

# Bölüm 6: Diziler (Arrays)



# Problem

Yüz adet sayıyı klavyeden okuyan, bu sayıların ortalamasını hesaplayan ve kaç adet sayının ortalamasının üzerinde olduğunu bulan programı yazınız.



# Çözüm

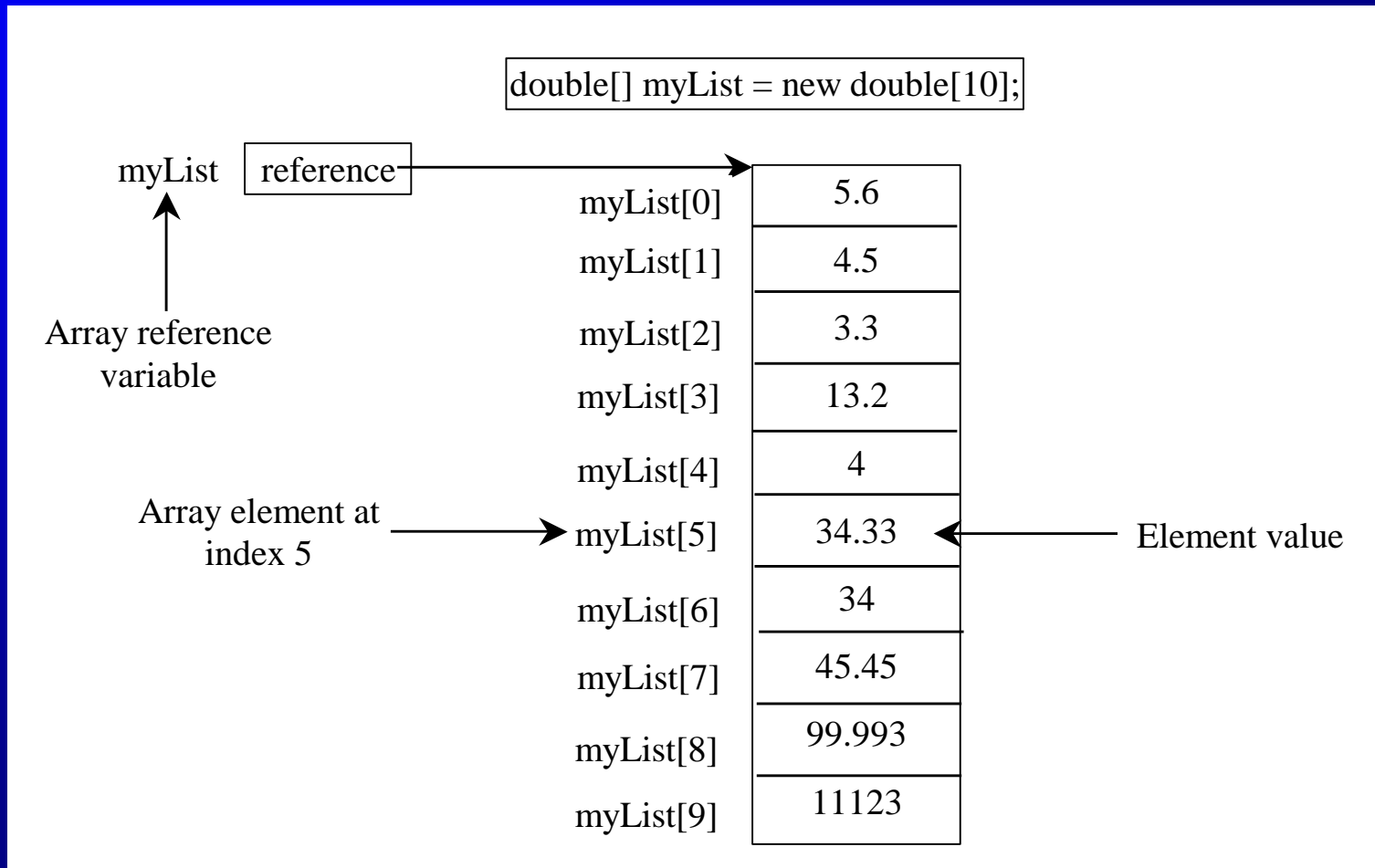
AnalyzeNumbers

Run

Run with prepared input

# Dizilere Giriş

Dizi, aynı türden veri koleksiyonunu temsil eden bir veri yapısıdır.



# Dizi Değişkenlerini Bildirme

□ `datatype[] arrayRefVar;`

Örnek:

```
double[] myList;
```

□ `datatype arrayRefVar[];` // Bu stile izin verilir, ancak tercih edilmez

Örnek:

```
double myList[];
```



# Dizileri Oluşturmak

```
arrayRefVar = new datatype[arraySize];
```

Örnek:

```
myList = new double[10];
```

`myList[0]` dizideki ilk öğeye başvurur.

`myList[9]` dizideki son öğeye başvurur.



