

# BBS515 Nesneye Yönelik Programlama

---

Ders 6 - Sıralama ve Arama Algoritmaları  
Çok boyutlu diziler

Zümra Kavafoğlu  
*<https://zumrakavafoglu.github.io/>*

# Sıralama algoritmaları

---

Çoğu bilimsel çalışmada bir dizi nesnenin artan ya da azalan şekilde sıralanabilmesi büyük önem taşır. Biz bu sıralamayı kendi mantığımızla çok basit bir şekilde yapabiliriz, ancak fazla sayıda veriyi sıralamak zaman alır. Java programlama diliyle sıralama yapmak için bazı yöntemler vardır.

# Selection Sort(Seme Sıralaması)

---

Başlangı adımı  $i=0$  olarak ata. Dizinin uzunluęu  $n$  olsun.

1. adım: dizideki  $i$ . elemandan sonra gelen ve ondan küçük en küçük elemanın yerini bul.
2. Böyle bir en küçük eleman varsa bu elemanla  $i$ . elemanın yerini deęiştir.
3. adım :  $i < n-1$  ise  $i$ 'yi bir arttır ve 1. adıma git, deęilse sıralamayı sonlandır.

# SELECTION SORT $O(n^2)$

5 1 12 -5 16 2 12 14

5 1 12 -5 16 2 12 14

-5 1 12 5 16 2 12 14

-5 1 12 5 16 2 12 14

-5 1 2 5 16 12 12 14

-5 1 2 5 16 12 12 14

-5 1 2 5 12 16 12 14

-5 1 2 5 12 12 16 14

-5 1 2 5 12 12 14 16

# Selection Sort(Seme Sıralaması)

---

```
public static void selectionSort(int[] arr) {  
    int i, j, minIndex;  
    int n = arr.length;  
    for (i = 0; i < n - 1; i++) {  
        minIndex = i;  
        for (j = i + 1; j < n; j++)  
            if (arr[j] < arr[minIndex])  
                minIndex = j;  
        if (minIndex != i) {  
            swapElements(arr, minIndex, i);  
        }  
    }  
}
```

# Bubble Sort (Kabarcık Sıralaması)

---

$j = 0$  ve dizinin uzunluğu  $n$  olsun .

1. adım:  $i = 0$  olsun.
2. adım:  $i$ . elemanla  $(i+1)$ nci elemanı karşılaştır.  $i$ . eleman  $(i+1)$ . elemandan büyükse yerlerini değiştir.
3. adım :  $i$ ,  $n-j$ 'den küçükse  $i$ 'yi bir arttır ve 2. adıma git.
4. adım: Eğer  $j$  adımı için 2. adımda bir değişiklik yapıldıysa  $j$ 'yi 1 arttır ve 1. adıma git. Yapılmadıysa sıralamayı sonlandır.

# Bubble Sort (Kabarcık Sıralaması)

BUBBLE  
SORT  $O(n^2)$

5	1	12	-5	16	unsorted
5	1	12	-5	16	5 > 1, swap
1	5	12	-5	16	5 < 12, ok
1	5	12	-5	16	12 > -5, swap
1	5	-5	12	16	12 < 16, ok
1	5	-5	12	16	1 < 5, ok
1	5	-5	12	16	5 > -5, swap
1	-5	5	12	16	5 < 12, ok
1	-5	5	12	16	1 > -5, swap
-5	1	5	12	16	1 < 5, ok
-5	1	5	12	16	-5 < 1, ok
-5	1	5	12	16	sorted

# Bubble Sort (Kabarcık Sıralaması)

---

```
static void bubbleSort(int[] arr) {  
  
    boolean swapped = true;  
    int j = 0;  
  
    while (swapped) {  
        swapped = false;  
        j++;  
        for (int i = 0; i < arr.length - j; i++) {  
            if (arr[i] > arr[i + 1]) {  
                swapElements(arr, i, i + 1);  
                swapped = true;  
            }  
        }  
    }  
}
```



