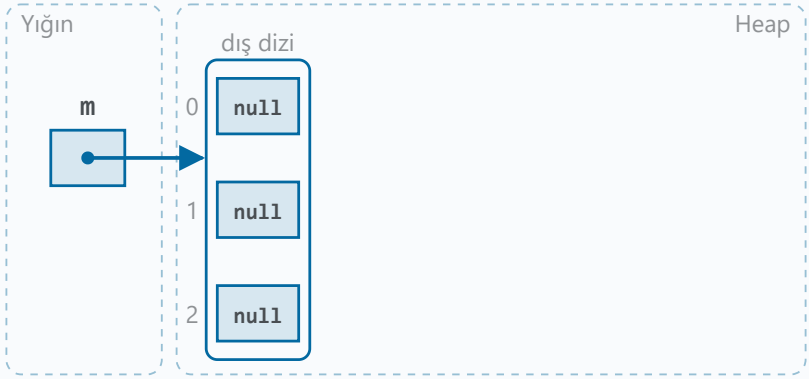
İki Boyutlu Dizi

```
int[][] matris = {
    {1, 2, 3},
    {4, 5, 6}
};
int deger = matris[1][2]; // 6
```



- İlk indis **satır**, ikinci indis **sütun**.
- Java'da çok boyutlu dizi, dizilerin dizisidir: dış dizinin her hücresi bir satır dizisini gösterir; satırlar farklı uzunlukta olabilir (düzensiz dizi).
- `matris.length` = satır sayısı, `matris[0].length` = sütun sayısı.

Düzensiz (Jagged) Dizi: `new int[3][]`



`new int[3][]` yalnızca **dış diziyi** (3 hücre) oluşturur. Satırlar henüz yok: üç hücre de **null**.

Adım 1 / 7

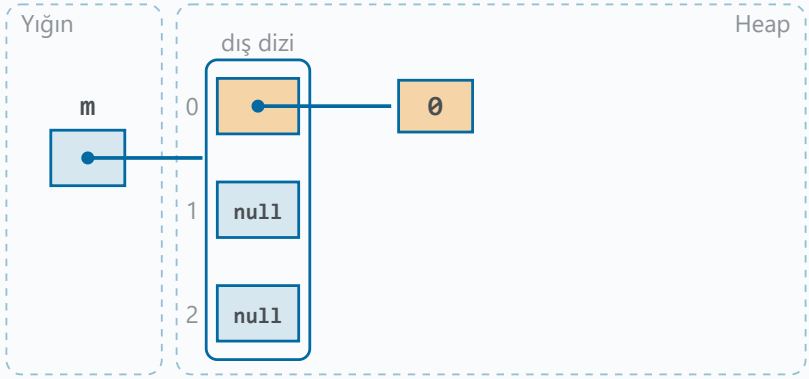
```
▶ int[][] m = new int[3][];
2 m[0] = new int[1];
3 m[1] = new int[2];
4 m[2] = new int[3];
5 for (int i = 0; i < m.length; i++)
6     System.out.print(m[i].length + " ");
```

DEĞİŞKENLER

`m = →int[3][]`

ÇIKTI

Düzensiz (Jagged) Dizi: `new int[3][]`



m[0] = new int[1]: ilk hücre 1 öğeli bir satır gösterir. m[1] ve m[2] hâlâ null.

Adım 2 / 7

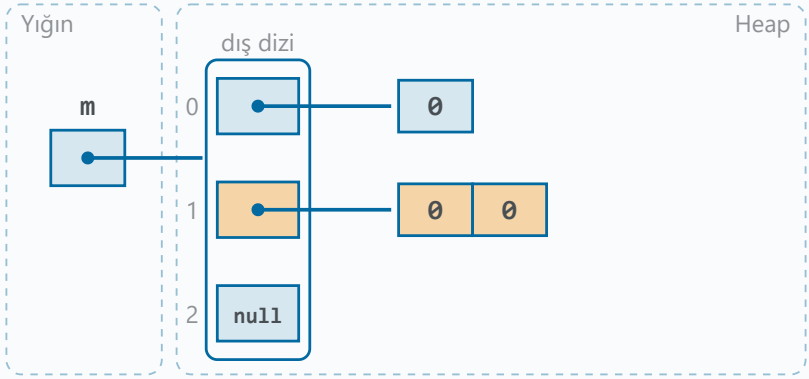
```
1 int[][] m = new int[3][];
▶ m[0] = new int[1];
3 m[1] = new int[2];
4 m[2] = new int[3];
5 for (int i = 0; i < m.length; i++)
6     System.out.print(m[i].length + " ");
```

DEĞİŞKENLER

`m = →int[3][]`

ÇIKTI

Düzensiz (Jagged) Dizi: `new int[3][]`



m[1] = new int[2]: ikinci satır 2 öğeli.

Adım 3 / 7

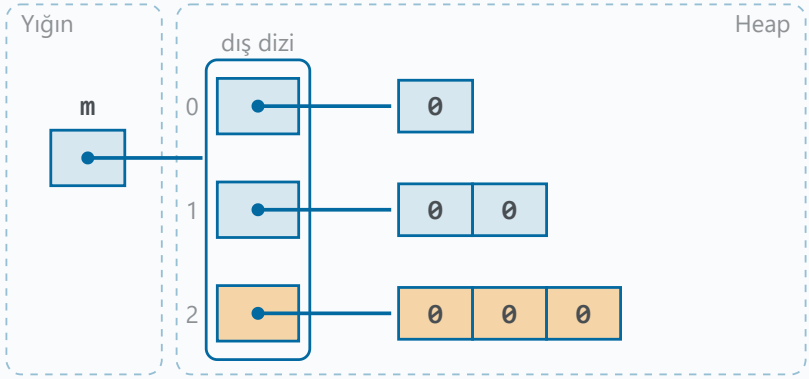
```
1 int[][] m = new int[3][];
2 m[0] = new int[1];
▶ m[1] = new int[2];
4 m[2] = new int[3];
5 for (int i = 0; i < m.length; i++)
6     System.out.print(m[i].length + " ");
```

DEĞİŞKENLER

`m = →int[3][]`

ÇIKTI

Düzensiz (Jagged) Dizi: `new int[3][]`



m[2] = new int[3]: satırlar farklı uzunlukta, yani **düzensiz (jagged) dizi**. Her satır ayrı bir nesnedir.

