

BBS515 Nesneye Yönelik Programlama

Ders 5 - Diziler

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

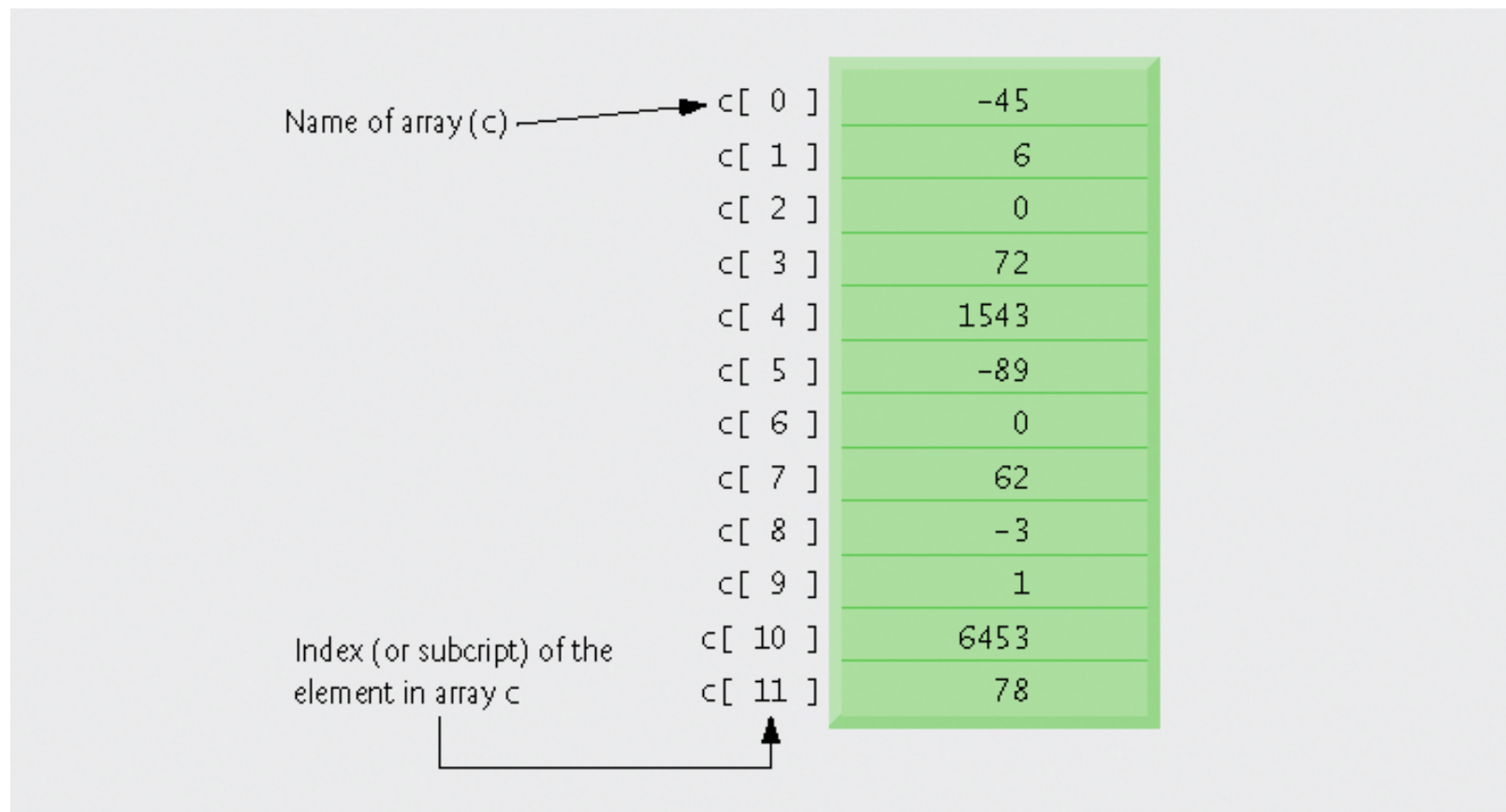
Diziler

Şimdiye kadar gördüğümüz int, char, boolean gibi tiplere sahip değişkenler, skaler değişkenlerdi. Yani, tek bir değer saklayabiliyorlardı.

- Aynı tipe sahip birden fazla değeri saklamak için diziler kullanılır.

Diziler

- Her dizinin bir adı olmalıdır. Aşağıdaki dizinin adı c'dir.
- Her dizinin bir uzunluğu vardır. Dizinin uzunluğu sahip olduğu eleman sayısıdır. Aşağıdaki dizinin uzunluğu 12'dir.
- Bir dizinin uzunluk değeri diziAdı.length ifadesiyle alınır. Örneğin aşağıdaki dizi için c.length ifadesinin değeri 12 olacaktır.



Diziler

- İndis:
 - Bir elemanın dizinin içindeki yerini belirten sayıdır
 - İndis negatif olmayan bir tamsayı olmalıdır.
 - Her dizinin ilk elemanının indisi 0'dır.

Name of array (c) →	c[0]	-45
	c[1]	6
	c[2]	0
	c[3]	72
	c[4]	1543
	c[5]	-89
	c[6]	0
	c[7]	62
	c[8]	-3
	c[9]	1
	c[10]	6453
Index (or subscript) of the element in array c →	c[11]	78

Diziler

- Bir dizinin her elemanı diziAdı[indis] ifadesi ile temsil edilir. Bu elemanlara indisli değişken denir.
- Örneğin aşağıdaki dizi için ilk eleman c[0], 5. eleman c[4] ile gösterilir.
- c[0] elemanının değeri -45, c[4] elemanının değeri 1543'tür.

Name of array (c) →	c[0]	-45
	c[1]	6
	c[2]	0
	c[3]	72
	c[4]	1543
	c[5]	-89
	c[6]	0
	c[7]	62
	c[8]	-3
	c[9]	1
	c[10]	6453
Index (or subscript) of the element in array c →	c[11]	78

Diziler

- İndisli değişkenlerle o tipteki normal değişkenlerle yaptığımız tüm işlemleri yapabiliriz.
- Örneğin:
 - `int x = c[3]` ifadesi x'in değerini 72 yapar(c dizisinin 4. elemanı)
 - `c[4] = c[7] + c[9]` ifadesi c[4]'ü yani c dizisinin 5. elemanının değerini 62+1 yani 63 yapar.

Name of array (c) →	c[0]	-45
	c[1]	6
	c[2]	0
	c[3]	72
	c[4]	1543
	c[5]	-89
	c[6]	0
	c[7]	62
	c[8]	-3
	c[9]	1
	c[10]	6453
Index (or subscript) of the element in array c →	c[11]	78

Dizi sınırları

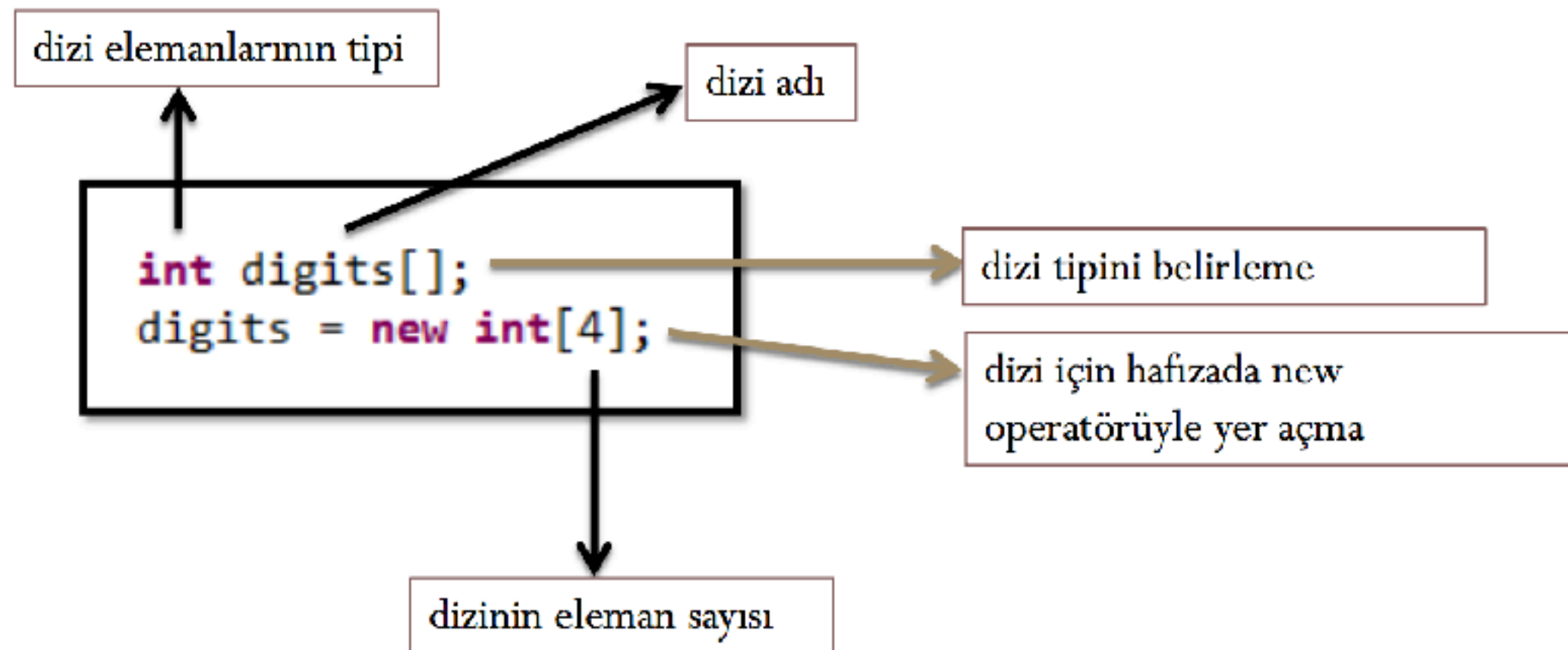
Dizi elemanlarına ulaşabileceğimiz indis sayısının dizinin uzunluğunun belirlediği bir sınır içinde yer alması gerekir. Yani n elemanlı bir dizinin elemanlarına 0'dan $n-1$ 'e kadar olan tamsayı değerlerle ulaşabiliriz. 0'dan küçük ve $n-1$ 'den büyük tamsayı değerleri bu diziye indeks olamazlar.

- - Program içinde bir dizinin elemanı sınırı aşan bir dizi indeksiyle çağırılırsa, program derleme zamanında değil çalışma zamanında hata verir.

Diziler

- Dizi oluşturmak:
 - Diziler `new` anahtar kelimesiyle oluşturulabilirler.
 - `c` isimli 12 `int` tipinde elemana sahip bir dizi aşağıdaki ifadeyle oluşturulur.
 - `int[] c = new int[12];`
- Bu ifade yerine aşağıdaki ifadeler de kullanılabilir.
 - `int[] c // diziyi deklare et`
 - `c = new int[12] //diziyi oluştur`

TEK BOYUTLU DİZİ TANIMLAMA



- `int digits[]` yerine `int[] digits` de yazılabilir.