# Arrays sınıfının sort metotları

---

Sıralama programlamada çok ihtiyaç duyulan bir işlem olduğundan Java overloaded sort metodunu otomatik olarak sağlamaktadır.

```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};  
java.util.Arrays.sort(numbers);
```

```
char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};  
java.util.Arrays.sort(chars);
```

# Arama Algoritmaları

---

Bir dizinin içinde belirli bir elemanın olup olmadığını ve bu elemanın yerini bulmamızı gerektiren birçok uygulamaya ihtiyaç duyulur. Bunun için çeşitli arama algoritmaları geliştirilmiştir.

# Arama Algoritmaları: Doğrusal Arama (Linear Search)

---

Doğrusal arama algoritması aranan elemanı dizinin her elemanı ile karşılaştırır. ( $O(n)$ )

```
public static int linearSearch(int key, int[] arr){  
    int n = arr.length;  
    for(int i=0; i<n; i++){  
        if(key == arr[i]){  
            return i;  
        }  
    }  
  
    return -1;  
}
```

# Arama Algoritmaları: İkili Arama (Binary Search)

---

Bir diğer yaygın arama algoritması ikili arama algoritmasıdır. İkili aramanın çalışabilmesi için dizinin sıralı olması gerekmektedir. Dizinin azalmayan biçimde sıralı olduğunu varsayalım. İkili arama önce aranan elemanla(anahtar) dizinin tam ortasındaki elemanı karşılaştırır. Şu üç durumu göz önüne alalım:

- Eğer anahtar ortadaki elemandan küçükse, anahtarı yalnızca dizinin ilk yarısında aramalısınız.
- Eğer anahtar ortadaki elemana eşitse arama başarılı olmuştur.
- Eğer anahtar ortadaki elemandan büyükse, anahtarı yalnızca dizinin ikinci yarısında aramalısınız.

Aynı işlemleri dizinin aramaya devam ettiğimiz yarısıyla tekrarlayarak algoritma devam eder.

# BINARY SEARCH $O(\log_2 n)$

$\{-1, 5, 6, 18, 19, 25, 46, 78, 102, 114\}$  dizisinde 6'yı bulalım:

Step 1 (middle element is $19 > 6$ ):	-1	5	6	18	19	25	46	78	102	114
Step 2 (middle element is $5 < 6$ ):	-1	5	6	18	19	25	46	78	102	114
Step 3 (middle element is $6 == 6$ ):	-1	5	6	18	19	25	46	78	102	114

$\{-1, 5, 6, 18, 19, 25, 46, 78, 102, 114\}$  dizisinde 103'ü bulalım:

Step 1 (middle element is $19 < 103$ ):	-1	5	6	18	19	25	46	78	102	114
Step 2 (middle element is $78 < 103$ ):	-1	5	6	18	19	25	46	78	102	114
Step 3 (middle element is $102 < 103$ ):	-1	5	6	18	19	25	46	78	102	114
Step 4 (middle element is $114 > 103$ ):	-1	5	6	18	19	25	46	78	102	114
Step 5 (searched value is absent):	-1	5	6	18	19	25	46	78	102	114

# Arama Algoritmaları: İkili Arama (Binary Search)

---

```
public class BinarySearch {  
  
    public int binarySearch(int[] arr, int key) {  
  
        int low = 0;  
        int high = arr.length - 1;  
  
        while (low <= high) {  
  
            int mid = (low + high)/2;  
  
            if (arr[mid] == key)  
                return mid;  
            else if (arr[mid] < key)  
                low = mid + 1;  
            else  
                high = mid - 1;  
  
        }  
        return -1 - low;  
    }  
}
```

# Arama Algoritmaları: İkili Arama (Binary Search)

---

- `binarySearch` metodu anahtar değerle aynı değere sahip dizi elemanı varsa o elemanın indisini döndürür. Aksi takdirde, `-insertionPoint-1` değerini döndürür.
- `insertionPoint` dizinin sıralamasına göre anahtar değerın yer alacağı indis değeridir.