Adım 4 / 7

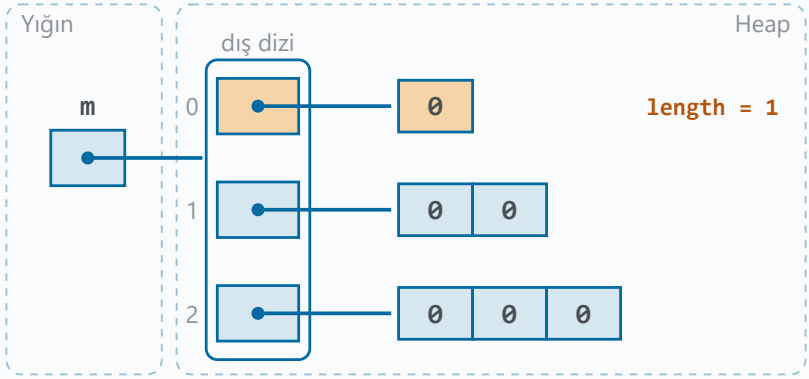
```
1 int[][] m = new int[3][];
2 m[0] = new int[1];
3 m[1] = new int[2];
▶ m[2] = new int[3];
5 for (int i = 0; i < m.length; i++)
6     System.out.print(m[i].length + " ");
```

DEĞİŞKENLER

`m = →int[3][]`

ÇIKTI

Düzensiz (Jagged) Dizi: `new int[3][]`



i = 0: `m[0].length` = 1 yazılır.

Adım 5 / 7

```
1 int[][] m = new int[3][];
2 m[0] = new int[1];
3 m[1] = new int[2];
4 m[2] = new int[3];
▶ for (int i = 0; i < m.length; i++)
▶     System.out.print(m[i].length + " ");
```

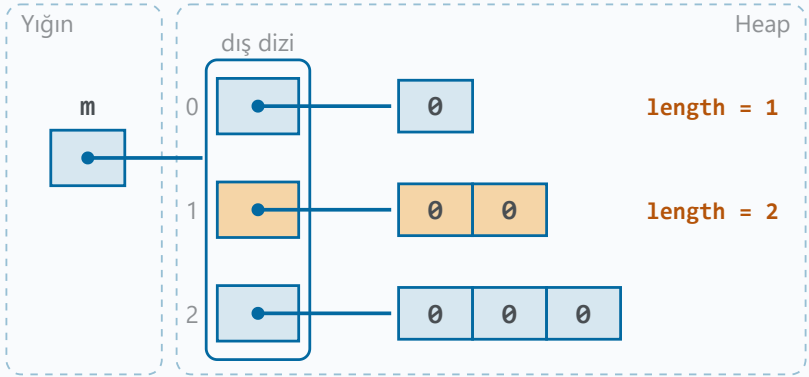
DEĞİŞKENLER

`m = →int[3][]` `i = 0`

ÇIKTI

```
1
```

Düzensiz (Jagged) Dizi: `new int[3][]`



i = 1: `m[1].length` = 2 yazılır.

Adım 6 / 7

```
1 int[][] m = new int[3][];
2 m[0] = new int[1];
3 m[1] = new int[2];
4 m[2] = new int[3];
▶ for (int i = 0; i < m.length; i++)
▶     System.out.print(m[i].length + " ");
```

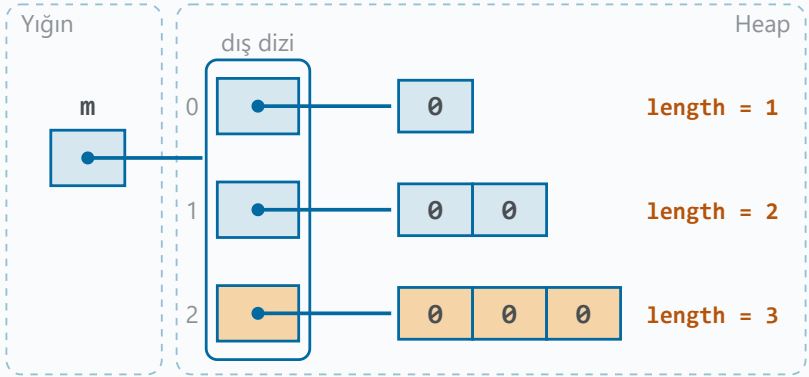
DEĞİŞKENLER

`m = →int[3][]` `i = 1`

ÇIKTI

```
1 2
```

Düzensiz (Jagged) Dizi: `new int[3][]`



i = 2: `m[2].length` = 3. Çıktı: **1 2 3**. Satır null iken `m[i].length` yazılsaydı NullPointerException olurdu.

Adım 7 / 7

```
1 int[][] m = new int[3][];
2 m[0] = new int[1];
3 m[1] = new int[2];
4 m[2] = new int[3];
▶ for (int i = 0; i < m.length; i++)
▶     System.out.print(m[i].length + " ");
```

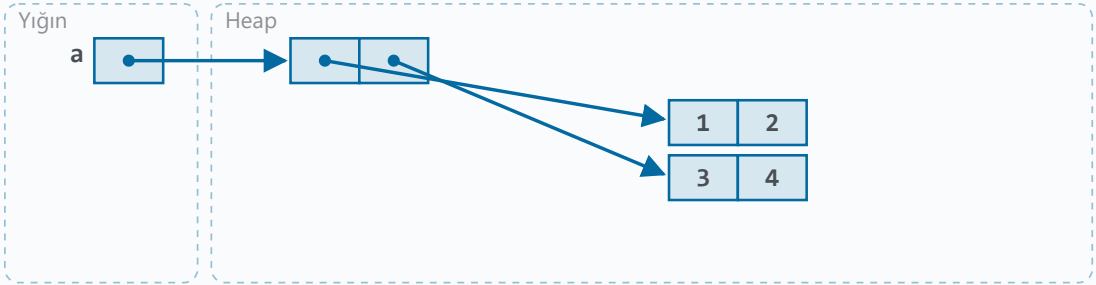
DEĞİŞKENLER

`m = →int[3][]` `i = 2`

ÇIKTI

```
1 2 3
```

2B Dizide Sığ Kopya Tuzağı



`a`, iki satır dizisini gösteren bir **dış dizidir**.

Adım 1 / 6

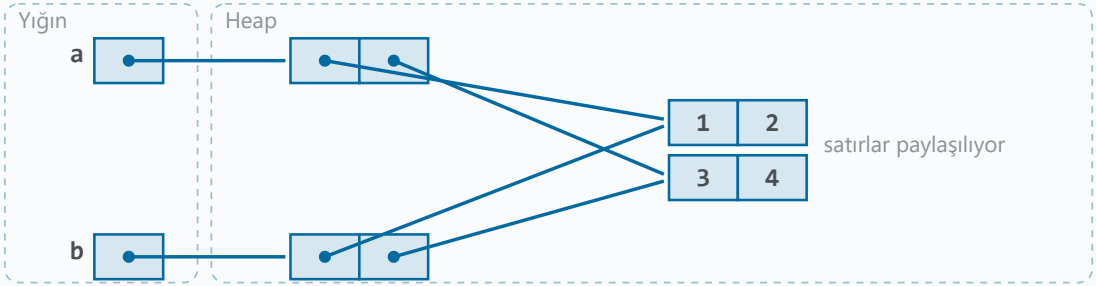
```
▶ int[][] a = {{1, 2}, {3, 4}};
2 int[][] b = a.clone();
3 b[0][0] = 99;
4 b[1] = new int[]{7, 8};
5 System.out.println(Arrays.deepToString(a));
6 System.out.println(Arrays.deepToString(b));
```

DEĞİŞKENLER

`a = [[1, 2], [3, 4]]`

ÇIKTI

2B Dizide Sığ Kopya Tuzağı



`a.clone()` yalnızca dış diziyi kopyalar: iki dış dizi, **aynı satır nesnelerini** gösterir (sığ kopya).