# Tek Adımda Dizi Bildirme (Declaring) ve Oluşturma (Creating)

```
□ datatype[] arrayRefVar = new  
    datatype[arraySize];
```

```
double[] myList = new double[10];
```

```
□ datatype arrayRefVar[] = new  
    datatype[arraySize];
```

```
double myList[] = new double[10];
```



# Dizinin Uzunluğu

Bir dizi oluşturulduktan sonra, boyutu sabittir, değiştirilemez. Dizinin boyutu aşağıda belirtilen kod satırı ile bulunabilir.

```
arrayRefVar.length
```

Örnek,

```
myList.length returns 10
```





# Varsayılan Değerler

Bir dizi oluşturulduğunda, öğelerine varsayılan değer atanması aşağıdaki gibidir.

Sayısal (Numerik) ilkel veri tipleri için 0,

Char veri tipi için '\u0000'

Boolean veri tipi için ise *false* değeri atanır.



# İndekslenmiş (Indexed) Değişkenler

Dizi elemanlarına indeks üzerinden erişilir. Dizi indeksleri 0 tabanlıdır, yani 0' dan başlayıp `arrayRefVar.length-1`'e kadar devam eder. Şekil 6.1'deki örnekte, `myList` on adet double değer içerir ve indeksler 0 - 9 arasındadır.

Dizideki her öge, dizinlenmiş bir değişken olarak bilinen aşağıdaki sözdizimini kullanarak temsil edilir:

```
arrayRefVar[index];
```



# İndekslenmiş Değişkenleri Kullanma

- Bir dizi oluşturulduktan sonra, indekslenmiş bir değişken normal bir değişkenle aynı şekilde kullanılabilir. Örneğin, aşağıdaki kod `myList [0]` ve `myList [1]` 'deki değerleri toplayarak `myList [2]` 'ye atar.

```
myList[2] = myList[0] + myList[1];
```



# Dizi Başlatıcılarım (Initializers)

□ Bir adımda diziyi bildirme, oluşturma, başlatma:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

Bu kısayol sözdizimi (syntax) bir ifade içerisinde olmalıdır.



# Kısayol Notasyonunu Kullanarak Bildirme, Oluşturma ve Başlatma

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

Yukarıdaki kısayol notasyonu aşağıdaki kod bloğu ile eşdeğerdir.

```
double[] myList = new double[4];
```

```
myList[0] = 1.9;
```

```
myList[1] = 2.9;
```

```
myList[2] = 3.4;
```

```
myList[3] = 3.5;
```



# UYARI !

Kısayol gösterimini kullanarak, dizinin tümünü bir ifadede bildirmeniz, oluşturmanız ve başlatmanız gerekir. Bölmek, bir sözdizimi hatasına neden olur. Örneğin, aşağıdakiler yanlış:

```
double[] myList;
```

```
myList = {1.9, 2.9, 3.4, 3.5};
```




# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

Dizi değişken değerlerini bildirin, bir dizi oluşturun ve referansını değerlere atayın

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the array is created



0	0
1	0
2	0
3	0
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

i 1'den başla

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0





# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

i (=1) 5'den daha küçük

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

Bu satır yürütüldüğünde, value[1] 'in değeri=1

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the first iteration

0	0
1	1
2	0
3	0
4	0

# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

i++ işlemi ile i değeri 2 oldu

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5],  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the first iteration

0	0
1	1
2	0
3	0
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

```
public class Test {  
    public static void main(String[]  
        args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] +  
            values[4];  
    }  
}
```

i (= 2) 5'den daha küçük

After the first iteration

0	0
1	1
2	0
3	0
4	0

# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

Bu satır yürütüldükten sonra,  
 $\text{values}[2] = 2 + \text{values}[2-1]$

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

i++ ile i değeri 3 oldu.

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

i (= 3) 5'den daha küçük

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

Bu satır yürütüldükten sonra,  $\text{values}[3] = 6$  ( $3 + 3$ )

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

i++ ile i değeri 4 oldu.

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

i (= 4) 5'den daha küçük

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

Bu satır yürütüldükten sonra,  $\text{values}[4] = 10$  ( $4 + 6$ )

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++)  
            values[i] = i + values[i-1];  
        values[0] = values[1] + values[4];  
    }  
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

i++ ile i değeri 5 oldu.

```
public class Test {  
    public static void main(String[] args)  
    {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

$i (= 5) < 5$  false döndürür, bu durumda döngüden çıkılır

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the fourth iteration

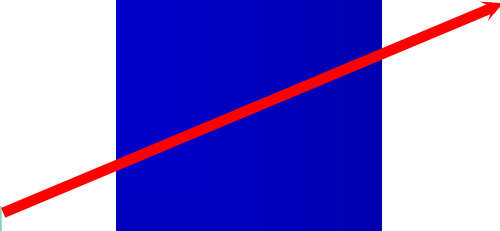
0	0
1	1
2	3
3	6
4	10



# Dizilerle ilgili Programın Trace Edilmesi (İzlenmesi)

Bu satır yürütüldükten sonra,  $\text{values}[0] = 11$  ( $1 + 10$ )

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5],  
        for (int i = 1; i < values.length; i++) {  
            values[i] = values[i-1] + values[i-2];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```



0	11
1	1
2	3
3	6
4	10



# İşlem Dizileri

Örnekler;

(Girdi(input) değerlerine sahip dizileri başlatma)

(Rasgele değerlere sahip dizileri başlatma)

(Dizileri yazdırma)

(Dizinin tüm öğelerini toplama)

(Dizinin en büyük elemanını bulma)

(En büyük elemanın en küçük indeksini bulma)

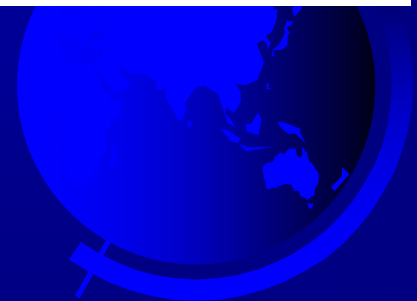
(Rasgele karıştırma)

(Elemanları kaydırma)



# Girdi Değerlerine Sahip Dizileri Başlatma