Diziler

- Bir dizi new komutuyla ilk oluşturulduğunda dizinin elemanları aşağıdaki default(varsayılan) değerleri alırlar:
 - Nümerik primitif tipler için: 0
 - char tipi için: '\u0000'
 - boolean tipi için: false

Diziler: İlk değer vererek dizi oluşturmak

- Bir dizi, elemanlarının ilk değerlerinin bir listesiyle de oluşturulabilir:
- Örneğin elemanları [1,5] aralığındaki tamsayılar olan arr isimli bir dizi oluşturmak için:

```
int arr[ ] = {1,2,3,4,5};
```

ifadesi kullanılabilir.

Diziler

- İlk değerle dizi oluşturma ifadesi iki satır halinde yazılamaz. Örneğin aşağıdaki ifade bir derleyici hatasına sebep olur.

```
int arr[ ];  
arr = {1,2,3,4,5};
```

Diziler

```
int arr[ ] = {1,2,3,4,5};
```

ifadesi ile oluşturduğumuz arr dizisini başka biçimlerde de oluşturabiliriz.

Diziler

```
int arr[ ] = {1,2,3,4,5};
```

ifadesi ile oluşturduğumuz arr dizisini başka biçimlerde de oluşturabiliriz.

Aynı diziyi new komutuyla oluşturmak için:

```
int arr[ ] = new int[5];  
arr[0] = 1;  
arr[1] = 2;  
arr[2] = 3;  
arr[3] = 4;  
arr[4] = 5;
```

Diziler

```
int arr[ ] = {1,2,3,4,5};
```

ifadesi ile oluşturduğumuz arr dizisini başka biçimlerde de oluşturabiliriz.

Aynı diziyi new komutuyla oluşturmak için:

```
int arr[ ] = new int[5];  
arr[0] = 1;  
arr[1] = 2;  
arr[2] = 3;  
arr[3] = 4;  
arr[4] = 5;
```

veya daha kısa bir biçimde

```
int arr[ ] = new int[5];  
for(int i=0; i<5; i++){  
    arr[i] = i+1;  
}
```

Diziler: Kod takibi

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

4	0
3	0
2	0
1	0
0	0

*arr dizisi
oluşturulduğunda*

Diziler: Kod takibi

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

i	0
---	---

4	0
3	0
2	0
1	0
0	0

*arr dizisi
oluşturulduğunda*

Diziler: Kod takibi

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

i

0

4	0
3	0
2	0
1	0
0	1

***Döngünün ilk
adımında***

Diziler: Kod takibi

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

i	1
---	---

4	0
3	0
2	0
1	0
0	1

Diziler: Kod takibi

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

i

1

4	0
3	0
2	0
1	2
0	1

***Döngünün 2.
adımında***

Diziler: Kod takibi

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

i	2
---	---

4	0
3	0
2	0
1	2
0	1

Diziler: Kod takibi

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

i

2

4	0
3	0
2	3
1	2
0	1

**Döngünün 3.
adımında**

Diziler: Kod takibi

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

i
3

4	0
3	0
2	3
1	2
0	1

Diziler: Kod takibi

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

i

3

4	0
3	4
2	3
1	2
0	1

**Döngünün 4.
adımında**

Diziler: Kod takibi

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

i 4

4	0
3	4
2	3
1	2
0	1

Diziler: Kod takibi

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

i

4

4	5
3	4
2	3
1	2
0	1

***Döngünün 5.
adımında***

Diziler: Kod takibi

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

i	5
---	---

4	5
3	4
2	3
1	2
0	1

Diziler: Kod takibi

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

false : döngüden çık

i	5
4	5
3	4
2	3
1	2
0	1

Dizi elemanlarını yazdırma

```
► public class PrintLetterArray {  
    ► public static void main(String[] args){  
        char[] letterArray = {'p', 't', 'q', 's', 'z'};  
  
        int numOfLetters = letterArray.length;  
  
        for(int i=0; i<numOfLetters; i++){  
            System.out.println(letterArray[i]);  
        }  
    }  
}
```

çıktı

p
t
q
s
z

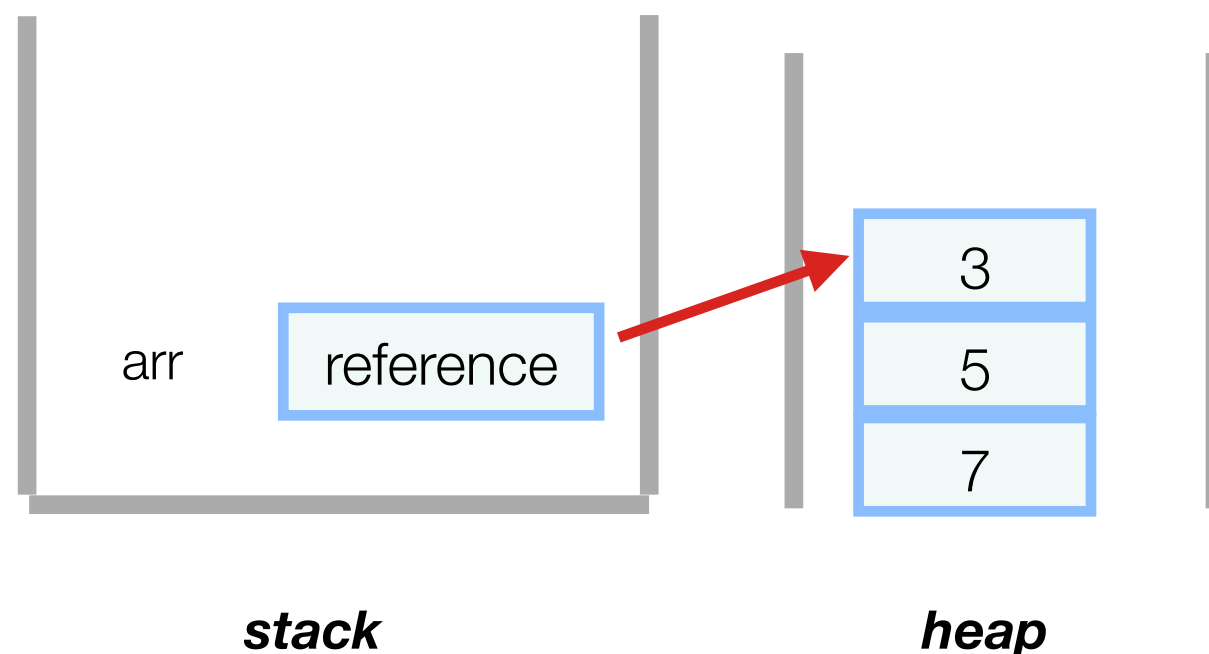
Örnek: Dizi elemanlarının toplamı

```
public class SumArray {  
    public static void main(String[] args){  
  
        double[] numberArray = {1.7, 2.3, 8.6, 9.0, 0.45, 3.9};  
  
        double sum = 0;  
  
        int arrayLength = numberArray.length;  
  
        for(int i=0; i<arrayLength; i++){  
            sum += numberArray[i];  
        }  
  
        System.out.printf("Sum of numbers in the array is: %.2f", sum);  
    }  
}
```

Dizilerin hafızada tutulması

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

```
arr[0] = 3;  
arr[1] = 5;  
arr[2] = 7;
```



Primitif tiplerden farklı olarak dizi elemanları heap adı verilen farklı bir hafıza bölümünde tutulur. Stack’de ise yalnızca dizi elemanlarının heapdeki adresini belirten bir referans tutulur.

Dizilerin kopyalanması

Çoğunlukla bir programda bir diziyi ya da dizinin bir parçasını kopyalamak gerekir. Bu işlem için primitif tiplerdeki gibi atama operatörünü (=) kullanmak tam anlamıyla bir kopyalama işi yapamaz.

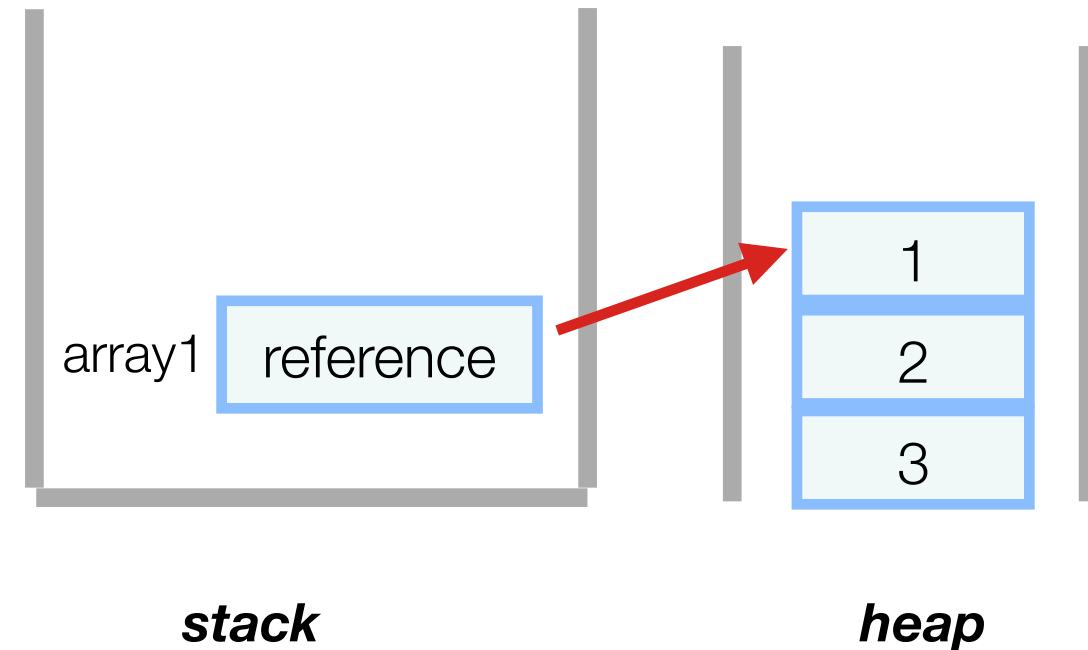
Dizilerin kopyalanması: TestCopyArray.java

```
public static void main(String[] args){  
  
    int[] array1 = {1, 2, 3};  
  
    int[] array2 = array1;  
  
    System.out.println("Before modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
  
    //Modify array 1  
    for(int i=0; i<array1.length; i++){  
        array1[i] = 0;  
    }  
  
    System.out.println("\n\nAfter modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
}
```

Örneğin array2 dizisini array1 dizisine = operatörüyle kopyalamak istediğimiz TestCopyArray programını inceleyelim.