Adım 2 / 6

```
1 int[][] a = {{1, 2}, {3, 4}};
▶ int[][] b = a.clone();
3 b[0][0] = 99;
4 b[1] = new int[]{7, 8};
5 System.out.println(Arrays.deepToString(a));
6 System.out.println(Arrays.deepToString(b));
```

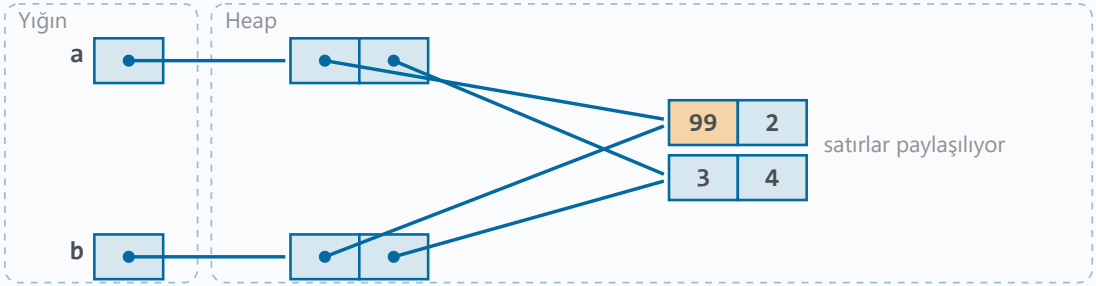
DEĞİŞKENLER

`a = [[1, 2], [3, 4]]`

`b = [[1, 2], [3, 4]]`

ÇIKTI

2B Dizide Sığ Kopya Tuzağı



`b[0][0] = 99` : b[0] ve a[0] aynı satırdır → **a[0][0] da 99** oldu.

Adım 3 / 6

```
1 int[][] a = {{1, 2}, {3, 4}};
2 int[][] b = a.clone();
▶ b[0][0] = 99;
4 b[1] = new int[]{7, 8};
5 System.out.println(Arrays.deepToString(a));
6 System.out.println(Arrays.deepToString(b));
```

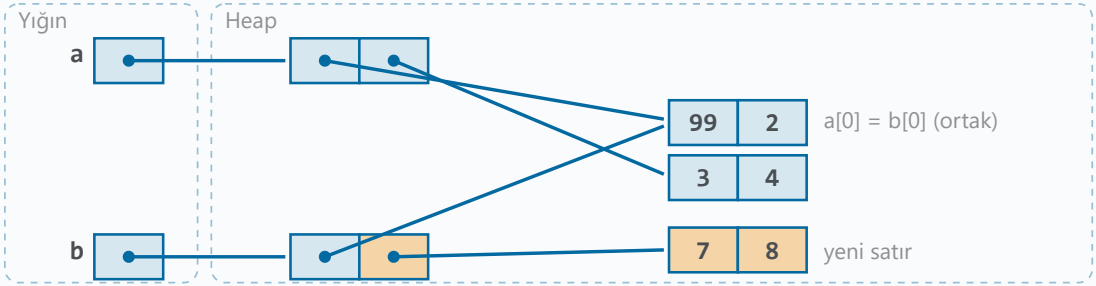
DEĞİŞKENLER

`a = [[99, 2], [3, 4]]`

`b = [[99, 2], [3, 4]]`

ÇIKTI

2B Dizide Sığ Kopya Tuzağı



`b[1] = new int[]{7, 8}` yalnızca b'nin hücresini yeni satıra bağlar; a[1] değişmez.

Adım 4 / 6

```
1 int[][] a = {{1, 2}, {3, 4}};
2 int[][] b = a.clone();
3 b[0][0] = 99;
▶ b[1] = new int[]{7, 8};
5 System.out.println(Arrays.deepToString(a));
6 System.out.println(Arrays.deepToString(b));
```

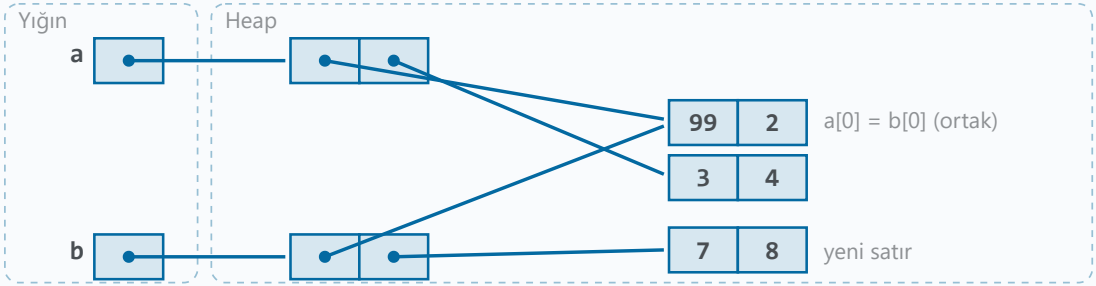
DEĞİŞKENLER

a = `[[99, 2], [3, 4]]`

b = `[[99, 2], [7, 8]]`

ÇIKTI

2B Dizide Sığ Kopya Tuzağı



a: ortak satır 0 değişti (99), satır 1 hâlâ {3, 4}.

Adım 5 / 6

```
1 int[][] a = {{1, 2}, {3, 4}};
2 int[][] b = a.clone();
3 b[0][0] = 99;
4 b[1] = new int[]{7, 8};
▶ System.out.println(Arrays.deepToString(a));
6 System.out.println(Arrays.deepToString(b));
```

DEĞİŞKENLER

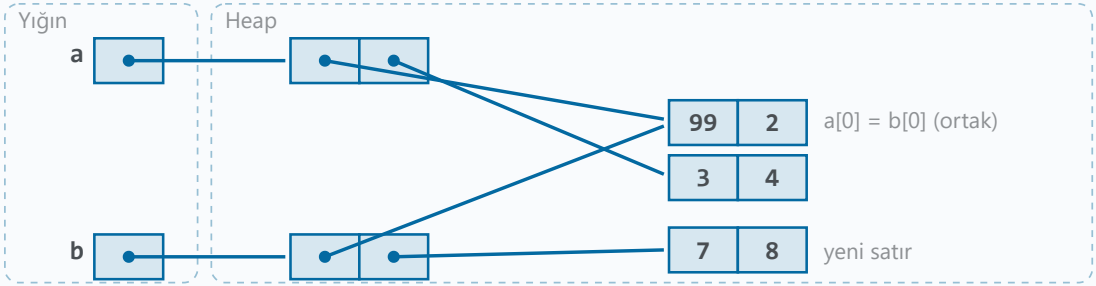
a = **[[99, 2], [3, 4]]**

b = **[[99, 2], [7, 8]]**

ÇIKTI

```
[[99, 2], [3, 4]]
```

2B Dizide Sığ Kopya Tuzağı



Gerçek kopya için her satır ayrıca kopyalanır: `b[i] = a[i].clone()`.