```
java.util.Scanner input = new java.util.Scanner(System.in);  
System.out.print("Enter " + myList.length + " values: ");  
for (int i = 0; i < myList.length; i++)  
    myList[i] = input.nextDouble();
```





# Dizileri Rasgele Değerlerle Başlatma

```
for (int i = 0; i < myList.length; i++) {  
    myList[i] = Math.random() * 100;  
}
```



# Dizileri Yazdırma

```
for (int i = 0; i < myList.length; i++) {  
    System.out.print(myList[i] + " ");  
}
```



# Tüm Öğeleri Toplama

```
double total = 0;  
for (int i = 0; i < myList.length; i++) {  
    total += myList[i];  
}
```



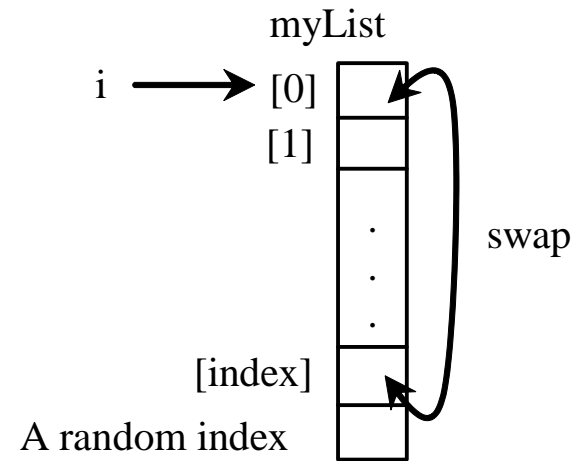
# Dizinin En Büyük Elemanını Bulma

```
double max = myList[0];  
for (int i = 1; i < myList.length; i++) {  
    if (myList[i] > max) max = myList[i];  
}
```



# Rasgele Karıştırma

```
for (int i = 0; i < myList.length; i++) {  
    // Generate an index j randomly  
    int index = (int) (Math.random()  
        * myList.length);  
  
    // Swap myList[i] with myList[j]  
    double temp = myList[i];  
    myList[i] = myList[index];  
    myList[index] = temp;  
}
```



# Elemanları Kaydırma

```
double temp = myList[0]; // Retain the first element
```

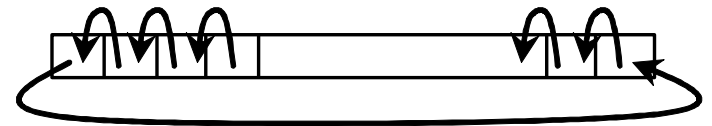
```
// Shift elements left
```

```
for (int i = 1; i < myList.length; i++) {  
    myList[i - 1] = myList[i];  
}
```

```
// Move the first element to fill in the last position
```

```
myList[myList.length - 1] = temp;
```

myList



# Döngü için Geliştirilmiş (for döngüsü için)

JDK 1.5 Bir indeks değişkeni kullanmadan tüm diziyi sıralı olarak geçmenizi sağlayan yeni bir for döngüsü. Örneğin, aşağıdaki kod myList dizisindeki tüm öğeleri görüntüler.

```
for (double value: myList)
    System.out.println(value);
```

Genel olarak, syntax