# Arrays.binarySearch

---

- İkili arama programlamada sıklıkla kullanıldığından Java'da hazır bir `binarySearch` metodu vardır. `binarySearch` metodu `int`, `double`, `char`, `short`, `long` ve `float` için overload edilmiş bir metoddur.

```
int arr[] = {-1, 5, 6, 18, 19, 25, 46, 78, 102, 114};  
int keyIndex = java.util.Arrays.binarySearch(arr, 6);
```



2

```
char letters[] = {'c', 'k', 'r', 's', 'z'};  
int keyIndex = java.util.Arrays.binarySearch(letters, 'p');
```



-3

( -insertionPoint-1 = -2-1)



# İki boyutlu diziler

---

- İki boyutlu diziler veya tablolara gerçek hayatta sıklıkla ihtiyaç duyulur:
  - Satranç tahtası
  - Otobüs tarifesi
  - Hesap çizelgeleri vs.

# İki boyutlu diziler

---

- Örneğin bir bilgisayar firmasının 4 mağazasının her birinde haftanın her bir gününde satılan bilgisayar sayısını aşağıdaki gibi bir tabloyla gösterebiliriz:

		Haftanın günleri						
Mağazalar		0	1	2	3	4	5	6
	0	15	7	3	0	12	10	4
	1	3	8	7	6	1	11	2
	2	1	17	3	9	10	1	1
	3	2	6	7	5	18	25	20

- Bu tablo için sales isminde iki boyutlu bir dizi oluşturabiliriz. Bu dizinin satır sayısı 4, sütun sayısı 7 olmalıdır. Bu dizi tek boyutlu dizilere benzer olarak aşağıdaki biçimde tanımlanabilir.

```
int[][] sales = new int[4][7];
```

# İki boyutlu dizi tanımlamak

---

Dizi adı

Sütun sayısı

```
int [] [] sales = new int [4] [7];
```

Dizi elemanlarının tipi

Satır sayısı

# İki boyutlu dizi: İndisler

---

- İki boyutlu bir dizideki belirli bir elemana ulaşmak için, o elemanın dizideki yerini belirten iki tamsayı değere ihtiyacımız vardır: Satır indisi ve sütun indisi.
- Örneğin satış tablosuna göre 3. mağazada Salı günü satılan bilgisayar sayısına ulaşmak için sales dizisinin satır indisi 2, sütun indisi 1 olan elemanına ulaşmalıyım. (Tek boyutlu dizilerde olduğu gibi, iki boyutlu dizilerde de indisleme 0'dan başlar.) Bu elemana aşağıdaki komutla ulaşabilirim:

Sütun indisi  
↓  
**int** n = sales[2][1];  
↑                   ↑  
17 değeri atanır    Satır indisi

# İki boyutlu dizi: İndisler

---

- Benzer biçimde dizideki herhangi bir elemanın değerini de yine elemana indisler yoluyla ulaşp değiştirebiliriz.
- Örneğin haftanın 6. günü 1. mağazada satılan bilgisayar sayısını 15 yapmak istersek aşağıdaki komutu kullanabiliriz.

```
sales[0][5] = 15;
```

ya da haftanın 1. günü 4. mağazada satılan bilgisayar sayısını aşağıdaki komutla 1 arttırabiliriz.

```
sales[3][1]++;
```

# İki boyutlu dizi: Dizi boyu

---

- İki boyutlu dizilerin boyutu belirlemek için iki farklı tamsayı değer kullanılır: Biri satır sayısı, diğeri ise sütun sayısı.
- Örneğin satış tablomuzda satır sayısı 4 iken sütun sayısı 7'dir.
- Satır sayısını hesaplamak için length komutu kullanılır. Yani satış tablomuz için satır sayısını aşağıdaki ifadeyle elde edebiliriz.