Adım 6 / 6

```
1 int[][] a = {{1, 2}, {3, 4}};
2 int[][] b = a.clone();
3 b[0][0] = 99;
4 b[1] = new int[]{7, 8};
5 System.out.println(Arrays.deepToString(a));
▶ System.out.println(Arrays.deepToString(b));
```

DEĞİŞKENLER

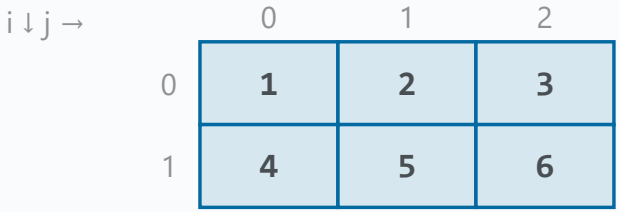
a = **[[99, 2], [3, 4]]**

b = **[[99, 2], [7, 8]]**

ÇIKTI

```
[[99, 2], [3, 4]]
[[99, 2], [7, 8]]
```

Matriste Satır Satır Gezinme



2 satır × 3 sütunluk matris; `matris.length = 2`.

Adım 1 / 14

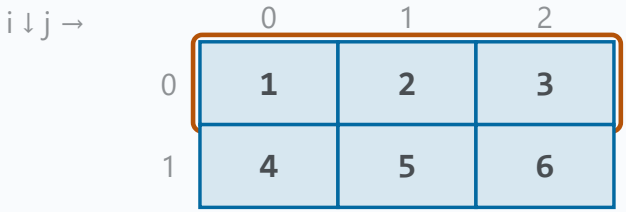
```
▶ int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
4     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

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

ÇIKTI

Matriste Satır Satır Gezinme



Dış döngü: **0 < 2** → **true** → 0. satır işlenecek.

Adım 2 / 14

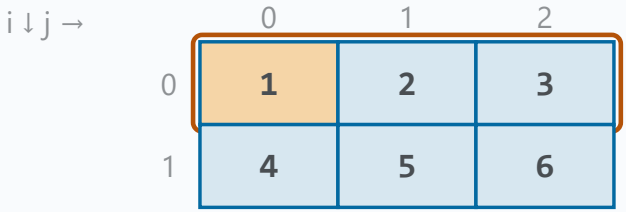
```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
▶ for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
4     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

matris	[[1, 2, 3], [4, 5, 6]]
i	0

ÇIKTI

Matriste Satır Satır Gezinme



İç döngü: **0 < 3** → **true**; `matris[0][0] = 1` yazılır (print: satır atlamaz).

Adım 3 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
▶     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

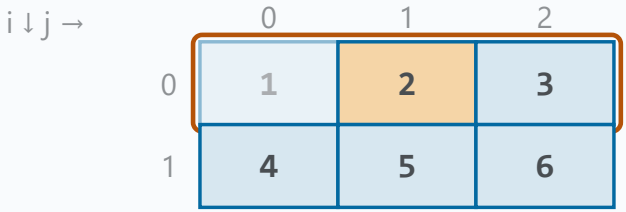
`matris = [[1, 2, 3], [4, 5, 6]]`

`i = 0` `j = 0`

ÇIKTI

```
1
```

Matriste Satır Satır Gezinme



İç döngü: **1 < 3** → **true**; `matris[0][1] = 2` yazılır (print: satır atlamaz).

Adım 4 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
▶     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

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

`i = 0` `j = 1`

ÇIKTI

```
1 2
```

Matriste Satır Satır Gezinme



İç döngü: **2 < 3** → **true**; `matris[0][2] = 3` yazılır (print: satır atlamaz).

Adım 5 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
▶     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

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

`i = 0` `j = 2`

ÇIKTI

```
1 2 3
```

Matriste Satır Satır Gezinme



3 < 3 → **false**: satır bitti, iç döngüden çıkılır.

Adım 6 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
▶   for (int j = 0; j < matris[i].length; j++) {
4     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

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

`i = 0` `j = 3`

ÇIKTI

```
1 2 3
```

Matriste Satır Satır Gezinme



`println()` alt satıra geçer; ardından `i++`.

Adım 7 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
4     System.out.print(matris[i][j] + " ");
5   }
▶   System.out.println();
7 }
```

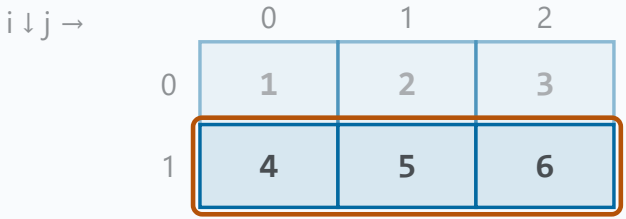
DEĞİŞKENLER

matris	[[1, 2, 3], [4, 5, 6]]
i	0

ÇIKTI

```
1 2 3
```

Matriste Satır Satır Gezinme



Dış döngü: **1 < 2** → **true** → 1. satır işlenecek.

Adım 8 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
▶ for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
4     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

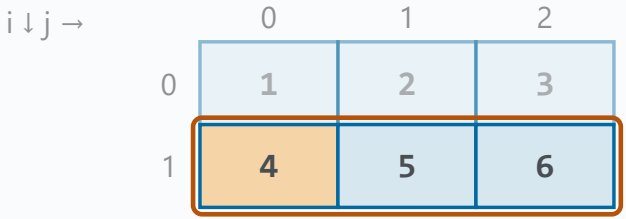
`matris = [[1, 2, 3], [4, 5, 6]]`

`i = 1`

ÇIKTI

```
1 2 3
```

Matriste Satır Satır Gezinme



İç döngü: **0 < 3 → true**; `matris[1][0] = 4` yazılır (print: satır atlamaz).

Adım 9 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
▶     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

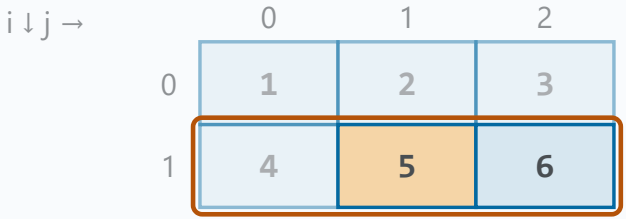
`matris = [[1, 2, 3], [4, 5, 6]]`

`i = 1` `j = 0`

ÇIKTI