```
for (elementType value: arrayRefVar) {
    // Process the value
}
```

Diziyi farklı bir düzende ilerletmek veya dizideki öğeleri değiştirmek istiyorsanız yine de bir dizin değişkeni kullanmanız gerekir.



# Problem: Loto Numaraları

Diyelim ki, 10 numara isimli şans oyununu oynuyorsunuz. Her bilette 1 ile 99 arasında değişen 10 benzersiz numara var. Çok sayıda bilet alıyorsunuz. 1'den 99'a kadar olan tüm numaraları kapsayacak şekilde biletlerinizi almak istediniz. Bir dosyadaki bilet numaralarını okuyan ve tüm numaraların olup olmadığını kontrol eden bir program yazın. Dosyadaki son sayının 0 olduğunu varsayalım.

Lotto Numbers Sample Data

LottoNumbers

Run



# Problem: İskambil Destesi

52' lik kart destesinden rastgele dört kart alan bir program yazın. Tüm kartlar aşağıdaki gibi 0 ila 52 başlangıç değerleri ile dolu, deck adı verilen bir dizi kullanılarak gösterilebilir:

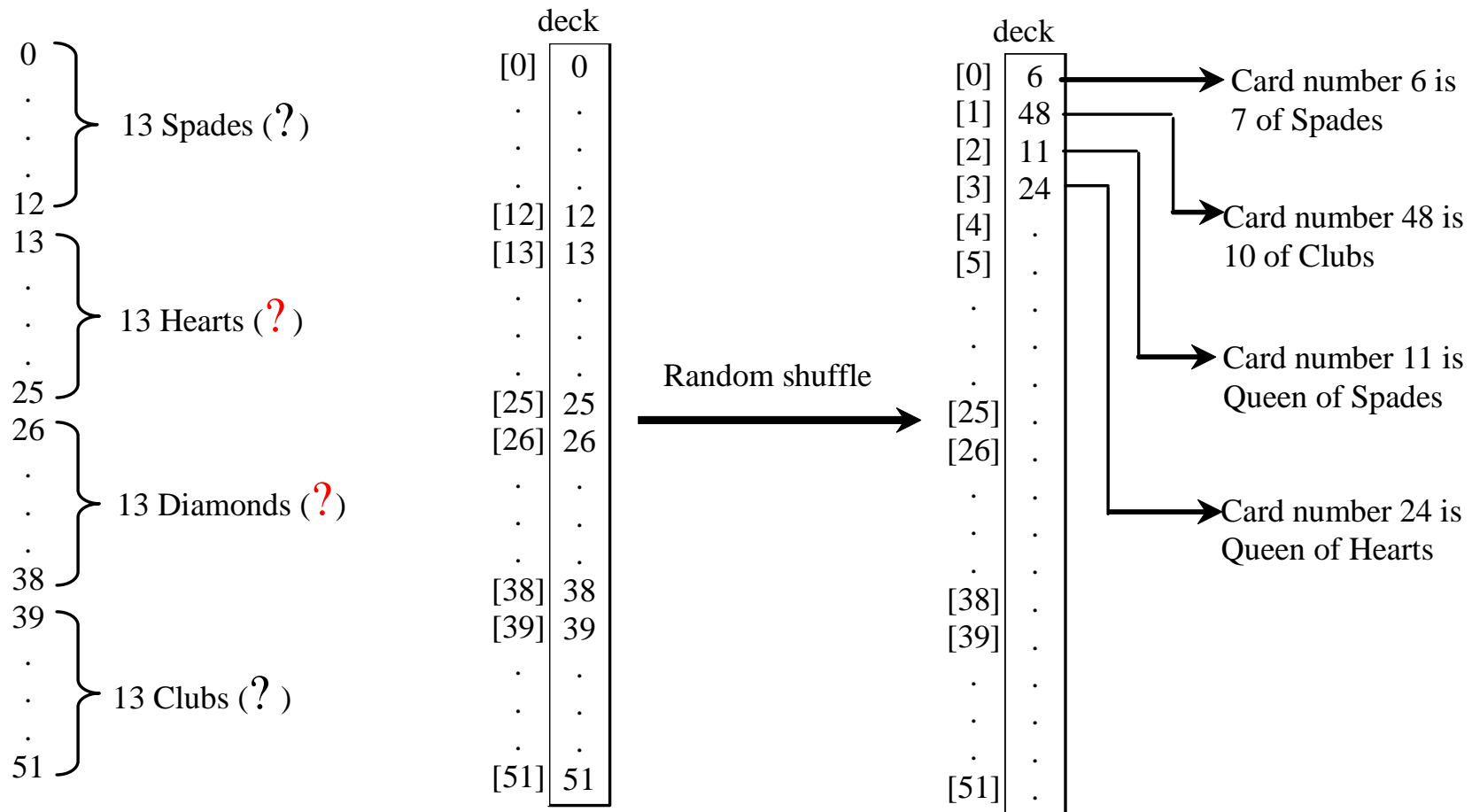
```
int[] deck = new int[52];  
// Initialize cards  
for (int i = 0; i < deck.length; i++)  
    deck[i] = i;
```

DeckOfCards

Run



# Problem: Iskambil Destesi



GUI Demo (picking four cards)

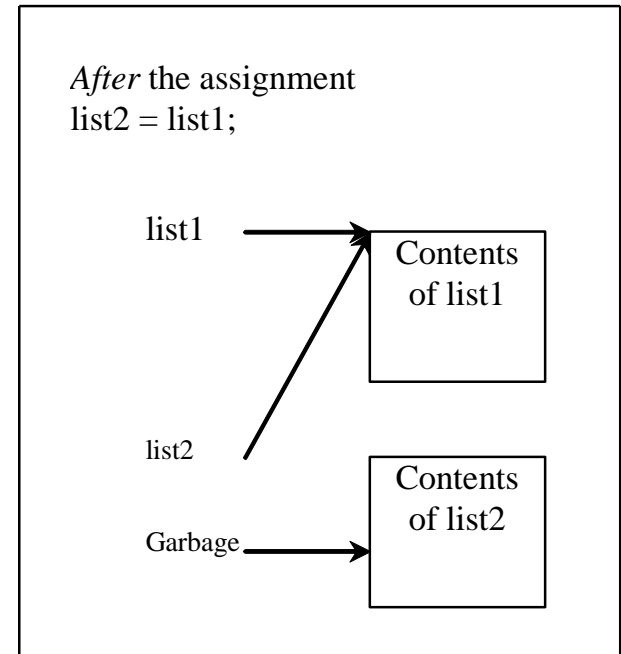
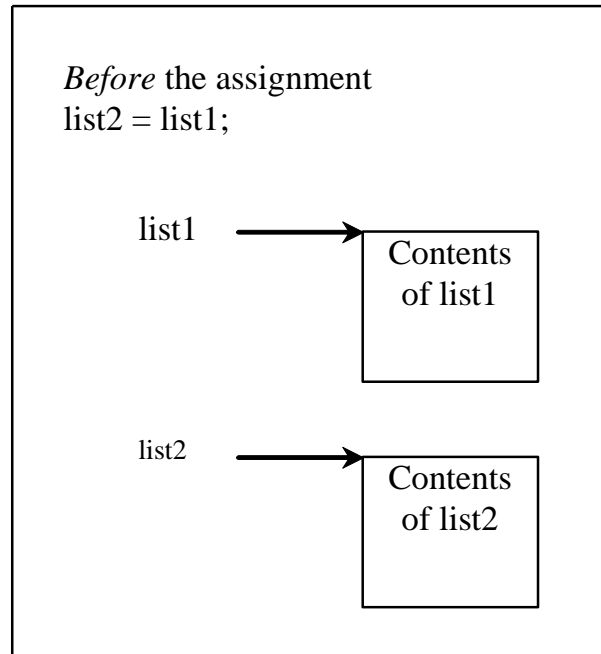
DeckOfCards

Run

# Dizilerin Kopyalanması

Genellikle, bir programda, bir diziyi veya dizinin bir bölümünü çoğaltmanız gerekebilir. Bu gibi durumlarda, atama deyimini (=) aşağıdaki gibi kullanmaya çalışabilirsiniz:

`list2 = list1;`



# Dizilerin Kopyalanması

Döngü Kullanarak;