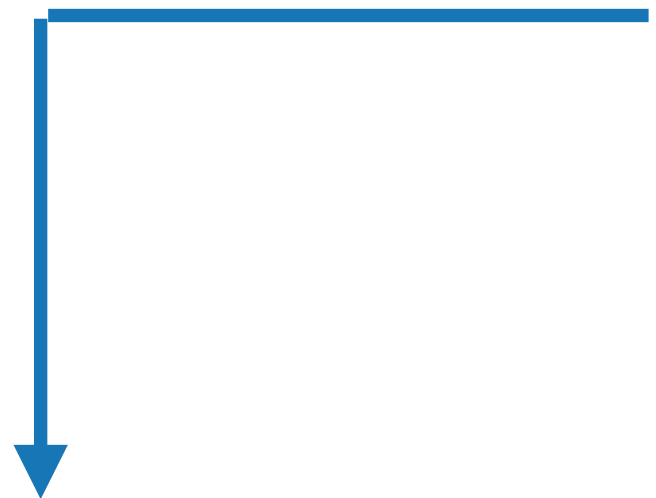
```
int rowSize = sales.length;
```

- Sütun sayısını hesaplamak için ise herhangi bir satırın uzunluğunu kullanabiliriz. (Hangi satırın uzunluğunu kullandığımız fark etmez çünkü tüm satırların uzunluğu aynıdır ve sütun sayısını verir)

```
int columnSize = sales[0].length;
```

# İki boyutlu dizi: Dizi boyu

---



	0	1	2	3	4	5	6
0	15	7	3	0	12	10	4
1	3	8	7	6	1	11	2
2	1	17	3	9	10	1	1
3	2	6	7	5	18	25	20

`sales[0] = {sales[0][0], sales[0][1], sales[0][2], sales[0][3], sales[0][4], sales[0][5], sales[0][6]}`  
`= {15, 7, 3, 0, 12, 10, 4}`

# İki boyutlu dizi: Dizi boyu

---

	0	1	2	3	4	5	6
0	15	7	3	0	12	10	4
1	3	8	7	6	1	11	2
2	1	17	3	9	10	1	1
3	2	6	7	5	18	25	20



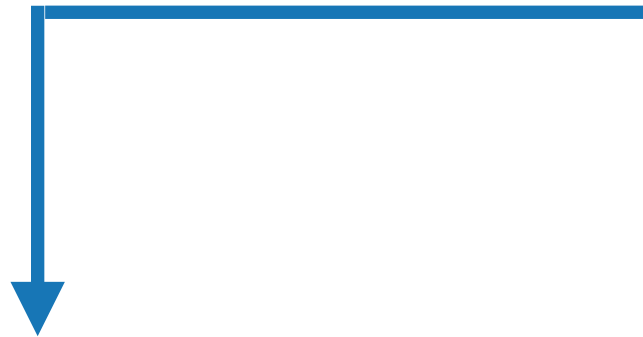
`sales[1] = {sales[1][0], sales[1][1], sales[1][2], sales[1][3], sales[1][4], sales[1][5], sales[1][6]}`  
`= {3, 8, 7, 6, 1, 11, 2}`



# İki boyutlu dizi: Dizi boyu

---

	0	1	2	3	4	5	6
0	15	7	3	0	12	10	4
1	3	8	7	6	1	11	2
2	1	17	3	9	10	1	1
3	2	6	7	5	18	25	20

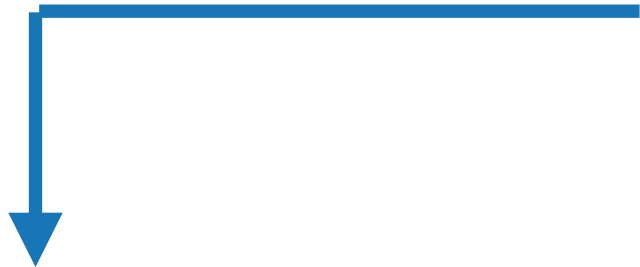


`sales[2] = {sales[2][0], sales[2][1], sales[2][2], sales[2][3], sales[2][4], sales[2][5], sales[2][6]}`  
`= {1, 17, 3, 9, 10, 1, 1}`