```
1 2 3
4
```

Matriste Satır Satır Gezinme



İç döngü: **1 < 3 → true**; `matris[1][1] = 5` yazılır (print: satır atlamaz).

Adım 10 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
▶     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

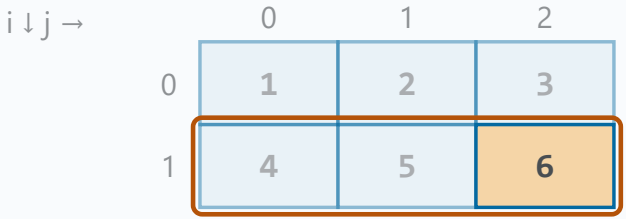
`matris = [[1, 2, 3], [4, 5, 6]]`

`i = 1` `j = 1`

ÇIKTI

```
1 2 3
4 5
```

Matriste Satır Satır Gezinme



İç döngü: **2 < 3 → true**; `matris[1][2] = 6` yazılır (print: satır atlamaz).

Adım 11 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
▶     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

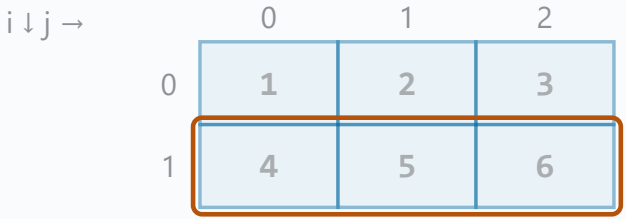
`matris = [[1, 2, 3], [4, 5, 6]]`

`i = 1` `j = 2`

ÇIKTI

```
1 2 3
4 5 6
```

Matriste Satır Satır Gezinme



3 < 3 → **false**: satır bitti, iç döngüden çıkılır.

Adım 12 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
▶   for (int j = 0; j < matris[i].length; j++) {
4     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

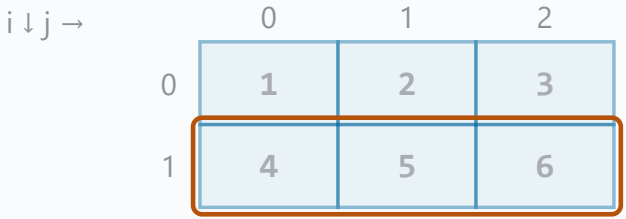
`matris = [[1, 2, 3], [4, 5, 6]]`

`i = 1` `j = 3`

ÇIKTI

```
1 2 3
4 5 6
```

Matriste Satır Satır Gezinme



`println()` alt satıra geçer; ardından `i++`.

Adım 13 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
4     System.out.print(matris[i][j] + " ");
5   }
▶   System.out.println();
7 }
```

DEĞİŞKENLER

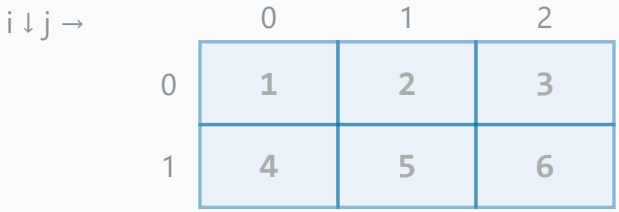
`matris` = `[[1, 2, 3], [4, 5, 6]]`

`i` = **1**

ÇIKTI

```
1 2 3
4 5 6
```

Matriste Satır Satır Gezinme



Dış döngü: **2 < 2** → **false** → bitti. Toplam 2 × 3 = 6 hücre gezildi.

Adım 14 / 14

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
▶ for (int i = 0; i < matris.length; i++) {
3   for (int j = 0; j < matris[i].length; j++) {
4     System.out.print(matris[i][j] + " ");
5   }
6   System.out.println();
7 }
```

DEĞİŞKENLER

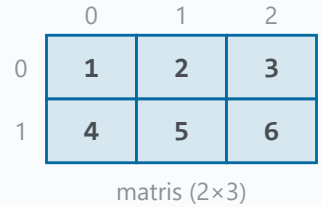
`matris = [[1, 2, 3], [4, 5, 6]]`

`i = 2`

ÇIKTI

```
1 2 3
4 5 6
```

Matris Transpozu



matris (2×3)

Kaynak matris 2 × 3.

Adım 1 / 11

```
▶ int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
7         t[j][i] = matris[i][j];
```

DEĞİŞKENLER

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

Matris Transpozu

	0	1	2
0	1	2	3
1	4	5	6

matris (2×3)

`matris.length` = satır sayısı = **2**.

Adım 2 / 11

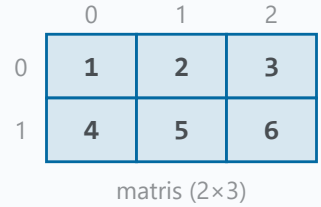
```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
▶ int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
7         t[j][i] = matris[i][j];
```

DEĞİŞKENLER

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

`satirSayisi = 2`

Matris Transpozu



`matris[0].length` = ilk satırın uzunluğu = **3**.

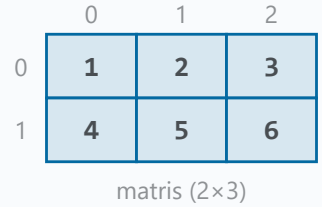
Adım 3 / 11

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
▶ int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
7         t[j][i] = matris[i][j];
```

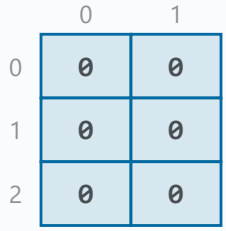
DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`

Matris Transpozu



matris (2×3)



t (3×2)

Hedef `t` boyutları ters: **3 × 2**, hücreler 0.

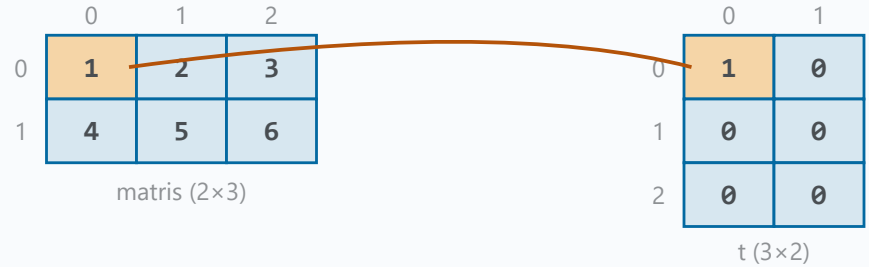
Adım 4 / 11

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
▶ int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
7         t[j][i] = matris[i][j];
```

DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`
- `t = [[0, 0], [0, 0], [0, 0]]`

Matris Transpozu



`t[0][0] = matris[0][0]` = **1**: satır ↔ sütun yer değiştirir.

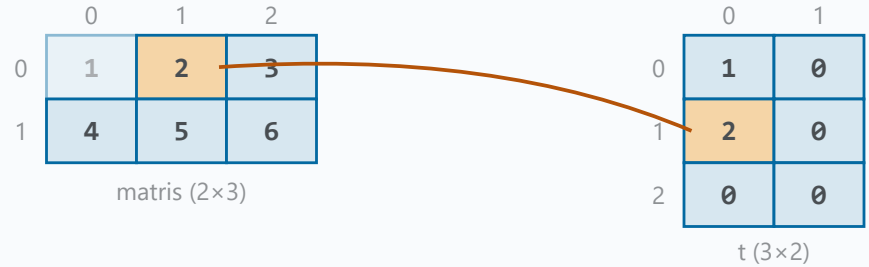
Adım 5 / 11

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
▶         t[j][i] = matris[i][j];
```

DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`
- `i = 0`
- `j = 0`
- `t = [[1, 0], [0, 0], [0, 0]]`

Matris Transpozu



`t[1][0] = matris[0][1]` = **2**: satır ↔ sütun yer değiştirir.