```
int[] sourceArray = {2, 3, 1, 5, 10};  
int[] targetArray = new int[sourceArray.length];  
  
for (int i = 0; i < sourceArray.length; i++)  
    targetArray[i] = sourceArray[i];
```



# arraycopy Yardımcı Programı

```
arraycopy(sourceArray, src_pos,  
          targetArray, tar_pos, length);
```

Örnek:

```
System.arraycopy(sourceArray, 0,  
                 targetArray, 0, sourceArray.length);
```



# Dizilerin Metotlara Geçirilmesi

```
public static void printArray(int[] array) {  
    for (int i = 0; i < array.length; i++) {  
        System.out.print(array[i] + " ");  
    }  
}
```

//Metodu çağır.

```
int[] list = {3, 1, 2, 6, 4, 2};  
printArray(list);
```

Invoke the method

```
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Anonim Dizi



# Anonim Dizi

İfade;

```
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Aşağıdaki sözdizimini kullanılarak bir dizi oluşturur

```
:new dataType[]{literal0, literal1, ..., literalk};
```

Dizi için açık bir referans değişkeni yoktur. Bu diziye isimsiz bir dizi adı verilir.



# Pass By Value

Java, bir metoda parametreleri iletmek için pass by value yapısını kullanır. İlkel veri tiplerinin değişkenlerinin değerini iletmek ile dizileri iletmek arasında önemli farklılıklar vardır.

İlkel tür değerinin bir parametresi için gerçek değer iletilir. Yöntemin içindeki yerel parametrenin değerini değiştirmek, yöntemin dışındaki değişkenin değerini etkilemez.

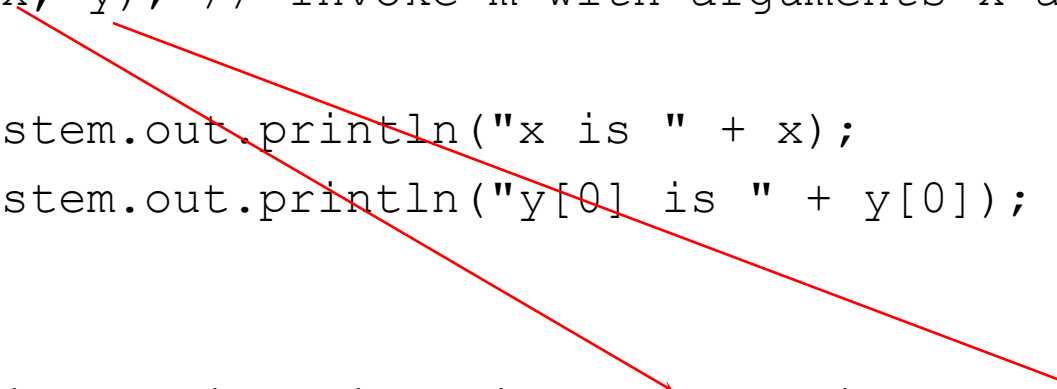
Bir dizi türünün bir parametresi için, parametrenin değeri bir diziye referans içerir; bu referans metoda iletilir. Metot gövdesinde gerçekleşen dizide yapılan değişiklikler, bağımsız değişken olarak geçirilen orijinal diziyi etkiler.





# Örnek