# İki boyutlu dizi: Dizi boyu

---

	0	1	2	3	4	5	6
0	15	7	3	0	12	10	4
1	3	8	7	6	1	11	2
2	1	17	3	9	10	1	1
3	2	6	7	5	18	25	20



`sales[3] = {sales[3][0], sales[3][1], sales[3][2], sales[3][3], sales[3][4], sales[3][5], sales[3][6]}`  
`= {2, 6, 7, 5, 18, 25, 20}`

# İki boyutlu dizi: İndisler ve dizi boyu

---

- İki boyutlu bir dizinin elemanları çağırılırken ilk indis satır boyundan, ikinci indis sütun boyundan küçük ve her iki indis de sıfırdan büyük olmalıdır. Aksi takdirde çalışma zamanı hatası oluşur.

# İki boyutlu dizi: İndisler ve dizi boyu

---

- İki boyutlu bir dizinin elemanları çağırılırken ilk indis satır boyundan, ikinci indis sütun boyundan küçük ve her iki indis de sıfırdan büyük olmalıdır. Aksi takdirde çalışma zamanı hatası oluşur.
- İki boyutlu dizinin satır ve sütun boyunu sabit değerler olarak tanımlamak programın okunurluğunu ve düzenlenebilirliğini artırır.

```
final int shops = 4;  
final int days = 7;
```

```
int[][] sales = new int[shops][days];
```

# İki boyutlu dizi: İlk değer vermek

---

- İki boyutlu bir dizinin elemanlarına ilk değer vermek için çeşitli yollar kullanılabilir:
- Örneğin ilk satırının tüm elemanları 0, ikinci satırının tüm elemanları 1 olan 2x3 boyutlu bir diziye ilk eleman vermek için aşağıdaki gibi for döngüsünden faydalanabiliriz:

```
final int rows = 2;
final int columns = 3;

int[][] matrix = new int[rows][columns];

for(int i=0; i<rows; i++){
    for(int j=0; j<columns; j++){
        matrix[i][j] = i;
    }
}
```

# İki boyutlu dizi: İlk değer vermek

- Dizilere bir diğer ilk değer verme biçimi de diziyi ilk değerleriyle tanımlamaktır. Örneğin sales dizisini aşağıdaki gibi tanımlayabiliriz. Bu tanımlamayla hem 4 satırlı ve 7 sütunlu bir dizi oluşturmuş hem de ona ilk değerini vermiş oluruz.

Haftanın günleri		0	1	2	3	4	5	6
Mağazalar	0	15	7	3	0	12	10	4
	1	3	8	7	6	1	11	2
	2	1	17	3	9	10	1	1
	3	2	6	7	5	18	25	20

```
int[][] sales = {{15, 7, 3, 0, 12, 10, 4},  
                 {3, 8, 7, 6, 1, 11, 2},  
                 {1, 17, 3, 9, 10, 1, 1},  
                 {2, 6, 7, 5, 18, 25, 20}};
```

# Dizi elemanlarını toplamak

---

- sales dizisini kullanarak o hafta toplam kaç bilgisayar satıldığını hesaplayan program:

```
public static void main(String[] args){  
    int[][] sales = {{15, 7, 3, 0, 12, 10, 4},  
                     {3, 8, 7, 6, 1, 11, 2},  
                     {1, 17, 3, 9, 10, 1, 1},  
                     {2, 6, 7, 5, 18, 25, 20}};  
  
    int rowSize = sales.length;  
  
    int columnSize = sales[0].length;  
  
    int totalSale = 0;  
  
    for(int i=0; i<rowSize; i++){  
        for(int j=0; j<columnSize; j++){  
            totalSale += sales[i][j];  
        }  
    }  
  
    System.out.print("Total sale this week: "+totalSale);  
}
```