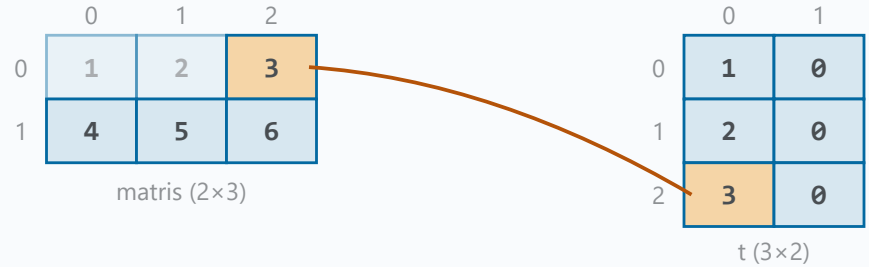
Adım 6 / 11

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
▶         t[j][i] = matris[i][j];
```

DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`
- `i = 0`
- `j = 1`
- `t = [[1, 0], [2, 0], [0, 0]]`

Matris Transpozu



`t[2][0] = matris[0][2]` = **3**: satır ↔ sütun yer değiştirir.

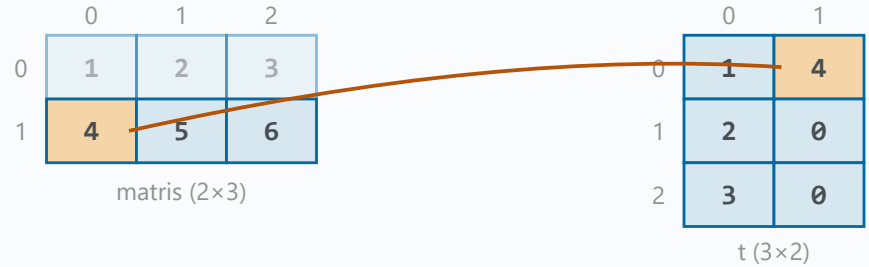
Adım 7 / 11

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
▶         t[j][i] = matris[i][j];
```

DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`
- `i = 0`
- `j = 2`
- `t = [[1, 0], [2, 0], [3, 0]]`

Matris Transpozu



`t[0][1] = matris[1][0]` = **4**: satır ↔ sütun yer değiştirir.

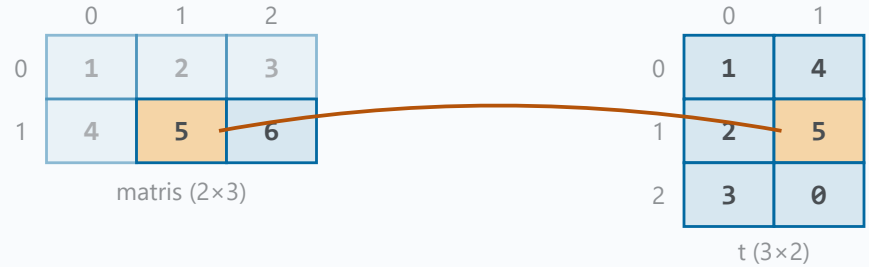
Adım 8 / 11

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
▶         t[j][i] = matris[i][j];
```

DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`
- `i = 1`
- `j = 0`
- `t = [[1, 4], [2, 0], [3, 0]]`

Matris Transpozu



`t[1][1] = matris[1][1]` = **5**: satır ↔ sütun yer değiştirir.

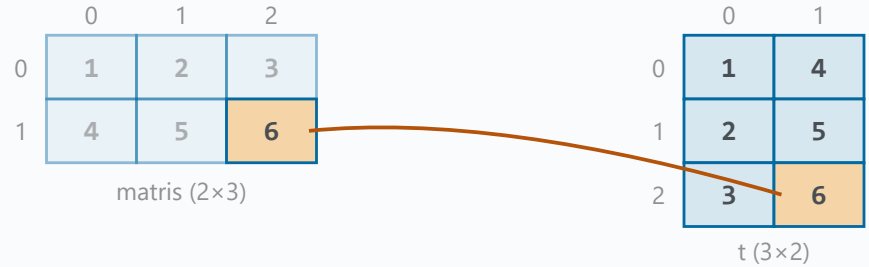
Adım 9 / 11

```
1  int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2  int satirSayisi = matris.length;
3  int sutunSayisi = matris[0].length;
4  int[][] t = new int[sutunSayisi][satirSayisi];
5  for (int i = 0; i < satirSayisi; i++)
6      for (int j = 0; j < sutunSayisi; j++)
▶          t[j][i] = matris[i][j];
```

DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`
- `i = 1`
- `j = 1`
- `t = [[1, 4], [2, 5], [3, 0]]`

Matris Transpozu



`t[2][1] = matris[1][2]` = **6**: satır ↔ sütun yer değiştirir.

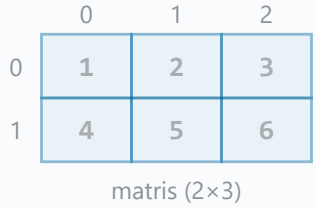
Adım 10 / 11

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
5 for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
▶         t[j][i] = matris[i][j];
```

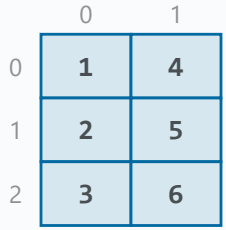
DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`
- `i = 1`
- `j = 2`
- `t = [[1, 4], [2, 5], [3, 6]]`

Matris Transpozu



matris (2×3)



t (3×2)

Döngüler bitti: matrisin **satırları** t'nin **sütunları** oldu.