```
public class Test {  
    public static void main(String[] args) {  
        int x = 1; // x represents an int value  
        int[] y = new int[10]; // y represents an array of int values  
  
        m(x, y); // Invoke m with arguments x and y  
  
        System.out.println("x is " + x);  
        System.out.println("y[0] is " + y[0]);  
    }  
  
    public static void m(int number, int[] numbers) {  
        number = 1001; // Assign a new value to number  
        numbers[0] = 5555; // Assign a new value to numbers[0]  
    }  
}
```



# Dizileri Parametre Olarak İletmek

- Amaç: İlkel veri tipi değişkenleri ve dizi değişkenlerini iletmenin farklılıklarını gösterme.

TestPassArray

Run

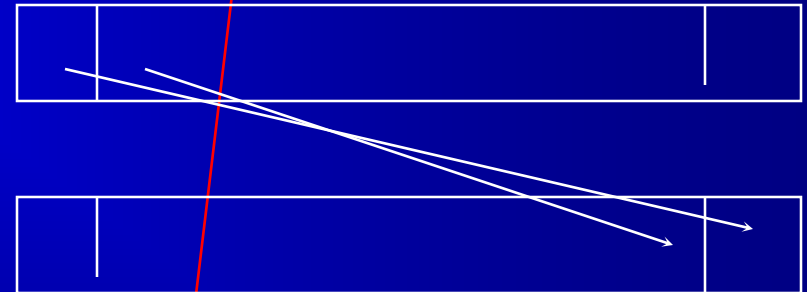


# Bir Metottan Dizi Döndürme

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

list

result



```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = {1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

result isimli bir dizi oluşturulur.

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	0	0
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 0 ve j = 5

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	0	0
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

i (= 0) değeri 6' dan küçük

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	0	0
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 0 ve j = 5  
list[0] değerini result[5]'e ata

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	0	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

İşlem sonrası, i değişkeni 1  
j değişkeni 4 olur.

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	0	1
---	---	---	---	---	---





# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i (=1) değeri 6' dan küçük

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	0	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 1 ve j = 4  
list[1] değerini result[4]'e ata

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

İşlem sonrası, i değişkeni 2  
j değişkeni 3 olur.

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i (=2) değeri 6'dan küçük

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	0	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 2 ve j = 3  
list[2] değerini result[3]'e ata

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

İşlem sonrası, i değişkeni 3  
j değişkeni 2 olur.

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i (=3) değeri 6' dan küçük

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	0	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
         i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 3 ve j = 2  
list[3] değerini result[2]'ye ata

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	4	3	2	1
---	---	---	---	---	---





# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

İşlem sonrası, i değişkeni 4  
j değişkeni 1 olur.

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	4	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i (=4) değeri 6' dan küçük

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	0	4	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
         i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 4 ve j = 1  
list[4] değerini result[1]'e ata

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	5	4	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

İşlem sonrası, i değişkeni 5  
j değişkeni 0 olur.

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	5	4	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i (=5) değeri 6' dan küçük

list

1	2	3	4	5	6
---	---	---	---	---	---

result

0	5	4	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 5 ve j = 0  
list[5] değerini result[0]'a ata

list

1	2	3	4	5	6
---	---	---	---	---	---

result

6	5	4	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

İşlem sonrası, i değişkeni 6  
j değişkeni -1 olur.

list

1	2	3	4	5	6
---	---	---	---	---	---

result

6	5	4	3	2	1
---	---	---	---	---	---



# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i (=6) < 6 ifadesi false döndürür. Döngüden çıkılır.

list

1	2	3	4	5	6
---	---	---	---	---	---

result

6	5	4	3	2	1
---	---	---	---	---	---



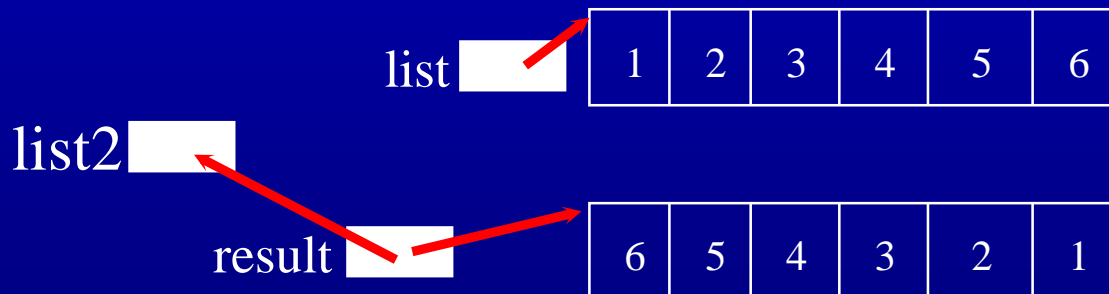


# «reverse» İsimli Metodun Trace Edilmesi

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

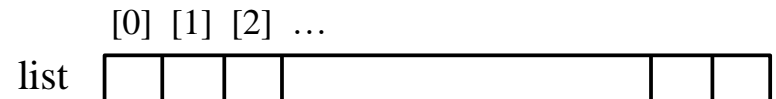
result'ı geri döndür.