# Metot parametresi olarak iki boyutlu diziler: sum metodu

---

- Parametre olarak aldığı iki boyutlu bir dizinin elemanlarının toplamını hesaplayan sum isimli bir metot yazalım:

```
public static int sum(int[][] arr){  
    int rowSize = arr.length;  
  
    int columnSize = arr[0].length;  
  
    int total = 0;  
  
    for(int i=0; i<rowSize; i++){  
        for(int j=0; j<columnSize; j++){  
            total += arr[i][j];  
        }  
    }  
  
    return total;  
}
```



# Metot parametresi olarak iki boyutlu diziler: sum metodu

---

- sum metodunu kullanarak haftalık bilgisayar satışını hesaplayan program:

```
public static void main(String[] args) {  
    int[][] sales = {{15, 7, 3, 0, 12, 10, 4},  
                     {3, 8, 7, 6, 1, 11, 2},  
                     {1, 17, 3, 9, 10, 1, 1},  
                     {2, 6, 7, 5, 18, 25, 20}};  
  
    int totalSale = sum(sales);  
  
    System.out.print("Total sale this week: "+ totalSale);  
}
```

# Metot parametresi olarak iki boyutlu diziler: columnSum metodu

---

- Parametre olarak aldığı iki boyutlu bir dizinin yalnızca bir sütunundaki elemanların toplamını hesaplayan metot:

```
public static int columnSum(int[][] arr, int columnIndex){  
    int rowSize = arr.length;  
  
    int total = 0;  
  
    for(int i=0; i<rowSize; i++){  
        total += arr[i][columnIndex];  
    }  
    return total;  
}
```

# Metot parametresi olarak iki boyutlu diziler: columnSum metodu

---

- columnSum metodunu kullanarak haftanın belirli bir gününde yapılan satışı hesaplayan program:

```
public static void main(String[] args) {  
    int[][] sales = {{15, 7, 3, 0, 12, 10, 4},  
                     {3, 8, 7, 6, 1, 11, 2},  
                     {1, 17, 3, 9, 10, 1, 1},  
                     {2, 6, 7, 5, 18, 25, 20}};  
  
    int dailySale;  
    int day;  
    Scanner input = new Scanner(System.in);  
  
    System.out.print("Enter the day number(1-7): ");  
    day = input.nextInt();  
  
    dailySale = columnSum(sales, day-1);  
  
    System.out.printf("Total sale on day %d is %d: ", day, dailySale);  
}
```

# Dizi elemanlarını yazdırmak: printArray metodu

---

```
public static void printArray(int[][] arr){  
    int rowSize = arr.length;  
    int columnSize = arr[0].length;  
    for(int i=0; i<rowSize; i++){  
        for(int j=0; j<columnSize; j++){  
            System.out.print(arr[i][j]+"\t");  
        }  
        System.out.println();  
    }  
}
```

# Jagged Array

---

- Java'da farklı sütun uzunluklarına sahip diziler de tanımlayabiliriz, bu dizilere jagged dizi denir.

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

- Satır sayısı: 3 (arr.length)
- 1. satırın eleman sayısı : 2 (arr[0].length)
- 2. satırın eleman sayısı: 6 (arr[1].length)
- 3. satırın eleman sayısı: 1 (arr[2].length)

# Jagged Array

---

- Java'da farklı sütun uzunluklarına sahip diziler de tanımlayabiliriz, bu dizilere jagged dizi denir.