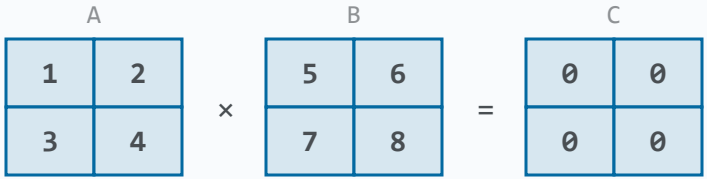
Adım 11 / 11

```
1 int[][] matris = {{1, 2, 3}, {4, 5, 6}};
2 int satirSayisi = matris.length;
3 int sutunSayisi = matris[0].length;
4 int[][] t = new int[sutunSayisi][satirSayisi];
▶ for (int i = 0; i < satirSayisi; i++)
6     for (int j = 0; j < sutunSayisi; j++)
7         t[j][i] = matris[i][j];
```

DEĞİŞKENLER

- `matris = [[1, 2, 3], [4, 5, 6]]`
- `satirSayisi = 2`
- `sutunSayisi = 3`
- `t = [[1, 4], [2, 5], [3, 6]]`

Matris Çarpımı (2×2)



C[i][j] = A'nın **i. satırı** ile B'nin **j. sütununun** çarpımlarının toplamı. C sıfırlarla başlar.

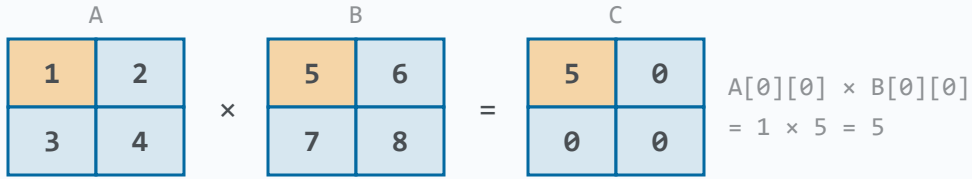
Adım 1 / 10

```
▶ int[][] A = {{1, 2}, {3, 4}};
▶ int[][] B = {{5, 6}, {7, 8}};
▶ int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
7       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

C = `[[0, 0], [0, 0]]`

Matris Çarpımı (2×2)



C[0][0] += 1 × 5 → 0 + 5 = **5**

Adım 2 / 10

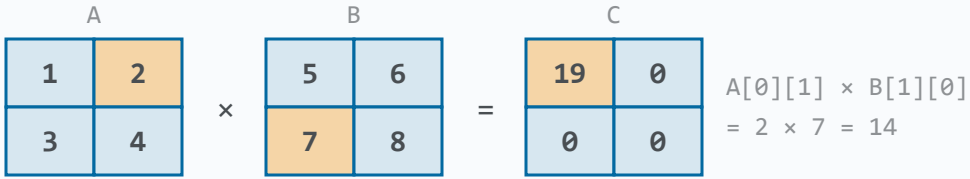
```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
▶       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

`i = 0` `j = 0` `k = 0`

`C = [[5, 0], [0, 0]]`

Matris Çarpımı (2×2)



C[0][0] += 2 × 7 → 5 + 14 = **19** (hücre tamam)

Adım 3 / 10

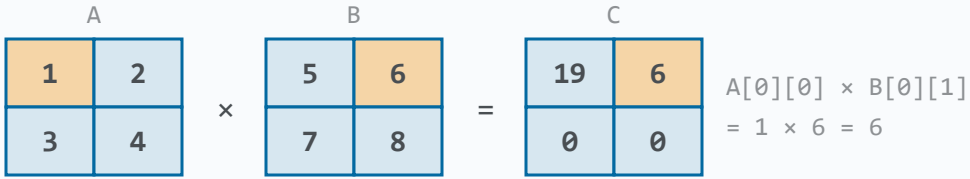
```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
▶       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

`i = 0` `j = 0` `k = 1`

`C = [[19, 0], [0, 0]]`

Matris Çarpımı (2×2)



C[0][1] += 1 × 6 → 0 + 6 = **6**

Adım 4 / 10

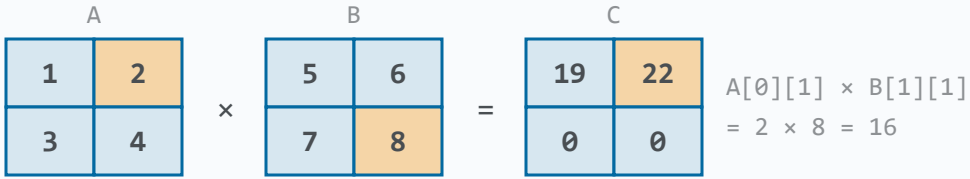
```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
▶       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

`i = 0` `j = 1` `k = 0`

`C = [[19, 6], [0, 0]]`

Matris Çarpımı (2×2)



C[0][1] += 2 × 8 → 6 + 16 = **22** (hücre tamam)

Adım 5 / 10

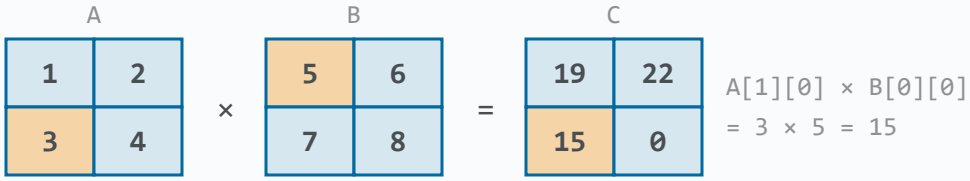
```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
▶       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

`i = 0` `j = 1` `k = 1`

`C = [[19, 22], [0, 0]]`

Matris Çarpımı (2×2)



C[1][0] += 3 × 5 → 0 + 15 = **15**

Adım 6 / 10

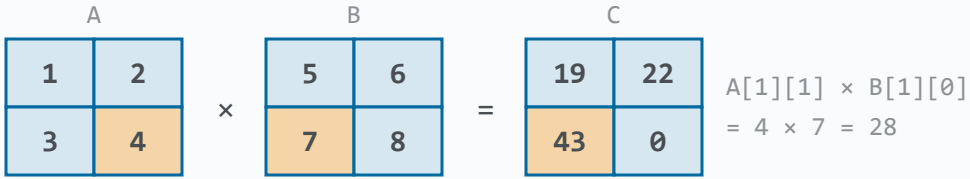
```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
▶       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

`i = 1` `j = 0` `k = 0`

`C = [[19, 22], [15, 0]]`

Matris Çarpımı (2×2)



C[1][0] += 4 × 7 → 15 + 28 = **43** (hücre tamam)

Adım 7 / 10

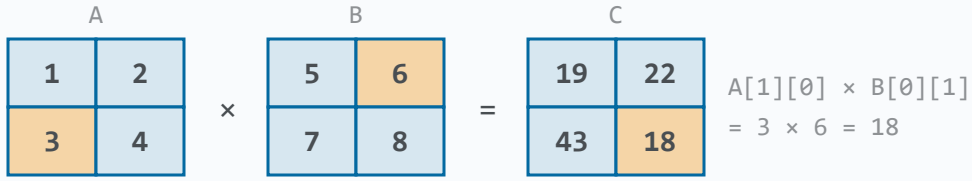
```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
▶       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

`i = 1` `j = 0` `k = 1`

`C = [[19, 22], [43, 0]]`

Matris Çarpımı (2×2)



C[1][1] += 3 × 6 → 0 + 18 = **18**

Adım 8 / 10

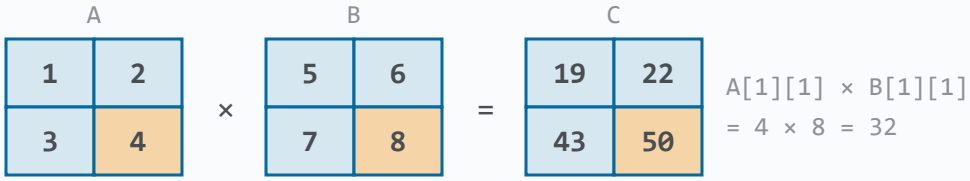
```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
▶       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