# Dizilerde Arama

Arama, bir dizideki belirli bir öğeyi arama işlemidir; Örneğin, belirli bir puan değerinin puan listesine dahil olup olmadığını bulmak. Arama, bilgisayar programlamasında yaygın kullanılan bir işlemdir. Arama için geliştirilmiş birçok algoritma ve veri yapısı vardır. Bu bölümde, yaygın olarak kullanılan iki yaklaşım, doğrusal arama ve ikili arama ele alınmıştır.

```
public class LinearSearch {  
    /** The method for finding a key in the list */  
    public static int linearSearch(int[] list, int key) {  
        for (int i = 0; i < list.length; i++)  
            if (key == list[i])  
                return i;  
        return -1;  
    }  
}
```



key    Compare key with list[i] for i = 0, 1, ...

# Doğrusal Arama (Linear Search)

Doğrusal arama yaklaşımı, anahtar elemanı (aranan elemanı), dizi listesindeki her elemanla sırayla karşılaştırır. Bu yöntem, anahtar eleman listedeki bir öğeyle eşleşene veya liste bir eşleşme bulmadan tamamlanana kadar devam eder. Eşleşme gerçekleşirse, doğrusal arama, anahtarla eşleşen dizideki öğenin indeksini döndürür. Eşleşme bulunmazsa, arama -1 değerini döndürür.



# Doğrusal Arama Animasyonu

Anahtar

Liste

3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8

3	6	4	1	9	7	3	2	8
---	---	---	---	---	---	---	---	---



# Fikirden Çözüm

```
/** The method for finding a key in the list */  
public static int linearSearch(int[] list, int key) {  
    for (int i = 0; i < list.length; i++)  
        if (key == list[i])  
            return i;  
    return -1;  
}
```

## Metodun trace edilmesi (izlenmesi)

```
int[] list = {1, 4, 4, 2, 5, -3, 6, 2};  
int i = linearSearch(list, 4); // returns 1  
int j = linearSearch(list, -4); // returns -1  
int k = linearSearch(list, -3); // returns 5
```