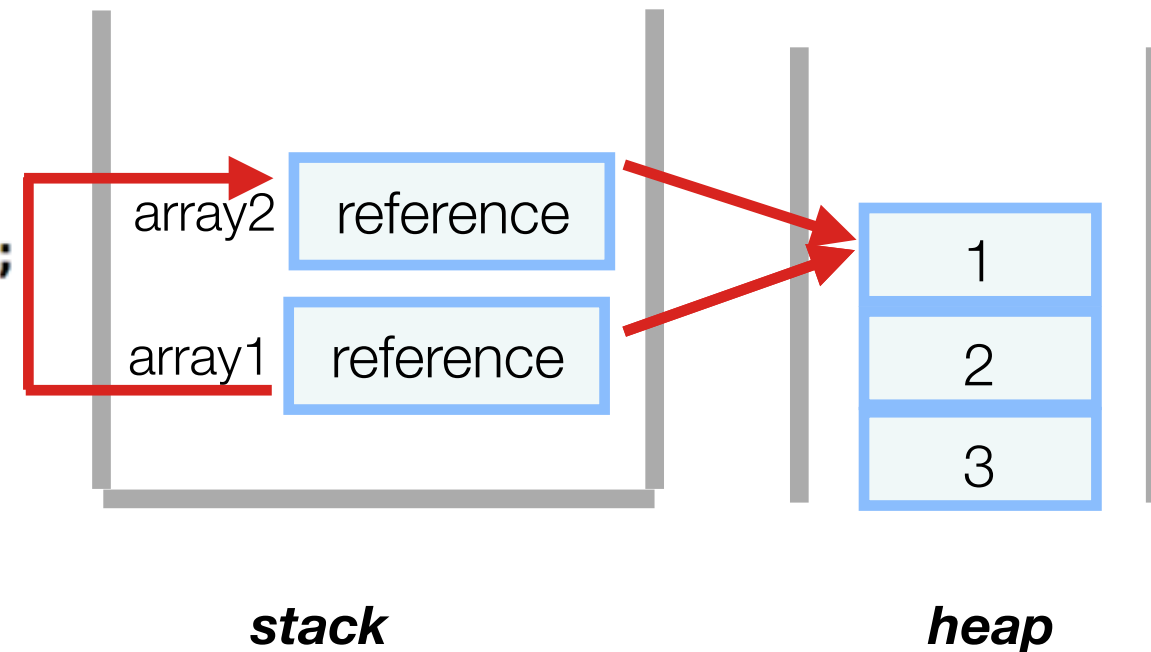
Dizilerin kopyalanması: TestCopyArray.java

```
public static void main(String[] args){  
    int[] array1 = {1, 2, 3};  
    int[] array2 = array1;  
  
    System.out.println("Before modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
  
    //Modify array 1  
    for(int i=0; i<array1.length; i++){  
        array1[i] = 0;  
    }  
  
    System.out.println("\n\nAfter modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
}
```



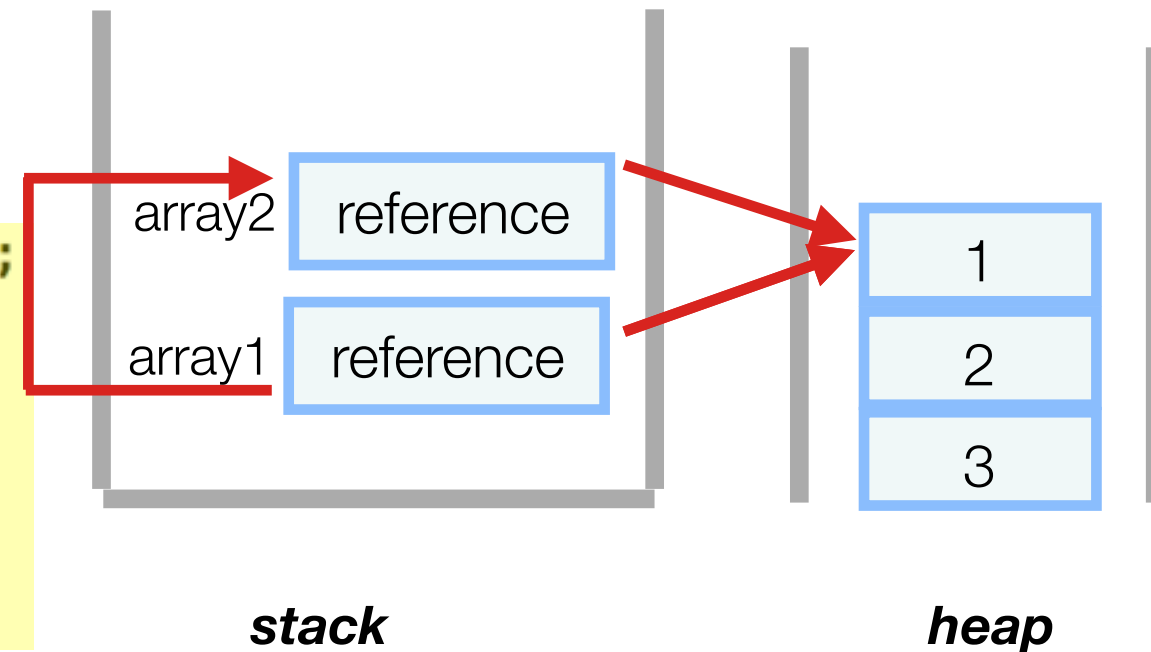
Dizilerin kopyalanması: TestCopyArray.java

```
public static void main(String[] args){  
  
    int[] array1 = {1, 2, 3};  
    int[] array2 = array1;  
  
    System.out.println("Before modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
  
    //Modify array 1  
    for(int i=0; i<array1.length; i++){  
        array1[i] = 0;  
    }  
  
    System.out.println("\n\nAfter modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
}
```



Dizilerin kopyalanması: TestCopyArray.java

```
public static void main(String[] args){  
  
    int[] array1 = {1, 2, 3};  
  
    int[] array2 = array1;  
  
    System.out.println("Before modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
  
    //Modify array 1  
    for(int i=0; i<array1.length; i++){  
        array1[i] = 0;  
    }  
  
    System.out.println("\n\nAfter modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
}
```

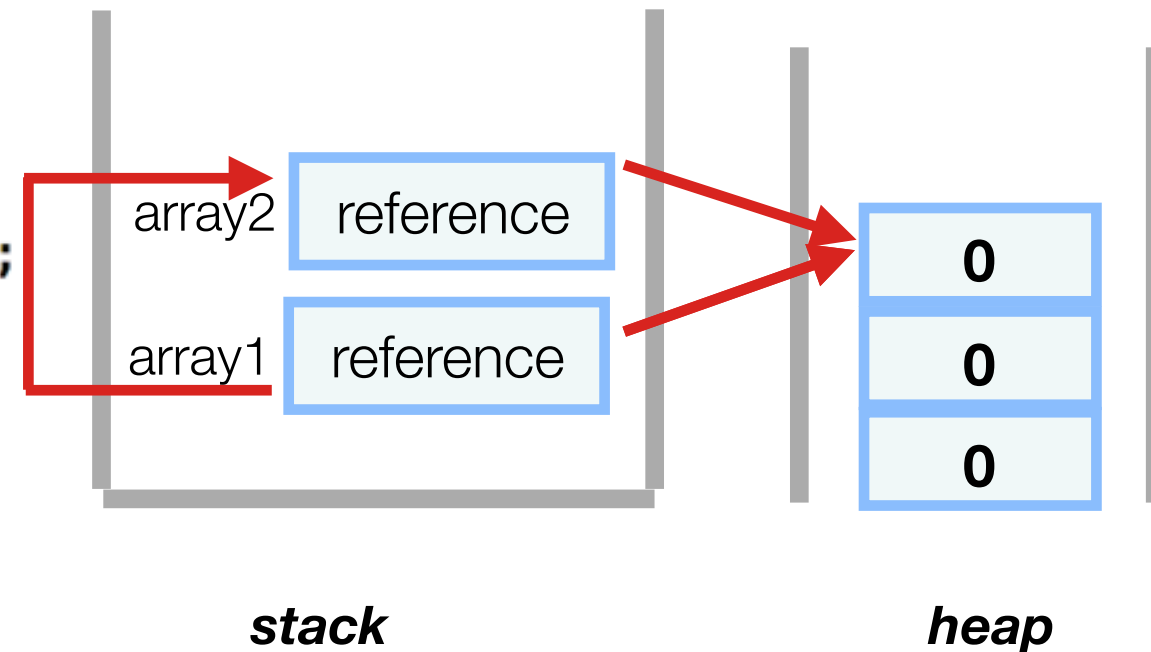


ÇIKTI

```
Before modifying array1:  
Elements of array1: 1 2 3  
Elements of array2: 1 2 3
```


Dizilerin kopyalanması: TestCopyArray.java

```
public static void main(String[] args){  
  
    int[] array1 = {1, 2, 3};  
  
    int[] array2 = array1;  
  
    System.out.println("Before modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
  
    //Modify array 1  
    for(int i=0; i<array1.length; i++){  
        array1[i] = 0;  
    }  
  
    System.out.println("\n\nAfter modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
}
```

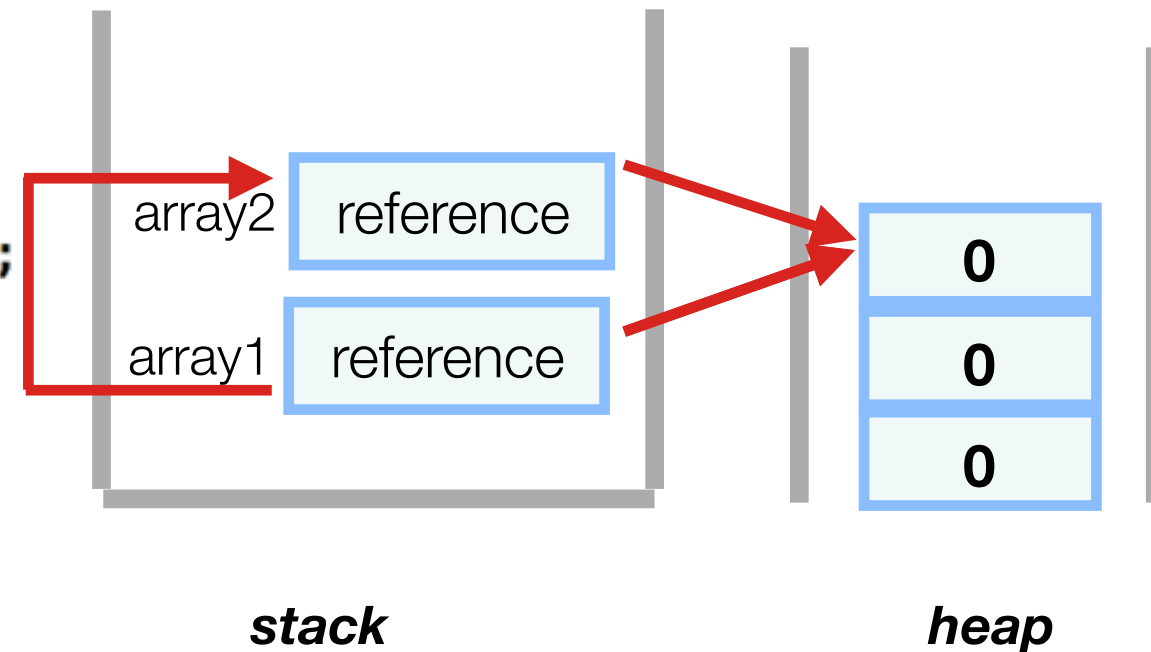


ÇIKTI

```
Before modifying array1:  
Elements of array1: 1 2 3  
Elements of array2: 1 2 3
```

Dizilerin kopyalanması: TestCopyArray.java

```
public static void main(String[] args){  
  
    int[] array1 = {1, 2, 3};  
  
    int[] array2 = array1;  
  
    System.out.println("Before modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
  
    //Modify array 1  
    for(int i=0; i<array1.length; i++){  
        array1[i] = 0;  
    }  
  
    System.out.println("\n\nAfter modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
}
```