`i = 1` `j = 1` `k = 0`

`C = [[19, 22], [43, 18]]`

Matris Çarpımı (2×2)



C[1][1] += 4 × 8 → 18 + 32 = **50** (hücre tamam)

Adım 9 / 10

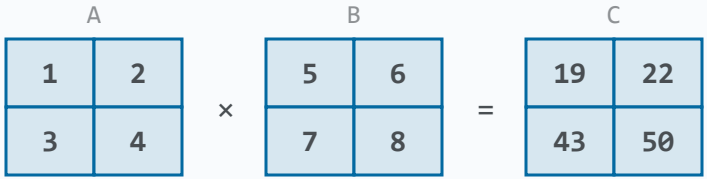
```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
4 for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
▶       C[i][j] += A[i][k] * B[k][j];
```

DEĞİŞKENLER

`i = 1` `j = 1` `k = 1`

`C = [[19, 22], [43, 50]]`

Matris Çarpımı (2×2)



Üç iç içe döngü: n×n matris için n^3 çarpma → **O(n^3)**. Sütun sayısı (A) = satır sayısı (B) olmalı.

Adım 10 / 10

```
1 int[][] A = {{1, 2}, {3, 4}};
2 int[][] B = {{5, 6}, {7, 8}};
3 int[][] C = new int[2][2];
▶ for (int i = 0; i < 2; i++)
5   for (int j = 0; j < 2; j++)
6     for (int k = 0; k < 2; k++)
7       C[i][j] += A[i][k] * B[k][j];
```

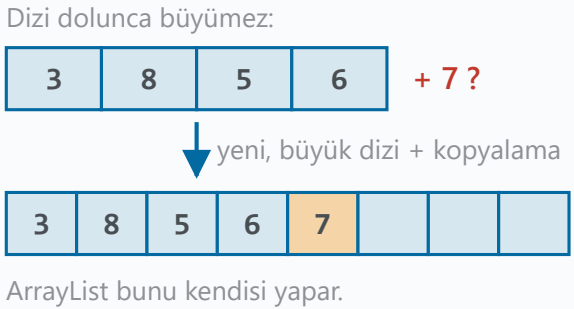
DEĞİŞKENLER

C = **[[19, 22], [43, 50]]**

TASARIM

Dizi mi Liste mi?

- Öğe sayısı baştan belli ve sabitse dizi yeterli ve hızlıdır.
- Öğe sayısı değişecekse `ArrayList` gibi koleksiyonlar daha kullanışlıdır.
- `int[]` sayıları doğrudan saklar; `ArrayList<Integer>` her sayıyı bir nesne içinde tutar.
- Seçim; kullanım biçimine, okunabilirliğe ve ölçülen performansa göre yapılır.



DOĞRULAMA

Yaygın Hatalar

Bir tur fazla

`i <= a.length` son turda `a[length]`'e erişir → `ArrayIndexOutOfBoundsException`. Doğrusu `i < a.length`.

Boş dizide ilk öğe

`int min = a[0];` boş dizide hata verir; önce `a.length == 0` kontrolü.

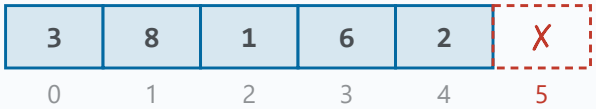
Referans kopyası

`int[] b = a;` yeni dizi değildir; `b[0] = 9` a'yı da değiştirir. Kopya için `a.clone()`.

Düzensiz matris

Her satırın uzunluğu `matris[0].length` değildir; iç döngüde `matris[i].length` kullanılır.

`i <= a.length` → son turda `a[5]`



a.length = 5 → geçerli indisler 0 … 4

`int[] b = a; b[0] = 9;`



tek dizi: a[0] de 9 olur

ÖZET

Bölüm Özeti

Yapı

Aynı tipte, sabit sayıda öğe; indis 0 … length − 1; değişken diziyi referansla gösterir.

Erişim

`a[i]` okuma/yazma sabit sürede; sınır dışı indis çalışma zamanı hatası.

Gezinme

Klasik for (indis gerekince) ve gelişmiş for (döngü değişkenine atama diziyi değiştirmez); toplam, min-max, ters çevirme, arama.

Matris

`int[][]` dizi dizisidir: `matris.length` satır, `matris[i].length` sütun sayısı.

ÖZET

Kontrol Soruları

1. Beş öğeli dizinin son indisi nedir?
2. Gelişmiş for ne zaman yetersiz kalır?
3. Dizi ataması ile öğe kopyası arasındaki fark nedir?
4. İkili arama için hangi ön koşul gerekir?
5. `matris.length` ile `matris[0].length` neyi verir?

▶ **Cevaplar**

ÖZET

Alıştırmalar

1. Bir `int[]` dizisindeki çift sayıların adedini ve toplamını bulan kodu yazın.
2. Diziyi ekstra dizi kullanmadan bir konum **sağa döndüren** (son öğe başa) kodu yazın: {1,2,3,4} → {4,1,2,3}.
3. İki sıralı diziyi tek bir sıralı dizide birleştirin (`{1,4,7}` + `{2,3,9}`).
4. Kabarcık sıralamasına, bir geçişte hiç takas olmazsa erken duran bir `boolean` bayrak ekleyin.
5. Eratosthenes eleğiyle 100'e kadar kaç asal olduğunu sayın (cevap: 25).
6. 3×3 bir matrisin ana köşegen toplamını ve transpozunu hesaplayın; matris simetrik mi kontrol edin.

ÖZET

Kaynaklar

- Deitel ve Deitel, Java: How to Program, Early Objects, 11. baskı, Pearson, 2018.
- Liang, Introduction to Java Programming and Data Structures, 13. baskı, Pearson, 2024.
- Oracle, The Java Tutorials: **Arrays**.
- Oracle, Java SE 21 API: **java.util.Arrays**; Java Language Specification SE 21: **Bölüm 10, Arrays**.
- Sedgewick ve Wayne, Computer Science: An Interdisciplinary Approach, Addison-Wesley, 2016: **1.4 Arrays**.
- Cormen, Leiserson, Rivest ve Stein, Introduction to Algorithms, 4. baskı, MIT Press, 2022.

Son

Diziler

Sonraki sunum: Dizgiler · Çalışma soruları · Simülatör: Dizi Laboratuvarı · Terim sözlüğü