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

- Satır sayısı: 3 (arr.length)
- 1. satırın eleman sayısı : 2 (arr[0].length)
- 2. satırın eleman sayısı: 6 (arr[1].length)
- 3. satırın eleman sayısı: 1 (arr[2].length)

# Jagged Array

---

- Satırları aşağıdaki alt alta satırlardan oluşan bir Jagged Array tanımlayalım:

```
0
1 2
3 4 5
6 7 8 9
10 11 12 13 14
```

- 5 satır var, o halde 5 satırlı bir jagged array tanımlamalıyız:

```
int r = 5;
```

```
int arr[][] = new int[r][];
```

# Jagged Array

---

- Satırları aşağıdaki alt alta satırlardan oluşan bir Jagged Array tanımlayalım:

```
0
1 2
3 4 5
6 7 8 9
10 11 12 13 14
```

- $i$  indisli satırın  $i+1$  adet elemanı var. O halde her bir satır için bu eleman sayısına sahip hafızada yer açmalıyım:

```
for (int i=0; i<arr.length; i++)
    arr[i] = new int[i+1];
```



# Jagged Array

---

- Satırları aşağıdaki alt alta satırlardan oluşan bir Jagged Array tanımlayalım:

```
0
1 2
3 4 5
6 7 8 9
10 11 12 13 14
```

- Satırlara istenen değerleri vermek için aşağıdaki gibi bir for döngüsü kullanabilirim:

```
int count = 0;
for (int i=0; i<arr.length; i++)
    for(int j=0; j<arr[i].length; j++)
        arr[i][j] = count++;
```

# Jagged Array elemanlarını yazdırmak: printJaggedArray

---

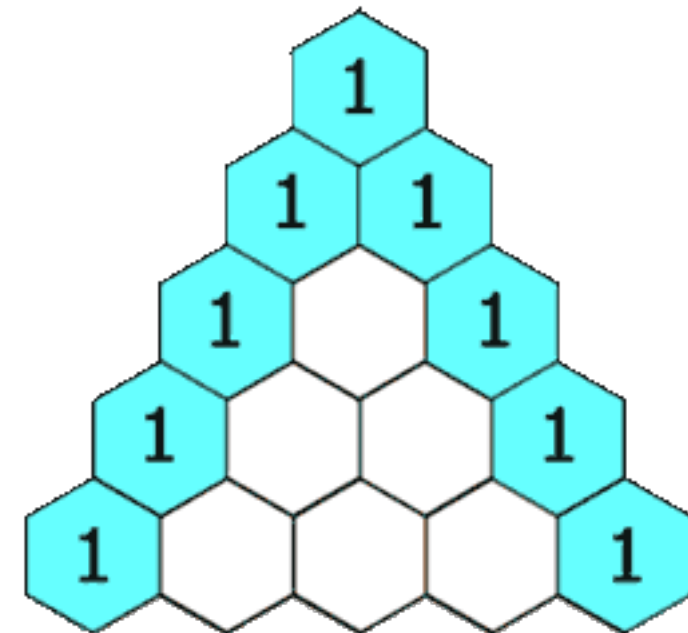
```
public static void printJaggedArray(int[][] arr){  
    int rowSize = arr.length;  
    for(int i=0; i<rowSize; i++){  
        for(int j=0; j<arr[i].length; j++){  
            System.out.print(arr[i][j]+" ");  
        }  
        System.out.println();  
    }  
}
```

# Jagged Array Örnek: Pascal Üçgeni (PascalTriangle.java)

---

- Her satırın başlangıç ve bitiş elemanları 1 olan ve ortadaki elemanların üst satırın elemanlarının toplamı ile bulunduğu sayı dizilimine Pascal Üçgeni denir. Yüksekliği verilen bir Pascal üçgenini bir Jagged Array içinde saklayan ve ekrana yazdıran bir Java programı yazınız.

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
```



# Çok boyutlu diziler

---

- İki boyutlu dizi tanımlama ve oluşturma yöntemleri  $n > 2$  olmak üzere  $n$ -boyutlu dizi tanımlama ve oluşturma yöntemlerine genişletilebilir. Örneğin aşağıdaki gibi üç boyutlu bir dizi oluşturulabilir:

```
double arr[ ][ ][ ] = new double[3][5][2];
```