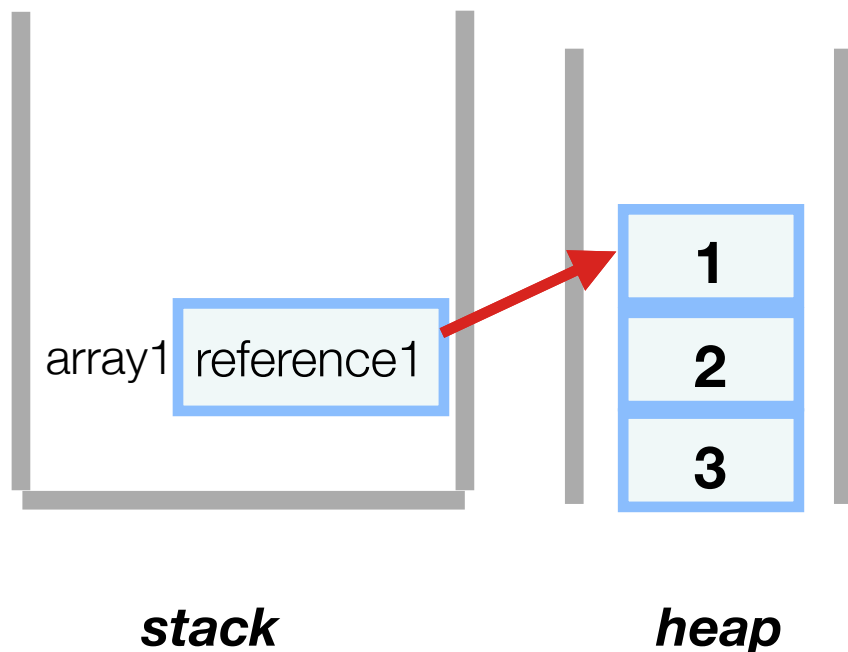


ÇIKTI

```
Before modifying array1:  
Elements of array1: 1 2 3  
Elements of array2: 1 2 3  
  
After modifying array1:  
Elements of array1: 0 0 0  
Elements of array2: 0 0 0
```

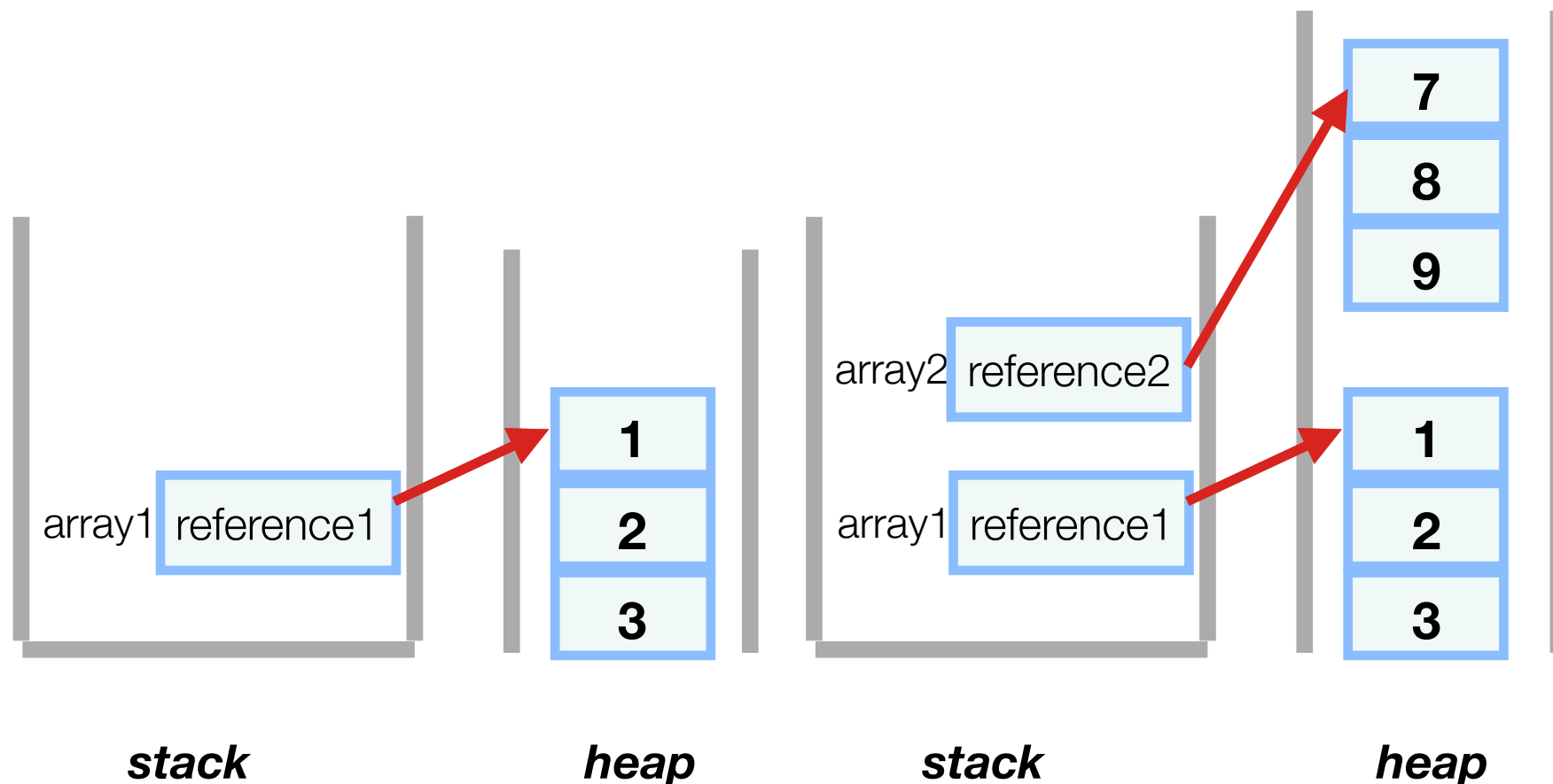
Dizilerin kopyalanması: TestCopyArray.java

```
int[] array1 = {1, 2, 3};
```



Dizilerin kopyalanması: TestCopyArray.java

```
int[] array1 = {1, 2, 3};    int[] array1 = {1, 2, 3};  
                             int[] array2 = {7, 8, 9};
```



Dizilerin kopyalanması: TestCopyArray.java

```
int[] array1 = {1, 2, 3};
```

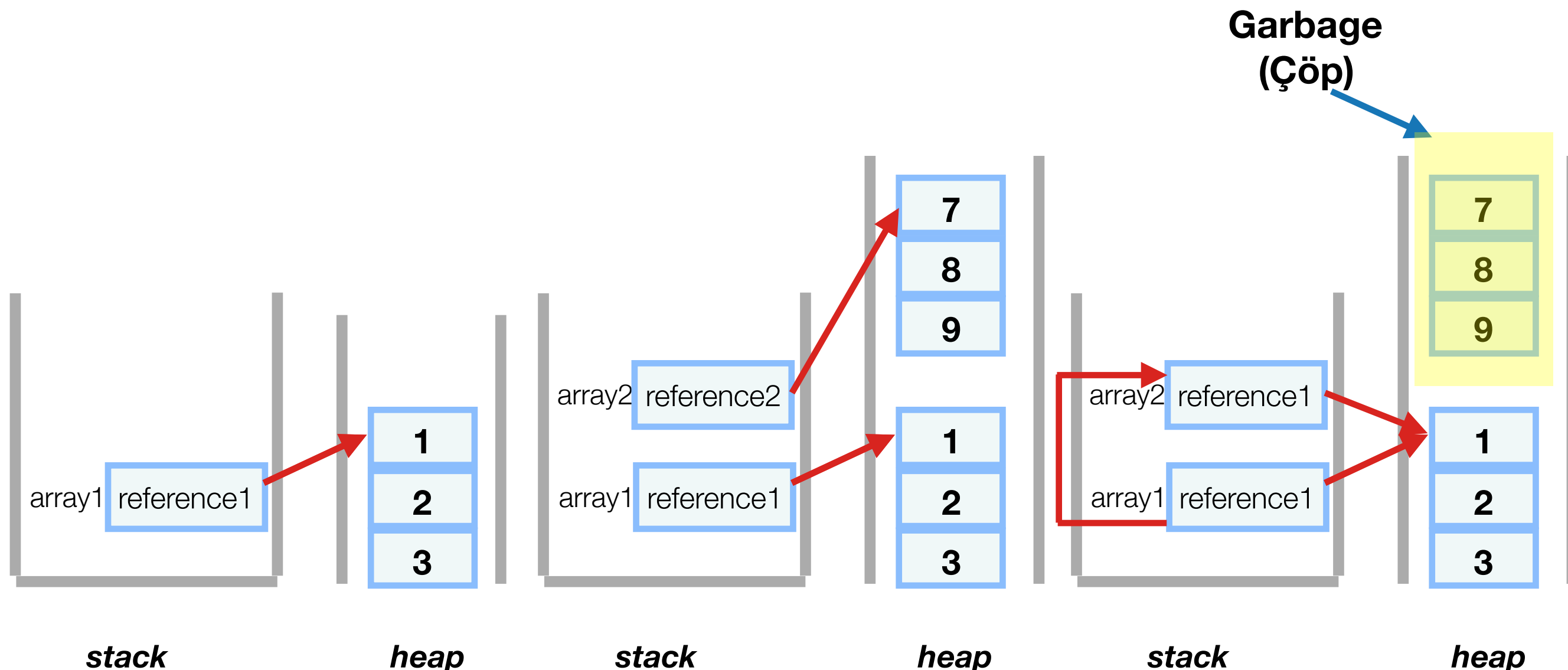
```
int[] array1 = {1, 2, 3};
```

```
int[] array1 = {1, 2, 3};
```

```
int[] array2 = {7, 8, 9};
```

```
int[] array2 = {7, 8, 9};
```

```
array2 = array1;
```



Dizilerin kopyalanması

Dizilerin referans değerlerinin değil de eleman değerlerinin kopyalanabilmesi için farklı yollar kullanabiliriz:

1. for döngüsünün kullanılması
2. arrayCopy yönteminin kullanılması

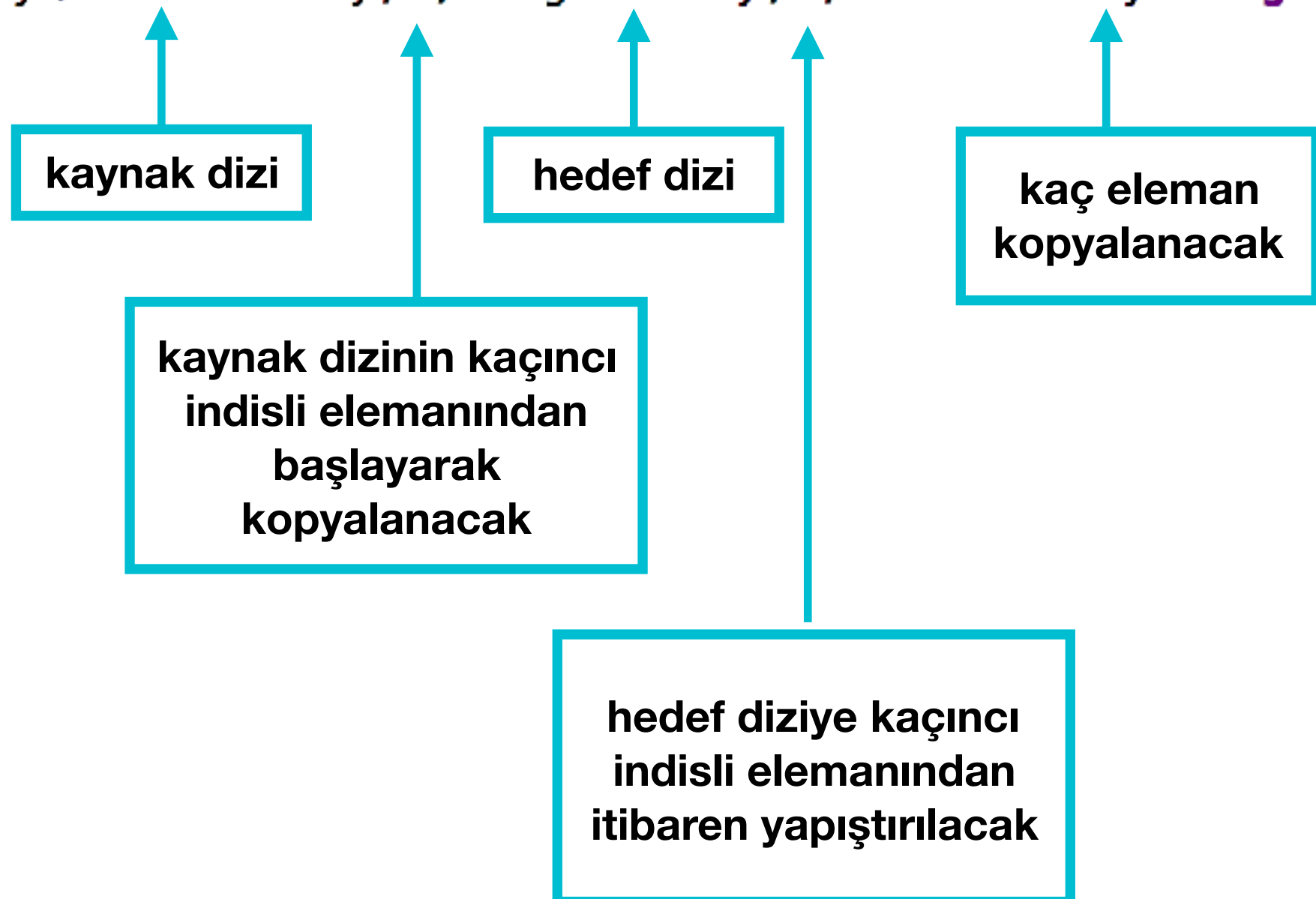
Dizilerin kopyalanması: for döngüsü

Kaynak dizideki her bir elemanın değerinin hedef dizideki aynı indisteki elemana for döngüsüyle aktarılmasıyla yapılır.

```
int[] sourceArray = {3, 5, 7, 9};  
int[] targetArray = new int[sourceArray.length];  
for(int i=0; i<sourceArray.length; i++)  
    targetArray[i] = sourceArray[i];
```

Dizilerin kopyalanması: arrayCopy yöntemi

```
int[] sourceArray = {3, 5, 7, 9};  
int[] targetArray = new int[sourceArray.length];  
System.arraycopy(sourceArray, 0, targetArray, 0, sourceArray.length);
```



Dizilerin kopyalanması: arrayCopy yöntemi

Örneğin arr1 dizisinin 1 indisli elemanından itibaren 5 elemanı arr2 dizisine 3 indisli elemanından itibaren kopyalanmak istiyorsa:

```
int[] arr1 = {1, 2, 3, 4, 5, 6, 7, 8, 9};  
int[] arr2 = {-1, -2, -3, -4, -5, -6, -7, -8, -9};
```

arr1

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

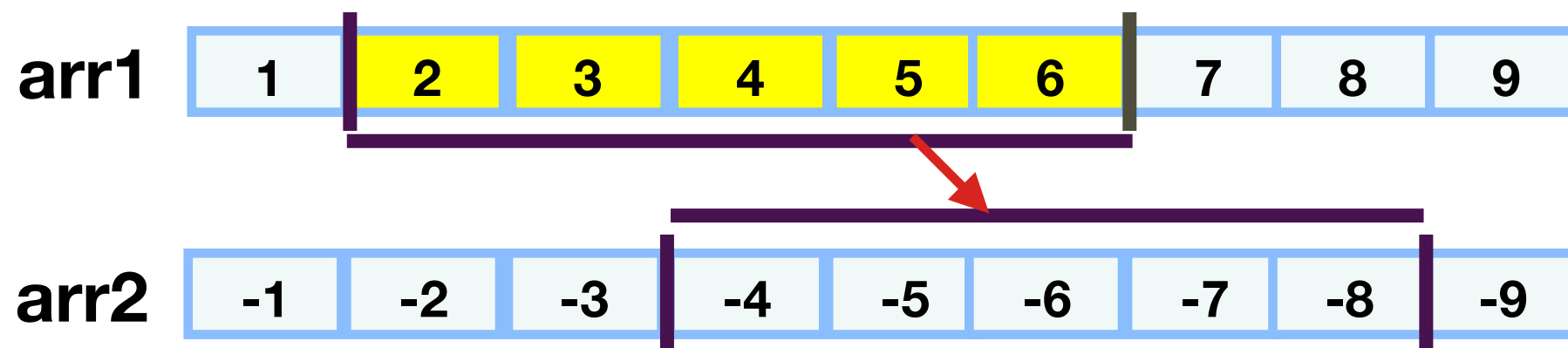
arr2

-1	-2	-3	-4	-5	-6	-7	-8	-9
----	----	----	----	----	----	----	----	----

Dizilerin kopyalanması: arrayCopy yöntemi

Örneğin arr1 dizisinin 1 indisli elemanından itibaren 5 elemanı arr2 dizisine 3 indisli elemanından itibaren kopyalanmak istiyorsa:

```
int[] arr1 = {1, 2, 3, 4, 5, 6, 7, 8, 9};  
int[] arr2 = {-1, -2, -3, -4, -5, -6, -7, -8, -9};  
System.arraycopy(arr1, 1, arr2, 3, 5);
```



Dizilerin kopyalanması: arrayCopy yöntemi

Örneğin arr1 dizisinin 1 indisli elemanından itibaren 5 elemanı arr2 dizisine 3 indisli elemanından itibaren kopyalanmak istiyorsa:

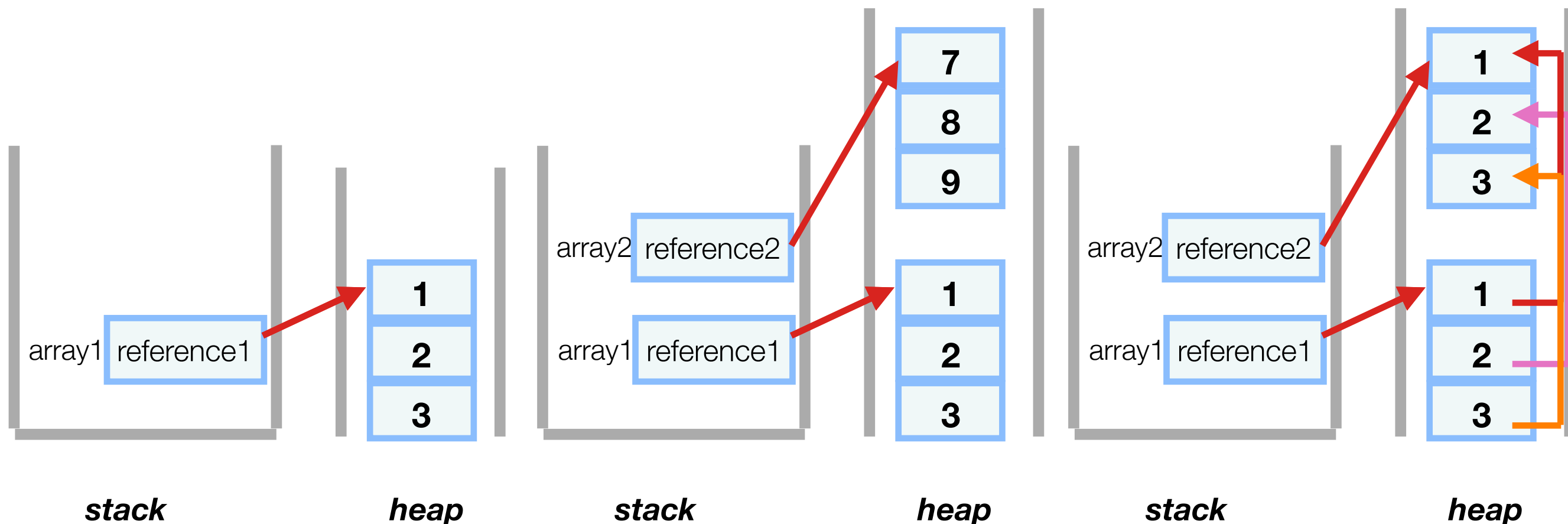
```
int[] arr1 = {1, 2, 3, 4, 5, 6, 7, 8, 9};  
int[] arr2 = {-1, -2, -3, -4, -5, -6, -7, -8, -9};  
System.arraycopy(arr1, 1, arr2, 3, 5);
```

Sonuç

arr1	1	2	3	4	5	6	7	8	9
arr2	-1	-2	-3	2	3	4	5	6	-9

Dizilerin kopyalanması: arrayCopy yöntemi

```
int[] array1 = {1, 2, 3}; int[] array1 = {1, 2, 3}; int[] array1 = {1, 2, 3};  
int[] array2 = {7, 8, 9}; int[] array2 = {7, 8, 9};  
System.arraycopy(array1, 0, array2, 0, array1.length)
```



Dizilerin kopyalanması: arrayCopy yöntemi

```
public static void main(String[] args){  
    int[] array1 = {1, 2, 3};  
    int[] array2 = new int[array1.length];  
  
    System.arraycopy(array1, 0, array2, 0, array1.length);  
  
    System.out.println("Before modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
  
    //Modify array 1  
    for(int i=0; i<array1.length; i++){  
        array1[i] = 0;  
    }  
  
    System.out.println("\n\nAfter modifying array1: ");  
  
    System.out.print("Elements of array1: ");  
    printArray(array1);  
  
    System.out.print("\nElements of array2: ");  
    printArray(array2);  
}
```

ÇIKTI

Before modifying array1:
Elements of array1: 1 2 3
Elements of array2: 1 2 3

After modifying array1:
Elements of array1: 0 0 0
Elements of array2: 1 2 3 |

Dizilerin metod parametresi olarak kullanılması

Bir yönteme parametre olarak verilen dizi, metodun tanımlanması sırasında, yine metod isminden sonra parantez içine tipi belirtilerek ve dizi parantezleri kullanılarak yazılır. Ancak parantezlerin içine dizinin boyu yazılmaz.