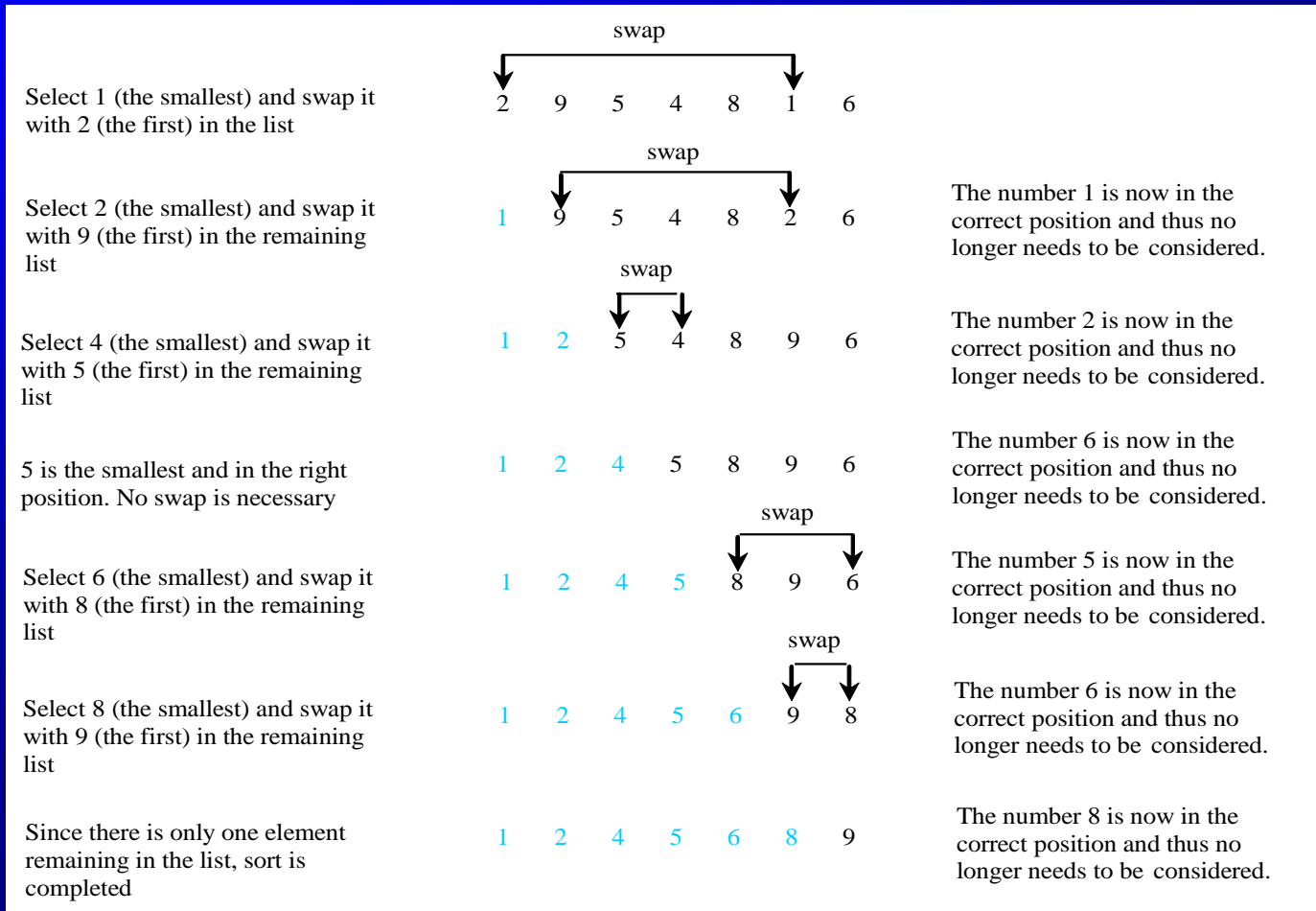
# Dizilerde Sıralama

Sıralama işlemi de, arama işlemi gibi bilgisayar programlamasında yaygın olarak kullanılan bir işlemdir. Sıralama için birçok farklı algoritma geliştirilmiştir. Bu bölümde iki basit, sezgisel sıralama algoritması tanıtılmaktadır: Seçmeli sıralama (Selection sort) ve Yerleştirmeli sıralama (insertion sort).



# Seçmeli sıralama (Selection Sort)

Seçmeli sıralama listedeki en büyük sayıyı bulur ve en sona yerleştirir. Daha sonra kalan en büyük sayıyı bulur ve en sondaki sayının öncesine yerleştirir, ve liste yalnızca bir sayı içerene kadar devam eder. Şekil 6.17, seçmeli sıralamayı kullanarak {2, 9, 5, 4, 8, 1, 6} listesinin nasıl sıralandığını gösterir.



# Fikirden Çözüme

```
for (int i = 0; i < list.length; i++)  
{  
    select the smallest element in list[i..listSize-1];  
    swap the smallest with list[i], if necessary;  
    // list[i] is in its correct position.  
    // The next iteration apply on list[i..listSize-1]  
}
```

list[0] list[1] list[2] list[3] ... list[10]

list[0] list[1] list[2] list[3] ... list[10]

list[0] list[1] list[2] list[3] ... list[10]

list[0] list[1] list[2] list[3] ... list[10]

list[0] list[1] list[2] list[3] ... list[10]

...

list[0] list[1] list[2] list[3] ... list[10]



```
for (int i = 0; i < listSize; i++)  
{  
    select the smallest element in list[i..listSize-1];  
    swap the smallest with list[i], if necessary;  
    // list[i] is in its correct position.  
    // The next iteration apply on list[i..listSize-1]  
}
```

## Expand

```
double currentMin = list[i];  
[redacted]  
for (int j = i; j < list.length; j++) {  
    if (currentMin > list[j]) {  
        currentMin = list[j];  
        [redacted]  
    }  
}
```



```
for (int i = 0; i < listSize; i++)  
{  
    select the smallest element in list[i..listSize-1];  
    swap the smallest with list[i], if necessary;  
    // list[i] is in its correct position.  
    // The next iteration apply on list[i..listSize-1]  
}
```

## Expand

```
double currentMin = list[i];  
int currentMinIndex = i;  
for (int j = i; j < list.length; j++) {  
    if (currentMin > list[j]) {  
        currentMin = list[j];  
        currentMinIndex = j;  
    }  
}
```



```
for (int i = 0; i < listSize; i++)  
{  
    select the smallest element in list[i..listSize-1];  
    swap the smallest with list[i], if necessary;  
    // list[i] is in its correct position.  
    // The next iteration apply on list[i..listSize-1]  
}
```

## Expand

```
if (currentMinIndex != i) {  
    list[currentMinIndex] = list[i];  
    list[i] = currentMin;  
}
```



# Bir Yöntemle Gerçekleştirin

```
/** The method for sorting the numbers */
```

```
public static void selectionSort(double[] list) {  
    for (int i = 0; i < list.length; i++) {  
        // Find the minimum in the list[i..list.length-1]  
        double currentMin = list[i];  
        int currentMinIndex = i;  
        for (int j = i + 1; j < list.length; j++) {  
            if (currentMin > list[j]) {  
                currentMin = list[j];  
                currentMinIndex = j;  
            }  
        }  
        // Swap list[i] with list[currentMinIndex] if necessary;  
        if (currentMinIndex != i) {  
            list[currentMinIndex] = list[i];  
            list[i] = currentMin;  
        }  
    }  
}
```

Invoke it

selectionSort(yourList)

# The Arrays.sort Method

Sıralama, programlamada sıklıkla kullanıldığından, Java, `java.util.Arrays` sınıfında bir `int`, `double`, `char`, `short`, `long` ve `float` dizisini sıralamak için çeşitli overloaded (aşırı yüklenmiş) sıralama yöntemleri sunar. Örneğin, aşağıdaki kod bir dizi sayıyı ve bir karakter dizisini sıralar.

```
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);
```