- Yöntem çağırılırken de dizi ismi parantezsiz olarak yazılır.

Dizilerin metod parametresi olarak kullanılması

```
public class PrintArrayWithMethod {  
    public static void printArray(int[] arr){  
        int n = arr.length;  
        for(int i=0; i<n; i++)  
            System.out.print(arr[i] + " ");  
    }  
  
    public static void main(String[] args){  
        int[] oddNumbers = {1,3,5,7,9};  
        printArray(oddNumbers);  
    }  
}
```

çıktı

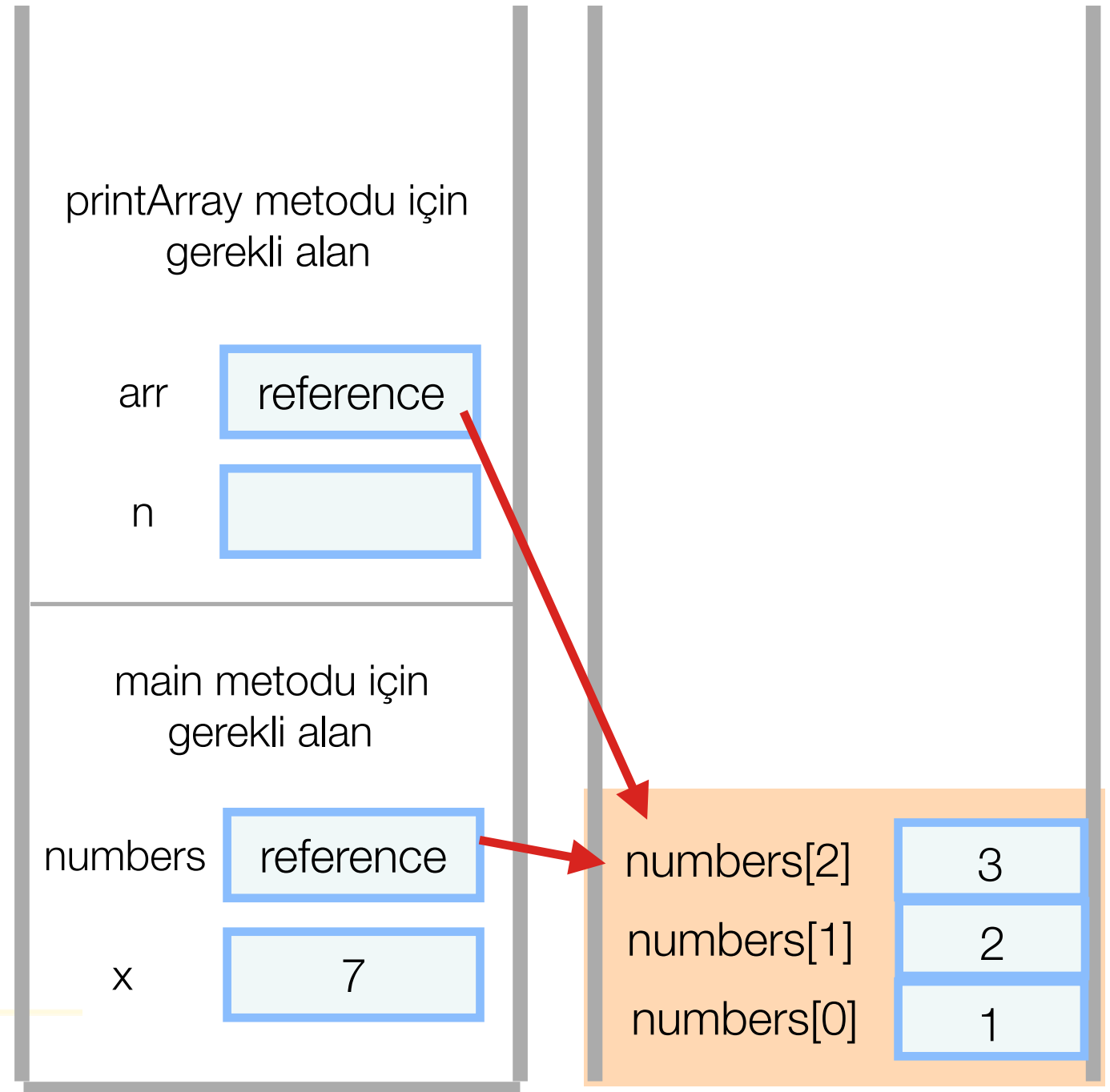
1 3 5 7 9

Dizilerin metod parametresi olarak kullanılması : Pass by reference

Diziler metotlara parametre olarak verilirken pass by reference yapılır. pass by value'nun tersine metotlar dizinin bir kopyası değil direkt kendisi üzerinde işlem yapar. Dolayısıyla parametre olarak verilen diziye metot içinde yapılan değişiklikler, dışarıya da yansır.

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo {  
  
    public static void printArray(int[] arr){  
  
        int n = arr.length;  
  
        for(int i=0; i<n; i++){  
            System.out.print(arr[i] + " ");  
        }  
  
        public static void main(String[] args){  
  
            int x = 7;  
  
            int[] numbers = {1,2,3};  
  
            System.out.println("x: " + x);  
            System.out.print("Numbers: ");  
            printArray(numbers);  
        }  
    }  
}
```



stack

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public static void changeValues(int[] arr, int x ){  
    x += 10;  
  
    int n = arr.length;  
  
    for(int i=0; i<n; i++){  
        arr[i] += 10;  
    }  
}
```

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public static void main(String[] args){  
    int x = 7;  
  
    int[] numberArray = {1, 2, 3};  
  
    System.out.println("Before calling changeValues method: ");  
  
    System.out.println("x: "+x);  
  
    System.out.print("Number Array: ");  
    for(int i=0; i<numberArray.length; i++){  
        System.out.print(numberArray[i]+" ");  
    }  
  
    changeValues(numberArray,x);  
  
    System.out.println("\n\nAfter calling changeValues method: ");  
  
    System.out.println("x: "+x);  
  
    System.out.print("Number Array: ");  
    for(int i=0; i<numberArray.length; i++){  
        System.out.print(numberArray[i]+" ");  
    }  
}
```

Dizilerin metod parametresi olarak kullanılması : Pass by reference

changeValues metodu çağırıldıktan sonra primitif tipli argümanın değeri değişmezken, dizi argümanın değeri değişir.

Before calling changeValues method:

x: 7

Number Array: 1 2 3

After calling changeValues method:

x: 7

Number Array: 11 12 13

Primitif tiplerden farklı olarak diziler heap adı verilen farklı bir hafıza bölümünde tutulur. Stack'de ise dizinin heapdeki adresini belirten bir referans tutulur.

Dizilerin metod parametresi olarak kullanılması : Pass by reference

changeValues metodu numberArray dizisiyle çağırıldığında, numberArray dizisinin referans değeri metod parametresi arr'ye aktarılır. Dolayısıyla hem numberArray hem de arr heap'te tutulan aynı değerlere işaret eder. Bu durumda arr'nin elemanlarında değişiklik yapmakla numberArray'in elemanlarında değişiklik yapmak aynıdır.

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {  
    public static void changeValues(int[] arr, int x ){  
        x += 10;  
        int n = arr.length;  
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }  
    }  
  
    public static void main(String[] args){  
        int x = 7;  
        int[] numberArray = {1, 2, 3};  
        changeValues(numberArray,x);  
    }  
}
```

main metodu için
gerekli alan

x

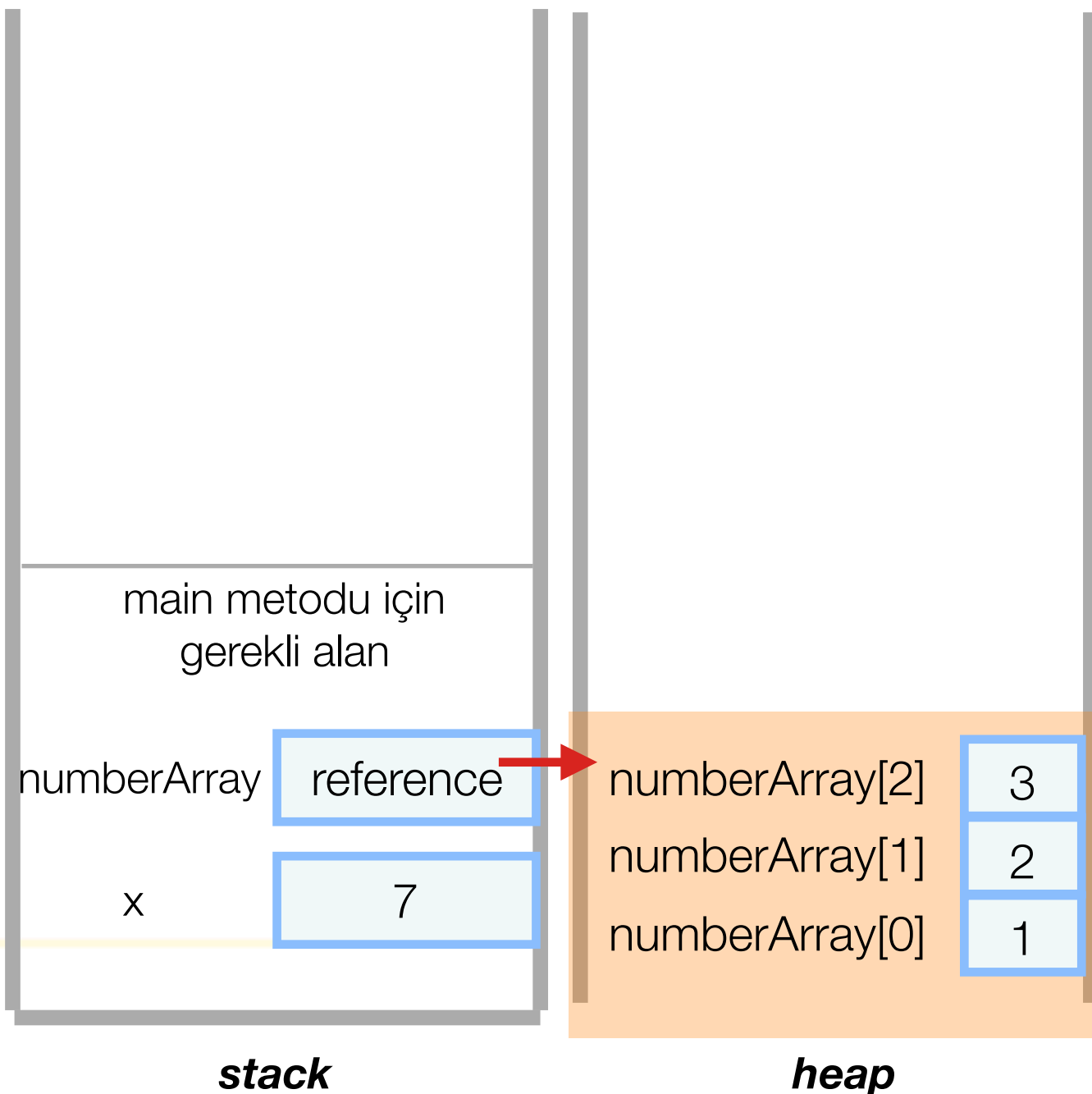
7

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {  
    public static void changeValues(int[] arr, int x ){  
        x += 10;  
        int n = arr.length;  
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }  
    }  
  
    public static void main(String[] args){  
        int x = 7;  
        int[] numberArray = {1, 2, 3};  
        changeValues(numberArray,x);  
    }  
}
```



Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

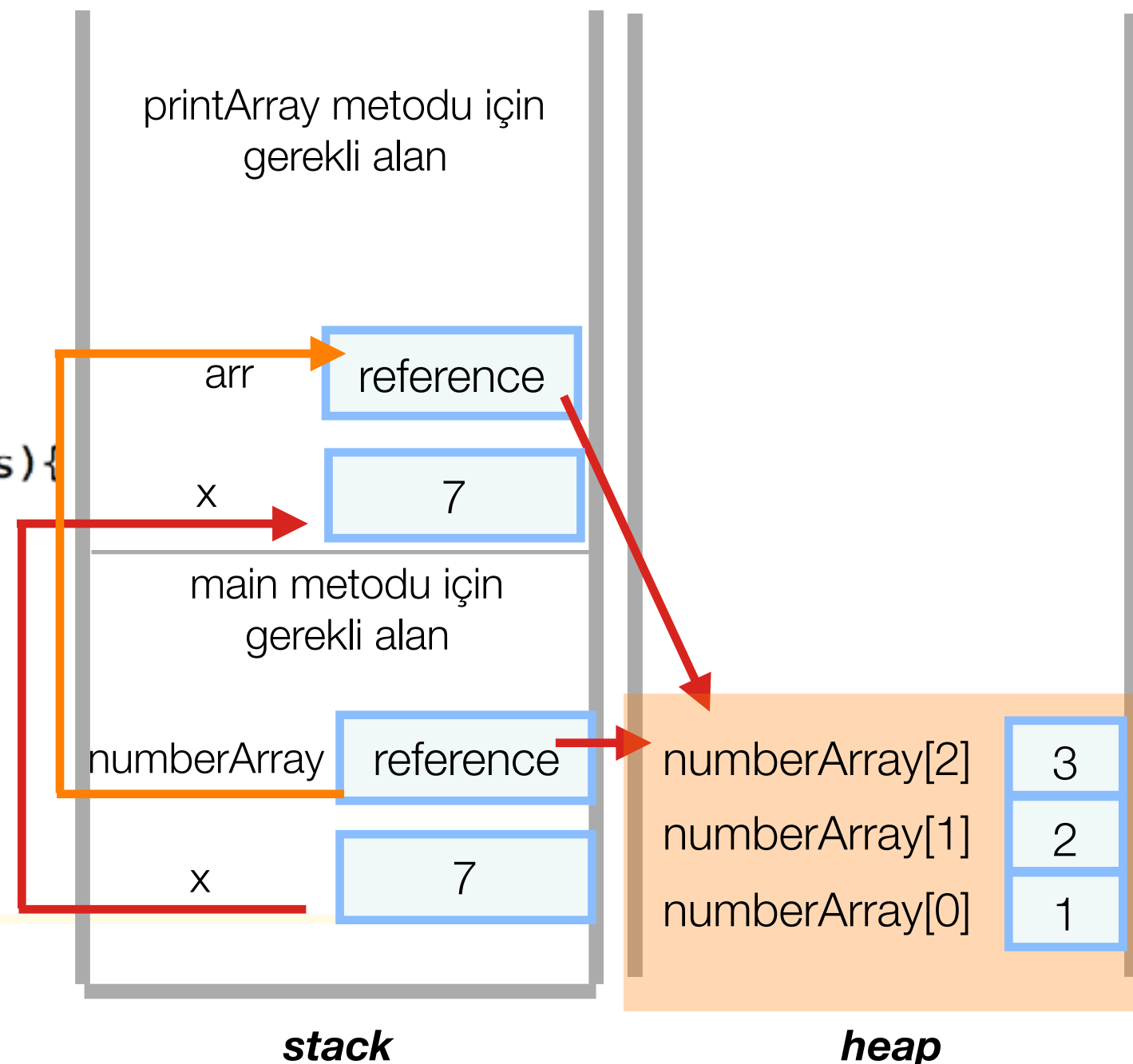
```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray, x);
```

```
    }
```



Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    }
```

```
    public static void main(String[] args){
```

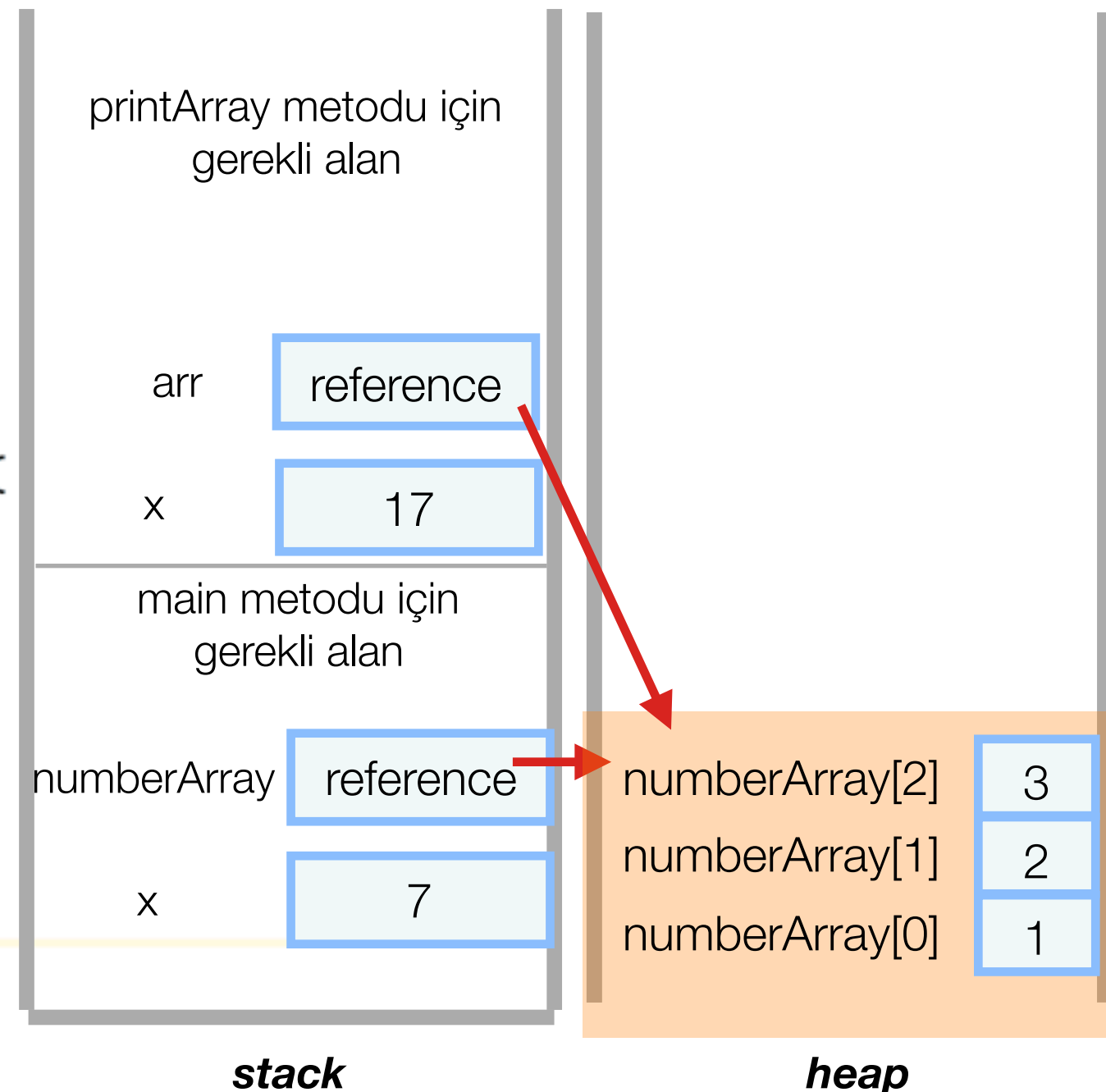
```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

```
}
```



Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

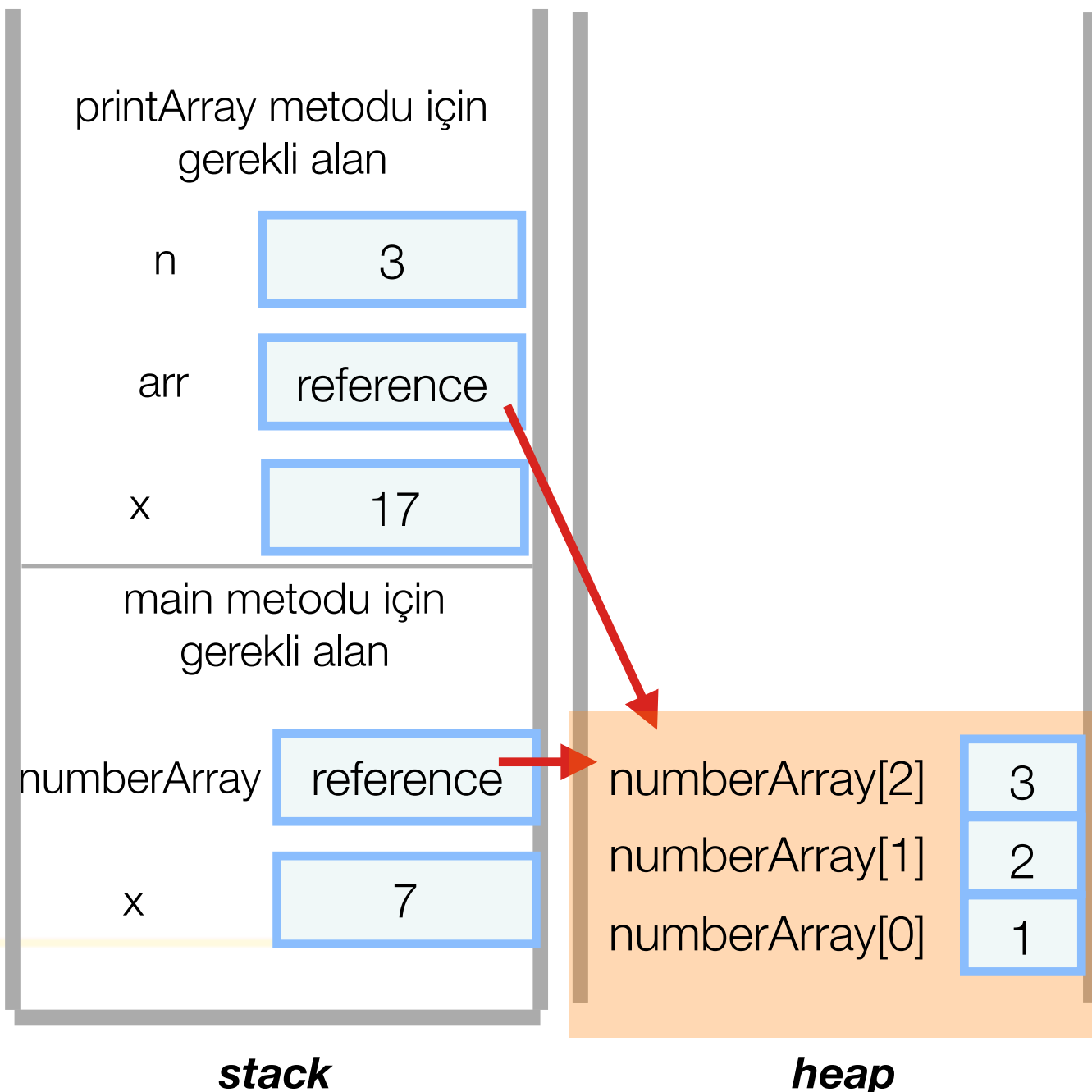
```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

```
}
```



Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i 0

n 3

arr reference

x 17

main metodu için
gerekli alan

numberArray reference

x 7

numberArray[2] 3
numberArray[1] 2
numberArray[0] 1

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

0

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

3

numberArray[1]

2

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

1

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

3

numberArray[1]

2

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

1

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

3

numberArray[1]

12

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

2

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

3

numberArray[1]

12

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);  
    }
```

printArray metodu için
gerekli alan

i

2

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

13

numberArray[1]

12

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

printArray metodu için
gerekli alan

i

3

n

3

arr

reference

x

17

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

13

numberArray[1]

12

numberArray[0]

11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){
            arr[i] += 10;
        }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray, x);
```

```
    }
```

false : döngüden çık

printArray metodu için
gerekli alan

i 3

n 3

arr reference

x 17

main metodu için
gerekli alan

numberArray reference

x 7

numberArray[2] 13
numberArray[1] 12
numberArray[0] 11

stack

heap

Dizilerin metod parametresi olarak kullanılması : Pass by reference

```
public class PassByReferenceDemo2 {
```

```
    public static void changeValues(int[] arr, int x ){
```

```
        x += 10;
```

```
        int n = arr.length;
```

```
        for(int i=0; i<n; i++){  
            arr[i] += 10;  
        }
```

```
    }
```

```
    public static void main(String[] args){
```

```
        int x = 7;
```

```
        int[] numberArray = {1, 2, 3};
```

```
        changeValues(numberArray,x);
```

```
    }
```

**changeValues metodundan
çıkıldıktan sonra arr referansı
hafızadan silindi ama heap'deki
değişmiş dizi değerleri olduğu gibi
kaldı**

main metodu için
gerekli alan

numberArray

reference

x

7

numberArray[2]

13

numberArray[1]

12

numberArray[0]

11

stack

heap

Metod dönüş tipi olarak diziler

- Bir dizi bir metodun dönüş tipi olabilir.
- Örneğin aşağıdaki metod, parametre olarak aldığı pozitif tamsayı kadar rasgele sayılar üretir ve bunları bir dizi halinde döndürür.

```
public static double[] generateRandomNumbers(int n){  
    double[] randomNumbers = new double[n];  
  
    for(int i=0; i<n; i++){  
        randomNumbers[i] = Math.random();  
    }  
  
    return randomNumbers;  
}
```


Metod dönüş tipi olarak diziler:

RandomNumbers.java

- generateRandomNumbers yöntemi kullanılarak kullanıcının istediği adette rasgele ondalık sayılar ekrana yazdıran aşağıdaki program yazılabilir.

```
public static void main(String[] args){  
    int n;  
  
    System.out.print("How many numbers do you want to generate?: ");  
  
    Scanner input = new Scanner(System.in);  
  
    n = input.nextInt();  
  
    System.out.println("Generated numbers are: ");  
    printArray(generateRandomNumbers(n));  
}
```

Geliştirilmiş for döngüsü (for each)

JDK 1.5 ile birlikte bir diziyi indis değişkeni kullanmadan dolaşmayı sağlayan yeni bir for döngüsü yapısı tanımlandı.

Örneğin bir arr dizisinin tüm elemanlarını yazdırmak istediğimizde

```
for(int i=0; i < arr.length; i++){  
    System.out.println(arr[i]);  
}
```

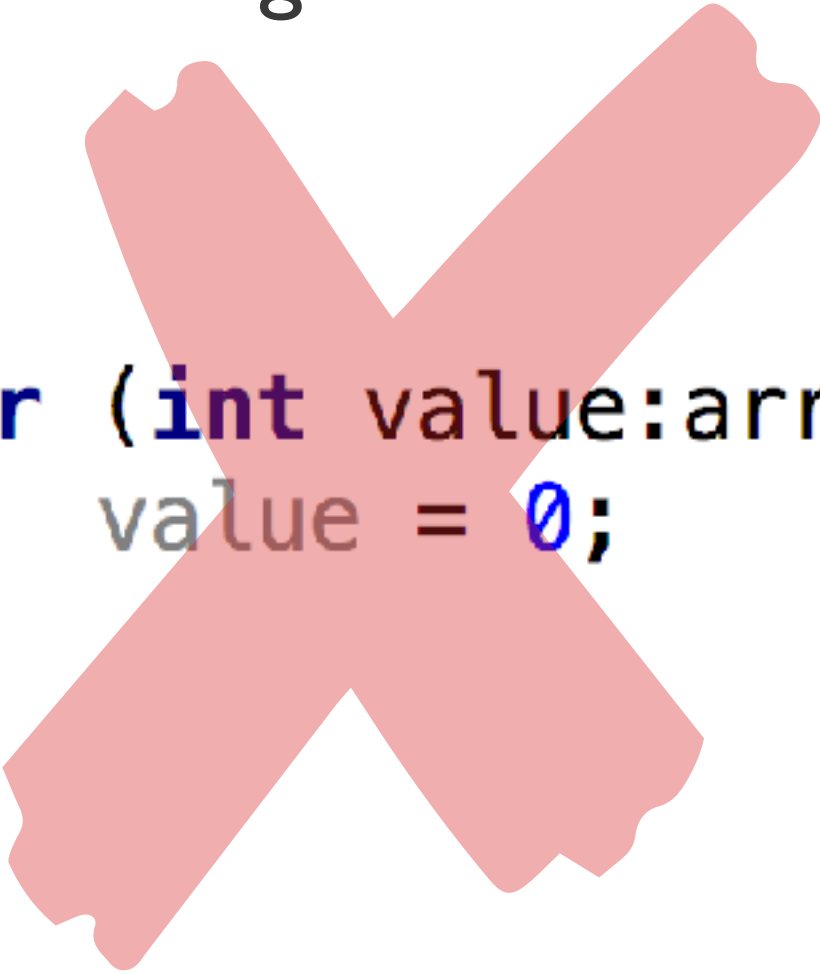
yerine kısaca

```
for (int value:arr) {  
    System.out.println(value);  
}
```

ifadesini kullanabiliriz.

Geliştirilmiş for döngüsü (for each)

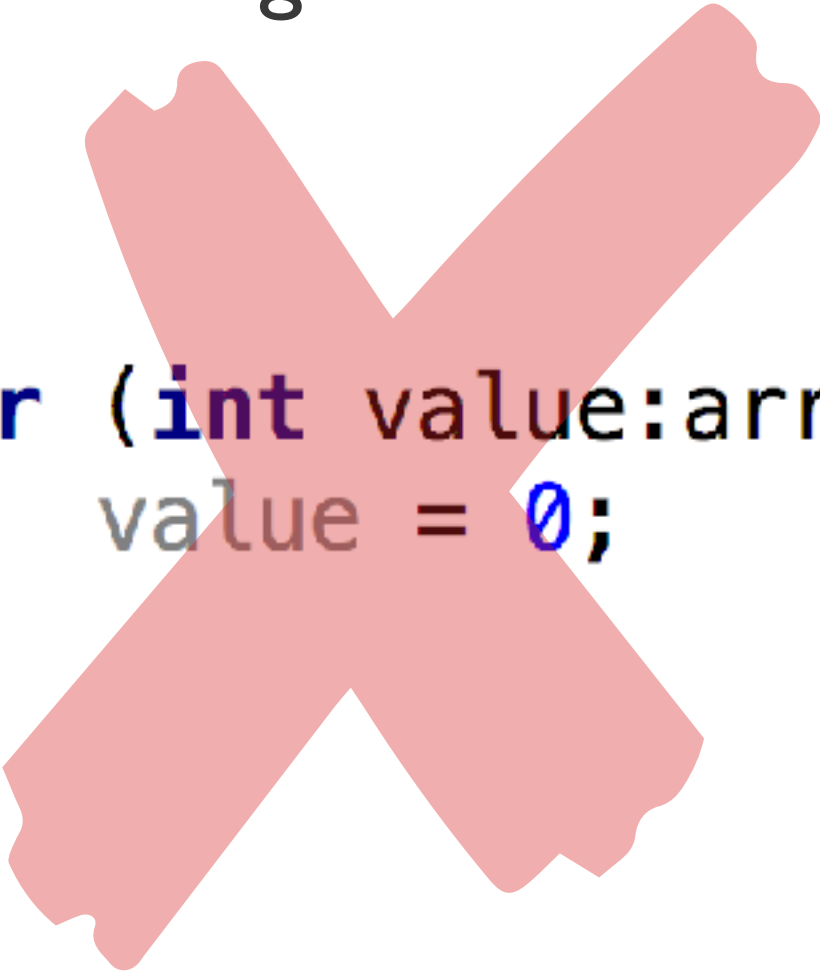
Gelişmiş for döngüsü dizi elemanlarını değiştirmek için kullanılamaz, yalnızca onlara erişmek için kullanılır. Örneğin dizi elemanlarının hepsinin değerini 0 yapmak için aşağıdaki gibi foreach döngüsü kullanmak hatalıdır.



```
for (int value:arr) {  
    value = 0;  
}
```

Geliştirilmiş for döngüsü (for each)

Gelişmiş for döngüsü dizi elemanlarını değiştirmek için kullanılamaz, yalnızca onlara erişmek için kullanılır. Örneğin dizi elemanlarının hepsinin değerini 0 yapmak için aşağıdaki gibi foreach döngüsü kullanmak hatalıdır.



```
for (int value:arr) {  
    value = 0;  
}
```

Dizi elemanlarının yerini değiştirme (Swapping)

```
public static void swapElements(int[] arr1, int index1, int index2){  
    int temp = arr1[index1];  
    arr1[index1] = arr1[index2];  
    arr1[index2] = temp;  
}
```

Dizi elemanlarının yerini değiştirme (Swapping)

SwapElements metodunu kullanarak bir dizinin 1.ve 4. elemanlarının yerini değiştiren program örneği:

```
public static void main(String args[]){  
    int[] intArray = {1, 3, 5, 7, 9};  
  
    System.out.println("Array before swapping");  
    printArray(intArray);  
  
    swapElements(intArray, 0, 3);  
  
    System.out.println("Array after swapping");  
    printArray(intArray);  
}
```

Çıktı

```
Array before swapping  
1 3 5 7 9  
Array after swapping  
7 3 5 1 9
```

Örnek 1: ShiftElements.java

Verilen bir dizinin elemanlarının yerlerini 1 adım geriye doğru kaydıran bir Java programı yazınız.

Örnek 2: GuessThePassword.java

Bir kasanın 5 tamsayı değerden oluşan şifresini rasgele karıştırarak ekrana yazdıran ve kullanıcıdan şifreyi tahmin etmesini isteyen bir Java programı yazınız.